UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

TESIS

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONSULTA SOBRE MODIFICACIONES POSTRADUCCIONALES EN LAS PROTEÍNAS DE LA LEVADURA SACCHAROMYCES CEREVISIAE

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

PRESENTA:

LEDESMA DOMÍNGUEZ LEONARDO

DIRECTOR DE TESIS

DR. JOSÉ FRANCISCO TORRES QUIROZ

ÁREA DE ESPECIALIDAD: INGENIERÍA BIOMÉDICA

CIUDAD UNIVERSITARIA 12 /JUNIO/ 2015.

Agradecimientos

En primer lugar quiero agradecer a Dios, porque me ha permitido terminar una etapa más en mi vida y me ha dado las fuerzas para cumplir cada una de las metas que me he propuesto. Por darme una visión diferente de vivir, de comportarme y de pensar.

A mis padres: Rogelio Ledesma González e Irene Domínguez Reyes que en la medida de lo posible siempre me han apoyado y han dado el 100% de los recursos económicos para yo poder seguir estudiando, pero más que eso, por el amor que me han brindado y por la educación tan valiosa que me han regalado.

A mi hermana Abigail, que con su amor y su ternura, con su propia vida me ha demostrado que todo es posible.

A mi novia Diana Barrera Hernández quien ha estado conmigo en los momentos más difíciles de mi carrera, que me ha apoyado siempre en lo que he necesitado y por recibirme todos los días con una sonrisa en su rostro.

A mi amigo y profesor el Ing. Rafael Sandoval que gracias su ejemplo, me ha demostrado que puedo creer en mí mismo para hacer todo lo que yo me proponga y sobre todo nunca darme por vencido. A mi director de tesis el Dr. Francisco Torres Quiroz que me ha apoyado con su conocimiento y su disposición para realizar este proyecto.

A mi amigo pero más hermano, Eduardo Sandoval, que ha sido para mí, el hermano que nunca tuve, que durante más de siete años lo he conocido, y que me ha apoyado en el trascurso de este proyecto. También a mi amigo David Rosado que con su ejemplo, me ha demostrado que no hay límites para realizar lo que uno se propone y porque me ha ensañado que con perseverancia y práctica se pueden alcanzar las metas.

A toda la Unidad de Servicios de Cómputo Académico (UNICA) por enseñarme los principios y valores que un profesionista tiene que tener y por regalarme todo el conjunto de conocimientos que ahora pongo por obra en mí trabajo.

A mi jefe de trabajo, Gerardo Coello, por haber creído en mí y por darme la grandiosa oportunidad de ser parte de la Unidad de Cómputo del Instituto de Fisiología Celular. Y en general a la Unidad de Cómputo por apoyarme en darme consejos y enseñarme cosas nuevas: Juan, Ivette, Pablo, Rodrigo y Ana.

A mis amigos de la carrera, con los cúales he compartido cada uno de los momentos de estos 4 años y medio. A mis profesores de la carrera que me han trasmitido sus conocimientos para poder desarrollar mi pensamiento ingenieril, a mi tutor el Ing. César Vázquez.

A mis amigos y compañeros de carrera Luis Enrique González, Yoás S. Ramírez y Juan Alfredo Núñez quienes han estado conmigo en muchos proyectos y que de ellos he aprendido que el trabajo en equipo es un elemento fundamental en la vida profesional y que no hay que dar lo necesario, sino más allá.

A los Cursos Facultad de Ingeniería, por darme la grande oportunidad de desarrollarme como docente, pero más que eso como persona; al impartir 15 cursos de matemáticas y física, y a ensañarme que las cosas se hacen por una única razón; por agradecimiento.

A mi profesora la Dra. Karina Mendoza, que ha sido una excelente profesora y una de las mejores que he tenido; porque gracias a su conocimiento que me enseñó del área de ciencias biomédicas, tome el interés en este tema de tesis y dedicarme en futuro a esta rama de la Ciencia.

A la Universidad Nacional Autónoma de México y a la Facultad de Ingeniería por que han hecho de mí lo que ahora soy.

Contenido

2. Marco Teorico.

INTRODUCCIÓN

CAPÍTULO I

En el área de las ciencias biomédicas el conjunto de conocimientos biológicos, químicos, estadísticos, computacionales y por supuesto médicos son la base para la construcción de una serie de teorías e hipótesis, que impactarán directamente en lo que hasta ahora se conoce en la vida del ser humano y su entorno, desde lo que consume en su alimentación, hasta en un estado posible de enfermedad.

Para que se logren los avances en las ciencias biomédicas es necesario el uso de tecnologías de la información en el área de la Ingeniería en Computación, específicamente, de las bases y redes de datos para el almacenamiento, organización, compartimiento, gestión de la información así como el tratamiento de la misma para proporcionar una interpretación de los datos generados por los expertos en esta área.

Hablando específicamente del tratamiento de la información, el punto central de esta tesis son las modificaciones postraduccionales. Este proceso biológico se lleva a cabo como siguiente etapa de la traducción de lenguaje génico a lenguaje proteico y tiene una importante relevancia debido a que éstas pueden regular procesos de señalización celular, mantener mecanismos de control, que son los responsables de mantener la homeostasis en todos los seres vivos.

Debido a que las investigaciones directamente en seres humanos tienden a ser un tema de discusión, se puede recurrir a la fisiología comparada, es decir realizar investigaciones en organismos que presenten similitudes fisiológicas.

La levadura *Saccharomyces cerevisiae* es un eucariote unicelular que tiene procesos fisiológicos parecidos a los de los seres humanos. Además posée características que permiten que su estudio sea relativamente más sencillo que un sistema tan complejo como el ser humano, esto haciendo referencia en los siguientes puntos:

- El genoma de la levadura es más reducido que un organismo complejo como en el caso del ser humano, en cantidad el genoma de la levadura es de 12 millones de pares de bases con cerca de 6000 genes que actualmente casi se conoce la función de cada uno de ellos esto quiere decir que es 200 veces más pequeño que el genoma del ser humano.
- La levadura presenta los dos ciclos de vida de reproducción asexual: haploide y diploide, e incluso en determinadas condiciones puede existir reproducción sexual.
- Su rápido crecimiento, reproducción, asilamiento y mutación permite hacer estudios directos a un objetivo en particular y hacer un análisis casi inmediato.
- Al ser un organismo unicelular se puede llevar a cabo estudios sobre sus procesos celulares sin gastar demasiados recursos económicos como si se haría en un organismo multicelular.

- Se estima que el conjunto de 16 cromosomas de la *Saccharomyces cerevisiae* con tamaños de alrededor de centenas hasta el orden de kilobases, comparten aproximadamente el 23% del genoma humano.

Con estas y otras características que se explicarán más adelante con mayor detalle hacen de la levadura *Saccharomyces cerevisiae* un excelente organismo para hacer investigación. Los datos obtenidos en los experimentos en modificaciones postraduccionales realizados son numerosos y se han realizado esfuerzos para construir bases de datos para almacenar la máxima información posible con la que se cuenta hasta la actualidad, basadas en experimentaciones de investigadores interesados en el área.

Las modificaciones postraduccionales que comúnmente se estudian son:

- Fosforilación.
- Metilación.
- Succinilación.
- Ubiquitinación.
- Acetilación.
- Glicosilación.
- Lipidación.
- Oxidación.

Entre otras más importantes que se explicarán a lo largo del desarrollo de los temas. De estas modificaciones se tiene documentado: la posición dentro de la proteína donde ocurre la modificación, el residuo que es afectado, el conjunto de péptidos asociados al lugar de la modificación, el artículo o publicación donde se extrae dicha información y otros datos relevantes que dependen de la modificación en cuestión. Esta información después podrá ser llevada a análisis estadísticos y probabilísticos para tener un mejor panorama del comportamiento celular de la levadura.

En los últimos años ha surgido el término de “Bioinformática” debido al gran auge del uso de herramientas de cómputo para el análisis de los cientos y miles incluso millones de datos que se generan al realizar investigaciones y estudios sobre las ramas de la genética, biológica molecular, bioquímica, medicina y todas las ciencias y disciplinas que involucren manejo de datos biológicos. Aunque falta mucho por hacer se han realizado grandes avances, a lo largo del desarrollo de éste trabajo se podrá ver el beneficio que puede otorgar.

Actualmente las bases de datos son la columna vertebral de muchos sistemas de búsquedas y consultas e incluso de funcionamiento de sistemas de control. Sin estas herramientas, todavía se trabajarían con grandes volúmenes de papeleo y registros que fácilmente se extraviarían o mezclarían. Es por esto que los fundamentos teóricos de las bases de datos se han ido mejorando cada día más, orientados a la acumulación de datos masivos, lo que se le conoce como *Big Data* y minería de datos, el área de las ciencias biomédicas no es la excepción. A lo largo

del desarrollo de esta tesis se ocuparán las técnicas conocidas para el diseño e implementación de una base de datos robusta y capaz de sustentar el manejo de miles de datos así como el propio tratamiento de la información para su almacenamiento.

Un medio muy recurrente por el cual se hace público el flujo de la información es mediante la web, existen muchos lenguajes que permiten una fácil y sencilla programación web que se han tornado populares como HTML y PHP.

Así como las bases son esenciales para el almacenamiento de la información así una página de consultas es importante ya que será la interfaz máquina-hombre en la cual todo esa colección de información será interpretada y se le dará un sentido propio del área de la información que se tenga, esto quiere decir su utilidad.

MARCO TEÓRICO

CAPÍTULO II

2.1. Biología Molecular.

2.1.1 ADN, ARN y el dogma central de la biología molecular.

2.1.1.1 Fundamento Químico; Los enlaces químicos la clave de las estructuras complejas biológicas.

Las células como la unidad anatómica y funcional de los seres vivos son diversas y su estudio puede ser extenso, tanto como uno lo desee. Estos sistemas complejos de interacciones físicas y químicas son el sustento de la vida, de ahí la importancia biológica que conlleva hacer cientos de experimentos para conocer más sobre su comportamiento. Será propósito de este subtema el identificar y reconocer la importancia de los enlaces químicos para la formación del ADN, el ARN y las proteínas. Para ello se explicarán las nociones básicas de los enlaces químicos enfocándonos a las moléculas y elementos que se trabajarán más adelante.

Por dar un ejemplo de la gran influencia de la química para la sustentación de la vida en nuestro organismo, en consecuencia de las células, es el papel funcional que posee el agua. Si bien se sabe que el agua representa del 70 al 80% del peso de la mayoría de las células y que es la molécula más abundante en el ser humano. Es el medio principal de vida primigenia que existió desde hace millones de años en la Tierra. Cabe mencionar que alrededor del 7% del peso de la materia viva son iones y algunas moléculas pequeñas como los aminoácidos, nucleótidos, lípidos y azúcares. Para que exista un funcionamiento adecuado de la célula dichas moléculas contribuyen a la estabilidad y a la reproducción celular. Si dicha estabilidad no se produce, se crean todos los factores necesarios para la manifestación de una enfermedad y la muerte celular de un organismo.

Los enlaces covalentes y no covalentes son fundamentales porque establecen las interacciones atómicas y moleculares de la célula; el enlace covalente es primordial para la unión de los aminoácidos y la conformación de los mismos, los enlaces no covalentes obligarán a las proteínas adoptar una forma tridimensional y ayuda a mantener unidas estructuras complementarias. Cuando se habla de los enlaces covalentes se hace referencia a interacciones químicas fuertes en cambio los enlaces no covalentes son interacciones químicas débiles ambos enlaces son relaciones interatómicas de atracción, de forma que cuando existen interacciones fuertes se forman los enlaces covalentes, cuando dos átomos comparten un par de electrones o múltiples pares de electrones. Por otro lado las interacciones débiles forman enlaces no covalentes y pueden ser de cuatro tipos; enlaces iónicos, puentes de hidrógeno, interacciones de Van der Waals y el efecto hidrófobo.

Como bien se sabe los elementos más abundantes en la Tierra son el carbono, nitrógeno, hidrógeno, fosforo, oxígeno y azufre; es muy raro verlos de forma aislada, por lo que tienden a formar enlaces con otros elementos y en muchas ocasiones estos enlaces son covalentes mediante el uso de sus electrones que residen en los orbitales electrónicos más externos. La regla que se cumple aquí es que cada átomo forma un número característico de enlaces covalentes con otro átomo, esto a la larga también definirá una geometría en específico para las diversas uniones que pueden existir. El carbono es capaz de formar cuatro enlaces covalentes, el hidrógeno sólo uno, el azufre dos, cuatro o hasta seis, el nitrógeno de tres a cuatro enlaces, el oxígeno dos y el fosforo cinco. Se pondrá como ejemplo el más comúnmente utilizado por muchos libros; el metano (CH_4). El metano tiene un carbono central y cuatro átomos de hidrógenos cuyo ángulo de separación entre cualquiera de dos enlaces es de 109.5º lo que da una forma similar a un tetraedro. Cuando un átomo de carbono está unido a cuatro átomos disimiles o grupos con configuración no planar se dice que es asimétrico y este tipo de moléculas se les llaman isómeros ópticos o estereoisómeros ya que pueden disponer dos formas distintas en el espacio tridimensional. En la célula muchas moléculas contiene un carbono asimétrico también llamado carbono quiral y contribuye a definir una estructura geométrica, que incluso dichas estructuras geométricas en tercera dimensión por ejemplo de una proteína, son modeladas en el área de la computación de virtualización y procesamiento de imágenes.

Como se mencionó un enlace covalente se puede presentar al compartir un par de electrones lo que se llama enlace simple; o bien múltiples: dobles, triples enlaces, estos son de vital importancia para una molécula en la determinación de su planaridad, rigidez y flexibilidad, si se tuviéra enlaces simples y si no existieran fuerzas externas podría rotar libremente, en cambio un doble enlace implica rigidez en la molécula Las moléculas como el amoniaco (NH_3) y el ion amonio $(NH_4)^+$ son importantes de destacar, debido a que estas moléculas además de formar enlaces covalentes pueden formar enlaces no covalentes y presentar formas tetraédricas respectivamente. Mientras que el ácido fosfórico (H_3PO_4), el fosforo y los fosfatos derivados que forman parte de los ácidos nucleicos, además de desempeñar un papel importante en la regulación de actividades de las proteínas, son parte de la molécula central de la energía celular, el ATP (Adenosín trifosfato).

En la naturaleza no todos los átomos ejercen la misma fuerza de atracción, esto va orientado a lo que se conoce como electronegatividad, está se define como la magnitud de la capacidad de atracción de un átomo para atraer un electrón, cuando ocurre un enlace con átomos de la misma electronegatividad, el enlace formado es un enlace no polar de lo contrario se tendrá un enlace polar. De manera que en el extremo de un enlace polar se tendrá una carga parcial positiva y en el otro extremo una carga parcial negativa; por ejemplo el dipolo eléctrico típico del agua, el electrón compartido del hidrógeno al oxígeno tenderá a estar más cercano a este último que al hidrógeno, por consecuencia en la moléculas del agua y con la existencia de los

dipolos, se podrán establecer enlaces no covalentes entre ellas mismas y con otras moléculas. Algo parecido ocurre con otras moléculas como H_3PO_4 donde se presenta un híbrido de resonancia. La energía requerida para romper un enlace covalente es mayor a la de un enlace no covalente en condiciones de temperatura ambiente, por lo que los enlaces covalentes son más estables, sin embargo, los enlaces no covalentes se pueden unir entre ellos mismos para producir asociaciones aún más estables aunque su existencia sea transitoria.

Uno de los tipos de enlaces no covalentes más conocidos es el enlace iónico, como resultado de atracción entre un anión (ion cargado negativamente) y catión (ion cargado positivamente), la orientación geométrica no es específica como en los enlaces covalentes, esto debido a que la atracción de un ion es uniforme en todas las direcciones, en la Biología Molecular los iones Na^+, K^+, Ca^{2+}, Mg^{2+} y Cl^- toman un papel crítico en mantener el equilibrio en el funcionamiento de la célula y participan en procesos regulatorios para la fisiología humana como la bomba sodio-potasio y los potenciales de acción. En general, los iones mencionados anteriormente están hidratados por un extremo del dipolo por medio de la carga negativa de las moléculas de agua, dicha capa hidratada es eliminada cuando los iones tienen interacción directa con las proteínas. Es más probable que dos iones se puedan enlazar cuando existen mayores concentraciones de otros iones, por ejemplo si se incrementa la concentración de sal en una solución de moléculas biológicas, las interacciones iónicas de la sal se debilitarán y posiblemente también la unión de otros iones que existan en esa solución.

Otro tipo de enlace no covalente son los puentes de hidrógeno, estos se producen debido a la interacción de un átomo de hidrogeno parcialmente cargado positivamente en un dipolo con electrones disponibles de otro átomo. Generalmente un átomo de hidrogeno puede formar un enlace covalente, sin embargo, puede formar una asociación débil con un átomo aceptor teniendo un par de electrones no compartidos. En los puentes de hidrógeno existe otra característica denominada direccionalidad, la cual puede ser lineal o no lineal, los puentes de hidrógeno no lineales contribuyen en estabilizar la estructura tridimensional de una proteína.

Cuando existen fluctuaciones aleatorias de átomos, cabe la posibilidad que uno de ellos se enlace débilmente con otro átomo pero dicha unión es transitoria, estas perturbaciones establecen dipolos de corta duración y pueden ocurrir en los enlaces no covalentes y los covalentes polares, este fenómeno químico es llamado interacciones de Van der Waals. La condición necesaria para la existencia de dichas interacciones va ligada a la cercanía de los átomos, es decir entre menor distancia exista en las moléculas de una región habrá mayores interacciones de Van der Waals, de lo contrario entre mayor sea la distancia habrá escasas interacciones de este tipo. Las interacciones de Van der Waals, son más débiles de romper que los puentes de hidrogeno aproximadamente 1 [Kcal/Mol].

Finalmente, el otro tipo de enlace no covalente es el que hace referencia al efecto hidrófobo, debido a la existencia de moléculas no polares y estos al no tener grupos cargados son insolubles en agua, son llamadas moléculas hidrófobas, como son los hidrocarburos, aceites vegetales y los triglicéridos, de hecho el agua no puede formar puentes de hidrogeno con estas sustancias. Sin embargo las moléculas de agua tienden a formar estructuras pentagonales y hexagonales alrededor de las moléculas no polares, estos enlaces son relativamente estables y no aumenta la entropía de las moléculas de agua. La relevancia del efecto hidrófobo es cuando se alcanza un estado agregado en las sustancias químicas en un ambiente determinado, puesto que si no se alcanza dicho estado, sería desfavorable para el papel que toma el agua en los procesos de la célula.

Para que exista un completo orden en las uniones de átomos, moléculas o iones se requiere de complementariedad molecular ejemplificado muchas veces como una llave en una cerradura haciendo hincapié al ajuste fino. Entonces se puede decir que cuando dos moléculas estructuralmente son complementarias pueden unirse al momento de chocar. Y unido al concepto de complementariedad molecular está la especificidad, como la fuerza para unirse a una molécula en especial o familias similares de esta molécula.

2.1.1.2 Unidades químicas estructurales de la célula.

Existen cuatro macromoléculas biológicas esenciales para las células: los aminoácidos, los ácidos nucleicos, los polisacáridos y los lípidos, será tema fundamental de este libro hablar de los primeros dos, puestos que son la base para la comprensión del DNA, ARN y proteínas. Existen subunidades muy pequeñas llamadas monómeros que son casi idénticas entre sí, las cuales al unirse mediante enlaces covalentes forman polímeros, en casi todos los casos los enlaces covalentes entre monómeros se forman por reacciones de deshidratación.

En el caso de las proteínas, son polímeros lineales que contienen desde diez hasta miles de aminoácidos unidos por enlaces peptídicos.

Los ácidos nucleicos:

En los seres vivos existen dos tipos de ácidos nucleicos: el DNA (ácido desoxirribonucleico) y el RNA (ácido ribonucleico), estas moléculas son portadoras de la información hereditaria y primeros responsables de la producción de proteínas dentro de la célula. Los monómeros que constituyen estos polímeros son los nucleótidos. La estructura común de un nucleótido es la siguiente:

- Un grupo fosfato: se le conoce así al ion formado por un átomo central de fosforo unido con 4 moléculas idénticas de oxígeno en disposición tetraédrica.
- Una pentosa: es una molécula de azúcar de cinco carbonos.

- Una base: estructura en forma de anillo compuesta por nitrógeno y carbono, también llamada base nitrogenada.

El grupo fosfato se une mediante un enlace fosfoéster a la pentosa y ésta se une con la base. Las bases adenina y guanina son purinas, las cuales contienen un par de anillos fusionados y las bases citosina, timina y uracilo son pirimidinas y contienen solo un anillo. Se abrevian A, G, C, T y U (tal y como fueron mencionadas) para denotar una cadena larga de bases en el DNA o RNA. En forma más específica, el átomo del carbono 1' de la pentosa se une al nitrógeno de la posición 9 de una purina (N_9) o en la posición 1 de una pirimidina (N_1), el grupo fosfato es la razón principal que provoca el carácter de acidez en los nucleótidos, debido a que en condiciones normales libera un ion hidrógeno. En el líquido extracelular hay presencia de nucleósidos, estos son similares a los nucleótidos excepto que no tienen el grupo fosfato, es decir, son bases nitrogenadas unidas a una pentosa. (Fig 2.1 y Tabla 2.2)

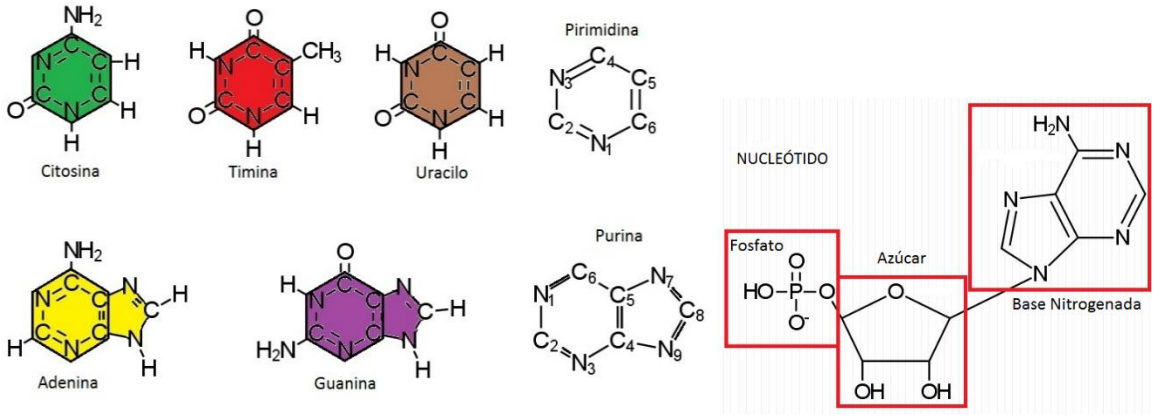


Fig. 2.1 En la parte de arriba se muestran las bases nitrogenadas y sus clasificaciones, y se muestra la estructura general de un nucleótido. Imagen realizada en ChemDoodle 2014

La gran diferencia entre RNA y DNA es la pentosa; en el RNA es ribosa y el DNA es desoxirribosa, además de que en el RNA se encuentra la base nitrogenada uracilo, en lugar de la timina que está presente en el DNA. En ambos existen las bases adenina, guanina y citosina.

Los aminoácidos:

Los monómeros que componen las proteínas son 20 aminoácidos (Tabla 2.1), todos tienen la siguiente estructura:

- Un átomo carbono α central (C_α).
- Un grupo amino (NH_2).
- Un grupo carboxilo ($COOH$).
- Un átomo de hidrogeno.
- Un grupo variable de cadena lateral (R).

El átomo de carbono se une a los cuatro grupos restantes y debido a que es asimétrico en todos los aminoácidos excepto en la glicina pueden existir formas

especulares como; isómeros *D* (dextro) y *L* (levo) estos últimos se encuentran en las proteínas. Las características y propiedades como tamaño, forma, carga, etc., del aminoácido están determinadas por el grupo variable, de forma que los aminoácidos también pueden ser clasificados por su solubilidad en agua. Las cadenas laterales polares son hidrofílicas, generalmente están en la superficie de una proteína, para poder interactuar con el agua y en la formación de enlaces no covalentes con otras moléculas. Así también, las cadenas laterales no polares son hidrofóbicas y se agrupan para formar el núcleo insoluble de una proteína. Se agrega una tabla al finalizar este subtema donde se pueden consultar los aminoácidos con sus clasificaciones de solubilidad.

La relevancia que adquieren los aminoácidos es tan importante que permitirá a una proteína realizar ciertas funciones, por ejemplo, en los aminoácidos hidrofílicos como la arginina y lisina están cargados positivamente y los ácidos aspártico y glutámico están cargados negativamente, estos cuatro aminoácidos contribuyen a la carga global de la proteína. Un caso peculiar es la histidina que puede estar cargada negativamente o positivamente con pequeños cambios de acidez del medio. La aspagarina y glutamina carecen de carga pero tienen la capacidad para formar puentes de hidrogeno muy fácilmente debido a la existencia de sus grupos amido, así como la serina y treonina que tienen grupos hidroxilos polares. En cambio los aminoácidos hidrófobos, que como su nombre lo dice son insolubles en el agua o levemente solubles, presentan cadenas no cíclicas como alanina, valina, leucina, isoleucina y metionina. Estos aminoácidos están compuestos por residuos de carbono y en el caso de la metionina, además contiene un átomo de azufre. En el caso de la fenilanina, tirosina y triptófano tiene cadenas grandes aromáticas y la cisteína, glicina y prolina realizan funciones muy especiales debido a la singularidad de sus cadenas laterales, por ejemplo en estabilizar la estructura plegada de una proteína.

Por las particularidades mencionadas arriba, cada aminoácido contribuye de manera muy especial en las proteínas, desde su funcionamiento hasta su estructura física. Por lo que es trabajo de la biología molecular el determinar la secuencia de aminoácidos que forma una proteína en un organismo particular mediante procedimientos que se han ido desarrollando a lo largo del tiempo. Al obtener la secuencia de aminoácidos se pueden generan ciertas estadísticas para el reconocimiento de patrones y visualizar ciertas características en común cuando se presenta una cierta enfermedad o una cierta función de una proteína en una célula.

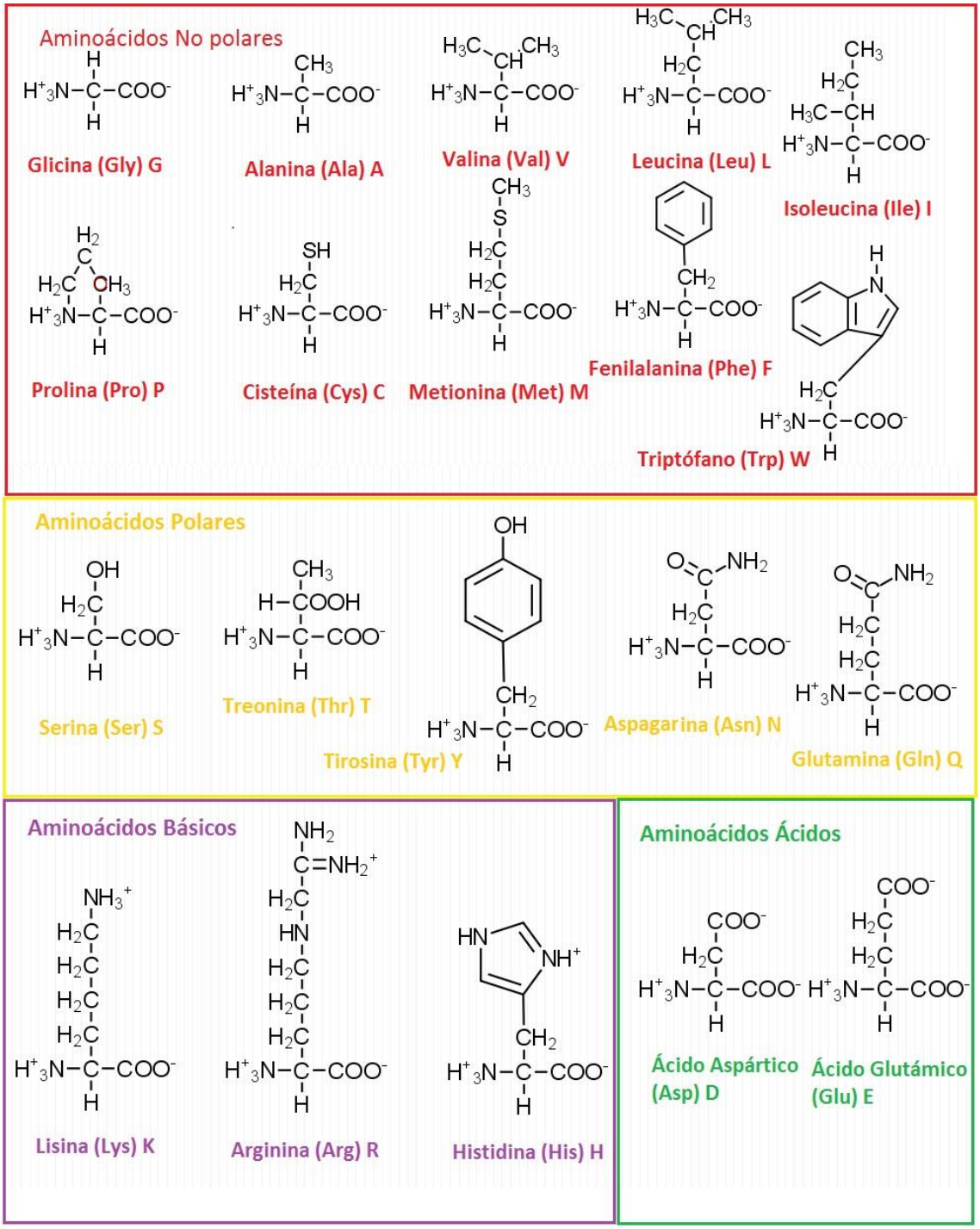


Tabla 2.1 Se muestran los 20 aminoácidos con su estructura química y los nombres que se muestra entre parentesis es la abreviación en 3 letras seguida de una letra utlizadas para identificarlos. Imagen realizada en ChemDoodle 2014.

	Bases			
	Purinas		Pirimidinas	
	Adenina (A)	Guanina (G)	Citosina(C)	Uracilo(U)/ Timina(T)
Nucleósidos: en RNA en DNA	Adenosina	Guanosina	Citidina	Uridina
	Desoxiadenosina	Desoxiguanosina	Desoxicitidina	Desoxitimidina
Nucleótidos: en RNA en DNA	Adenilato	Guanilato	Citidilato	Uridilato
	Desoxiadenilato	Desoxiguanilato	Desoxicitidilato	Desoxitimidilato
Nucleósidos monofosfatos	AMP	GMP	CMP	UMP
Nucleósidos difosfatos	ADP	GDP	CDP	UDP
Nucleósidos trifosfatos	ATP	GTP	CTP	UTP
Desoxinucleósidos monofosfatos, di y trifosfatos	dAMP, etc.	dGTP, etc.	dCTP, etc.	dUTP, etc.

Tabla 2.2. Se muestra la nomenclatura de los nucleósidos y nucleótidos separados por la base que contienen.

2.1.1.3 Mecanismos genéticos moleculares básicos; el dogma central de la biología.

Como se mencionó anteriormente, se busca analizar los procesos biológicos que determinan la formación de las proteínas para después estudiar los cambios que ocurren posteriormente a su formación y observar como alteran las funciones de las proteínas en la célula. Para ello es necesario entender el dogma central de la biología: la duplicación del DNA, la transcripción a RNA y la traducción proteica.

Se parte del hecho de que los ácidos nucleicos cumplen con las siguientes características:

1) Contienen la información para determinar la secuencia de aminoácidos de una proteína, que darán como resultado su estructura y función de la misma.
2) Seleccionan y alinean aminoácidos en el orden correcto a medida que se sintetiza una cadena polipeptídica.
3) Catalizan numerosas reacciones químicas necesarios para la producción de proteínas, como la formación de los enlaces peptídicos entre los aminoácidos de una proteína.

Se requiere tener un control estricto para la replicación del DNA y su transcripción, ya que en el DNA se contiene toda la información para construir un organismo completo, la replicación exacta del DNA asegura la continuidad genética de los seres vivos de generación en generación. La información acumulada en el DNA está organizada en unidades hereditarias llamadas genes. Será tema a tratar en este subtema el proceso de transcripción y traducción, así como la replicación del DNA.

En el proceso de la transcripción la información del DNA es copiada a RNA, mediante tres etapas importantes; la iniciación, elongación y terminación, el mRNA (RNA mensajero) lleva las instrucciones del DNA que especifican el orden correcto de los aminoácidos en la síntesis de la proteína. El ensamblaje altamente preciso y exacto es llevado a cabo en la traducción del mRNA; en este paso la información del mRNA es interpretada por un segundo tipo de RNA llamado tRNA (RNA de transferencia) en colaboración con el rRNA (RNA ribosómico) y algunas proteínas asociadas. A los procesos anteriores se le llama el dogma central de la biología, puesto que siempre se siguen estos pasos para la subsistencia de los seres vivos y su equilibro celular.

A pesar que la estructura del DNA fue descubierta en 1953 por el físico Francis Crick y el biólogo James Watson, aún falta mucho por saber de esta misteriosa molécula y el basto mundo complejo de lo que representa la genética y la biología molecular. Sin embargo, el descubrimiento del DNA y la descripción de los procesos de transcripción y traducción marcaron el inicio de una era de importantes estudios que permitieron ayudar a entender de mejor manera el mundo que nos rodea y la trascendencia como especie.

Como se puede observar, las proteínas también participan en la transcripción y traducción, el papel que asumen estas proteínas son de regulación y son fundamentales para el control y formación de otras proteínas. En la siguiente imagen se esquematiza los procesos biológicos del dogma central de la biología que se analizara con mayor detalle.

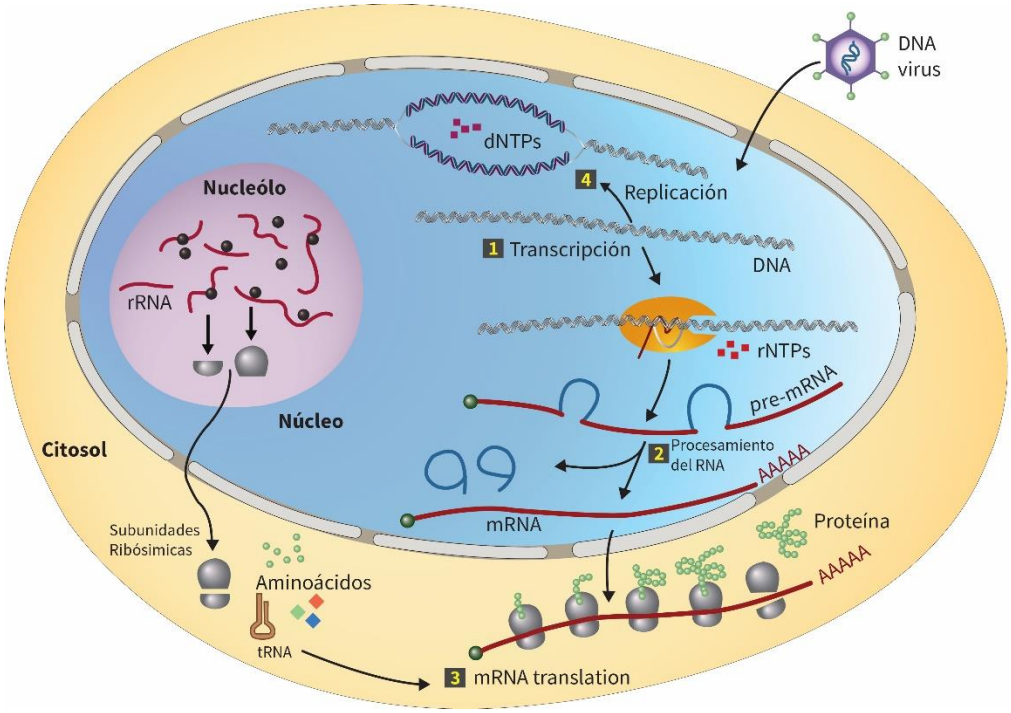


Fig. 2.2. En este esquema se muestran los procesos involucrados para producción de proteínas que van desde el 1 hasta el 3 y el proceso de la replicación del DNA el punto 4. 1. Se transcribe un gen

en especial, que codifica una proteína, la maquinaria de la RNA polimerasa participa dentro de este proceso. 2. Se construyen un precursor del mRNA mediante la proliferación de monómeros de rNTP (ribonucleósido trifosfato) y ocurre la eliminación de secuencias extrañas del pre-mRNA llamado el procesamiento del RNA. 3. El mRNA funcional producido en la etapa anterior se transporta al citoplasma para su traducción, en esta etapa el código de 4 bases que conforman los ácidos nucleótidos es traducido al lenguaje de los 20 aminoácidos. Los ribosomas compuestos por una subunidad mayor y otra menor y con la ayuda de otras proteínas llevan a cabo la síntesis proteica. 4. La replicación del ADN es llevada a cabo por aquellas células preparadas para su división, los monómeros dNTP (desoxirribonucleósido trifosfato) son polimerizados para la producción de dos copias exactas del DNA. [1]

2.1.1.4 La estructura biológica de los ácidos nucleicos.

Como bien sabe los ácidos nucleicos son dos en los seres vivos; el DNA y el RNA, y se diferencian uno del otro por la pentosa que los conforman y la presencia de una diferente base; la timina y el uracilo, pero fuera de ello, poseen estructuras similares. Es necesario conocer sus estructuras para conocer su replicación y transcripción. La estructura primaria son los polímeros lineales conformados por los nucleótidos, la longitud del RNA es relativamente más corta que la del DNA, de forma que la longitud de un RNA celular va desde un poco menos que los cien hasta miles de nucleótidos y el DNA celular puede contener millones de nucleótidos, estas unidades de DNA se pueden visualizar con técnicas de tinción usadas en biología molecular para visualizar cromosomas.

Como ya se mencionó, la unidad básica de conformación de los ácidos nucleicos es el nucleótido, y pueden existir dos tipos de bases denotadas por letras mayúsculas: A (Adenina), G (Guanina), C (Citosina), U (Uracilo) y T (Timina) clasificadas como bases puricas y pirimídicas. Estos nucleótidos se pueden unir entre sí y tener una orientación química, de tal manera que se puede formar una cadena de nucleótidos unidos por fosfatos y pentosas, las bases quedan expuestas como grupo laterales. La orientación química de una molécula de nucleótidos es de 5’ a 3’ de extremo a extremo, debido a que el carbono 5’ de la primera ribosa y el carbono 3’ de la última ribosa se encuentran libres.

Esta convención que se ha establecido de orientación (5’ -> 3’) de una cadena de ácido nucleicos también ha definido la forma de leer y escribir una secuencia de nucleótidos. Esta propiedad de direccionalidad es de vital importancia. Los enlaces fosfodiéster permiten la unión química entre nucleótidos adyacentes, uno en el carbono 5’ de la pentosa y otra el carbono 3’. Esta regla se cumple en todos los nucleótidos internos de la cadena polinucleotídica, como ya se mencionó, estos determinan la direccionalidad del ácido nucleico. La secuencia lineal de nucleótidos unidos por enlaces fosfodiéster constituyen la **estructura primaria.**

El modelo propuesto por Watson y Crick del DNA es de dos hebras, estas conclusiones se realizaron mediante el análisis de patrones de difracción de rayos X y mediante una construcción cuidadosa de un modelo. Estas dos hebras se

entrelazan entre sí para formar una doble hélice, la parte formada por los azúcares y fosfatos están ubicados en la parte exterior mediante la cual las bases se proyectan hacia el interior, estas se apilan una tras otra en planos paralelos. La orientación de los hebras es antiparalela, es decir las direcciones 5’ -> 3’ de ambas hebras son opuestas entre sí. Las hebras se mantienen juntas debido al apareamiento de las bases, la adenina se une a la timina mediante dos puentes de hidrogeno y la citosina se une a la guanina mediante tres puentes de hidrogeno, este apareamiento se debe a la afinidad química, a la forma y tamaño de las bases. De ahí la importancia de los puentes de hidrogeno ya que otorgan la estabilidad a la doble hélice.

Como se mencionó una molécula de adenina se une a una timina y una citosina con una guanina estas uniones entre una pirimidina y una purina son llamadas pares de bases. A las hebras que forman pares de bases se les dice que son complementarias (Fig 2.3). Existen pares de bases distintas a las típicamente conocidas, aunque en el DNA natural es complicado encontrarlas, en el RNA es muy usual encontrar uniones G-U.

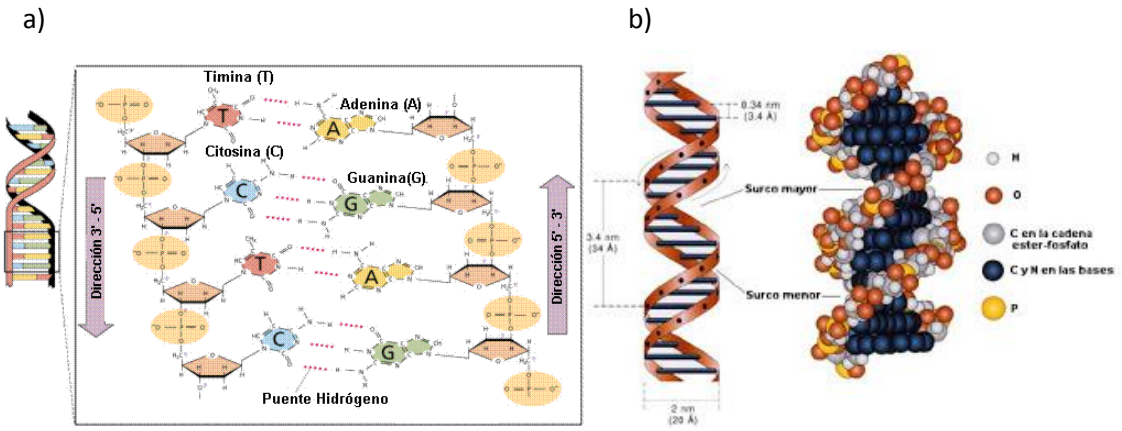


Fig. 2.3 a) En la imagen de la izquierda se muestra el modelo general del DNA con la orientación 5’ -> 3’ en cada una de las hebras y la unión de las bases a través de los puentes de hidrógeno que se muestran como líneas punteadas [2], b) en la imagen de la derecha se muestra la forma B del DNA, la forma más usual de DNA celular, se marcan las hendiduras (Surcos) que son superficies de unión. [3]

Durante la transcripción y la replicación del DNA, las hebras de la doble hélice deben separarse necesariamente para la formación de pares de bases con otros nucleótidos polimerizados en nuevas cadenas de nucleótidos. El proceso de desenrollamiento y separación del DNA es llamado desnaturalización o fusión, y se puede producir incrementando la temperatura en una solución de DNA, esto debido a que los puentes de hidrogeno se rompen resultado del movimiento molecular, las hebras se separan y alejan por repulsiones electrostáticas debido a que cada hebra está cargada negativamente (parte esquelética de la pentosa y el grupo fosfato). La temperatura de fusión (T_m) a la cual las hebras de DNA se separan es un elemento de estudio, puesto que esta temperatura es resultado de muchos factores como por

ejemplo, si existen más pares de bases G-C se requiere temperaturas más altas para la desnaturalización, debido a la presencia de los tres puentes de hidrógeno en G-C, pasa lo contrario con los pares de bases A-T ya que estos últimos sólo tienen dos puente de hidrogeno.

Si se quiere disminuir la temperatura de fusión del DNA, se puede aumentar la concentración iónica al adicionar formaldehído o urea, de esta forma los puentes de hidrogeno se desestabilizaran sin tener una temperatura alta. También se obtendría el mismo efecto si se trabajarían con los extremos de pH ya sea o muy alcalino o muy básico en una solución de DNA.

En el momento de la desnaturalización del DNA, se obtienen dos moléculas de DNA monocatenario o hebra simple, esta es capaz de adoptar formas aleatorias sin una estructura organizada, al igual que se puede probar la separación de las hebras de DNA existe el proceso inverso para volverse a unir, siguiendo los pasos contrarios a la fusión, es decir disminuyendo la temperatura o neutralizando el pH. El proceso de la renaturalización depende del tiempo y sobre todo de la relación en secuencia que debe existir, es decir su complementariedad, si se llegará a tener dos moléculas de DNA monocatenarios en una misma solución y los dos no poseen secuencias relacionadas, no existirá la renaturalización.

Los procesos de desnaturalización y renaturalización son utilizados para estudiar la relación entre dos muestras de DNA como un posible parentesco.

En el caso del RNA, la presencia de un grupo hidroxilo en el C_2 permite que éste tenga mayor habilidad química que el DNA así también provee un grupo reactivo que participa en la catálisis mediada por él mismo. Gracias a esta habilidad, el RNA se activa para formar nucleótidos en una solución alcalina.

El RNA puede estar como una doble hebra o una hebra sencilla y su estructura puede ser lineal o circular y como ya se mencionó las dobles hélices RNA-RNA y DNA-RNA tienen una forma semejante al DNA, sin embargo el RNA celular generalmente existe como una hebra simple y su estructura presenta diversas formas, estas diferencias en tamaños y formas le permite realizar diversas funciones en la célula, por ejemplo “las horquillas” son estructuras secundarias del RNA monocatenario que se presentan por apareamiento de bases separadas entre 5 a 10 nucleótidos entre una y otra, también los “tallos y bucles” se forman por apareamiento de bases separadas por más de 10 nucleótidos (Fig 2.4).

Las formas de RNA secundario cooperan en la formación de un RNA terciario más complicado, como por ejemplo el llamado seudonudo. Para este subtema es importante destacar que el tRNA adquiere una arquitectura tridimensional muy específica ya que esa forma toma un papel muy importante para la síntesis de proteínas.

Se puede decir que las moléculas de RNA poseen dominios bien estructurados conectados por extensiones flexibles menos estructuradas, esta misma característica la tienen las proteínas. En muchas ocasiones esos dominios tienen actividad catalítica, las moléculas de RNA que tienen esta actividad son los ribosomas y participan en la eliminación de intrones, que más adelante se hablara sobre ese tema.

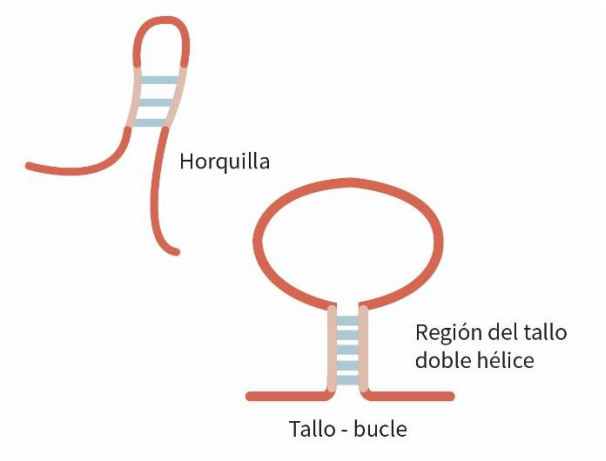


Fig. 2.4 Se muestran dos estructuras secundarias del RNA; la horquilla formada por el apareamiento de nucleótidos con separación de aproximadamente de 5 a 10 nucleótidos y el tallo con separación mayor. [4]

2.1.2 La replicación del ADN.

Uno de los proceso básicos que forman parte del dogma de la Biología Molecular, es la replicación de la molécula que posee la información hereditaria de todo un organismo, es de vital importancia que esta información se mantenga fiel y su copiado tenga un estricto control. El apareamiento de pares de bases en una cadena de DNA mostró a Watson y a Crick que las hebras existentes son sintetizadas para formar nuevas moléculas de DNA hijas. Esta hipótesis se comprobó más adelante como un mecanismo conservativo y uno semiconservativo.

En el modelo conservativo, las hebras hijas podrían formar una nueva cadena de DNA bicatenario mientras que el DNA parental permanecería intacto, en el modelo semiconservativo las hebras hijas formaran dos molécula de DNA dúplex con las dos hebras del DNA parental.

La replicación del DNA será simplemente un copiado de una hebra molde de DNA a una hebra hija de DNA, entonces se puede decir que el RNA y el DNA que se encuentra en las células es producto de un DNA pre-existente.

El DNA se sintetiza a partir de precursores de desoxinucleósidos 5’-trifosfato (dNTP), este proceso tendrá la misma orientación 5’ -> 3’, debido a que el

crecimiento de la cadena resulta de la formación de una enlace fosfoéster entre el oxígeno 3’ de una hebra creciente y el fosfato α de un dNTP. Se necesita de la DNA polimerasa para iniciar el proceso de la replicación del DNA, que a diferencia de la RNA polimerasa que se verá más adelante, no podrá identificar por si sola un sitio de unión en la cadena de DNA, se necesita de una hebra de DNA corta o RNA preexistente llamada cebador. De forma que teniendo el cebador apareado a una hebra molde, una DNA polimerasa adiciona desoxinucleótidos al grupo hidroxilo libre del extremo 3’ del cebador.

El cebador es una molécula de RNA, la hebra hija que se forma es RNA en el extremo 5’ y DNA en el extremo 3’. Para que un DNA bicatenario funcione como molde, las hebras enrolladas entre sí deberán ser desenrolladas, logrando así exponer las bases y que se encuentren disponibles para el apareamiento con las bases de los dNTP.

Las helicasas son las proteínas responsables para la separación de las hebras del DNA, comenzando en segmentos únicos del DNA comúnmente llamados orígenes de replicación u orígenes. Las secuencias que codifican estas regiones particulares del DNA en los organismos suelen ser ricas en A-T. Una vez que la helicasa ha hecho su labor, enseguida una RNA polimerasa primasa forma un cebador corto de RNA complementario a las hebras desenrolladas. Este cebador es elongado por la DNA polimerasa, formando de esta forma una nueva hebra hija.

La zona donde ocurre todo este proceso de abertura, separación y unión para la replicación del DNA es llamada horquilla de replicación o bien horquilla de crecimiento. A medida que la replicación avanza la horquilla de replicación y las proteínas ya mencionadas se van alejando poco a poco del origen. Para aliviar la tensión que se producen durante la separación de las hebras de DNA la topoisomerasa I ayuda a eliminar los superenrollamientos que se forman debido a esa tensión.

La operación de replicación de una horquilla de crecimiento se complica debido a que las dos hebras de DNA son antiparalelas y además las DNA polimerasas solo pueden adicionar nucleótidos en la dirección 5’ -> 3’, de modo que la síntesis de una hebra hija puede continuar a partir de un cebador de RNA en una sola dirección (hebra conductora) o bien una hebra hija rezagada. Esta última se presenta debido a que la hebra hija debe ser sintetizada en dirección opuesta al movimiento de la horquilla de replicación. La célula utiliza un mecanismo alternativo para facilitar la síntesis de la hebra hija rezagada, esto mediante la colocación de un nuevo cebador cada número determinado de bases, haciendo de cierta manera una cadena nueva de DNA discontinua, estos segmentos discontinuos se llaman fragmentos de Okazaki, en honor al descubridor de este proceso.

Finalmente el cebador de RNA para cada fragmento de Okazaki es eliminado y se reemplaza por una cadena de DNA creciente a partir del fragmento de Okazaki vecino, la enzima DNA ligasa se encarga de unirlos.

Como se mencionó anteriormente el proceso de la replicación del DNA parental a hebras hijas se lleva en la dirección 5’ -> 3’, sin embargo dos horquillas de replicación podrían ensamblarse en solo un origen y moverse en direcciones opuestas provocando un crecimiento bidireccional. De hecho esta es la forma las células eucariontes y procariontes llevan su replicación de DNA (Fig. 2.5).

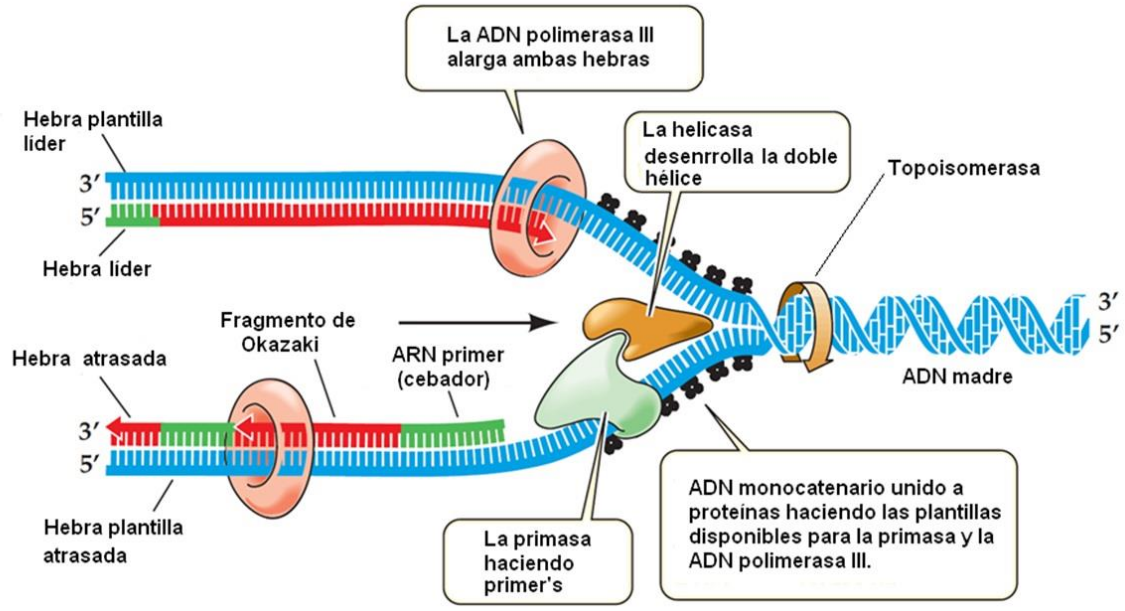


Fig. 2.5. En el proceso de replicación del DNA la hebra conductora (hebra plantilla líder) es sintetizada en la orientación 5’ -> 3’ a partir de un RNA cebador, la DNA polimerasa se encarga de agregar nucleótidos a lo largo de la hebra parental, esta primera hebra sintetizada se hace de forma continua es decir sin interrupciones, en cambio la hebra rezagada (atrasada) es sintetizada en forma discontinua a partir de distintos cebadores de RNA, produciendo de esta forma fragmentos de Okazaki. Posteriormente los cebadores de RNA son eliminados y se religan los fragmentos de Okazaki. [5]

2.1.3 La transcripción y traducción, procesos genéticos.

Para abordar este tema, se partirá de la definición de un gen:

“Es la unidad de DNA que contiene la información para especificar la síntesis de una única cadena polipeptídica o RNA funcional.” [6]

“El gen es la unidad física básica de la herencia. Los genes se transmiten de los padres a la descendencia y contienen la información necesaria para precisar sus rasgos.” [7]

Casi todos los genes tienen la información necesaria para la construcción de proteínas, a estos genes se les llama codificadores, estos constituyen las moléculas del mRNA de las células. En la síntesis del RNA, el lenguaje de 4 bases del DNA

(A, G, T, C) es copiado o transcrito al lenguaje de 4 bases del RNA, es decir se cambia solamente la T por la U. Y de igual forma en la síntesis de proteínas el lenguaje de cuatro bases del RNA es traducido al lenguaje de los veinte aminoácidos. Para el proceso de la transcripción todo estará en torno en la formación de los mRNA funcionales a partir de los genes codificadores de proteínas. También existe un proceso similar para la formación de rRNA y tRNA funcionales con genes codificadores para producir precursores de estos tipos de RNA.

En el proceso de la transcripción del DNA, una hebra actúa como patrón o molde para formar una cadena complementaria de RNA, esto último a través de la polimerización de monómeros de rNTP complementarios en el área distante.

Las bases que forman parte de la hebra patrón se unen a las bases de los monómeros de rNTP complementarios, al provocarse esta unión se produce otro proceso de polimerización catalizado por la RNA polimerasa, dando como resultado un enlace fosfodiéster entre bases, gracias a este proceso se facilita la adición de ribonucleótidos a la cadena RNA en crecimiento, la proteína encargada de mantener el equilibrio en la reacción es la pirofosfatasa.

Como resultado de la adición de ribonucleótidos y el proceso de polimerización de rNTP, la cadena RNA resultante será complementaria en dirección opuesta 5’ -> 3’ con respecto a la hebra molde de DNA (Fig 2.6). Para facilitar el estudio del proceso de transcripción, los biólogos y genetistas han establecido que el lugar donde ocurre el inicio de este proceso como la posición +1. De forma que la síntesis de mRNA llevará un orden de corriente abajo, es decir que a partir de la posición +1 hasta el extremo 3’ del DNA.

Con esta convención se puede decir que los nucleótidos que se hayan corriente arriba de la posición inicial del proceso de transcripción se numeran con signos negativos y los que están cuesta abajo números con signo positivo.

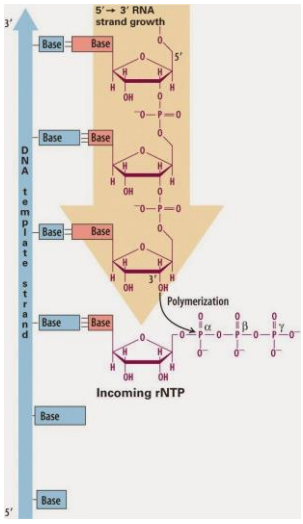


Fig. 2.6 Representación de la polimerización de los monómeros de rNTP en la cadena creciente de RNA, que consiste en una ataque nucleofílico y la producción de un enlace fosfodiéster. [8]

2.1.3.1 Etapas del proceso de transcripción.

El papel que toma la proteína RNA polimerasa es sumamente indispensable ya que no sólo cumple con una función en especial sino con todo una seria de eventos para el control y un adecuado proceso de transcripción (Fig. 2.7), a continuación se explican brevemente:

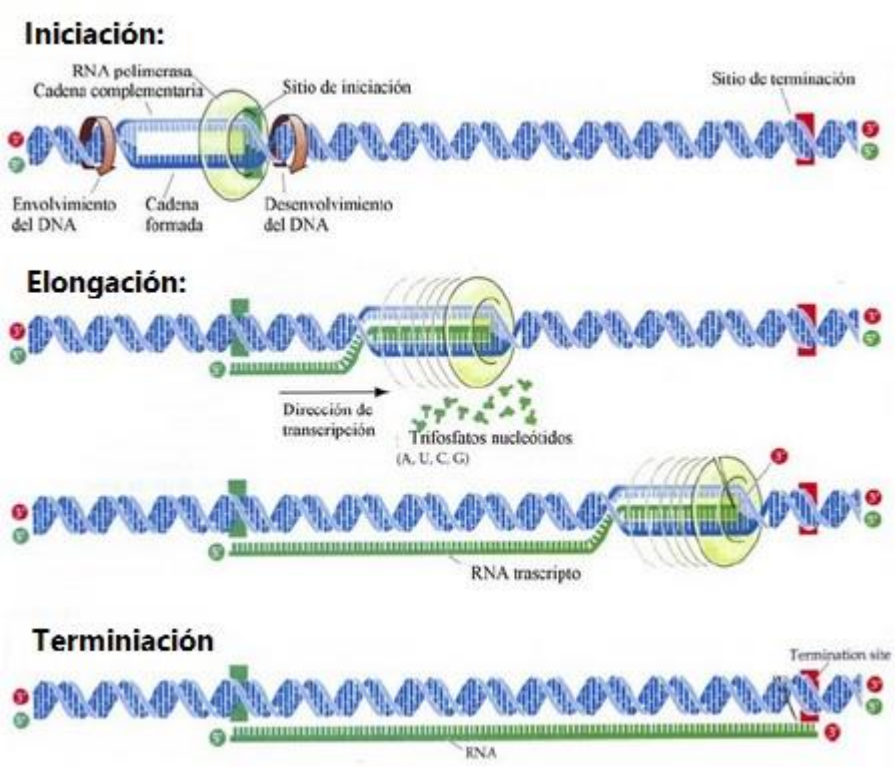


Fig. 2.7 En la imagen se pueden observar las etapas de la transcripción; la iniciación cuando la RNA polimerasa se une al promotor y separa las hebras de DNA, después la etapa de elongación donde la RNA polimerasa se va desplazando base por base para dar como resultado el surgimiento de una cadena mRNA en crecimiento y finalmente la etapa de terminación donde exista la secuencia necesaria dentro de la cadena de DNA para finalizar este proceso. [9]

- Etapa de Iniciación: En primer lugar la RNA polimerasa reconoce un sitio específico del DNA bicatenario, llamado promotor. En este paso de reconocimiento participan un conjunto de proteínas llamados factores de transcripción. Cuando se haya establecido la unión de la RNA polimerasa con el promotor, se disocia las hebras del DNA para que las bases de los ribonucleósidos trifosfatados se apareen y se lleve a cabo el proceso de polimerización. La etapa de iniciación termina cuando se establece la unión de las primeras dos bases de ribonucleótidos de una cadena de RNA, a través de un enlace fosfodiéster.

- Etapa de elongación: Después de la polimerización de varios ribonucleótidos, la RNA polimerasa se separa del promotor y también de los factores de transcripción para llevar a cabo una etapa de elongación. En esta etapa la RNA polimerasa se mueve a lo largo del DNA molde de base en base, abriendo el DNA por adelante e hibridando las hebras detrás de sí. Durante esta etapa más ribonucleótidos se van adicionando uno por uno al extremo 3’ de la cadena de RNA creciente. La labor de la RNA polimerasa y del complejo de elongación es mantener una velocidad constate de pares de bases por minuto, aunque la molécula de RNA naciente es sumamente estable, se requiere mantener este proceso intacto de otros para asegurar la síntesis interrumpida de mRNA.

- Etapa de terminación: Durante la etapa final de la síntesis de mRNA, el transcrito primario, se libera de la RNA polimerasa y también se separa del DNA molde. La culminación se realiza mediante secuencias específicas de nucleótidos en el promotor que le señalan a la RNA polimerasa cuando parar. Cuando se libera la RNA polimerasa, esta lisa para volver a transcribir el mismo gen u otro distinto.

Como se puede observar el papel que adquiere la RNA polimerasa es muy importante esto debido a su conformación estructural, es similar en todos los organismos, tanto en función como en estructura. A continuación se muestra la estructura de una RNA polimerasa que ha sido caracterizada:

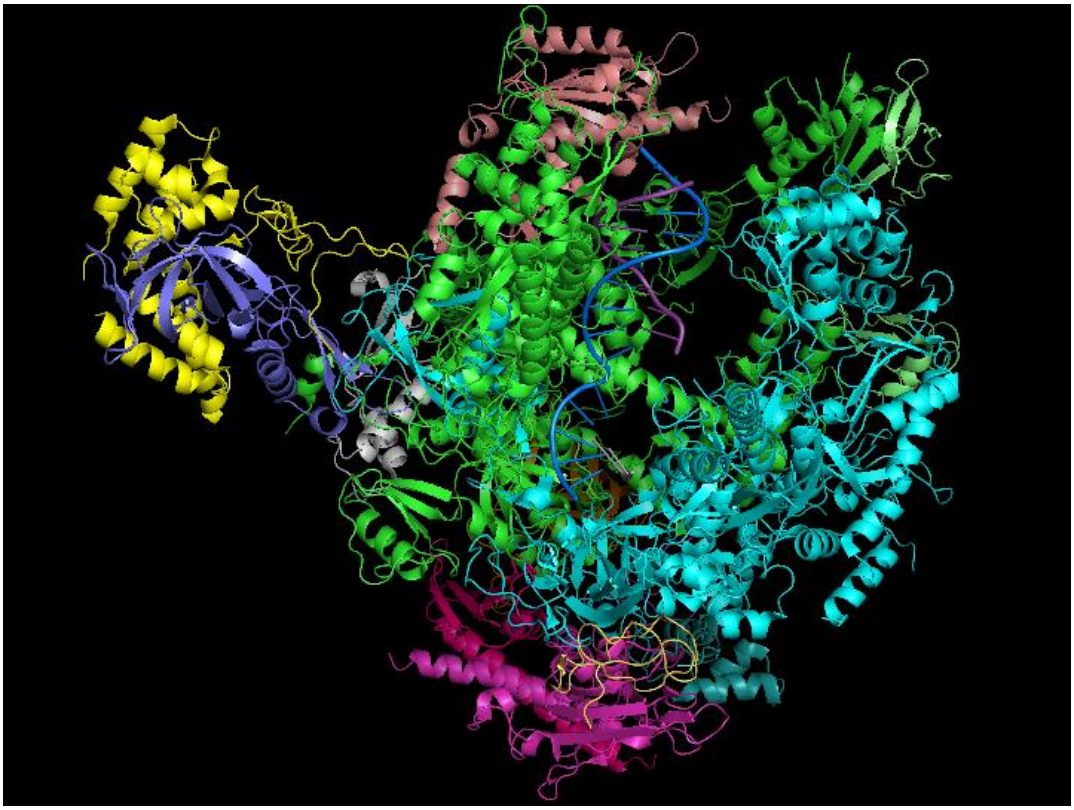


Fig. 2.8 Estructura de la RNA polimerasa, se puede ver un modelo general con sus subunidades en formas de cadena (señaladas por colores), se puede observar la cadena de DNA que está siendo transcrita. Imagen realizada en PyMol Molecular Graphics System v.1.3. [10]

Estudios recientes de genomas de organismos han revelado el número de genes codificadores de proteínas y así como la organización peculiar entre procariontes y eucariontes, por ejemplo en las células procariontes existe una notable lógica, es decir los genes dedicados a un único objetivo metabólico se suelen encontrar en una disposición contigua en el DNA. Dicha disposición de genes en un grupo funcional es llamada operón. La transcripción del operón produce una hebra continua de mRNA que después de ser traducido formará varias proteínas diferentes, de forma tal que cada parte del mRNA que contiene la información de un gen será una proteína funcionalmente relacionada con la que le antecede en la hebra de RNA. En el DNA de los procariontes casi no existen discontinuidades entre promotores por lo que el mRNA se transcribe directamente en forma colineal.

