Capítulo 2:

CONCEPTOS TÉCNICOS PARA LA ELABORACIÓN DEL SISTEMA.

2.1 Lenguaje Orientado a Objetos.

Clases.

Las clases son declaraciones de objetos, también se podrían definir como abstracciones de objetos. Esto quiere decir que la definición de un objeto es la clase. Cuando programamos un objeto y definimos sus características y funcionalidades en realidad lo que estamos haciendo es programar una clase.

Objetos.

Los objetos son ejemplares de una clase cualquiera. Cuando creamos un ejemplar tenemos que especificar la clase a partir de la cual se creará. Esta acción de crear un objeto a partir de una clase se llama instanciar

Principios de la programación orientada a objetos.

ABSTRACCIÓN.
Denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar cómo se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos y cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción. El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real. La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere atacar.

ENCAPSULAMIENTO.
El encapsulamiento permite a los objetos elegir qué información es publicada y qué información es ocultada al resto de los objetos. Para ello los objetos suelen presentar sus métodos como interfaces públicas y sus atributos como datos privados e inaccesibles desde otros objetos. Para permitir que otros objetos consulten o modifiquen los atributos de los objetos, las clases suelen presentar métodos de acceso. De esta manera el acceso a los datos de los objetos es controlado por el programador, evitando efectos laterales no deseados.
Con el encapsulado de los datos se consigue que las personas que utilicen un objeto sólo tengan que comprender su interfaz, olvidándose de cómo está implementada, y en definitiva, reduciendo la complejidad de utilización.

POLIMORFISMO.
Quiere decir comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama asignación tardía o asignación dinámica.

HERENCIA.
En la programación orientada a objetos las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el

polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Esto suele hacerse habitualmente agrupando los objetos en *clases* y estas en *árboles* o *enrejados* que reflejan un comportamiento común.

2.2 Bases de datos

Conceptos generales.

BASE DE DATOS.

Todas las empresas requieren almacenar información. Desde siempre lo han hecho. La información puede ser de todo tipo. Cada elemento informativo (nombre, dirección, sueldo, etc.) es lo que se conoce como dato (en inglés data); en tanto las soluciones utilizadas por las empresas para almacenar los datos son diversas.

Antes de la aparición de la informática se almacenaban en ficheros con cajones y carpetas y fichas. Tras la aparición de la informática estos datos se almacenan en archivos digitales dentro de las unidades de almacenamiento de una computadora (a veces en archivos binarios, o en hojas de cálculo, etc.).

Además las empresas requieren utilizar aplicaciones informáticas para realizar tareas propias de la empresa a fin de mecanizar a las mismas. Estas aplicaciones requieren manejar los datos de la empresa.

Lógicamente la solución a este problema es hacer que todas las aplicaciones utilicen los mismos datos. Esto provoca que los datos deban estar mucho más protegidos y controlados. Además los datos forman una estructura física y funcional que es lo que se conoce como **base de datos.**

SISTEMA DE BASES DE DATOS.

Un sistema de bases de datos sirve para integrar los datos. Lo componen los siguientes elementos:

- **Hardware**. Máquinas en las que se almacenan las bases de datos. Incorporan unidades de almacenamiento masivo para este fin.
- **Software**. Es el sistema gestor de bases de datos. El encargado de administrar las bases de datos.
- **Datos**. Incluyen los datos que se necesitan almacenar y los metadatos que son datos que sirven para describir lo que se almacena en la base de datos.
- **Usuarios**. Personas que manipulan los datos del sistema. Hay tres categorías:
    - Usuarios finales.
    - Desarrolladores.
    - Administradores.

ESTRUCTURA DE UNA BASE DE DATOS.

Donde se especifica la organización de los datos se producen dos visiones:

- **Estructura lógica.** Indica la composición y distribución teórica de la base de datos. La estructura lógica sirve para que las aplicaciones puedan utilizar los elementos de la base de datos sin saber realmente cómo se están almacenando.

Es una estructura que permite idealizar a la base de datos. Sus elementos son objetos, entidades, nodos, relaciones, enlaces,... que realmente no tienen presencia real en la forma física del sistema. Por ello para acceder a los datos tiene que haber una posibilidad de traducir la estructura lógica en la estructura física.

- **Estructura física.** Es la estructura de los datos tan cual se almacenan en las unidades de disco. La correspondencia entre la estructura lógica y la física se almacena en la base de datos (en los metadatos).

Ventajas.

- Independencia de los datos y los programas y procesos.
- Menor redundancia.
- Integridad de los datos.
- Mayor seguridad en los datos.
- Datos más documentados.
- Acceso a los datos más eficiente.
- Menor espacio de almacenamiento.

Desventajas.

- Instalación costosa.
- Requiere personal calificado.
- Implementación larga y difícil.
- Ausencia de estándares reales.

Sistema Gestor de Bases de Datos

Un sistema gestor de bases de datos o SGBD (aunque se suele utilizar más a menudo las siglas DBMS procedentes del inglés, Data Base Management System) es el software que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos, cuyas funciones principales son:

- **Función de descripción.** Sirve para describir los datos, sus relaciones y sus condiciones de acceso e integridad. Además del control de vistas de usuarios y de la especificación de las características físicas de la base de datos. Para poder realizar todas estas operaciones se utiliza un lenguaje de definición de datos o DDL.
- **Función de manipulación.** Permite buscar, añadir, suprimir y modificar datos de la base de datos. El DBMS proporciona una lenguaje de manipulación de datos (DML) para realizar esta función.
- **Función de control.** Incorpora las funciones que permiten una buena comunicación con la base de datos. Además proporciona al DBA los procedimientos necesarios para realizar su labor.

FUNCIONAMIENTO BÁSICO DE UN DBMS.

Resumido en este esquema con las siguientes especificaciones:

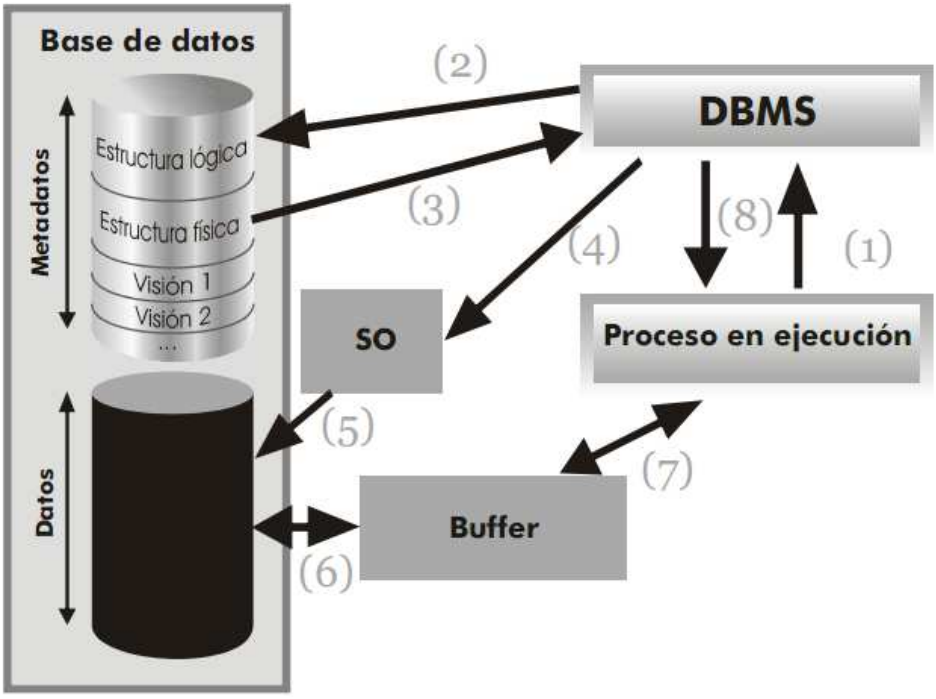


Figura 2.2.1.1 Esquema de los procesos básicos en el funcionamiento de un DBMS.