Estos grandes cúmulos de genes dedicados a una función metabólica no se presentan en las células eucariontes, aunque dichas funciones metabólicas se parezcan como es el caso de la levadura. En las células eucariontes los genes especializados en una única vía metabólica están separados físicamente en el DNA, e incluso suelen estar en diferentes cromosomas. Cada gen es codificado por su propio promotor y transcrito en un mRNA que se traducirá en un péptido único. Por estudios sobre los mRNA como una secuencia codificante ininterrumpida llegaron a la conclusión de que la secuencia es discontinua en la secuencia del DNA molde,

desde entonces se dice que existen partes o secciones codificantes llamadas exones separados por segmentos no codificantes llamados intrones. Este increíble hallazgo en la biología molecular determinó que se requiere un proceso auxiliar altamente fiel y seguro para eliminar los intrones y juntar los exones, de esta forma se podrá tener un mRNA altamente funcional en los eucariontes.

La siguiente etapa después de la terminación de la transcripción, en el caso de los eucariontes como ya se mencionó, es un procesamiento del mRNA precursor a un mRNA funcional, en el caso de las procariontes el proceso de traducción puede llevarse a cabo simultáneamente en el extremo 5' del mRNA mientras que el extremo 3' aún se sigue sintetizando, esto no es posible en los eucariontes. Además de la distancia que existe entre el núcleo y el retículo endoplásmico rugoso, se tiene primero un precursor de mRNA también llamado transcrito primario que tiene que ser procesado antes de iniciar la traducción.

Una vez que fue procesado el mRNA que ahora es funcional es llevado al citoplasma para iniciar el segundo proceso básico de la genética y biología molecular, la síntesis de las proteínas, de tal forma que la transcripción y la traducción no pueden existir como procesos simultáneos en los eucariontes. Todos los pre-mRNA son sometidos a dos modificaciones iniciales en sus extremos; una de las primeras modificaciones que ocurre es en el extremo 5' del pre-mRNA, a medida que este va emergiendo hacia la superficie de una RNA polimerasa II, el extremo es sometido bajo control por varias enzimas que se encargan de sintetizar el llamado *casquete 5*'.

El casquete 5' consiste en unir un 7-metilguanilato con el nucleótido terminal que se encuentre en ese extremo a través de un enlace inusual 5'-5' trifosfato, este casquete protege al mRNA de cualquier degradación enzimática y posteriormente ayudará a su traslado al citoplasma. Adicional a este casquete se une un complejo proteico que inicializará más adelante el proceso de la traducción. Mientras en el extremo 3' sufre una escisión por acción de una endonucleasa para formar un grupo hidroxilo libre. La enzima denominada poli(A) polimerasa le agrega a este grupo libre una hilera de residuos de ácido adenílico, esta zona del mRNA es llamada *cola Poli (A)*.

La poli(A) polimerasa es parte de un complejo de proteínas que tiene como función localizar y activar un transcrito, agregando un número determinado de residuos de adenina. Finalmente se necesita de los pasos de corte y empalme para dar por terminado el procesamiento del mRNA, esto es el corte de los intrones y la unión de los exones, de tal manera que se tendrá un mRNA funcional donde los extremos son regiones no codificantes llamadas también regiones no traducidas 5' y 3' por sus siglas en inglés UTR. El mRNA en procarionte suele tener un UTR más corto que el mRNA en eucariontes (Fig. 2.9).

En los genes de las células eucariontes al tener múltiples intrones y un número libre de combinaciones de exones, cabe la posibilidad de ciertas variaciones entre ellos,

se piensa que debido a que los exones codifican ciertos dominios proteicos, estos grupos de exones o exón difieren muy poco respecto a su secuencia, es decir son casi idénticos o idénticos. Existe una hipótesis que plantea que estos exones que codifican un mismo dominio existen por la duplicación accidental de un tramo de DNA, que se extendía a lo largo de dos intrones, dando como consecuencia la repetición del exón entre otros intrones.

Lo anterior permite llegar a la conclusión que gracias a la presencia de intrones se permite la expresión de proteínas múltiples y emparentadas de un mismo gen por acción del corte y empalme. Las proteínas isomorfas son un claro ejemplo del claro mecanismo de corte y empalme para los eucariontes superiores.

"La reciente determinación de la secuencia de un gran número de mRNA asilados de diversos tejidos y la comparación de sus secuencias con el DNA genómico ha revelado que casi el 60% de todos los genes humanos se expresan como mRNA proveniente del corte y empalme alternativo." [11]

a)

b)

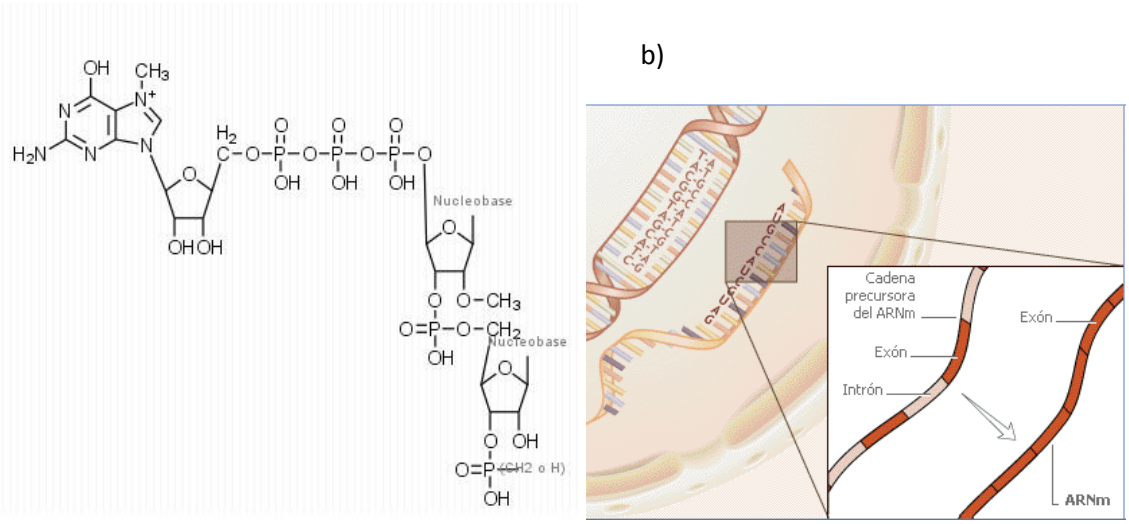


Fig. 2.9 a) Se muestra la estructura química del casquete 5' en un mRNA precursor, como se ha mencionado es importante la existencia de esta región UTR para el proceso de la traducción, imagen realizada en ChemDoodle 2014 b) También se ejemplifica las regiones de los intrones y exones en una cadena precursora de mRNA y que mediante un etapa de procesamiento se tendrá un mRNA funcional. Imagen diseñada en GIMP.

Puesto que para la célula, las proteínas son las que determinan su función y su estructura, su estudio es muy importante en biología molecular. Las decisiones que toma la célula para iniciar la transcripción de un gen que posteriormente se verá reflejado como una proteína, es un mecanismo de control para regular una producción regulada de esta proteína considerando las condiciones fisiológicas, químicas, físicas del ambiente en las que se haya. Este mecanismo de control se basa en detener o incentivar el inicio de la transcripción, el factor a controlar es la frecuencia con que se realizan los procesos, por ejemplo; si la célula quiere reprimir

la transcripción de un gen lo que hará será sintetizar a muy baja frecuencia el mRNA y la proteína o proteínas correspondientes, en caso contrario si quiere activar la transcripción de ese gen aumentara la frecuencia de síntesis de mRNA y proteínas correspondientes.

2.1.3.2 La Traducción del mRNA al lenguaje proteico.

Como ya se mencionó las proteínas realizan la mayoría de las actividades biológicas además de que ellas mismas participan en su síntesis. El ensamble correcto de los aminoácidos durante el proceso de traducción es crítico puesto que determina la síntesis de proteínas funcionales y por consiguiente un funcionamiento correcto de la célula.

	U		C		A		G		
U	UUU	Phe	UCU	Ser	UAU	Tyr	UGU	Cys	U C A G
	UUC	Phe	UCC	Ser	UAC	Tyr	UGC	Cys	
	UUA	Leu	UCA	Ser	UAA	Stop	UGA	Stop	
	UUG	Leu	UCG	Ser	UAG	Stop	UGG	Trp	
C	CUU	Leu	CCU	Pro	CAU	His	CGU	Arg	U C A G
	CUC	Leu	CCC	Pro	CAC	His	CGC	Arg	
	CUA	Leu	CCA	Pro	CAA	Gln	CGA	Arg	
	CUG	Leu	CCG	Pro	CAG	Gln	CGG	Arg	
A	AUU	Ile	ACU	Thr	AAU	Asn	AGU	Ser	U C A G
	AUC	Ile	ACC	Thr	AAC	Asn	AGC	Ser	
	AUA	Ile	ACA	Thr	AAA	Lys	AGA	Arg	
	AUG	Met	ACG	Thr	AAG	Lys	AGG	Arg	
G	GUU	Val	GCU	Ala	GAU	Asp	GGU	Gly	U C A G
	GUC	Val	GCC	Ala	GAC	Asp	GGC	Gly	
	GUA	Val	GCA	Ala	GAA	Glu	GGA	Gly	
	GUG	Val	GCG	Ala	GAG	Glu	GGG	Gly	

Tabla 2.3 Se muestran las combinaciones posibles de bases nucleótidas (lenguaje de 4 bases mRNA) para la formación de los 20 aminoácidos conocidos (representados por 3 letras), la columna de la izquierda es el primer nucleótido del codón, la fila de letras de arriba es la segundo y la columna de la derecha es el tercero. Es importante mencionar que la combinación GUG suele codificar valina, sin embargo, también puede codificar metionina, como es el caso de la leucina codificada por CUG, esta metionina es codón de inicio pero rara vez se presenta.

La Traducción es un proceso biológico en donde la secuencia de nucleótidos del mRNA se utiliza para ordenar y unir los aminoácidos en una cadena polipeptídica, este proceso tiene lugar en el citoplasma y participan tres tipos de moléculas distintas de RNA en forma cooperativa:

1. El RNA mensajero (mRNA): tiene la información genética transcrita desde el DNA, la secuencia de nucleótidos es interpretada de tres en tres llamados codones, cada codón especifica un aminoácido en especial.

2. El RNA de transferencia (tRNA): es necesario para descifrar los codones en el mRNA. Cada tipo de aminoácido tiene su subgrupo de tRNA, que une al aminoácido y lo transporta en el extremo de la cadena creciente polipeptídica. Para evitar errores, el tRNA correcto es seleccionado dependiendo de su anticodón que tendrá que aparearse con las tres bases que forman el codón complementario en el mRNA.
3. El RNA ribosomal (rRNA): se mueve a lo largo del mRNA para catalizar el ensamblaje de aminoácidos en formación de cadenas polipeptídicas, generalmente se asocian a un conjunto de proteínas para formar los ribosomas, adicionalmente también se unen a los tRNA y diversas proteínas auxiliares para la síntesis de proteínas. Los ribosomas están compuestos por una subunidad mayor y una menor, cada una tiene su propia molécula de rRNA.

Se explicará con mayor detalle cuál es la función de cada uno de ellos en el proceso de la traducción y como se relacionan entre sí. Las cadenas polipéptidicas resultantes sufren plegamientos postraduccionales y estos cambios son necesarios para hacer a una proteína funcional madura como por ejemplo algunas modificaciones químicas, que más adelante se explicarán.

De los 64 codones posibles (Tabla 2.3) en el código genético, 61 codifican aminoácidos individuales y tres codones son de terminación, la mayoría de los aminoácidos son codificados por más de un codón, sólo la metionina y el triptófano son codificados por un único codón en caso contrario la leucina, la serina y la arginina están codificados por seis codones, se dice que el código es degenerado porque más de un codón puede especificar el mismo aminoácido. Como se mencionó estos codones están contenidos de tres en tres bases del mRNA y su traducción comienza con la metionina (AUG), que será codón de inicio o iniciador para las células eucariontes y procariontes.

En algunas células eucariontes se consideran como codón inicial CUG, en el caso de los codones UAA, UGA y UAG serán considerados como codones de terminación que marcan el carboxilo terminal de las cadenas polipeptídicas. Se ha establecido un marco de lectura para conocer la secuencia de aminoácidos que tiene una proteína, para ello, se considera la secuencia a partir del codón de inicio específico hasta un codón de terminación, debido a que este código genético es un código de triplete habrá tres marcos de lectura diferente. Esta disposición lineal de ribonucleótidos en el mRNA determina la secuencia lineal exacta de los aminoácidos que conformarán una proteína.

Algunos mRNA han demostrado tener información superpuesta en los diferentes marcos de lectura, pero generalmente los mRNA sólo pueden ser leídos en un marco de lectura por exclusión de los otros dos marcos de lectura, debido que estos llegan a los codones de terminación primero antes de que se complete toda la secuencia de la proteína funcional. También existe otra forma de leer la secuencia

de aminoácidos en una proteína como el *frameshifting* o corrimiento de marco de lectura.

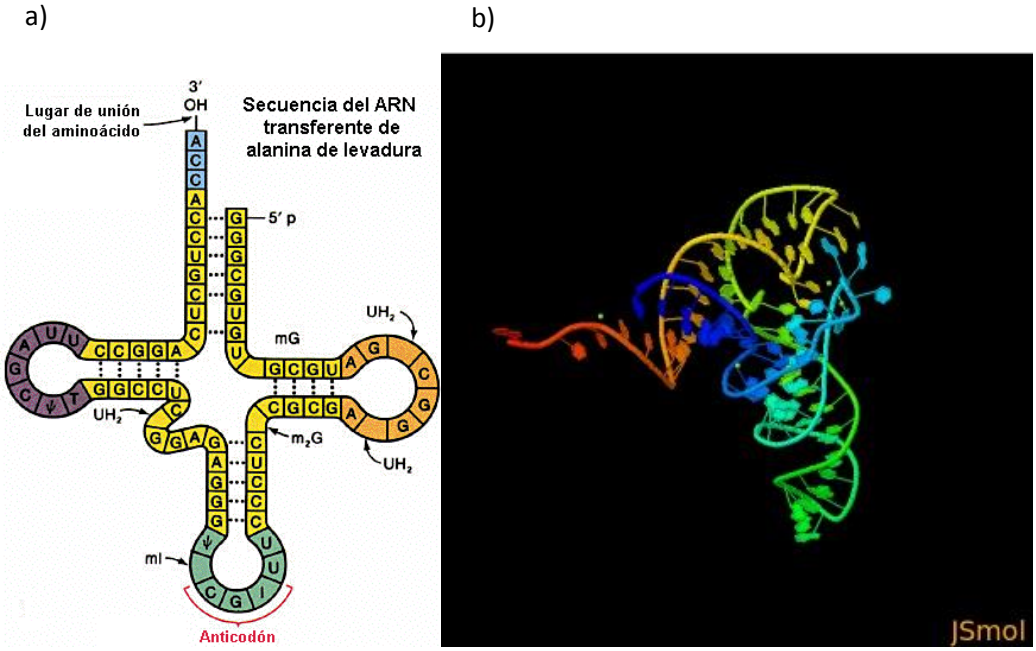


Fig. 2.10 a) Se muestra la estructura del tRNA; cuatro tallos de bases apareadas y tres bucles, la secuencia CCA aparece en todos los tRNA en el extremo 3', también la unión de un aminoácido en este extremo forma aminoacil-tRNA. Las bases estándares A, U, G y C se modifican, tal es el caso de la dihidrouridina (UH_2) que siempre aparece en ese bucle variable del tRNA, también la ribotimidina (T) y seudouridina (Ψ) en el bucle morado (Bucle TΨCG), entre otras bases modificadas. El anticodón es la parte de abajo que se unirá a sus bases complementarias en el mRNA. [12] b) Se muestra un modelo en 3D del tRNA. Hecha en JSMol. [13]

Para que se lleve a cabo la traducción del lenguaje de cuatro bases del DNA y mRNA al lenguaje de los 20 aminoácidos a parte de los RNA mencionados, se necesita las enzimas denominadas aminoacil-tRNA sintetasas. El tRNA participa en este proceso, estando unido a un aminoácido en particular a través de un enlace de alta energía, a esta conformación se llama aminoacil-tRNA, para que el aminoácido que contiene esta unión se active y se una a la cadena polipeptídica creciente, el anticodón en el tRNA deberá aparearse con un codón del mRNA. Se han identificado alrededor de 30 a 40 tRNA en las células bacterianas y cerca de 50 hasta 100 en las células animales y vegetales, lo que permite concluir que el número de tRNA en la mayoría de las células es mayor al número de aminoácidos utilizados en la síntesis de proteínas en consecuencia un aminoácido tendrá más de un tRNA al que pueda fijarse.

Las moléculas de tRNA que poseen de 70 a 80 nucleótidos de largo dependen de la exactitud de sus estructuras tridimensionales, típicamente se pliegan en una

disposición de tallo con un bucle, como una hoja de trébol. Sus cuatro tallos son hélices dobles y cortas, mantienen esta disposición por el apareamiento de pares de bases, de estos tres tienen bucles, de forma que el tallo sin bucle contienen los extremos 3’ y 5’ de la cadena. El anticodón estará en una posición accesible para el apareamiento, esto es en el centro de un bucle.

Además de esta disposición estructural de los tRNA, todos tienen en el extremo 3’ del tallo aceptor la secuencia CCA, la cual se adiciona al terminar la síntesis y procesamiento de tRNA, incluso ocurren ciertas modificaciones después de su síntesis que también son estudiadas en la biología molecular. Si se viera la molécula plegada de tRNA en tres dimensiones, el tRNA tiene una forma de L con el bucle del anticodón y el tallo aceptor formando los extremos de los dos brazos. (Fig. 2.10)

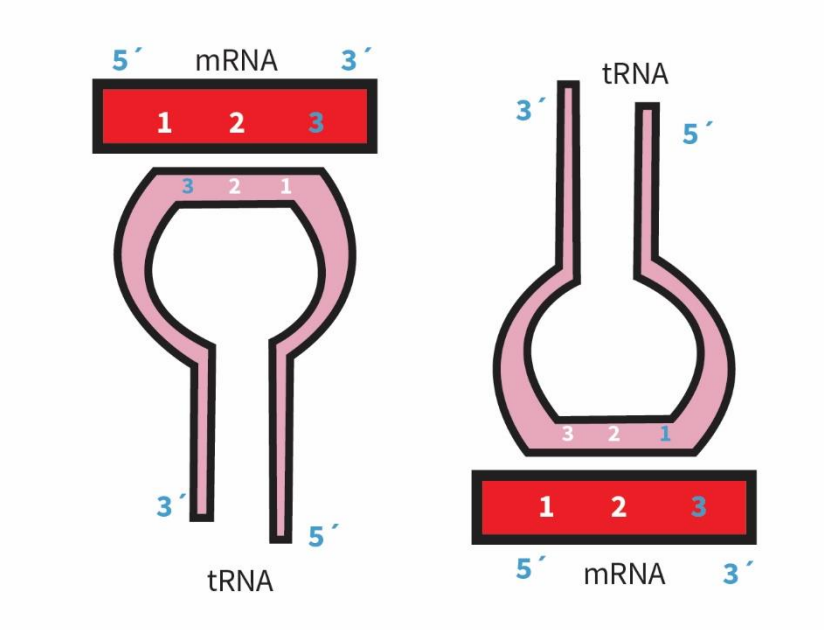
Si el apareamiento de bases entre codones y anticodones fuera perfecto, las células deberías tener exactamente 61 especies distintas de tRNA, es decir una por cada codón, sin embargo no es así, porque el número de tRNA es menor. La explicación de que el número de tRNA sea menor depende de la capacidad de un solo tRNA puede reconocer más de un codón correspondiente a un aminoácido, esto no ocurre con todos los tRNAs como los específicos para metionina y triptófano que sólo reconocen un anticodón específico.

El reconocimiento de más de un codón se debe a un apareamiento inusual entre la bases en la posición llamada de balanceo. La tercera base (3’) de un codón de mRNA y la primera base (5’) del anticodón del tRNA, en otras palabras, la primera y la segunda base de un codón en el mRNA forma un par de base estándar Watson y Crick con la tercera y segunda base de un anticodón en el tRNA (siguiendo la orientación 5’ -> 3’), pero en la posición de balanceo pueden ocurrir cuatro interacciones no estándar.

En particular cabe destacar el par de bases GU que es muy similar estructuralmente con el GC, entonces en la posición de balanceo en el tRNA cuando se presente G se podrá aparear con C o U en la tercera posición del codón en el mRNA. Por ejemplo, los codones que traducen la fenilalanina UUU y UUC son reconocidos por el anticodón del tRNA como GAA. Los codones del tipo NNPir, donde N es cualquier base y Pir es una pirimidina codifican un único aminoácido y son decodificados por un único tRNA con G en la primera posición del anticodón. Es raro que se encuentre una adenida en la primera posición de balanceo del anticodón, en muchas plantas y animales en esta posición contienen inosina (I), que químicamente hablando es un producto desaminado de la adenina.

En el caso de inosina, esta pueda formar un apareamiento no estándar con las bases A, C y U. Entonces un tRNA con inosina en la posición de balanceo puede reconocer los codones de mRNA que tengan en su tercera posición con A, C o U, esta es la principal razón que los tRNA con inosina tiene una gran aplicación en el proceso de la traducción de codones sinónimos que especifican un único aminoácido. Por dar otro ejemplo, la leucina que se codifica con los siguientes

cuatro codones de los seis: CUA, CUC, CUU y UUA son reconocidos por el mismo tRNA cuyo codón es GAI (posición 3’ -> 5’, también leída IAG de la orientación normal), debido a que la inosina en la posición de balanceo forma un apareamiento no estándar con la tercera base de los cuatro codones (Fig 2.11).



C	A	G	U
G I	U I	C U	A G I

C	A	G	U	I
G	U	U C	A G	C A U

Fig. 2.11 En los cuadros de arriba se muestran las combinaciones que se pueden presentar como apareamiento no estándar de bases entre el codón y el anticodón, como se mencionó la posición de balanceo se establece entre el nucleótido 3 del mRNA y el 1 el tRNA. En el caso del anticodón del tRNA, la inosina puede formar pares de bases con C, A y U, así como la U y G puede aparearse con dos bases diferentes. La teoría dice que es posible en encontrar A en la posición de balanceo aunque su presencia en la naturaleza es muy improbable. [14]

Una vez estudiados los detalles para la conformación de la paridad de bases entre el tRNA y el mRNA, se procederá a la explicación de las etapas que se llevan a cabo para la traducción, es importante decir que la síntesis de proteínas y la traducción no es lo mismo, entiéndase mejor como un proceso necesario para la conformación de otro proceso. Es decir que para que exista la síntesis de proteína se necesita del proceso de la traducción.

El reconocimiento del codón en el mRNA por el tRNA es el segundo paso de la decodificación del mensaje genético. El primer paso que debe de ocurrir antes de este reconocimiento, es la fijación de un aminoácido específico al tRNA, mediante el trabajo de una aminoacil-tRNA sintetasa. Cada una de las 20 sintetasas diferentes reconoce un aminoácido así como todas las moléculas de tRNA

análogas. Lo que hacen estas enzimas acoplantes es unir un aminoácido al hidroxilo 2’ o 3’ libre de la adenosina en extremo 3’ terminal del tRNA (como se mencionó anteriormente el extremo 3’, tiene una terminación CCA) mediante una reacción que requiere ATP.

En algunas ocasiones las aminoacil-tRNA sintetasas pueden llegar a producir errores debido a la similitud de los aminoácidos, para corregir estos posibles errores las mismas enzimas realizan una actividad de corrección llamada *proofreading*, esta función de las sintetasas ayuda a garantizar que un tRNA entregue el aminoácido correcto.

Para una eficiente traducción del mRNA, el complejo RNA-proteína más abúndate en las células, el ribosoma participa en este proceso, de cualquier otra forma la velocidad de polimerización de aminoácidos sería muy baja. El ribosoma dirige la elongación de un polipéptido a una velocidad de tres a cinco aminoácidos agregados por segundo. Esto también permitirá concluir que; una proteína con menor número de aminoácidos se sintetizará más rápido que una que tenga mayor número de aminoácidos, por ejemplo la titina que tiene aproximadamente 30,000 residuos tarda mucho más tiempo en su síntesis que otras proteínas que tienen entre 100 y 200 aminoácidos. Gracias a las técnicas como la radiomarcación in vitro y el uso del microscopio electrónico se llegó a entender más de la función del ribosoma en la síntesis de proteínas.

2.1.3.2.1 El ribosoma, estructura y función.

El ribosoma se compone de tres o cuatro moléculas de rRNA y un cúmulo de proteínas distribuidas en la subunidades mayor y menor tanto como estas y las moléculas de RNA su suelen designar con unidades de velocidad de sedimentación, Svedberg (S).

La subunidad menor contiene una molécula única de RNA llamada rRNA pequeño y la subunidad mayor contiene una molécula de rRNA grande. Son características distinguibles de los subunidades en las células eucariontes y bacterianas, la longitud de las moléculas de rRNA, la cantidad de proteínas en cada subunidad y tamaños de las mismas (Fig 2.12). Actualmente se conocen las secuencias de rRNA grandes y pequeños de miles de organismos y a pesar de las variaciones de los nucleótidos se tienen estructuras similares de rRNA de tallos y bucles.

Los ribosomas tienen 3 sitios de unión llamados por letras A, E, y P, en el sitio A se une el aminoacil-tRNA, en el sitio P es donde ser formará la cadena creciente que más adelante será la proteína y el sitio E será el sitio de salida de los tRNA.

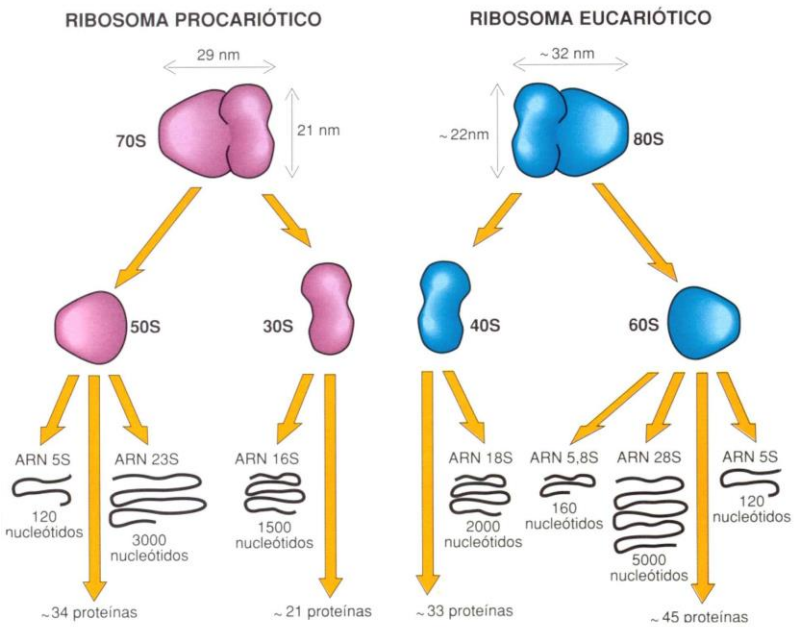


Fig. 2.12 Las estructuras generales de los ribosomas en células eucariontes y procariontes, se muestran las dos subunidades que conforman al ribosoma así como las cadenas de rRNA con su longitud correspondiente en nucleótidos y adicionalmente se muestran sus equivalentes en unidades Svedberg(S). [15]

Durante la traducción, un ribosoma se mueve a lo largo de una cadena de mRNA con la interacción de distintos factores proteicos y los tRNA, sufriendo algunos cambios conformacionales. La cristalografía de rayos X ha permitido saber no sólo las dimensiones y formas del ribosoma sino también conocer las posiciones exactas de unión con el tRNA durante la elongación.

2.1.3.2.2 La síntesis de proteínas.

Una vez conocidos los participantes involucrados en la traducción, es decir los diferentes tipos de RNA, también los tRNA aminoacilados y los estructuras complejas de los ribosomas, se verá con mayor detalle cada proceso bioquímico necesario para la síntesis de proteínas, al igual que el proceso de la transcripción se suele explicar el proceso de la traducción con tres etapas: iniciación, elongación y terminación.

El primer evento en la traducción es una identificación del codón de inicio que como se dijo anteriormente en la mayoría de los casos es AUG (metionina), esto con la intención de establecer el marco de lectura correcto, por consiguiente se vuelve un aspecto crítico. En las células eucariontes como procariontes existen dos metionina-tRNA diferentes: $tRNA_1^{Met}$ y $tRNA^{Met}$, el primero puede iniciar con la síntesis de

proteínas y el segundo solo se puede incorporar la metionina a la cadena polipeptídica como un aminoácido más.

La existencia de estos dos tRNA que tienen activados a la metionina, se debe a que la aminoacil-tRNA sintetasa (MetRS) carga ambas metioninas al tRNA, pero sólo $tRNA_1^{Met}$ es capaz de unirse al sitio apropiado sobre la subunidad ribosómica menor, en el sitio P para comenzar la síntesis de una proteína, en cambio los demás tRNA cargados y la $tRNA^{Met}$ solo se unirán al sitio A.

Antes de iniciar el proceso de la traducción, tuvo que haberse formado un complejo de preiniciación, el cual está conformado por la subunidad menor del ribosoma unido a un conjunto de proteínas y el Met-$tRNA_1^{Met}$ con un GTP. Se puede inhibir la síntesis de proteínas a través de la hidrólisis del GTP, debido a que, por si solo a este complejo le es imposible intercambiar el GDP por GTP. Si no está unida la molécula de GTP no se podrá unir a un $tRNA_1^{Met}$ dando como lugar la inhibición de todo el proceso.

Las etapas de la traducción son:

1) Iniciación: Un complejo proteico detecta y se une el casquete 5' del mRNA que será traducido, después se producirá la unión de estos dos últimos con el complejo de preiniciación, para formar el complejo de iniciación. Una vez realizado este paso, el complejo será capaz de deslizarse y rastrear la cadena de mRNA gracias a la energía que proporciona el ATP, así como desenrollar la estructura secundaria que pudiera tener el mRNA.
   El rastreo termina cuando el anticodón del $tRNA_1^{Met}$ encuentra el codón de inicio (AUG) en dirección rio abajo respecto a la orientación 5' a 3'.
   El reconocimiento del codón iniciador conduce la hidrólisis del GTP unido al complejo (proceso irreversible), una forma de facilitar el reconocimiento correcto del codón iniciador es mediante la presencia circundante de nucleótidos específicos llamados la secuencia de Kozak, en honor a Marilyn Kozak quien la definió así: **(5')ACCAUGG(3')** .

   Una vez producida la unión correcta del anticodón del $tRNA_1^{Met}$ con el codón de inicio, se produce la unión de la subunidad ribosoma mayor con la menor, dando como resultado, la unidad ribosómica completa , se necesita de la acción del factor eIF5 y la hidrólisis de un GTP asociado, este proceso también es irreversible, las subunidades no se separarán hasta que haya finalizado toda la traducción del mRNA, más adelante se verá porque se mantiene fija la cadena polipeptídica creciente en la epata de elongación (Fig. 2.13)

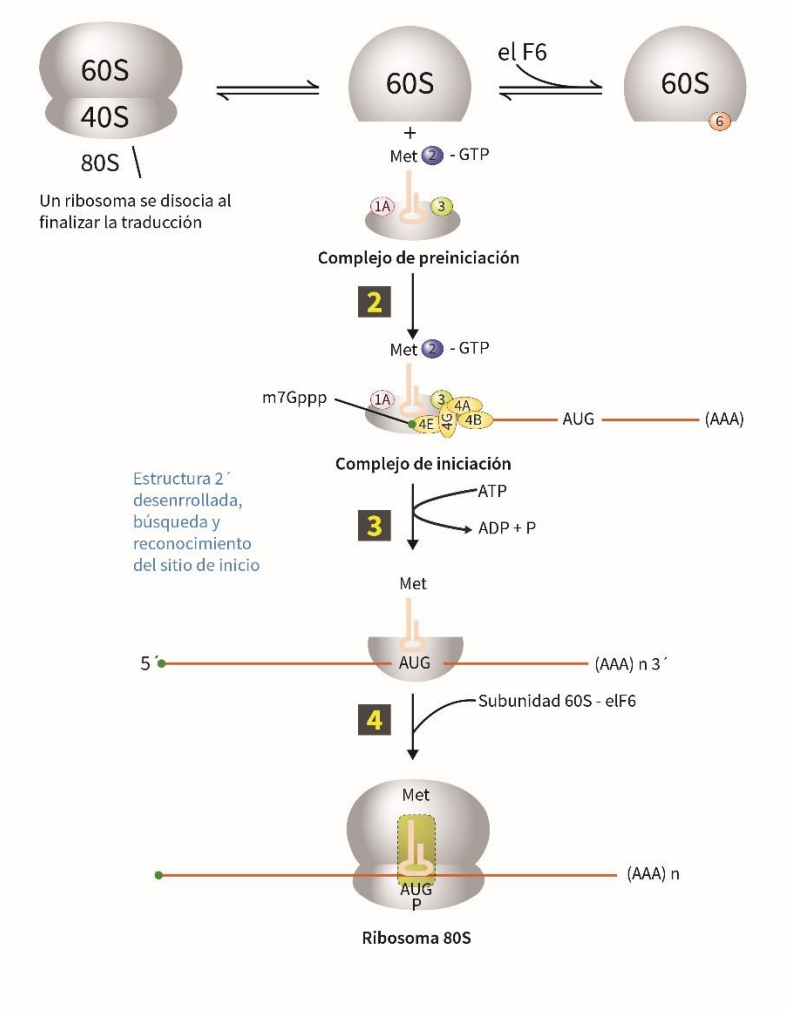


Fig. 2.13 En la etapa de iniciación, un ribosoma disociado en sus dos subunidades se prepara para la traducción del mRNA, posteriormente se une un aminoacil-$tRNA_1^{Met}$ con el factor eIF1A y el complejo eIF2-GTP a la subunidad menor. Después se une un complejo de proteínas eIF4 con el mRNA (unión a través de eIF4E-casquete 5' del mRNA), para formar mediante la unión de la subunidad eIF4G con eIF3 un complejo de iniciación. Gracias al reconocimiento del codón de inicio, se lleva a cabo un proceso irreversible de hidrolización del GTP, liberando de esta forma las proteínas asociadas al complejo de iniciación, este proceso también requiere de ATP y su descomposición en ADP y un fosfato. Una vez que el aminoacil-$tRNA_1^{Met}$ y codón de inicio forman un apareamiento, la subunidad mayor con el factor eIF6 y la ayuda de un complejo eIF5-GTP formaran una unión entre subunidades, dando lugar a un ribosoma con el codón mRNA apareado con un anticodón de un aminoacil-tRNA, específicamente del aminoácido metionina (la mayoría de las veces). [16]

2) Elongación: Una vez construido el complejo completo ribosómico con el $tRNA_1^{Met}$ y el codón de inicio, el proceso siguiente es de la elongación, donde se agrega aminoácido por aminoácido a la cadena polipéptidica dependiendo de lo que establezca el marco de lectura del mRNA. Al igual que la iniciación se requieren de proteínas especiales en este proceso llamadas factores de

elongación (EF), que participarán en tres pasos claves: la entrada de cada aminoacil-tRNA, la formación del enlace peptídico y el movimiento de translocación del ribosoma de codón en codón.

Cuando el Met-tRNA$_1^{Met}$ está ubicado en el sitio P (sitio de unión) del ribosoma recién ensamblado, el segundo aminoacil-tRNA entra al ribosoma como un complejo ternario con EF1α-GTP y se une al sitio A. Si se forman un par de base correcto entre el codón del mRNA en esa zona y el anticodón del aminoacil-tRNA, el GTP se hidroliza del EF1 α-GTP, este a su vez provocará un cambio conformacional en el ribosoma, estableciendo una unión fuerte entre el aminoacil-tRNA y el sitio A, así como la liberación del complejo EF1 α-GDP resultante, también se aproxima el extremo 3' aminoacilado del tRNA con el extremo 3' Met-tRNA$_1^{Met}$ en el sitio P.

Si no se llegara a producir la paridad entre el codón y el anticodón, no se podría realizar la hidrolización del GTP y por ende no se produce la unión fuerte ya mencionada. En este caso el complejo ternario se difunde y deja libre el sitio A, para que se una el aminoacil-tRNA adecuado después, esta acción contribuye a la fidelidad de la síntesis de una proteína.

Una vez que exista el Met-tRNA$_1^{Met}$ iniciador en el sitio P y un segundo aminoacil-tRNA perfectamente apareado en el sitio A, el grupo α-amino del segundo aminoácido reacciona con la metionina activada formando un enlace peptídico, para aumentar la velocidad de la formación de estos enlaces se lleva a cabo una reacción peptidiltransferasa y es catalizada por la subunidad mayor del ribosoma, permitiendo la orientación adecuada de los átomos para dichos enlaces.

Una vez producido un enlace péptido entre dos aminoácidos, el ribosoma se desplaza un codón, esta translocación es promovida por la hidrólisis del GTP en el EF2-GTP. Después el tRNA$_1^{Met}$ es desplazado al sitio E (de salida) sobre el ribosoma, en ese mismo instante el segundo tRNA unido por un enlace covalente a un dipéptido, llamado peptidil-tRNA se mueve al sitio P. Este proceso de translocación vuelve a cambiar la conformación inicial del ribosoma, en donde el sitio A esta disponible y es capaz de aceptar a otro tRNA con su complejo EF1 α-GTP.

La repetición de cada ciclo de elongación adiciona un aminoácido al extremo C-terminal de un polipéptido creciente, siguiendo el marco de lectura del mRNA hasta que se encuentra un codón de stop o de finalización, falta mencionar que los tRNA ubicados en el sitio E son expulsados en cada ciclo. En el orden en que la cadena polipéptidica va creciendo va siendo dirigida a

un canal la subunidad mayor del ribosoma cuya salida es un lado opuesto de la subunidad menor que interactú (Fig. 2.14).

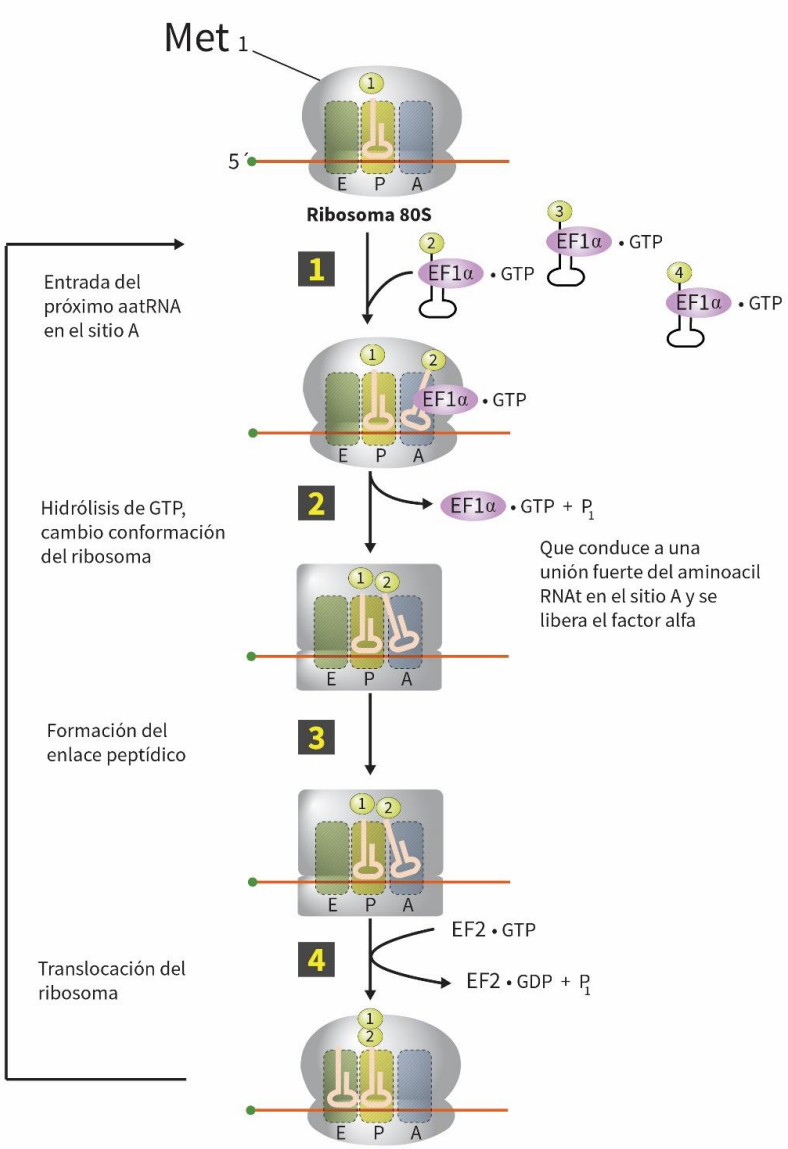


Fig. 2.14 El ciclo de elongación permite la adición de aminoácido por aminoácido en la cadena polipeptídica, una vez que en el sitio P se encuentra el amonoacil-tRNA$_1^{Met}$, un complejo terciario conformado por un aminoacil-TRNA y la unión de EF1α-GTP se une al sitio A, siempre y cuando exista complementariedad química. La hidrólisis del GTP permite la liberación de EF1α-GDP y un fosfato, así como un cambio conformacional del ribosoma que servirá para la translocación de codón en codón, antes de este movimiento, el rRNA mayor cataliza un enlace peptídico entre los dos aminoácidos, provocando que la cadena creciente se mantenga en el tRNA ubicado en el sitio A. Finalmente cuando el desplazamiento es producido, el sitio A queda libre para la llegada de un nuevo aminoacil-tRNA y el primer tRNA es expulsado posteriormente en el sitio E. [17]

3) Finalización: para esta última etapa existirán señales moleculares que decidirán el destino de todo el complejo de traducción, estas señales son llamadas factores de liberación (RF). Se han estudiado dos tipos de factores de liberación: el eRF1 eucarionte y el eRF3.

El eRF1 es similar en estructura al tRNA y reconoce el codón de terminación en el sitio A y el eRF3 es una proteína de unión al GTP, de tal manera que ambas proteínas promueven la ruptura del peptidil-tRNA, liberando de esta forma la cadena proteica completa.

Después de la liberación de la cadena polipéptidica, se pliega en su conformación tridimensional nativa, este proceso es facilitado por las proteínas conocidas como chaperonas. Los factores de liberación también promueven la liberación del ribosoma en sus subunidades, así como la liberación del mRNA y el tRNA terminal. Cabe mencionar que en el momento que realiza la liberación de la cadena polipéptidica ya se puede llamar proteína (Fig 2.15).

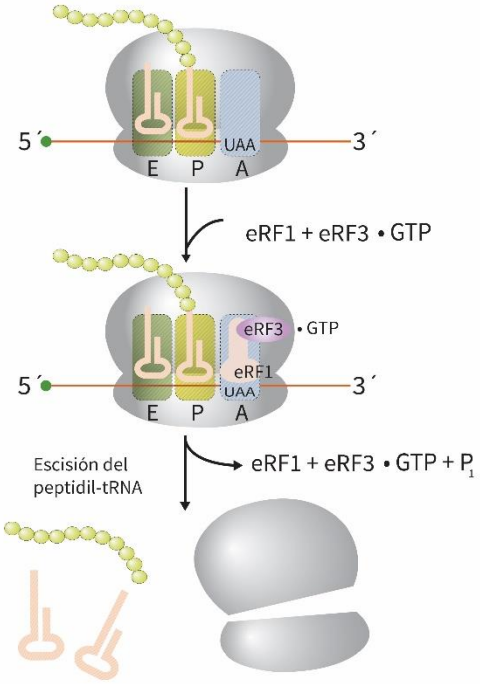


Fig. 2.15 Cuando se identifica un codón de terminación por los factores de terminación (eRF1 + Erf3-GTP), se hidroliza el GTP por GDP y un fosfato provocando la rotura del peptidil-tRNA y la separación de las unidades ribosómicas y el mRNA ya traducido. De esta forma se puede volver a traducir el mismo mRNA u otro. [18]

Como se puede observar a lo largo de todo ese proceso biológico, se necesita la intervención de varias proteínas de unión al GTP, estas proteínas pertenecen a la superfamilia de las GTPasas que actúan como interruptores en la presencia del GTP (activadas) o la ausencia del mismo en forma con GDP (inactivas). Se piensa que la hidrólisis del GTP no sólo provoca un cambio conformacional a la GTPasa sino también a las proteínas asociadas en los procesos moleculares.

2.2 Proteínas.

Una vez estudiados los procesos que forman parte del dogma central de la biología, se iniciaría con el tema principal de esta tesis; las proteínas. Se requiere de la transcripción y de la traducción para la síntesis de proteínas, y como ya se mencionó las proteínas son las moléculas operadoras de la célula, toman un papel importante para la sobrevivencia y subsistencia a través de la reproducción celular. Además desarrollan todas las actividades de todo el programa codificado por los genes.

Las enzimas son un tipo de proteína que se encargan de catalizar las reacciones químicas intracelulares y extracelulares, con una frecuencia y especificidad precisa. Otras proteínas también son agrupadas como proteínas estructurales, las cuales ayudan a darle a la célula una rigidez estructural. También hay proteínas transportadoras que controlan el flujo de materiales a través de la membrana celular, las proteínas reguladoras, que actúan como sensores e interruptores para controlar la actividad de otras proteínas y la función de los genes, las proteínas señalizadoras como las que están en la superficie celular y actúan como receptoras o bien las que transmiten señales del exterior al interior de la célula. Y finalmente las proteínas motoras que producen el movimiento celular.

Como se puede observar, las múltiples funciones que desarrollan las proteínas son tan variadas que, no depender de ellas sería un suicidio para la célula. Las proteínas tienen un diseño funcional muchas de las cuales tienen partes móviles, las cuales son capaces de transmitir fuerzas y utilizar la energía del medio de forma ordenada, sin embargo, para procesos más complejos como la traducción, transcripción, la fotosíntesis y los mecanismos de transducción de señales en los organismos se necesitan enormes conglomerados macromoleculares llamadas máquinas moleculares.

Los biólogos moleculares y genetistas se han encargado de hacer una lista de las partes que componen dichas maquinarias, gracias a la secuenciación masiva de organismos se ha logrado a pasos agigantados el conocimiento de esta parte de la biología, también se han hecho listas de todas las proteínas que conforman un organismo llamadas proteomas. La bioinformática y el súper cómputo han permitido que las secuencias de organismos se obtengan de manera más rápida y además de predecir el número y estructura primaria de proteínas codificadas.

Será de interés de esta tesis, el proteoma de la levadura *Saccharomyces cerevisiae* puesto que se diseñará la base de datos para más de 6,000 proteínas que conforman su proteoma. Sin embargo, el proteoma del ser humano es mucho más grande que la levadura, aproximadamente cinco veces más. Al comparar los proteomas, se pueden clasificar las proteínas de un organismo y deducir sus funciones por homología con proteínas conocidas. Incluso la función de una proteína puede ser deducida por las interacciones con otras proteínas o bien de la bioquímica del complejo al que pertenece.

En este subtema, se hablará de las estructuras moleculares de una proteína y sus funciones que desempaña en disposición de su estructura. También aquellas proteínas que contribuyen al plegamiento de las demás proteínas y las modificaciones que tienen lugar después de la síntesis de la cadena proteica llamadas modificaciones postraduccionales y finalmente de los mecanismos de degradación.

2.2.1 Estructura de las proteínas.

Las proteínas se pliegan adquiriendo una forma única, estabilizada por los enlaces no covalentes entre regiones de la secuencia lineal de aminoácidos. La organización especial de la proteína será clave para el entendimiento de su función, de forma que cuando una proteína se encuentra en su estructura tridimensional correcta será capaz de funcionar con eficiencia y bien se puede definir así:

"La función deriva de la estructura tridimensional y la estructura tridimensional está establecida por la secuencia de aminoácidos." [19]

La estructura general de una proteína se conforma por cuatro niveles de organización, empezando con sus monómeros que la constituyen; los aminoácidos.

a) Estructura primaria: Como se mencionó al inicio de este tema, en la parte de las unidades estructurales químicas de la célula, una proteína es una cadena polipeptídica mediante la unión de los grupos N amidos, los carbonos α y los grupos carboxilos de los aminoácidos que la conforman, donde se proyectan los distintos grupos de cadenas laterales. Debido al establecimientos de los enlaces peptídicos, la cadena principal exhibe una polaridad porque todos los grupos aminos se ubican siempre del mismo lado de los átomos de carbono α, entonces unos de los extremos de una proteína tendrá un grupo amino libre (N-terminal) y otro en el extremo un grupo carboxilo libre (C-terminal). (Fig. 2.16)

   Por convención la secuencia de una cadena polipeptídica inicia con un aminoácido N-terminal hasta un aminoácido C-terminal (de izquierda a derecha respectivamente). La estructura primara de una proteína es la

disposición lineal de los residuos o comúnmente llamada secuencia, de los aminoácidos que la componen.

Las cadenas cortas bien definidas entre la unión de varios aminoácidos se le llama péptido y en consecuencia las cadenas más largas se llaman polipéptidos y como es notable se emplea el termino polipéptido para hacer referencia a la proteína. El tamaño de una proteína se da en su peso molecular o en su masa (daltons [Da]), en el caso de las proteínas de la levadura, el peso molecular promedio es de 50.8 kDa y contienen en promedio 456.67 residuos de aminoácidos. [20]

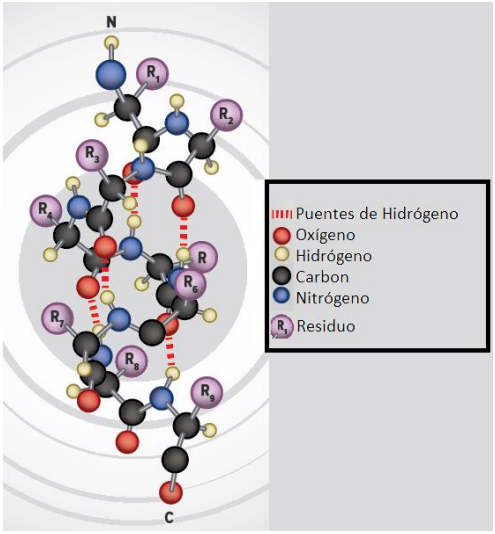


Fig. 2.16. Estructura primera de una proteína formada por la unión de los aminoácidos mediante los enlaces péptidos, los residuos se proyectan hacia al exterior y se puede observar que van desde una región N terminal hasta una C terminal.

b) Estructura secundaria: este segundo nivel de jerarquía consiste en la diversidad de disposiciones espaciales que una proteína puede tener. Estas disposiciones resultan del plegado de partes específicas de una cadena polipeptídica, una proteína puede exhibir múltiples tipos de estructuras secundarias en función de su secuencia. Si no existen las interacciones estabilizadores no covalentes, la proteína adquiere estructuras conocidas como enrollamientos al azar, en caso contrario sí se establecen puentes de hidrógeno estabilizadores, las proteínas pueden formar estructuras periódicas bien definidas como: hélices alfa (α), hojas beta (β) y giros cortos en forma de U.

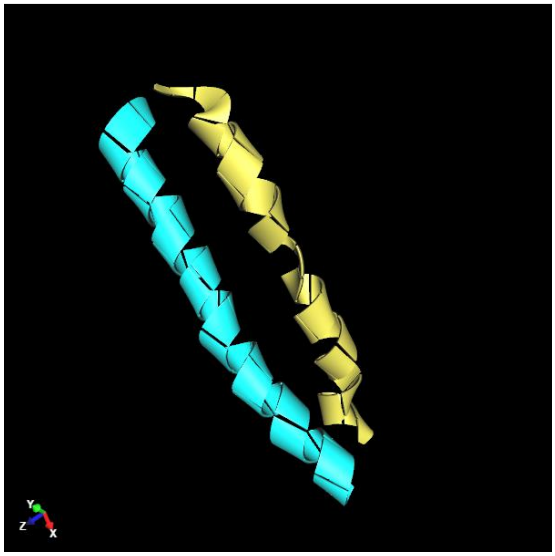
   Estudios revelan que el 60% de una proteína se encuentra en forma de hélices α y hojas β, el resto son enrollamientos y plegamientos al azar por lo tanto se dice que las hélices α y hojas β, son las estructuras de sostén en las proteínas (Fig. 2.17).

I) Hélice α: Es un segmento bien definido considerado estructura secundaria, donde el átomo de oxigeno del carbonilo de cada enlace peptídico se une mediante un puente de hidrógeno al átomo de hidrogeno amido del aminoácido ubicado cuatro residuos después (dirección al C terminal). Esta estructura secundaria además de dar forma, proporciona cierta polaridad, la razón es que los donantes de los puentes de hidrógeno no tiene la misma orientación. La cadena principal se mantiene como un cilindro parecido a un bastón, y las cadenas laterales apuntan hacia el exterior, estos además determinarán la cualidad hidrófoba de la hélice.

II) Hoja β: esta estructura está compuesta por hebras β agrupadas lateralmente. Estas últimas hebras son segmentos cortos de alrededor de cinco a ocho residuos, la unión entre hebras β adyacentes es mediante puentes de hidrógeno ya sea dentro de la misma cadena polipeptídica o entre otras cadenas polipeptídicas diferentes. Esta unión dará como resultado la hoja. La planaridad del enlace peptídico fuerza a la hoja β a plegarse, de ahí que también reciba el nombre de hoja β plegada o hoja plegada. Poseen de una polaridad definida esto debido a la orientación de su enlace peptídico, por lo tanto las hebras adyacentes pueden estar en disposición antiparalela o paralela respecto a la otra. En algunas proteínas forman con las hojas β, la base de un bolsillo de unión.

III) Giros: se ubican en la superficie de una proteína compuestos por tres o cuatro residuos, formando plegamiento agudos que vuelven a dirigirse al interior de la cadena principal. Al igual que los casos anteriores, estas estructuras secundarias se estabilizan mediante puentes de hidrógeno entre los residuos terminales, adquiriendo una forma de U. Generalmente en estos giros están involucradas la glicina y la prolina. Estos giros le permiten a la proteína plegarse en estructuras muy compactas, también pueden tener curvaturas más largas llamadas bucles. Estos últimos exhiben formas muy variadas a diferencia de los giros.

a)



b)

c)

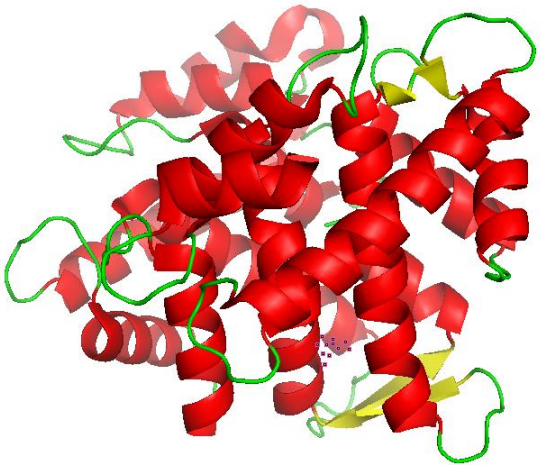


Fig. 2.17. En las imágenes de arriba se muestran las estructuras secundarias de las proteínas, a) las hélices α marcadas con colores azul y amarillo, imagen realizada en ChemDoodle Web Componentes [21], b) se marcan dos hojas β de lo color rojo, imagen realizada en ChemDoodle Web Componentes [22] y finalmente c) se puede ver los giros de color verde y algunas hélices α y hojas β, esta imagen fue realizada con PyMol. [23]

c) Estructura terciaria: referida a la conformación global de la proteína o bien su disposición tridimensional de todos sus residuos de aminoácidos, estas estructuras no se estabilizan mediante puentes de hidrógeno sino entre interacciones hidrófobas entre las cadenas laterales polares y enlaces peptídicos, además de compactar las estructuras secundarias ya mencionadas. Las interacciones hidrófobas al ser enlaces químicos débiles, hacen que las estructuras terciaras de las proteínas no se mantengan fijas, sino que sufren fluctuaciones continuas y momentáneas. Esta característica es muy importante ya que determinará el funcionamiento y regulación de las proteínas.

Existen diversas formas de representar la estructura tridimensional de una proteína (Fig 2.18), estas son:

a) Trazar el curso de los átomos de la cadena principal mediante una línea sólida.
b) Trazar la localización de cada átomo mediante esferas y bastones.
c) Utilizar símbolos taquigráficos comunes para representar la estructura secundaria, como por ejemplo cintas enrolladas o cilindros para las hélices α y cintas planas o flechas para las hojas β y finalmente hebras delgadas y flexibles para representar los bucles o los giros.

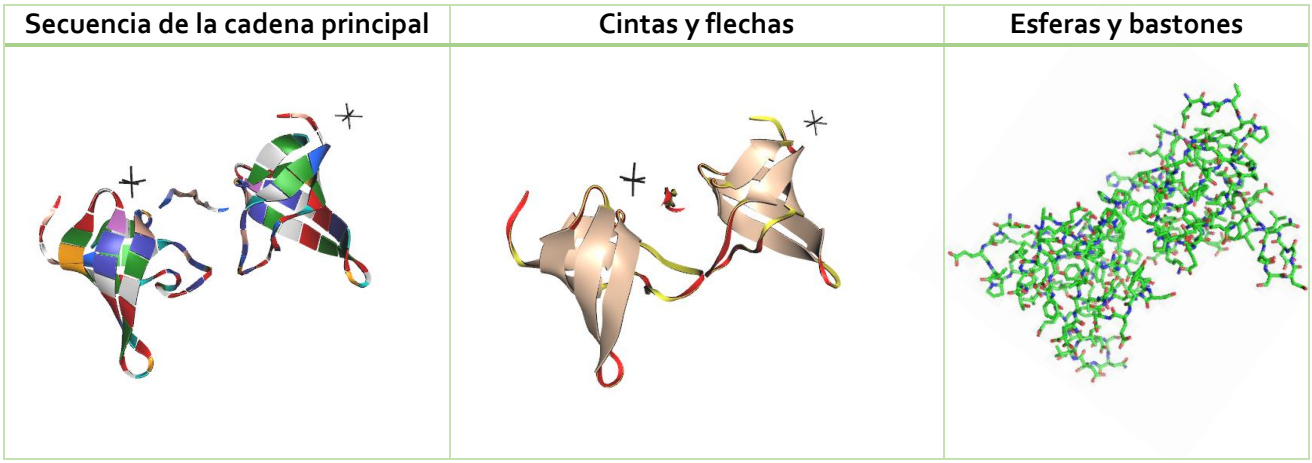


Fig. 2.18. Muestras de las diferentes representaciones con la misma proteína, la representación de la cadena principal del lado izquierdo, donde los aminoácidos adquieren un color en especial y la imagen del centro es la representación con cintas y flechas, ambas imágenes fueron hechas en ChemDoodle Web Componentes y finalmente la representación de esferas y bastones, esta imagen fue hecha en PyMol. [24]

La imagenología y el cómputo han permitido describir más características que las diversas formas de presentación de una proteína ya mencionadas,

como por ejemplo configurar un mapa de las regiones que tienen ciertas características químicas, como la hidrofobicidad, acidez o basicidad, además de revelar la topografía de la superficie proteica y la distribución de carga.

Existen combinaciones particulares de estructuras secundarias llamadas motivos o plegamientos, estos construirán las estructuras terciarias de una proteína. Las topologías particulares de motivos pueden determinar una función en particular, por ejemplo el motivo hélice-bucle-hélice se une con Ca^{+2}, que tiene ciertos residuos hidrófilos en posiciones invariables en el bucle, en donde sus átomos de oxígeno se unen al ion calcio mediante puentes de hidrógeno.

Este motivo también es llamado mano EF, se ha encontrado en más de 100 proteínas de unión de calcio. Otro motivo muy conocido son los dedos de cinc, es muy común encontrarlo en las proteínas de unión al DNA o al RNA, posee tres estructuras secundarias (una hélice α y dos hebras β con orientación antiparalela) formando un haz que se mantiene unido mediante un ion zinc. Muchas proteínas (casi siempre las fibrosas) también ocupan un tercer motivo conocido como la espiral enrollada para asociarse mutuamente en oligómeros. La formación de dicho motivo ha sido estudiado a detalle debido al carácter antipático de algunas hélices α que participan en su estructura (Fig. 2.19). Existen otros cientos más de motivos y en la actualidad las proteínas se pueden clasificadas de acuerdo a ellos.

En proteínas de peso molecular mayores a 15,000 sus estructuras terciarias están subdividas en regiones denominadas dominios. Un dominio es considerado una zona de un polipéptido plegada en forma compacta, mediante la difracción por rayos X y la cristalografía se han podido reconocer alguno de ellos. Éstos están conformados por diversos motivos y generalmente suelen tener la abundancia de un aminoácido en especial o un motivo en particular.

Existen diversas formas de clasificar los dominios, como por ejemplo dependiendo de su función (funcionales). Los dominios funcionales van dirigidos a la actividad de la proteína, logrando así identificar aquellos dominios relacionados a la actividad catalítica, a la capacidad de unión, etc.

De la misma forma que los motivos ayudan a determinar la estructura secundaria de una proteína, así los dominios para la estructura terciara.

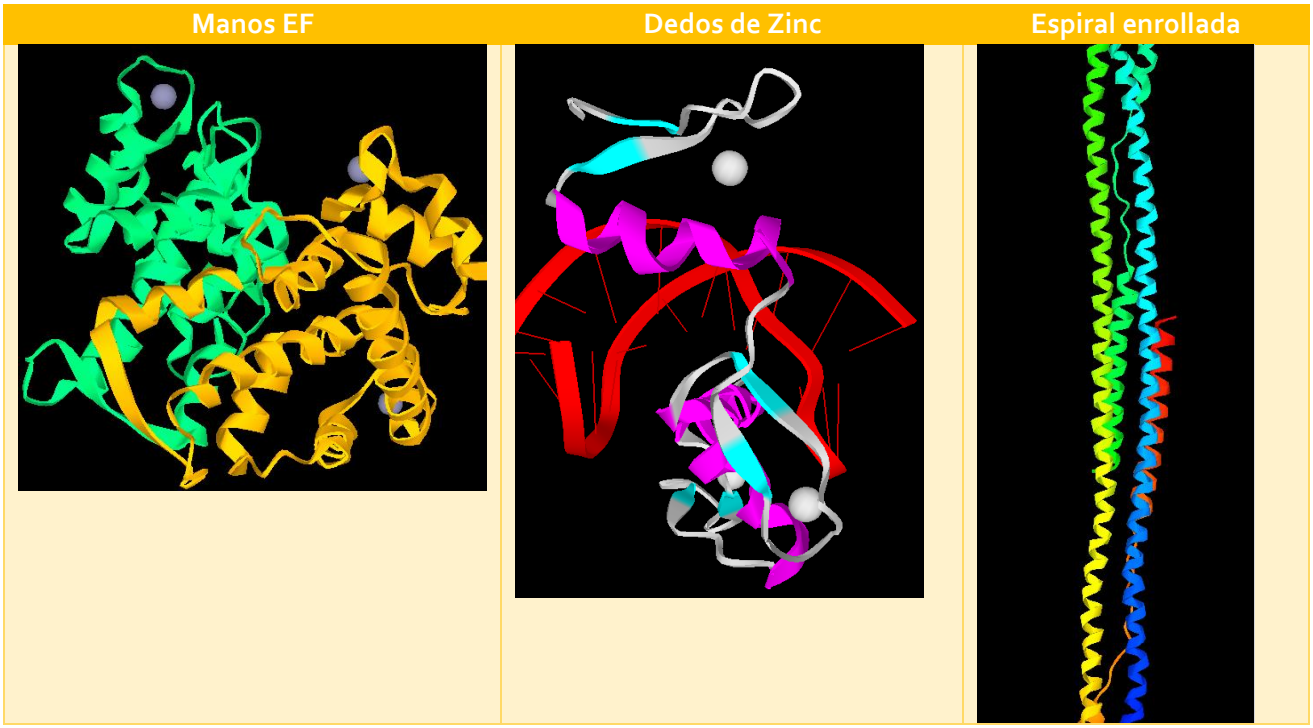


Fig. 2.19 Diferentes motivos se han caracterizado en la imágenes de arriba, del lado derecho se muestra el motivo mano EF, se pueden observar dos subunidades y las estructuras secundarias hélice, bucle y hélice además se resalta el ion Ca^{2+} como una esfera. [25] La imagen del centro es el motivo dedos de zinc, al igual que la imagen anterior se resalta los iones de zinc y se puede observar la estructura hélice alfa y dos hojas betas antiparalelas. [26] Y finalmente la espiral enrollada. [27] Imágenes hechas en NDKmol para Android.

Estructura cuaternaria: es el conjunto de proteínas multiméricas (dos o más polipéptidos) que tienen un número específico de proteínas asociadas y posiciones relativas conocidas, como por ejemplo: la hemoglobina, una proteína conformada por tres subunidades idénticas. Esta característica de multiplicidad de las proteínas es un mecanismo importante para regular sus funciones.

d) Macromoléculas: Finalmente el nivel más alto de organización de las proteínas son las asociaciones entre ellas para formar macromoléculas. Cuyo funcionamiento es muy variado, desde llevar acabo procesos celulares coordinados hasta participar en la forma de la membrana plasmática. Por ejemplo, las que participan en los procesos de transcripción y traducción ya mencionados, solo basta recordar la RNA polimerasa y los factores de transcripción.

2.2.2 Plegamiento, modificaciones y degradación de proteínas.

Una vez que la proteína es sintetizada en el proceso de traducción, sufre un proceso de plegamiento dependiendo de la cadena polipeptídica naciente, en muchas ocasiones esto implica la modificación de ciertos residuos para producirse la proteína final. También la célula promueve un proceso exclusivo de verificación de errores, el cual elimina aquellas proteínas que están plegadas de manera incorrecta o mal sintetizadas. La importancia de este control se debe a que aquellas proteínas que están plegadas de forma incorrecta no tienen actividad biológica, y en muchas ocasiones están relacionadas a enfermedades.

El plegamiento erróneo de las proteínas se suprime mediantes dos mecanismos diferenciados, el primero es el que provee la misma célula para reducir esta posibilidad y el segundo un sistema de desecho de basura que se encarga de degradar estas proteínas.

2.2.2.1 Plegamiento.

Una proteína de *n* residuos es capaz de plegarse 8^n conformaciones, esto debido a la estereoquímica de la cadena principal polipeptídica la cual permite sólo ocho ángulos de enlace, a pesar de estas posibilidades, una proteína adopta una conformación única llamada estado nativo. El estado nativo de una proteína es la forma plegada más estable en la que se puede encontrar. Diversos factores como el pH, la energía térmica del calor y entre otros, estabilizarán la conformación nativa de la proteína, la desnaturalización de estos factores dará como consecuencia una proteína sin actividad bilógica y la pérdida de su estado nativo.

Para acelerar los plegamientos de proteínas en forma precisa, se requiere de una vía auxiliar que las células tienen que recurrir, de lo contrario gastarían mucha energía, incluso necesitarían procesos de degradación para aquellas proteínas incorrectamente plegadas. Esta vía auxiliar es llevada a cabo por las chaperonas, las cuales son una clase de proteínas que participan en el plegamiento de proteínas. Existen dos familias generales de chaperonas:

- Chaperonas moleculares: se unen a las proteínas que están desplegadas o parcialmente plegadas, evitando que las éstas se agrupen y sean degradadas (Fig. 2.20).
- Chaperoninas: facilitan el trabajo de plegado de proteínas.

Generalmente estas chaperonas tienen un bolsillo hidrófobo expuesto que se une a regiones hidrófobas expuestas en la proteína diana desplegada y mediante la hidrólisis del ATP se acelera este procedimiento.

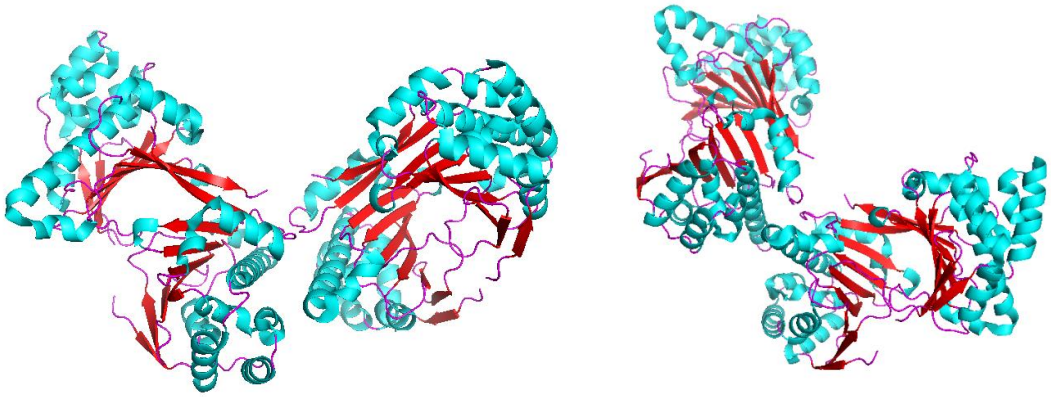


Fig. 2.20. Se muestra la chaperona HSP90 de la levadura, como se puede observar son dos subunidades exactamente iguales. [28]

2.2.2.2 Introducción a las modificaciones postraduccionales.

Una vez que la proteína es sintetizada y plegada, es modificada químicamente. Estas modificaciones pueden alterar la actividad, la vida media e incluso la localización celular de las proteínas, estas implican la unión de un grupo químico a los grupos libres $–NH_2$ o –COOH ya sea en los extremo de una proteína o en los residuos internos. Como se explicó anteriormente, en el proceso de la traducción se necesitan los 20 aminoácidos para formar una proteína, sin embargo ésta puede tener más de 100, esto se debe a la existencia de modificaciones postraduccionales.

Por ejemplo, la acetilación, que consiste en la adición de un grupo acetilo (CH_3CO) al grupo amino del residuo N-terminal, es una de las modificaciones más comunes, se dice que afecta al 80% de todas las proteínas. Esta modificación desempeña un papel importante en el control de la vida media de las proteínas en la célula, debido a que las proteínas no acetiladas son degradadas rápidamente. Otro ejemplo de otra modificación es la adición de grupos largos como los lípidos en los residuos ubicados en los extremos terminal de algunas proteínas, esta fijación de colas hidrófobas permite anclar a la proteína a la bicapa lipídica y es un mecanismo mediante cual la célula restringe ciertas proteínas a las membranas.

También el grupo acetilo y otros grupos químicos se puede adicionar a residuos internos específicos de la proteína. La fosforilación de residuos de serina, treonina, tirosina e histidina es muy importante, en la medicina principalmente en el estudio de la fisiología, muchas proteínas regulan su actividad siendo fosforiladas o desfosforiladas, e incluso esta modificación suele ser reversible. Otras cadenas laterales como la aspagarina, serina y treonina son sitios de glicosilación, donde se adicionan cadenas de hidratos de carbono y otras modificaciones postraduccionales como metilación, ubiquitinación, hidroxilación, etc. que se abordarán en el siguiente subcapítulo (Fig. 2.21) .

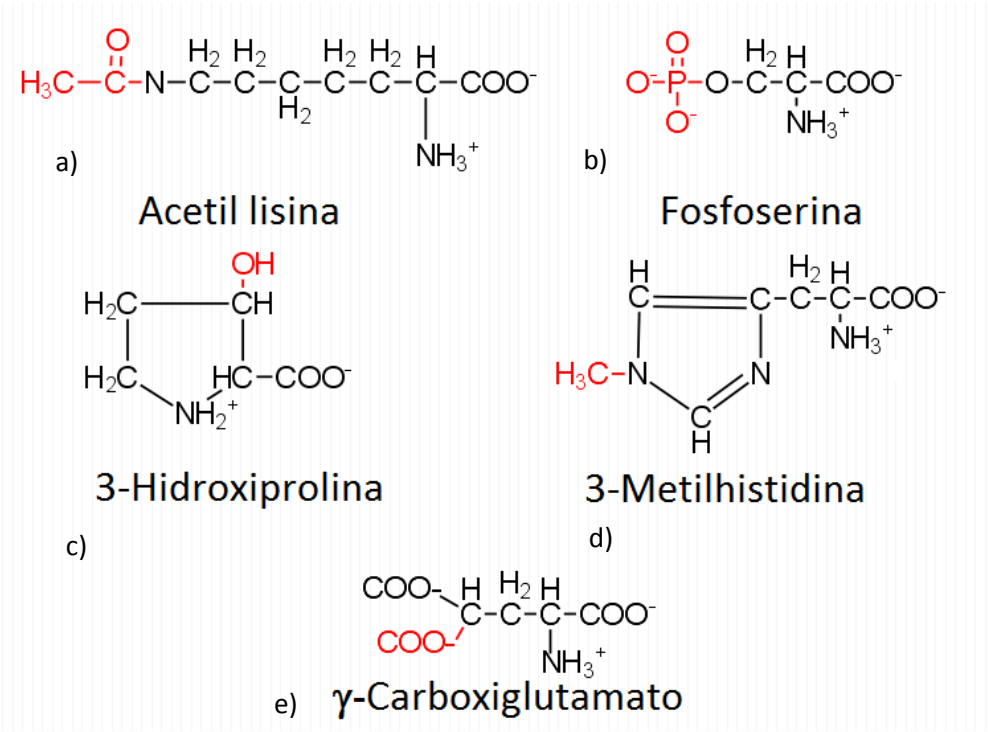


Fig. 2.21. Algunos ejemplos de modificaciones postraduccionales, se tiene la Acetil lisina que consiste en la adición de un grupo acetil en este caso de la lisina (a), la adición de un grupo fosfato al residuo de serina (b), así como la adición en el elemento número tres un grupo hidroxilo y un grupo metilo en la prolina (c) e histidina (d) respectivamente. Finalmente la adición de un grupo carboxilo a un glutamato (e).

2.2.2.3 Degradación.

Así como existen modificaciones reversibles, existe la degradación que es una modificación irreversible, no precisamente de modificar un residuo en particular, sino más bien como un procesamiento directo. La forma más común es la escisión enzimática mediante proteasas, que eliminan residuos del C o N terminales de una cadena polipeptídica.

La actividad de una proteína está regida por el balance entre la frecuencia de síntesis y la frecuencia de su degradación, en otras palabras de la cantidad presente de dicha proteína, la vida de las proteínas intracelulares es variable desde unos cuantos minutos hasta casi la edad del organismo que las posea.

Una de las principales vías de degradación en las células es mediante las enzimas que están ubicadas en lo lisosomas, organelos limitados por una membrana cuyo interior ácido contiene enzimas hidrolíticas. La degradación lisosómica está dirigida a las proteínas extracelulares tomadas por las células, orgánulos envejecidos o bien, defectuosos. Otra vía para la degradación de proteínas mal plegadas son mecanismos citosólicos, este incluye la modificación química de la cadena lateral

de lisina por la adicción de ubiquitina e inmediatamente después su degradación, más adelante se verá con mayor detalle esta modificación.

Aquellas proteínas degradadas por la vía mediada por la ubiquitina se dividen en dos categorías; proteínas citosólicas nativas cuya duración de vida es controlada y proteínas que se pliegan erróneamente, ambas son reconocidas y destruidas.

2.2.3 Enzimas y su función.

Lo que distingue a las proteínas de otras moléculas de la célula, es su capacidad para unirse con otras moléculas, desde iones simples hasta otras proteínas o maquinarias proteicas complejas y como se ha visto también los ácidos nucleicos. La función de la proteína dependerá en consecuencia de su habilidad para unirse a otras moléculas o ligandos, con su respectivo grado de especificidad.

Hablando específicamente de las enzimas, que son proteínas con actividad catalítica, deben de unirse primeramente a una molécula diana específica o una macromolécula, para desempeñar su tarea. Esta forma de trabajo, se ve reflejada en las proteinas de unión en la membrana, ya que son utilizadas en los mecanismos de transducción de señales. A menudo la unión de ligandos provocará un cambio conformacional de la proteína, por consiguiente estos ligandos serán fundamentales para la presencia de las funciones correctas de las proteínas.

Es necesario hablar dos características importantes en las proteínas, para definir la complementariedad molecular, estas son:

a) La especificidad: capacidad de una proteína de unirse a una molécula con mayor preferencia que otra.
b) La afinidad: la fuerza producida por la unión de la proteína con la molécula de mayor prioridad.

Ambas características dependen de la estructura del sitio de unión del ligando, que está diseñada como un molde para con la proteína. Se dice que la complementariedad, es la capacidad de la proteína de unirse al ligando, con el sitio de unión correspondiente, funcionando como una llave-cerradura. Esta complementariedad molecular garantiza una alta especificidad y una alta afinidad química.

Se puede alterar la proteína para que distinga otras moléculas diferentes, estos son los anticuerpos, y el organismo las utiliza la mayoría de las veces para detectar agentes infecciosos o sustancias extrañas. La presencia de un antígeno obliga al organismo a la elaboración de anticuerpos, para provocar la unión correspondiente y así realizar una serie de actividades que sirvan como proceso de defensa.

A diferencia de los anticuerpos, las enzimas promueven no solo la unión sino también la alteración química de sus ligandos, llamados sustratos. Toda reacción

química en la célula es catalizada por enzimas, las cuales tienen la función de incrementar la frecuencia de la reacción bajando la energía de activación. Lo que hace especiales a las enzimas es su enorme poder catalítico y su alta especificidad, estas características contribuyen que la frecuencia de las reacciones químicas incremente a $10^6 - 10^{12}$ veces más que las reacciones no catalizadas en condiciones similares.

En la actualidad se han clasificado en las bases de datos de enzimas alrededor de 3,700 tipos, cada una de las cuales cataliza una reacción química en especial. No solo contribuyen en la producción de proteínas, ácidos nucleicos o fosfolípidos sino también para la producción de energía de la glucosa y del oxígeno.

Las cadenas laterales de determinados aminoácidos de una enzima determinan su especificidad y su capacidad catalítica, estas cadenas laterales se aproximan los suficiente para formar un sitio activo, esto ocurre en la conformación nativa de la enzima. Los sitios activos se componen de dos regiones funcionales; la que reconoce y fija el sustrato y la otra que realiza la acción de catálisis. Puede que ambas regiones sean diferentes y estructuralmente bien definidas o que la región de unión este contenida en la región catalítica.

Para una mayor comprensión de las enzimas como proteinas catalizadoras, se utilizará el ejemplo de la proteincinasa dependiente del AMP cíclico llamada comúnmente proteincinasa A (PKA). En general este tipo de enzimas agregan un grupo fosfato a los residuos de serina, treonina y tirosina en las proteinas. En las células eucariontes, la estructura del sitio activo y los mecanismos de fosforilación son similares gracias a que las proteincinasas pertenecen a la misma superfamilia. Por ello la PKA se utiliza frecuentemente como modelo de enzima.

El sitio activo de la PKA esta en el centro, llamado núcleo de la cinasa, es donde ocurrirá la unión del sustrato (ATP y una secuencia peptídica diana), separado estructuralmente por un dominio grande y otro pequeño, en la base de la hendidura entre estos dos dominios existen un anillo de adenina y además una tapa de glicina cuya secuencia esta conservada (Gly-X-Gly-X-X-Gly-X-Val) la cual se cierra sobre el anillo.

La secuencia peptídica diana varía entre distintas cinasas, sin embargo el ATP siempre es común, para la PKA, la secuencia diana es Arg-Arg-X-Ser-Y, donde X es cualquier aminoácido y Y es un aminoácido hidrófobo. La unión ocurre cuando un residuo de serina o treonina diana se fija a la cavidad poco profunda del dominio grande, y éste gracias a su especificidad formará puentes salinos con los dos residuos de Arg de la secuencia diana. El núcleo catalítico del PKA tiene dos conformaciones una abierta y una cerrada, en la primera los dominios están separados para que se produzca la unión de los aminoácidos del sustrato, y una vez que éste se fije cambie la conformación a cerrada.

El cambio inducido a la estructura terciara de la proteína promueve que algunos residuos de la secuencia polipeptídica diana acepten un grupo fosfato del ATP unido, una vez producido este fenómeno, se produce la liberación de los productos (el ADP y el péptido fosforilado diana), ocurriendo de este modo la fosforilación de la PKA.

El trabajo y funcionamiento de la PKA es más complejo de lo que se ha mencionado, por lo que ha aquí solo se ha descrito de forma generalizada.

Las enzimas que son participes de procesos metabólicos comunes, como la degradación de la glucosa, se encuentra generalmente en el mismo comportamiento celular, los productos de una reacción química en particular pueden moverse mediante difusión hacia una enzima en la misma vía, pero en muchas ocasiones la difusión es demasiado lenta e ineficiente, por lo que las células han aproximado las enzimas dentro de una misma vía. Por ejemplo; se llegan a agrupar como subunidades para formar enzimas multiméricas o se ensamblan en modo de plataforma, en la levadura tres proteincinasas ensambladas en plataforma forman el complejo proteico Ste5.

En esta proteína, las cinasas se colocan en forma de cascada para transportar la señal de una a otra (Fig 2.22).

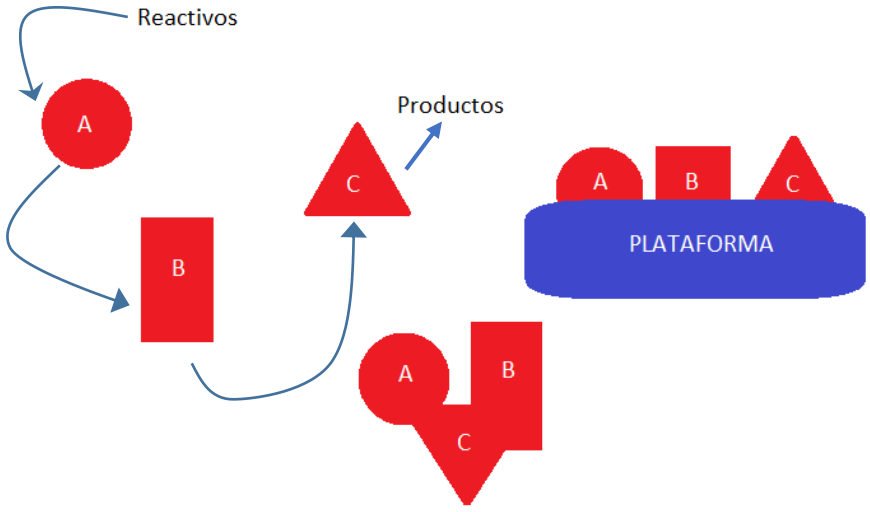


Fig. 2.22. Evolución de una enzima multifuncional, donde se muestran tres enzimas independientes A, B y C, donde se puede ver la reacción en cadena a partir de unos elementos reactivos hasta los productos pasando por todas las enzimas, como se mencionó se pueden encontrar enzimas unidas en forma de plataforma para facilitar la difusión de los sustratos, como el caso de la Ste5 en la levadura.

2.2.4 Motores moleculares.

Las células tienen la capacidad de moverse de un lugar a otro, exhiben esta habilidad tanto interior como exteriormente, este movimiento es el resultado del trabajo mecánico realizado por ciertas proteínas que funcionan como motores. El objetivo de este apartado será conocer la importancia de las proteínas como dadoras del movimiento celular y algunas propiedades importantes de destacar.

Como es bien sabido el movimiento es resultado de una acción física producida por una fuerza, las células obtienen esta fuerza mediante enzimas especializadas llamadas proteínas motoras, estas máquinas mecanoquímicas transforman la energía liberada por la hidrólisis del ATP o por gradientes de concentración de iones a fuerzas mecánicas. Se puede generar movimiento lineal o rotatorio, como por ejemplo las DNA y RNA polimerasas son proteinas motoras lineales porque se trasladan a lo largo de las cadenas de los ácidos nucleicos. Se puede deducir las siguientes características:

a) La transformación de energía química a energía mecánica, en este caso movimiento lineal o rotatorio.
b) La capacidad de unión y traslado en los filamentos del citoesqueleto o de las hebras de DNA u otros complejos proteicos.
c) El movimiento total a una dirección en particular, es decir el sentido.

Debido a que el movimiento está definido dependiendo del organismo y del proceso en particular que se esté estudiado, la siguiente tabla muestra algunos motores moleculares importantes, no es propósito de esta tesis profundizar sobre este tema, sino el conocer las distintas funciones que tiene las proteínas y una de ellas: la del movimiento **(Lodish, H. 2004, Molecular Cell Biology; Alberts, B., 2010, Biología Molecular de la Célula).**

Motores moleculares selectos				
Motor	Fuente de Energía	Estructura y componentes	Localización celular	Movimiento generado
Motores Lineales				
DNA polimerasa	ATP	Subunidades de polimerasa δ dentro del replisoma	Núcleo	Translocación a lo largo de la cadena de DNA en el proceso de replicación.
RNA polimerasa	ATP	Subunidades de polimerasa dentro del complejo de elongación en la transcripción	Núcleo	Translocación a lo largo de la cadena de RNA en el proceso de replicación.
Ribosoma	GTP	Factor de elongación con ribosoma	Citoplasma/ membrana RE	Translocación a lo largo del mRNA

				durante la traducción.
Miosinas	ATP	Cadenas pesadas y livianas; dominios cabeza con actividad ATPasa.	Citoplasma	Transporte de carga en vesículas; contracción muscular.
Cinesinas	ATP	Cadenas pesadas y livianas; dominios cabeza con actividad ATPasa.	Citoplasma	Transporte de carga en vesículas y en cromosomas durante la mitosis.
Dineínas	ATP	Múltiples cadenas pasadas, intermedias y livianas, dominios cabeza con actividad ATPasa y sitios de unión a microtúbulos.	Citoplasma	Transporte de carga en vesículas: golpeteo de los cilios y flagelos eucariontes.
Motores no Lineales				
ATP sintetasa	Gradiente H^+	Múltiples subunidades forman las partículas F_0 y F_1	Membrana mitocondrial interna	Rotación de la subunidad γ que resulta en la síntesis de ATP

Tabla 2.4 Se muestran algunos motores moleculares lineales y no lineales en donde las proteínas son una pieza fundamental.

2.3 Modificaciones postraduccionales.

Las modificaciones postraduccionales son mecanismos bioquímicos en algunos residuos en la secuencia de aminoácidos de una proteína que son covalentemente modificados. Las modificaciones postraduccionales cambian las propiedades de los aminoácidos en respuesta a las necesidades de la célula. Múltiples sitios que son modificados traerán como consecuencia un número de estados potenciales moleculares.

Se han identificado 450 modificaciones postraduccionales, la espectrometría de masas ha sido el instrumento para identificarlas. Un reciente estudio de SwissProt [28] menciona que se han realizado alrededor de 87,300 experimentos de modificaciones postraduccionales, y que la modificación que más prevalece es la fosforilación de residuos de serina y treonina, aunque esto realmente refleja que sean realizado más estudios de fosforilación.

Las bases de datos que recolectan información de modificaciones postraduccionales:

Base de datos	URL	PTMs	Organismo
UniProt (UNIPROT)	www.uniprot.org	Varias	Varios
HPRD	www.hprd.org	Fosforilación	Ser Humano

Phospho. ELM	phospo.elm.eu.org	Fosforilación	Células eucariontes
PhosphoSitePlus	www.phosphosite.org	Acetilación, metilación, ubiquitinación y fosforilación.	Ratón y ser humano
PHOSIDA	www.phosida.com	Acetilación y fosforilación	Varios
PhosphoPep	www.phosphopep.org	Fosforilación	Varios
dbPTM	dbptm.mbc.nctu.edu.tw	Varias	Varios
CPLA	cpla.biocuckoo.org	Acetilación	Varios
P3DB	www.p3db.org	Fosforilación	Varios
phosphoGRID	www.phosphogrid.org	Fosforilación	Levadura, otros
SGD	www.yeastgenome.org	Ubiquitinación, Acetilación, Succinilación y Fosforilación	Levadura

Tabla 2.5 Bases de datos importantes en el mundo de la biología molecular principalmente en el área de proteínas.

Las modificaciones postraduccionales pueden ser clasificadas en dos categorías: aquellas que están basadas en pequeños grupos (acetil, fosforil, etc.) y aquellos que están basados en pequeñas modificaciones de moléculas como la ubiquitina y sus parecidos (Fig 2.23).

En la siguiente tabla se muestran algunas modificaciones con sus donantes y modificadores así como los residuos que afectan:

Modificación	Modificador	Donante	Residuos
Fosforilación	PO_3^{2-}	ATP	S,T,Y
Acetilación	CH_3CO	AcCoA	K
Metilación	CH_3	SAM	K
Ubiquitinación	Ub, SUMO, etc.	-	K

Tabla 2.6 Principales modificaciones con los residuos que modifican

Las modificaciones postraduccionales forman una capa dinámica de eventos que se han ido estudiando lo largo de los años, en diferentes organismos intentando descubrir la existencia de reglas para la interpretación de toda esta dinámica. La visión a futuro es la creación de un código PTM, este código sería más complejo que el código genético, por la variabilidad y combinatoria de la secuencias de aminoácidos.

a)

b)

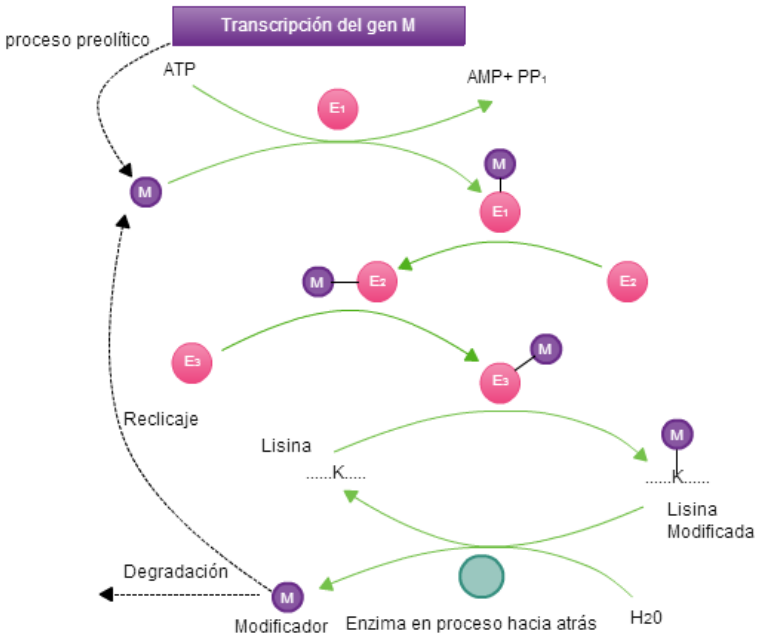


Fig. 2.23. En el esquema anterior se refleja las dos categorías de modificaciones postraduccionales; aquellas que están basadas en la adición de pequeños grupos químicos en los algunos residuos (a) mediante la ayuda de dos enzimas (una que realice el proceso y otro que lo regrese). Y aquellas que involucran modificaciones a pequeñas moléculas (b). [29]

2.3.1 Impacto en la señalización celular y mecanismos de transducción de señales.

Las modificaciones postraduccionales son cruciales para la regulación de funciones dentro de la célula, en la fosforilación los residuos modificados son serina, treonina y tirosina, la lisina en ubiquitinación (sumoilación y neddilación) y acetilación, lisina y argenina en metilación. Como se puede observar la lisina es el factor de modificación en varias modificaciones, pero estas modificaciones son mutuamente excluyentes en la misma lisina y genera un gran potencial de corregulación.

Las modificaciones postraduccionales afectan la estructura de la proteína y además actúan como swicthes moleculares, que regulan las interacciones entre ésta y el DNA, cofactores, lípidos y otras proteínas. La combinación y relación entre modificaciones postraduccionales ha dado como consecuencia hablar ahora de un PTM code, en donde se puede reconocer efectos de activación e inhibición, para inducir o retener señales en tiempo y espacio correcto.

Las células están expuestas constantemente a estímulos que provienen del ambiente y estos producen un cambio repentino en las condiciones anteriores en que se encontraba la célula. Todos estos cambios dentro y fuera de célula irán dirigidas al mantener la homeóstasis.

Se tiene el lado de la expresión génica y el control de la transcripción que es relativamente lento y del lado de las PTMs que es el mayor nivel de regulación en rapidez y la reversibilidad de sus procesos como la fosforilación o bien, lentos e irreversibles como la glicosilación. A través de las PTMs de una proteína muchas proteinas pueden ser generadas, compensando el numero bajo de genes codificantes y el numero grande de variables regulatorias que requiere un organismos complejo.

En el ser humano un 5% de su genoma codifica enzimas que están dedicadas a la realización de estas modificaciones y ellas mismas se autorregulan. A diferencia del control genético que sólo afecta a la abundancia de proteinas, las PTMs afectan las estructuras terciaras de las proteínas, activando o desactivando sitios activos y puertas de comunicación con otras proteínas.

Como se vio en el subtema pasado, las proteinas tienen una estructura terciaria y también algunos dominios a lo largo de su superficie o bien motivos, ambos participan en la comunicación intracelular, trasmitiendo, procesando y generando señales. Los motivos presentan secuencias moldeables que pueden ser leídas y procesadas por dominios. Finalmente los dominios, una vez que hayan reconocido, trasferido o removido una modificación postraduccional, terminarán de leer, borrar o escribir. Por ejemplo, en el caso de la fosforilación, las cinasas actúan como escritoras, mientras que los dominios que reconocen residuos específicos, como el

dominio SH2 al reconocer la tirosina fosforilada, es considera lectora, o las fosfatasas que son consideradas borradoras o eliminadoras.

Para que una proteína pueda actuar como lectora, eliminadora o escritora tiene que existir una modificación postraduccional antecesora o bien un patrón de PTMs. Por ejemplo la histona H3 cuando es acetilada en los residuos Lys9 y Lys14 puede reconocer dominios de bromo, o si es trimetilada en Lys9 dispara las interacciones con dominios de cromo de HP1. Estas interacciones pueden ser inhibidas fosforilando la Ser10.

Según varias referencias [30] una PTM puede ser clasificada en positiva o negativa:

- Forma positiva: La PTM inicial sirve como un disparador para la adición o eliminación de una segunda PTM o para reconocimiento de otras modificaciones, por ejemplo: una fosforilación que depende de una ubiquitinación y sumoilación.
- Forma negativa: Existe una competencia entre dos modificaciones postraduccionales en el mismo aminoácido o cuando hay efectos indirectos en donde una PTM marca un sitio reconocible para una segunda PTM, provocando un cambio conformacional evitando así la acción de la segunda PTM (Fig 2.24).

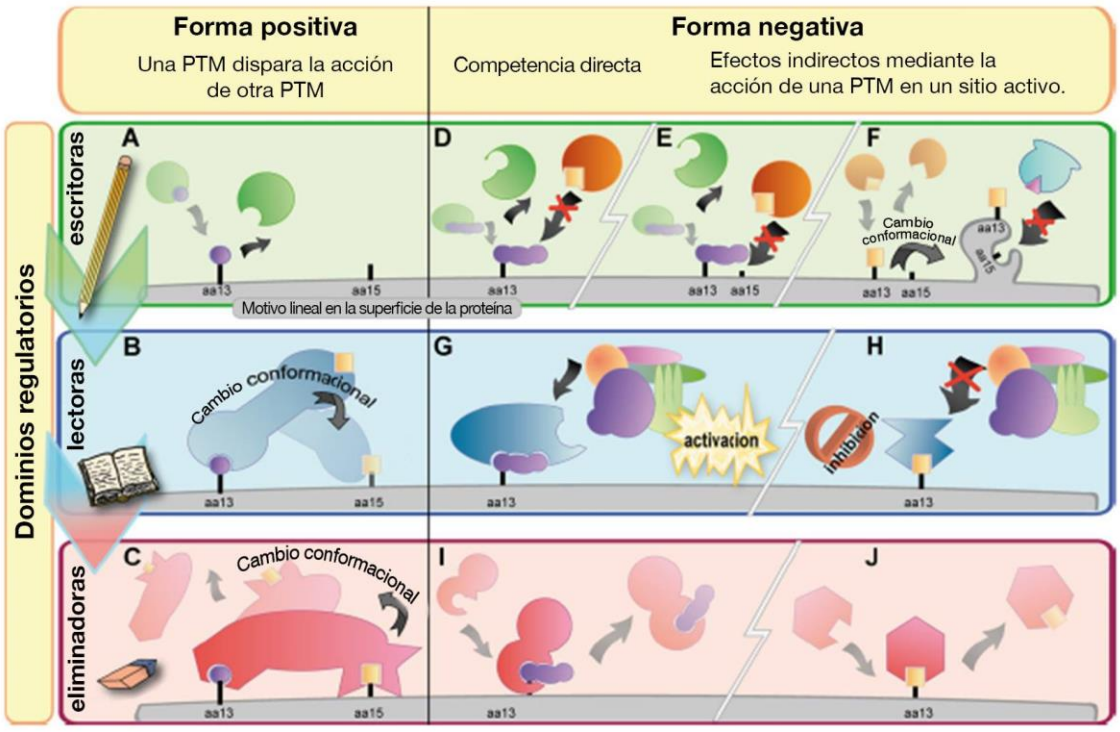


Fig. 2.24. Clasificación de las modificaciones postraduccionales relacionado a los mecanismos de traducción de señales, acompañado de las diferentes acciones que puede realizar un dominio regulatorio. [31]

Gracias a la hipótesis de Gerald Hart y sus colaboradores sobre la relación entre PTMs, llamada como el “Yin-Yang” en donde se describe la intercomunicación entre la fosforilación y la N-Acetilglucosamina, se puede ahora hablar un PTM code. Y no sólo este caso, sino también la palmitoilación en regulaciones de algunos canales iónicos o la fosfodegradación en la degradación de proteinas mediante la ubiquitina.

El descubrimiento de estas pequeñas y delicadas redes de PTMs, puede ser un avance importante en la medicina, ya que se pueden inducir señales artificiales y de esta manera promover otro tipo de señales que serán disparadas, que contribuirán sustancialmente en ciertos comportamientos del organismo. Esta tipo de estrategias revolucionarias pueden alterar de manera versátil el mecanismo celular desde un número limitado de genes hasta diferentes rutas y señales para transducción de señales.

Recientes estudios están concentrados en descifrar y describir un código PTM potencial y otro código PTM conservativo**. (Sudhakaran Prabakaran, Guy Lippens, Hanno Steen,Jeremy Gunawardena, 2012, Advanced Review Post-translational modification: nature’s escape from genetic imprisonment and the basis for dynamic information encoding)**

2.3.2 Tipos de modificaciones postraduccionales.

Se hablará de sólo cinco modificaciones de manera particular, ya que son las más estudiadas y de las demás se hablará de manera general. Se describirá que acciones se llevan a cabo para cada modificación así como su relevancia para el funcionamiento celular. Aunque existen muchas modificaciones, sólo 14 modificaciones han sido ampliamente descritas en levadura, de las cuales existen registros en las diferentes bases de datos y están respaldadas por varios artículos.

2.3.2.1 Fosforilación.

Es la modificación postraduccional más estudiada en el mundo de la biología molecular, debido a su gran presencia en casi todos los mecanismos de traducción de señales en los seres vivos. Consiste en la adición de un grupo fosfato en un aminoácido serina, treonina o tirosina en los grupos libres $–NH_2$ o –COOH, este fosfato es donado por el ATP la mayoría de las veces, o también puede ser una desfosforilación (la rotura de un grupo fosfato). Se dice que la modificación es reversible (Fig 2.25).

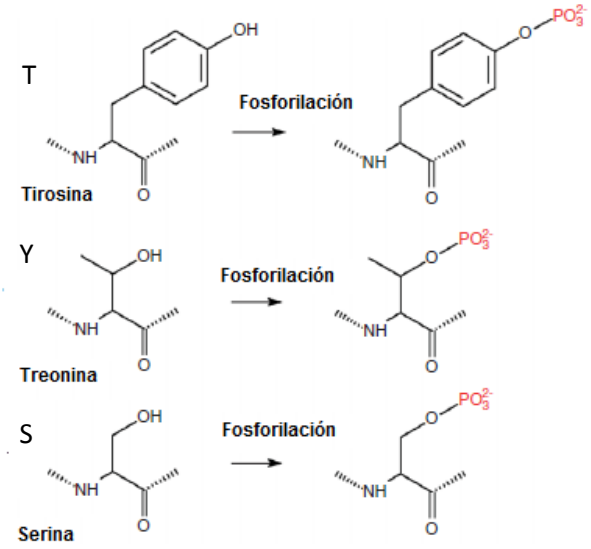


Fig. 2.25. Modificaciones de los aminoácidos T, Y, S por fosforilación (el grupo fosfato está marcado con color rojo).

Las proteinas que llevan a cabo esta modificación son llamadas cinasas (fosforilación) o fosfatasas (desfosforilación). Existen tres tipos de cinasas:

- Cinasas PKA, PKC y CAM PK: Dedicadas a fosforilar residuos de serina o treonina, son solubles y participan en los mecanismos de traducción de señales. También participan en la activación de cascadas de fosforilación.
- Cinasas Try: Con dominios TryK en receptores de membrana para insulina y factores de crecimiento, y procesos como transfosforilación, dimerización y activación.
- Cinasas múltiples.

2.3.2.2 Ubiquitinación.

La ubiquitina es un polipéptido de 76 residuos que marca la cadena lateral de un aminoácido de lisina, la ubiquitina realiza los siguientes tres pasos (Fig. 2.26):

- Activación de la enzima activadora de ubiquitina (E1) por adición de una molécula de ubiquitina, esta reacción requiere ATP.
- Transferencia de la molécula de ubiquitina a un residuo de cisteína en la enzima combinadora de ubiquitina E2.
- Formación de un enlace peptídico entre la molécula de ubiquitina unida a la E2 y un residuo de lisina en la proteína diana, esta reacción es catalizada por la ligasa de ubiquitina E3.

Una vez que la proteína es ubiquitinada es reconocida por una proteasoma (una maquinaria molecular), ésta por su parte activa de manera proteolítica las proteinas marcadas por ubiquitina quedando como resultado moléculas peptídicas cortas de siete a ocho residuos dejando las ubiquitinas intactas.

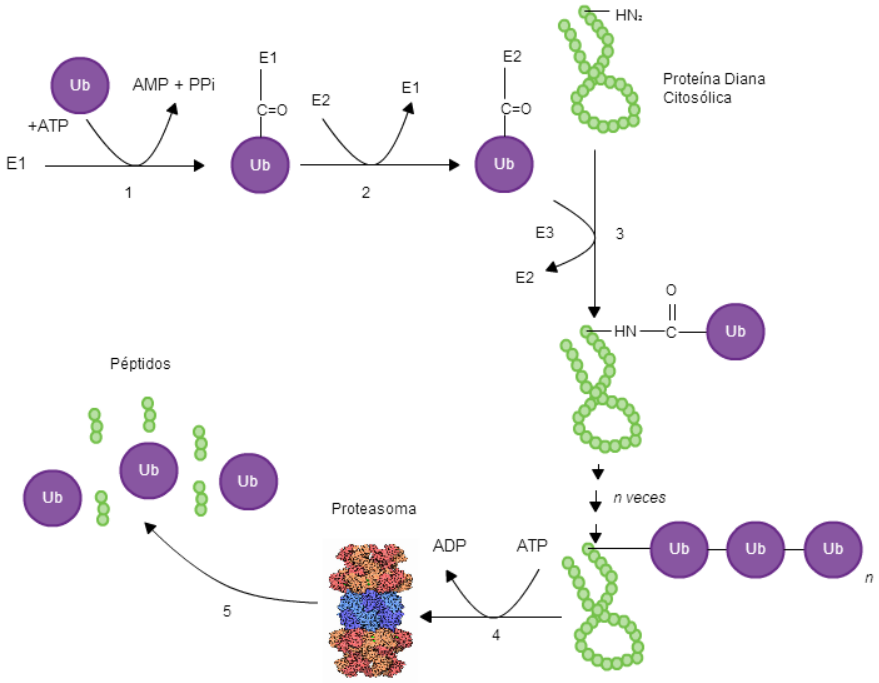


Fig. 2.26. La ubiquitina tiene relación con tres enzimas: la enzima E1 activadora (1) que más adelante es intercambiada por la enzima E2 combinadora (2), finalmente la proteína diana se une en su residuo de lisina con la ubiquitina (3). Una vez que la proteína fue marcada n veces, la proteasoma (4) separa las ubiquitinas y fragmenta en pequeñas cadenas péptidas (5). [32]

Existen otras modificaciones parecidas a la ubiquitinación que en vez de trabajar con la ubiquitina trabajan con otras moléculas similares, por ejemplo: la sumoilación añade una pequeña molécula llamada SUMO (pequeña molécula de ubiquitina). (Fig. 2.27)

Las función de la ubiquitinación como PTM es principalmente la degradación de proteínas y se considerado actualmente que también participa en la señalización interna de la célula. Además como se mencionó las PTM trabajan en conjunto, por ejemplo: las fosforilación y la ubiquitinación trabajan en forma positiva para la degradación de cíclinas (proteinas involucradas en la regulación del ciclo celular), en algún momento las cíclinas son fosforiladas por las cíclinas cinasas, que trae como consecuencia la exposición de los aminoácidos de lisina que son reconocidos por la ubiquitina, para después ser degradadas.

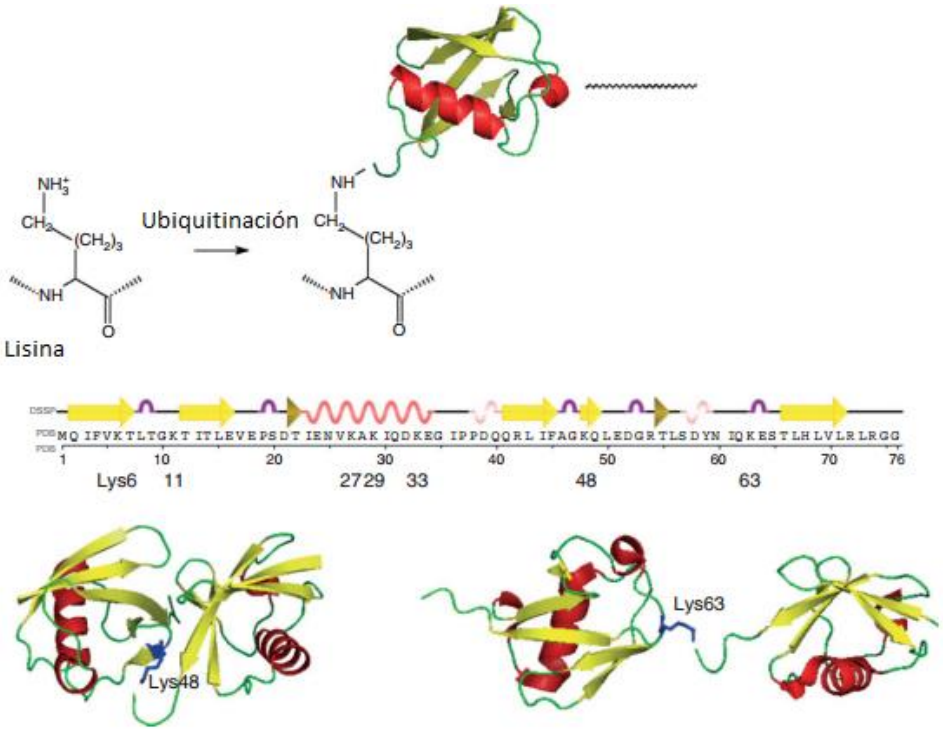


Fig. 2.27. Residuo de lisina ubiquitinado. Se muestra la estructura de primaria de la ubiquitina en humano, donde las flechas amarillas marcan el inicio de la estructura de una hojaβ, y la espiral en rojo la hélice α, en los imágenes de abajo se muestran como dímero y un residuo lisina coloreado en azul.

2.3.2.3 Acetilación.

La acetilación ha sido identificada en 80 factores de transcripción, así como en muchas regulaciones nucleares y en varias proteínas en el citoplasma. A pesar de que la acetilación no es la única modificación crucial en núcleo, es muy importante en la regulación de:

- La dinámica del citoesqueleto
- Metabolismo energético
- Endocitosis
- Fagocitosis
- Señalización en la membrana celular

Consiste en la adición de un grupo acetilo (CH_3CO) en un residuo de lisina como se muestra en la imagen:

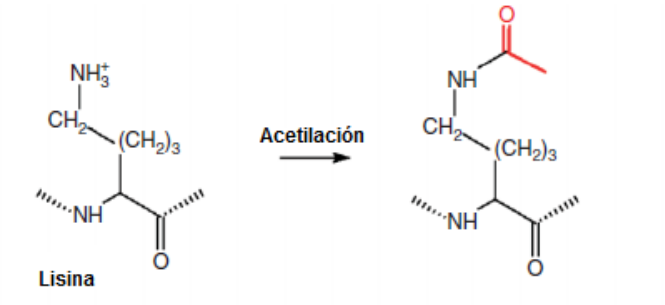


Fig. 2.28. Acetilación de lisina

Al igual que la acetilación; la metilación, ubiquitinación y fosforilación han sido identificadas en histonas, la suma de todas estas modificaciones postraduccionales que crea un complejo programa de desarrollo y patogénesis de las enfermedades y un complejo programa regulatorio. No sólo las histonas son objeto de modificaciones de acetilación sino otras más. Los biólogos moleculares y genetistas han llegado a la conclusión de poder extrapolar los cambios ocurridos en las histonas en otras proteínas. Por ejemplo: La acetilación en la histona H3 en Lys9 y Lys27 en unión con una fosforilación de la Ser10 y Ser 28 respectivamente, puede ocurrir en otras proteínas.

La fosforilación es generalmente la primera respuesta producida por estímulos a la célula, lo que puede resolver en convertir señales basadas en fosforilación a acciones en acetilación. Esto nos habla de que la acetilación puede mezclarse con otras modificaciones postraduccionales para llevar acabo complejos programas regulatorios.

Inicialmente la acetilación fue descubierta en las histonas, mediante dos procesos contrarios: la acetilación y deacetilación de residuos de lisina, las enzimas que participan en estos procesos son las HATs (histona acetiltransferas) y HDACs (histona desacetiltranferasa).

Para explicar la importancia de la acetilación en los mecanismos de traducción de señales, se hablará de un proceso, aunque no precisamente es de la levadura sino del ser humano; la acetilación del factor FoxO que es el encargado del crecimiento celular y la diferenciación. El factor IFG activa AKT, que fosforila tres sitios del FoxO, los cuales son Thr24, Ser256 y Ser319, que posteriormente se une a un complejo de proteínas llamado 14-3-3 (encargado del transporte al núcleo). Otra proteína llamada CBP acetila al factor FoxO1 en Lys245, Lys248 y Lys265. La Ser256 fosforilada promueve la retención citoplásmica de FoxO1, porque lo estimula a la unión con la ligasa Skp2 ubiquitina que es como una especie de caja llamada F-box que controla al factor FoxO1 (se tiene aquí un proceso de ubiquitinación dependiente de fosforilación), las acetilaciones podrían actuar indirectamente sobre la ubiquitinación.

Con esto se puede observar, una proteína puede ser modificada para acelerar su funcionamiento o limitarlo, así como existe un proceso para inhibir la acción de

FoxO1, existe un proceso contrario para acelerar su acción. Existen un sinfín de ejemplos que se pudieran dar, pero no es objetivo de esta tesis ver proceso por proceso, solo aquellos ejemplos que parezcan relevantes y tenga como objetivo ejemplificar las interacciones entre modificaciones.

Es claro que la acetilación ocurre desde proteinas en núcleo hasta en la membrana, cuyos cambios impactan a los mecanismos funcionales no sólo de proteína modificada sino de otras. Según el científico Xiang-Jiao Yang y su grupo de colaboradores han determinado que existen tres tipos de acetilaciones [33]:

- Acetilaciones de switch on/off: modificación de uno o varios residuos cuya función va dirigida a la activación o inhibición. Como la operación que realiza la acetil-CoA sintetasa.
- Acetilaciones en grupo: modificación de un grupo de residuos de lisina que forman un ajuste cargado que producirá un cambio funcional. Los multisitios de acetilación ejercen efectos calibre o proporcionar mecanismos a prueba de fallos.
  Acetilaciones multirelacionales: La interacción simultánea o secuencial con otras modificaciones postraduccionales como el ejemplo que se vio aquí de FoxO1. **(Xiang-Jiao Yang, Edward Seto, 2008, Lysine Acetylation: Codified Crosstalk with Other Posttranslational Modifications)**

2.3.2.4 Metilación.

La metilación o alquilación afecta a los siguientes aminoácidos: lisina, argenina, aspagarina, histidina, acido glutámico y en fenilalanina N-terminal o bien Cisteína C-terminal, entre otros. Al igual que la ubiquitinación se requieren de ciertas enzimas, así también en metilación se requiere de las metil-transferasas, estas utilizan como donador de metilos a SAM (S-adenosil-L-metionina). Para el caso de la lisina se pueden adicionar hasta tres grupos metilos (Fig. 2.29)

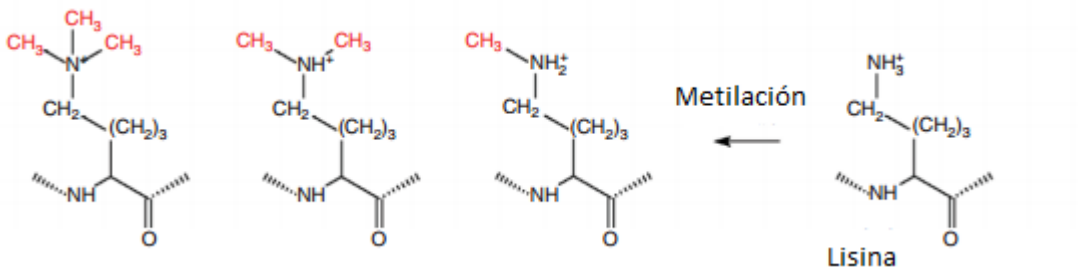


Fig. 2.29. Diferentes tipos de metilación (met, di y tri) de lisina

La metilación es un mecanismo epigenético (factores no genéticos que determinan el desarrollo del organismo), las ADN-metil-transferasas regulan la actividad de los genes. Cuando ocurre la metilación del ADN se desactiva por completo la expresión

de un gen. La metilación no es exclusiva de regulación de los genes sino de otros procesos importantes para la homeostasis del cuerpo como por ejemplo cuando la PNMT (enzima que convierte la norepinefrina en epinefrina) hace su función.
Incluso el estado de metilación de los genes puede ser utilizados como marcadores para detectar teratogénesis (agentes que provocan defectos congénitos), estos marcadores se han estudiado para diagnosticar el cáncer precoz.

Se puede clasificar la metilación en dos tipos:

a) De mantenimiento: adición en grupos metilos en ciertas parte del ADN de la célula madre, para que las células hijas mantengan el mismo patrón de metilación.
b) De novo: adición de grupos metilos en posiciones específicas para cambiar un patrón de metilación en el genoma. **(A. Saskia Venne, Laxmikanth Kollipara and René P. Zahedi, 2013, The next level of complexity: Crosstalk of posttranslational modifications)**

2.3.2.5 Glicosilación.

Esta PTM ocurre cuando se adicionan varios glucosilos (radicales de glúcidos) sobre las cadenas laterales de los aminoácidos, casi siempre en proteínas extracelulares o secretadas, cuando la proteína presenta un mayor peso molecular de cadena proteica que de parte glucídica se llama glicoproteína, en caso contrario se le llama proteoglicano.

Esta modificación se puede presentar de manera paralela a la síntesis de proteínas (modificación co-traduccional). Las funciones de la glicosilación son las siguientes:

- Estabilizar la conformación final de una proteína de membrana.
- Aumentar la vida media de una proteína incrementando su estabilidad y resistencia a los procesos de ubiquitinación.
- Aumentar la solubilidad de la proteína en un medio acuoso.
- Aportar las estructuras que reconocerán algunos mensajes del ambiente.

Las enzimas que participan en esta PTM, son las glicosil transferasas. En las células eucariontes la glicosilación tiene lugar en retículo endoplásmico (RE) y el aparato de Golgi.

Existen los siguientes tipos de glicosilaciones (Fig 2.30):

a) O-glicosilación: Afecta en residuos de serina y treonina en su grupo hidroxilo, produciendo oligosacáridos que empiezan por N-acetil-galactosamina. Generalmente ocurre en el aparato de Golgi.

b) N-glicosilación: adición al grupo amino libre de un residuo de asparagina, esta modificación se considera más co-traduccional, es ocurrida en el RE.

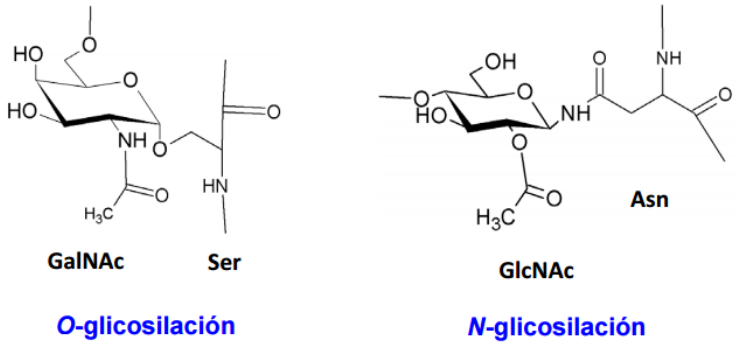


Fig. 2.30. Tipos de Glicosilación.

2.3.2.6 Otros tipos de modificaciones postraduccionales.

Hablar cada una de las catorce modificaciones sería complejo, en este libro nada más se hace mención de cinco modificaciones, pero las otras 9 se resumen en este apartado:

- Por puentes de disulfuro: en residuos de cisteína, tiene como función la dimerización de cisteínas para la formación de cistinas a través de un puente disulfuro. La cistina tiene la función de contribuir al mantenimiento de la estructura terciaria de una proteína.
- Succinilación: adición de un radical succinil en un residuo de lisina. [34]
- Por unión de iones metálicos: forman parte del centro activo de enzimas. La adición de un ion metálico como cofactor (componente no proteico y termoestable) de una proteína.
- Oxidación: La PTM de oxidación es una característica importante del estrés oxidativo y del envejecimiento, esta modificación es causada por las especies reactivas de oxígeno (ROS) o bien especies reactivas de nitrógeno (RNS), existen dos clasificaciones de oxidación; las reversibles (sulfenilación) y las irreversibles (nitrotirosina y carbonilación).
- N-terminal acetilación: es una variante de acetilación, pero estas modificaciones ocurren en el grupo amino inicial (extremo N-terminal) de varias proteínas.
- Lipidación: Es la adición de restos lipídicos con ciertos residuos de la proteína, aumentando la hidrofobicidad, y generalmente es una PTM que participa en el anclaje de proteínas en la membrana, también es conocida como prenilación.
- Por Calcio: El calcio es considerado un segundo mensajero en el citosol de las células eucariontes. Es un metal suave que es considerado relevante en la biología molecular por su estado de oxidación y la formación de iones que

permite a la proteinas sufrir algunos cambios conformacionales. Tiene una gran importancia como mensaje extracelular.

- Nitrosilación: Es la adición de un grupo NO (nitroxilo) en residuos de tirosina.
- Sitios Activos: zona de una enzima donde un sustrato es catalizado.

A continuación se muestra una tabla de las modificaciones postraduccionales y los aminoácidos que modifican, especialmente de las proteinas de levadura.

Aminoácido PTM	A	R	N	D	C	G	E	Q	H	I	L	K	M	F	P	S	T	W	Y	V
Fosforilación	×	×	×	×	×	×	×	×	×	×	×	✓	×	×	×	✓	×	×	✓	×
Ubiquitinación	×	×	×	×	×	×	×	×	×	×	×	✓	×	×	×	×	×	×	×	×
Acetilación	×	×	×	×	×	×	×	×	×	×	×	✓	×	×	×	×	×	×	×	×
Metilación	×	✓	✓	✓	✓	×	✓	✓	✓	×	×	✓	×	×	×	×	×	×	×	×
Succinilación	×	×	×	×	×	×	×	×	×	×	×	✓	×	×	×	×	×	×	×	×
Glicosilación	×	×	✓	×	×	×	×	×	×	×	×	×	×	×	×	✓	✓	×	×	✓
Oxidación	×	×	×	×	×	×	×	×	×	×	×	×	✓	×	×	×	×	×	×	×
Lipidación	✓	×	✓	×	✓	✓	×	×	×	×	×	×	×	×	×	✓	×	×	×	×
Por Calcio	✓	×	×	✓	×	✓	✓	×	×	×	×	×	×	×	×	×	✓	×	×	×
Por iones de metales	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	✓	✓	×	✓	✓
Nterminal Acetilación	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Puentes de disulfuro	×	×	×	×	✓	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Nitrosilación	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	✓	×
Sitios Activos	×	✓	✓	✓	✓	×	✓	✓	✓	×	✓	✓	×	×	×	✓	✓	×	✓	×

Tabla 2.7 Se muestran todos los aminoácidos con sus respectivas modificaciones que pudieran llegar a tener.

APLICACIONES Y LA BIOINFORMÁTICA

CAPÍTULO III

3.1 Aplicaciones médicas, farmacéuticas e industriales.

La levadura *Saccharomyces cerevisiae* es un organismo fundamental para la producción de productos industriales, por ejemplo en la producción de butanol que es usado como biocarburante así como la producción isoprenoides (compuestos químicos derivados del isopropeno) para aplicaciones farmacéuticas.

Existe un campo de reciente surgimiento llamado Ingeniería Metabólica como una rama de la Ingeniería Genética, que se encarga de extraer las ventajas de los metabolismos de los organismos para la producción de químicos como glucosa, sacarosa, biomasa, etc.

Algunos de los avances en la industria farmacéutica son:

- Producción de propanodiol utilizado como materia prima de fabricación de alfombras, telas y plásticos.
- El antibiótico cefalexina para tratar la neumonía e infecciones de los huesos, en general combatir contra enfermedades bacterianas en el ser humano.
- La producción de vitamina B2, que favorece el crecimiento corporal y la producción de glóbulos rojos.
- Creación de biocombustibles como el Butamax, como fuente alternativa de energía.
- Producción de bioetanol y biodiesel.
- Producción de insulina, vacunas contra la hepatitis y contra el virus del papiloma humano.

Además de las aplicaciones conocidas de levadura como son la producción de cerveza, vino, sake y pan. El reto de la ciencia en esta área es la selección de cepas adecuadas para producción de los químicos ya mencionados.

3.2 El mundo de la bioinformática.

La bioinformática es una rama de la Ingeniería en Computación, para muchos es considerado un campo de la ciencia interdisciplinario, que tiene como objetivo gestión y análisis de datos de experimentos biológicos, genéticos y médicos. Esta disciplina surgió en los 80´s y tiene un importe auge actualmente.
Gracias al descubrimiento del genoma humano en la década de los 90´s y la investigación genómica reciente, el interés por esta área se tornó más fuerte.

Con la ayuda de la inteligencia artificial, estadística y probabilidad, computación gráfica, programación de algoritmos complejos, el súper computo, las bases de datos y las matemáticas discretas.