1. El proceso lanzado por el usuario llama al DBMS indicando la porción de la base de datos que se desea tratar.
2. El DBMS traduce la llamada a términos del esquema lógico de la base de datos. Accede al esquema lógico comprobando derechos de acceso y la traducción física.
3. El DBMS obtiene el esquema físico.
4. El DBMS traduce la llamada a los métodos de acceso del Sistema Operativo que permiten acceder a los datos requeridos.
5. El Sistema Operativo accede a los datos tras traducir las órdenes dadas por el DBMS.
6. Los datos pasan del disco a una memoria intermedia o buffer. En ese buffer se almacenarán los datos según se vayan recibiendo.
7. Los datos pasan del buffer al área de trabajo del usuario (ATU) del proceso del usuario.
8. El DBMS devuelve indicadores en los que manifiesta si ha habido errores o advertencias a tener en cuenta. Esto se indica al área de comunicaciones del proceso de usuario. Si las indicaciones son satisfactorias, los datos de la ATU serán utilizables por el proceso de usuario.

Modelado de datos.

Los modelos se utilizan en todo tipo de ciencias. Su finalidad es la de simbolizar una parte del mundo real de forma que sea más fácilmente manipulable. En definitiva es un esquema mental (conceptual) en el que se intentan reproducir las características de una realidad específica.
En el caso de los modelos de datos, lo que intentan reproducir es una información real que deseamos almacenar en un sistema informático.
Se denomina esquema a una descripción específica en términos de un modelo de datos. El conjunto de datos representados por el esquema forma la base de datos.
Pasos para el modelado:

1. Encontrar entidades (conjuntos de entidades).
2. Identificar atributos de las entidades.
3. Buscar identificadores.
4. Especificar las relaciones y cardinalidades.
5. Identificar entidades débiles.
6. Especializar y generalizar entidades donde sea posible.

MODELO ENTIDAD RELACIÓN E/R.

Fue ideado por Peter Chen en los años 1976 y 1977 a través de dos artículos. Se trata de un modelo que sirve para crear esquemas conceptuales de bases de datos. De hecho es prácticamente un estándar para crear esta tarea.

Se le llama modelo E/R, y para poder utilizarlo son necesarios los siguientes conceptos:

- **Entidad:** Se trata de cualquier objeto u elemento (real o abstracto) acerca del cual se pueda almacenar información en la base de datos. Las entidades que poseen las mismas propiedades forman conjuntos de entidades.
- **Entidades regulares:** Son las entidades normales que tienen existencia por sí mismas sin depender de otras.
- **Entidades débiles:** Su existencia depende de otras.
- **Relación:** Representan asociaciones entre entidades. Es el elemento del modelo que permite relacionar en sí los datos del modelo.
- **Cardinalidad:** Indica el número de relaciones en las que una entidad puede aparecer.
- **Cardinalidad mínima:** Indica el número mínimo de asociaciones en las que aparecerá cada ejemplar de la entidad (el valor que se anota es de cero o uno).
- **Cardinalidad máxima:** Indica el número máximo de relaciones en las que puede aparecer cada ejemplar de la entidad (puede ser uno o muchos).
- **Rol:** Los roles representan el papel que juega una entidad en una determinada relación.
- **Atributos:** Describen propiedades de las entidades y las relaciones, pueden ser compuestos: varios atributos forman un atributo general, múltiples: con diferentes valores, opcionales: pueden ser nulos.
- **Llave foránea:** Utilizada para establecer relaciones entre entidades (tablas), se trata de un atributo o llave primaria que comparten 2 o más tablas

IDENTIFICADOR O LLAVE PRIMARIA

Se trata de uno o más campos cuyos valores son únicos en cada ejemplar de una entidad. Se indican subrayando el nombre del identificador. Para que un atributo sea considerado un buen identificador tiene que cumplir:

- Deben distinguir a cada ejemplar teniendo en cuenta las entidades que utiliza el modelo. No tiene que ser un identificador absoluto.
- Todos los ejemplares de una entidad deben tener el mismo identificador.
- Cuando un atributo es importante aun cuando no tenga una entidad concreta asociada, entonces se trata de una entidad y no de un atributo

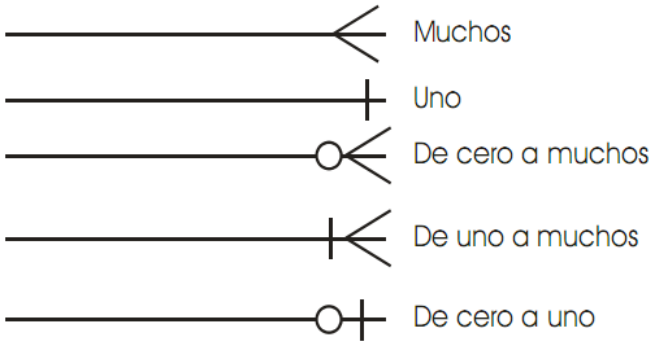
REPRESENTACIÓN GRÁFICA.

Entidad.
Entidad fuerte.

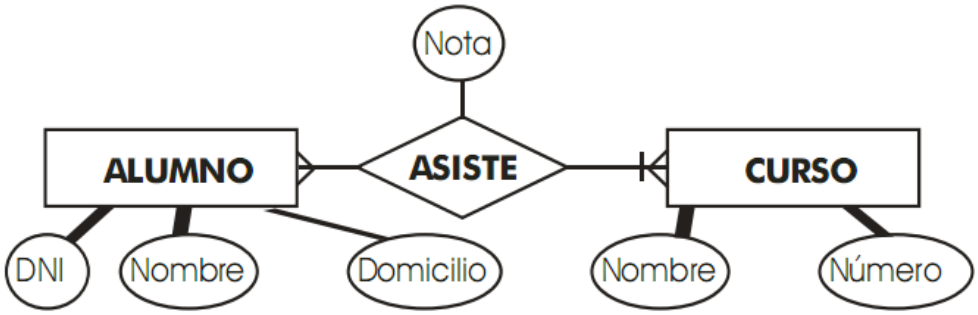
Entidad débil.

Relación.

Cardinalidad.



Ejemplo.



Normalización.

La normalización es el proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y más estables, son más fáciles de mantener. También se puede entender la normalización como una serie de reglas que sirven para ayudar a los diseñadores de bases de datos a desarrollar un esquema que minimice los problemas de lógica. Cada regla está basada en la que le antecede.

La normalización de una relación es un proceso muy definido, en el cual existen algoritmos de normalización.

El proceso de normalización se trata de las dependencias que existen entre los atributos (campos) de la relación. Los pasos a seguir son los siguientes:

1. Cálculo de las dependencias: funcionales, multivaluadas, jerárquicas y en combinación que existen entre los atributos de una relación.
2. Cálculo del recubrimiento minimal.
3. Cálculo de las claves candidatas de la relación de los atributos principales y de los no principales.
4. Cálculo de la forma normal en la que se encuentra la relación.
5. Aplicar los métodos de síntesis o análisis para obtener la forma normal deseada.

GRADO DE NORMALIZACIÓN DESEADO

La siguiente decisión es ¿qué tan lejos debe llevar la normalización? La normalización es una ciencia subjetiva. Determinar las necesidades de simplificación depende del diseñador. Si la base de datos va a proveer información a un solo usuario para un propósito simple y existen pocas posibilidades de expansión, normalizar los datos hasta la 3FN es exagerado. Las reglas de normalización existen como guías para crear tablas que sean fáciles de manejar, así como flexibles y eficientes. A veces puede ocurrir que normalizar los datos hasta el nivel más alto no tenga sentido.

FORMAS NORMALES

Existen básicamente tres niveles de normalización: Primera Forma Normal (1NF), Segunda Forma Normal (2NF) y Tercera Forma Normal (3NF). Cada una de estas formas tiene sus propias reglas. Cuando una base de datos se conforma a un nivel, se considera normalizada a esa forma de normalización.

- Primera forma normal: establece que las columnas repetidas deben eliminarse y colocarse en tablas separadas.
- Segunda forma normal: establece que todas las dependencias parciales se deben eliminar y separar dentro de sus propias tablas.
- Tercera forma normal: se da cuando todas las columnas que no son llave son funcionalmente dependientes por completo de la llave primaria y no hay dependencias transitivas.

Existen seis niveles más de normalización que no se han discutido aquí. Ellos son Forma Normal Boyce-Codd, Cuarta Forma Normal (4NF), Quinta Forma Normal (5NF) o Forma Normal de Proyección-Unión, Forma Normal de Proyección-Unión Fuerte, Forma Normal de Proyección-Unión Extra Fuerte y Forma Normal de Clave de Dominio. Estas formas de normalización pueden llevar a situaciones más allá de lo que se necesita. Éstas existen para hacer una base de datos realmente relacional. Tienen que ver principalmente con dependencias múltiples y claves relacionales.

2.3 Base de datos distribuida.

Conceptos generales.

Una Base de Datos Distribuida (BDD) es un colección de varias Bases de Datos interrelacionadas lógicamente y colocadas sobre una red de computadoras; donde son 2 los rasgos principales en una BDD. Primero la integración de los datos, es decir, los datos no son una colección de archivos colocados individualmente en los nodos de una red, sino que forman parte de una estructura global que los relaciona lógicamente. Después la transparencia, es decir, que los datos que se encuentran en una red de computadoras parecerán para los usuarios de la BDD como datos homogéneos y únicos.

CARACTERÍSTICAS DE UNA BASE DE DATOS DISTRIBUÍDA.

- Los datos deben estar físicamente en más de una computadora.
- Los nodos deben estar interconectados en una red.
- Los datos estarán lógicamente colocados en una estructura única o esquema global.
- Los usuarios deberán tener acceso a los datos aunque no se encuentren en la interfaz del mismo.
- Cada nodo representará un entorno para la ejecución de transacciones.
- No será necesario que el usuario sepa que los datos se encuentran distribuidos en varios nodos.

SISTEMA GESTOR DE UNA BASE DE DATOS DISTRIBUÍDA. (SGBDD)

Es el software que permite gestionar los datos y hace la distribución homogénea para los usuarios y cumple las mismas funciones y propósitos que un Sistema Gestor de Base de Datos de una Base de Datos centralizada. Se compone principalmente de 4 componentes:

- **Procesador de datos locales.** Se encarga del manejo de datos, concurrencia y atención a fallos a nivel local.
- **Diccionario o directorio global.** Donde se guarda la información de dónde y cómo se almacenan los datos en la red, el modo de acceso y otras características físicas.

- **Procesador de aplicaciones distribuidas.** El encargado de acceder a la información de otro Godoy se ocupa de procesar las acciones que involucren a más de una sede.
- **Software y red de comunicaciones.** Provee al SGBDD de aplicaciones primitivas para la correcta intercomunicación de las sedes.

CLASES Y ARQUITECTURAS DE UN SGBDD

Dependiendo de la homogeneidad (Que sean iguales entre sí) de los SGBD locales, de la distribución de los datos y de su autonomía (Referido a la distribución del control) tendremos distintos tipos de SGBDD.

- De arquitectura ANSI/X3/SPARC.
- De arquitectura Multibase.

Diseño de la base de datos distribuida.

En cualquier proceso de diseño hay dos aproximaciones básicas, la ascendente o Top-Down y la descendente Bottom-Up

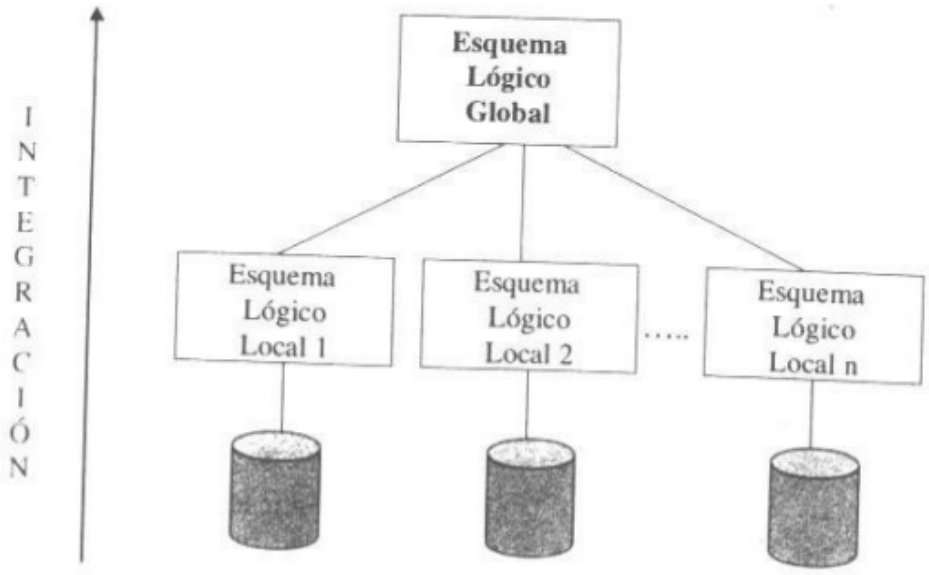


Figura 2.3.2.1 Metodología ascendente del diseño de una BDD.

La metodología ascendente se utiliza principalmente para el caso en el que existen varias BD locales y se desea construir una BDD. Por lo que se parte de varios esquemas locales y mediante la integración de parte de ellos o de todo en un único Esquema Lógico Global.