En general la bioinformática trabaja a nivel molecular haciendo las siguientes actividades:

- Alineamiento de secuencias
- Comparación de genomas y proteomas (BLAST)
- Diseño tridimensional de proteínas
- Predicción de estructura de proteínas
- Interacciones entre proteinas
- Secuenciación masiva

La bioinformática también implica la construcción de nuevas herramientas para la comprobación de hipótesis y para dar información relevante de los datos biológicos. Antes de la llamada revolución genómica, la bioinformática era considera simplemente como el mantenimiento y gestión de las bases de datos asociadas a información biológica, pero su importante rol que tiene para resolver las necesidades en el áreas de la biología y medicina, ha sido un punto clave y ahora se considera una rama de estudio.
Entre las aplicaciones conocidas, se muestran en las siguientes imágenes:

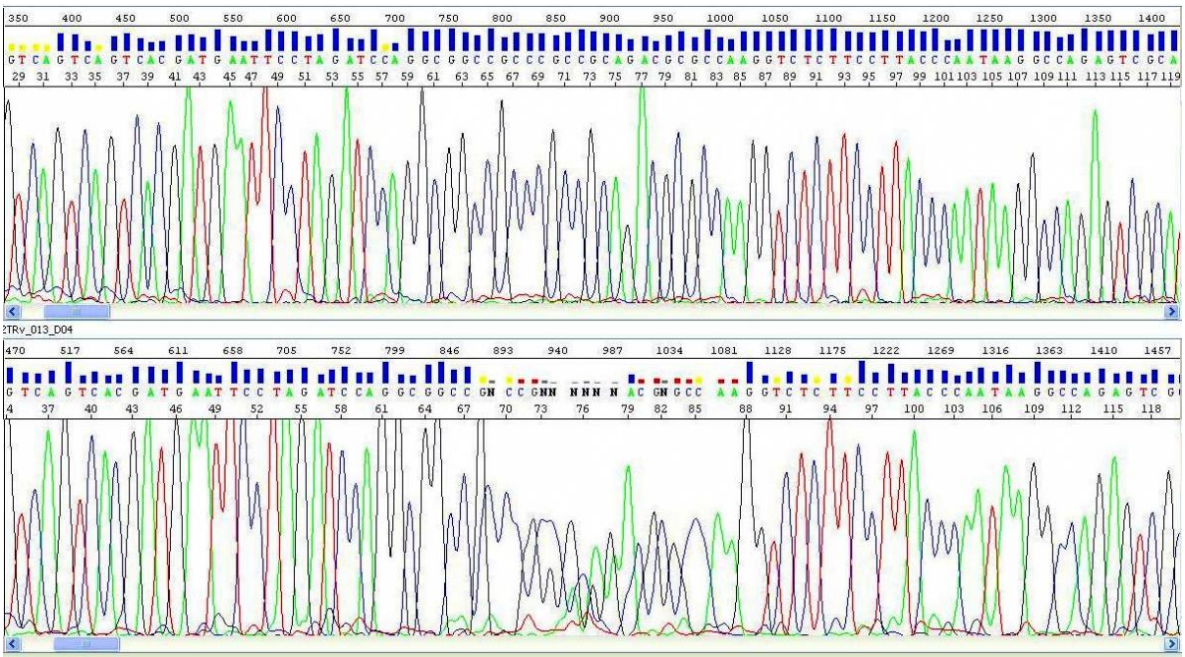


Fig 3.1 Secuenciación de un plásmido mediante un programa de computación llamado Templiphi [1]

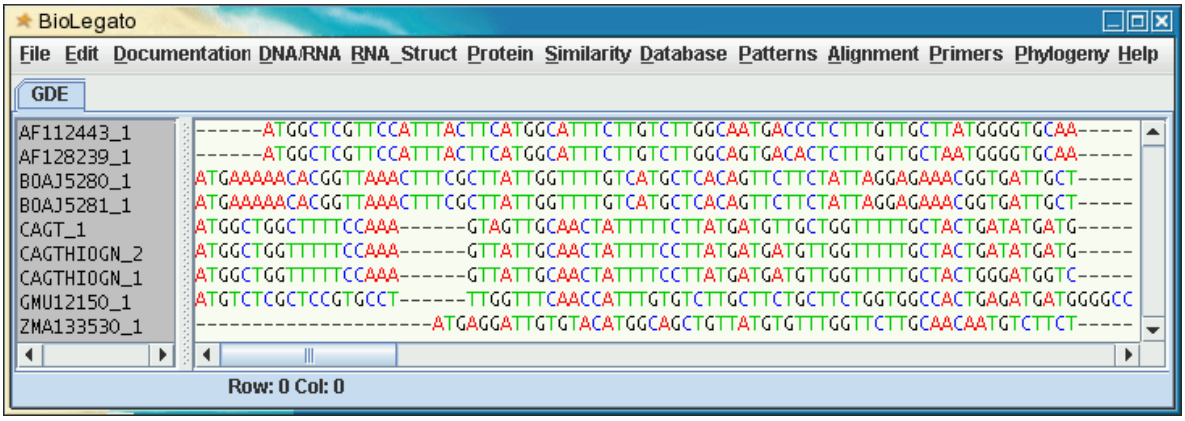


Fig 3.2 Técnica BLAST en alineación de varios genomas con un programa llamado BioLegato. [2]

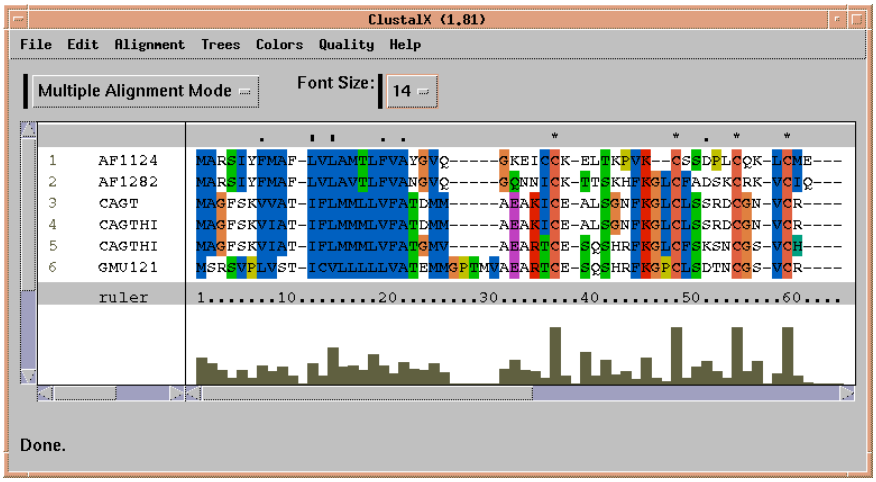


Fig 3.3 Técnica BLAST en alineación de varios genomas con un programa llamado ClustalX. [3]

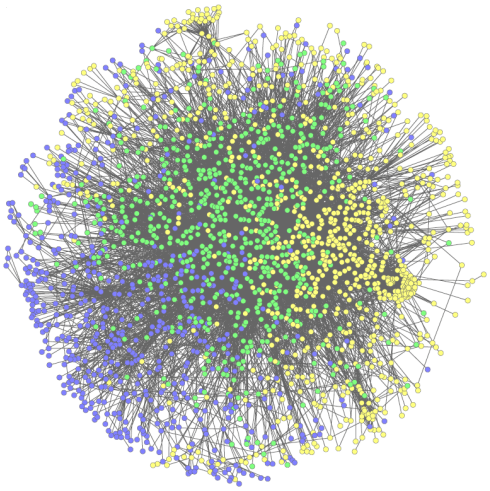


Fig 3.4 Red que muestra Interacciones entre proteinas que expresa la relación entre el síndrome QT y la epilepsia en el ser humano [4]

ANÁLISIS Y DESAROLLO

CAPÍTULO IV

4.1 Objetivos

Objetivos generales:

1. Recopilar toda la información que se tenga hasta la actualidad sobre las modificaciones postraduccionales.
2. Implementar un sistema de consulta en línea para facilitar el análisis y estudio de información referida a las modificaciones postraduccionales en las proteínas de levadura *Saccharomyces cerevisiae.*
3. Crear un ambiente gráfico flexible y de fácil uso para que los investigadores, estudiantes o personas interesadas en esta parte de la Biología Molecular o áreas afines; puedan interactuar.
4. Proporcionar un espacio de aportes y sugerencias (en la página web) para la mejora del sistema de consultas tanto en la base de datos como en la página web, sin olvidar una forma de contacto con los desarrolladores y los investigadores encargados del proyecto para dudas o aclaraciones.

Objetivos particulares:

1. Desarrollar los programas necesarios para el tratamiento de archivos con grandes volúmenes de información de modificaciones postraduccionales con la intención de que estos sean almacenados en la base de datos.
2. Diseñar una base de datos que cumpla con la teoría de normalización y la reducción de replicación de datos, así como la concurrencia y carga máxima de consultas. Esto con lleva al diseño de un modelo relacional (entidad-relación), modelo lógico y modelo físico bien estructurado.
3. Programación de una página web de orden público con una ip real asociada a un dominio perteneciente a la Universidad Nacional Autónoma de México, para la comunicación cliente-servidor.
4. Seleccionar un servidor con las capacidades suficientes para cumplir con los objetivos planteados anteriormente, dicho servidor se adaptará a los recursos disponibles.

4.2 Análisis.

4.2.1 Descripción del problema para el diseño de la base de datos.

Existen varias bases de datos en el mundo científico asociadas a las proteínas de levaduras, dichas bases de datos contienen información esencial para el análisis, comportamiento y desarrollo de investigación principalmente en el área de Biología Molecular e Ingeniería Genética. Algunas de las bases de datos más utilizadas por la comunidad científica en este rubro son: Saccharomyces Genome Database [1], phosphoGRID [2], ProteomicsDB powered [3], entre otras más, desarrolladas por

universidades como la Universidad de Stanford, Universidad de Princeton, University of Edinburgh, también algunos institutos de investigación o incluso por empresas que venden anticuerpos.

El problema fundamental es que no existe una base de datos que contenga información de modificaciones postraduccionales completa, en estas bases de datos mencionadas existen sólo parte de los registros ya que la mayoría no toma en cuenta los trabajos más recientes, debido al uso de nuevas tecnologías que generan miles de datos. El objetivo de ésta base de datos es resolver éste problema. Se necesita una base de datos que cumpla con los requisitos de normalización y la teoría de las bases de datos, para minimizar los problemas que en algún momento se puedan presentar al tener una gran cantidad de registros y también los problemas de incidencia, concurrencia y replicación.

La información que se obtiene de las bases de datos ya mencionadas es cruda y se requiere una investigación ardua para obtener la mayor información posible, además de que la mayoría de los sitios no dejan descargar al usuario la información ni realizar análisis con los datos. Una vez obtenida la información se necesita depurar. Regularmente se descargan archivos de Excel y CSV con cientos y miles de registros, incluso estos archivos se tienen que pasar a archivos de texto plano para hacer más flexible su uso, entonces se parsea, se depura el archivo para que estos pueden ser almacenados por lenguaje SQL a la base de datos, evitando ingresar registro por registro, sino tabla por tabla. Por lo tanto también se requiere de algoritmos de programación prácticos y lo más simples posibles para el tratamiento de archivos. Algunos algoritmos de programación se pueden volver más complejos y se necesitará de lenguajes de programación apropiados cuando de análisis sintáctico se trate como puede ser Perl y su extensión BioPerl, además de BioPython.

El estudio de las modificaciones postraduccionales y sus bases de datos son el resultado de investigaciones reportadas mediante publicaciones científicas, éstas publicaciones son almacenadas en la base de datos PumMed [4], cada publicación tiene asociado un identificador conocido como PMID, también será importante que la base de datos maneje correctamente estos PMID con la finalidad de darle al usuario la suficiente información para que pueda buscar con más detalle la fuente de la información.

Una vez teniendo la columna vertebral del proyecto con las querys de búsqueda que se definirán a lo largo de este capítulo, se necesitará una página web bien estructurada que tenga una interfaz amigable para los usuarios. En las páginas web mencionadas en las referencias, no se presentan gráficos en 3D para visualizar mejor la morfología de la proteína del producto de procesos de cristalización o espectrometría de masas. Para la construcción de un gráfico en 3D de una proteína se necesita de estudios de computación gráfica, estadística, probabilidad y de

Inteligencia Artificial en algunos casos. Como una mejora, se anexará un espacio en la base de datos y en la página web para su implementación.

Gracias a los avances que han hecho en las áreas de la visualización y computación gráfica, actualmente es posible hacerlo desde una computadora personal e incluso desde un dispositivo móvil. Los archivos que permiten hacer esto son de extensión pdb, los cuales contienen un formato bien definido y complicado de interpretar para un lenguaje de programación usual. Por lo que los algoritmos tendrían que ir dirigidos a obtener información útil para la lectura de estos archivos y rescatar información vital para la construcción de una tabla en la base de datos.

Existen programas y aplicaciones móviles que pueden utilizar estos archivos pdb para visualizar en 3D las proteínas y de las diferentes representaciones que existen descritas en el capítulo 2, de hecho, se utilizaron algunos para hacer imágenes representativas en capítulos anteriores, tales como son PyMol, ChemDoodle con sus APIs; Web Componentes Mobile y Desktop, JSMol, NDKmol, entre otras. Se tendrá como objetivo describir cual será la aplicación que se emplee para la presentación de las proteínas en 3D en la página. Así como el parseo de los pdbs contenidos en las bases de datos de levaduras, esto con la finalidad de presentar la respectiva visualización marcando solo aquellos aminoácidos donde ocurren modificaciones.

Casi todos los pdbs que caracterizan proteínas de levaduras no están completamente cristalizadas, solo ciertas regiones o partes de dominios, también se necesitará, hacer depuración de pdbs que sí sirvan como representación de las modificaciones contenidas en la base de datos.

Y finalmente se necesitará la programación web para hacer la fusión entre las consultas de la BD y la página web, de forma que la información se muestre amigable y de fácil uso; para resolver este problema se tendrá que escoger una forma de colocación de tablas y seleccionar adecuadamente el espacio disponible con el diseño web.

Para la actualización constante de la base de datos se necesitará de un área de contacto así como una sección en la página web donde los usuarios que estén interesados podrán agregar o actualizar información de alguna tabla que ellos deseen. Posteriormente esta información tendrá que ser valorada por los expertos en el área, para ser actualizada dentro de la base de datos.

En general, se puede resumir en los siguientes puntos los problemas a resolver:

a) Analizar los medios disponibles, tanto de hardware como software para el desarrollo del proyecto, desde la elección del servidor hasta los servicios de red.

b) Diseñar e implementar una base de datos que contenga la información completa de modificaciones postraduccionales de la levadura *Saccharomyces cerevisiae* que se tengan hasta este momento. Hacer las consideraciones suficientes para un funcionamiento adecuado tanto las que van orientadas a la teoría como las orientadas a la práctica
c) Realizar los algoritmos de parseo necesarios para la depuración de la información que se almacenará a la base de datos.
d) Programación de las querys de la base de datos orientadas a búsquedas que puedan dar información valiosa a los usuarios finales. Estás se definirán a lo largo de esta tesis.
e) Clasificar, ordenar y mostrar los modelos de proteínas en 3D con sus respectivas modificaciones postraduccionales.
f) Con base a un diseño web, unir las consultas al espacio disponible dentro de las páginas, para que las querys sean accesibles y de fácil uso.

4.2.2 Metodología.

De acuerdo a la Ingeniería de Software se utilizó lo siguiente:

- Modelo de desarrollo incremental para el ciclo de vida del proyecto: Este modelo se basa en el modelo en cascada en forma interactiva para la construcción de prototipos. En este proyecto se generaron tres prototipos:

    1) Primer prototipo: Desarrollo en WAMP. (Desarrollo primario)
    2) Segundo prototipo: Desarrollo en LAMP con el primer diseño web. (Desarrollo secundario)
    3) Tercer prototipo: Desarrollo en LAMP con diseño web finalizado (representación en 3D de proteínas). (Fig 4.0)

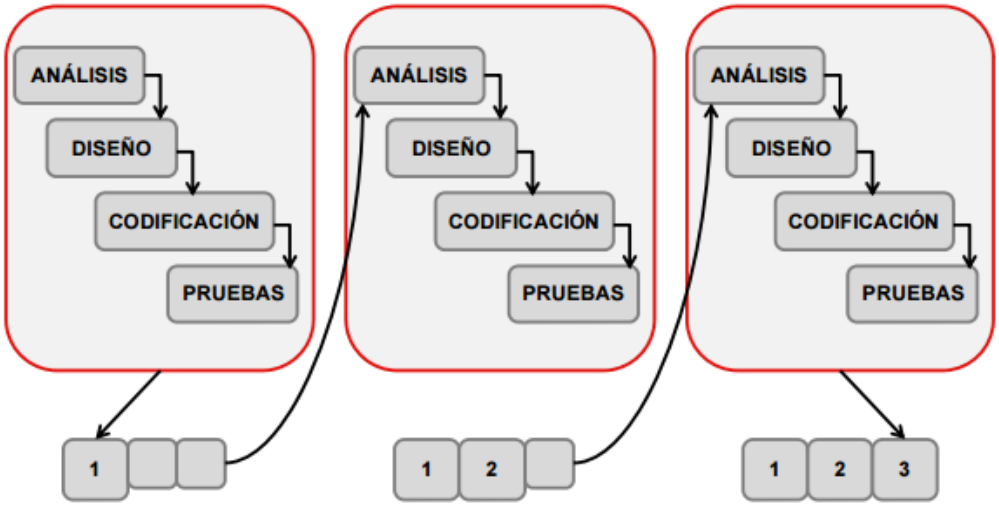


Fig. 4.0 Ciclo de vida utilizado en este proyecto.

- Metodología ágil predictiva y a la vez adaptativa: *Extreme Programming (XP)*.
  Formulado por Kent Beck, esta metodología hace hincapié a la adaptabilidad más que la previsibilidad. Debido a que los cambios de requisitos en un proyecto son naturales, se necesita una metodología referida a resolver esta problemática; la metodología XP es capaz de proporcionar tal solución.

  La metodología XP consiste en:
  - El cliente (experto del área de Biología Molecular) define un(los) requerimiento(s) a implementar.
  - El (los) desarrollador (es) estiman el esfuerzo para la implementación.
  - Se lleva a cabo la implementación.
  - Se válida por el cliente dicha implementación y plantea un posible cambio o un nuevo requerimiento.

- El ciclo de vida referido por la metodología XP es de: exploración y planificación de la entrega (análisis), las iteraciones (diseño), la producción (codificación y pruebas) y mantenimiento. Todo lo anterior basado en comunicación constante con cliente.

- Para el diseño de la base de datos se utilizó la metodología tradicional: diseño del modelo entidad relacional (con el modelo desarrollado por Peter Chen), seguido del modelo relacional (según las reglas de Frank Codd) y finalmente el diseño físico dependiendo del manejador de base de datos que se elijará más adelante.

4.2.3 Alternativas de solución, selección del manejador de base de datos y servidor de almacenamiento y despliegue de la página web.

Una vez conocidas las necesidades y los problemas a resolver, lo primero que se tiene que dar a conocer son algunas alternativas de solución, donde se discute las ventajas y desventajas entre las diferentes posibilidades que existan para resolver este proyecto, pero considerando de igual forma los recursos con los que se cuentan.
En primer lugar se cuenta de una infraestructura virtual robusta construida en Citrix XenServer y con una IP real del segmento 132.248.16.X, se trabajará con estos elementos puesto que son los asignados para este proyecto.

4.2.3.1 Servidor YAAM (Yeast Amino Acid Modifications).

Se trabajará con un servidor virtualizado ya que se cuenta con la infraestructura virtual y debido a que será un servidor web que será consultado en cualquier parte del mundo, se considera que deberá tener una memoria RAM apta para desempeñar un trabajo adecuado, un disco duro y un procesador lo suficientemente planeados para que en un futuro esta base pueda seguir creciendo.

En buenas prácticas de manejo de servidores, para este proyecto se consideran los siguientes tres elementos (Tabla 4.1), otros características como la tarjetas de red, monitores, etc. no son del todo relevantes para la planeación de este servidor. Se tendrá una tarjeta de red para acceso a internet y cualquier otro servicio de mantenimiento será realizará propiamente desde un servicio de VPN, desde SSH o bien desde la consola de administración de XenServer.

RECURSOS DE HARDWARE

Característica	Uso	Capacidad		Número máximo de usuarios		Número mínimo de usuarios	
Para etapas de:		Prueba y desarrollo	Publicación	Prueba y desarrollo	Publicación	Prueba y desarrollo	Publicación
Memoria RAM	Para el acceso rápido a la información contenida en la base de datos y los programas involucrados de respuesta a las consultas.	4 Gb	32 Gb	Desarrolladores: 15* Usuarios finales 50*	Desarrolladores: 30* Usuarios finales 300*	Desarrolladores y usuarios finales: 1	Desarrolladores y usuarios finales: 1
Disco Duro	Para almacenamiento local de la base de datos y herramientas involucradas para el desarrollo web	50-100 Gb	100-250 Gb	NA	NA	NA	NA
Procesadores Virtuales	Para dar respuesta a todos los procesos internos desde la capa de aplicación hasta la de datos	4 núcleos	8 núcleos o más	Desarrolladores: 15* Usuarios finales 50*	Desarrolladores: 30* Usuarios finales 300*	Desarrolladores y usuarios finales: 1	Desarrolladores y usuarios finales: 1

Tabla 4.1 Especificaciones de hardware

* estas cifras son evaluadas a partir de usos moderados, usuarios finales que hagan consultas una tras otra (secuenciales), no paralelas y desarrolladores o administradores del servidor, personas dedicas al mantenimiento del servidor y la programación.
NA- No aplica, puesto que no será un servidor orientado al almacenamiento neto.

Se plantean las siguientes alternativas para sistemas operativos:

a) Servidor UNIX: con sistema operativo Linux o de la familia de Red Hat, CentOS o Fedora, o de cualquier otra distribución, tiene la gran facilidad de administración y monitoreo constante, la instalación del sistema operativo es gratuita al ser de software libre.

b) Servidor Windows: con este sistema operativo, se tendrá que pagar la licencia original ya sea de un sistema operativo Windows Server o una versión de desktop, así como los programas para el desarrollo de este proyecto.

c) Servidor Mac OS: al igual que los sistemas operativos Linux, el software del sistema operativo es gratuito sin embargo las herramientas de desarrollo como algunos manejadores de bases de datos no están del todo disponibles o algunas aplicaciones de programación no están actualizadas.

Discusión: se podría trabajar con las tres alternativas sin embargo, en el caso de un servidor Windows se requiere un gasto económico adicional a diferencia de las otras dos alternativas, por lo tanto esa opción queda descalificada. Los desarrollos de páginas web y bases de datos no son del todo comunes en iOS, generalmente la programación va orientada a asuntos más comerciales o de beneficios privados, el interés de este sistema de consultas será más académico y para el desarrollo de la ciencia. Se considera la opción a) como la más adecuada, la distribución será la versión más estable de CentOS, la cual es la versión 6.5. En este sistema operativo se instalará el manejador de bases de datos y toda la infraestructura necesaria para la programación web.
Aunque existen otras distribuciones de la misma familia, CentOS es un sistema operativo que va a la par de la distribución padre Red Hat, pero las aplicaciones contenidas en él están del todo probadas y estandarizas en la comunidad de desarrolladores. Otras distribuciones como Fedora, Ubuntu, Debian, etc. son muy cambiantes o no usuales.

4.2.3.2 Manejador de bases de datos.

Es necesario escoger un manejador de bases de datos acorde a las necesidades del proyecto, debido a que esta herramienta nos proveerá del mantenimiento, rendimiento e implementación de la base de datos. Existen muchos de ellos, tanto como de software libre tanto de paga, en el primer grupo uno de los máximos representantes es MySQL y en el segundo Oracle. Para este proyecto se trabajará con MySQL, al ser muy flexible y accesible, además de ser multiplataforma.

MySQL cuenta con herramientas auxiliares para conexión y manejo, como la aplicación de desarrollo Workbench, además de contar con programas especializados para elaboración de diagramas lógicos y conceptuales.
La instalación se realiza de manera muy sencilla en CentOS, se puede instalar de manera gráfica como la aplicación ya mencionada y también en consola. La administración de la base de datos será de las dos formas. En caso de necesitar migraciones, exportaciones o importaciones de tablas, con este programa será muy flexible de realizarlo.

4.2.3.3 Lenguajes de programación y diseño web.

Los lenguajes seleccionados para el desarrollo de este proyecto:

- PHP: El lenguaje para realizar las conexiones a la base de datos y realizar las querys específicas cuyos resultados se desplegarán en la página, también será la unión de JSON con HTML.
- JSON: Es un formato ligero de intercambio de datos recibidos por la query para después ser trabajados en JavaScript de manera más sencilla.
- JavaScript: nos dará la posibilidad de trabajar con tablas de manera más sencilla y explotar al máximo toda el API, donde se puede declarar variables y funciones que serán llamadas después en HTML para ser visualizadas.
- HTML: es un lenguaje de etiquetas multiplataforma y altamente escalable y el más utilizado para páginas web, se ocupará para darle formato a la página y manejo de CSS (hojas de estilo).
- Ajax: Es una técnica de desarrollo web para crear aplicaciones iterativas con el cliente, mientras que la comunicación con el servidor es de manera asíncrona y se usará para mantener intercambio de datos con el servidor sin alterar la visualización del cliente.

Las siguientes bibliotecas de desarrollo y consulta:

- ChemDoodle Web Componentes: Es una API de desarrollo de paga, aunque tiene una versión para el ámbito académico y de investigación, esta última versión es la que se utilizó para este proyecto. Con esta API se puede visualizar los pdbs de las proteínas de levaduras y colorear de las formas que se deseen.
- YeastMine: es una de las bases de datos que otorga información útil de lo que respecta a levadura, además de otorgar estadísticas sobre los proteomas, como peso moleculares, tamaño en aminoácidos, etc.
- DataTables: es un plugin dentro de jQuery que permite dar un formato a las tablas como resultados de las querys obtenidas. Así como hacer una paginación correcta y agradable.

- Google Charts: es una herramienta de desarrollo para el manejo de tablas y gráficas, en este proyecto será utilizado para versiones de pruebas.
- RSCB Protein Data Bank: es la base de datos que contiene todos los pdbs que serán utilizados además de ofrecer archivos fastas de cada una de las proteínas.
- PubMed: es una página que tiene el control de las publicaciones realizadas, donde se encuentran documentadas los experimentos e investigaciones sobre las modificaciones encontradas en la levadura.

4.2.3.4 Ambiente de desarrollo.

Hay una amplia gama de IDEs (Entorno de desarrollo integrado) que son utilizados para la programación en los lenguajes mencionados, el más apropiado para este proyecto es NetBeans, aunque existen otros más, tienen más desventajas con respecto a éste. Además, también se utilizarán herramientas auxiliares como los editores de texto de sistemas UNIX que se mencionan en la lista de abajo.

- NetBeans: es un ambiente de desarrollo que tiene la característica de ser multiplataforma, no tiene ningún problema para trabajar con programas alternativos de edición de texto y de servicios web como Apache. Es altamente escalable y tiene la posibilidad de manejar más de quince lenguajes de programación de alto nivel.
- Gedit: es un editor de texto en sistemas UNIX, es muy utilizado porque su manejo es sencillo, es sensible a sintaxis a la mayoría de los lenguajes de programación, es ligero y se utilizará para programación en Perl.
- Vim: es un editor de texto vía consola en sistemas UNIX, que se utilizará para la programación de varios parsers en Perl y Python.

Otros editores de texto que son útiles para este proyecto son notepad++, nano y Python Scripter, todos ellos son gratuitos y cuentan con las herramientas suficientes para revisar, programar y analizar archivos.

Algo importante de instalar además de las aplicaciones ya mencionadas, es el escenario para la comunicación de la base de datos con la página web y esta a su vez con los usuarios finales, es decir los servicios de red, programación y de bases de datos. Para esto existen las siguientes opciones:

- XAMPP: por sus siglas X, cualquier sistema operativo, Apache, MySQL, PHP y Perl, es una infraestructura de desarrollo y servicio de red específicamente para sistemas de consultas, se puede decir que es exactamente a los mismos sistemas que se muestran a continuación, solo que en vez de tener dos lenguajes de programación, en los otros sólo se mantiene uno, ya se Perl o PHP.

- LAMP: por sus siglas Linux, Apache, MySQL y PHP en todo caso también puede ser Python o Perl, esta infraestructura es exclusiva de distribuciones Linux, cumple con la misma función de actuar como un conjunto de servicios de red y desarrollo.
- MAMP y WAMP, estas alternativas quedan descalificadas, puesto que se ha elegido un sistema Linux, las siglas M y W hacen referencia a Mac OS y Windows respectivamente aunque para este proyecto se harán algunos pruebas en WAMP.

Discusión: Como se puede observar XAMPP y LAMP es prácticamente lo mismo, de hecho implícitamente para este proyecto se utilizará también Perl, porque lo que se usarán los dos. En lo que respecta a WAMP, se utilizará para realizar las primeras querys y para ver si el funcionamiento de la base de datos es el adecuado. Es importante mencionar que para el sistema operativo Windows utilizado para estas pruebas se contaba con la licencia pagada, por lo que no significó un gasto.

La instalación de la infraestructura de LAMP en CentOS 6.5, se hace por paquetes, se instala uno por uno, y se van uniendo a través de la asignación de permisos, con los comandos chown y chmod en la consola. Como ya se mencionó NetBeans no tiene problemas para trabajar con estas dependencias, por lo que es accesible.

4.2.3.5 Implementación de la infraestructura de desarrollo para etapa de pruebas y fin de producto.

Una vez establecidas las condiciones y las elecciones para el desarrollo del proyecto, a continuación se muestra como se fue formando poco a poco el escenario de desarrollo, esta sección se dividirá en dos, en la etapa de pruebas; donde se realizó la mayor parte del proyecto y la etapa final donde se emigró toda la programación al servidor YAAM. Ésta última todavía se siguió trabajando para ligarla con la página web.

4.2.3.5.1 Escenario de pruebas y desarrollo primario.

Se trabajó con un servidor Windows, lo primero que se hizo fue instalar el WAMP, éste a diferencia del LAMP se instala como una paquetería dependiente, es decir se instala todo en su conjunto. La página donde se puede encontrar el software es el sitio oficial: http: //www.wampserver.com/en/, de descarga el servicio dependiendo la versión del sistema operativo, ya sea de 64 bits o 32 bits. Una vez descargado el programa el paso siguiente es la instalación.

- Se ejecuta el setup de instalación, se muestra un cuadro especificando la versión de WAMP que se instalará, así como los programa asociados (Apache, MySQL, PHP) y adicionalmente una consola de administración

PHPMyAdmin y otras dos aplicaciones que son irrelevantes puesto que no se ocuparán.

- Se aceptan las condiciones de licencia y posteriormente se indica la ruta de instalación, en el servidor se dejó en el disco local, y se sigue con la instalación. Es muy importante recordar el lugar dónde se instalará puesto que los archivos de configuración de los archivos se encontrarán ahí.
- Una vez que se instaló, se ejecuta el servicio, en la barra de tareas se muestra el icono de WAMP (una W, Fig 4.1) que puede variar en su color, desde rojo hasta verde, si esta rojo esta pausado, si esta verde todos los servicio están corriendo. Para comprobar que los servicios están corriendo solo basta con abrir la página de proyectos. (esta imagen se muestra abajo).
- Después se instaló NetBeans de la página oficial: https: //netbeans.org/, se creó un proyecto especial para el manejo de programas llamado Proteins. La instalación de este programa es típica como cualquier otro programa en Windows, en particular se descargó con el Java JDK, de la página de Sun Microsystems.
- Finalmente se instaló el Workbench de MySQL también de la página oficial: https: //www.mysql.com/products/workbench/, esta herramientas sirve para conectarse a la base de datos que más adelante se diseñará.

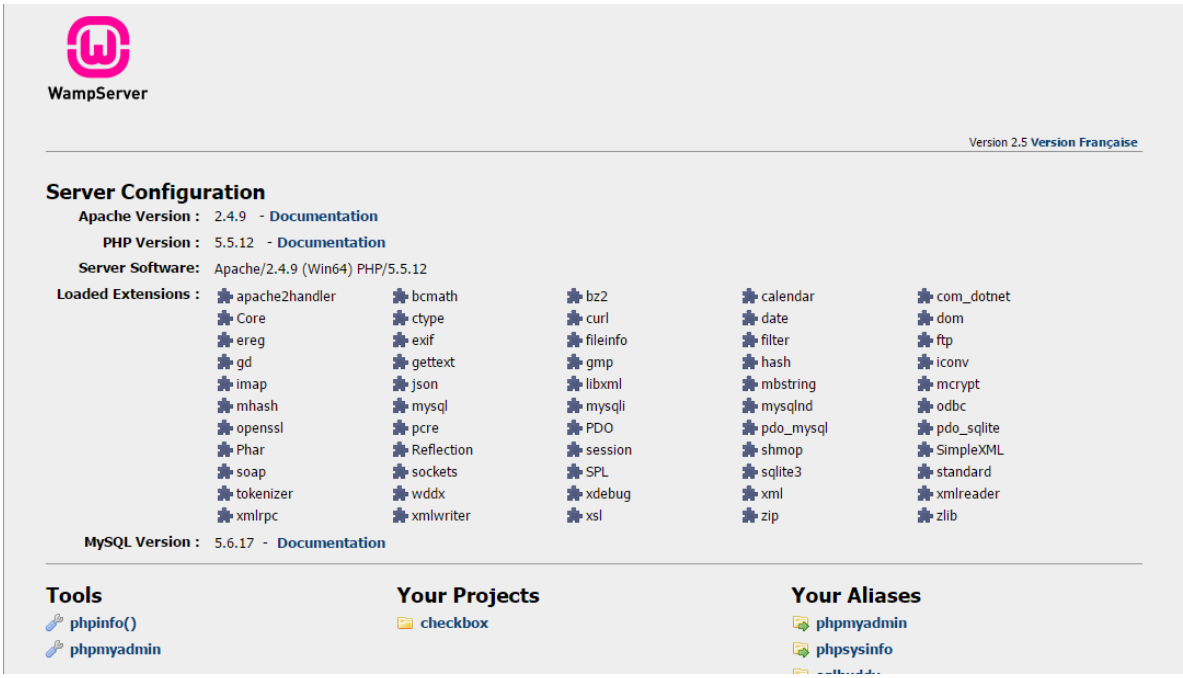


Fig. 4.1 Instalación completada de WAMP server en un servidor Windows, se puede observar los servicios involucrados con sus respectivas versiones.

4.2.3.5.2 Escenario del Servidor YAAM y desarrollo secundario.

Como ya se mencionó se utilizó el sistema operativo CentOS 6.5, se instaló el servidor en un pool virtual de Citrix XenServer llamado Cronos y en el servidor central virtualizado Júpiter. Se le asignaron los recursos planteados anteriormente para las etapas de pruebas y desarrollo (Fig. 4.2).

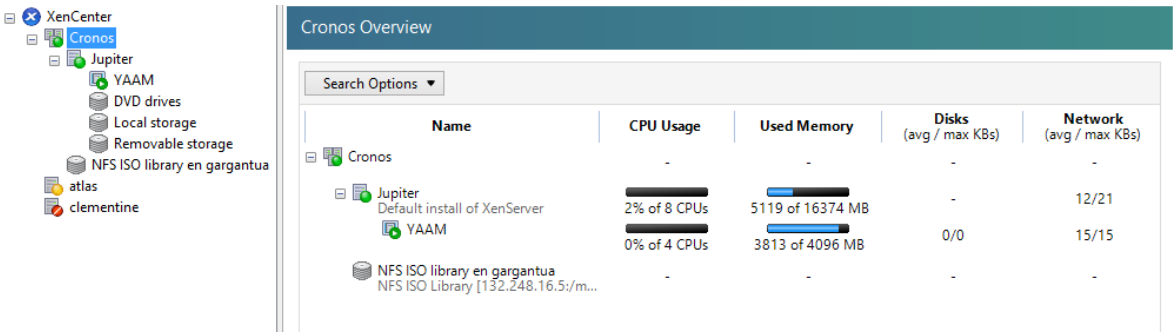


Fig. 4.2 Muestra de la consola de administración del Citrix XenServer, donde se puede observar el pool Cronos, el servidor virtual Júpiter y como host virtual el servidor YAAM.

Cabe mencionar que el disco duro del servidor YAAM es adquirido desde un Servidor de Almacenamiento NAS llamado Gargantua. Para facilitar administración del servidor se instaló VNC Server, de esta manera se evitará la molestia de estar constantemente accediendo a la consola de administración del XenServer. Se puede acceder al servicio de VNC instalando una aplicación de VNC viewer, que existe para desktop hasta para dispositivos móviles (Fig 4.3).

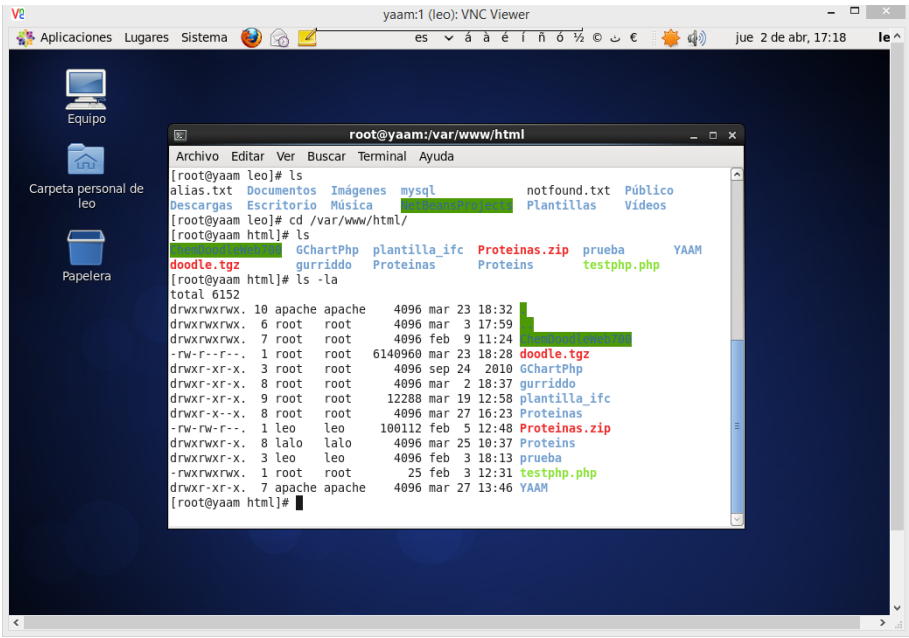


Fig. 4.3 Entrada a Consola VNC del Servidor YAAM para hacer más accesible la administración e instalación de herramientas de desarrollo.

El paso que siguió, es la instalación de toda la infraestructura LAMP, se instaló MySQL primero con su herramienta Workbench y el IDE NetBeans, posteriormente Apache y las independencias de PHP. Todas estas instalaciones fueron en modo consola, ejecutando comandos sencillos de instalación como yum o rpm. El directorio donde se leerá todos los archivos para el funcionamiento del proyecto es /var/www/html, es aquí donde Apache lee los proyecto y los despliega, cualquier otro proyecto que este fuera de esta ruta, se tendrá que editar en los archivos de configuración de Apache, así como cambiar los privilegios para que pueda ser visualizado por todos los usuarios.

De esta forma se tienen listas todas las herramientas para programar los parsers y el diseño de las base de datos. Lo que se tiene planeado es diseñar la base de datos e implementarla en el servidor de pruebas así como realizar los parsers necesarios que más adelante se explicarán. Para finalmente migrar toda la base a este servidor que será el final y en el que se mostrarán los resultados.

4.3 Diseño de la base de datos para modificaciones postraduccionales.

4.3.1 Requerimientos.

Para tener una idea más clara para el diseño de la base de datos, se presentan los requerimientos en los siguientes enunciados, adquiridos por expertos en el área de modificaciones postraduccionales:

Se tiene información del todo el proteoma de la levadura *Saccharomyces cerevisiae* donde cada proteína tiene un identificador universal también llamado nombre esquemático, una secuencia de aminoácidos, un identificador único de la base de datos de SGD (Saccharomyces Genome Database). Algunas proteínas tienen un nombre estándar como comúnmente se les conoce, también algunas proteínas tienen una serie de nombres que se les asigna debido a la función que realizan llamados alias, así como una descripción breve de sus funciones y relevancia.

El identificador universal es construido de una manera muy particular:

- Inicia con la letra Y, porque es el organismo al que pertenece, en este caso Yeast del inglés, levadura.
- La segunda letra es número de cromosoma donde está ubicado, donde el 1 corresponde a la letra A, el 2 al B y así sucesivamente.
- La tercera letra es a que brazo del cromosoma corresponde al izquierdo (L) o al derecho (R).

- Los siguientes tres dígitos indica el orden de los ORFs (marco abierto de lectura) a partir del centrómero.
- Y la última letra indica si la cadena tiene un orden ascendente o descendente respecto a la convención de lectura, ya sea Watson (W) o Crick (C), en otras palabras las cadenas complementarias.

En muchas proteínas, al identificador se le asocia un guion acompañado de una letra (-A,-B,-C, etc.), esto debido a que se encontró el gen que codifica para esta proteína después de el genoma reportado.

No todas las proteínas siguen con la estructura del identificador universal ya mencionado, cuando son ORFs mitocondriales los identificadores inician con la letra Q seguido de cuatro dígitos y cuando son obtenidas en plásmido inician con la letra R más 4 dígitos seguida de una W o una R.
El identificador de la base de SGD es más sencillo inicia con la letra S seguida de 9 dígitos.

Se debe de prever a futuro la adición de información sobre regiones ordenadas y desordenadas para cada proteína relacionada a su secuencia de aminoácidos, estas regiones se caracterizan por manejar dos caracteres ya sea O´s y D´s o bien 1´s y 0´s.

Además existe información adicional de la localización de muchas de las proteínas dentro de la célula. Algunas proteína tienen varias localizaciones. Por ejemplo la proteína YDR395W se encuentra en el núcleo y en citoplasma, sin embargo, de la proteína Q0017 no se tiene información de sus ubicaciones.
Se cuentan con la información correspondiente de catorce modificaciones postraduccionales las cuales son:

1. Acetilación.
2. Ubiquitinación.
3. Succinilación.
4. Metilación.
5. Fosforilación.
6. Nitrasilación.
7. Modificaciones ocurridas en N terminal
8. Oxidación.
9. Modificaciones por iones de metal
10. Lipidación.
11. Glicosilación.
12. Modificaciones por el ion Calcio.
13. Por puentes de Disulfuro.
14. Sitios Activos

De las cuales, las primeras ocho se tiene información del identificador de la proteína que está sufriendo la modificación, la posición donde ésta ocurre, el residuo que es modificado, una secuencia de péptidos después del residuo modificado de unos 10 a 15 residuos en general, y finalmente cada modificación tiene asociado un artículo que lo respalda.

El resto de las modificaciones no tienen péptidos asociados, en la modificación ocurrida por iones de metal se tiene el ion involucrado.

Una proteína puede tener ninguna, una o más de una modificación. Un tipo de modificación al menos tiene una proteína asociada. Es decir una acetilación puede ocurrir en 3 posiciones diferentes de una proteína, esa misma proteína puede tener otras 5 posiciones que se ubiquitinen y puede haber proteínas que no sufran ninguna modificación. Puede suceder el caso que en una misma posición ocurran 2 o más modificaciones.

Los artículos tienen un pmid designado por la página de PubMed y una referencia donde aparece el autor o autores, el año de publicación, el nombre del artículo y otros datos. Las referencias de los artículos sólo cumplirán la función de dar información auxiliar de dónde encontrar la base experimental que se hizo para encontrar la modificación postraduccional en cuestión.
Un artículo tiene 1 o más modificaciones documentadas, una modificación tiene al menos un artículo.
Algunas modificaciones fueron adquiridas de la base de datos de UniProt (Universal Protein Resources), dichas registros no cuentan con la información del artículo del cual fueron adquiridos.

La información otorgada por las bases de datos en modificaciones no otorgan la información completa de nombres estándares puesto que cada base maneja un nombre estándar como mejor le convenga, se necesita también rescatar estos nombres.

Los pdbs tienen toda la información para representar las proteínas en 3D, contienen cada uno de los átomos obtenidos mediante experimentos de cristalografía, generalmente estos cristales puede tener ciertas regiones que no están bien formados, a estas regiones se les llama regiones desordenadas o inestables, esto debido al resultado de cambiar la forma nativa de la proteína, al hacerla pasar por ciertos procesos químicos.

Por eso los pdb especifican siempre los residuos que fueron cristalizados de manera adecuada, puede que un archivo pdb contenga más de una proteína u otras moléculas que no son de interés. No todas las proteínas tienen un cristal, esto

debido que aún no se ha investigado o ésta en ese proceso. Se deberá seguir un criterio particular para clasificar los pdbs que se tomarán en cuenta para almacenarlos en el servidor (más adelante se explicara a detalle). Los archivos pdbs tienen un artículo de publicación, aunque no todos, puesto que hay algunos que están siendo preparados para ser publicados.

Estas especificaciones se tomarán en cuenta para el diseño de la base de datos, se seguirá las reglas planteadas por la teoría, sin embargo, al final se harán validaciones que irán dirigidas a la programación y a la vistas de los usuarios (la práctica). En dado caso que no se sigua al pie de la letra la teoría de las bases de datos, siempre se justificara el porqué. En muchas ocasiones aunque la teoría dicte seguir ciertas reglas y patrones, no siempre es lo que conviene para resolver las necesidades del proyecto.

Aunque en los requerimientos se solicite algo determinado, muchas veces para hacer ese algo se requieren otras actividades, no es directo, esas otras actividades generalmente son parsers o algunos algoritmos reprogramables e incluso curación de la información manualmente, revision tras revision.

4.3.2 Identificación de entidades y relaciones.

Partiendo de los requerimientos mostrados, se pudieron identificar las siguientes entidades:

- ✓ Modificación: Aunque existan 14 modificaciones, se entenderá como modificación la entidad general de todas. Después se asignará para cada modificación sus atributos respecto a la información que se tiene del proyecto. La entidad modificación formará dos relaciones; una con la entidad Proteína y otra con la entidad Artículo.
- ✓ Artículo/Fuente: Entidad que representará los artículos que son la referencia tanto de los experimentos realizados para las modificaciones y para los pdbs. Tendrá como atributos su pmid, un identificador único y su referencia, en esta última se involucrará toda la cita del pdb, es decir autores o autor, título o tema, año, etc. No se considera necesario hacer entidades como Autor o Temas del artículo, puesto que para las consultas son irrelevantes. No se harán consultas por Autor y una modificación en especial, pues como se mencionó en los requerimientos sólo es referencial.
- ✓ Proteína: Esta entidad tendrá como atributos una clave única, su identificador universal, su secuencia lineal de aminoácidos, su identificador sgdid, el nombre estándar dado por el archivo fasta de SGD, su lista de alias y finalmente su descripción. La entidad proteína tendrá tres relaciones; con modificación, con pdb y con equivalencia.

- ✓ PDB: Entidad formada por los atributos de un archivo pdb, como son una identificador único y su nombre, se podría establecer que su nombre sea su identificador único, sin embargo para facilitar las búsquedas que se hagan más adelante, se marcará de forma independiente.
- ✓ Equivalencia: Esta entidad solo tendrá una relación, con proteína. Con la finalidad de tener información auxiliar que luego será utilizada para ciertas búsquedas, se establece esta nueva entidad. La entidad equivalencia es una entidad débil, puesto que no se describe a si misma con sus dos únicos atributos; un identificador único y el nombre estándar.
- ✓ Localización: Se considera como entidad y no como atributo de la entidad proteína, en caso contrario habría muchos valores nulos, porque como se menciona en los requerimientos no todas las proteínas tienen ubicación conocida y para evitar problemas de normalización más adelante, localización se establece como entidad débil de proteína.

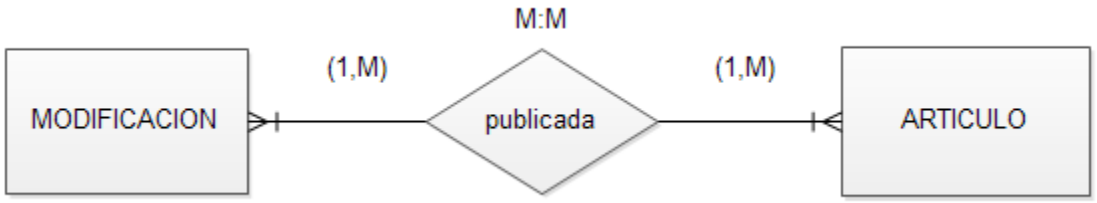
En algún caso también se podría establecer al nombre de las relaciones como el nombre de las entidades dadas. Por ejemplo, que la modificación no sea en sí una entidad, sino más bien una relación entre proteína y otra entidad que se llame residuo, pero para ser más representativo al área afín, se ha establecido el nombre de las entidades ya mostradas.

A continuación se muestran las relaciones con su respectivas cardinalidades, se utiliza al modelo entidad-relación de Peter Chen. Para todas las relaciones se ocupó el programa Edraw Max.

Relación Modificación-Proteína

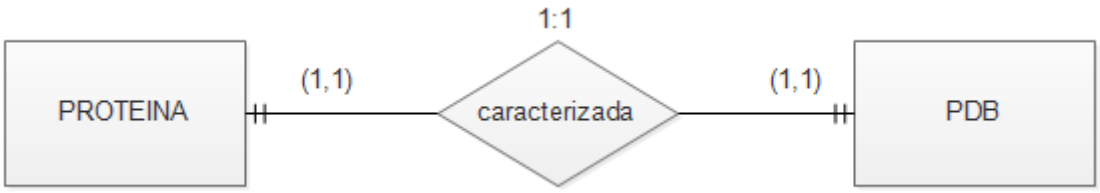
Una modificación tiene una proteína, una proteína puede tener muchas modificaciones. En los requerimientos se mencionó que no todas las proteínas tienen modificaciones (posible cardinalidad mínima igual a cero), en ese caso, las proteínas existirán en la base de datos pero cuando se haga la vista correspondiente, se mostrará sólo la información que de ellas se tenga. Cabe mencionar que se considera como registro único (tomado en cuenta para la cardinalidad) al conjunto de toda la información de la modificación, es decir, su posición, su residuo y el tipo de modificación, de no ser así, puede que la relación estrictamente hablando sea muchos a muchos, esto llevaría hacer tablas auxiliares.

Relación Modificación-Artículo



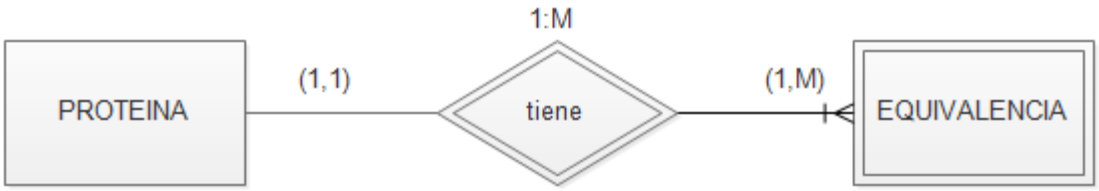
Una modificación puede tener muchos artículos y un artículo puede tener muchas modificaciones. Aunque en realidad esta relación debería tener este comportamiento, se verá más adelante que no es del todo conveniente.

Relación Proteína-PDB



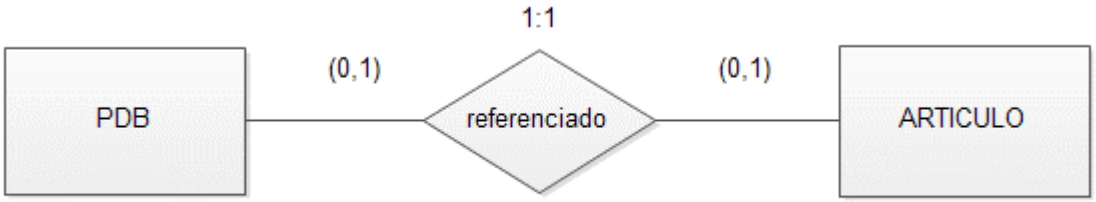
Una proteína tiene un solo pdb, y un pdb caracteriza sólo una proteína de interés, aunque esta relación no es el todo cierta, en realidad se sabe que una proteína puede que no tenga pdb o muchos y que un pdb puede tener muchas proteínas, lo que se hizo para transformar la relación a uno y sólo uno con uno y sólo uno, fue hacer una depuración de la información de todos los pdbs que caracterizan proteínas de levaduras, que son aproximadamente 2,800. Se ocuparon 4 parsers y 1 programa automático de descarga, hasta llegar la curación manual. Aunque ciertamente la gran mayoría de las proteínas no tendrán pdb, puesto que la lista de pdbs se redujo hasta 600, ese no afectará en la cardinalidad de 1 a 1, gracias a que la cardinalidad mínima no se toma en cuenta para la cardinalidad de la relación. Se puede decir con certeza que de esos 600 pdbs existen los residuos cristalizados que tienen modificaciones documentadas.

Relación Proteína-Equivalencia



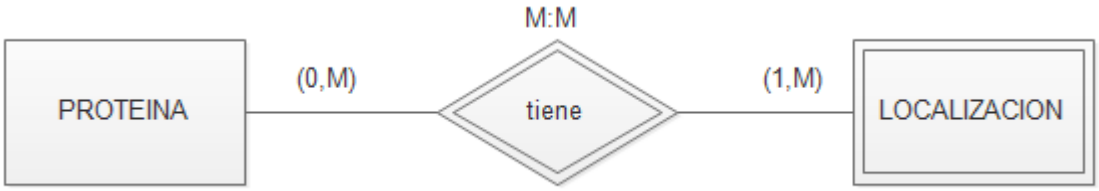
Una proteína tiene muchas equivalencias, una equivalencia solo está en una proteína. Como se mencionó la entidad equivalencia es una entidad débil, formando una relación débil con la entidad proteína.

Relación PDB-Artículo



Un PDB puede que este documentado en un artículo y puede que un artículo sea de un PDB. En particular esta relación también obligaría hacer tablas intermedias, una corrección en la cardinalidad sería lo más apropiado.

Relación Proteína-Localización



Una proteína puede que tenga una o muchas localizaciones y una localización puede estar en muchas proteínas, sin embargo para que exista esa localización tendrá que existir la proteína a la cual hace referencia.

En conclusión: las únicas relaciones que representa un problema es la de modificación-artículo y PDB-artículo, como se puede ver en la relación modificación-artículo la cardinalidad de la relación es muchos a muchos, eso más adelante en el modelo relacional implicaría la construcción de tablas intermedias por cada modificación. Es decir, siguiendo la regla cuando se presenta una regla persistente

(muchos a muchos) indica la creación de una entidad auxiliar que heredará las llaves primaras de cada entidad. En la práctica eso no sería provechoso de la forma en que se tiene la información y como se piensan hacer las consultas, resulta mejor trabajar con las querys y la programación para que la relación quede 1: M, este arreglo tendrá el mismo resultado como si se tuvieran las tablas auxiliares.

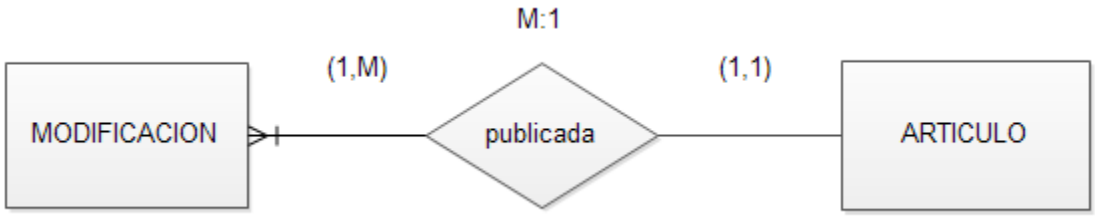
4.4 Modelo entidad relación.

Para la elaboración del modelo entidad relación, se reconsideraron las relaciones obtenidas anteriormente con la intención de hacer posibles mejoras que ahorrarán espacio en la base de datos y aumentar el tiempo de respuesta de las consultas realizadas.

4.4.1 Ajustes de cardinalidades.

Las relaciones proteínas-equivalencia, modificación-proteína, PDB-proteína y proteína-localización, están de cierta forma correctas para el manejo de la base de datos, en el caso de las otras dos, se hicieron ajustes haciendo ciertas consideraciones:

Ajuste en la relación Modificación-Artículo



Aunque el comportamiento ideal nos habla de una relación muchos a muchos, consideraremos esta nueva relación, donde una modificación puede estar documentada en un solo artículo, pero un artículo documenta muchas modificaciones, considerando a la modificación con todos sus atributos residuo, posición, péptidos (según sea el caso), etc. Es decir puede que se presente el caso donde una fosforilación de la proteína YAL001C cuya posición es 41, cuyo residuo es S y cuya serie de péptidos es FDLSDKKV está presente el artículo cuyo pmid es 22817900 y también en otro con la misma información, pero en cualquiera de los dos artículos, esa modificación es única puesto que los procedimiento de estadística y experimentación no fueron los mismos, aunque se trate de la misma modificación.

Esta consideración es del todo local, no global, por eso aún existe la posibilidad de seguir manteniendo la relación muchos a muchos, más adelante se remarcará la relevancia del porqué de esta consideración en el momento de elaboración el modelo relacional.

Relación PDB-Artículo

La cardinalidad de la relación se mantiene pero se modifica la cardinalidad mínima de la entidad artículo para evitar la construcción de tablas intermedias. En este caso se retomó solamente aquellos PDBs que están en la base de datos, ya que estos fueron considerados como útiles después de elaboración de parsers y curación manual. Entonces se puede establecer que: puede ser que un PDB tenga un artículo asociado o no, pero un artículo (que se consideró como útil) esta asociado a un PDB.

Esta consideración también es local, así como no se considera todo el universo de artículos de todos los organismos sino solo aquellos que sean de levadura y específicamente hablando de modificaciones postraduccionales.

4.4.2 Selección de atributos de las entidades.

Para la entidad modificación se determinaron los siguientes atributos a partir de las especificaciones dadas:

MODIFICACIONES					
Modificación	Atributos				
	Id	Posición	Péptidos	Residuo	Ion
Ubiquitinación	✓	✓	✓	✓	×
Fosforilación	✓	✓	✓	✓	×
Acetilación	✓	✓	✓	✓	×
Metilación	✓	✓	✓	✓	×
Succinilación	✓	✓	✓	✓	×
Oxidación	✓	✓	✓	✓	×
Metal	✓	✓	×	✓	✓

Glicosilación	✓	✓	×	✓	×
Calcio	✓	✓	×	✓	×
Nterminal	✓	✓	✓	✓	×
Nitrosilación	✓	✓	✓	✓	×
Disulfuro	✓	✓	×	✓	×
Lipidación	✓	✓	×	✓	×
Sitio Activo	✓	✓	×	✓	×

Tabla 4.2 Atributos para cada una de las tablas modificación

En la tabla anterior muestra los atributos que dispone cada modificación, aquellos atributos marcados por palomas blancas indican atributos no nulos, aquellos atributos marcados con palomas amarillas pueden ser nulos y cuando aparece un tache es que ese atributo se descarta. Aquellos atributos marcados con palomas rojas son las llaves primarias y únicas llaves candidatas, todas las palomas sin importar el color son la súper llave de las entidades.

PROTEINA	
Atributos	Valor
Id	✓
Identificador Universal	✓
Nombre Estándar	✓
SGDid	✓
Alias	✓
Secuencia	✓
Secuencia de regiones	✓
Información	✓

FUENTE(ARTICULO)	
Atributos	Valor
Id	✓
Referencias	✓
Pmid	✓

EQUIVALENCIAS	
Atributos	Valor
Id	✓
Nombre Estándar	✓

LOCALIZACION	
Atributos	Valor
Id	✓
Ubicaciones	✓

PDB	
Atributos	Valor
Id	✓
Pmid	✓

Tabla 4.3 Atributos para las otras entidades

Los atributos mostrados en estas tablas son la súper llave de la entidad, la paloma de color rojo es la llave primaria, las palomas azules son aquellos atributos que pueden ser nulos, las palomas marcadas en blanco son las llaves candidatas. La paloma verde habla de un atributo utilizado como discriminante. Sólo habrá una consideración semántica con respecto al atributo de secuencia de regiones desordenadas, ya que estas podrán tener un valor de 0's o de 1's dentro de su campo.

Una vez conocidos los atributos y las relaciones con sus respectivas cardinalidades, el siguiente paso a seguir es elaborar el diseño conceptual a través del modelo entidad relación, este paso definirá el modelo relacional para dar por terminado el modelo lógico de la base de datos.
Los ajustes hechos con anterioridad sobre las relaciones se hicieron de una vez en este nivel de diseño para no irlos arrastrando, aunque cabe mencionar que se pueden realizar en el momento del diseño físico y la normalización.

En el modelo entidad relación que se muestra en la siguiente hoja; el nombre dado a las relaciones son nombres compuestos o verbos que reflejan la presencia de la relación, cuando son nombres compuestos se le suele agregar un guion bajo para indicar las tablas de procedencia. Todas las llaves primarias están señaladas con un óvalo azul. Las entidades de la parte superior son aquellas que tienen péptidos como atributos y específicamente las entidades marcadas con rojo son las más grande de la base de datos, debido a que sus registros representarán más del 92% de los totales.

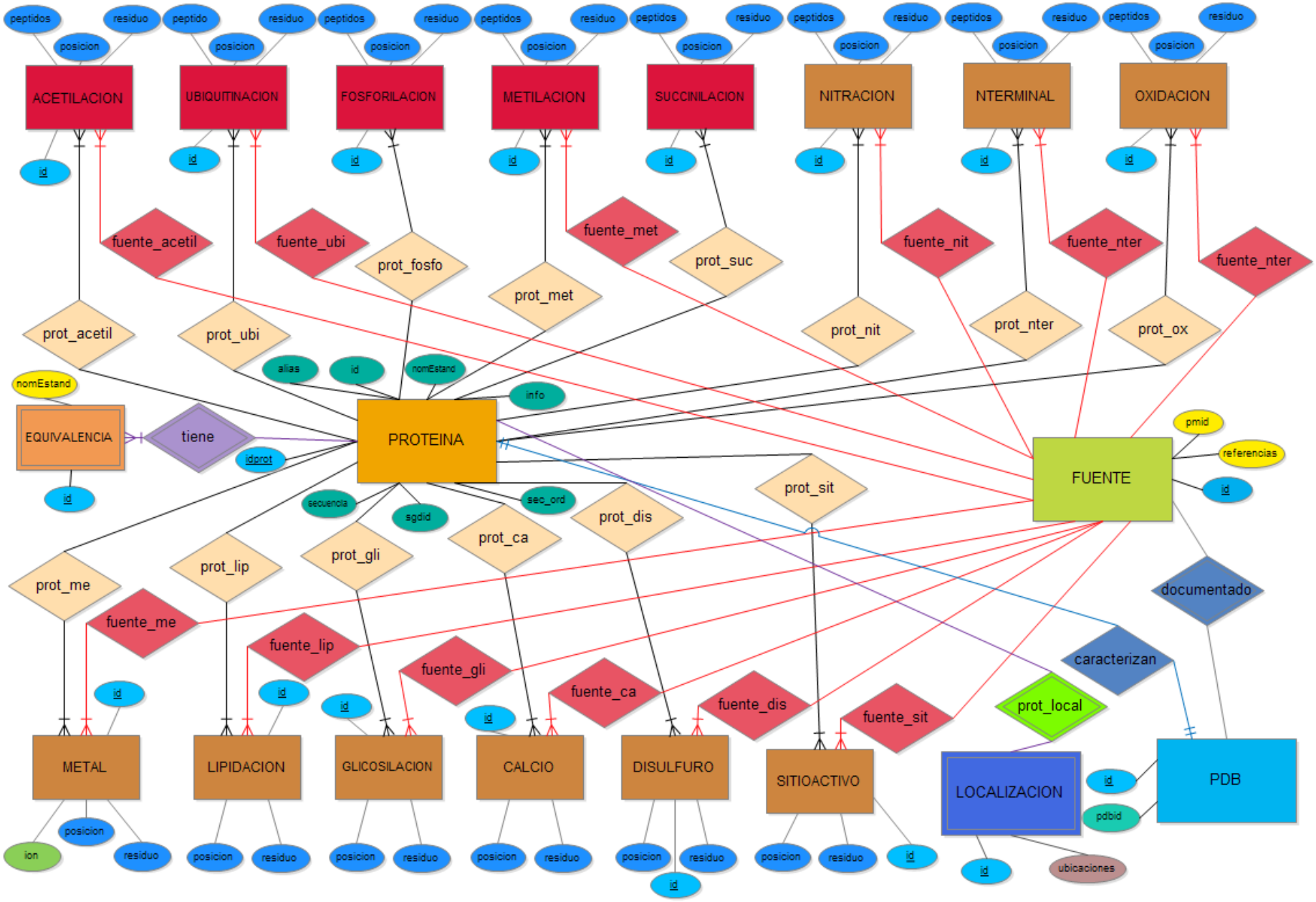


Fig. 4.4 Modelo Entidad- Relación

4.5 Diseño Lógico.

Se aplicaron las reglas del modelo relacional para la formación del diseño lógico, así como la presentación de los tipos de datos de los atributos. Enseguida se hablará del proceso de normalización que es de vital importancia para el desarrollo del modelo físico y el tratamiento de la información.

4.5.1 Construcción del modelo relacional, revisión de relaciones.

Se aplicaron las siguientes reglas:

Cardinalidad		Regla
M:M		Se crea una nueva entidad que hereda las dos llaves primarias de cada entidad que participan en la relación
1:M o M:1		La llave primaria de la entidad 1 pasa como llave foránea a la otra entidad.
Entidad1	Entidad2	En caso de presentar la cardinalidad 1:1
0,1	0,1	Se crea una nueva entidad que hereda las dos llaves primarias de cada entidad que participan en la relación
0,1	1,1	La llave primaria de la entidad que tiene como cardinalidades mínima y máxima 1 hereda su llave como llave foránea a la otra entidad
1,1	1,1	La llave primaria de cualquier entidad pasa a la otra entidad, el diseñador tendrá la elección de decidir donde se obtenga la mayor funcionalidad.

Tabla 4.4 Reglas de transformación de modelo entidad relación a relacional

En el modelo relacional todas entidades fuertes pasan a ser relaciones, éstas a su vez tablas de dos dimensiones, aplicando las reglas ya mencionadas y las referidas a entidades débiles y relaciones débiles (en el caso de la entidad equivalencias) se obtuvo el modelo relacional. Cabe mencionar que no hubo la necesidad del manejo de atributos multivaluados, todos los atributos de la base son únicos respecto a su presencia en una proteína en los estudios de biología molecular.