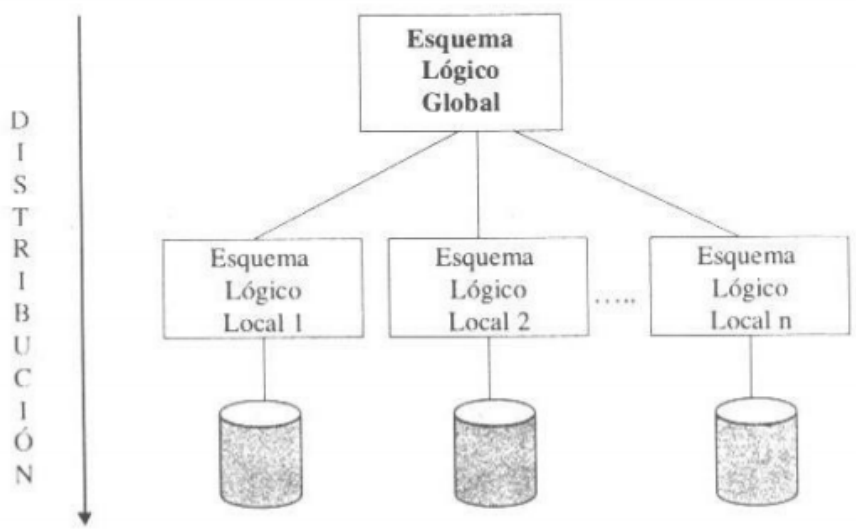


Figura 2.3.2.2 Metodología descendente del diseño de una BDD.

La metodología descendente parte de un Esquema Lógico Global que mediante la fragmentación, replicación y asignación de los objetos pertenecientes a éste constituyen los Esquemas Lógicos Locales.

ESQUEMA DE FRAGMENTACIÓN.

El esquema de fragmentación se compone de las distintas relaciones en las que se divide una relación proveniente del Esquema Lógico Global junto con la condición empleada para dicha división demostrándose con el álgebra relacional.
Para asegurar que la BDD no sufrirá de cambios semánticos durante la fragmentación, se tendrán que cumplir las siguientes condiciones.

- **Completitud.** Todos los datos se encontrarán en al menos un fragmento.
- **Disyunción.** Los fragmentos deberán se disjuntos.
- **Reconstrucción.** Siempre se podrá reconstruir la base global a partir de los fragmentos.

Existen distintas formas de dividir una relación, con fragmentación vertical, horizontal y mixta.

- **Fragmentación vertical.** Divide la relación R en conjuntos de columnas, así cada fragmento mantiene ciertos atributos de la relación original.
- **Fragmentación horizontal.** Divide la relación en un conjunto de tuplas, cada una de ellas con un significado lógico.
- **Fragmentación mixta.** Cuando los esquemas anteriores no son suficientes para las necesidades del cliente, se necesita fragmentar nuevamente los fragmentos obtenidos.

ESQUEMA DE ASIGNACIÓN Y REPLICACIÓN.

El esquema de asignación y replicación consiste en la realización de la correspondencia entre los fragmentos y los nodos que constituyen la red de comunicaciones de la BDD. Dicha

correspondencia debe hacerse de la manera más óptima tomando en cuenta los parámetros siguientes:

- Mínimo coste.
- Rendimiento.

Administración de base de datos distribuidas.

CONSULTA REMOTA.

Para consultar una base de datos remota, se debe de crear un enlace de base de datos (database link) en la base de datos en la que se origine la consulta.
El enlace de base de datos especifica el nombre que se va a utilizar y puede especificar también el nombre del usuario que se va a conectar a la base de datos remota. Se puede configurar el enlace de base de datos para que utilice el nombre de usuario y la contraseña locales en la base de datos remota.

REPLICACIÓN DINÁMICA.

Se pueden emplear disparadores (triggers) de base de datos para duplicar datos de una tabla en otra. Por ejemplo, después de hacer una inserción (**insert**) en una tabla, un disparador puede activar la inserción del mismo registro en otra tabla, pudiendo estar la tabla en una base de datos remota. Por tanto, los disparadores se pueden utilizar para forzar la duplicación de datos en configuraciones sencillas.

Cuando se usan las funciones de procesamiento distribuido de Oracle, se pueden emplear vistas materializadas para duplicar datos entre bases de datos, excepto los datos de tipo LONG, LONG RAW o tipos abstractos de datos.

TRANSPARENCIA DE LA UBICACIÓN.

En las bases de datos distribuidas, debe identificarse:

- El nombre de la instancia que accede a la base de datos.
- El nombre del nodo en el que reside la instancia.

El objetivo de la transparencia de ubicación es hacer que las 3 primeras partes que componen el **nombre global de objeto** sean transparentes al usuario. Estas tres partes se especifican mediante enlaces de base de datos.

- Servidor (host),
- Base de datos (instancia).
- Usuario (esquema).

DOMINIOS.

Un servicio de nombres de dominio (Domain Name Service o DNS) permite la organización jerárquica de los nodos de una red. Los nodos de la organización se conocen como dominios y cada uno de estos dominios se etiqueta según la función que desempeñe.
Estas funciones pueden incluir:

- COM para compañías y
- EDU para escuelas.

Cada dominio puede tener múltiples subdominios, donde los nombres de nodos pueden constar de hasta 4 partes: la parte izquierda es el nombre del nodo y el resto representa el dominio al que pertenece el nodo.

Por ejemplo HQ.MYCORP.COM.

- ✓ El nombre del host es HQ.
- ✓ Se identifica como parte del subdominio MYCORP del dominio COM.

ENLACES COMPARTIDOS.

Si se producen múltiples accesos concurrentes a una base de datos remota a través de un enlace de bases de datos, puede utilizar un enlace compartido de base de datos con la finalidad de reducir el número de conexiones de servidor. En el mejor de los casos, se emplean enlaces de usuarios conectados junto con nombres de servicio o alias.

TRANSACCIONES DISTRIBUIDAS.

Una unidad lógica de trabajo puede incluir transacciones sobre múltiples bases de datos.
Por ejemplo se envía un comando commit después de haberse actualizado (update) dos tablas almacenadas en bases de datos diferentes. Esto se lleva acabo de forma automática mediante confirmación en dos fases.
En la primera fase, la fase de preparación, cada nodo implicado en una transacción prepara los datos que se necesitarán para confirmar o anular los datos.
Una vez preparados los datos, se dice que el nodo está en estado de duda. Los nodos notifican su estado al nodo que se inicia la transacción (conocido como el coordinador global).
Una vez que todos los nodos están preparados, la transacción entra en la fase de confirmación.
Las bases de datos confirman los datos al mismo tiempo, conservando la integridad de los datos distribuidos.

SUPERVISIÓN DE BASES DE DATOS DISTRIBUIDAS.

Existen otras medidas de rendimiento importantes que hay que tener en cuenta para las bases de datos.

- El rendimiento del host.
- La distribución de la carga de E/S entre los discos y controladores.
- El uso de memoria disponible.

Para bases de datos distribuidas también se debe de tener en cuenta lo siguiente:

- La capacidad de la red y su hardware.
- La carga en los segmentos de red.
- El uso de diferentes rutas de acceso físicas entre máquinas host.

2.4 HTML, CSS Y Javascript.

Elementos principales de diseño.

Cuando hablamos sobre diseño de páginas Web, realmente nos referimos al HTML o Hyper-text Markup Language. El lenguaje HTML consiste en una serie de comandos que le indican al programa navegador de WWW cómo darle formato al texto que contienen los archivos. En la actualidad no hace falta ser un experto en HTML debido a que los procesadores de texto existentes se encargan de hacer el trabajo, agregando el código automáticamente, a lo que queremos mostrar.

Los servicios de diseño de páginas Web cubren un rango que va desde convertir archivos de texto a formato HTML, hasta desarrollar extensos conjuntos de páginas cargadas de ilustraciones e hipervínculos relacionados entre sí. Podría dársele un enfoque especial a su página para algún programa navegador específico, pero siempre debe asegurarse de que las páginas sean vistas por el mayor número de plataformas posibles.

La tarea de diseño de páginas Web convoca por lo menos tres especialidades diferentes: comunicación (humana), programación (HTML, CGI, Java), diseño gráfico y multimedia.

HTML es el lenguaje con el que se construyen las páginas Web. La idea inicial de HTML era describir la estructura y el contenido de un documento, sin embargo la tendencia actual es utilizarlo también como un lenguaje de descripción, controlando el aspecto de documento (tipografía, posicionado, etc.).

Por supuesto, la estética de los documentos escritos en HTML no se limita a texto digamos normal; consigue todos los efectos que habitualmente se pueden producir con un moderno procesador de textos: negrita, cursiva, distintos tamaños y fuentes, tablas, párrafos tabulados, sangrías, incluso texto y fondo de página de colores, y muchos más.

El programa encargado de interpretar el texto HTML es el navegador o browser. El navegador puede recibir el código HTML junto con los elementos integrados en la página (imágenes, sonidos, vídeo, etc.) desde un servidor remoto o de un servidor de red (utilizando el protocolo de transferencia de hypertexto y HTTP) o leer las páginas directamente de nuestro disco duro(sin un protocolo de transmisión tipo HTTP, sino el equivalente a abrir un documento con un procesador de textos).

En HTML todas las codificaciones de efectos en el texto que lo forman no son más que instrucciones para el visualizador. Partiendo de esto, se entiende el porqué no se ve lo mismo con todos los visualizadores. Depende de cómo estén diseñados y para qué versión de lenguaje estén diseñados.

Hasta no hace mucho los programadores de HTML cobraban una barbaridad por crear una página Web. Eran los inicios del boom WWW.

HTML no es un lenguaje de programación como puede serlo C, Pascal o Java; HTML tan sólo es un lenguaje para crear documentos en formato electrónico, una forma de definir efectos en el texto de manera similar a como se hacía en los antiguos procesadores de texto pero con complicados y poderosos servidores de información.

Esqueleto Básico de la estructura HTML:
Entre <html> y </html> encontraremos la definición de la página propiamente dicha. En el bloque delimitado por <head> y </head> se establecen ciertas características de la página, tales como el título, quien las construyó, etc. De estas características de la página, la única que es obligatoria declarar es el título. Esto se hace mediante el par de tags <title> y </title>.

Por último está <body> y </body>, entre los cuales se encierra toda la información que el navegador debe mostrar.

No todos los tags son iguales; hay alguno cuyas acciones están acotadas por las funciones que cumplen, por lo que no es necesario incluir otro tag para finalizar su acción. Algunos de ellos son:

-
 Genera un retorno de carro
- <p> Equivale a un retorno de carro + un avance de línea
- <hr> Crea una línea divisoria horizontal.

Existen tags que llevan parámetros asociados cómo:
<img src="/cm/marzo.gif" width=104 height=97>

Este tag <img> permite incluir imágenes dentro de una página. El parámetro src indica la ruta de acceso al archivo donde está la imagen, mientras que width y height detallan su ancho y alto en pixeles.

Los códigos principales son los que provocan salto de línea y el que se centra.

- El código <P> señala el inicio de un párrafo y provoca un salto de línea precedido por un renglón en blanco.
- El código
 hace lo mismo pero sin renglón en blanco.
- Los códigos y centran el texto entre los márgenes.

Listas.
Hay varias maneras de tratar listas. Las principales son la lista numerada (OL) y de los puntos conductores (UL) que tiene un par de variantes. También hay una lista pensada para glosarios de términos (DL).

Tablas.
Las tablas nos permiten distribuir las cosas en columnas y en filas, aprovechando mejor el ancho de página. Se puede especificar un montón de parámetros de formato tanto de tabla (TABLE) como de filas (TR) y las celdas (TD) individuales.

Vínculos.

Un vínculo es un texto, botón o imagen que al seleccionar nos lleva a otra dirección URL, página WEB o recurso.

La etiqueta <A> que viene de “ancla”, denota el inicio y el final de una instrucción que contiene alguna forma de vínculo o hipervínculo. Esta etiqueta permite al usuario vincularse a otra ubicación dentro del mismo documento HTML, a otro sitio WEB, a un servidor FTP, enlace de correo electrónico, etc.

Los atributos de esta etiqueta son:

- HREF: Recurso al cual se hace referencia el hipervínculo.
- NAME: Especifica el nombre de la posición a donde apuntar.
- TARGET: Destino del enlace (generalmente para páginas con Frames).

Introducción a css.

Cascading Style Sheets u Hojas de Estilo en Cascada, es la tecnología desarrollada por el World Wide Web Consortium (W3C) con el fin de separar la estructura de la presentación.

CSS es un lenguaje de estilo que define la presentación de los documentos HTML. Por ejemplo, CSS abarca cuestiones relativas a fuentes, colores, márgenes, líneas, altura, anchura, imágenes de fondo, posicionamiento avanzado y muchos otros temas.

Es posible usar HTML, o incluso abusar del mismo, para añadir formato a los sitios web. Sin embargo, CSS ofrece más opciones y es más preciso y sofisticado. CSS está soportado por todos los navegadores hoy día.

CSS proporciona tres caminos diferentes para aplicar las reglas de estilo a una página Web:

1. Una hoja de estilo externa, que es una hoja de estilo que está almacenada en un archivo diferente al archivo donde se almacena el código HTML de la página Web. Esta es la manera de programar más potente, porque separa completamente las reglas de formateo para la página HTML de la estructura básica de la página.
2. Una hoja de estilo interna, que es una hoja de estilo que está incrustada dentro de un documento HTML. (Va a la derecha dentro del elemento <head>). De esta manera se obtiene el beneficio de separar la información del estilo, del código HTML propiamente dicho. Se puede optar por copiar la hoja de estilo incrustada de una página a otra, (esta posibilidad es difícil de ejecutar si se desea para guardar las copias sincronizadas). En general, la única vez que se usa una hoja de estilo interna, es cuando se quiere proporcionar alguna característica a una página Web en un simple fichero, por ejemplo, si se está enviando algo a la página web.
3. Un estilo en línea (inline), que es un método para insertar el lenguaje de estilo de página, directamente, dentro de una etiqueta HTML. Esta manera de proceder no es totalmente adecuada. El incrustar la descripción del formateo dentro del documento de la página Web, a nivel de código se convierte en una tarea larga, tediosa y poco elegante de resolver el problema de la programación de la página. Este modo de trabajo se podría usar de manera ocasional si se pretende aplicar un formateo con prisa, al vuelo. No es todo lo claro, o estructurado, que debería ser, pero funciona. Este es el método recomendado para maquetar correos electrónicos en HTML.

Las ventajas de utilizar CSS (u otro lenguaje de estilo) son:

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.
- Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web, con lo que aumenta considerablemente la accesibilidad. Por ejemplo, personas con deficiencias visuales pueden configurar su propia hoja de estilo para aumentar el tamaño del texto o remarcar más los enlaces.
- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario. Por ejemplo, para ser impresa, mostrada en un dispositivo móvil, o ser "leída" por un sintetizador de voz.
- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño (siempre y cuando no se utilice estilo en línea).