A continuación se muestran los tipos de datos de cada atributo:

Atributo	**Nombre en la B.D.**	**Entidad de origen**	**Tipos de dato**	**Nulidad**
Id	Id	Modificación	Number(11)	NOT NULL
Id	Id	Fuente	Number(11)	NOT NULL
Id	Id	PDB	Number(11)	NOT NULL
Id	Id	Equivalencia	Number(11)	NOT NULL
Id	Id	Proteína	Number(11)	NOT NULL
Id	Id	Localización	Number(11)	NOT NULL

Identificador Universal	idprot	Proteína	Varchar(10)	NOT NULL
Posición	posicion	Modificación	Number(11)	NOT NULL
Residuo	residuo	Modificación	Varchar(1)	NOT NULL
Secuencia	sequence	Proteína	Text	NOT NULL
Secuencia de regiones	sequence_order	Proteína	Text	NULL
Sgdid	sgdid	Proteína	Varchar(15)	NOT NULL
Nombre Estándar	standard_name	Proteína	Varchar(30)	NOT NULL
Alias	aliases	Proteína	Text	NULL
Información	header	Proteína	Text	NOT NULL
Péptidos	peptidos	Modificación	Varchar(50)/ Varchar(45)/ Varchar(32)*	NULL
Ion	ion	Modificación (Metal)	Varchar(25)	NOT NULL
Nombre Estándar	standard_name	Equivalencia	Varchar(20)	NOT NULL
PDBid	pdb_file	PDB	Varchar(10)	NOT NULL
Pmid	Pmid	Fuente	Varchar(20)	NOT NULL
Referencias	fuente	Fuente	Text	NOT NULL
Ubicaciones	ubicaciones	Localización	Varchar(50)	NOT NULL

Tabla 4.5 Tipos de datos de atributos

*depende de la modificación en cuestión.

Es importante mencionar que el nombre de la mayoría de los atributos se cambió a inglés con la intención de hacer universal la información contenido en esta base de datos, atributos como residuo y peptidos también se le cambia el nombre a inglés pero en la programación web. El nombre de las modificaciones también se cambió a inglés, por cuestiones de comodidad se dejará el nombre en español para la definición del modelo relacional, más adelante en el modelo físico se verán los nombres definitivos.

Los tipos de datos asignados para cada atributo se definieron por el funcionamiento que estos cumplen, por ejemplo: el tipo de dato para el idprot es Varchar (10), debido a que como se puede ver en los requerimientos, el nombre de una proteína está formado por 7 caracteres alfanuméricos o bien de 9, cuando éste tien un guion y una letra.

La aparición de proteínas con estos nombres son peculiares, incluso también son estandarizados siguiendo una estricto control, pero como también se puede ver los idprot puede tener 5 o 4 caracteres alfanuméricos si son ORFs mitocondriales o de plásmido, por lo que se decidió redondear el número de caracteres más grande que puede presentar el idprot.
Es necesario mencionar que el atributo ubicaciones es Varchar (150) por mayor comodidad, el valor del atributo hablará de las ubicaciones que tiene la proteína separada por comas.

Cuando una llave foránea se presente en el modelo relacional se adiciona al principio de su nombre el nombre de la entidad de donde provine.

Modificaciones

ACETILACION= {id (PK) NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL, peptidos VARCHAR (32), residuo VARCHAR (1) NOT NULL, posicion NUMBER (11) NOT NULL, source_id (FK) NUMBER (11) NOT NULL}

FOSFORILACION= {id (PK) NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL, peptidos VARCHAR (50), residuo VARCHAR (1) NOT NULL, posicion NUMBER (11) NOT NULL, source_id (FK) NUMBER (11) NOT NULL}

UBIQUITINACION= {id (PK) NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL, peptidos VARCHAR (50), residuo VARCHAR (1) NOT NULL, posicion NUMBER (11) NOT NULL, source_id (FK) NUMBER (11) NOT NULL}

METILACION= {id (PK) NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL, peptidos VARCHAR (32), residuo VARCHAR (1) NOT NULL, posicion NUMBER (11) NOT NULL, source_id (FK) NUMBER (11) NOT NULL}

SUCCINILACION= {id (PK) NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL, peptidos VARCHAR (32), residuo VARCHAR (1) NOT NULL, posicion NUMBER (11) NOT NULL, source_id (FK) NUMBER (11) NOT NULL}

NITRACION= {id (PK) NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL, peptidos VARCHAR (32), residuo VARCHAR (1) NOT NULL, posicion NUMBER (11) NOT NULL, source_id (FK) NUMBER (11) NOT NULL}

NTERMINAL= {id (PK) NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL, peptidos VARCHAR (45), residuo VARCHAR (1) NOT NULL, posicion NUMBER (11) NOT NULL, source_id (FK) NUMBER (11) NOT NULL}

OXIDACION= {id (PK) NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL, peptidos VARCHAR (32), residuo VARCHAR (1) NOT NULL, posicion NUMBER (11) NOT NULL, source_id (FK) NUMBER (11) NOT NULL}

METAL= {id (PK) NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL, ion VARCHAR (25) NOT NULL, residuo VARCHAR (1) NOT NULL, posicion NUMBER (11) NOT NULL, source_id (FK) NUMBER (11) NOT NULL}

LIPIDACION= {id (PK) NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL, residuo VARCHAR (1) NOT NULL, posicion NUMBER (11) NOT NULL, source_id (FK) NUMBER (11) NOT NULL}

GLICOSILACION= {id (PK) NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL, residuo VARCHAR (1) NOT NULL, posicion NUMBER (11) NOT NULL, source_id (FK) NUMBER (11) NOT NULL}

CALCIO= {id (PK) NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL, residuo VARCHAR (1) NOT NULL, posicion NUMBER (11) NOT NULL, source_id (FK) NUMBER (11) NOT NULL}

DISULFURO= {id (PK) NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL, residuo VARCHAR (1) NOT NULL, posicion NUMBER (11) NOT NULL, source_id (FK) NUMBER (11) NOT NULL}

SITIOACTIVO= {id (PK) NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL, residuo VARCHAR (1) NOT NULL, posicion NUMBER (11) NOT NULL, source_id (FK) NUMBER (11) NOT NULL}

Otras entidades

PROTEINA= {id (UNIQUE) NUMBER (11), idprot (PK) VARCHAR (10) NOT NULL, sequence TEXT NOT NULL, sequence_order TEXT, sgdid VARCHAR (15) NOT NULL, standard_name VARCHAR (30) NOT NULL, aliases TEXT, header TEXT NOT NULL}

EQUIVALENCIA= {[id NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL] (PK) standard_name VARCHAR (20) NOT NULL}
SOURCE= {id (PK) NUMBER (11), pmid VARCHAR (20) NOT NULL, fuente TEXT NOT NULL}

PDB= {id (PK) NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL, pdbid VARCHAR (10) NOT NULL, source_id (FK) NUMBER (11) NOT NULL}

LOCALIZACION= {[id NUMBER (11), proteina_idprot (FK) VARCHAR (10) NOT NULL] (PK), ubicaciones VARCHAR (150) NOT NULL}

FUENTE= {id (PK) NUMBER (11), pmid VARCHAR (20) NOT NULL fuente TEXT NOT NULL}

Las entidades de modificaciones y PDB adquieren las llaves primarias de las entidades de fuente y proteína, las consideraciones y ajustes que se hicieron con anterioridad repercutieron en este cambio. La entidad equivalencia adquiere la llave primaria de proteína como llave foránea, ésta además del discriminante de la

entidad se transforma también llave primaria, esto debido a que es una entidad débil, lo mismo sucede con la entidad localización.

Todas las demás entidades quedan tal cual con sus atributos, y se transforman en relaciones, que finalmente dará lugar a las tablas. Una vez terminado el diseño lógico, el siguiente paso es la normalización de la base de datos.

4.6 Normalización de la base de datos.

La normalización es un proceso que se lleva a cabo dentro del diseño de la base de datos para evitar problemas a futuro, evitando las anomalías y redundancias de datos:

En anomalías:

- ✓ Inserción: Tener datos iguales en varias tablas.
- ✓ Borrado: Perdida de datos al borrar datos relacionados en otras tablas.
- ✓ Actualización: Inconsistencia de los datos como resultado de datos redundantes y actualizaciones parciales.

Y finalmente la redundancia de datos, como el síntoma de tener información replicada de forma exagerada; el problema de la redundancia de datos no es tanto el espacio que ocupa en la base de datos, sino que no permite tener la información lo más condensada posible.

Para esta parte se tomará como ejemplo una tabla de modificación, ésta nos llevará a otras tablas; proteína y fuente. Al final se analizarán las tablas de PDB, equivalencia y localización

4.6.1 Primera Forma Normal de las tablas modificación, proteína y fuente.

En este primer grado de normalización lo primero que hay que reconocer es que todos los valores de los atributos son atómicos, es decir, un atributo cuyo valor no se puede descomponer en partes más pequeñas, y principalmente que:

- Todos los atributos son dependientes de su llave primaria

Siendo muy estricto con este punto, se dice que la tabla tiene que ser isoforma, aunque esta definición fue del todo desarrollada por Chris Date, se evaluarán con los aspectos ya mencionados; la atomicidad de los datos y la dependencia de la clave primaria. Finalmente se dice que una base de datos está en primera forma normal cuando todas las tablas están en primera forma normal.

Se cuenta con los siguientes registros de la modificación acetilación:

idprot	peptidos	residuo	posicion	source_id
YALO12W	AALSANKIAEFLA	K	267	1
	NYDVVLKQHRDAL	K	298	
	KVTDIQKVADLIK	K	160	
	GAEAASKFASSTR	K	322	
	VNYPGLKTHPNYD	K	288	

Esta tabla pasaría en primera forma normal como:

id	idprot	peptidos	residuo	posicion	source_id
1	YALO12W	AALSANKIAEFLA	K	267	1
2	YALO12W	NYDVVLKQHRDAL	K	298	1
3	YALO12W	KVTDIQKVADLIK	K	160	1
4	YALO12W	GAEAASKFASSTR	K	322	1
5	YALO12W	VNYPGLKTHPNYD	K	288	1

Todos los datos son atómicos y existe un identificador (la llave primaria) del cual dependerán todos los atributos de la tabla, es por eso que se definió desde el principio un id para cada modificación, se dice entonces que esta tabla ya está en primera forma normal.
De la forma que en se almacenó la información, fue más fácil utilizar un id autoincremento de esta manera se llevó un cierto orden.

4.6.2 Segunda Forma Normal en la tabla modificación, proteína y fuente.

Antes de trabajar con la segunda forma normal, se tiene que entender bien el concepto de determinante y de relación. En primer lugar un determinante es aquel atributo cuyo valor depende otro valor de otro atributo, a esto se le conoce como dependencia funcional. Se puede dar el caso de que el determinante es compuesto, conocido como llave compuesta. Para que una base de datos esté en segunda forma normal tiene que cumplir con:

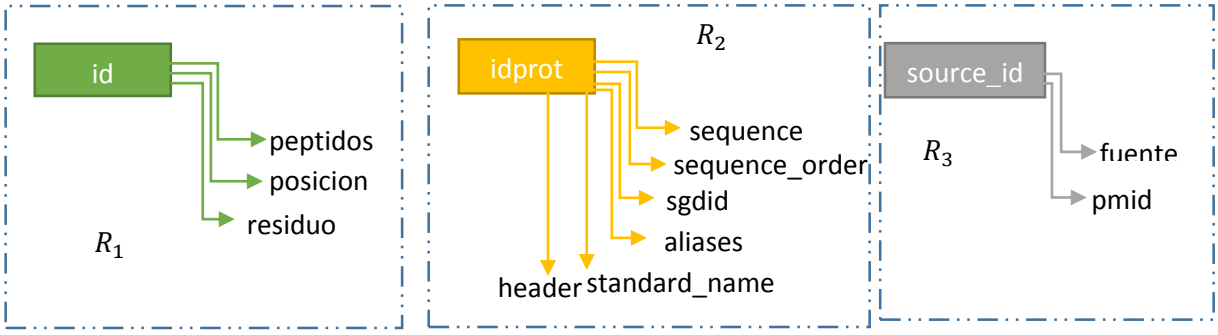
- Estar en primera forma normal.
- Todo atributo que no sea llave, depende funcionalmente de la clave primaria en modo completo.

Aquí surge un detalle de diseño, se puede plantear que el atributo clave en donde recaen todas dependencias funcionales de los atributos es el id, sin embargo del idprot depende todo, dependen los péptidos, puesto que son un fragmento de aminoácidos de la secuencia de la proteína, el residuo es el aminoácido modificado de ese fragmento y la posición es el lugar en la secuencia donde ocurre. Por lo que se establecerá de la siguiente manera: el id efectivamente será la clave primaria de

la tabla, de manera independiente se mostrará tanto como el idprot y el source_id, porque idealmente éste es el comportamiento. Cabe mencionar que se está trabajando con una tabla que se construyó en el diseño lógico y conceptual y los atributos heredados como llaves foráneas ya fueron considerados como llaves primarias de sus tablas correspondientes, entonces es aquí donde aparecen las otras dos tablas, tanto la de proteína como la de fuente.

idprot	sequence	sequence_order	sgdid	standard_name	aliases	header
YAL011W	MPAVLRTRSKESSIEQKPASRTRTRSRRGKRGRDDDDDDDDEESDDAYDEVGNDYDEYASRAKLATNRPFEIVAGLPASVELPNYNSSLTHPQSIKNSGVLYDSLVSSRRTWVQGEMFELYWRRPKKIVSESTPAATESPTSGTIPLIRDKMQKMCDCVMSGGPHTFKVRLFILKNDKIEQKWQDE….		S000000009	SWC3	SWC1	Protein of unknown function; component of the SWR1 complex, which exchanges histone variant...
YAL012W	MTLQESDKFATKAIHAGEHVDVHGSVIEPISLSTTFKQSSPANPIGTYEYSRSQNPNRENLERAVAALENAQYGLAFSSGSATTATILQSLPQGSHAVSIGDVYGGTHRYFTKVANAHGVETSFTNDLLNDLPQLIKENTKLVWIETPTNPTLKVTDIQKVA…		S000000010	CYS3	cystathionine gamma-lyase CYS3 CYI1 FUN35 STR1	Cystathionine gamma-lyase; catalyzes one of the two reactions involved in the transsulfuration…
YAL013W	MSQQTPQESEQTTAKEQDLDQESVLSNIDFNTDLNHNLNLSEYCISSDAGTEKMDSDEEKSLANLPELKYAPKLSSLVKQETLTESLKRPHEDEKEAIDEAKKMKVPGENEDESKEEEKSQELEEAIDSKEKSTDARDEQGDEGDNEEENNEEDNENENEHTAPPALVMPSPIEMEEQRMTALKEITDIEYKFAQLRQKLYDNQLVRLQTELQMCLE…		S000000011	DEP1	FUN54 Rpd3L histone deacetylase complex subunit DEP1	Component of the Rpd3L histone deacetylase complex; required for diauxic shift-induced histone H2B deposition onto rDNA genes…

Id	pmid	fuente
1	22865919	Henriksen, P; Wagner, SA; Weinert, BT; Sharma, S; Bacinskaja, G; Rehman, M; Juffer, AH; Walther, TC; Lisby, M; Choudhary, C (2012) Proteome-wide analysis of lysine acetylation suggests its broad regulatory scope in Saccharomyces cerevisiae. Mol. Cell Proteomics. 11:(11) 1510-22.
2	24489116	Weinert, BT; Iesmantavicius, V; Moustafa, T; Scholz, C; Wagner, SA; Magnes, C; Zechner, R; Choudhary, C (2014) Acetylation dynamics and stoichiometry in Saccharomyces cerevisiae. Mol. Syst. Biol.. 10:() 716.



Como se puede observar la llave primaria idprot retoma dependencia funcional con otros atributos de la tabla de origen, pasa lo mismo con source_id. También las tablas de proteína (tabla amarilla) y fuente (tabla gris) ya están en primera forma normal puesto que los atributos son atómicos y estos dependen de su llave primaria. El campo sequence_order se queda en blanco porque en un futuro se llenará con la información específica que se tenga en ese momento.

En la tabla de modificaciones se puede observar que el idprot se repite tantas veces como modificaciones tenga, la teoría de normalización platea precisamente que no debería existir esta redundancia de datos, porque lo que lleva a modificar la tabla de proteína agregando un id, como se puede observar en la siguiente tabla:

id	idprot	sequence	sequence_order	sgdid	standard_name	aliases	header
1	YAL011W	MPAVLRTRSKESSIEQKPAS RTRTRSRRGKRGRDDDDDD DDEESDDAYDEVGNDYDEY ASRAKLATNRPFEIVAGLPAS VELPNYNSSLTHPQSIKNSG VLYDSLVSSRRTWVQGEMF ELYWRRPKKIVSESTPAATE SPTSGTIPLIRDKMQKMCDC VMSGGPHTFKVRLFILKNDKI EQKWQDE….		S000000009	SWC3	SWC1	Protein of unknown function; component of the SWR1 complex, which exchanges histone variant...
2	YALo12W	MTLQESDKFATKAIHAGEHV DVHGSVIEPISLSTTFKQSSP ANPIGTYEYSRSQNPNRENL ERAVAALENAQYGLAFSSGS ATTATILQSLPQGSHAVSIGD VYGGTHRYFTKVANAHGVE TSFTNDLLNDLPQLIKENTKL VWIETPTNPTLKVTDIQKVA …		S000000010	CYS3	cystathionine gamma-lyase CYS3 CYI1 FUN35 STR1	Cystathionine gamma-lyase; catalyzes one of the two reactions involved in the transsulfuration…
3	YAL013W	MSQQTPQESEQTTAKEQDL DQESVLSNIDFNTDLNHNLN LSEYCISSDAGTEKMDSDEE KSLANLPELKYAPKLSSLVK QETLTESLKRPHEDEKEAID EAKKMKVPGENEDESKEEE KSQELEEAIDSKEKSTDARD EQGDEGDNEEENNEEDNEN ENEHTAPPALVMPSPIEMEE QRMTALKEITDIEYKFAQLRQ KLYDNQLVRLQTELQMCLE …		S000000011	DEP1	FUN54 Rpd3L histone deacetylase complex subunit DEP1	Component of the Rpd3L histone deacetylase complex; required for diauxic shift-induced histone H2B deposition onto rDNA genes…

Y teóricamente ese id pasaría a formar la llave primaria de la tabla proteína, esto también llevaría a modificar el contenido de la tabla modificación, sin embargo, no se realizará de esta forma. Se considera como llave primaria al identificador de la

proteína, y no al id, aunque se dejó un id autoincremental en la tabla, para que las tablas sean uniformes.

Discusión:

Se considera al idprot como llave primaria y no al id, porque es único y universal, las comunidades de investigadores y científicos en el área de biología molecular y principalmente del estudio de proteínas de levadura, han estandarizado este nombre (se puede ver en los requerimientos), aunque se repita la información en la tabla de modificación y aparezca tantas veces como modificaciones tenga, se pondrá el idprot para:

- Tener un mejor control de los datos, esto es la seguridad de que la proteína de la que se dice ser modificada realmente sea esa.
- El hecho de colocar un id en lugar del nombre de la proteína, también implicaría la repetición de ese número tanto como modificaciones se tenga, no habría diferencia entre repetir un número al repetir el idprot, podría considerarse incluso como una redundancia de datos controlada, esto pesando en el tiempo de respuesta de las consultas solicitadas.
- Se mantendrá el id de la tabla proteína por razones uniformidad con todas las tablas, sin embargo el determinante de la relación seguirá siendo el idprot.

Por lo que se dice que las tres tablas están en la segunda forma normal, las relaciones quedarían así:

$R_1 = \{id, peptidos, residuo, posicion\}$
$R_2 = \{idprot, sequence, sequence\_order, sgdid, standard\_name, aliases, header\}$
$R_3 = \{source\_id, pmid, fuente\}$

Están en primera forma normal y existe dependencia funcional entre el determinante (el primer atributo a la izquierda) con el resto de los atributos.

4.6.3 Tercera y Cuarta Forma Normal en la tabla modificación, proteína y fuente.

Para ahora aplicar la tercera forma normal, se debe entender el concepto de dependencia transitiva, esta condición establece que si un atributo A es determinante de un atributo B, y que si un atributo C es dependiente de un atributo B, entonces C depende transitivamente de A. Esto puede decir que un atributo pudiera llegar a depender de un atributo no clave. Para poder decir que una tabla está en 3FN se tiene que cumplir que:

- La tabla está en segunda forma normal.
- Y que no contiene dependencias transitivas.

Aplicando estas nociones a las tres tablas anteriores se llegaría a las relaciones obtenidas anteriormente, claro que se pudiera pensar que el atributo residuo de la tabla proteína y la posición son atributos calculados que dependen de los péptidos, incluso los péptidos también se pudieran llegar a obtener de la secuencia original de aminoácidos de la proteína. Sin embargo son datos que están en artículos de manera explícita y obtenidos de manera experimental, por lo que dependencias transitivas no existen.

Todos los atributos no claves dependen de un atributo clave, tal y como lo muestra el esquema de normalización anterior. Gracias a la buen diseño del modelo entidad relación y relacional, esta parte no llevó a un gran conflicto. Por lo que se concluye que las tres tablas están en tercera forma normal.

Existe una forma normal intermedia entre la tercera y la cuarta, que es conocida como la forma normal de Boyce-Codd (FNBC), y existe para lidiar con las dependencias funcionales no triviales, aquellas que no son eliminadas en la 3FN. Estas dependencias funcionales no triviales se llaman así por su ocurrencia en atributos como llaves candidatas y además son determinantes, la forma para eliminar éstas sólo depende de formar un sólo determinante en cada relación. En las tres tablas solo existe un único determinante por lo que las tres relaciones están en forma normal de Boyce-Codd.

Para la cuarta forma normal se necesita recurrir al concepto de dependencia multivalorada que dice:

Sean tres atributos A, B y C de una relación R, se dice que el atributo B es multidependiente de A, si y solamente si en todo valor válido de R, el conjunto de valores de B coincide con un valor par de (A, C), donde solo depende de A y es independiente de C.
A partir de esta definición se puede ver que las tres relaciones están también en cuarta formal normal, puesto que no existe dependencias multivaloradas o multivaluadas. En muchas ocasiones se permite de manera controlada la redundancia de datos multivaluados de un conjunto de valores de B que es dependiente de A, siempre y cuando B sea un subconjunto de valores de A, en esta tabla esto no se presentará. Al colocar un id en la tabla modificación como llave primaria y también utilizar una única llave primaria en las tablas de proteína y fuente, se reduce la existencia de dependencias multivaluadas.

Pueden existir muchos valores del atributo B sin embargo estarán asociados a un identificador único, como por ejemplo: En la tabla proteína existirán *n* secuencias (atributo B) y *n* sgdid (atributo C) sin embargo para cada secuencia existiría un idprot único (atributo A) del cual dependa, y para cada sgdid un idprot (atributo A), pero ese sgdid y esa secuencia solo estarán en esa proteína y no en otra, eliminando por completo la idea de atributos multivaluados.

En el caso de la tabla de modificación ocurre algo parecido, por ejemplo si se tiene:

id	idprot	peptidos	residuo	posicion	source_id
1	YALO12W	AALSANKIAEFLA	K	267	1
2	YALO12W	NYDVVLKQHRDAL	K	298	1
3	YALO12W	KVTDIQKVADLIK	K	160	1
4	YALO12W	GAEAASKFASSTR	K	322	1
5	YALO12W	VNYPGLKTHPNYD	K	288	1
6	YALO12W	VNYPGLKTHPNYD	K	288	2
7	YALO12W	AALSANKIAEFLA	K	267	2
8	YALO12W	KVTDIQKVADLIK	K	160	2

Si se sigue el esquema de normalización, donde el id de la tabla modificación es determinante de los péptidos, del residuo y de la posición. Se puede observar en esta tabla que tres posiciones se repiten: 160, 267 y 288 con mismos peptidos y mismos residuos (en acetilación el residuo siempre es Lys (K)), sin embargo la source_id cambia en uno es el 1 y en el otro 2. El simple hecho de que vengan de artículos distintos es un indicio y elemento discriminatorio y suficiente para repetir los datos, esta redundancia será permisiva en todas las modificaciones.

Discusión: Como se mencionó con anterioridad no todos los artículos siguen las mismas reglas de clasificación para considerar si un residuo es modificado o no, de hecho para llegar a esa conclusión se tuvo que haber repetido el evento una serie de veces y con el manejo de la estadística llegan a la afirmación y la certeza de que hubo una modificación postraduccional. Sin embargo aunque esa redundancia sea permisiva es válida para las necesidades del proyecto por:

- El manejo del id como llave primaria (atributo A) es determinante de todos los atributos; posición (atributo B) y atributo péptidos (C), aunque el atributo B sea independiente del atributo C, esa posición no volverá a parecer en el mismo valor de determinante otra vez, sino en otro, y automáticamente se rompe la regla para la existencia de dependencia multivariada.
- El manejo de consultas han demostrado ser rápidas con las consideraciones hechas, de otra forma sería más tardado porque la búsquedas no involucrarían una sola tabla sino dos o más.

Por lo demás, se puede decir que las tres tablas están en cuarta forma normal y última. Cabe mencionar que existe una quinta forma normal pero no es muy aplicada, porque rara vez una cuarta forma normal tampoco es quinta forma normal.

4.6.4 Normalización de las tablas equivalencias, PDB y localización.

Faltaría por normalizar dos últimas tablas, sólo se aplicará la teoría y se darán las razones concernientes a su cumplimento o corrección de posibles errores (Tabla 4.5).

En PDB:

id	idprot	pdb_file	pmid_3d
1	YDR390C	3ONH	525
2	YDR530C	4I5T	526
3	YLR208W	3JRP	527
4	YG252W	1YGH	528

En Equivalencias:

id	idprot	standard_name
1	YMR056C	AAC1
2	YBR085W	AAC3
3	YJR155W	AAD10
4	YNL331C	AAD14

En Localización:

id	idprot	Ubicaciones
1	YHR055C	cytoplasm
2	YPR161C	nucleus
3	YOL138C	cytoplasm, vacuole
4	YDR295W	cytoplasm, nucleus

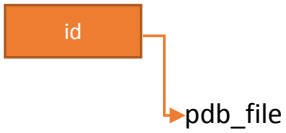
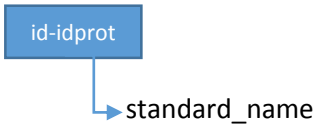
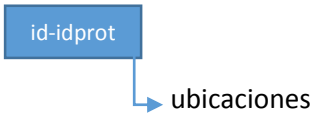
Valores de atributos atómicos		
✓	✓	✓
Todos los atributos son dependientes a su llave primaria		
✓	✓	✓
En 1FN		
✓	✓	✓
Determinante: id	Determinante: id	Determinante: id
Dependencia funcional en el determinante		
✓	✓	✓
En 2FN		
✓	✓	✓
Relaciones		
Determinante: id	Determinante compuesto: id y idprot	Determinante compuesto: id y idprot
$\boldsymbol{R_4 = \{id, pdb\_file\}}$	$R_5 = \{(id, idprot), standard\_name\}$	$R_6 = \{(id, idprot), ubicaciones\}$
		
Existencia de depedencias transitivas		
×	×	×
En 3FN		
✓	✓	✓
En forma Normal de Boyce-Codd		
✓	✓	✓
Existencia de depedencias multivaluadas		
×	×	×
En 4FN		
✓	✓	✓

Tabla 4.5 Normalización de equivalencias, localización y PDB.

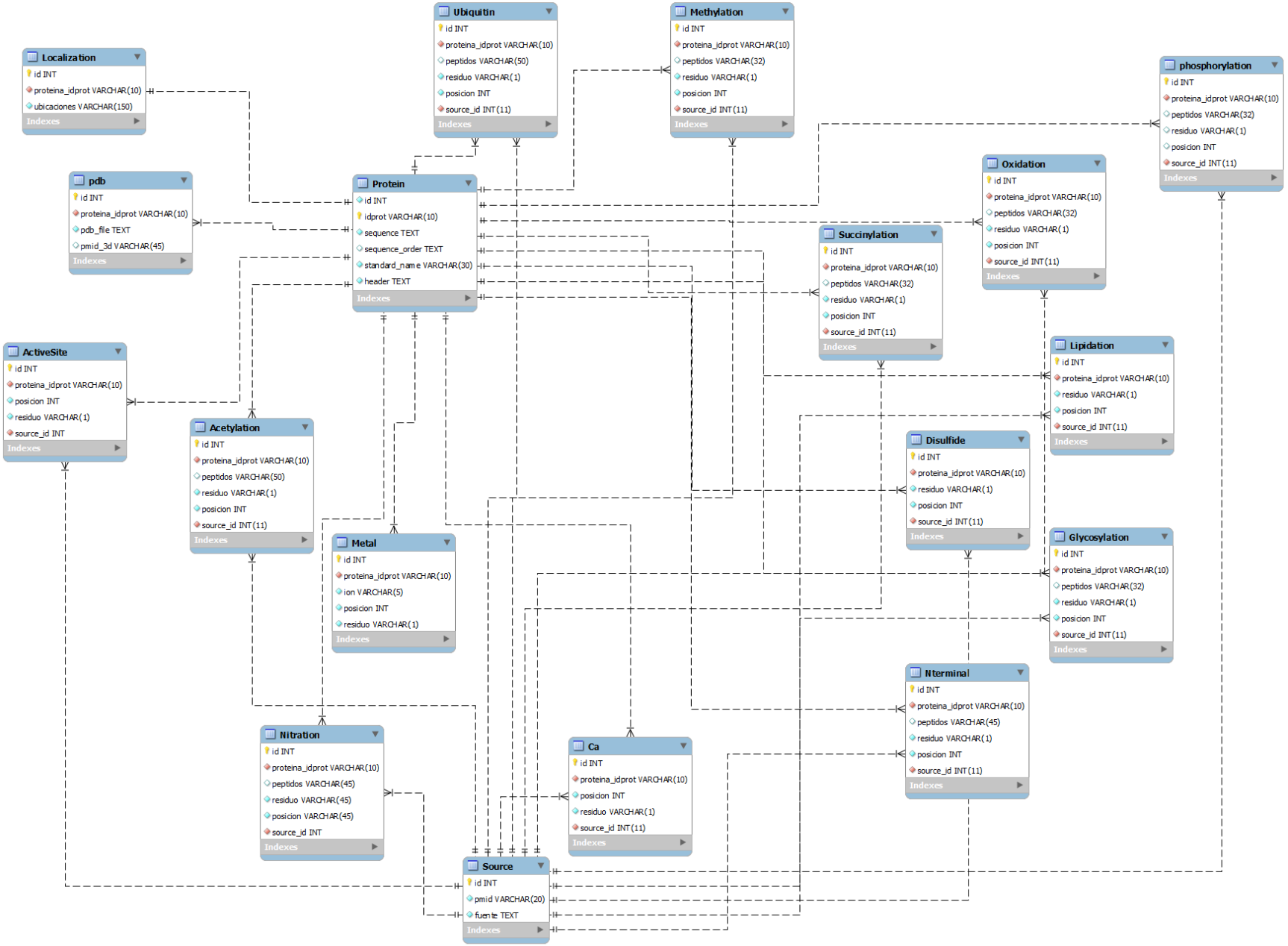
Los determinantes compuestos o también llamados llaves compuestas, en las tablas localización y equivalencias son así porque las llaves primarias de cada entidad están conformada también por la llave foránea que adquieren de su entidad fuerte que es Proteína. No hay ningún problema que se considere de esta forma puesto que el id e idprot son únicos, no habrá otra proteína con el mismo id que tenga las mismas localizaciones y de igual forma no habrá un id e idprot que tengan mismo nombre estándar.

4.7 Diseño Físico.

El siguiente paso a realizar es el diseño físico de la base de datos, mediante el modelo relacional obtenido en el diseño lógico, el diseño físico a diferencia del diseño lógico depende mucho del SGDB, como se mencionó al principio de este capítulo el manejador de bases de datos para este proyecto fue MySQL por lo que se los tipos de datos y modos de acceso a la base irán dirigidos a éste.
En la siguiente página se puede observar el modelo físico, obtenido después de todas las normalizaciones realizadas y los pasos ya planteados anteriormente. Los nombres de las tablas fueron cambiados a inglés sin embargo los atributos mantienen su nombres en español. Éstos últimos se mantendrán en la nivel interno de esta forma, sin embargo, en el momento de la visualización en la página web también se mostrarán en inglés.

El diseño físico fue hecho a través de la aplicación de Workbench de MySQL, que permite crear modelos para después convertirlos en tablas (Fig 4.5).

Fig. 4.5 Modelo físico de la base de datos



4.8 Algoritmos de programación para parseo y depuración de la información de modificaciones postraduccionales. Manejo de BioPerl y Python.

Una vez creado el modelo físico de la base de datos, lo que siguió fue llenar las tablas con la información correspondiente de los artículos y bases de datos sobre modificaciones postraduccionales, sin duda, esta parte es del todo delicada, porque la información que se adquirió tuvo que ser revaluada una y otra vez. Se tiene que tener la completa seguridad que lo que se está poniendo en la base de datos es correcto.
Se puede aplicar un BLAST para saber si la secuencia de peptidos es correcta, o incluso con un programa sencillo recorrer toda la secuencia de aminoácidos de la proteína y llegar a la posición indicada para saber de qué residuo se trata y compararlo con lo dice el artículo

La etapa de parseo fue una de las que tomó más del 40% de programación de total del proyecto, hubo distintos sucesos que podrían resumir las actividades hechas de la siguiente manera:

- Primera etapa: Se recibieron 5 archivos originales de 5 modificaciones (acetilación, fosforilación, ubiquitinación, succinilación y calcio). Estos archivos previamente ya habían sido tratados con BioPython y tomado de los artículos originales.
- Segunda etapa: Se trabajó directamente con esos 5 archivos fuente para volverlos parsear por posibles errores. Se trabajó con un archivo fuente extraído de UNIPROT para conseguir las modificaciones pequeñas como metal, lipidación, glicosilación, sitio activo y disulfuro. Se extrajo de las modificaciones acetilación y fosforilación, la información para oxidación. Toda esta etapa fue trabajada con Python y adicionalmente se tuvo que manejar un intercambiador de nombres puesto que los nombres de las proteínas asignados por UNIPROT cambian a los universales.
- Tercera etapa: Se agregaron nuevas modificaciones que anteriormente no estaban consideradas (Nitrosilación, Nterminal) y metilación. Hubo correcciones de las modificaciones de fosforilación y ubiquitinación puesto que los archivos originales había muchas repeticiones. Se llenó la tabla de equivalencias con los datos dados por UNIPROT, rescatando el standard_name de cada proteína.
- Cuarta etapa: Se creó la tabla localización, aunque en los requerimientos se menciona, realmente no se pensaba antes agregar las localizaciones de cada proteína, pero finalmente se decidió anexar al proyecto dicha información. Se trabajó con un archivo fuente otorgado por los expertos en el área con diversas categorías de localizaciones dentro de la célula.

- Quinta etapa: A partir de la anterior etapa se inició al programación web y la construcción de las querys (en el siguiente subtema se hablará de eso), en esta etapa se inició el desarrollo para la integración de los PDBs, que consistió en una curación manual y en la programación de 8 parsers para seleccionar aquellos que realmente interesaban al proyecto, esta etapa fue crucial porque aparecieron nuevos problemas en algunas modificaciones que tuvieron que ser revisadas una vez más, debido a que los archivos originales tenían información errónea se tuvo que trabajar desde los artículos originales y volver a parsear, especialmente acetilación, fosforilación y ubiquitinación.
- Sexta etapa: Adicción de nuevos registros de disulfuro y finalmente se realizó la segunda curación manual de PDBs, con la intención de rescatar aquellas proteínas que tenían pdbs pero que no tienen modificaciones en la secuencia cristalizada.

Como resultado del parseo se obtuvieron como resultados: 43 parsers y 168 archivos de texto plano como producto de la compilación de los parsers. Para explicar este subtema, se dividirá en modificaciones y parsers, se pondrán aquellos parsers más representativos y los otros se anexarán en la parte de apéndice de esta tesis.

4.8.1 Tratamiento de la información de metal, lipidación, sitio activo, glicosilación, disulfuro y calcio.

Para estas modificaciones se extrajo la información directamente de UNIPROT, como ya se mencionó, es una base de datos reconocida que recopila información de diferentes publicaciones sobre proteínas y tiene documentación de diferentes modificaciones postraduccionales.

Archivo fuente: El documento original es un documento en Excel que contiene información de las proteínas de levadura y las modificaciones que están sufren, contiene un idprot de UNIPROT seguido por el alias, información sobre la fisiología de la proteína, algunos nombres estándares de la proteína, el organismo de procedencia, tamaño de la cadena de aminoácidos y finalmente una serie de columnas que representan diferentes modificaciones de interés metal, lipidación, sitio activo, glicosilación, calcio y disulfuro, de esta última modificación se hablará después. Se puede ver en la siguiente tabla los campos ya mencionados, que servirán como referencia para el parseo.

Idprot(UNIPROT)	Alias	Función	Nombre de genes	Organismo	Tamaño	Modificacion1	Modificacion2	…

Programa:

```
infile=open("C:\Users\Eduardo\Documents\\uniprot_metal_new.txt","r")

#infile2=open("C:\Users\Eduardo\Documents\\uniprot_ID.txt","r")

outfile=open("C:\Users\Eduardo\Documents\\uniprot_metal_brand.txt","w")

for line in infile:

   lista=line.split('\t')

   iones=lista[1].split(';')

#stan_name=lista[0]

 #if('METAL' in line):

#    outfile.write(lista[0]+'\t'+lista[12]+'\t'+lista[14])

   for item in iones:

      if('METAL' in item):

         str=item.replace("METAL","")

         str2=str.replace(" (By similarity)","")

         str3=str2.replace(".","")

         str4=str3.replace('  ',' ')

         lista2=str4.split('\t')

         print lista2
outfile.write('\t'+lista[0]+'\t'+lista2[0]+'\t'+lista[2])

infile.close()

outfile.close()
```

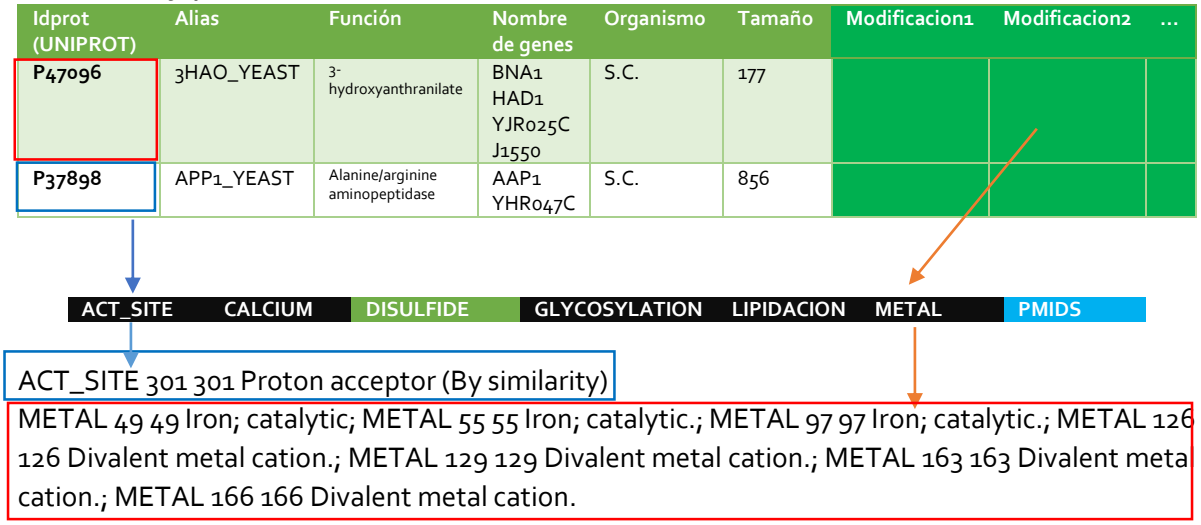
Funcionamiento: el programa tiene la capacidad de buscar dentro del archivo fuente registros asociados a modificaciones específicas, en la primera línea comentada, aparece el archivo fuente el cual se abre para analizar registro por registro, el archivo tiene 6622 registros y cada que encuentre una modificación deseada la escribirá en el archivo de salida (en este caso se trabajó con la modificación metal). Aparece un ciclo *for* que leerá registro por registro el archivo fuente (se cambió el archivo en Excel a extensión txt), para cada línea que se lea se crea un arreglo a través de la función *split*, que separa los elementos de la línea mediante un tabulador, en la línea comentada del *if* es donde se válida que modificación se está buscando.

Si la línea tiene METAL, ACT_SITE, etc. se sabrá a que modificación hace referencia, y una vez encontrada la modificación de interés, se escribe en el archivo de salida el idprot (el elemento 0 de la lista) seguido de un tabulador y aquellas columnas que corresponden a esa modificación, en este caso para metal fue la elemento 12 y 14, donde la columna 12 corresponde a las posiciones donde ocurren las modificaciones por iones de metal y la columna 14 las lista de pmids, que al final fue omitida.
Esta misma operación se realizó para las seis modificaciones, en caso de que haya más de dos posiciones mencionadas se hace una segunda lista de elementos (en este caso la lista se llama iones) ocupando un *split* con el punto y coma. Después se recorre el arreglo por elementos mediante un *for ítem*, de esta forma cada que se encontré el nombre de la modificación habrá una nueva posición excepto para disulfuro (se parseo a parte, después se mostrará el código para esta modificación), se borra todo aquello que no sea un número o un ion, con las líneas de *replace* y

finalmente se vuelve a separar los elementos por tabuladores y como en casi todas las modificaciones las posiciones se repiten, se toma el primer elemento del arreglo.

Una vez hecho esto se abre un segundo archivo marcado como infile2 que contiene las equivalencias para intercambiar nombres del idprot (línea comentada, cambia a idUniprot.txt).
Y se manda al archivo de salida toda esta información en el mismo orden que lo plantea el diseño físico de la base de datos. Para mayor claridad el siguiente esquema lo explica:

En lectura y procesamiento:



Se guarda en arreglo [line]:
[P37898, ACT_SITE 301 301 Proton acceptor (By similarity).]
Corrimiento del Arreglo mediante For ítem
Número de veces que aparece ACT_SITE,1 Entra una vez al ciclo
En el ciclo, se limpia el elemento del arreglo y se crea un segundo arreglo [line2]
[P37898, 301 301] Nueva variable str4='301 301' line2=[301 301]
Se considera el primer elemento del arreglo line2 y se busca equivalencia de idprot al identificador universal
P37898-----> YHR047C
Se imprime en archivo de salida YHR047C 301

Se guarda en arreglo [line]
[P47096, METAL 49 49 Iron; catalytic.; METAL 55 55 Iron; catalytic.; METAL 97 97 Iron; catalytic.; METAL 126 126 Divalent metal cation.; METAL 129 129 Divalent metal cation.; METAL 163 163 Divalent metal cation.; METAL 166 166 Divalent metal cation.]
Corrimiento del Arreglo mediante For ítem
Número de veces que aparece METAL, 6 Entra seis veces al ciclo
En el ciclo, se limpia el arreglo y se crea un segundo arreglo [line2]

En esta proteína con esa posición hay un sitio activo.

[P47096, 49 49 Iron 55 55 Iron catalytic 97 97 Iron; catalytic 126 126 Divalent metal cation 129 129 Divalent metal cation 163 163 Divalent 166 166 Divalent metal cation]
Seis Veces
Nueva variable str4=´49 49´
line2=[49 49]
Nueva variable str4=´55 55´
line2=[55 55]
Nueva variable str4=´97 97´
line2=[97 97]
Nueva variable str4=´126 126´
line2=[126 126]
Nueva variable str4=´129 129´
line2=[129 129]
Nueva variable str4=´163 163´
line2=[163 163]
Nueva variable str4=´166 166´
line2=[166 166]

Se considera el primer elemento del arreglo line2 el número de veces que aparece metal y se busca equivalencia de idprot al identificador universal

P47096-----> YJR025C

Se imprime en archivo de salida
YJR025C Iron 49
YJR025C Iron 55
YJR025C Iron 97
YJR025C Divalent metal cation 126
YJR025C Divalent metal cation 163
YJR025C Divalent metal cation 166

En esta proteína hay seis posiciones que sufren modificación por un ion.

Se puede observar dos rutas, la ruta azul que habla de proceso de parseo de una modificación cualquiera que no sea metal y la ruta roja-naranja que es la de metal. En tablas se muestra el proceso de compilación que hace paso por paso el programa, la explicación general del programa se habla al inicio del funcionamiento.

Cabe mencionar que el archivo fuente para hacer el traslado de idprot se obtuvo de un archivo proporcionado por la misma base de UNIPROT, aunque también se tuvo que parsear para otras modificaciones.

Hubo muchos pasos intermedios para obtener la modificación metal que no son explicados aquí por razones de síntesis, de manera general se fueron creando archivos intermedios hasta llegar a lo que se muestra en la tabla, pero la lógica es la misma.

Después de obtener estos primeros 6 archivos secundarios donde se tiene la proteína y la posición, se utilizó el siguiente programa para obtener el residuo modificado:

Programa:

```
infile=open("C:\Users\Eduardo\Documents\\calcium_bind.txt","r")

infile2=open("C:\Users\Eduardo\Documents\BaseDeDatoswBio\\proteinas_yeast.txt","r")

outfile=open("C:\Users\Eduardo\Documents\\calcium_last.txt","w")

pt={}

for line in infile2:

  listadicc=line.split('\t')

  pt[listadicc[1]]=listadicc[2]


for line in infile:

  #lista2=infile2.readline().split('\t')

  lista1=line.split('\t')

  protkey = lista1[0]

  pos=lista1[1]

  try:

    pept=pt[protkey]

  except KeyError as ee:

    print "no"

  prot=list(pept)

  posi=int(pos)

  try:

    res=prot[posi-1]

  except IndexError as e:

    print "no"


  print res

  outfile.write("\t"+lista1[0]+"\t"+res+"\t"+pos)

  #for item in lista:

  #outfile.write(peptidos)

outfile.close()

infile.close()

infile2.close()
```

Funcionamiento: El programa recibe como archivos de lectura, los 6 archivos intermedios generados en programa anterior y el proteoma, y se generará un archivo de salida que será el archivo donde ya no sólo se tenga la posición y la proteína modificada sino también el residuo modificado:

- Se lee el proteoma línea por línea para crear una lista asociada de idprot (llave) con su secuencia (valor).
- Se lee el archivo producto del programa anterior y se obtiene el idprot y la posición para después buscar dentro de la lista asociada esa secuencia en particular.
- Se coloca una sección de código try-except para mostrar un mensaje sí la proteína no existe, sí se obtiene la secuencia se obtiene el carácter en esa posición que será el residuo modificado, de igual forma si no lo encuentra se muestra un mensaje que el residuo no se encontró.

- Finalmente se escribe en el archivo de salida el idprot, el residuo y la posición, en caso de ser metal sólo se adiciona el ion.

Con esto, se tiene la información completa de esas modificaciones con los registros ordenados, lo que siguió fue agregar el pmid para cada uno de los archivos generados (el programa se puede ver en los apéndice: pmid_parser.py). Para estas modificaciones solo se le agrego el pmid 7 que corresponde a uniprot.org.

El parseo fue el mismo para las 6 modificaciones sin embargo se tuvo que volver a parsear la modificación disulfuro que se muestra a continuación.

Programa:

```
#!/usr/bin/perl

my $proteoma="proto.txt";
my $uniprot="uniprot_yeast.csv";
my $posis="posicionesdis.txt";
my $file= "uniprot_ID.txt";

my %disulf;
open(IN,$proteoma)|| die('File not found');

while(<IN>){
        chomp;
        my (undef,$idprot,$seqfather, undef)= split /\t/,$_;
        $universe{$idprot}=$seqfather;
}
close(IN);

open(IN,$file);
while(<IN>){
        chomp;
        push @linea,$_;
}
close(IN);

$cadena="DISUL la cadena tiene 353 (leo 32)";
$cadena=~ s/[a-zA-Z]//g;
#print $cadena;

open(EN,$uniprot);
while(<EN>){
        chomp;
        my ($uniprotid,undef,undef,undef,undef,undef,$sitio_activo,undef,$calcio,$disulfide,$glicosilacion,$lipidacion,undef)= split /\t/,$_;
        if ($disulfide ne ""){
        #print $uniprotid."\t".$disulfide."\n";
        $disulfide=~ s/[a-zA-Z]//g;
        $disulfide=~ s/[\.;]//g;
        $disulfide=~ s/\(..\)//g;
        $disulfide=~ s/\( \)//g;
        $disulfide=~ s/\(\)//g;
```

```
		$disulfide=~ s/-//g;
		#print $uniprotid."\t".$disulfide."\n";
		}
}
close(EN);

open(IN,$posis);
while(<IN>){
		chomp;
		my ($idprot,$posiciones)= split /\t/,$_;
		#print $idprot."\n";
		my @posiciones= split /\s+/,$posiciones;
		foreach my $ele (@posiciones){
			if ($ele ne ""){

		$disulf{$idprot.'*'.$ele}='1';
				#print $idprot."\t$ele\n";
			}
		}

}
close(IN);

foreach $keys (keys %disulf)
{		my ($id,$pos)=split /\*/,$keys;
		foreach $elemento(@linea){
			if ($elemento =~ /$id/){
				@valores=split /\s+/,$elemento;
foreach $valor(@valores){
if ((($valor=~ /\w{7}/) && ($valor=~ /^Y/) && ($valor=~ /\d{3}/)){

		$onres{$valor.'*'.$pos}=1;
#print $valor."\n";
}
}
}
		}
		#print "$keys con $disulf{$keys}\n";
}

foreach $key (keys %onres)
{
		#print "$key con $onres{$key}\n";
		my ($id,$pos)=split /\*/,$key;
		$key=~ s/;//g;
		if (exists($universe{$id})){
			my @aminoacidos= split //,$universe{$id};
			$pos=$pos+54,if ($id eq "YLR218C");
			$residuo=$aminoacidos[$pos-1];
			print "\t$id\t$residuo\t$pos\t7\n";
		}else{
			#print "no exito en proteoma $key\n";
		}
}
```

Funcionamiento: Este programa fue realizado en Perl, hace el funcionamiento de los dos programas anteriores. En general hace lo siguiente:

- Se crea un *hash* llamada universe que contempla todas las proteínas con sus secuencias, obtenido del archivo que contiene todo el proteoma (primer ciclo *while*).
- Se procede a leer todo el archivo UNIPROT y se guarda en un arreglo especial llamado línea.
- Se lee el archivo donde están las modificaciones y se obtiene solo aquellas asociadas a disulfuro (tercer ciclo *while*) y se elimina todo lo que no sea un número, todo eso se guarda en un nuevo archivo llamado posicionesdis.txt.
- Posicionesdis.txt es leído para guardar en un arreglo todas las posiciones que pueda tener una proteína, que después se ocuparán para crear una *hash* llamada disulf, cuya llave será el idprot y la posición que encuentre.
- Se recorre toda la tabla *hash* disulf y también todas los elementos del arreglo línea, donde se hayan todos las equivalencias, cuando uno de esos idprot sea igual a una de las llaves, se separa todas las equivalencias para esa proteína, y sólo se acepta aquellas idprot que cumplan con la nomenclatura universal.
- Se crea una *hash* auxiliar donde el nombre de la proteína ya no corresponde el idprot de UNIPROT sino al universal (a partir de ahora se le llamará nombre esquemático), la *hash* se llama onres.
- Finalmente se recorre todas las llaves de onres, y se separa cada una en el idprot y la posición, se valida que la proteína exista, de ser así se almacena todo la secuencia de aminoácidos y mediante la posicion se busca el residuo dentro del arreglo, como en Perl los índices de un arreglo empiezan en cero, le restamos uno a la posición. En esta parte se encontró un error de UNIPROT en la proteína YLR218C, arrojaba un residuo distinto a cisteína, que como se recordará la modificación disulfuro afecta sólo a la cisteína. Esto quiere decir que no siempre los artículos manejarán información fehaciente, habrá algunos registros que tendrán un residuo o posición errónea, en esta tesis se mostrarán algunos.

Al terminar de ordenar y limpiar la información mediante los programas explicados, se obtuvieron los 6 archivos con los que se llenaron las tablas correspondientes. Para hacer más ameno este subtema a partir de ahora no se colocarán los programas utilizados para las concernientes modificaciones (éstos se anexarán en en el apéndice para su consulta). A menos de que sea muy relevante su adición, para mayor facilidad se manejarán esquemas para explicar lo que se tuvo que programar para llegar a lo deseado.

4.8.2 Tratamiento de la información de acetilación.

Para esta modificación se utilizaron varios parsers que se pueden ver en el apéndice como: acetil_nuevos.pl y acetilCorrec.pl.

Archivo fuente: Se trabajó con dos artículos dedicados a acetilación: Henriksen 2012 y Weinrt 2014, adicionalmente información de otros artículo de cuyos datos están en un archivo llamado PTMfunc y también algunos cuantos registros del articulo Kang 2015. De estos dos últimos no se hablará como estaba la información original, porque ya estaba semiordenada sólo se tuvo que agregar algunos atributos. Como ya se mencionó se tuvo que corregir esta modificación debido a que algunos registros se repetían y algunas posiciones estaban erróneas en los archivos originales. Aunque si existen los primeros parsers para acetilación aquí sólo se hablará de los dos mencionados al inicio de este subtema.

En el artículo Henriksen 2012, la información está así:

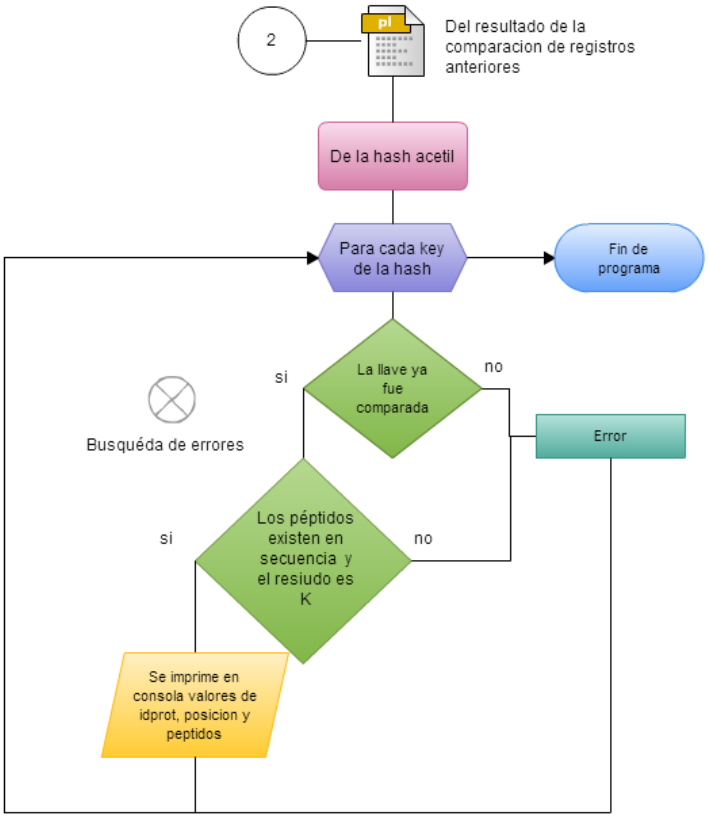
Idprot(s)	Posición(es)	Idprot(s)	Posición	Información	Probabilidad	Marcas/Eventos	Repeticiones	Péptidos	Otros Datos

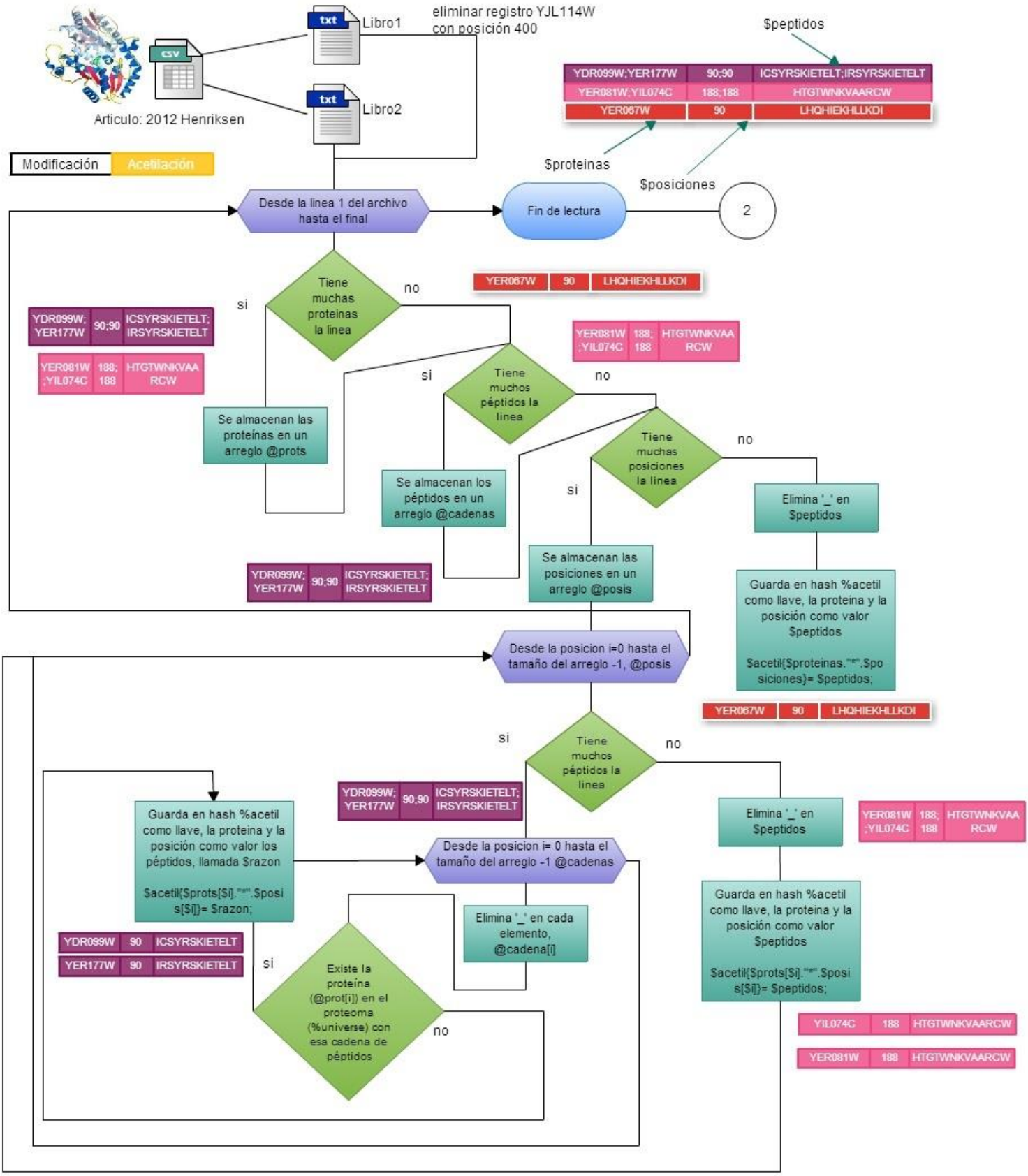
En el artículo Weinrt 2014 de esta forma:

Id	Posición(es)	Posición	Idprot(s)	Idprot	Información	Probabilidad	Péptidos	Otros Datos

Esquema y funcionamiento del programa

A continuación se muestra un diagrama que explica el funcionamiento del parser acetilCorrec.pl que trabaja con los dos artículos de arriba. Por razones de espacio se coloca primero la segunda parte del programa y después el inicio.




Articulo: 2012 Henriksen
csv
txt
Libro1
txt
Libro2
eliminar registro YJL114W con posición 400
$peptidos
YDR099W;YER177W 90;90 ICSYRSKIETELT;IRSYRSKIETELT
YER081W;YIL074C 188;188 HTGTWNKVAARCW
YER067W 90 LHQHIEKHLLKDI
$proteinas
$posiciones
Modificación
Acetilación
Desde la linea 1 del archivo hasta el final
Fin de lectura
2
Tiene muchas proteinas la linea
si
no
YDR099W; YER177W 90;90 ICSYRSKIETELT; IRSYRSKIETELT
YER081W ;YIL074C 188; 188 HTGTWNKVAA RCW
Se almacenan las proteinas en un arreglo @prots
Tiene muchos péptidos la linea
si
no
Se almacenan los péptidos en un arreglo @cadenas
Tiene muchas posiciones la linea
si
no
Se almacenan las posiciones en un arreglo @posis
Elimina '_' en $peptidos
Guarda en hash %acetil como llave, la proteina y la posición como valor $peptidos
$acetil{$proteinas."*".$posiciones}= $peptidos;
Desde la posicion i=0 hasta el tamaño del arreglo -1, @posis
Tiene muchos péptidos la linea
si
no
Guarda en hash %acetil como llave, la proteina y la posición como valor los péptidos, llamada $razon
$acetil{$prots[$i]."*".$posis[$i]}= $razon;
Desde la posicion i= 0 hasta el tamaño del arreglo -1 @cadenas
Elimina '_' en cada elemento, @cadena[i]
YDR099W 90 ICSYRSKIETELT
YER177W 90 IRSYRSKIETELT
Existe la proteína (@prot[i]) en el proteoma (%universe) con esa cadena de péptidos
si
no
Elimina '_' en $peptidos
Guarda en hash %acetil como llave, la proteina y la posición como valor $peptidos
$acetil{$prots[$i]."*".$posis[$i]}= $peptidos;
YIL074C 188 HTGTWNKVAARCW
YER081W 188 HTGTWNKVAARCW

En el esquema anterior sólo muestra el trabajo realizado sobre un solo artículo, el artículo Weinert 2014 tiene tres libros a diferencia de Henriksen 2012, pero el programa hace exactamente lo mismo. En particular existe una sección de código que se marca como un archivo en perl (pl) dentro del esquema, que no se trata a profundo, esta sección de código fue utilizada para la comparación de los registros que se tenían y ver cuál de ellos estaba erróneos, con la intención de sobreguardar loa primera curación manual de pdbs.

Como resultado de este parser se tuvo el archivo completo de modificaciones ocurridas por acetilación con la adición de los otros registros de los otros dos artículos (PTMfunc y Kang).

4.8.3 Tratamiento de la información de ubiquitinación y fosforilación.

Sin lugar a dudas estas dos modificaciones son las que más llevaron tiempo en parsear. Se programaron los siguientes parsers que se pueden encontrar en el apéndice:

- parser.py
- residuofosfo.py
- phospho.py
- ubi.pl
- parser3.py

Sin embargo los que finalmente se realizaron para corregir los errores y obtener la información de los artículos son: PhospoSwaney.pl, Iesmantavicius.pl, Swaney.pl y pruebaseq.pl.

Archivo fuente: Se trabajó con dos artículos del artículo Swaney 2013 y Iesmantavicius 2014, ambos archivos tienen información de ubiquitinación y fosforilación, aunque también del archivo PTMfunc se obtuvieron algunos registros. Fosforilación es la modificación estudiada, por ello más de la mitad de los artículos mencionan esta modificación, uno de las bases de datos que aporta información sobre esta modificación es PhosphoGRID. El parser para estas modificaciones está en programa detecction.pl y detsequencias.pl. No se hablará de estos parsers por razones de relevancia, por lo cual se enfatizará más sobre Swaney 2013 y específicamente más de Iesmantavicius 2014.