Introducción a javascript.

Javascript es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web. Es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con Javascript podemos crear diferentes efectos e interactuar con nuestros usuarios.

Este lenguaje posee varias características, entre ellas podemos mencionar que es un lenguaje basado en acciones que posee menos restricciones. Además, es un lenguaje que utiliza Windows y sistemas X-Windows, gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas entre otros.

Es necesario resaltar que hay dos tipos de JavaScript: por un lado está el que se ejecuta en el cliente, este es el Javascript propiamente dicho, aunque técnicamente se denomina Navigator JavaScript. Pero también existe un Javascript que se ejecuta en el servidor, es más reciente y se denomina LiveWire Javascript.

Javascript nació con la necesidad de permitir a los autores de sitio web crear páginas que permitan intercambiar con los usuarios, ya que se necesitaba crear webs de mayor complejidad. El HTML solo permitía crear páginas estáticas donde se podía mostrar textos con estilos, pero se necesitaba interactuar con los usuarios.

Entre los diferentes servicios que se encuentran realizados con Javascript en Internet se encuentran:

- Correo
- Chat
- Buscadores de Información
- Reloj
- Contadores de visitas
- Fechas
- Calculadoras
- Validadores de formularios
- Detectores de navegadores e idiomas

Algunas ventajas que ofrece Javascript es que no requiere un tiempo de compilación, los scripts se pueden desarrollar en un periodo de tiempo relativamente corto. A esto podemos añadirle las características de interfaz como, por ejemplo, cuadro de diálogo, formularios y otros elementos GUI (Interfaz Gráfico de Usuario), son gestionados por el navegador y por el código HTML. Por lo tanto los programadores que utilizan JavaScript no se deben preocupar en crear o controlar dichos elementos en sus aplicaciones.

Aunque JavaScript tiene muchas similitudes con Java, no incluye la sintaxis y reglas complejas de Java.

Como WWW es independiente de la plataforma hardware o sistema operativo, los programas escritos en Javascript también lo son, siempre y cuando exista un navegador con soporte JavaScript para la plataforma en cuestión.

Los programas JavaSript tienden a ser pequeños y compactos (en comparación con los applets de Java), no requieren mucha memoria ni tiempo adicional de transmisión. Además, al incluirse dentro de las mismas páginas HTML se reduce el número de accesos independientes a la red.

2.5 Java y Java Server Pages.

Tecnología java.

Java es un lenguaje de programación con el que podemos realizar cualquier tipo de programa. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general.

Actualmente Java se utiliza en un amplio abanico de posibilidades y casi cualquier cosa que se puede hacer en cualquier lenguaje se puede hacer también en Java y muchas veces con grandes ventajas. Para lo que nos interesa a nosotros, con Java podemos programar páginas web dinámicas, con accesos a bases de datos, con cualquier tipo de conexión de red entre cualquier sistema. En general, cualquier aplicación que deseemos hacer con acceso a través web se puede hacer utilizando Java.

Para garantizar que los programas son independientes de la plataforma (capacidad del programa de trasladarse con facilidad de un sistema computacional a otro) hay una sola arquitectura a la que todos los programas Java son compilados, es decir, cuando se compila un programa Java en una plataforma Windows/Intel, se obtiene la misma salida compilada que en un sistema Macintosh o Unix. El compilador compila no a una plataforma determinada, sino a una plataforma abstracta llamada Máquina Virtual de Java.

La especificación de la Máquina Virtual de Java define la JVM como una máquina imaginaria que se implementa emulando por software una máquina real. El código para la Máquina Virtual de Java se almacena en ficheros .class, cada uno de los cuales contiene al menos el código de una clase pública.

Alcances de la tecnología java.

APLICACIONES DE CONSOLA.
Son programas independientes al igual que los creados con los lenguajes tradicionales. Consta de la utilización de las librerías más básicas de java y se ejecutan desde línea de comandos.

APLICACIONES GRÁFICAS.
Aquellas que utilizan las clases con capacidades gráficas (como awt o swing por ejemplo). Este tipo de aplicaciones son comúnmente utilizadas para ejecutarse en una máquina en particular y tienen la peculiaridad de ser instaladas o ejecutadas desde un archivo jar.

APPLETS.
Son programas Java pensados para ser colocados dentro de una página web. Pueden ser interpretados por cualquier navegador con capacidades Java. Son programas que necesitan descargados a la máquina cliente por lo que consumen recursos de la misma.

SERVLETS.
Son aplicaciones que se ejecutan en un servidor de aplicaciones web y que como resultado de su ejecución resulta una página web. La aplicación más utilizada hoy en día dado que se ejecuta en la maquina que funje como servidor, permitiendo centralizar la información para pronto y más adecuado manejo.

Caracteristicas principales de java.

SIMPLE:
Java elimina muchas de las características de otros lenguajes como C++, para mantener reducidas las especificaciones del lenguaje y añadir características muy útiles como el garbage collector (reciclador de memoria dinámica). No es necesario preocuparse de liberar memoria, el reciclador se encarga de ello y como es un thread de baja prioridad, cuando entra en acción, permite liberar bloques de memoria muy grandes, lo que reduce la fragmentación de la memoria.

Java reduce en un 50% los errores más comunes de programación con lenguajes como C y C++ al eliminar muchas de las características de éstos, entre las que destacan:

- aritmética de punteros
- no existen referencias
- registros (struct)
- definición de tipos (typedef)
- macros (#define)
- necesidad de liberar memoria (free)

ORIENTADO A OBJETOS:
Java implementa la tecnología básica de C++ con algunas mejoras y elimina algunas cosas para mantener el objetivo de la simplicidad del lenguaje. Java trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo. Las plantillas de objetos son llamadas, como en C++, clases y sus copias, instancias. Estas instancias, como en C++, necesitan ser construidas y destruidas en espacios de memoria.

DISTRIBUIDO:
Java se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como http y ftp. Esto permite a los programadores acceder a la información a través de la red con tanta facilidad como a los ficheros locales.
La verdad es que Java en sí no es distribuido, sino que proporciona las librerías y herramientas para que los programas puedan ser distribuidos, es decir, que se corran en varias máquinas, interactuando.

ROBUSTO:
Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo. Java obliga a la declaración explícita de métodos, reduciendo así las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de memoria.

SEGURO:
La seguridad en Java tiene dos facetas. En el lenguaje, características como los punteros o el casting implícito que hacen los compiladores de C y C++ se eliminan para prevenir el acceso ilegal

a la memoria. Cuando se usa Java para crear un navegador, se combinan las características del lenguaje con protecciones de sentido común aplicadas al propio navegador.

PORTABLE:
Más allá de la portabilidad básica por ser de arquitectura independiente, Java implementa otros estándares de portabilidad para facilitar el desarrollo. Los enteros son siempre enteros y además, enteros de 32 bits en complemento a 2. Además, Java construye sus interfaces de usuario a través de un sistema abstracto de ventanas de forma que las ventanas puedan ser implantadas en entornos Unix, Pc o Mac.