El parseo se hizo lo más sencillo posible; Iesmantavicius 2014 cuenta con dos libros, uno para ubiquitinación y otros para fosforilación los cuales contiene la información de la siguiente manera:

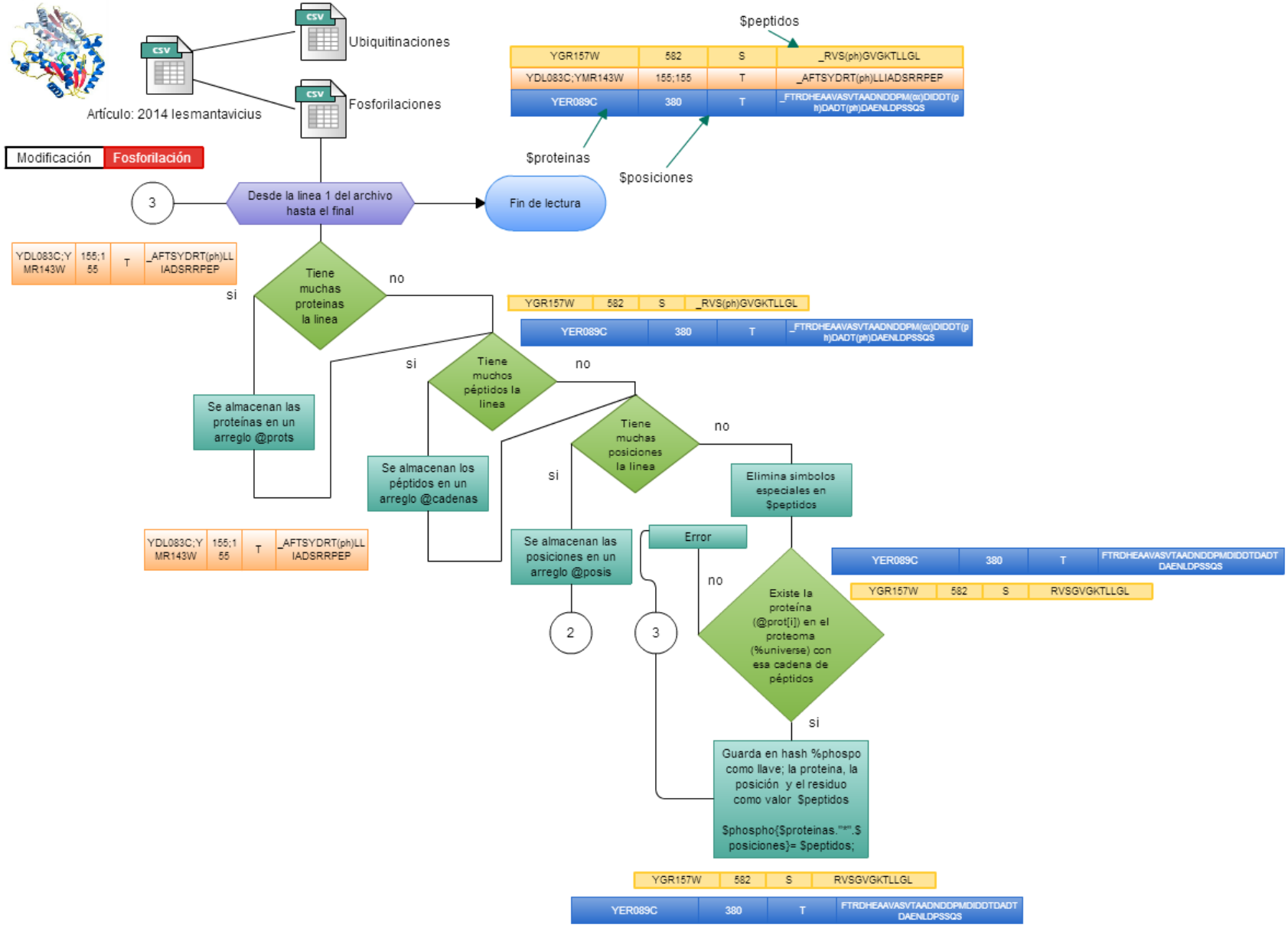
Para fosforilación:

Id	Idprot(s)	Posición(se)	Proteína Experimental	Posición	Información	Probabilidad	Residuo	Datos	Péptidos	Otros Datos

Para ubiquitinación:

Id	Idprot(s)	Posición(se)	Proteína Experimental	Posición	Información	Probabilidad	Datos	Péptidos	Otros Datos

Esquema y funcionamiento del programa:



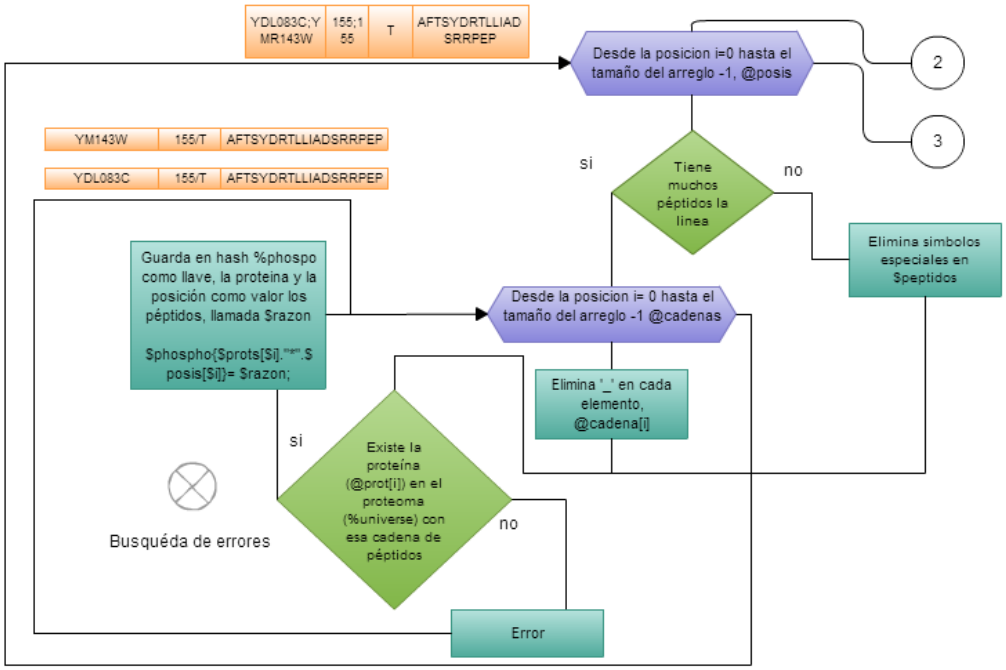


Fig 4.7 Esquema que explica el parseo de fosforilación y ubiquitinación

Para la ubiquitinación se hizo exactamente los mimo, solo que en vez de validar si los residuos eran Y, T, y S era solamente lisina (K). Los otros parsers utilizados en las etapas intermedias se pueden encontrar en el apéndice como:

- asocacion.pl
- mighty.pl
- parser_u.pl
- tosave.pl
- try.pl

Éstos se utilizaron para sobreguardar la información que anteriormente se tenía ubiquitinación, sin embargo, los parsers que realmente terminaron por formar la tabla actual son los que se explican en este apartado.

4.8.4 Tratamiento de la información de localización, equivalencias, proteína y otras modificaciones.

Para localización:

Se recibió un archivo original de localizaciones el cual tiene información del idprot y su respectiva ubicación(es) en la célula. Se tienen las siguientes categorías:

1. Golgi apparatus

2. cytoplasm
3. mitochondrion
4. peroxisome
5. endoplasmic reticulum
6. vacuole
7. vesicle
8. cell wall
9. prospore membrane
10. plasma membrane
11. membrane protein (se sabe que está insertada en alguna membrana)
12. putative membrane protein (se predice tener cruces transmembranales)
13. cellular bud (dará origen a célula hija)
14. site of polarized growth (protuberancia que afecta la circunferencia de la célula)
15. nucleus
16. nucleolus

De las 6,717 proteínas que existen en el proteoma de levadura, sólo de 5165 proteínas se sabe exactamente sus localizaciones dentro de la célula. Por eso se consideró de es un principio manejar el campo ubicaciones de localización como tipo de dato text y no se optó por separar ubicación por ubicación con su respectivo idprot e id de la ubicación.
El programa utilizado para acomodar dicha información, se encuentra en el apéndice como localización.pl.

Para equivalencias:

Como se mencionó anteriormente; pensando en las consultas que un usuario pueda hacer se necesita tener en la base de datos, los diferentes nombres de cómo se conocen a las proteínas, no sólo su nombre esquemático sino también sus alias y nombres estándares o convencionales.
Por ejemplo: la proteína YBL016W se conoce también como Fus3 o la proteína YDR103W se le conoce como Ste5 y también como Nul3.
UNIPROT otorga un archivo de equivalencias, el cual también fue paseado en algunos programas para obtener el nombres esquemático y sustituirlo por otros id de la proteína. Este archivo nos da la información de los nombres comunes para la mayoría de las proteínas.

Los programas ocupados fueron name.pl y last_parser.pl y generalmente lo que se hizo fue:

a) Extraer del archivo original los nombres comunes de la proteína.

b) Asociar el nombre esquemático con sus nombres comunes.
c) Imprimirlos en consola para generar la tabla deseada.

Para proteína:
Lo que se hizo para la formación de esta tabla fue trabajar con el archivo original que se tenía (proto.csv) y quitarle los caracteres especiales. Ver partproto.py.
Se tuvo que hacer de cierta manera una actualización de algunos registros pero esto se hizo muy fácilmente a través de la ayuda de YeastMine.

Otras modificaciones:

Para las modificaciones restantes se ocuparon los siguientes parsers:

- Nitrasilación: nitra.pl
- Metilación: metparser.pl
- Nterminal: equal.pl
- Oxidación: oxidacion.py (extraída de acetilación y fosforilación).
- Succinilación (parser igual a primeros de acetilación y fosforilación).

4.8.5 Tratamiento y obtención de los PDBs.

Una de las partes que tomo tiempo en obtener, es la información de los pdbs. Estos archivos cumplen con un formato regular pero no siempre siguen una misma convención respecto a posiciones o nombres de residuos, por lo que fue necesario hacer la depuración manual.

- Primeramente, con el programa pdb_yeast.pl se obtuvo la lista de pdbs asociado con su idprot.
- Después, un segundo programa (verificarPDB.pl) verifica que los PDBs con su idprot realmente existan en la base, y realiza un método de burbuja para saber qué pdbs caracterizan muchas moléculas y cuáles sólo una proteína, esto con la finalidad de quedarnos sólo con aquellos que tengan una única representación (de ser posible) y tener una lista auxiliar para depuración manual. En cuanto va formando la lista va descargando automáticamente los pdbs de la base de datos RSCB PDB (mediante el programa pdb_download.py).
- Una vez descargados los pdbs que son posibles de obtener. Se crea una nueva lista de pdbs descargados (se aplica un programa para convertir los nombres de los pdbs en mayúsculas con changename.pl), se hace una nueva lista de aquellos pdbs que no se pudieron descargar y se investiga la razón del porqué (se vuelve aplicar el programa verificarPDB.pl). Generalmente los pdbs que no se pudieron descargar se debe a que son muy pesados y por

consecuencias existen en otro tipo de formato para obtener la representación de la proteína.

- Una vez que se tiene toda lista de pdbs que se pudieron descargar, se necesita del proteoma de la levadura con sus alias para la construcción de una lista definitiva (programa tabla.pl). Se utiliza doble método de la burbuja para tener una lista de aquellos pdbs que caracterizan una molécula o menor número de moléculas (primer criterio) y que el peso del archivo sea menor a los demás pdbs que caracterizan la misma proteína (segundo criterio).
- De esta forma se obtuvieron dos listas: lista de proteínas que tienen más de un pdb y una lista de cada proteína con su único pdb que la representa o bien el que cumplió con las condiciones ya mencionadas. Se obtuvieron que de los 2838 pdbs descargados de levadura sólo 709 podrían ser considerados para ser agregados a la base de datos.
- Se hizo la primera curación manual con la lista de 709 pdbs, en esa curación se tuvo una estricta validación, considerado sólo aquellos pdbs que tenía la misma secuencia de aminoácidos en las posiciones correctas como indica el proteoma. Además se verificaba si la proteína tenían modificaciones cuyos residuos estén cristalizados. También se revisaba si el pmid era correcto o si se cambia a nuevo pdb, si éste tiene mayor número de residuos modificados (Fig 4.8).
- En esta curación manual se encontraron los errores en las modificaciones: acetilación, fosforilación y ubiquitinación, que estaban desde los archivos originales, por lo que se tuvieron que corregir y una vez hechos esos cambios, ver que pdbs de la curación manual podrían volver a ser aceptados.
- Se realizó la segunda curación manual, y se extrajo además información concerniente de aquellos pdbs que se quedaban en blanco (que no tenían ninguna modificación). De los 709 pdbs sólo se quedaron 655.
- Con la segunda curación manual se realizaron 3 listas: de borrado, de adición y de cambio de pdb, que se tomaron en cuenta para crear la lista definitiva que se agregó a la base de datos como la tabla de pdb. (tabla.pl)
- Como último paso se extrajo el pmid del pdb y se utilizó un programa de consulta automática (siguiente subtema) y se le agrego un nuevo source_id. Dando como resultado un archivo que tiene los idprot seguidos de los archivos pdbs con respectivo source_id. (add_fuentes.pl).

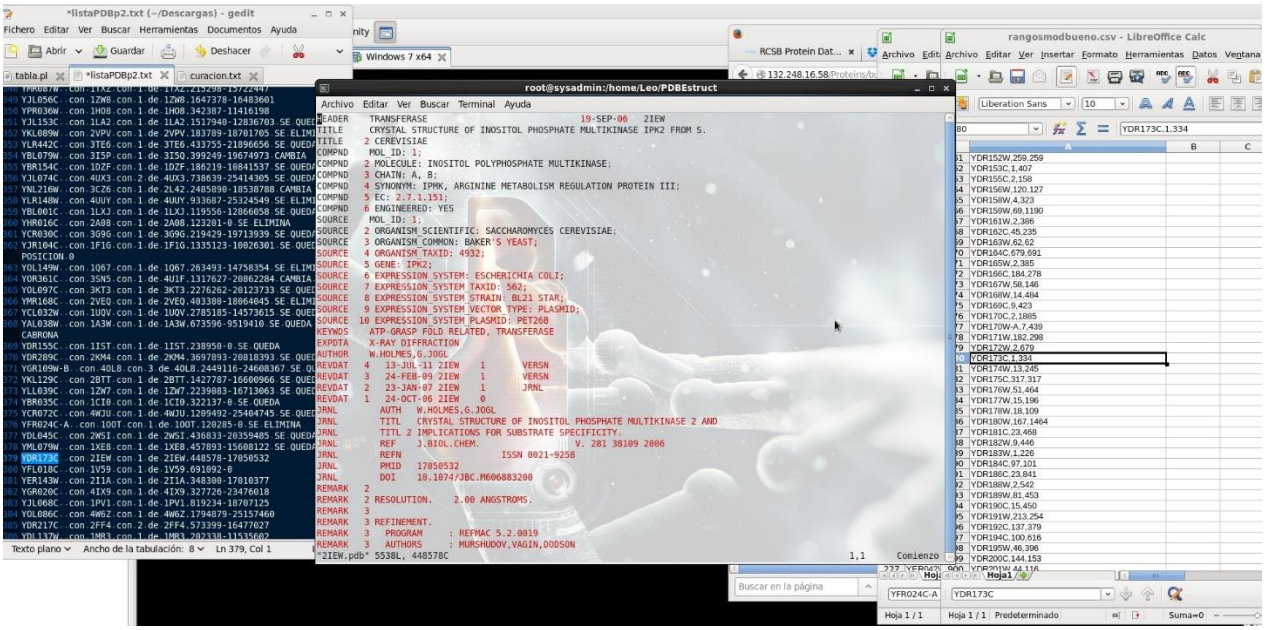


Fig 4.8 Curación Manual durante el tratamiento de información de PDBs

4.8.6 Tratamiento y manejo de fuentes.

Para el manejo de fuentes lo que se utilizó fue un programa de consulta automática llamado get_store_pmid.pl, este programa recibe como argumentos externos; el pmid y las operaciones sobre la base de consulta. Por ejemplo: . /get_store_pmid 21209884 3 0

La consola mostrará la referencia:

Wang, J; Taherbhoy, AM; Hunt, HW; Seyedin, SN; Miller, DW; Miller, DJ; Huang, DT; Schulman, BA (2010) Crystal structure of UBA2(ufd)-Ubc9: insights into E1-E2 interactions in Sumo pathways. PLoS ONE. 5 :(12) e15805.

Este programa se ejecutó en cada programa que finalizaba con la depuración de los archivos, y lo que se hacía era asociar a cada pmid, un id propio de la tabla source y de forma paralela se buscaba referencia. De esta forma se tiene el archivo de la modificación con sus pmids asociados (en source_id) y por otro lado otra tabla de los pmids con sus referencias.

id	idprot	Pe	residuo	posición	source_id
44421	YDR216W		S	98	41
44437	YDR216W		S	230	42
44439	YDR216W		S	230	43
44440	YDR216W		S	230	44
44442	YDR216W		S	230	45

id	pmid	fuente
41	18791642	Kacherovsky, N...
42	20139423	Ratnakumar, S; Young...
43	19398770	Ratnakumar, S; Kacherovsky,
44	1917932	Denis, CL; Kemp...
45	1549108	Denis, CL; Fontaine...

En caso de UNIPROT, la referencia es la 7, que señala a directamente a la página de internet.

Como resultado de este subtema fueron los archivos necesarios para el llenado de los registros en la base de datos, ahora el siguiente paso es mostrar las consultas que si hicieron en el desarrollo primario y secundario.

La instrucción para llenar cada una de las tablas en la base de datos a partir de los archivos parseados fue la siguiente:

```
LOAD DATA INFILE '/var/lib/mysql/proteinas/citas_phospo.txt'
INTO TABLE source
FIELDS TERMINATED BY '\t'
LINES TERMINATED BY '\n';
```

4.9 Construcción de las consultas mediante lenguaje SQL.

Se realizaron aquellas consultas (querys) que fueron consideradas por los especialistas en el área muy importantes y que pudieran proporcionar información relevante. En la primera etapa de desarrollo lo que se hizo fue programar todas las querys y en la segunda etapa de desarrollo se juntó en una sólo script de programación.

En este apartado se explicará, como se realizaron las consultas y que objetivo cumplirán y finalmente se mostrará los programas que englobaron todo. La razón por la cual se hizo así es porque se quería una versión predifinitva de consultas para después asociarla a un diseño web.
Como ya se mencionó el primer desarrollo se hizo utilizando WAMP, para después emigrar todo a un servidor Linux ahora utilizando LAMP.

4.9.1 Álgebra Relacional.

Se utilizarán los símbolos estandarizados para el cálculo relacional:

Símbolo	Operación
σ	Selección
$\bowtie$	Natural Join
$\bowtie_{\theta}$	Theta Join
π	Proyección
ρ_{x}	Renombramiento
$\times$	Producto Cartesiano
$\cup$	Unión
$R = \lvert x \rvert S$	Unión externa izquierda, Extensión del Join
$R \lvert x \rvert = S$	Unión externa derecha, Extensión del Join

Aunque existen más símbolos en cálculo relacional se utilizaran sólo mostrados aquí:

1. Primera consulta: que se muestre de una tabla de cualquier modificación o de cualquier otra entidad; toda su información contenida en la base de datos. Esta consulta es la más sencilla de todas, consiste en hacer un simple *join* con el nombre de la tabla (modificación) que se desee y con la tabla source (sr) o simplemente una proyección (cualquier otra entidad).

   a) Para las modificaciones (m) que tengan péptidos como atributo, el cálculo relacional sería:

$$\prod_{m.id,\ m.proteina\_idprot,\ m.peptidos,\ m.posicion,\ sr.id} \left(m \bowtie_{m.source\_id = sr.id} sr\right)$$
$$= \prod_{m.id,\ m.proteina\_idprot,\ m.peptidos,\ m.posicion,\ sr.id} \left(\sigma^{(m \times sr)}_{m.source\_id = sr.id}\right)$$

   b) Para modificaciones que no tengan peptidos como atributo:

$$\prod_{m.id,\ m.proteina\_idprot,\ m.posicion,\ sr.id} \left(m \bowtie_{m.source\_id = sr.id} sr\right)$$
$$= \prod_{m.id,\ m.proteina\_idprot,\ m.posicion,\ sr.id} \left(\sigma^{(m \times sr)}_{m.source\_id = sr.id}\right)$$

   c) Para otras tablas (t), en general:

$$\prod_{t.x,t.y,t.z,\ldots,t.n} (t)$$

   Para la modificación metal solo se agregó el atributo ion en la proyección.

2. Segunda consulta: búsqueda general, que trabaje con una cadena de texto, ya sea un alias o nombre estándar de la proteína o bien el identificador universal de la proteína, que muestre la información de sus ubicaciones, su secuencia, nombre estándar y su header.

$$\prod_{idprot,ubicaciones,sequence,standard\_name,header} \left(Protein \bowtie_{idprot = proteina\_idprot} Location\right)$$

$$= \prod_{idprot,ubicaciones,sequence,standard\_name,header} \left(\sigma_{idprot = proteina\_idprot}^{(Protein \times Location\ )}\right)$$

Cabe mencionar que en este caso no se indican los atributos con su tabla de procedencia puesto que los atributos son únicos en cada entidad, pero también se pueden escribir como la primera consulta (es decir se puede escribir Protein.idprot en vez de sólo idprot, así para cada atributo).

3. Tercera consulta: Obtener todas las modificaciones de todas las tablas de una proteína en especial, rescatando el nombre de la tabla de procedencia, además el residuo modificado, la posición y el pmid con sus referencias.

$$\prod_{residuo,posicion,pmid,fuente,TABLE\_NAME} \left(\sigma_{Acetylation.protiena\_idprot='\$proteina' \vee e.alias='\$proteina'}^{E1}\right)$$
U
$$\prod_{residuo,posicion,pmid,fuente,TABLE\_NAME} \left(\sigma_{Phosphorilaytion.protiena\_idprot='\$proteina' \vee e.alias='\$proteina'}^{E2}\right)$$
U
$$\prod_{residuo,posicion,pmid,fuente,TABLE\_NAME} \left(\sigma_{Glycosylation.protiena\_idprot='\$proteina' \vee e.alias='\$proteina'}^{E3}\right)$$
U
.....
$$\prod_{residuo,posicion,pmid,fuente,TABLE\_NAME} \left(\sigma_{Disulfide.protiena\_idprot='\$proteina' \vee e.alias='\$proteina'}^{E14}\right)$$

$$E1 = \left(\prod_{residuo,posicion,pmid,fuente,TABLE\_NAME} \rho_{tn}^{\left(\sigma_{TABLE\_NAME='Acetylation' \wedge TABLE\_SCHEMA='proteinas'}^{(information\_schemas\ )}\right)}\right) \times \left(\sigma_{Acetylation.source\_id=Source.id}^{(Acetylation \times Source\ )}\right) \times \left(\sigma_{Acetylation.proteina\_idprot=Equivalences.proteina\_idprot}^{(Acetylation \times Equivalences\ )}\right)$$

$$E2 = \left(\prod_{residuo,posicion,pmid,fuente,TABLE\_NAME} \rho_{tn}^{\left(\sigma_{TABLE\_NAME='Phosphorylation' \wedge TABLE\_SCHEMA='proteinas'}^{(information\_schemas\ )}\right)}\right) \times \left(\sigma_{Phosphorylation.source\_id=Source.id}^{(Phosphorylation \times Source\ )}\right) \times \left(\sigma_{Phosphorylation.proteina\_idprot=Equivalences.proteina\_idprot}^{(Phosphorylation \times Equivalences\ )}\right)$$

$$E3 = \left(\prod_{residuo,posicion,pmid,fuente,TABLE\_NAME} \rho_{tn}^{\left(\sigma_{TABLE\_NAME='Glycosylation' \wedge TABLE\_SCHEMA='proteinas'}^{(information\_schemas\ )}\right)}\right) \times \left(\sigma_{Glycosylation.source\_id=Source.id}^{(Glycosylation \times Source\ )}\right) \times \left(\sigma_{Glycosylation.proteina\_idprot=Equivalences.proteina\_idprot}^{(Glycosylation \times Equivalences\ )}\right)$$

.....

$$E14 = \left(\prod_{residuo,posicion,pmid,fuente,TABLE\_NAME} \rho_{tn}^{\left(\sigma_{TABLE\_NAME='Disulfide' \wedge TABLE\_SCHEMA='proteinas'}^{(information\_schemas)}\right)}\right) \times \left(\sigma_{Disulfide.source\_id=Source.id}^{(Disulfide \times Source)}\right) \times \left(\sigma_{Disulfide.proteina\_idprot=Equivalences.proteina\_idprot}^{(Disulfide \times Equivalences)}\right)$$

Esta consulta es un poco larga debido a que se quiere saber el nombre de la modificación, y se necesita recurrir a los atributos de las tablas. Se ponen las primeras 3 modificaciones y luego se salta hasta la 14, esto con la intención de no poner todo, porque prácticamente es lo mismo. Se deja expresada la proteína que se desea buscar como una variable ($proteína).

4. Cuarta consulta: Se sabe que el residuo lisina es común para cinco modificaciones: fosforilación, ubiquitinación, acetilación, succinilación y metilación. Se necesita saber aquellas proteínas que sufran una, dos o tres modificaciones (de las cinco) en la misma posición. Mostrar el residuo modificado, la posición y el source_id.

   Sea las modificaciones m1, m2 y m3 de las cinco ya mencionadas y la tabla source como sr, el cálculo relacional sería:

$$E: \prod_{m1.proteina\_idprot,\ m1.residuo,\ m1.pos,\ sr.id} \Big(\big(m1 \bowtie_{m1.source\_id=sr.id} sr \big) \times \big(m1 \bowtie_{m1.proteina\_idprot=m2.proteina\_idprot \wedge m1.pos=m2.pos} m2 \big) \times \big(m1 \bowtie_{m1.proteina\_idprot=m3.proteina\_idprot \wedge m1.pos=m3.pos} m3 \big) \Big)$$

$$\cup \prod_{m2.proteina\_idprot,\ m2.residuo,\ m2.posicion,\ sr.id} \Big(\big(m2 \bowtie_{m2.source\_id=sr.id} sr \big) \times \big(m2 \bowtie_{m2.proteina\_idprot=m1.proteina\_idprot \wedge m2.pos=m1.pos} m1 \big) \times \big(m2 \bowtie_{m2.proteina\_idprot=m3.proteina\_idprot \wedge m2.pos=m3.pos} m3 \big) \Big)$$

$$\cup \prod_{m3.proteina\_idprot,\ m3.residuo,\ m3.posicion,\ sr.id} \Big(\big(m3 \bowtie_{m3.source\_id=sr.id} sr \big) \times \big(m3 \bowtie_{m3.proteina\_idprot=m2.proteina\_idprot \wedge m3.pos=m2.pos} m2 \big) \times \big(m3 \bowtie_{m3.proteina\_idprot=m1.proteina\_idprot \wedge m3.pos=m1.pos} m1 \big) \Big)$$

   La consulta que se plantea aquí es considerando tres modificaciones, sin embargo si se quiere hacer con sólo dos modificaciones solo basta con

eliminar la proyección de una unión y eliminar el *join* correspondiente en las restantes.

5. Quinta consulta: Expresar la consulta anterior, pero ahora en términos de cualquier modificación, validando ahora que el residuo sea el mismo en la misma posición:

   Sería exactamente la misma consulta de arriba; cuando se pone como condición que sea la misma posición dentro de la misma proteína obliga a que sea el mismo residuo. Aunque sea la misma consulta, en la programación web si abra diferencias significantes.

6. Sexta consulta: Buscar todas aquellas proteínas que están ubicadas en una localización específica de la célula y además que sufran una o dos modificaciones en particular. Ejemplo: Buscar las proteínas que están en la membrana y además se fosforilen y se ubiquitinen.

$$\prod_{proteina.id\_prot}\left(m1 \bowtie_{m1.proteina\_idprot=m2.proteina\_idprot} m2 \bowtie_{m1.proteina\_idprot=E1.proteina\_idprot} E1\right)$$

$$E1 = \prod_{proteina.id\_prot,ubicaciones} \sigma^{Location}_{ubicaciones='\$localizacion'}$$

Las variables involucradas aquí son; $localizacion, y el nombre de las modificaciones (m1, m2).

Como se puede observar las consultas van muy inclinadas en la forma en que se utilizaron en la programación web.
Una vez teniendo las seis consultas lo que se hizo fue juntarlas todas adicionando la parte de los pdbs, todo en la segunda parte del desarrollo.

Ver en el apéndice las consultas ya programas en lenguaje SQL en los archivos: search.php y localization.php.

4.10 Elaboración y diseño de la página web. Manejo de PHP, JavaScript, JSON y Ajax.

Lo que se hizo fue realizar un diseño amigable y de fácil uso para los usuarios finales, se optó por manejar:

- Programas intermedios (del primer desarrollo) que realizan las consultas y arrojan los datos en formato JSON a través de jQuery. Antes esta operación se realizaba con Google Charts.
- Los programas finales reciben el objeto JSON que es manipulado en DataTable para ser presentado en la página, todo a través de Ajax. Ver yaam_ini.php y detalle_final.php
- Se crea un apartado especial de comentarios, para agregar nuevas modificaciones sugeridas por los usuarios que así lo deseen. En seguida se manda un correo al administrador de YAAM para que valide que la información sea verídica. Ver add_yaam.php
- Se utilizan los archivos pdbs para ser presentados en detalle_final.php, si la proteína que se está buscando tiene pdb en la base de datos se muestra con los aminoácidos que son modificados con el color correspondiente al mapa de colores que se verá más adelante.
- Para visualizar el pdb se utilizó la librería de desarrollo ChemDoodle Web Componentes y se manda llamar en JavaScript dando como atributo el archivo pdb de la proteína. El programa original de ChemDoodle fue modificado para colorear los aminoácidos con las modificaciones específicas, este programa no se muestra en el apéndice porque tiene una extensión considerable, pero el archivo original se puede descargar en: http://web.chemdoodle.com/installation/download. (Fig. 4.10)

El mapa de colores utilizado fue el siguiente:

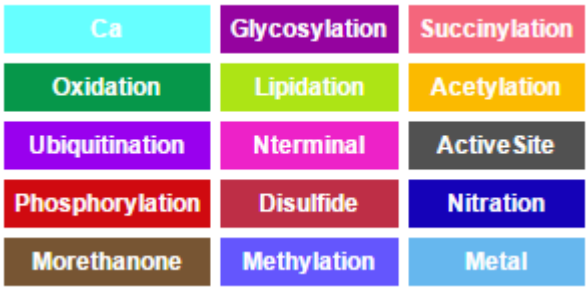
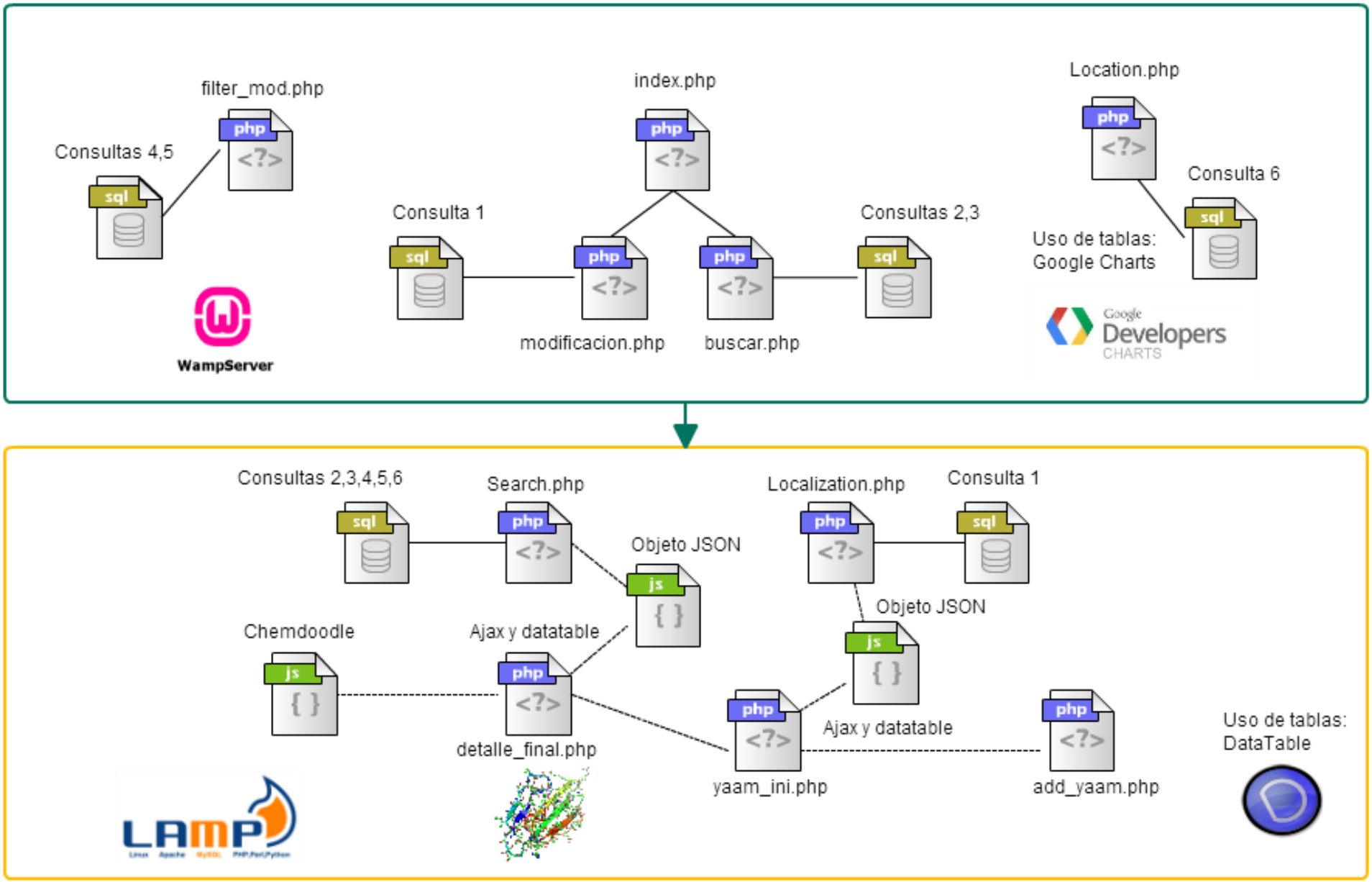


Fig 4.9 Mapa de colores para modificaciones YAAM

Fig 4.10 Desarrollo web del proyecto: primera etapa (recuadro verde) y segunda etapa (recuadro amarillo)



Los archivos de programación del primer desarrollo no se encuentran en esta tesis, puesto que ya están involucrados en los programas del segundo desarrollo

A continuación se muestran el funcionamiento de las consultas en el primer desarrollo:

Etapa de pruebas de consultas

Primera Consulta y página en index.php

Una tabla sencilla, donde se tiene un menú despegable en la parte donde dice modificación y un botón que toma la función de realizar la primera consulta.

Modificacion Acetylation ▼ Go

Buscar:

ID	Schematic Name	Peptides	Residue	Position	Source
1	R0040C	NARGAYKLQNTIT	K	262	22865919
2	YAL003W	ASTDFSKIETLKQ	K	8	22865919
3	YAL003W	SKIETLKQLNASL	K	13	22865919
4	YAL005C	EVQADMKHFPFKL	K	86	22865919
5	YAL012W	AALSANKIAEFLA	K	267	22865919
6	YAL012W	NYDVVLKQHRDAL	K	298	22865919
7	YAL012W	KVTDIQKVADLIK	K	160	22865919
8	YAL012W	GAEAASKFASSTR	K	322	22865919
9	YAL012W	VNYPGLKTHPNYD	K	288	22865919
10	YAL013W	RANPVDKLEVIVD	K	355	22865919
11	YAL013W	ANLPELKYAPKLS	K	69	22865919
12	YAL013W	DLEGLRKYFHSFP	K	377	22865919
13	YAL013W	KLSSLVKQETLTE	K	79	22865919
14	YAL023C	SSTGYSKNNAAHI	K	11	22865919
15	YAL023C	NNAAHIKQENTLR	K	18	22865919
16	YAL025C	KALVAAKIEKAIE	K	157	22865919
17	YAL035W	SDWLLLKKLKVVF	K	993	22865919
18	YAL035W	RSIDTLKDKAFRD	K	976	22865919

Segunda y tercera consulta y página en buscar.php

Se guarda un valor obtenido de la caja de texto de Buscar y realiza la segunda y tercera consulta. Se puede observar que se programó el algoritmo para colorear la secuencia con el color específico de cada modificación.

Modificacion Acetylation ▼ Go

Buscar: hog1

Schematic Name	Localization	Sequence	Sequence Order	Standard Name	Header
YLR113W	cytoplasm,nucleus	MTTNEEFIRTQIFGTVFEITNRYNDLNPVGMGAFGLVCSATDTLTSQPVAIKKIMKPFST AVLAKRTYRELKLLKHLRHENLICLQDIFLSPLEDIYFVTELQGTDLHRLLQTRPLEKQF VQYFLYQILRGLKYVHSAGVIHRDLKPSNILINENCDLKICDFGLARIQDPQMTGYVSTR YYRAPEIMLTWQKYDVEVDIWSAGCIFAEMIEGKPLFPGKDHVHQFSIITDLLGSPPKDV INTICSENTLKFVTSLPHRDPIPFSERFKTVEPDAVDLLEKMLVFDPKKRITAADALAHP YSAPYHDPTDEPVADAKFDWHFNDADLPVDTWRVMMYSEILDFHKIGGSDGQIDISATFD DQVAAATAAAAQAQAQAQAQVQLNMAAHSHNGAGTTGNDHSDIAGGNKVSDHVAANDTIT DYGNQAIQYANEFQQ		HOG1	Mitogen-acti kinase involv osmoregulati global realloc in osmotic sh CDC28 by st antisense RN mediates recruitment/a RNAPII at H promoters; w novel S-phas prevent confl DNA replica transcription; represses pse growth; autop protein abun under DNA r

	Residue	Position	PMID	Modification
1	T	2	22814378	Nterminal
2	D	144	uniprot.org	ActiveSite
3	K	146	23749301	Ubiquitination
4	T	174	18329363,11875433,11113180,19684113,20489023,21177495,22817900,23749301	Phosphorylation
5	Y	176	18329363,11875433,11113180,19779198,19684113,18407956,21126336,21177495,22817900,23749301	Phosphorylation
6	S	178	18407956	Phosphorylation
7	T	179	19684113,18407956,22817900	Phosphorylation
8	K	288	23954790	Succinylation

Cuarta y quinta consulta y página en filter.php

Se programaron selecciones dependientes para ir restando los nombres de las tablas cada que se seleccione una. Al final mostrará aquellas proteínas que sufran tales modificaciones en la misma posición. En la primera imagen se tiene las coincidencias de las cinco modificaciones cuyo residuo es lisina y en la segunda imagen aquellas proteínas que sufran cualquiera modificación(es) sin importar el residuo que sea.

Elige | Selecciona Opción... | Selecciona Opción...

ALL Go

	Schematic Name in Acetylation, Methylation and Succinylation	Residue	Position	Source
1	YGR192C	K	137	22865919,23954790,24489116,25109467
2	YGR254W	K	264	22865919,23954790,24489116,25109467
3	YIL069C	K	37	22865919,23954790,24489116,25109467
4	YKR057W	K	5	22865919,23954790,24489116,25109467
5	YLR441C	K	174	22865919,23954790,24489116,25109467

Elige | Selecciona Opción... | Selecciona Opción...

ALL Go

	Schematic Name in Disulfide and Metal	Residue	Position
1	YCL033C	C,C,C,C,C,C	79,82,97,128,131,157
2	YEL024W	C,H,C,C,C,H	159,161,164,178,180,181
3	YHR132C	H,E,C,C,H	182,185,251,272,310
4	YJR104C	E,H,H,C,H,H,H,D,H,C	43,47,49,58,64,72,81,84,121,147
5	YLL009C	C,C,C,C,C,C	23,24,26,36,47,57
6	YLR080W	Y,C,C	177,196,230
7	YMR038C	H,C,C,C,C,C,C	16,17,20,27,64,229,231
8	YNL238W	D,D,N,C,D,D,C,E,C,C	135,184,227,230,277,320,322,350,352,377

Sexta consulta en Location.php

Se selecciona una ubicación que se desee junto con una o más modificaciones; arrojará como resultado una tabla, de aquellas proteinas que satisfagan la consulta, en este caso se buscó en la modificación de sitio activo y acetilación en el citoplasma.

cytoplasm

Selecciona Opción...

Elige

Go

	Schematic Name (ActiveSite and Acetylation in cytoplasm)
1	YHR215W
2	YGR240C
3	YLR270W
4	YDR232W
5	YNL292W
6	YML075C
7	YNR016C
8	YER151C
9	YDL125C
10	YIL053W

El formato de tabla que se utilizó aquí no es común de HTML, sino de Google Charts. Este tipo de herramienta recibe como dato, uno de tipo JSON es reconocido por la API y finalmente se le adiciona un formato que mejor convenga.

RESULTADOS Y CONCLUSIONES

CAPÍTULO V

5.1 Base de datos de modificaciones postraduccionales. Posibles mejoras.

Una vez concluido el diseño de la base de datos y su llenado a través de los archivos que fueron parseados, se tuvieron los siguientes resultados:

MODIFICACIONES	
Modificación	**Registros**
Ubiquitinación:	10878
Acetilación	9956
Lipidación	173
Fosforilación	86791
Calcio	26
Por iones de metal	1923
Oxidación	794
Succinilación	1752
Puentes de disulfuro	262
Glicosilación	1967
Sitio activo	1100
Amino Nterminal	758
Metilación	117
Nitrasilación	14
Total Modificaciones	**116,511**

TABLAS AUXILIARES	
Tabla	**Registros**
PROTEINA	6717
Equivalencias	9980
Fuentes PMID	1089
pdb	655
Localización	5165
Total tablas auxiliares	23,606

REGISTROS TOTALES YAAM	**140,117**
TABLAS	19

Tabla 5.1 Cantidad de registros de las modificaciones en la base de datos, también se muestra la cantidad de registros de otras tablas.

Se puede observar que hay 116,511 registros de modificaciones en las 6717 proteinas que tiene la levadura. Como se mencionó anteriormente, solamente se almacenan aquellos datos de modificaciones que están estrictamente documentadas (existe un artículo que la respalda). Aunque pueden ver más de 400 modificaciones aquí sólo se trabaja con 14, especialmente en la levadura.

Las modificaciones que tienen más registros en la base son fosforilación, ubiquitinación y acetilación, como se puede ver en la siguiente gráfica, las tres juntas representan el 92% de los registros en YAAM (Fig 5.1).

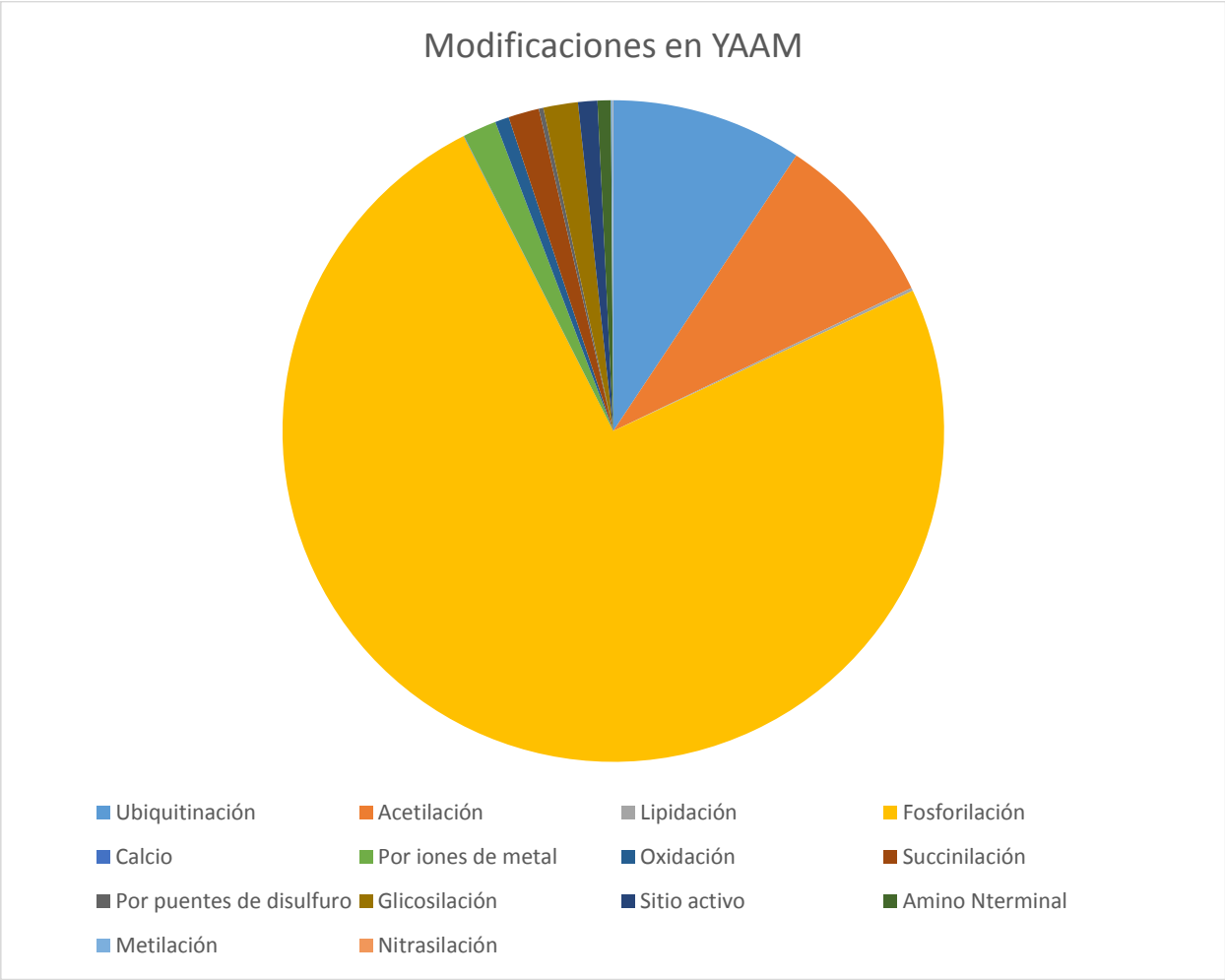


Fig. 5.1 Gráfica de porcentajes de la cantidad de registros de modificaciones postraduccionales en YAAM.

Dentro de las posibles mejoras en la base de datos, podría ser que se realice un programa que consulta constantemente las bases de datos de proteínas de levadura y que esté al pendiente de las publicaciones de modificaciones postraduccionales, de tal manera que extraiga la información, la procese y la envié a tabla correspondiente en la base de datos. Esta actualización podría programarse a través de un cron y que se ejecute un determinado tiempo.

También se dejó preparado un atributo en la tabla proteína sobre regiones desordenadas y ordenadas, que más adelante podrían surgir nuevas consultas que den información relevante a los biólogos.

Durante el proceso de diseño se hicieron algunas observaciones sobre algunas cardinalidades de relaciones, posiblemente si se hubieran hecho las catorce tablas auxiliares de aquellas relaciones, las consultas hubieran sido más fáciles todavía. Esto implicaría más trabajo y una nueva curación de datos, de tal manera que los archivos finales se parezcan al nuevo modelo físico. Se considera que el diseño de la bases de datos fue el adecuado, ya que la consultas son rápidas e incluso cuando la proteína tiene PDB el tiempo de espera es mínimo.

5.2 Página web final de modificaciones postraduccionales.

La página inicial (ver apéndice a yaam_ini.php):

Fig. 5.2 Página de inicio del sistema de consulta

Se tiene dos búsquedas principales la primera en general y la segunda es por modificación o localización:

- En la búsqueda general lo que hace es buscar una proteína mediante su nombre esquemático o un alias.
- YAAM Search busca aquellas proteínas que tengan una o varias modificaciones (hasta tres) y con una determinada localización (hasta tres), adicionalmente se agrega una apartado de residuo, qué lo que hace es aumentar la consulta agregando una condición; buscando además aquellas proteinas cuyas modificaciones se produzcan en el mismo residuo. (Ver las siguientes figuras)

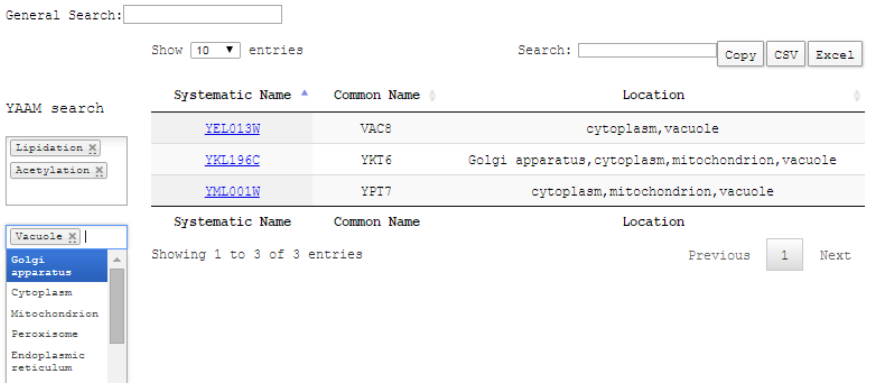


Fig. 5.3 Ejemplo de consulta, busca aquellas proteínas que se lipiden y se acetilen y este ubicadas en la vacuola.

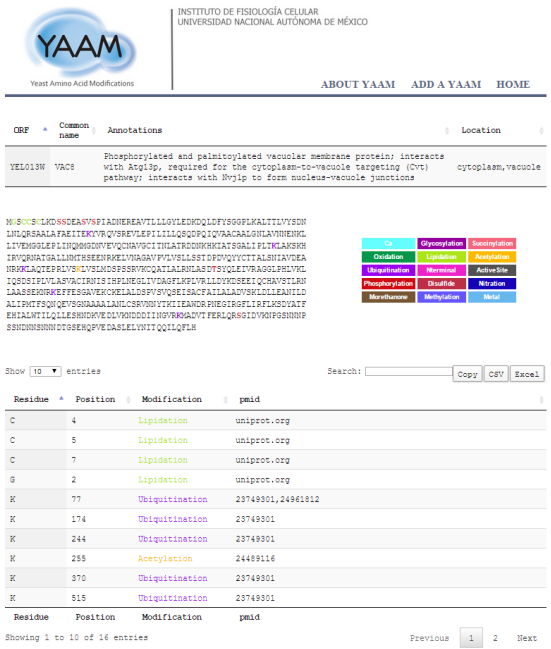


Fig. 5.4 Búsqueda por el nombre esquemático: en este caso se buscó la proteína YEL013W, la consulta muestra la información de la proteína y su secuencia coloreada por aquellos residuos modificados, en la parte inferior se muestran todas sus modificaciones.

Cuando la proteína tiene PDB se muestra en la página (ver detalle_final.php)

YLR362W	STE11	Signal transducing MEK kinase; involved in pheromone response and pseudohyphal/invasive growth pathways where it phosphorylates Ste7p, and the high osmolarity response pathway, via phosphorylation of Pbs2p; regulated by Ste20p and Ste50p; protein abundance increases in response to DNA replication stress	cytoplasm

Fig. 5.5 Muestra del PDB de la proteína YLR362W, esta proteína no tiene aminoácidos cristalizados que se modifican por lo tanto solo se ponen en blanco.

YCL035C	GRX1	Glutathione-dependent disulfide oxidoreductase; hydroperoxide and superoxide-radical responsive, heat-stable, with active site cysteine pair; protects cells from oxidative damage; GRX1 has a paralog, GRX2, that arose from the whole genome duplication; protein abundance increases in response to DNA replication stress	cytoplasm,nucleus

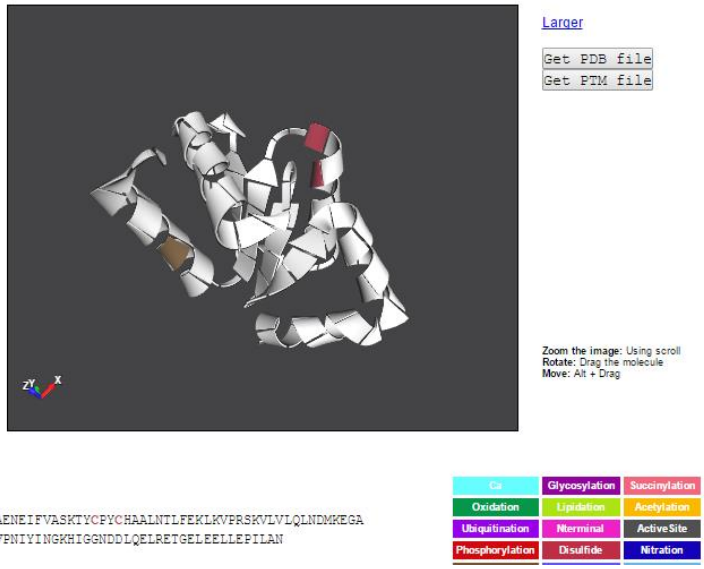


Fig. 5.6 La proteína YCL035C a diferencia de la imagen anterior si muestra residuos modificados.

Finalmente cuando alguien quiere hacer alguna observación o agregar una nueva modificación se tiene la siguiente pagina. (ver add_yaam.php)

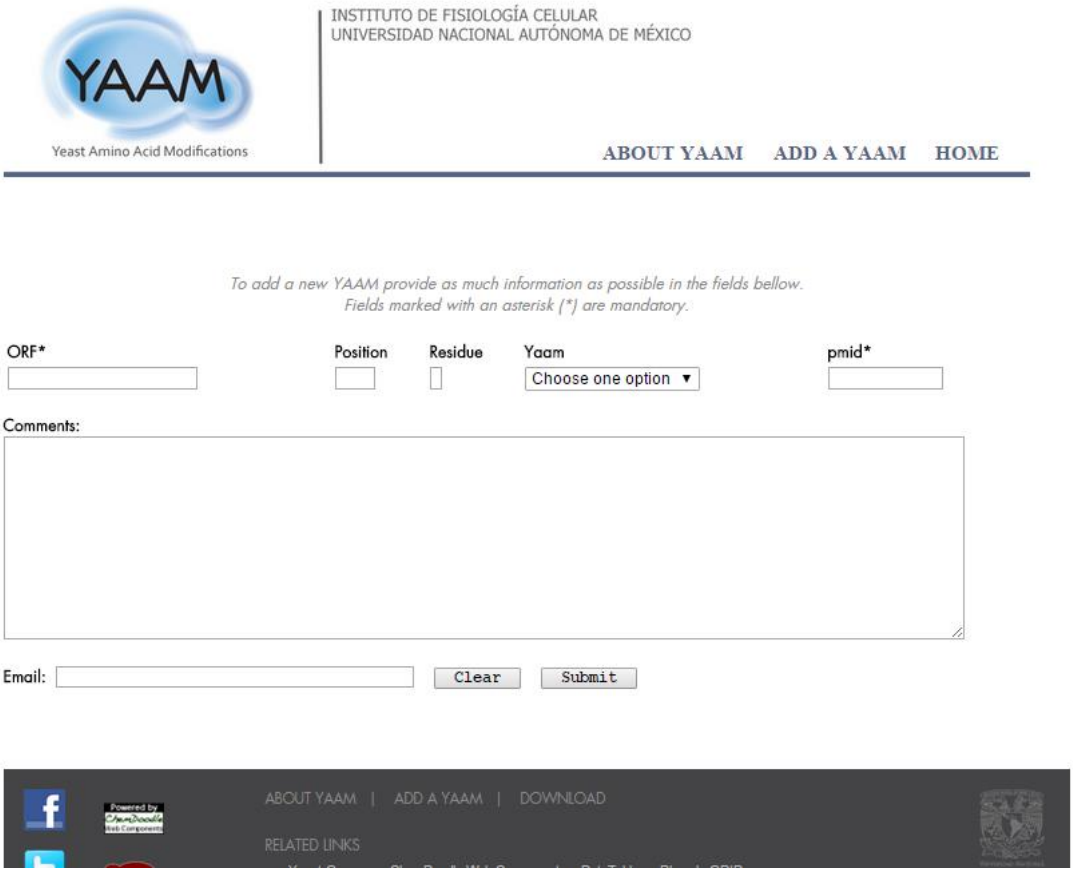


Fig. 5.7 Área de contacto para agregar una nueva modificación.

El usuario escribirá el nombre esquemático, la posición donde ocurre la modificación, el residuo, a qué tipo de modificación pertenece, el pmid del artículo que hace referencia, algún comentario que desee colocar y finalmente el correo del usuario. Este parte de la página mandará un correo a yaam@correo.ifc.unam.mx, donde el administrador validará la información.

5.3 Conclusiones.

Después del desarrollo de este proyecto y los resultados obtenidos, se concluye lo siguiente:

- Se recopiló toda la información documentada de modificaciones postraduccaciones en la levadura *Saccharomyces cerevisiae* que se tiene hasta la fecha.

- El sistema que se implentó facilita la búsqueda de la información por el cual fue diseñado.
- La página web que fue diseñada cumple con la funcionalidad y accesibilidad definida en la parte de los objetivos.
- El espacio de aportes y sugerencias que se construyó cumple con la función de actualización de la base de datos.
- Los programas elaborados durante el trancurso del desarrollo fueron de vital importancia para la curación de datos.
- La etapa de normalización de la base de datos fue dirigida a la funcionalidad y al cambio de requerimientos a futuro.
- La base de datos no presenta redudancia o anomalías de información.
- El comportamiento de la base con la página web cuando se realizan las consultas es rápido.
- Los diseños lógico, conceptual y físico de la base de datos cumplen con la teoría de las bases de datos relacionales.
- El servidor virtual presentó el comportamiento deseado durante las diferentes etapas de desarolo de este proyecto.
- La presentación de los PDBs que se pueden observar en el proyecto cumplen con las espeficicaciones establecidas por los expertos en el área.

Este proyecto es una forma de comprobar que las ciencias y la ingeniería (hablando específicamente de la ingeniería en computación) pueden trabajar en conjunto para la creación de nuevas herramientas que facilitan el desarrollo de ambas.

Se provecharón los recursos otorgados por el Instituto de Fisiología Celular y la infraestructura virtual.

REFERENCIAS

Del capítulo II, Marco teórico:

1. Lodish, H. (2004). *Molecular Cell Biology* (Fifth ed.). New York, the United States of America: Freeman, p.102. Imagen resideñada.
2. Rangel Villalobos, D. H. (12 de Enero de 2015). *DNA-Profile*. Obtenido de http://dnaprofile.com.mx/informacion-prueba-de-paternidad-adn.php
3. *El modelo de la Doble Hélice: Watson y Crick*. (12 de Enero de 2015). Obtenido de http://pendientedemigracion.ucm.es/info/genetica/grupod/Estruadn/estruadn.htm#adn
4. Lodish, H. (2004). *Molecular Cell Biology* (Fifth ed.). New York, the United States of America: Freeman, p.108. Imagen modificada.
5. Masapanta, J. (13 de Enero de 2015). *Replicación del DNA*. Obtenido de http://jessicamasapanta.blogspot.mx/2013/03/duplicacion-del-adn-en-la-hipertension.html
6. Lodish, H. (2005). *Biología Molecular y Celular* (Quinta ed.). Buenos Aires, Argentina: Panamericana, p.108.
7. Collins, F. (13 de Enero de 2015). *National Human Genome Research Institute*. Obtenido de http://www.genome.gov/GlossaryS/index.cfm?id=70
8. Lodish, H. (2004). *Molecular Cell Biology* (Fifth ed.). New York, the United States of America: Freeman, p.109.
9. *Transcripción del DNA*. (14 de Enero de 2015). Obtenido de http://datateca.unad.edu.co/contenidos/203017/Modulo_EXE/modulo_exe/leccin_38_trascripcin_del_adn.html
10. Cheung, A. (10 de Marzo de 2015). *RSCB Protein Data Bank*. Obtenido de http://www.rcsb.org/pdb/explore/explore.do?structureId=4A3G
11. Lodish, H. (2004). *Molecular Cell Biology* (Fifth ed.). New York, the United States of America: Freeman, p.114.
12. *Proceso genéticos de la síntesis de proteínas: Traducción*. (23 de Enero de 2015). Obtenido de http://pendientedemigracion.ucm.es/info/genetica/grupod/Traduccion/traduccion.htm
13. *Transfer RNA*. (14 de Marzo de 2015). Obtenido de http://www.rcsb.org/pdb/101/motm.do?momID=15

14. Lodish, H. (2004). *Molecular Cell Biology* (Fifth ed.). New York, the United States of America: Freeman, p.123. Imagen modificada

15. *Apuntes de Biología Molecular*. (22 de Enero de 2015). Obtenido de http://apuntesbiologiamol.blogspot.mx/

16. Lodish, H. (2005). *Biología Celular y Molecular* (Quinta ed.). Buenos Aires: Panamericana, p.126.

17. Ibíd. p.127.

18. Ibíd. p.130.

19. Ibíd. p.60.

20. Estadísticas de proteínas en levadura. (8 de Mayo de 2015). Obtenido de http://yeastmine.yeastgenome.org/yeastmine/bagDetails.do?bagName=Gene+list+for+all+organisms+22+Mar+2015+20.2&trackExecution=false

21. Meshcheryakov, V.A. (8 de Mayo de 2015). RSCB Protein Data Bank. Obtenido de http://www.rcsb.org/pdb/explore/explore.do?structureId=3AZD

22. Kursula, P. (8 de Mayo de 2015).RSCB Protein Data Bank. Obtenido de http://www.rcsb.org/pdb/explore/explore.do?structureId=1ZUK

23. Jia, J. (8 de Mayo de 2015). RSCB Protein Data Bank. Obtenido de http://www.rcsb.org/pdb/explore/explore.do?structureId=1K94

24. Jia, J. (9 de Mayo de 2015). RSCB Protein Data Bank. Obtenido de http://www.rcsb.org/pdb/explore/explore.do?structureId=1K94

25. Sánchez, J.G. (9 de Mayo de 2015). RSCB Protein Data Bank. Obtenido de http://www.rcsb.org/pdb/explore/explore.do?structureId=4LTB

26. Elrod-Erickson, M. (9 de Mayo de 2015). RSCB Protein Data Bank. Obtenido de http://www.rcsb.org/pdb/explore/explore.do?structureId=1A1L

27. Jia, J. (9 de Mayo de 2015). RSCB Protein Data Bank. Obtenido de http://www.rcsb.org/pdb/explore/explore.do?structureId=1K94

28. Day, J.E. (9 de Mayo de 2015). RSCB Protein Data Bank. Obtenido de http://www.rcsb.org/pdb/explore/explore.do?structureId=2XX2

29. Sudhakaran Prabakaran, Guy Lippens, Hanno Steen and Jeremy Gunawardena. (2012). Advanced Review Post-translational modification: nature's escape from genetic imprisonment and the basis for dynamic

information encoding. WIREs Syst Biol Med, p.4. 19 de Mayo de 2015, De PubMed Base de datos.

30. A. Saskia Venne, Laxmikanth Kollipara and René P. Zahedi. (2013). The next level of complexity: Crosstalk of posttranslational modifications. Proteomics 2014, 14, p. 514-517. 19 de Mayo de 2015, De pmic Base de datos.

31. Ibíd. p.514.

32. Lodish, H. (2005). *Biología Molecular y Celular* (Quinta ed.). Buenos Aires, Argentina: Panamericana, p.71.

33. Xiang-Jiao Yang, Edward Seto. (2008). Lysine Acetylation: Codified Crosstalk with Other Posttranslational Modifications, p.449-455. 22 de Agosto de 2008, De molcel Base de datos.

34. Weinert BT1, Schölz C, Wagner SA, Iesmantavicius V, Su D, Daniel JA, Choudhary C.. (Agosto 2013). Lysine succinylation is a frequently occurring modification in prokaryotes and eukaryotes and extensively overlaps with acetylation. De PubMed Base de datos.

Bibiografía

Para consulta de información contenida en este capítulo:

Lodish, H. (2004). *Molecular Cell Biology* (Fifth ed.). New York, the United States of America: Freeman, pp.29-131

Alberts, B. (2010). Biología Molecular de la Célula. (Quinta Edición). Barcelona, España, Ediciones Omega. pp. 1-323.

Sudhakaran Prabakaran, Guy Lippens, Hanno Steen and Jeremy Gunawardena. (2012). Advanced Review Post-translational modification: nature's escape from genetic imprisonment and the basis for dynamic information encoding. WIREs Syst Biol Med, 19 de Mayo de 2015, De PubMed Base de datos.

A. Saskia Venne, Laxmikanth Kollipara and René P. Zahedi. (2013). The next level of complexity: Crosstalk of posttranslational modifications. Proteomics 2014, 14, 19 de Mayo de 2015, De pmic Base de datos.

Xiang-Jiao Yang and Edward Seto. (2008). Lysine Acetylation: Codified Crosstalk with Other Posttranslational Modifications. 22 de Agosto de 2008, De molcel Base de datos.

Christian Schöneich, Lorena B. Barrón. (2006). Posttranslational Oxidative Modifications of Proteins. De Encyclopedia of Analytical Chemistry, Online Base de datos.

Zhiyou Cai, Liang-Jun Yan. (2013). Protein Oxidative Modifications: Beneficial Roles in Disease and Health. Journal of Biochemical and Pharmacological Research, vol.1, p. 15-26. De Encyclopedia of Analytical Chemistry, Online Base de datos.

Del capítulo III, Aplicaciones y la bionformática:

1. *DNA Sequencing: Strong Stop solved with Templiphi*. (29 de Mayo de 2015). Obtenido de https://pmgf.osu.edu/services/sequencing/templiphi
2. TCOFFEE - Multiple Alignment. (29 de Mayo de 2015). *Examples*. Obtenido de http://cs.mcgill.ca/~birch/tutorials/bioLegato/multalign/multalign.html
3. CLUSTALX - Alignment by hierarchical clustering (29 de Mayo de 2015). *Examples*. Obtenido de http://cs.mcgill.ca/~birch/tutorials/bioLegato/multalign/multalign.html
4. National Institute of General Medical Sciences (29 de Mayo de 2015).. Obtenido de http://images.nigms.nih.gov/index.cfm?event=viewDetail&imageID=2735

Del capítulo IV, Desarollo

1. S*accharomyces cerevisiae Genome Databse* (24 de Mayo de 2015). Obtenido de http://www.yeastgenome.org/
2. *Phosphorylation Site Research* (24 de Mayo de 2015). *Examples*. Obtenido de http://www.phosphogrid.org/
3. ProteomicsDB powered (24 de Mayo de 2015) de https://www.proteomicsdb.org/

APÉNDICE

Curación Datos

pmid_parser.py

```
infile=open("C:\Users\Eduardo\Documents\\BaseDeDatoswBio\\Ubiparser\\ubiquitinacion_vacios.txt","r")
#infile2=open("C:\Users\Eduardo\Documents\\uniprot_ID.txt","r")
outfile=open("C:\Users\Eduardo\Documents\\BaseDeDatoswBio\\Ubiparser\\ubi_vaciospmid.txt","w")

for line in infile:
   if line[-1] == '\n':
      line=line[:-1]

   #line=line.replace(" ",'')

   lista=line.split('\t')


   pmid=lista[5]

   if pmid[-1]=='\n':
      pmid = pmid[:-1]
   if pmid[0]==' ':
      pmid=pmid[1:]

   if(pmid ==
'2012_Henriksen'):
      lista[5]='5'

elif(pmid=="Weinert_2014
"):
      lista[5]='2'

elif(pmid=="2013_Weinert
_Lysine"):
      lista[5]='3'

elif(pmid=="2014_Weinert
_Lysine"):
      lista[5]='4'
   elif(pmid=="Paco"):
      lista[5]='6'



outfile.write('\t'+lista[1]+'\t
'+lista[2]+'\t'+lista[3]+"\t"+l
ista[4]+'\t'+lista[5]+'\n')

#outfile.write('\t'+lista[1]+'
\t'+lista[2]+'\t'+lista[3]+"\t"
+lista[4]+"\t"+'7'+'\n')
infile.close()
outfile.close()
```

acetil_nuevos.pl

```
open(EN,"Acetilation.csv")|
|die;

while(<EN>) {
        chomp;
        my($idprot,$seq,$p
os,$residuo,$pmid)= split
/\t/,$_;
        print
"\t$idprot\t$seq\t$residuo
\t$pos\t524\n"
}
close(EN)
```

acetilCorrec.pl

```
#!/usr/bin/perl

#Analisis de la modificacion
ACETILACION
#para la correcion de
errores
#abrimos articulo
my $proteoma="proto.txt";
my %proto;

open(IN,$proteoma)||
die('File not found');
while(<IN>){
        my
(undef,$idprot,$seqfather,
undef)= split /\t/,$_;

$universe{$idprot}=$seqfat
her;
}
close(IN);

$libro1="2012_Henriksen_
acetylome_S1.txt";
$libro2="2012_Henriksen_
acetylome_S1b.txt";
my %acetil;
my %acetil2;


my @prots;
my @posis;
$cont=0;
#libro 1 Henriksen
open(IN,$libro1)||die('File
not found');
while(<IN>){
        chomp;
        my
($proteinas,$posiciones,un
def,undef,undef,undef,und
ef,undef,undef,undef,unde
f,undef,undef,$peptidos,un
def)= split /\t/,$_;
        #print "$proteinas
con $posiciones en
$peptidos\n";
```

```
	if($proteinas=~
/;/){
		@prots=
split /;/,$proteinas;
	}
	if ($peptidos=~ /;/){
		@cadenas=
split /;/,$peptidos;
	}


	if ($posiciones=~
/;/){
		@posis=
split /;/, $posiciones;
		for (my
$i=0;
$i<scalar(@posis);$i++){

	#print
"$prots[$i]\t$posis[$i]\n";

				if
($peptidos=~ /;/){

	for (my $kk=0;
$kk<scalar(@cadenas);$kk
++){


	$cadenas[$kk]=~
s/_//g;

		if
(($universe{$prots[$i]})=~
/$cadenas[$kk]/){


	$razon=$cadenas[$
kk];

		}

	}
```

```
	$acetil{$prots[$i]."
*".$posis[$i]}= $razon;

	}else{

	$peptidos=~
s/_//g;

	$acetil{$prots[$i]."
*".$posis[$i]}= $peptidos;
				}
			}
	}else{

	#print
"$proteinas\t$posiciones\n
";

	$peptidos=~
s/_//g;

	$acetil{$proteinas.
"*".$posiciones}=
$peptidos;
	}
}
close(IN);

#libro 2 Henriksen
open(IN,$libro2)|| die('File
not found');
while(<IN>){
	chomp;
	my
($proteinas,$posiciones,un
def,undef,undef,undef,und
ef,undef,$peptidos,undef)=
split /\t/,$_;
	#print "$proteinas
con $posiciones en
$peptidos\n";
	if($proteinas=~
/;/){
```

```
		@prots=
split /;/,$proteinas;
	}
	if ($peptidos=~ /;/){
		@cadenas=
split /;/,$peptidos;
	}
	if ($posiciones=~
/;/){
		@posis=
split /;/, $posiciones;
		for (my
$i=0;
$i<scalar(@posis);$i++){

	#print
"$prots[$i]\t$posis[$i]\n";
				if
($peptidos=~ /;/){

	for (my $kk=0;
$kk<scalar(@cadenas);$kk
++){


	$cadenas[$kk]=~
s/_//g;

		if
(($universe{$prots[$i]})=~
/$cadenas[$kk]/){


	$razon=$cadenas[$
kk];

		}

	}

	$acetil{$prots[$i]."
*".$posis[$i]}= $razon;

	delete
```

```
$acetil{$prots[$i]."*".$posis[$i]},if(($prots[$i] eq 'YJL114W') && ($posis[$i] == 400));

        }else{

        $peptidos=~ s/_//g;

        $acetil{$prots[$i]."*".$posis[$i]}= $peptidos;

        delete $acetil{$prots[$i]."*".$posis[$i]},if(($prots[$i] eq 'YJL114W') && ($posis[$i] == 400));
                        }
                }
        }else{

        #print "$proteinas\t$posiciones\n";

        $peptidos=~ s/_//g;

        $acetil{$proteinas."*".$posiciones}= $peptidos;

        delete $acetil{$proteinas."*".$posiciones},if(($proteinas eq 'YJL114W') && ($posiciones == 400));
        }
}
close(IN);

#$contador= keys %acetil;
#print $contador."\n";
#foreach $key (keys %acetil){        print "$key con $acetil{$key}\n";}


$file="acetylationafter2.csv";
open(EN,$file)||die('File not found');
while(<EN>){
        chomp;
        my ($idbd,$idprot,$seq,$residuo,$pos,$fuente)= split /,/,$_;
        #Verificando si todos los que estan en la BD estan en Henriksen
        $seq=~ s/_//g;
        if ($fuente == 1){
                #Las que estan erroneas
                foreach $j (keys %acetil){

        #error en posicion
                        if ($j=~/^$idprot/){

        if($seq eq $acetil{$j}){

                my ($iden,$poscomp)=split /\*/,$j;

                #print "error en:$idprot con la pos:$pos deberia ser $poscomp con $seq\n",if ($pos ne $poscomp);

                #son 79 que tienen la posicion mal con respecto a la seq

        $acetil{$j}.="1";

        }
                        }

        #error de secuencia
                        if ($j=~/^$idprot/){

        my ($iden,$poscomp)=split /\*/,$j;

        if($pos eq $poscomp){

                #print "error en:$idprot con la pos:$pos con $seq deberia ser $acetil{$j}\n",if ($seq ne $acetil{$j});

                #2 secuencias equivocadas

        #$acetil{$j}.="1";


        }
                        }
                }
                #print "no existo en articulos $idprot con la pos:$pos con $seq\n", if ((exists($acetil{$idprot}))
```

```
&&
($pos==($acetil{$idprot})));
	}

}
close(EN);


#termina Henriksen 2012
inician los dos libro de
Weinert 2014
$libro3="2014_Weinrt_sup
plementary_material_17.tx
t";
$libro4="2014_Weinrt_sup
plementary_material_14.tx
t";
$libro5="2014_Weinrt_sup
plementary_material_11.tx
t";

open(RA,$libro3)||
die('File not found');
while(<RA>){
	my($id,$posiciones
,undef,$proteinas,undef,un
def,undef,$peptidos,undef)
= split /\t/,$_;
	#print "$proteinas
con $seq\n";
	if($proteinas=~
/;/){
		@prots=
split /;/,$proteinas;
	}
	if ($peptidos=~ /;/){
		@cadenas=
split /;/,$peptidos;
	}


	if ($posiciones=~
/;/){
		@posis=
split /;/, $posiciones;
		for (my
$i=0;
$i<scalar(@posis);$i++){

	#print
"$prots[$i]\t$posis[$i]\n";

			if
($peptidos=~ /;/){

	for (my $kk=0;
$kk<scalar(@cadenas);$kk
++){


	$cadenas[$kk]=~
s/_//g;

		if
(($universe{$prots[$i]})=~
/$cadenas[$kk]/){


	$razon=$cadenas[$
kk];

		}

	}

	$acetil2{$prots[$i].
"*".$posis[$i]}= $razon;

	}else{

	$peptidos=~
s/_//g;

	$acetil2{$prots[$i].
"*".$posis[$i]}= $peptidos;
			}
		}
	}else{

	#print
"$proteinas\t$posiciones\n
";

	$peptidos=~
s/_//g;

	$acetil2{$proteinas
."*".$posiciones}=
$peptidos;
	}
}
close(RA);

open(RB,$libro4)|| die('File
not found');
while(<RB>){
	my($id,$posiciones
,undef,$proteinas,undef,un
def,undef,$peptidos,undef)
= split /\t/,$_;
	if($proteinas=~
/;/){
		@prots=
split /;/,$proteinas;
	}
	if ($peptidos=~ /;/){
		@cadenas=
split /;/,$peptidos;
	}


	if ($posiciones=~
/;/){
		@posis=
split /;/, $posiciones;
		for (my
$i=0;
$i<scalar(@posis);$i++){

	#print
```

```
"$prots[$i]\t$posis[$i]\n";

                        if
($peptidos=~ /;/){

        for (my $kk=0;
$kk<scalar(@cadenas);$kk
++){


        $cadenas[$kk]=~
s/_//g;

                if
(($universe{$prots[$i]})=~
/$cadenas[$kk]/){


        $razon=$cadenas[$
kk];

                }

        }

        $acetil2{$prots[$i].
"*".$posis[$i]}= $razon;

        }else{

        $peptidos=~
s/_//g;

        $acetil2{$prots[$i].
"*".$posis[$i]}= $peptidos;
                        }
                }
        }else{

        #print
"$proteinas\t$posiciones\n
";

        $peptidos=~
s/_//g;

        $acetil2{$proteinas
."*".$posiciones}=
$peptidos;
        }
        #print "$proteinas
con $seq\n";
}
close(RB);

open(RC,$libro5)|| die('File
not found');
while(<RC>){
        my($id,$posiciones
,undef,$proteinas,undef,un
def,undef,$peptidos,undef)
= split /\t/,$_;
        if($proteinas=~
/;/){
                @prots=
split /;/,$proteinas;
        }
        if ($peptidos=~ /;/){
                @cadenas=
split /;/,$peptidos;
        }


        if ($posiciones=~
/;/){
                @posis=
split /;/, $posiciones;
                for (my
$i=0;
$i<scalar(@posis);$i++){

        #print
"$prots[$i]\t$posis[$i]\n";

                        if
($peptidos=~ /;/){

        for (my $kk=0;
$kk<scalar(@cadenas);$kk
++){


        $cadenas[$kk]=~
s/_//g;

                if
(($universe{$prots[$i]})=~
/$cadenas[$kk]/){


        $razon=$cadenas[$
kk];

                }

        }

        $acetil2{$prots[$i].
"*".$posis[$i]}= $razon;

        delete
$acetil2{$prots[$i]."*".$po
sis[$i]},if(($prots[$i] eq
'YHR214C-C') &&
($posis[$i] == 250));

        delete
$acetil2{$prots[$i]."*".$po
sis[$i]},if(($prots[$i] eq
'YHR214C-C') &&
($posis[$i] == 322));

        delete
$acetil2{$prots[$i]."*".$po
sis[$i]},if(($prots[$i] eq
'YGL041W-A') &&
($posis[$i] == 54));

        delete
$acetil2{$prots[$i]."*".$po
```

```
sis[$i]},if(($prots[$i] eq
'YGL129C') && ($posis[$i]
== 260));

        #print "$prots[$i]
con $posis[$i] con
$posis_new en $razon";

        }else{

        $peptidos=~
s/_//g;

        $acetil2{$prots[$i].
"*".$posis[$i]}= $peptidos;

        delete
$acetil2{$prots[$i]."*".$po
sis[$i]},if(($prots[$i] eq
'YHR214C-C') &&
($posis[$i] == 250));

        delete
$acetil2{$prots[$i]."*".$po
sis[$i]},if(($prots[$i] eq
'YHR214C-C') &&
($posis[$i] == 322));

        delete
$acetil2{$prots[$i]."*".$po
sis[$i]},if(($prots[$i] eq
'YGL041W-A') &&
($posis[$i] == 54));

        delete
$acetil2{$prots[$i]."*".$po
sis[$i]},if(($prots[$i] eq
'YGL129C') && ($posis[$i]
== 260));

        #print
"$proteinas\t$posiciones\n
";
                }
                }
        }else{

        #print
"$proteinas\t$posiciones\n
";

        $peptidos=~
s/_//g;

        $acetil2{$proteinas
."*".$posiciones}=
$peptidos;

        delete
$acetil2{$proteinas."*".$p
osiciones},if(($proteinas eq
'YHR214C-C') &&
($posiciones == 250));

        delete
$acetil2{$proteinas."*".$p
osiciones},if(($proteinas eq
'YHR214C-C') &&
($posiciones == 322));

        delete
$acetil2{$proteinas."*".$p
osiciones},if(($proteinas eq
'YGL041W-A') &&
($posiciones == 54));

        delete
$acetil2{$proteinas."*".$p
osiciones},if(($proteinas eq
'YGL129C') && ($posiciones
== 260));
        }
        #print "$proteinas
con $seq\n";

}
close(RC);
#foreach $key (keys
%acetil2){      print "$key
con $acetil2{$key}\n";}

open(EN,$file)||die('File
not found');
while(<EN>){
        chomp;
        my
($idbd,$idprot,$seq,$resid
uo,$pos,$fuente)= split
/,/,$_;
        #Verificando si
todos los que estan en la
BD estan en Henriksen
        $seq=~ s/_//g;
        if ($fuente == 2){
                #Las que
estan erroneas
                foreach $jj
(keys %acetil2){

        #error en posicion
                        if
($jj=~/^$idprot/){

        if($seq eq
$acetil2{$jj}){

                my
($iden,$poscomp)=split
/\*/,$jj;

                #print
"error en:$idprot con la
pos:$pos deberia ser
$poscomp con $seq\n",if
($pos ne $poscomp);

                #son 96
que tienen la posicion mal
con respecto a la seq
```

```
	$acetil2{$jj}.="1";

	}
			}

	#error de
secuencia
			if
($jj=~/^$idprot/){

	my
($iden,$poscomp)=split
/\*/,$jj;

	if($pos eq
$poscomp){

		#print
"error en:$idprot con la
pos:$pos con $seq deberia
ser $acetil2{$jj}\n",if ($seq
ne $acetil2{$jj});

		#35
secuencias equivocadas


	#$acetil2{$jj}.="1";



	}
			}
		}

		print "no
existo en articulos $idprot
con la pos:$pos con
$seq\n", if
((exists($acetil2{$idprot}))
&&
($pos==($acetil2{$idprot}))
);
		}
		#else{ #print "no
entre\n"}
}
close(EN);

#una vez que se arreglaron
las secuencias verificar
cuantas modificaciones
faltan por agregarse
#MIENTRAS NO
#foreach $key (keys
%acetil){ print "$key con
$acetil{$key}\n";}
foreach $key (keys
%acetil){
	if
($acetil{$key}=~/1$/){
		#print
"modoficacion_noagregada
: $key con $acetil{$key}\n";
	}else
	{
		#pre-
control secuencia por si el
articulo esta mal
		#apelamos
a la secuencia
		my
($iden,$poscomp)=split
/\*/,$key;
		foreach
$llave (keys %universe){
			if
($llave eq $iden){

	#print
$universe{$llave};

	my @secuencia=
split //,$universe{$llave};

	if(@secuencia[$pos
comp-1] eq 'K'){

		#print
"modoficacion_noagregada
: $key con $acetil{$key}\n";

		print
"\t$iden\t$acetil{$key}\tK\
t$poscomp\t1\n";

	}else{

		#print
"error en el aritculo porque
$key con $acetil{$key} no
hay Lys\n";

	}
			}
		}
		}
}

foreach $key (keys
%acetil2){
	if
($acetil2{$key}=~/1$/){
		#print
"modoficacion_noagregada
: $key con $acetil{$key}\n";
	}else
	{
		#pre-
control secuencia por si el
articulo esta mal
		#apelamos
a la secuencia
		my
($iden,$poscomp)=split
/\*/,$key;
		foreach
$llave (keys %universe){
```

```
                    if
($llave eq $iden){

        #print
$universe{$llave};

        my @secuencia=
split //,$universe{$llave};

        if(@secuencia[$pos
comp-1] eq 'K'){

                #print
"modoficacion_noagregada
: $key con
$acetil2{$key}\n";

                print
"\t$iden\t$acetil2{$key}\tK
\t$poscomp\t2\n";

        }else{

                #print
"error en el aritculo porque
$key con $acetil2{$key} no
hay Lys\n";

                #error en el
aritculo porque
YBR085W*98 con
ALNFAFKDKIKAM no hay
Lys

        }
                    }
                }
                }
}
```

parser.py

```
infile=open("C:\Users\Edua
rdo\Documents\BaseDeDa
toswBio\\fosfoparser\Resi
duosnoREPLICADOS\\fosfo
rilacion.txt","r")
#infile2=open("C:\Users\Ed
uardo\Documents\\fosforil
acion_last.txt","r")
outfile=open("C:\Users\Ed
uardo\Documents\\fosforil
aciones_last.txt","w")
for line in infile:


newstr=
line.replace("\'","")

newstr2=newstr.replace("[
","")

newstr3=newstr2.replace("
]","")

newstr4=newstr3.replace('
','')
   print newstr4
   lista=newstr4.split(',')


   for item in lista:
      outfile.write(item+'\t')

outfile.close()
infile.close()
```

residuofosfo.py

```
infile=open("C:\Users\Edua
rdo\Documents\\fosforilaci
ones_last.txt","r")
#infile2=open("C:\Users\Ed
uardo\Documents\\fosforil
acion_last.txt","r")
#outfile=open("C:\Users\E
duardo\Documents\\fosfor
ilaciones_last_last.txt","w")

for line in infile:

   lista=line.split('\t')
   peptidos=lista[2]

listapeptidos=list(peptidos)
   print listapeptidos
   residuo= int(lista[4])-
int(lista[5]);
   print residuo



   #for item in lista:
   #outfile.write(peptidos)

#outfile.close()
infile.close()
#infile2.close()
```

calcuimf(2).py

```
infile=open("C:\Users\Edua
rdo\Documents\\BaseDeD
atoswBio\\fosfoparser\\fos
forilaciones_dos.txt","r")
infile2=open("C:\Users\Edu
ardo\Documents\\proto.tx
t","r")
outfile=open("C:\Users\Ed
uardo\Documents\\BaseDe
DatoswBio\\phospo_pen.tx
t","w")

pt={}

for line in infile2:
   listadicc=line.split('\t')
```

```
pt[listadicc[1]]=listadicc[2]


for line in infile:
   if line[-1]=='\n':
      line = line[:-1]

   line=line.replace(" ",'')

#lista2=infile2.readline().sp
lit('\t')
   lista1=line.split('\t')


   protkey = lista1[1]
   pos=lista1[4]
   try:
      pept=pt[protkey]
   except KeyError as ee:
      res= "no"
      print protkey
   prot=list(pept)
   posi=int(pos)
   try:
      res=prot[posi-1]
   except IndexError as e:
      res= "nel"

outfile.write("\t"+lista1[1]+
"\t"+lista1[2]+"\t"+res+"\t"
+pos+"\t"+'6'+"\n")
   #outfile.write(res+"\n")


outfile.close()
infile.close()
infile2.close()
```

ubi.pl

```
#!/usr/bin/perl -w

open(IN,"ubiquitinacion_va
ciospmid.txt") || die;

my %proteins;

while(<IN>) {
  chomp;

my(undef,$id,undef,$res,$
pos,$pmid) = split /\t/, $_;
  $res =~ s/\s+//g;
  $pos =~ s/\s+//g;
  $pmid =~ s/\s+//g;
  if($pmid eq '5') {
    $pos = $pos + 1;
  }
  $proteins{$id.$pmid} .=
"$pos ";

}

foreach my $k (keys
%proteins) {
  my %pos_hash;
  my @values = split /\s/,
$proteins{$k};
  foreach my $v (@values)
{
    $pos_hash{$v} = 1;
  }
  my $valores = join " ",
(sort {$a <=> $b} keys
%pos_hash);
  print "$k\t$valores\n";
}
```

parser3.py

```
def findST(cadena):

cadenapar=cadena.find('(')
   subcadena=
cadena[:cadenapar]
   return subcadena[-1]

def drift(cadena):
   sub=cadena.split('(ph)')
   return sub

infile=open("C:\Users\Edua
rdo\Documents\\ubiquitin
acion_last_last.txt","r")
infile2=open("C:\Users\Edu
ardo\Documents\\Ubiquiti
naciones_last_last.txt","r")

outfile=open("C:\Users\Ed
uardo\Documents\\Ubiquit
inaciones.txt","w")
for line in infile:
   for line2 in infile2:

numero=line.count(line2);
   print numero;



outfile.close()
infile.close()
```

phospho.py

```
infile=open("C:\Users\Edua
rdo\Documents\infoPacoVi
ejos\\acetilacion.txt","r")
#infile2=open("C:\Users\Ed
uardo\Documents\\fosforil
acion_last.txt","r")
outfile=open("C:\Users\Ed
uardo\Documents\\BaseDe
DatoswBio\\oxi2.txt","w")
for line in infile:


newstr=
line.replace("\'","")
```

```
newstr2=newstr.replace("[
","")

newstr3=newstr2.replace("
]","")

newstr4=newstr3.replace('
','')
   print newstr4
   lista=newstr4.split(',')


   for item in lista:
      outfile.write(item+'\t')

outfile.close()
infile.close()
```

detsecuencias.pl

```
open(EN,"phoso_nuevas.tx
t")||die;
open(IN,"proto.txt")||die;

my %seqs;
while(<IN>) {
        chomp;
        my(undef,$idprot,$
secuencia,undef)= split
/\t/,$_;
        #print
"$idprot\t$secuencia\n";
        $seqs{$idprot}=$se
cuencia;
}

while(<EN>){
        chomp;
        my(undef,$idp,$res
,$pos,$pmid)= split /\t/,$_;
        my $peptidos  =
substr $seqs{$idp}, $pos,
10;
        print
"\t$idp\t$peptidos\t$res\t
$pos\t$pmid\n";

}

close(IN);
close(EN);
```

deteccion.pl

```
open(EN,"pmids.txt")||die;
open(IN,"phosphosites.txt"
)||die;

my %pmids;
my %hash_pmids;
my $i=20;
while(<EN>) {
        chomp;
        my($idp,$pmidf)=
split /\s/,$_;
        #print
"$idp\t$pmidf\n";
        $pmids{$idp}=$pmi
df;
}

#comprobando fuentes
#foreach $k(keys
%pmids){print
"$k\t$pmids{$k}\n";}

while(<IN>) {
        chomp;
        my($id,undef,$res_
pos,undef,$pmid,undef)=
split /\t/,$_;
        #print "$pmid\n";

        #$pmid = ~ s/\|/'
'/g;
        @pmid_before =
split /\|/,$pmid;
        foreach my
$val(@pmid_before){
        if (exists
($hash_pmids{$val}))
        {
        my
($res,$garbache) = split
/\d+/,$res_pos;
        #my
($name,$lastname)= split
/\w/,$proof;
        my
($garbache2,$pos) = split
/^[A-Z]/,$res_pos;
        #print
"\t$id\t$res\t$pos\t$hash_
pmids{$val}\n";
        #print
"$res_pos\n";
        }
        else{

        $hash_pmids{$val}
=$i;
                $i++;
                my
($res,$garbache) = split
/\d+/,$res_pos;
                #my
($name,$lastname)= split
/\w/,$proof;
                my
($garbache2,$pos) = split
/^[A-Z]/,$res_pos;
                #print
"\t$id\t$res\t$pos\t$hash_
pmids{$val}\n";
```

```
                #print
"$res_pos\n";
        }
}
}
#NUMERO MAXIMO DE
PMDIS EN ARCHIVO
#print "$i";
foreach $k(keys
%hash_pmids){print
"$k\t$hash_pmids{$k}\n";}


close(EN);
close(IN);
```

asocacion.pl

```
#open(IN,"ubipmid.txt") ||
die;
open(IN,"ubiquitinacion_va
ciospmid.txt")||die;

my %proteins;

while(<IN>) {
        chomp;
        my @posiciones=[];
        my(undef,$idprot,u
ndef,$res,$pos,$fuente) =
split /\t/, $_;
        if (not exists
$proteins{$idprot}){
        push(@$posiciones
,$pos);
        #@posiciones=($po
s,@posiciones);
        $proteins{$idprot}=
$posiciones;
        }else{
                #print
        "entre2","\n";
                if ($pos ~~
@{$proteins{$idprot}}){
                #if
(exists($proteins{$idprot}))
{

        print "posicion
repetida, gracias","\n";

        }else
                {

        @posiciones=$prot
eins{$idprot};

        push(@$posiciones
,$pos);

        $proteins{$idprot}=
$posiciones;

        #@posiciones=($po
s,@posiciones);
                }
#print
"\t",$idprot,"\t",$seq,"\t",$
res,"\t",$pos_ac,"\t",$fuen
te,"\n";
}
}


        foreach my
$proteina(keys %proteins)
        {
                #my
@posicion =
$proteins{$proteina};
                print
"protein: ". $proteina,"\t";
                foreach
(@{$proteins{$proteina}})
                        {

        print $_,",";

                        }
                print "\n";
        }

        my
$number=scalar(keys
%proteins);
        print $number;

#$pos_ac= $pos;
#$pos_ac++;
close(IN);
```

mighty.pl

```
#!/usr/bin/perl -w

open(IN,"ubiquitinacion.txt
") || die;

while(<IN>) {
  chomp;
  my $line = $_;
    $line =~ s/^\[//;
        $line =~ s/\]$//;

  my
($id,$peptidos,undef,undef
,$pos,$biblio) = split /,/,
$line;

  $unicos{$id."\:".$pos} =
"$peptidos\:$biblio";

#   print
"\t$id\t$peptidos\t,'K',\t$p
os\t$biblio\n";

}

foreach $key (keys
%unicos) {
```

```
  my($ident,$sitio) = split
/\:/, $key;
  my($p,$b) = split /\:/,
$unicos{$key};


  $ident =~ s/\'//g;
  $p =~ s/\'//g;
  $b=~ s/\'//g;
  $res='K'; $res=~ s/\'//g;



# print
"$ident\t$s\t$r\t$sitio\t$b
\n";
  print
"\t$ident\t$p\t$res\t$sitio
\t$b\n";
}
```

parser_u.pl

```
#!/usr/bin/perl -w

open(IN,"ubiquitinacion_la
st_last.txt") || die;
while(<IN>) {
  chomp;

my(undef,$id,$seq,$residu
o,$lugar,$biblio) = split
/\s+/, $_;
  $id=~ s/\s+//g;
  $lugar =~ s/\s+//g;
  $residuo=~ s/\s+//g;
  $unicos{$id."\:".$lugar} =
"$seq\:$residuo\:$biblio";

}

foreach $key (keys
%unicos) {
  my($ident,$sitio) = split
/\:/, $key;
  my($s,$r,$b) = split /\:/,
$unicos{$key};
  print
"\t$ident\t$s\t$r\t$sitio\t$
b\n";
}
```

tosave.pl

```
open(IN,"primerapart.txt")
|| die;
open(EN,"ubi_nuevo.txt")|
|die;

my %prot;
$ik=0;
#TO SAVE seqs peptidos
while(<EN>) {
  chomp;

my(undef,$id,$seq,$res,$p
os,$pmid) = split /\t/, $_;

  #posibles errores
  #$res =~ s/\s+//g;
  #$pos =~ s/\s+//g;
  #$pmid =~ s/\s+//g;
  $id=~ s/\s+//g;
  #$seq=~ s/\s+//g;

$prot{$id."."$pos}="$seq:$
pmid";
}
close (EN);

#foreach my $k (keys
%prot) { print
"$k\t$prot{$k}\n";}
```

```
while(<IN>) {
  chomp;

my(undef,$idprot,$res2,$p
os2,$pmid2) = split /\t/, $_;
  #buscando secuencia por
idprot
  if (exists
$prot{$idprot."."$pos2}){

        my
($seq_ord,$pmid_ord) =
split
/:/,$prot{$idprot."."$pos2}
;
        if ($pmid2==5) {
        print
"\t$idprot\t$seq_ord\t$res
2\t$pos2\t$pmid2\n";
        delete
($prot{$idprot."."$pos2});
    }
        else{
        print
"\t$idprot\t\t$res2\t$pos2
\t$pmid2\n";
        }
  }
  else{
        print
"\t$idprot\t\t$res2\t$pos2
\t$pmid2\n";
  }
  #print "$prot{$idprot}\n";
}
 #foreach my $k (keys
%prot) { print
"$k\t$prot{$k}\n";}
close (IN);
```

try.pl

```
#!/usr/bin/perl -w
open(IN,"ubiquitinacion_vaciospmid.txt")||die;

my %proteins;
my @posiciones=();

while(<IN>) {
        chomp;

        my(undef,$idprot,undef,$res,$pos,$fuente) = split /\t/, $_;

        $res =~ s/\s+//g;
        $pos =~ s/\s+//g;
        $fuente =~ s/\s+//g;

        $keygen=$idprot.$fuente;

        if(exists $proteins{$keygen}){

$proteins{$keygen}.="$pos"


                if (grep (/$pos/,@posiciones) eq 0)
                {

        push(@$posiciones,$pos);

        $proteins{$idprot}= $posiciones;
                }}
        else{

        push(@$posiciones,$pos);

        $proteins{$idprot}= $posiciones;
                #print "nueva cadena: ",$proteins{$idprot},"\n";

        }
}


foreach $k (keys %proteins) {
  foreach (@{$proteins{$k}}) {
    print $_,"\n";
  }
}

close(IN);
```

localizacion.pl

```
open(IN,"location2.csv") || die;

while(<IN>){
        chomp;
        my ($idprot,$num,$ubicacion) =split /\t/,$_;
        $ubicacion=~ s/:/,/g;
        print "\t",$idprot,"\t",$ubicacion,"\n";

}

close(IN);
```

name.pl

```
#!/usr/bin/perl -w
open(IN,"alias.csv") || die;

while(<IN>) {

        chomp;
#       if($_ =~ /^\w+\;\s(.*)/) {
#         print $1 , "\n";
#       }
      my $cadena;
        my(@alias)= split /\;/,$_;
      foreach my $a (@alias) {
         $cadena .= "$a\," if($a ne '');
      }
      $cadena =~ s/\,$//;
      print $cadena , "\n";
}
close(IN);
```

name.pl

```
#!/usr/bin/perl -w
open(IN,"alias.csv") || die;



while(<IN>) {

          chomp;
#       if($_ =~
/^\w+\;\s(.*)/) {
#         print $1 , "\n";
#       }
      my $cadena;
          my(@alias)= split
/\;/,$_;
      foreach my $a (@alias)
{
        $cadena .= "$a\,"
if($a ne '');
      }
      $cadena =~ s/\,$//;
      print $cadena , "\n";


}
close(IN);
```

last_parser.pl

```
#!/usr/bin/perl -w
open(IN,"equiProt.txt") ||
die;



while(<IN>) {

          chomp;
          my(undef,$idprot,$
alias) = split /\t/, $_;
          my(@alias)= split
/\,/,$alias;
      foreach my $a (@alias)
{
        print
"\t",$idprot,"\t",$a,"\n";
      }


}
close(IN);
```

parprot.py

```
infile=open("C:\Users\Edua
rdo\Documents\\ParserDo
cBaseDatoswt\\proteinas_l
ast.txt","r")
infile2=open("C:\Users\Edu
ardo\Documents\\csvtotxt
\\yeast_db_campstxt.txt","
r")
outfile=open("C:\Users\Ed
uardo\Documents\\proto.t
xt","w")


pt={}
lost={}
#construccion


for line in infile2:
   if line[-1]=='\n':
      line = line[:-1]
   listadicc=line.split('\t')

lost[listadicc[0]]=listadicc[1
]+"\t"+listadicc[2]


for line in infile:
   if line[-1]=='\n':
      line = line[:-1]
   lista=line.split('\t')
   protkey = lista[0]
   seq=lista[1]

   #if(lost[protkey]):


   if(protkey in lost):

outfile.write("\t"+protkey+
"\t"+seq+"\t"+"\t"+lost[pro
tkey]+"\n")
   else:


outfile.write("\t"+protkey+
"\t"+seq+"\n")


outfile.close()
infile.close()
infile2.close()
```

nitra.pl

```
open(EN,"Nitration.csv")||
die;

while(<EN>) {
          chomp;
          my(undef,$idprot,$
seq,$pos,$residuo,$pmid)=
split /\t/,$_;
          $idprot =~ s/\s+//g;
          $pos =~ s/\s+//g;
          $residuo =~
s/\s+//g;
          $pmid =~ s/\s+//g;
```

```
        print
"\t$idprot\t$seq\t$residuo
\t$pos\t564\n"

        #print "$pos\n";
        #$seqs{$idprot}=$s
ecuencia;
}
```

metparse.pl

```
open(IN,"25109467.txt") ||
die;

$pmid='8';
$coma=',';

while(<IN>) {

        chomp;

        my($id_prot,$sequ
ence,$relevante)= split
/\t/,$_;

        my
($cadena1,$cadena2)=split
/,/,$relevante;


        my @met1=  split
//,$cadena1;


        if  (defined
($cadena2))
                {
                my
@met2= split //,$cadena2;
                #print
$cadena1,"\t",$cadena2,"\
n";
                #print
$met1[0],"\t",$met2[0],"\n
";
                #para la
modificacion 1

        $tipo=$met1[0];

        $residuo=$met1[1]
;

                if($met1[3]
eq '')
                {

        $pos=$met1[2];
                }
                else{

        if($met1[4] eq '')
                                {

        $pos=$met1[2].$m
et1[3]; }

        else{

        if($met1[5] eq ''){

        $pos=$met1[2].$m
et1[3].$met1[4];
}else{$pos=$met1[2].$met
1[3].$met1[4].$met1[5]}

                                }
                }


        $tipo2=$met2[0];

        $residuo2=$met2[1
];


        $secuencia2=$sequ
ence;
                #print
$secuencia2,"\n";

                if($met2[3]
eq '')
                {

        $pos2=$met2[2];
                }
                else{

        if($met2[4] eq '')
                        {

        $pos2=$met2[2].$
met2[3]; }

        else{

        if($met2[5] eq ''){

        $pos2=$met2[2].$
met2[3].$met2[4];
}else{$pos2=$met2[2].$me
t2[3].$met2[4].$met2[5]}

                        }
                }

                print
$id_prot,"\t",$sequence,"\
t",$residuo,"\t",$pos,"\t",$
pmid,"\n";
                print
$id_prot,"\t",$sequence,"\
t",$residuo2,"\t",$pos2,"\t
",$pmid,"\n";
```

```
	}

	else{
		#print
$cadena1,"\n";

	$tipo=$met1[0];

	$residuo=$met1[1]
;

	$pos=$met1[2].$m
et1[3].$met1[4].$met1[5];
		print
$id_prot,"\t",$sequence,"\
t",$residuo,"\t",$pos,"\t",$
pmid,"\n";
		#print
$met1[0],"\n";
	}
}
close(IN);
```

equal.pl

```
open(IN,"acetilequival.txt")
||die;
open(ID,"Ntermini-
acetylation.csv")||die;

my %equals;
while(<IN>) {
	chomp;
	my($id,$id2)= split
/\s+/,$_;
	$id =~ s/\s+//g;
	$id2 =~ s/\s+//g;
	$equals{$id2}=$id;
	#print
$id,"\t",$id2,"\n";
}
close(IN);

while(<ID>) {
	chomp;
	my($idprot_uni,$p
os,$seq,$pmid) = split /\t/,
$_;
	my @seqS = split
//,$seq;
	#foreach (@seqS) {
print $_,"\t";}
	if (exists
$equals{$idprot_uni}) {
	#print
$equals{$idprot_uni},"\n";
	print
"\t",$equals{$idprot_uni},"
\t",$seq,"\t",$seqS[0],"\t",
$pos,"\t","19","\n";
	}
    #else{print
"$idprot_uni\n";}
	}
close(ID);
```

oxidacion.py

```
def buscarElemento(lista,
elemento):
  for i in range(0,len(lista)):
    if(lista[i] == elemento):
      return i

def escribe(cadena):
  if cadena.count('(ox)')>1:
    listaox=list(cadena)

posi=buscarElemento(lista
ox,'(')
    res=listaox[posi-1]
    del listaox[posi:posi+4]

    oxstr="".join(listaox)

seqox=oxstr.replace('(ox)',''
)
    sequence=pt[lista[1]]

posicion=sequence.find(se
qox)
    posf=posicion+posi
    post=str(posf)
    outfile.write
("\t"+lista[1]+"\t"+lista[2]+
"\t"+res+"\t"+post+"\t"+lis
ta[8])

    escribe(oxstr)
  else:
    listaox=list(cadena)

posi=buscarElemento(lista
ox,'(')
    res=listaox[posi-1]
    del listaox[posi:posi+4]

    oxstr="".join(listaox)

seqox=oxstr.replace('(ox)',''
)
    sequence=pt[lista[1]]

posicion=sequence.find(se
qox)
    posf=posicion+posi
    post=str(posf)
    outfile.write
("\t"+lista[1]+"\t"+lista[2]+
"\t"+res+"\t"+post+"\t"+lis
ta[8])

infile=open("C:\Users\Edua
rdo\Documents\\BaseDeD
atoswBio\\oxi2.txt","r")
infile2=open("C:\Users\Edu
ardo\Documents\\BaseDe
DatoswBio\\Buenos\\proto
.txt","r")
outfile=open("C:\Users\Ed
uardo\Documents\\BaseDe
```

```
DatoswBio\\oxidation2.txt"
,"w")
outfile2=open("C:\Users\E
duardo\Documents\\Base
DeDatoswBio\\oxidaciones
totales.txt","w")

pt={}

for line in infile2:
   listadicc=line.split('\t')

pt[listadicc[1]]=listadicc[2]

for line in infile:
   lista=line.split('\t')
   oxis=lista[3]


   if("(ox)" in oxis):


oxis=oxis.replace('(ph)','')

oxis=oxis.replace('(ac)','')
      oxid=oxis.replace('_','')

outfile2.write(oxid+"\n")
      escribe(oxid)
      #if oxis.count('(ox)')>1:
      #       listaox=list(oxid)
      #
posi=buscarElemento(lista
ox,'(')
      #       res=listaox[posi-
1]
      #       del
listaox[posi:posi+4]

      #       print
"".join(listaox)
      #       print posi-1
            #print
lista[1]+"\t"+str(posi)+"\t"+
listaox[posi-1]

#posif=int(lista[6])+int(posi
)
            #posifs=str(posif)
      #outfile.write
("\t"+lista[1]+"\t"+lista[2]+
"\t"+listaox[posi-
1]+"\t"+posifs+"\t"+lista[9]
)
      #outfile.write
("\t"+lista[1]+"\t"+lista[2]+
"\t"+oxid+"\t"+lista[9])

      #print posi
infile.close()
outfile.close()
outfile2.close()
```

verificarPDB.pl

```
#!/usr/bin/perl -w
#Para verificar si un pdb
puede caracterizar
# mas de una proteina
my %pdbs;
my @comp;
my @comp_abc;
my @pordes;
my @ihave;
my $time;
my $max_large=0;

my $a=0;
open(IN,"pdb_orf.csv") ||
die;
while(<IN>) {
  chomp;
  my($prot_id,$pdbid) =
split /\t/, $_;
  $prot_id=~ s/\s/./g;
  #creo una hash para
saber si existen
  $valor=length($prot_id);
  #print $valor."\n";

  if (length($prot_id)==7){
  $pdbs{$pdbid}.="$prot_id
";
            }
  else{
            if
(length($prot_id)==9){

$pdbs{$pdbid}.="$prot_id
";}
            else{
              print ("no estoy
en la base: $prot_id con
$pdbid con $valor\n");
                        }
            }
if ((grep $_ eq
$pdbid,@comp)==0){
            push
@comp,$pdbid;
            }
  #print
$prot_id."\t".$pdbid."\n";
}
foreach my $k (keys
%pdbs) {
  #print "$k\t$pdbs{$k}\n";
  $a=$a+1;
  $var=$pdbs{$k};
  $large=length($var);

#metodo de la burbuja
para saber que pdb tiene
mas proteinas
#caraterizadas

if($max_large < $large){
  $max_large=$large;
```

```
 $llave= $k;
}
 #print $large."\n";
 if($large>=10){  #print
"$k\t$pdbs{$k}\n";
        #$time=$time+1
        }
        }

#print $a."\n";
#$size=keys %pdbs;
#print $size."\n";
#print $time,"\n";
print "soy el pdb mas
grande $llave con
$max_large\n";
close(IN);

#para saber si se hace una
tabla auxiliar ?
open(EI,"listapdbac.txt") ||
die;
while(<EI>){
        chomp;
        my
($pdb_actual,undef)=split
/\./,$_;
        #print
$pdb_actual,"\n";
        push
@comp_abc,$pdb_actual;
        #if ((grep $_ eq
$pdb_actual,@comp)==0){
print "no aparezco:
$pdb_actual\n";}else{ print
"aparezco\n";}
}
close(EI);

foreach my $element
(@comp) {
        #print
$element,"\n";
        if ((grep $_ eq
$element,@comp_abc)==0
){
        #print "no
aparezco: $element\n";
        #system("python
pdb_download.py
$element");
        push
@pordes,$element;
        #system("mv
$element.pdb  PDBEstruct/
");
        }
}

#foreach my $ele
(@pordes){      print
$ele,"\n";}
open(EF,"listanueva.txt")
|| die;
while (<EF>){
        chomp;
    my
($pdb_actual2,undef)=split
/\./,$_;
        #$pdb_actual2=
tr/[a-z]/[A-
Z],$pdb_actual2);
        $pdb_actual2=
uc($pdb_actual2);
        push
@ihave,$pdb_actual2;
        #print
"$pdb_actual2\n";
}

close(EF);

foreach my
$kelem(@pordes){
 if ((grep $_ eq
$kelem,@ihave)==0){
                #print
"$kelem\n";
}}
```

pdb_download.py

```
import os, sys, ftplib, shutil,
gzip

HOSTNAME="ftp.wwpdb.o
rg"
DIRECTORY="/pub/pdb/dat
a/structures/all/pdb/"
PREFIX="pdb"
SUFFIX=".ent.gz"

array_pdbs=[];

def
unZip(some_file,some_out
put):
   """
   Unzip some_file using
the gzip library and write to
some_output.
   """

   f =
gzip.open(some_file,'r')
   g =
open(some_output,'w')
   g.writelines(f.readlines())
   f.close()
   g.close()

   os.remove(some_file)

def
pdbDownload(file_list,host
name=HOSTNAME,director
y=DIRECTORY,prefix=PREFI
X,
```

```
        suffix=SUFFIX):
    """
    Download all pdb files in
file_list and unzip them.
    """

    success = True

    # Log into server
    print "Connecting..."
    ftp = ftplib.FTP()
    ftp.connect(hostname)
    ftp.login()

    # Remove .pdb
extensions from file_list
    for file_index, file in
enumerate(file_list):
        try:
            file_list[file_index] =
file[:file.index(".pdb")]
        except ValueError:
            pass

    # Download all files in
file_list
    to_get = ["%s/%s%s%s"
% (directory,prefix,f,suffix)
for f in file_list]
    to_write = ["%s%s" %
(f,suffix) for f in file_list]
    for i in
range(len(to_get)):
        try:
            ftp.retrbinary("RETR
%s" %
to_get[i],open(to_write[i],"
wb").write)
            final_name =
"%s.pdb" %
to_write[i][:to_write[i].ind
ex(".")]
unZip(to_write[i],final_nam
e)
            print "%s retrieved
successfully." % final_name
        except
ftplib.error_perm:

os.remove(to_write[i])
            print "ERROR!  %s
could not be retrieved!" %
file_list[i]

array_pdbs.append(file_list
[i])
                print array_pdbs
            success = False

    # Log out
    ftp.quit()

    if success:
        return True
    else:
        return False


def main():
    """
    If the function is called
from the command line.
    """

    try:
        file_input =
sys.argv[1:]
    except IndexError:
        print __usage__
        sys.exit()

    pdb_list = []
    for arg in file_input:
        if os.path.isfile(arg)
and arg[-4:] != ".pdb":
            # Read in input file
            f = open(arg,"r")
            tmp_list =
f.readlines()
            f.close()

            # Strip comments
and blank lines, recombine
file, then split on all
            # white space
            tmp_list = [p for p in
tmp_list if p[0] != "#" and
p.strip() != ""]
            tmp_list =
"".join(tmp_list)
            tmp_list =
tmp_list.split()
            tmp_list = [p.lower()
for p in tmp_list]

pdb_list.extend(tmp_list)

        else:

            # Lower case,
remove .pdb if appended
            pdb_id = arg.lower()
            if pdb_id[-4:] ==
".pdb":
                pdb_id = pdb_id[:-
4]


pdb_list.append(pdb_id)

    # Download pdbs
    pdbDownload(pdb_list)

if __name__ ==
"__main__":
    main()
```

changename.pl

```
#!/usr/bin/perl -w
#Para cambiar nombre de
pdbs

open(IN,"listaTotal.txt") ||
die;
while(<IN>) {
	my
($directory,undef)=split
/\t/,$_;
	my ($pdb,$ext)=
split /\./,$directory;
	$nuevopdb =
uc($pdb);
	print
$nuevopdb,"\n";
	system("mv
$pdb.pdb
$nuevopdb.pdb")
}
close(IN);
```

tabla.pl

```
#!/usr/bin/perl
#creando tabla para pdbs
#primero: obtenemos la
lista de todos los pdb
descargados
my @comp;
my %pdbs;
my @descr;
my @proteoma;
my %alias;
my @request;
my %proteinas;
my %files;
my @comp2;
my @borradores;
$moleculas=0;

open(IN,"Leo.csv") || die;
while(<IN>) {
	chomp;

my($idprot,$standard_nam
e,$aliases)=split /,/,$_;
	$aliases=~ s/"//g;

$alias{$idprot}="$standard
_name $aliases";
}
close(IN);

open(ES,"proteinas.csv") ||
die;
while(<ES>) {
		chomp;
		my($idprot2,undef)
=split /\s/,$_;
		#print
$idprot2,"\n";
		if ((grep $_ eq
$idprot2,@proteoma)==0)
		{
			push
@proteoma,$idprot2;
		}
}
close(ES);

open(ED,'listaTotal.txt')||
die;
while(<ED>){
		chomp;
		my($pdbfile,$ext)=
split /\./,$_;
		#print
$pdbfile."\n";
		if ((grep $_ eq
$pdbfile,@descr)==0)
		{
			push
@descr,$pdbfile;
		}
}
close(ED);

#vamos al archivo original
para crear una hash y un
array auxiliar
#vamos a la proteinas en la
base para descartar las que
no estan
#y las que si estan
utilizarlas para hacer una
tabla de replicacion
open(IN,"pdb_orf.csv") ||
die;
while(<IN>) {
		chomp;
		my($prot_id,$pdbi
d) = split /\t/, $_;
		if ((grep $_ eq
$prot_id,@proteoma)==1){

		$pdbs{$pdbid}.="$
prot_id ";
```

```
                if ((grep $_
eq $pdbid,@comp)==0)
                {

        push
@comp,$pdbid;
                }
        }else
        {
                foreach my
$kk (keys %alias){
                        my
@array= split
/\s/,$alias{$kk};

        if ((grep $_ eq
$prot_id,@array)==1){

                #print "si
existo: $prot_id\n";

                if ((grep $_
eq
$prot_id,@resquest)==0){


        push
@resquest,$prot_id;

                }

        }
                        }
                if ((grep $_
eq
$prot_id,@resquest)==0){

        #print "no estoy en
la base: $prot_id con
$pdbid.pdb\n";
                        #if
($prot_idant eq $prot_id){

        $pdbs{$pdbid}=~
s/$prot_id//g;
                                #}

#$prot_idant=$prot_id;
                        }
        }
}

close(IN);

#para verificar contenido
#foreach $key(@comp){
        print $key,"\n";}
#foreach $key(@descr){
        print $key,"\n";}
#foreach my $k (keys
%pdbs) {  print
"$k\t$pdbs{$k}\n"; }
#$size=keys %pdbs; print
$size;

#una vez tenida la lista
limpia en %pdbs
mandamos aquellos que
tienen mas de una
#caraterizacion y que
ademas tengamos

foreach my $k (keys
%pdbs){
        $var=$pdbs{$k};
        $large=length($var)
;
        #print "$large\n";
        if($large>11){
                if((grep $_
eq $k,@descr)==1){

        #print
"$k\t$pdbs{$k}\n";
                }
                #else{
        system("python
pdb_download.py $k")}
                else{

        #print "de plano no
estoy: $k\n";

        delete($pdbs{$k});
                }
        $nuevaskeys= split
/\s/,$pdbs{$k};
        foreach $jj
(@nuevaskeys){

        $proteinas{$jj}.="$
k ";
        }
        }else{
                if((grep $_
eq $k,@descr)==1){

        #print
"$k\t$pdbs{$k}\n";

        $proteinas{$pdbs{$
k}}.="$k ";
                }else{

        #print
"$k\t$pdbs{$k}\n";

        #print "no se pudo
descargar: $k\n";

        #system("python
pdb_download.py $k")

        delete($pdbs{$k});
                }

        }

}
```

```
#creamos una hash de
tamaños con pdbs para
eliminar la
#redudancia de pdbs con
proteinas asociadas
my $pmid_find='';
open(IN,"weights.txt") ||
die;
while(<IN>) {
	chomp;
	my ($privileges,
$num,$user,$group,$weigh
t,$mounth,$day,$hour,$file
)= split /\s+/,$_;
	#print $file." con
".$weight."\n";
	#peso dado en
bytes
	#print $file."\n";
	open(EN,"$file");
	while(<EN>){
		chomp;
		#print
"entre\n";

	my($cadena,undef)
= split /\t/,$_;
		#print
$cadena."\n";
		if
($cadena=~ " MOLECULE:
"){

	$moleculas=$mole
culas+1;
		}

	if($cadena=~
/^JRNL/){
		my
($word,$ref,$data,undef)=
split /\s+/,$cadena;

	if($ref=~ /^PMID/){

		$pmid_find=$data;

		#print "$pmid con
$file\n";

		#my
($pdb_des,$ext)= split
/\./,$file;

		#if ($pmid eq ''){
				#
		$files{$pdb_des.".$
weight"."-0"}=$moleculas;

		#}else{
				#
		$files{$pdb_des.".$
weight"."-
$pmid"}=$moleculas;
				#}
				}
			}

		}
		#print $file."\n";
		close(EN);
		my
($pdb_des,$ext)= split
/\./,$file;
		if ($pmid_find eq
''){

		$files{$pdb_des.".$
weight"."-0"}=$moleculas;
		}else{

		$files{$pdb_des.".$
weight"."-
$pmid_find"}=$moleculas;
		}

		$moleculas=0;
		$pmid_find='';
}
close(IN);
$flag=0;
$flag2=0;
foreach my $kkk (keys
%proteinas){
		#print
"$kkk\t$proteinas{$kkk}\n"
;
		$var=$proteinas{$k
kk};
		$large=length($var)
;
		#print "$large\n";
		if($large>5){
			my
@elements= split
/\s+/,$proteinas{$kkk};
			#foreach $g
(@elements){print "$g con
$kkk\n";			}
			#metodo
de la burbuja aplicado a
numero de moleculas
			foreach $g
(@elements){

		#obtenemos del
hash el numero de
moleculas

		foreach my $gg
(keys %files){

		if ($gg =~ /^$g/){


		$number2=$files{$
gg};


		if($flag==0){
```

```
	$number=$number
2;

	$flag=1;
		}
		if
($number2<=$number){

	#segundo
criteriode seperacion el
peso del archivo
			my
($pdbwithw,$ppmidn)=
split /\./,$gg;

	my($peso2,$pmidn
)= split /-/,$ppmidn;

	if($flag2==0){

	$peso=$peso2;

	$flag2=1;

	#$gg_ant=$g;
			}

			if
($peso2>=$peso){

	$number=$number
2;
		$peso=$peso2;

		$minimo=$g;

		#pop
@elements,$g_ant;

		#$g_ant=$g;
				}
			}
		}
				}
			}

			#print
"$minimo de $kkk\n" ;
			$flag=0;
			$flag2=0;

		$proteinas{$kkk}=$
minimo;
			#print
"$kkk\t$proteinas{$kkk}\n"
;
		}else{
			#print
"$kkk\t$proteinas{$kkk}\n"
;
		}

}
#foreach my $ii(keys
%files){ print
"$ii\t$files{$ii}\n";}

#foreach my $ii(keys
%proteinas){ print
"$ii\t$proteinas{$ii}\n";}

#resultados de curación
manual, proteinas que no
tienen pdbs útiles
open(IN,"curacion.txt") ||
die;
while(<IN>) {
		chomp;
		my
($badfile,undef)=split
/\t/,$_;
		if ((grep $_ eq
$badfile,@borradores)==0)
{
			push
@borradores,$badfile;
		}

}
close(IN);

#BORRAR PDBS
#foreach
$key(@borradores){
		print $key,"\n";}
foreach my $llave (keys
%proteinas){
		$llave=~ s/\s+//g;
		$llave2=$llave;
		if ((grep $_ eq
$llave2,@borradores)==1){
			delete
($proteinas{$llave." "});
			#if ((grep
$_ eq
$llave2,@borradores2)==0)
{
			#
		push
@borradores2,$llave2;
			#}
```

```
                #print
"entre\n";
        }
}
#foreach
$key(@borradores2){
        print $key,"\n";}
#foreach my $ii(keys
%proteinas){ print
"$ii\t$proteinas{$ii}\n";}

#HACER CAMBIOS por pdbs
mejores
my @compb;
my $cont=0;

open(EI,"cambios.txt")||
die;
while(<EI>){
        chomp;
        my
($asoc,$nuevopdb,$pmid,u
ndef)=split /\t/,$_;
        push
@compb,$asoc;
        foreach my $qq
(keys %proteinas){
                $qq=~
s/\s+//g;
                if($asoc eq
$qq){

        $proteinas{$qq."
"}=$nuevopdb;

        $cont=$cont+1;

        foreach my $qj
(keys %files ){

        my
($idpdbfile,$others)= split
/\./,$qj;

        my
($trash,$pmidcomp)= split
/-/,$others;

        if
(($proteinas{$qq." "}) eq
$idpdbfile){

                if ($pmid
eq $pmidcomp){


        #print "es
correcto\n";

                        pop
@compb;

                }else{print
"no es igual $pmid y
$pmidcomp\n"};

        }
                        }
                }
        }
}
close(EI);


# en curación manual

#$size=keys %pdbs; print
$size."\n";
#$size=keys %proteinas;
print $size."\n";
#foreach $key
(@compb){print
$key,"\n";}
#print $cont."\n";

foreach my $ii(keys
%proteinas){

        foreach my $jj
(keys %files ){

        $proteinas{$ii}=~
s/\s+//g;
                if ($jj =~
/^$proteinas{$ii}/){
                        my
($pdbid,$weightpmid)=
split /\./,$jj;
                        my
($weight,$pmid)=split /-
/,$weightpmid;

        print
"$ii\t$proteinas{$ii}\t$pmi
d\n";
                }
        }
}
```

get_store_pmid.pl

```
#!/usr/bin/perl -w

use XML::LibXML;
use LWP::Simple;
use SQL::Abstract;

my $pmid = $ARGV[0];
#my $pmid = '21774816';

my $pmid_xml =
get("http://eutils.ncbi.nlm.
nih.gov/entrez/eutils/efetc
h.fcgi?db=pubmed&id=$p
mid\[UID\]&retmode=xml"
);
#my $pmid_xml =
get("http://www.ncbi.nlm.
nih.gov/pubmed/$pmid?re
port=xml&format=text");
#print $pmid_xml . "\n";

my %pm_en_prensa = ();

$pm_en_prensa{pmid} =
$ARGV[0]; #OK
$pm_en_prensa{id_investi
gador} = $ARGV[1];
$send2db = $ARGV[2];

my $parser=XML::LibXML-
>new();
my $tree = $parser-
>parse_string($pmid_xml);
my $pubmed = $tree-
>getDocumentElement;

$pm_en_prensa{issn} =
$pubmed-
>findnodes('PubmedArticle
/MedlineCitation/Article/Jo
urnal/ISSN');
$pm_en_prensa{volume} =
$pubmed-
>findnodes('PubmedArticle
/MedlineCitation/Article/Jo
urnal/JournalIssue/Volume
');
$pm_en_prensa{issue} =
$pubmed-
>findnodes('PubmedArticle
/MedlineCitation/Article/Jo
urnal/JournalIssue/Issue');
$pm_en_prensa{year} =
$pubmed-
>findnodes('PubmedArticle
/MedlineCitation/Article/Jo
urnal/JournalIssue/PubDat
e/Year');
$pm_en_prensa{journal_tit
le} = $pubmed-
>findnodes('PubmedArticle
/MedlineCitation/Article/Jo
urnal/Title');
$pm_en_prensa{journal_tit
le_abbr} = $pubmed-
>findnodes('PubmedArticle
/MedlineCitation/Article/Jo
urnal/ISOAbbreviation');
$pm_en_prensa{articulo_ti
tle} = $pubmed-
>findnodes('PubmedArticle
/MedlineCitation/Article/A
rticleTitle');
$pm_en_prensa{paginas} =
$pubmed-
>findnodes('PubmedArticle
/MedlineCitation/Article/P
agination/MedlinePgn');

my @abstract_all =
$pubmed-
>findnodes('PubmedArticle
/MedlineCitation/Article/A
bstract/AbstractText');
my $abstract_text;
foreach my $abstract_label
(@abstract_all) {
  $abstract_text .=
$abstract_label-
>textContent . " ";
}
$pm_en_prensa{abstract}
= $abstract_text;

my @autor_all = $pubmed-
>findnodes('PubmedArticle
/MedlineCitation/Article/A
uthorList/Author');
my $autor_list;

foreach my $autor
(@autor_all) {
  my $last = $autor-
>findnodes('LastName');
  my $initials = $autor-
>findnodes('Initials');
  $autor_list .= "$last\,
$initials\; ";
}
$autor_list =~ s/\; $//;

#$pm_en_prensa{autor_lis
t} = $autor_list;

my $status = $pubmed-
>findnodes('PubmedArticle
/PubmedData/PublicationS
tatus');

$pm_en_prensa{pub_type}
= $status->string_value();

if($status->string_value()
eq "ppublish") {
  $nota = 'Publicado en
papel';
}
```

```
elsif($status-
>string_value() eq
"epublish") {
  $nota = 'Publicado sólo
electrónicamente';
}
elsif($status-
>string_value() eq
'aheadofprint') {
  $nota = 'No ha sido
publicado en papel';
}

my @ids = $pubmed-
>findnodes('PubmedArticle
/PubmedData/ArticleIdList
/ArticleId');
foreach my $id (@ids) {
  my $attribute = $id-
>getAttribute('IdType');
  if ($attribute eq 'doi') {
    $pm_en_prensa{doi} =
$id->string_value()
#     print "DOI " . $id-
>string_value() . "\n";
  }
}

if($pm_en_prensa{journal_
title_abbr} =~ /\w+/) {   #Si
existe Journal abbreviation
  if($nota =~ /No ha sido
publicado en papel/) {
    $pm_en_prensa{cita} =
"$autor_list
\($pm_en_prensa{year}\)
$pm_en_prensa{articulo_ti
tle}
$pm_en_prensa{journal_tit
le_abbr}\. ";
  }
  else {
    $pm_en_prensa{cita} =
"$autor_list
\($pm_en_prensa{year}\)
$pm_en_prensa{articulo_ti
tle}
$pm_en_prensa{journal_tit
le_abbr}\.
$pm_en_prensa{volume}\:
\($pm_en_prensa{issue}\)
$pm_en_prensa{paginas}\.
";
  }
}
else {    #No existe Journal
abbr, ponemos el titulo
largo
  if($nota =~ /No ha sido
publicado en papel/) {
    $pm_en_prensa{cita} =
"$autor_list
\($pm_en_prensa{year}\)
$pm_en_prensa{articulo_ti
tle}
$pm_en_prensa{journal_tit
le}\. ";
  }
  else {
    $pm_en_prensa{cita} =
"$autor_list
\($pm_en_prensa{year}\)
$pm_en_prensa{articulo_ti
tle}
$pm_en_prensa{journal_tit
le}\.
$pm_en_prensa{volume}\:
\($pm_en_prensa{issue}\)
$pm_en_prensa{paginas}\.
";
  }
}

# ESCRIBE EN LA BD

if ( $send2db eq '1' ){
        my $dbh =
conectadb();
        my
$articulos_pubmed =
SQL::Abstract->new;
        my($stmt, @bind)
= $articulos_pubmed-
>insert('articulos_pubmed',
\%pm_en_prensa);

        $sth = $dbh-
>prepare($stmt);
        $sth-
>execute(@bind);
}else{

        my $sin =
$pm_en_prensa{abstract};
        my $cit =
$pm_en_prensa{cita};

        #print  'ABSTRACT
'. $sin."\n";
     print   $cit."\n";
}

# UTIL

sub conectadb {
  use DBI;
  use strict;
  use vars qw($dbhost
$dbuser $dbpassword
$dsn);

  my $dbhost = 'localhost';
  my $dbuser = 'root';
  my $dbpassword =
'guasita';
  my $dsn =
"DBI:mysql:database=infor
me_2014;host=$dbhost;";
  my $dbh = DBI-
>connect($dsn, $dbuser,
```

```
$dbpassword,
                    {
                     RaiseError
=> 1,

AutoCommit => 1
                    }
                   );

          $dbh->do(qq{SET
NAMES 'utf8';});

   return undef unless
(defined $dbh);
   return $dbh
}
```