INTERPRETADO:
El intérprete Java (sistema run-time) puede ejecutar directamente el código objeto. Enlazar un programa, normalmente, consume menos recursos que compilarlo, por lo que los desarrolladores con Java pasarán más tiempo desarrollando y menos esperando por el ordenador. No obstante, el compilador actual del JDK es bastante lento. Por ahora, que todavía no hay compiladores específicos de Java para las diversas plataformas, Java es más lento que otros lenguajes de programación, como C++, ya que debe ser interpretado y no ejecutado como sucede en cualquier programa tradicional.

JDK Y JSE6.

JDK es el acrónimo de "Java Development Kit", es decir Kit de desarrollo de Java. Se puede definir como un conjunto de herramientas, utilidades, documentación y ejemplos para desarrollar aplicaciones Java.

JDK consta de una serie de aplicaciones y componentes, para realizar cada una de las tareas de las que es capaz de encargarse.

- Intérprete en tiempo de ejecución (JRE): Permite la ejecución de los programas Java (*.class) no gráficos (aplicaciones).
- Compilador: Se utiliza para compilar archivos de código fuente Java (habitualmente *.java), en archivos de clases Java ejecutables (*.class). Se crea un archivo de clase para cada clase definida en un archivo fuente.
- Visualizador de applets: Es una herramienta que sirve como campo de pruebas de applets, visualizando cómo se mostrarían en un navegador, en lugar de tener que esperar. Al ser activado desde una línea de órdenes abre una ventana en la que muestra el contenido de la applet.
- Depurador: Es una utilidad de línea de comandos que permite depurar aplicaciones Java. No es un entorno de características visuales, pero permite encontrar y eliminar los errores de los programas Java con mucha exactitud. Es parecido en su funcionamiento al depurador gdb que se incluye con las distribuciones del compilador gcc/g++ para C/C++.
- Desensamblador de archivo de clase: Se utiliza para desensamblar un archivo de clase. Su salida por defecto, muestra los atributos y métodos públicos de la clase desensamblada, pero con la opción -c también desensambla los códigos de byte, mostrándolos por pantalla. Es útil cuando no se tiene el código fuente de una clase de la que se quisiera saber cómo fue codificada.
- Generador de cabecera y archivo de apéndice: Se utiliza para generar archivos fuentes y cabeceras C para implementar métodos Java en C (código nativo). Esto se consigue mediante la generación de una estructura C cuya distribución coincide con la de la correspondiente clase Java.

El generador de cabeceras java, crea los ficheros de cabecera C/C++ para implementar en esos lenguajes los métodos nativos que presente un programa Java.

- Generador de documentación: Es una herramienta útil para la generación de documentación API directamente desde el código fuente Java. Genera páginas HTML basadas en las declaraciones y comentarios javadoc, con el formato /** comentarios */:

  ```
  /** Comentarios sobre la clase
  @autor: Ignacio Cruzado
  */
  class MiClase {
  }
  ```

  La documentación que genera es del mismo estilo que la documentación que se obtiene con el JDK.

Java Platform, Standard Edition o Java SE (conocido anteriormente hasta la versión 5.0 como Plataforma Java 2, Standard Edition o J2SE), es una colección de APIs del lenguaje de programación Java útiles para muchos programas de la Plataforma Java.

Comenzando con la versión J2SE 1.4 (Merlin), la plataforma Java SE ha sido desarrollada bajo la supervisión del Java Community Process. JSR 59 la especificación para J2SE 1.4 y JSR 176 especificó J2SE 5.0 (Tiger). En 2006, Java SE 6 (Mustang) está siendo desarrollada bajo el JSR 270.

Entre las novedades de Java 6 destacan los soportes para lenguajes dinámicos y de scripting, como pueden ser Netscape's Rhino, un motor Javascript, awk, Jelly, Pnuts, Python, Ruby y Scheme. Además se ha trabajado en la mejora de las librerías y del compilador en tiempo de ejecución, y el soporte para Windows Vista.

Entre sus mejoras encontramos:

- Mejora en el soporte de WebServices. Nuevo core JAX-WS 2.0 API y soporte para XML Binding (JAXB) 2.0.
- Soporte de lenguajes de script (JavaScript, Phyton, AWK, Ruby,...). Incluye el motor Mozilla Rhino.
- Bases de datos, actualización al estandar JDBC 4.0 e inclusión de Java DB (que es una base de datos Java del Apache Derby).
- Nuevas APIs de escritorio
- Mejoras en la monitorización y gestionabilidad.
- Acceso programático al compilador.
- Capacidad de definir anotaciones y poder enchufarlas en el código para procesarlo.
- Despliegue en escritorio. Mejoras en la instación, integración con Windows Vista.
- Seguridad. Integración con PKI, Java GSS, Kerberos.
- Rendimiento. Mejoras en el rendimiento del doble dígito.

Java server pages.

Instalación y configuración del servidor Apache Tomcat.

- Bajar la versión binaria de Tomcat en:
  http://tomcat.apache.org/
- Dar clic en el icono de Apache-Tomcat para iniciar la instalación.
- Dar siguiente y Aceptar el contrato.
- En la siguiente pantalla seleccionar todas las opciones.

Se nos muestra una pantalla que nos pide el puerto que se utilizará, por default dejamos el 8080 y elegimos un nombre y contraseña para el administrador.

Por último debemos indicar la ruta donde se instalo el JRE, lo cual se hizo cuando se instalo el JDK.

Estructura de Directorios para Tomcat
La instalación de Tomcat crea una estructura de directorios de la siguiente forma:

- Tomcat
  - bin
  - common
  - conf
  - webapps
  - work

De forma general el contenido de los directorios es el siguiente:

bin: Contiene los archivos necesarios para hacer funcionar el Servidor Apache Tomcat.

Common: Contiene el subdirectorio lib en el cual se localizan los archivos JAR, estos contienen clases que dan soporte a todas las palicaciones web que se desarrollen como Tomcat.

conf : Se localizan archivos de configuración para Tomcat tales como Server.xml.
webapps: Este es el directorio en donde deben colocarse todas las aplicaciones web que desarrollemos. En este directorio podremos crear todas las carpetas (subdirectorios) que necesitemos, con el fin de tener mejor organizados nuestras aplicaciones.

work: Dentro de él se encuentran clases necesarias para el desarrollo de JSP.

PREPARANDO LOS DIRECTORIOS.
Antes de iniciar con nuestras aplicaciones, es necesario que dentro del directorio
webapps (mencionando anteriormente), se agreguen directorios en donde vamos a tener todos los archivos que una aplicación vaya a ejecutar. Por ejemplo, se creará un directorio llamado *SIGEB* y dentro de él es necesario crear un directorio *WEB-INF* y dentro de este último, un subdirectorio el cual llamaremos *classes*, por lo que la estructura de directorios ahora es:

Tomcat

- bin
- common
- conf
- webapps
  - *SIGEB*
  - *WEB-INF*
  - *classes*
- work

En el subdirectorio *SIGEB*, se colocaran todos los archivos que necesitemos ellos pueden ser jsp, html, imágenes, etc.

El subdirectorio *classes*, contendrá todos los archivos .java y .class que nuestra aplicación web requiera.