yaam_ini.php

```
<!doctype html>
<html><!-- InstanceBegin template="/Templates/Index.dwt" codeOutsideHTMLIsLocked="false" --
>
<head>
<meta charset="utf-8">
<title>YAAM Yeast Amino Acid Modifications</title>
<link href="CSS/Index.css" rel="stylesheet" type="text/css">
<!-- <link rel="stylesheet" href="chosen/docsupport/style.css">
<link rel="stylesheet" href="chosen/docsupport/prism.css"> -->
<link rel="stylesheet" href="chosen/chosen.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script src="chosen/chosen.jquery.js" type="text/javascript"></script>
 <script src="chosen/docsupport/prism.js" type="text/javascript" charset="utf-8"></script>
 <script type="text/javascript">
   var config = {
     '.chosen-select'          : {},
     '.chosen-select-deselect'  : {allow_single_deselect:true},
     '.chosen-select-no-single' : {disable_search_threshold:10},
     '.chosen-select-no-results': {no_results_text:'Oops, nothing found!'},
     '.chosen-select-width'     : {width:"95%"}
   }
   for (var selector in config) {
     $(selector).chosen(config[selector]);
   }
 </script>
<link rel="stylesheet" type="text/css" media="screen"
href="//cdn.datatables.net/1.10.5/css/jquery.dataTables.min.css" />
<script src="//cdn.datatables.net/1.10.5/js/jquery.dataTables.min.js"
type="text/javascript"></script>
<link rel="stylesheet" type="text/css" media="screen"
href="TableTools/dataTables.tableTools.css" />
<script src="TableTools/dataTables.tableTools.min.js" type="text/javascript"></script>

</head>

<body>

<script>
```

```
$(document).ready(function() {
   var sameres;
   var modif;
   $('#mainres').hide();
   $("#busca").submit(function() {

            $('#secres').hide();
            $('#mainres').show();
            $('#mainres').dataTable( {
                  sDom:'T<"clear">lfrtip',
                  tableTools:{
                        "aButtons": ["copy","csv","xls"],
                        "sSwfPath": "TableTools/copy_csv_xls.swf"
                  },
                  destroy:true,
               "aoColumnDefs": [
                 { "sClass": "dt-center", "aTargets": [ '_all' ] }
                ],
               "ajax": '../Proteinas/modificaciones.php?modificacion='+modif,
               "columns": [
                  { data: "systematic_name" ,
                    "fnCreatedCell": function (nTd, sData, oData, iRow, iCol) {
                  $(nTd).html("<a href=detalle_final.php?orf="+sData+">"+sData+"</a>");
                     }
                  },
                  { data: "common_name" },
                  { data: "residue" },
                  { data: "position" },
                  { data: "source" }
                ]
            } );

      return false;

   });
});
</script>


<script>
$(document).ready(function() {

   var lobject = {};
   var mobject = {};
```

```
$('#secres').hide();
$(".chosen").chosen({
  width: "95%",
  include_group_label_in_selected: true,
});

$("#doit").click(function () {

   var loc = Array();
   var mod = Array();
   loc = $("#location").val();
   mod = $("#modification").val();


   for(i in loc){
         lobject[i]=loc[i];
   }
   for(j in mod){
         mobject[j]=mod[j];
   }
   lobject = JSON.stringify(loc);
   mobject = JSON.stringify(mod);


   $('#busca2').submit();

   });

   $('#searching').submit(function(){
   var bus = $('#buscar').val();
    $('#mainres').hide();
   $('#Caja_texto2').hide();
   $('#secres').show();
         $('#secres').dataTable( {
               sDom:'T<"clear">lfrtip',
               tableTools:{
                     "aButtons": ["copy","csv","xls"],
                     "sSwfPath": "TableTools/copy_csv_xls.swf"
               },
               destroy:true,
            "aoColumnDefs": [
              { "sClass": "dt-center", "aTargets": [ '_all' ] }
             ],
            "ajax": '../Proteinas/search.php?busqueda4='+bus,
```

```
            "columns": [
                { data: "systematic_name" , "fnCreatedCell": function (nTd, sData, oData, iRow, iCol)
{
                $(nTd).html("<a href=detalle_final.php?orf="+sData+">"+sData+"</a>");
                   }
                },
                { data: "common_name" },
                { data: "location" }
             ]
          } );

    return false;


    });

    $("#busca2").submit(function(){

    var resan;
    if(document.busca2.residue[0].checked)
        resan="yes";
    else
        resan="no";

     $('#mainres').hide();
    $('#Caja_texto2').hide();
    $('#secres').show();
        $('#secres').dataTable( {
            sDom:'T<"clear">lfrtip',
            tableTools:{
                "aButtons": ["copy","csv","xls"],
                "sSwfPath": "TableTools/copy_csv_xls.swf"
            },
            destroy:true,
          "aoColumnDefs": [
            { "sClass": "dt-center", "aTargets": [ '_all' ] }
          ],
          "ajax":
'../Proteinas/Localization.php?modification='+mobject+'&location='+lobject+'&residue='+resan,
          "columns": [
            { data: "systematic_name" ,
              "fnCreatedCell": function (nTd, sData, oData, iRow, iCol) {
            $(nTd).html("<a href=detalle_final.php?orf="+sData+">"+sData+"</a>");
               }
```

```
                },
                { data: "common_name" },
                { data: "location" }
            ]
        } );

    return false;

  });//eldelclick
});
</script>

<div id="Contenedor">
<header onclick="location.href='/YAAM/about_yaam.php';" id="header">
  <div onclick="location.href='/YAAM/about_yaam.php';" id="ifc"></div>

  <div id="menu">
    <ul>
      <li><a href="/YAAM/about_yaam.php">About YAAM</a></li>
      <li><a href="/YAAM/add_yaam.php">Add a YAAM</a></li>
      <li><a href="/YAAM/yaam_ini.php">Home</a></li>
 </ul>
  </div>
</header>
<div id="search">
    <form id="searching">
        General Search:<input type="text" name="buscar" id="buscar">
    </form>
</div>
<div id="Cuerpo">
    <div id="Columna">
    <br>
    <form id="busca2" name="busca2">
            <br>
            <br>
            YAAM search
            <br>
            <br>
        <select multiple id="modification" class="chosen" data-placeholder="Select
modifications" name="categories2[]" style="width: 300px" >
            <option>Acetylation</option>
            <option>ActiveSite</option>
            <option>Ca</option>
            <option>Disulfide</option>
```

```
            <option>Glycosylation</option>
            <option>Lipidation</option>
            <option>Metal</option>
            <option>Methylation</option>
            <option>Nitration</option>
            <option>Nterminal</option>
            <option>Oxidation</option>
            <option>Phosphorylation</option>
            <option>Succinylation</option>
            <option>Ubiquitination</option>
          </select>
           <br><br>
          <select multiple id="location" class="chosen" data-placeholder="Select localizations"
name="categories[]" style="width: 300px" >
           <option></option>
            <option>Golgi apparatus</option>
            <option>Cytoplasm</option>
            <option>Mitochondrion</option>
            <option>Peroxisome</option>
            <option>Endoplasmic reticulum</option>
            <option>Vacuole</option>
            <option>Vesicle</option>
            <option>Cell wall</option>
            <option>Prospore membrane</option>
            <option>Plasma membrane</option>
            <option>Membrane protein</option>
            <option>Putative membrane protein</option>
            <option>Cellular bud</option>
            <option>Site of polarized growth</option>
            <option>Nucleus</option>
            <option>Nucleolus</option>
          </select>
           <br>
           <br>
           <input id="doit" type="button" value=" Search ">
           <br>
           <br>
           Same Residue
           <br>
           <br>
           <input id="resans" type="radio" name="residue" value="Yes">Yes
<input id="resans" type="radio" name="residue" value="No" checked>No
     </form>
              <br>
```

```
                <br>
  </div>

  <div id="Visualizar">
    <div id="Caja_texto2">
      Select one or more Modification and/or Location and click the button "Search" to display the
results.
      <br><br><br><img src="Imagenes/bg_yaam.jpg">
    </div>
<!-- InstanceBeginEditable name="EditRegion1" -->
    <table id="mainres" class="display" cellspacing="0" width="100%">
      <thead>
        <tr>
          <th>Systematic Name</th>
          <th>Common Name</th>
          <th>Residue</th>
          <th>Position</th>
          <th>pmid</th>
        </tr>
      </thead>

      <tfoot>
        <tr>
          <th>Systematic Name</th>
          <th>Common Name</th>
          <th>Residue</th>
          <th>Position</th>
          <th>pmid</th>
        </tr>
      </tfoot>

    </table>

    <table id="secres" class="display" cellspacing="0" width="100%">
      <thead>
        <tr>
          <th>Systematic Name</th>
          <th>Common Name</th>
          <th>Location</th>
        </tr>
      </thead>

      <tfoot>
        <tr>
```

```
            <th>Systematic Name</th>
            <th>Common Name</th>
            <th>Location</th>
          </tr>
        </tfoot>

    </table>
     <div id="resultados">

       </div>


<!-- InstanceEndEditable -->
</div>
</div>
<?php include("footer_yaam.php") ?>
</div>
</body>
<!-- InstanceEnd --></html>
```

detalle_final.php

```
<?php

      $quest=utf8_decode($_GET['orf']);
      // $quest = utf8_decode('YBR009C');

      $con=mysqli_connect("localhost","user","password") or die("Failed to connect with
database!!!");
      mysqli_select_db($con,"proteinas");



      $query = "select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
      where TABLE_NAME='Acetylation' and TABLE_SCHEMA='proteinas') as tn join Acetylation as
ac join Source as src on
      ac.source_id=src.id join Equivalences as e on ac.proteina_idprot=e.proteina_idprot where
ac.proteina_idprot='$quest'
      or e.standard_name='$quest'
      union
```

```
select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
where TABLE_NAME='Phosphorylation' and TABLE_SCHEMA='proteinas') as tn join
Phosphorylation as fs join Source as src on
fs.source_id=src.id join Equivalences as e on fs.proteina_idprot=e.proteina_idprot where
fs.proteina_idprot='$quest'
or e.standard_name='$quest'
union
select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
where TABLE_NAME='Glycosylation' and TABLE_SCHEMA='proteinas') as tn join Glycosylation
as gl join Source as src on
gl.source_id=src.id join Equivalences as e on gl.proteina_idprot=e.proteina_idprot where
gl.proteina_idprot='$quest'
or e.standard_name='$quest'
union
select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
where TABLE_NAME='Lipidation' and TABLE_SCHEMA='proteinas') as tn join Lipidation as lp
join Source as src on
lp.source_id=src.id join Equivalences as e on lp.proteina_idprot=e.proteina_idprot where
lp.proteina_idprot='$quest'
or e.standard_name='$quest'
union
select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
where TABLE_NAME='Succinylation' and TABLE_SCHEMA='proteinas') as tn join Succinylation
as sc join Source as src on
sc.source_id=src.id join Equivalences as e on sc.proteina_idprot=e.proteina_idprot where
sc.proteina_idprot='$quest'
or e.standard_name='$quest'
union
select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
where TABLE_NAME='Ubiquitination' and TABLE_SCHEMA='proteinas') as tn join
Ubiquitination as ub join Source as src on
ub.source_id=src.id join Equivalences as e on ub.proteina_idprot=e.proteina_idprot where
ub.proteina_idprot='$quest'
or e.standard_name='$quest'
union
select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
where TABLE_NAME='Methylation' and TABLE_SCHEMA='proteinas') as tn join Methylation as
mt join Source as src on
```

```
mt.source_id=src.id join Equivalences as e on mt.proteina_idprot=e.proteina_idprot where
mt.proteina_idprot='$quest'
or e.standard_name='$quest'
union
select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
where TABLE_NAME='Oxidation' and TABLE_SCHEMA='proteinas') as tn join Oxidation as ox
join Source as src on
ox.source_id=src.id join Equivalences as e on ox.proteina_idprot=e.proteina_idprot where
ox.proteina_idprot='$quest'
or e.standard_name='$quest'
union
select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
where TABLE_NAME='Ca' and TABLE_SCHEMA='proteinas') as tn join Ca as cb join Source as
src on
cb.source_id=src.id join Equivalences as e on cb.proteina_idprot=e.proteina_idprot where
cb.proteina_idprot='$quest'
or e.standard_name='$quest'
union
select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
where TABLE_NAME='Metal' and TABLE_SCHEMA='proteinas') as tn join Metal as mb join
Source as src on
mb.source_id=src.id join Equivalences as e on mb.proteina_idprot=e.proteina_idprot where
mb.proteina_idprot='$quest'
or e.standard_name='$quest'
union
select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
where TABLE_NAME='Nterminal' and TABLE_SCHEMA='proteinas') as tn join Nterminal as at
join Source as src on
at.source_id=src.id join Equivalences as e on at.proteina_idprot=e.proteina_idprot where
at.proteina_idprot='$quest'
or e.standard_name='$quest'
union
select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
where TABLE_NAME='ActiveSite' and TABLE_SCHEMA='proteinas') as tn join ActiveSite as acs
join Source as src on
acs.source_id=src.id join Equivalences as e on acs.proteina_idprot=e.proteina_idprot where
acs.proteina_idprot='$quest'
or e.standard_name='$quest'
union
```

```
	select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
	where TABLE_NAME='Disulfide' and TABLE_SCHEMA='proteinas') as tn join Disulfide as db join
Source as src on
	db.source_id=src.id join Equivalences as e on db.proteina_idprot=e.proteina_idprot where
db.proteina_idprot='$quest'
	or e.standard_name='$quest' order by posicion;";

	$pdb = "select pdb_file from pdb where proteina_idprot='$quest'";

	$sth = mysqli_query($con,$query);

	$sthpdb = mysqli_query($con,$pdb);

	$rows = array();
	$modifications = array();
	$positions = array();
	$table = array();
	$table['cols']=array(
		array('label' => 'Residue','type' => 'string'),
		array('label' => 'Position', 'type' => 'number'),
		array('label' => 'PMID', 'type' => 'number'),
		array('label' => 'Modification', 'type' => 'string')
	);

	while($r = mysqli_fetch_assoc($sth)){
		$temp = array();
		if($r['pmid'])
			$temp = array('Residue' => (string)
utf8_encode($r['residuo']),'Position'=>(int)$r['posicion'],'PMID'=>(int)$r['pmid'],'Modification'=>(s
tring)utf8_encode($r['TABLE_NAME']));
		else
			$temp = array('Residue' =>
(string)utf8_encode($r['residuo']),'Position'=>(int)$r['posicion'],'PMID'=>(string)utf8_encode($r['u
niprot.org']),'Modification'=>(string)utf8_encode($r['TABLE_NAME']));

		array_push($rows,$temp);
	}

	if(mysqli_num_rows($sthpdb)){

		$resu = mysqli_fetch_array($sthpdb);
		$pdbname =$resu['pdb_file'];
	}else{
```

```
        $pdbname="";
    }

    for($s=0;$s<count($rows);$s++){
        if($s>0){
            if($rows[$s]['Position']===$rows[$s-1]['Position'] &&
$rows[$s]['Modification']===$rows[$s-1]['Modification']){
                $rows[$s]['PMID']=$rows[$s]['PMID'].",".$rows[$s-1]['PMID'];
                array_splice($rows,$s-1,1);
                $s-=1;
            }
        }
    }

    for($s=0;$s<count($rows);$s++){
        if($s>0){
            if($rows[$s]['Position']===$rows[$s-1]['Position'] &&
$rows[$s]['Modification']!==$rows[$s-1]['Modification']){
                $rows[$s]['Modification']=" ".$rows[$s]['Modification'].",".$rows[$s-
1]['Modification'];
$rows[$s]['PMID']=$rows[$s]['PMID'].",".$rows[$s-1]['PMID'];
                array_splice($rows,$s-1,1);
                $s-=1;
            }
        }
    }

    function startsWith($haystack, $needle){
        return $needle === "" || strrpos($haystack, $needle, -strlen($haystack)) !== FALSE;
    }

    if($pdbname!=""){
    $fp = fopen("../pdbs/$pdbname.pdb","r");

    while(!feof($fp)){
        $linea = fgets($fp);
        if(startsWith($linea,"ATOM")){
            $dp=(int) substr($linea,22,4);
            break;
        }

    }
    fclose($fp);
    }
```

```
    for($i=0; $i<count($rows); $i++){
        $pmids=explode(',',$rows[$i]['PMID']);
        $modifs=explode(',',$rows[$i]['Modification']);
        $rows[$i]['PMID']=implode(",",array_unique($pmids));
        $rows[$i]['Modification']=implode(",",array_unique($modifs));
        $modifications[] = implode(",",array_unique($modifs));
        if($pdbname==="2M88" || $pdbname==="5PGM" || $pdbname==="1EJB" ||
$pdbname==="3SBC" || $pdbname==="1D1Q"){
            $positions[] = $rows[$i]['Position']-1;
        }
        else if($pdbname==="1F1G" || $pdbname==="2K88"){
            $positions[] = $rows[$i]['Position']+1;
        }
        else if($pdbname==="1DVR"){
            $positions[] = $rows[$i]['Position']-2;
        }
        else if($pdbname==="2HV4"){
            $positions[] = $rows[$i]['Position']-6;
        }
        else if($pdbname==="1YAT"){
            $positions[] = $rows[$i]['Position']-7;
        }
        else{
            $positions[] = $rows[$i]['Position'];
        }
    }



    $table['rows'] = $rows;

    $jsonPos = json_encode($positions);
    $jsonMod = json_encode($modifications);
    $jsonPDB = json_encode($pdbname);


?>
<!doctype html>
<html><!-- InstanceBegin template="/Templates/Index.dwt" codeOutsideHTMLIsLocked="false" --
>
<head>
<meta charset="utf-8">
```

```
<title>YAAM Yeast Amino Acid Modifications</title>
<link href="CSS/Index.css" rel="stylesheet" type="text/css">
<link href="CSS/Tablas_detalle.css" rel="stylesheet" type="text/css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
<link rel="stylesheet" type="text/css" media="screen"
href="//cdn.datatables.net/1.10.5/css/jquery.dataTables.min.css" />

<link rel="stylesheet" href="../ChemDoodleWeb700/install/ChemDoodleWeb.css"
type="text/css">
<script type="text/javascript"
src="../ChemDoodleWeb700/src/ChemDoodleWeb_yaam.js"></script>
<script src="//cdn.datatables.net/1.10.5/js/jquery.dataTables.min.js"
type="text/javascript"></script>
<link rel="stylesheet" type="text/css" media="screen"
href="TableTools/dataTables.tableTools.css" />
<script src="TableTools/dataTables.tableTools.min.js" type="text/javascript"></script>
</head>

<body>

<script>

var orf = '<?php echo $quest;?>';
var colores = {
      "Acetylation"    : "#FBBA00", //amarillo 13
      "Phosphorylation": "#D00A10",  //rojo 9
      "Ca"             : "#66FFFF",  //aqua 7
      "Disulfide"      : "#BE2E46",   //amariillo 14
      "Glycosylation"  : "#95039F",  //morado 3
      "Lipidation"     : "#ADE415",  //limon 11
      "Metal"          : "#66B7EE",  //azul 6
      "Methylation"    : "#6456FD",  //morado 15
      "Oxidation"      : "#06974A",  // verde 12
      "Succinylation"  : "#F4667C",  //rosa 1
      "Ubiquitination" : "#9900EE",  //morado 4
      "Nterminal"      : "#ED22C8",  //rosa 2
      "Nitration"      : "#1502B8",  //azul 5
      "ActiveSite"     : "#515151",  // gris 10
      "Morethanone"    : "#775533"    // verde
}

function obj(){
   obj=new Object();
   this.add=function(key,value){
```

```
      obj[""+key+""]=value;
   }
   this.obj=obj
}

$(document).ready(function() {
 //   $('#modif').css('color',"#00FF00"),
    $('#mainres').dataTable( {
        sDom:'T<"clear">lfrtip',
        tableTools:{
          "aButtons": ["copy","csv","xls"],
          "sSwfPath": "TableTools/copy_csv_xls.swf"
        },
        destroy:true,
        "aoColumnDefs": [
        { "sClass": "dt-left", "aTargets": [ '_all' ] },
        ],
        ajax: '../Proteinas/search.php?busqueda3='+orf,
        "columns": [
{ data: "residue" },
                    { data: "position" },
                    { data: "modification" },
                    { data: "pmid" }
                ]
        });
    $('#main').dataTable( {
        "sDom": '<"top">rt<"bottom"rt><"clear">',
        "aoColumnDefs": [
        { "sClass": "dt-left", "aTargets": [ '_all' ] }
                ],
        ajax: '../Proteinas/search.php?busqueda1='+orf,
        "columns": [
                    { data: "systematic_name" },
                    { data: "standard_name" },
                    { data: "header" },
                    { data: "location" }
                ]
     });
});


$(document).ready(function() {
  var color_posicion = [];
  $.ajax({
```

```
    async: false,
    url: '../Proteinas/search.php?busqueda1='+orf,
    success: function(response) {
      var res = JSON.parse(response);
      var seq = res.data[0].sequence;


      rows = Math.floor(seq.length / 60);
      residuo = seq.length % 60;
      if(residuo >0) {
        rows++;
      }
      $.ajax({
        async: false,
        url: '../Proteinas/search.php?busqueda2='+orf,
        success: function(response2) {
          var res2 = JSON.parse(response2);
          var colour;
          var posicion;
          for (i in res2.data) {
            if(typeof res2.data[i].modification !== 'undefined' &&
(res2.data[i].modification.match("\,"))) {
              colour = colores["Morethanone"]; // More than one modifications
              posicion = res2.data[i].position;
            }
            else if(typeof res2.data[i].modification !== 'undefined') {
              colour = colores[res2.data[i].modification];
              posicion = res2.data[i].position;
            }
            color_posicion[posicion] = colour;
          }
        }
      });
     // console.log(color_posicion);
      var area_table = new String();
      var max;
      var min;
      var row1;
var col;
      for ( i=1; i<=rows; i++) {
        lista1 = [];
        max = (i*60) - 1;
        min;
        if( i == 1) {
```

```
          min = 0;
        }
        else {
          min = (i-1)*60;
        }
        for (var pos in color_posicion) {
           if (pos <= max && pos >= min) {
               lista1.push(pos);
              // console.log(lista1);
          //    alert(pos);
           }
        }
        row1 = seq.substring((i-1)*60,(i*60));
        //console.log(row1);
        for ( j=1; j<=6; j++) {
          var max2 = (j*10) - 1;
          var min2;
          if( j == 1) {
            min2 = 0;
          }
          else {
            min2 = (j-1)*10;
          }
          col = row1.substring((j-1)*10,(j*10));
         // console.log(min + ':' + max + ':' + i + ':' + j + ':' + min2  + ':' + max2) ;
          lista1_sorted = lista1.sort(function(a, b){return b-a});
          for( k=0; k<=lista1_sorted.length; k++) {
            if((lista1_sorted[k]-1) <= max2 + (i-1)*60 && (lista1_sorted[k]-1) >= min2 + (i-1)*60) {
               //console.log(lista1_sorted[k]-1);
               var carac;
               var insert;
               position = ((lista1[k]-1) - ((j-1)*10) - (i-1)*60);
               //console.log('POSITION ' + position);
               carac = col.charAt(position);
               insert =  '<span style="font-weight: bold; color:' + color_posicion[lista1[k]] + '">' +
carac + '</span>';
               col = col.substring(0,position) + insert + col.substring(position+1);
             }
           }
           if( typeof area_table !== 'undefined' || area_table !== '') {
             if( typeof col !== '' || typeof area_table !== 'undefined') {
               area_table += '<td>' + col + '</td>';
             }
           }
```

```
          }
          area_table = '<tr>' + area_table + '</tr>';
        }
        //console.log(area_table);
        $("#secuencia").append(area_table);
    }
    });
});

//$(document).ready(function() {
//   $('#modificaciones tr').each(function() {
//        alert($(this).html());
//    });
//});
</script>

<script>

      //var loc = <?php echo $jsonPDB; ?>;
    function muestra_pdb( w, h) {
      var ribbonTransformer = new ChemDoodle.TransformCanvas3D('ribbonTransformer', w, h);
      ribbonTransformer.specs.proteins_residueColor = 'none';
      ribbonTransformer.specs.compass_display = true;
      ribbonTransformer.specs.backgroundColor = '#444445';  //Option F1F1F21
      var pdb;
      var residuos;

      var posit = <?php echo $jsonPos; ?>;
      var modi = <?php echo $jsonMod; ?>;

      var loc =<?php echo $jsonPDB; ?>;
      if(loc==""){
            $('#contpdb').hide();
      //     $('#divpdb').hide();
      //     $('#cajatexto').hide();
      //     $('#optpdb').hide();
      }



      ChemDoodle.io.file.content('../pdbs/'+loc+'.pdb', function(fileContent){
        pdb = ChemDoodle.readPDB(fileContent,1,posit,modi);
        var residueSpecs = new ChemDoodle.structures.VisualSpecifications();
        residueSpecs.proteins_residueColor = 'modif';
```

```
        ribbonTransformer.specs.proteins_residueColor = residueSpecs.proteins_residueColor;
        ribbonTransformer.specs.atoms_display = false;
        ribbonTransformer.specs.atoms_font_size_2D = 10;
   //      ribbonTransformer.specs.atoms_displayLabels_3D = true;
        ribbonTransformer.specs.proteins_ribbonCartoonize = true;
        ribbonTransformer.specs.bonds_display = false;
        ribbonTransformer.loadMolecule(pdb);
        var c=0;
        var resi=[];
        for(var i = 0, ii=pdb.chains.length; i<ii; i++){
          for(var j = 0, jj=pdb.chains[i].length-2; j<jj; j++){
             c++;
            resi[c] =  pdb.chains[i][j+1].name;
          }
        }
       }

      );
    }

</script>


<div id="Contenedor">
<header onclick="location.href='/YAAM/about_yaam.php';" id="header">
  <div onclick="location.href='/YAAM/about_yaam.php';" id="ifc"></div>

  <div id="menu">
    <ul>
      <li><a href="/YAAM/index_YAAM.html">About YAAM</a></li>
      <li><a href="/YAAM/add_yaam.php">Add a YAAM</a></li>
      <li><a href="/YAAM/yaam_ini.php">Home</a></li>
    </ul>
  </div>
</header>

<div ="Cuerpo">
    <div >
      <table id="main" class="display" cellspacing="0" width="100%">
      <thead>
        <tr>
          <th>ORF</th>
```

```
            <th>Common name</th>
            <th>Annotations</th>
            <th>Location</th>
          </tr>
         </thead>
        </table>
        <br><br>
       <div id='contpdb'>
        <div id='divpdb' style='margin-left:150px; margin-right:30px; height: 500px; width: 600px;
float: left; margin-bottom:20px'>
          <script>
                muestra_pdb(600,500);
          </script>
         </div>
        <div id='optpdb' style='padding: 10px; margin-top: 30px;'>
         <!--    <a href="javascrip:;" onclick="muestra_pdb(500,500);">Larger</a> -->
            <?php echo "<font face='Helvetica,Arial' size=3><a
href='pdb_larger.php?orf=$quest'>Larger</a></font>" ?>
            <br><br>
            <input type="button" value="Get PDB file" onclick="msg()">
            <br>
            <input type="button" value="Get PTM file" onclick="msg()">
        </div>
         <div id='Caja_texto'>
            <b>Zoom the image:</b> Using scroll<br>
            <b>Rotate:</b> Drag the molecule<br>
            <b>Move:</b> Alt + Drag<br>
        </div>
       </div>
        <table style="width:100%">
          <tbody>
            <tr>
              <td id="secuencia" align=left></td>
              <td>
                <table id="Color" style="width:300" border="0" cellspacing="5" >
                <tbody>
                  <tr>
                    <td class="Ca">Ca</td>
                    <td class="Glycosylation">Glycosylation </td>
                    <td class="Succinylation">Succinylation </td>
                  </tr>
                  <tr>
                    <td class="Oxidation">Oxidation</td>
                    <td class="Lipidation">Lipidation</td>
```

```
                <td class="Acetylation">Acetylation </td>
              </tr>
              <tr>
                <td class="Ubiquitination">Ubiquitination</td>
                <td class="Nterminal">Nterminal</td>
                <td class="ActiveSite">ActiveSite</td>
              </tr>
<tr>
                <td class="Phosphorylation">Phosphorylation </td>
                <td class="Disulfide">Disulfide </td>
                <td class="Nitration">Nitration</td>
              </tr>
              <tr>
                <td class="Morethanone">Morethanone</td>
                <td class="Methylation">Methylation </td>
                <td class="Metal">Metal </td>
              </tr>
            </tbody>
            </table>
          </td>
        </tr>
      </tbody>
    </table>

    <br><br><br>

    <table id="mainres" class="display" cellspacing="0" width="100%">
    <thead>
      <tr>
        <th>Residue</th>
        <th>Position</th>
        <th>Modification</th>
        <th>pmid</th>
      </tr>
     </thead>
     <tbody id="modificaciones">
     </tbody>

     <tfoot>
      <tr>
        <th>Residue</th>
        <th>Position</th>
        <th>Modification</th>
        <th>pmid</th>
```

```
          </tr>
        </tfoot>
       </table>

    </div>
<!--     <div id="pdb" align="right">
       <script>
         muestra_pdb();
       </script>
    </div> -->

<!-- InstanceEndEditable -->
</div>
<?php include("footer_yaam.php") ?>
</div>
</body>
<!-- InstanceEnd --></html>
```

add_yaam.php

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Add a YAAM</title>
<link href="CSS/Index.css" rel="stylesheet" type="text/css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
</head>
<script>
      function Comprobar(obj){
            if(obj.entradaorf.value=="" || obj.entradapmid.value==""){
                  alert("ORF and pmid fields are mandatory");
                  return false;
            }else{
                  return true;
            }
}
</script>
<div id="Contenedor">
<header onclick="location.href='/YAAM/about_yaam.php';" id="header">
  <div onclick="location.href='/YAAM/about_yaam.php';" id="ifc"></div>
```

```
  <div id="menu">
    <ul>
      <li><a href="/YAAM/about_yaam.php">About YAAM</a></li>
      <li><a href="/YAAM/add_yaam.php">Add a YAAM</a></li>
      <li><a href="/YAAM/yaam_ini.php">Home</a></li>
    </ul>
  </div>
</header>

<div id="Cuerpo">
      <div id="Caja_texto2">
      <font color=#888><i>To add a new YAAM provide as much information as possible in the
fields bellow.<br>
      Fields marked with an asterisk (*) are mandatory.</font></i><br><br>
      <form id="add_yaam" method="POST" onsubmit="return Comprobar(this)"
            action="send.php" enctype="text/plain" >
        <table width="100%" border="0" align="center" cellspacing="5" cellpadding="5" >
          <tr align="left">
            <th>ORF*</th>
            <th>Position</th>
            <th>Residue</th>
            <th>Yaam</th>
            <th>pmid*</th>
          </tr>
          <tr align="left">
            <td><input id="entradaorf" name="entradaorf" type="text" size="20"></td>
            <td><input id="entradaposition" name="entradaposition" type="text" size="4"></td>
            <td><input id="entradaresidue" name="entradaresidue" type="text" size="1"></td>
            <td>
              <select id="entradacomoboyaam" name="entradacomboyaam" style="font-size:16px">
                <option selected>Choose one option</option>
                 <option>Acetylation</option>
                 <option>Phosphorylation</option>
                 <option>Ubiquitination</option>
                 <option>Succinylation</option>
                 <option>Methylation</option>
                 <option>Glycosylation</option>
                 <option>Lipidation</option>
                 <option>ActiveSite</option>
                 <option>Nitration</option>
                 <option>Nterminal</option>
                 <option>Ca</option>
                 <option>Disulfide</option>
```

```
                <option>Metal</option>
                <option>Oxidation</option>
              </select></td>
            <td><input id="entradapmid" name="entradapmid" type="text" size="12"></td>
          </tr>
        </table>
        <div id="Caja_texto3"><b>Comments:</b><br>
          <textarea id="comments" name="comments" rows="10" cols="100" style="font-
size:16px">
          </textarea><br><br>
        <b>Email:</b>  <input type="email" name="emailaddress" size=40 style="font-
size:16px">
           <input type="reset" value="  Clear  ">
           <input id="send" type="submit" value="  Submit  ">
        </div>
      </form>
      </div>
<br><br>
<?php include("footer_yaam.php") ?>
</div>
</html>
```

send.php

```
<?php

if(isset($_GET['emailaddress'])) {

// Debes editar las próximas dos líneas de código de acuerdo con tus preferencias
$email_to = "mail@correo.com.mx";
$email_subject = "New YAAM";

// Aquí se deberían validar los datos ingresados por el usuario
if(!isset($_GET['entradaorf']) ||
!isset($_GET['entradapmid']) ||
!isset($_GET['emailaddress'])) {

echo "<b>You must indicate ORF and pmid. </b><br />";
die();
}

$email_message = "Details of the possible new yaam:\n\n";
$email_message .= "ORF: " . $_GET['entradaorf'] . "\n";
$email_message .= "Position: " . $_GET['entradaposition'] . "\n";
```

```
$email_message .= "Residue: " . $_GET['entradaresidue'] . "\n";
$email_message .= "PMID: " . $_GET['entradapmid'] . "\n";
$email_message .= "Comments: " . $_GET['comments'] . "\n\n";


// Ahora se envía el e-mail usando la función mail() de PHP
$headers = 'From: '.$_GET['emailaddress']."\r\n".
'Reply-To: '.$_GET['emailaddress']."\r\n" .
'X-Mailer: PHP/' . phpversion();
if(mail($email_to, $email_subject, $email_message,$headers)){

echo "¡El formulario se ha enviado con éxito ha!.$email_to";
}else{
echo "ERROR";
}
}else{
echo "FAIL";
}

header("Location:add_yaam2.php");
?>
```

search.php

```
<?php

   $colr='';
   $k=0;
   $flag=0;
   $ctl=0;
   $colour="";

      if(isset($_GET['busqueda1'])){
            $busqueda=utf8_decode($_GET['busqueda1']);
            $ctl=1;
      }else if(isset($_GET['busqueda2'])){
            $busqueda=utf8_decode($_GET['busqueda2']);
            $ctl=0;
      }else if(isset($_GET['busqueda3'])){
            $busqueda=utf8_decode($_GET['busqueda3']);
            $ctl=2;
      }else if(isset($_GET['busqueda4'])){
            $busqueda=utf8_decode($_GET['busqueda4']);
            $ctl=3;
```

```
    }


  //$busqueda= utf8_decode('YPL268W');
  $busqueda= str_replace("'","",$busqueda);
  $busqueda = str_replace("""","",$busqueda);
  //$busqueda= str_replace('"','',utf8_decode('"aminopeptidase"'));

  $con=mysqli_connect("localhost","user","password") or die("Failed to connect with
database!!!!");
  mysqli_select_db( $con,"proteinas");


 // $query2="select distinct idprot,ubicaciones,sequence,sequence_order,standard_name,header
from Protein as pro join Location as la on pro.idprot=la.proteina_idprot join Equivalences as e
 // on pro.idprot=e.proteina_idprot where idprot='$busqueda' or e.alias='$busqueda';";



  $result = mysqli_query($con,"select distinct
idprot,ubicaciones,sequence,sequence_order,pro.standard_name,header from Protein as pro left
outer join Location as la on pro.idprot=la.proteina_idprot left outer join Equivalences as e on
pro.idprot=e.proteina_idprot  where pro.idprot='$busqueda' or e.standard_name='$busqueda' or
pro.aliases like '%$busqueda%';" );

  $gral = mysqli_query($con,"select distinct
idprot,ubicaciones,sequence,sequence_order,pro.standard_name,header from Protein as pro left
outer join Location as la on pro.idprot=la.proteina_idprot left outer join Equivalences as e on
pro.idprot=e.proteina_idprot  where pro.idprot='$busqueda' or e.standard_name='$busqueda' or
pro.aliases like '%$busqueda%' or header like '%$busqueda%';" );

  $query="select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
  where TABLE_NAME='Acetylation' and TABLE_SCHEMA='proteinas') as tn join Acetylation as ac
  join Source as src on ac.source_id=src.id join Equivalences as e on
ac.proteina_idprot=e.proteina_idprot where ac.proteina_idprot='$busqueda' or
e.standard_name='$busqueda'
  union
  select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
  where TABLE_NAME='Phosphorylation' and TABLE_SCHEMA='proteinas') as tn join
Phosphorylation as fs
```

```
   join Source as src on fs.source_id=src.id join Equivalences as e on
fs.proteina_idprot=e.proteina_idprot where fs.proteina_idprot='$busqueda' or
e.standard_name='$busqueda'
   union
   select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
   where TABLE_NAME='Glycosylation' and TABLE_SCHEMA='proteinas') as tn join Glycosylation as
gl
   join Source as src on gl.source_id=src.id join Equivalences as e on
gl.proteina_idprot=e.proteina_idprot where gl.proteina_idprot='$busqueda' or
e.standard_name='$busqueda'
   union
   select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
   where TABLE_NAME='Lipidation' and TABLE_SCHEMA='proteinas') as tn join Lipidation as lp
   join Source as src on lp.source_id=src.id join Equivalences as e on
lp.proteina_idprot=e.proteina_idprot where lp.proteina_idprot='$busqueda' or
e.standard_name='$busqueda'
   union
   select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
   where TABLE_NAME='Succinylation' and TABLE_SCHEMA='proteinas') as tn join Succinylation as
sc
   join Source as src on sc.source_id=src.id join Equivalences as e on
sc.proteina_idprot=e.proteina_idprot where sc.proteina_idprot='$busqueda' or
e.standard_name='$busqueda'
   union
   select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
   where TABLE_NAME='Ubiquitination' and TABLE_SCHEMA='proteinas') as tn join Ubiquitination
as ub
   join Source as src on ub.source_id=src.id join Equivalences as e on
ub.proteina_idprot=e.proteina_idprot where ub.proteina_idprot='$busqueda' or
e.standard_name='$busqueda'
   union
   select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
   where TABLE_NAME='Methylation' and TABLE_SCHEMA='proteinas') as tn join Methylation as
mt
   join Source as src on mt.source_id=src.id join Equivalences as e on
mt.proteina_idprot=e.proteina_idprot where mt.proteina_idprot='$busqueda' or
e.standard_name='$busqueda'
   union
```

```
   select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
   where TABLE_NAME='Oxidation' and TABLE_SCHEMA='proteinas') as tn join Oxidation as ox
   join Source as src on ox.source_id=src.id join Equivalences as e on
ox.proteina_idprot=e.proteina_idprot where ox.proteina_idprot='$busqueda' or
e.standard_name='$busqueda'
   union
   select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
   where TABLE_NAME='Ca' and TABLE_SCHEMA='proteinas') as tn join Ca as cb
   join Source as src on cb.source_id=src.id join Equivalences as e on
cb.proteina_idprot=e.proteina_idprot where cb.proteina_idprot='$busqueda' or
e.standard_name='$busqueda'
   union
   select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
   where TABLE_NAME='Metal' and TABLE_SCHEMA='proteinas') as tn join Metal as mb
   join Source as src on mb.source_id=src.id join Equivalences as e on
mb.proteina_idprot=e.proteina_idprot where mb.proteina_idprot='$busqueda' or
e.standard_name='$busqueda'
   union
   select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
   where TABLE_NAME='Nterminal' and TABLE_SCHEMA='proteinas') as tn join Nterminal as at
   join Source as src on at.source_id=src.id join Equivalences as e on
at.proteina_idprot=e.proteina_idprot where at.proteina_idprot='$busqueda' or
e.standard_name='$busqueda'
   union
   select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
   where TABLE_NAME='Nitration' and TABLE_SCHEMA='proteinas') as tn join Nitration as nt
   join Source as src on nt.source_id=src.id join Equivalences as e on
nt.proteina_idprot=e.proteina_idprot where nt.proteina_idprot='$busqueda' or
e.standard_name='$busqueda'
   union
   select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
   where TABLE_NAME='ActiveSite' and TABLE_SCHEMA='proteinas') as tn join ActiveSite as acs
   join Source as src on acs.source_id=src.id join Equivalences as e on
acs.proteina_idprot=e.proteina_idprot where acs.proteina_idprot='$busqueda' or
e.standard_name='$busqueda'
   union
   select residuo,posicion,pmid,fuente,TABLE_NAME from (select TABLE_NAME from
information_schema.TABLES
```

```
   where TABLE_NAME='Disulfide' and TABLE_SCHEMA='proteinas') as tn join Disulfide as db
   join Source as src on db.source_id=src.id join Equivalences as e on
db.proteina_idprot=e.proteina_idprot where db.proteina_idprot='$busqueda' or
e.standard_name='$busqueda' order by posicion;";

   $sth = mysqli_query($con,$query);
   //$sth1= mysqli_query($con,$query2);

   $rows = array();
   $data = array();
   $data2=array();

      if($ctl === 1){
   if(mysqli_num_rows($result)){

      while($r = mysqli_fetch_assoc($result)) {

         $temp = array();

         $temp = array('systematic_name' => (string)
utf8_encode($r['idprot']),'location'=>(string)utf8_encode($r['ubicaciones']),'sequence'=>(string)utf
8_encode($r['sequence']),'sequence_order'=>(string)utf8_encode($r['sequence_order']),'standard
_name'=>(string)utf8_encode($r['standard_name']),'header'=>(string)utf8_encode(substr($r['hea
der'],0,-1)));

            array_push($rows,$temp);
      }


  //si la proteina no tiene localizacion
   }else{

      $result = mysqli_query($con,"select distinct
idprot,sequence,sequence_order,pro.standard_name,header from Protein as pro join
Equivalences as e on pro.idprot=e.proteina_idprot  where pro.idprot='$busqueda' or
e.standard_name='$busqueda';");
      //si no se encuentra la proteina se busca en el header
      if(mysqli_num_rows($result) === 0){
         $result = mysqli_query($con,"select idprot, standard_name,header from Protein where
header like '%$busqueda%'");
         $rows = array();

         while($r=  mysqli_fetch_assoc($result)){
            $temp = array();
```

```
            $temp = array('systematic_name' => (string)
utf8_encode($r['idprot']),'common_name'=>(string)utf8_encode($r['standard_name']),'header'=>(
string)utf8_encode($r['header']));
            array_push($rows,$temp);
          }
          $flag=1;
       }else{

          while($r = mysqli_fetch_assoc($result)) {

            $temp = array();

            $temp = array('systematic_name' => (string)
utf8_encode($r['idprot']),'sequence'=>(string)utf8_encode($r['sequence']),'sequence_order'=>(stri
ng)utf8_encode($r['sequence_order']),'standard_name'=>(string)utf8_encode($r['standard_name'
]),'header'=>(string)utf8_encode($r['header']));
            array_push($rows,$temp);


              }


          }
}
}else if($ctl ===0){

   //Llenado de arreglo rows
   while($r = mysqli_fetch_assoc($sth)) {

      $temp = array();
      if($r['pmid']){
         $temp = array('residue' => (string)utf8_encode($r['residuo']),'position'
=>(int)$r['posicion'],'pmid' => (int)$r['pmid'],'modification'
=>(string)utf8_encode($r['TABLE_NAME']));
      }else{
         $temp = array('residue' => (string)utf8_encode($r['residuo']),'position'
=>(int)$r['posicion'],'pmid' => (string)utf8_encode('uniprot.org'),'modification'
=>(string)utf8_encode($r['TABLE_NAME']));
      }
      array_push($rows,$temp);
   }

   //concatena pmid
   for($s=0;$s<count($rows);$s++){
```

```
      if($s>0){
         if($rows[$s]['position']===$rows[$s-1]['position'] && $rows[$s]['modification']===$rows[$s-
1]['modification']){
            $rows[$s]['pmid']=$rows[$s]['pmid'].",".$rows[$s-1]['pmid'];
            array_splice($rows, $s-1,1);
            $s-=1;
         }
      }
   }
      //concatena modificaciones
   for($s=0;$s<count($rows);$s++){
      if($s>0){
         if($rows[$s]['position']===$rows[$s-1]['position'] && $rows[$s]['modification']!==$rows[$s-
1]['modification']){
            $rows[$s]['modification']=" ".$rows[$s]['modification'].",".$rows[$s-1]['modification'];
            $rows[$s]['pmid']=$rows[$s]['pmid'].",".$rows[$s-1]['pmid'];
            array_splice($rows, $s-1,1);
            $s-=1;
         }
      }
   }
         //elimina valores repetidos de pmid y modificaciones
   for($i=0; $i<count($rows); $i++){
      $pmids=explode(',',$rows[$i]['pmid']);
      $modifs=explode(',',$rows[$i]['modification']);

      $rows[$i]['pmid']=implode(",",array_unique($pmids));
      $rows[$i]['modification']=implode(",",array_unique($modifs));
   }


}else if($ctl===2){

   //Llenado de arreglo rows
   while($r = mysqli_fetch_assoc($sth)) {

      $temp = array();
      if($r['pmid']){
         $temp = array('residue' => (string)utf8_encode($r['residuo']),'position'
=>(int)$r['posicion'],'pmid' => (int)$r['pmid'],'modification'
=>(string)utf8_encode($r['TABLE_NAME']));
      }else{
```

```
        $temp = array('residue' => (string)utf8_encode($r['residuo']),'position'
=>(int)$r['posicion'],'pmid' => (string)utf8_encode('uniprot.org'),'modification'
=>(string)utf8_encode($r['TABLE_NAME']));
      }
      array_push($rows,$temp);
   }

   //concatena pmid
   for($s=0;$s<count($rows);$s++){
      if($s>0){
         if($rows[$s]['position']===$rows[$s-1]['position'] && $rows[$s]['modification']===$rows[$s-
1]['modification']){
            $rows[$s]['pmid']=$rows[$s]['pmid'].",".$rows[$s-1]['pmid'];
            array_splice($rows, $s-1,1);
            $s-=1;
         }
      }
   }
      //concatena modificaciones
   for($s=0;$s<count($rows);$s++){
      if($s>0){
         if($rows[$s]['position']===$rows[$s-1]['position'] && $rows[$s]['modification']!==$rows[$s-
1]['modification']){
            $rows[$s]['modification']=" ".$rows[$s]['modification'].",".$rows[$s-1]['modification'];
            $rows[$s]['pmid']=$rows[$s]['pmid'].",".$rows[$s-1]['pmid'];
            array_splice($rows, $s-1,1);
            $s-=1;
         }
      }
   }
         //elimina valores repetidos de pmid y modificaciones
   for($i=0; $i<count($rows); $i++){
      $pmids=explode(',',$rows[$i]['pmid']);
      $modifs=explode(',',$rows[$i]['modification']);

      $rows[$i]['pmid']=implode(",",array_unique($pmids));
      $rows[$i]['modification']=implode(",",array_unique($modifs));
   }

      for($k=0;$k<count($rows);$k++){



      switch($rows[$k]['modification']){
```

```
case "Acetylation":
	$colour="#FBBA00";
	break;
case "Phosphorylation":
	$colour="#D00A10";
	break;
case "Ca":
	$colour="#66FFFF";
	break;
case "Disulfide":
	$colour="#BE2E46";
	break;
case "Glycosylation":
	$colour="#95039F";
	break;
case "Lipidation":
	$colour="#ADE415";
	break;
case "Metal":
	$colour="#66B7EE";
	break;
case "Methylation":
	$colour="#6456FD";
	break;
case "Oxidation":
	$colour="#06974A";
	break;
case "Succinylation":
	$colour="#F4667C";
	break;
case "Ubiquitination":
	$colour="#9900EE";
	break;
case "Nterminal":
	$colour="#ED22C8";
	break;
case "Nitration":
	$colour="#1502B8";
	break;
case "ActiveSite":
	$colour="#515151";
	break;
default:
	$colour="#775533";
```

```
                break;
        }

        $rows[$k]['modification']="<span style='color:$colour'>".$rows[$k]['modification']."<span>";
        }

}else if($ctl===3){

   if(mysqli_num_rows($gral)){

        while($r = mysqli_fetch_assoc($gral)) {

            $temp = array();

            $temp = array('systematic_name' => (string)
utf8_encode($r['idprot']),'common_name'=>(string)utf8_encode($r['standard_name']),'location'=
>(string)utf8_encode($r['ubicaciones']));

                array_push($rows,$temp);
        }

  //si la proteina no tiene localizacion
   }else{

        $gral = mysqli_query($con,"select distinct
idprot,sequence,sequence_order,pro.standard_name,header from Protein as pro left outer join
Equivalences as e on pro.idprot=e.proteina_idprot  where pro.idprot='$busqueda' or
e.standard_name='$busqueda';");
        //si no se encuentra la proteina se busca en el header
        if(mysqli_num_rows($gral) === 0){
            $gral = mysqli_query($con,"select idprot, standard_name,header,ubicaciones from Protein
as p left outer join Location as l on p.idprot=l.proteina_idprot where header like '%$busqueda%'");
            $rows = array();

            while($r=  mysqli_fetch_assoc($gral)){
                $temp = array();
                $temp = array('systematic_name' => (string)
utf8_encode($r['idprot']),'common_name'=>(string)utf8_encode($r['standard_name']),'location'=
>(string)utf8_encode($r['ubicaciones']));
                array_push($rows,$temp);
            }
            $flag=1;

        }else{
```

```
        // echo "AQUI";
         while($r = mysqli_fetch_assoc($gral)) {

            $temp = array();

            $temp = array('systematic_name' => (string)
utf8_encode($r['idprot']),'common_name'=>(string)utf8_encode($r['standard_name']),'location'=
>"");
            array_push($rows,$temp);


              }


        }
}
}


     $data['data']=$rows;

   if(!empty($rows)){
      if($flag === 0){
        echo  $jsonTable= json_encode($data);
          $jsonTableh = "";
      }else{
        echo  $jsonTableh= json_encode($data);
          $jsonTable = "";
      }
   }else{
     echo  $jsonTable=  json_encode($data2);
   }

   mysqli_close($con);

?>
```

Localization.php

```
<?php

   $locations= urldecode($_GET['location']);
   $modifications = urldecode($_GET['modification']);
```

```
   $residue=$_GET['residue'];

      $locations=stripslashes($locations);
      $modifications=stripslashes($modifications);

      $locations=json_decode($locations,true);
      $modifications=json_decode($modifications,true);

   $con=mysqli_connect("localhost","user","password") or die("Failed to connect with
database!!!!");
   mysqli_select_db( $con,"proteinas");


      $longmod = count($modifications);
      $longloc = count($locations);

      switch($longmod){
         case 0:
            $query=onlyLoc($longloc,$locations);
            break;
         case 1:
            if($longloc ===0){
               $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                  . "$modifications[0] as a on p.idprot=a.proteina_idprot left outer join Location as l on
a.proteina_idprot=l.proteina_idprot";
            }elseif($longloc ===1){
               $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                  . "$modifications[0] as a on p.idprot=a.proteina_idprot join Location as l on
a.proteina_idprot=l.proteina_idprot where "
                  . "ubicaciones like '$locations[0]%' or ubicaciones like '%,$locations[0]%' or
ubicaciones like '%,$locations[0],%'";
            }elseif($longloc === 2){
               $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                  . "$modifications[0] as a on p.idprot=a.proteina_idprot join Location as l on
a.proteina_idprot=l.proteina_idprot where "
                  . "ubicaciones like '$locations[0]%' or ubicaciones like '$locations[1]%' or ubicaciones
like '%,$locations[0]%' or "
                  . "ubicaciones like '%,$locations[1]%' or ubicaciones like '%,$locations[0],%' or
ubicaciones like '%,$locations[1],%'";
            }elseif($longloc === 3){
```

```
            $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                . "$modifications[0] as a on p.idprot=a.proteina_idprot join Location as l on
a.proteina_idprot=l.proteina_idprot where "
                . "ubicaciones like '$locations[0]%' or ubicaciones like '$locations[1]%' or ubicaciones
like '$locations[2]%' or "
                . "ubicaciones like '%,$locations[0]%' or ubicaciones like '%,$locations[1]%' or
ubicaciones like '%,$locations[2]%' or "
                . "ubicaciones like '%,$locations[0],%' or ubicaciones like '%,$locations[1],%' or
ubicaciones like '%,$locations[2],%'";
            }
            break;
        case 2:
            if($longloc===0){
                if($residue==="yes"){
                    $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                        . "$modifications[0] as a on p.idprot=a.proteina_idprot join
$modifications[1] as b on a.proteina_idprot=b.proteina_idprot left outer join"
                        . " Location as l on b.proteina_idprot=l.proteina_idprot where
a.residuo=b.residuo and a.posicion=b.posicion";
                }else{
                    $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                        . "$modifications[0] as a on p.idprot=a.proteina_idprot join
$modifications[1] as b on a.proteina_idprot=b.proteina_idprot left outer join"
                        . " Location as l on b.proteina_idprot=l.proteina_idprot";
                }
            }elseif($longloc===1){
                if($residue==="yes"){
                    $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                        . "$modifications[0] as a on p.idprot=a.proteina_idprot join
$modifications[1] as b on a.proteina_idprot=b.proteina_idprot join"
                        . " Location as l on b.proteina_idprot=l.proteina_idprot where ubicaciones
like '$locations[0]%' or ubicaciones like '%,$locations[0]%'"
                        . " or ubicaciones like '%,$locations[0],%' and a.residuo=b.residuo and
a.posicion=b.posicion";
                }else{
                    $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                        . "$modifications[0] as a on p.idprot=a.proteina_idprot join
$modifications[1] as b on a.proteina_idprot=b.proteina_idprot join"
```

```
                        . " Location as l on b.proteina_idprot=l.proteina_idprot where ubicaciones
like '$locations[0]%' or ubicaciones like '%,$locations[0]%'"
                        . " or ubicaciones like '%,$locations[0],%'";
                }
            }elseif($longloc===2) {
                if($residue==="yes"){
                    $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                        . "$modifications[0] as a on p.idprot=a.proteina_idprot join
$modifications[1] as b on a.proteina_idprot=b.proteina_idprot join"
                        . " Location as l on b.proteina_idprot=l.proteina_idprot where ubicaciones
like '$locations[0]%' or ubicaciones like '$locations[1]%'"
                        . " or ubicaciones like '%,$locations[0]%' or ubicaciones like
'%,$locations[1]%' or ubicaciones like '%,$locations[0],%' or ubicaciones"
                        . " like '%,$locations[1],%' and a.residuo=b.residuo and
a.posicion=b.posicion";
                }else{
                    $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                        . "$modifications[0] as a on p.idprot=a.proteina_idprot join
$modifications[1] as b on a.proteina_idprot=b.proteina_idprot join"
                        . " Location as l on b.proteina_idprot=l.proteina_idprot where ubicaciones
like '$locations[0]%' or ubicaciones like '$locations[1]%'"
                        . " or ubicaciones like '%,$locations[0]%' or ubicaciones like
'%,$locations[1]%' or ubicaciones like '%,$locations[0],%' or ubicaciones"
                        . " like '%,$locations[1],%'";

                }
            }elseif($longloc===3){
                if($residue==="yes"){
                    $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                        . "$modifications[0] as a on p.idprot=a.proteina_idprot join
$modifications[1] as b on a.proteina_idprot=b.proteina_idprot join"
                        . " Location as l on b.proteina_idprot=l.proteina_idprot where ubicaciones
like '$locations[0]%' or ubicaciones like '$locations[1]%'"
                        . " ubicaciones like '$locations[2]%' or ubicaciones like '%,$locations[0]%' or
ubicaciones like '%,$locations[1]%' or ubicaciones like"
                        . " '%,$locations[2]%' or ubicaciones like '%,$locations[0],%' or ubicaciones
like '%,$locations[1],%' or ubicaciones like '%,$locations[2],%'"
                        . " and a.residuo=b.residuo and a.posicion=b.posicion";
                }else{
                    $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
```

```
                        . "$modifications[0] as a on p.idprot=a.proteina_idprot join
$modifications[1] as b on a.proteina_idprot=b.proteina_idprot join"
                        . " Location as l on b.proteina_idprot=l.proteina_idprot where ubicaciones
like '$locations[0]%' or ubicaciones like '$locations[1]%'"
                        . " ubicaciones like '$locations[2]%' or ubicaciones like '%,$locations[0]%' or
ubicaciones like '%,$locations[1]%' or ubicaciones like"
                        . " '%,$locations[2]%' or ubicaciones like '%,$locations[0],%' or ubicaciones
like '%,$locations[1],%' or ubicaciones like '%,$locations[2],%'";
                }
            }
            break;
        case 3:
            if($longloc===0){
                if($residue==="yes"){
                    $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                        . "$modifications[0] as a on p.idprot=a.proteina_idprot join
$modifications[1] as b on a.proteina_idprot=b.proteina_idprot join "
                        . "$modifications[2] as c on b.proteina_idprot=c.proteina_idprot left outer
join Location as l on b.proteina_idprot=l.proteina_idprot"
                        . " where a.residuo=b.residuo and b.residuo=c.residuo and
a.posicion=b.posicion and b.posicion=c.posicion";
                }else{
                    $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                        . "$modifications[0] as a on p.idprot=a.proteina_idprot join
$modifications[1] as b on a.proteina_idprot=b.proteina_idprot join "
                        . "$modifications[2] as c on b.proteina_idprot=c.proteina_idprot left outer
join Location as l on b.proteina_idprot=l.proteina_idprot";
                }
            }elseif($longloc===1){
                if($residue==="yes"){
                    $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                        . "$modifications[0] as a on p.idprot=a.proteina_idprot join
$modifications[1] as b on a.proteina_idprot=b.proteina_idprot join "
                        . "$modifications[2] as c on b.proteina_idprot=c.proteina_idprot join
Location as l on b.proteina_idprot=l.proteina_idprot "
                        . "where ubicaciones like '$locations[0]%' or ubicaciones like
',%$locations[0]%' or ubicaciones like '%,$locations[0],%'"
                        . " and a.residuo=b.residuo and b.residuo=c.residuo and
a.posicion=b.posicion and b.posicion=c.posicion";
                }else{
```

```
                $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                    . "$modifications[0] as a on p.idprot=a.proteina_idprot join
$modifications[1] as b on a.proteina_idprot=b.proteina_idprot join "
                    . "$modifications[2] as c on b.proteina_idprot=c.proteina_idprot join
Location as l on b.proteina_idprot=l.proteina_idprot "
                    . "where ubicaciones like '$locations[0]%' or ubicaciones like
',%$locations[0]%' or ubicaciones like '%,$locations[0],%'";
                }
            }elseif($longloc===2){
                if($residue==="yes"){
                    $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                    . " $modifications[0] as a on p.idprot=a.proteina_idprot join
$modifications[1] as b on a.proteina_idprot=b.proteina_idprot join "
                    . " $modifications[2] as c on b.proteina_idprot=c.proteina_idprot join
Location as l on b.proteina_idprot=l.proteina_idprot "
                    . " where ubicaciones like '$locations[0]%' or ubicaciones like
'$locations[1]%' or ubicaciones like '%,$locations[0]%' or ubicaciones"
                    . " like '%,$locations[1]%' or ubicaciones like '%,$locations[0],%' or
ubicaciones like '%,$locations[1],%' and a.residuo=b.residuo and"
                    . " b.residuo=c.residuo and a.posicion=b.posicion and
b.posicion=c.posicion";
                }else{
                    $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                    . " $modifications[0] as a on p.idprot=a.proteina_idprot join
$modifications[1] as b on a.proteina_idprot=b.proteina_idprot join "
                    . " $modifications[2] as c on b.proteina_idprot=c.proteina_idprot join
Location as l on b.proteina_idprot=l.proteina_idprot "
                    . " where ubicaciones like '$locations[0]%' or ubicaciones like
'$locations[1]%' or ubicaciones like '%,$locations[0]%' or ubicaciones"
                    . " like '%,$locations[1]%' or ubicaciones like '%,$locations[0],%' or
ubicaciones like '%,$locations[1],%'";
                }
            }elseif($longloc===3){
                if($residue==="yes"){
                    $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                    . " $modifications[0] as a on p.idprot=a.proteina_idprot join
$modifications[1] as b on a.proteina_idprot=b.proteina_idprot join "
                    . " $modifications[2] as c on b.proteina_idprot=c.proteina_idprot join
Location as l on b.proteina_idprot=l.proteina_idprot "
```

```
                        . " where ubicaciones like '$locations[0]%' or ubicaciones like
'$locations[1]%' or ubicaciones like '$locations[2]%' or ubicaciones"
                        . " like '%,$locations[0]%' or ubicaciones like '%,$locations[1]%' or
ubicaciones like '%,$locations[2]%' or ubicaciones like '%,$locations[0],%'"
                        . " or ubicaciones '%,$locations[1],%' or ubicaciones like '%,$locations[2],%'
and a.residuo=b.residuo and b.residuo=c.residuo"
                        . " and a.posicion=b.posicion and b.posicion=c.posicion";
                }else{
                    $query="select distinct a.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                        . " $modifications[0] as a on p.idprot=a.proteina_idprot join
$modifications[1] as b on a.proteina_idprot=b.proteina_idprot join "
                        . " $modifications[2] as c on b.proteina_idprot=c.proteina_idprot join
Location as l on b.proteina_idprot=l.proteina_idprot "
                        . " where ubicaciones like '$locations[0]%' or ubicaciones like
'$locations[1]%' or ubicaciones like '$locations[2]%' or ubicaciones"
                        . " like '%,$locations[0]%' or ubicaciones like '%,$locations[1]%' or
ubicaciones like '%,$locations[2]%' or ubicaciones like '%,$locations[0],%'"
                        . " or ubicaciones '%,$locations[1],%' or ubicaciones like
'%,$locations[2],%'";
                }
            }
            break;
    }

    function onlyLoc($longloc,$locations){
       switch($longloc){
          case 1:
             $query="select distinct l.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                . "Location as l on p.idprot=l.proteina_idprot where ubicaciones like '$locations[0]%'
or ubicaciones like "
                . "'%,$locations[0]%' or ubicaciones like '%,$locations[0],%'";
             break;
          case 2:
             $query="select distinct l.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                . "Location as l on p.idprot=l.proteina_idprot where ubicaciones like '$locations[0]%'
or ubicaciones like '$locations[1]%'"
                . "or ubicaciones like '%,$locations[0]%' or ubicaciones like '%,$locations[1]%' or
ubicaciones like '%$locations[0]%' or ubicaciones like '%,$locations[1],%'";
             break;
          case 3:
```

```
            $query="select distinct l.proteina_idprot as orf, p.standard_name as common,
ubicaciones from Protein as p join "
                . "Location as l on p.idprot=l.proteina_idprot where ubicaciones like '$locations[0]%'
or ubicaciones like '$locations[1]%' "
                . "or ubicaciones like '$locations[2]%' or ubicaciones like '%,$locations[0]%' or
ubicaciones like '%,$locations[1]%' or ubicaciones like '%,$locations[2]%'"
                . "or ubicaciones like '%,$locations[0],%' or ubicaciones like '%,$locations[1],%' or
ubicaciones  like '%,$locations[2],%'";
            break;
        }
        return $query;
    }
    //$query="select l.proteina_idprot as orf,p.standard_name as common from Location as l join
Protein as p on l.proteina_idprot=p.idprot where ubicaciones like '%$location%';";

  $sth = mysqli_query($con,$query);

   $rows = array();

  $data=array();

  while($r = mysqli_fetch_assoc($sth)) {

    $temp = array();
    $temp = array('systematic_name' => (string) utf8_encode($r['orf']),'common_name' =>
(string)utf8_encode($r['common']),'location' => (string)utf8_encode($r['ubicaciones']));

    array_push($rows,$temp);

  }

  $data['data']=$rows;
  $jsonTable= json_encode($data);

  echo $jsonTable;

  mysqli_close($con);

?>
```