EXPRESIONES.
Son fragmentos de código Java, con la forma <%= *expresión %*> que se evalúan y se muestran en la salida del navegador. La expresión se evalúa, su resultado se convierte a String y se inserta en la salida.
Las expresiones son una forma abreviada de utilizar *System.out.println()*, ya que el resultado es el mismo, además las expresiones podemos usarlas como si estuviéramos trabajando con un println() en java, es decir, podemos concatenar tantos elementos como queramos.

SCRIPTLETS.
Son fragmentos de código Java con la forma <% código %>, en general, podemos insertar cualquier código que pudiéramos usar dentro de una función Java. Permiten ejecutar código arbitrario, cuyo resultado no es necesario enviar a la salida. Para acceder a la salida del navegador, usamos el objeto implícito out.
Un uso común de los *scriptlets* es hacer que ciertas partes de código HTML aparezcan, o no en función de una condición.
Recordemos que las paginas JSP son una extensión de java, por lo cual podemos usar todo lo que la API de java nos ofrece.
De la misma manera como se puede hacer con las expresiones, el código HTML lo podemos combinar, viendo de esta manera la gran potencia de las JSP.

DECLARACIONES.
Permiten definir variables o métodos que se insertarán dentro del cuerpo del servlet generado.
Las variables declaradas conservarán su valor entre sucesivas llamadas a la página, ya que son variables miembro del servlet y no locales al método jspService.
Sintaxis: <%!declaracion;[*otra declaracion*;]+…%>

ACCIONES.
Las acciones tienen la forma <jsp:accion [parámetros]/>, y tienen diversos usos, entre los que destacan la inclusión de páginas y transferencia de control.

- *Inclusión de páginas*: Se realiza con la acción <jsp:include page="pagina.jsp">. Incluye la salida de otra página JSP en la actual, al contrario que con la directiva <%@include file="fichero.ext"%> la página incluida se ejecuta y su salida se inserta en la página que la incluye, con la directiva se incluye el contenido del archivo (no su salida) y se ejecuta conjuntamente con la página principal. La página incluída tiene acceso a los parámetros

enviados a la principal, y podemos enviarle nuevos parámetros con la subetiqueta <jsp:param name="nombre" value="valor"/>.

- *Transferencia de control*: Se realiza con la acción < jsp: forward page="pagina.jsp" />. La petición es redirigida a otra página, y la salida de la actual se descarta. Al igual que con la inclusión, la página a la que se redirige tiene acceso a los parámetros pasados a la actual, y es posible el envío de nuevos parámetros.

DIRECTIVAS.
Las directivas influyen en la estructura que tendrá el servlet generado a partir de la página JSP. Hay tres tipos de directivas, *page, taglib* e *include*.

page: Se indica con la forma <%@ *page atributo="valor"*>.Tiene diversos usos, entre los cuales destacaremos:

- extends=*"package.class".*

  Este atributo especifica un nombre totalmente cualificado de una superclase que será extendida por la clase Java en el fichero JSP.

- import= "{ *package.class* | *package.** }, ...".

  Especifica una lista separada por comas de uno o más paquetes o clases que el fichero JSP debería importar. Las clases de los paquetes se ponen a disposición de los scriptlets, expresiones, declaraciones y etiquetas dentro del fichero JSP.

- session=*"true*|false"

  Todo cliente debe unirse a una sesión HTTP para poder usar una página JSP. Si el valor es true, el objeto session se refiere a la sesión actual o a una nueva sesión. Si el valor es false, no podemos utilizar el objeto session en el fichero JSP. El valor por defecto es true.

include:
Se utiliza de esta manera <%@ include file="fichero" %>. Permite incluir un archivo en el lugar donde se especifique, permite insertar código en la página antes de que ésta se transforme en un servlet. De este modo se pueden reutilizar fragmentos de código JSP o HTML.

MANEJO DE SESIONES.
Una sesión es una serie de comunicaciones entre un cliente y un servidor en la que se realiza un intercambio de información. Por medio de una sesión se puede hacer un seguimiento de un usuario a través de la aplicación.

El tiempo de vida de una sesión comienza cuando un usuario se conecta por primera vez a un sitio web pero su finalización puede estar relacionada con tres circunstancias:

- Cuando se abandona el sitio web.
- Cuando se alcanza un tiempo de inactividad que es previamente establecido, en este caso la sesión es automáticamente eliminada. Si el usuario siguiera navegando se crearía una nueva sesión.
- Se ha cerrado o reiniciado el servidor.

Una posible aplicación de las sesiones es en el comercio electrónico. En este caso una sesión permite ir eligiendo una serie de productos e irlos añadiendo a nuestro “carrito” y así hasta finalizar la compra. También se utilizan para la identificación de usuarios, en la que se deben de introducir un *login* y un *password*. Después de haber hecho esto el usuario tendrá una serie de permisos sobre las páginas que va a visitar, de tal forma que si un usuario intenta pasar a una página si

haberse identificado, el sistema comprobará que no se ha identificado y sería redireccionado a la página de identificación.

Para poder hacer uso de las sesiones en JSP hay que poner el atributo *session* de la directiva *page* a *true*, de esta forma se notifica al contenedor que la página interviene en un proceso que utiliza las sesiones del protocolo HTTP:
<%@page session='true'%>.

En JSP las acciones que se pueden realizar sobre las sesiones se llevan a cabo mediante la interface HttpSession y los métodos que implementa. Esta interfaz está incluida dentro del paquete javax.servlet.http y es utilizada por el contenedor de páginas JSP para crear una sesión entre el servidor y el cliente.
Para obtener la sesión de un usuario se utiliza el método getSession() que devuelve una interfaz de tipo HttpSession. Una vez creado el objeto de tipo sesión es posible acceder a una serie de datos sobre la misma.

Para guardar un objeto en una sesión se utiliza el método setAttribute().Este método utiliza dos argumentos:

- El primero es el nombre que identificará a esa variable.
- El segundo es el dato que se va a guardar.

Para poder recuperar un valor almacenado en un objeto de una sesión se utiliza el método getAttribute(), utilizando como argumento el nombre que identifica al objeto que se quiere recuperar.

CONEXIÓN ENTRE EL SISTEMA Y LA BASE DE DATOS.
JDBC es un API incluido dentro del lenguaje Java para el acceso a bases de datos. Consiste en un conjunto de clases e interfaces escritos en Java que ofrecen un completo API para la programación de bases de datos, por lo tanto es la una solución 100% Java que permite el acceso a bases de datos, la primera aparición de JDBC (JDBC 1.0) se encuentra dentro del paquete java.sql que ha fue incorporado en la versión del JDK 1.1.x (Java Development
Kit) correspondiente a la versión 1.1 del lenguaje Java. Para la versión 1.6 de Java se tiene la versión 4.0 de JDBC.

Funciones.
Básicamente el API JDBC hace posible la realización de las siguientes tareas:

- Establecer una conexión con una base de datos.
- Enviar sentencias SQL.
- Manipular los datos.
- Procesar los resultados de la ejecución de las sentencias.

Un objeto Connection representa una conexión con una base de datos. Una sesión de conexión con una base de datos incluye las sentencias SQL que se ejecuten y los resultados que devuelvan. Una sola aplicación puede tener una o más conexiones con una misma base de datos, o puede tener varias conexiones con diferentes bases de datos. Connection es un interfaz, ya que como ya se había dicho anteriormente, JDBC ofrece una plantilla o especificación que deben implementar los fabricantes de drivers de JDBC.

Una URL de JDBC ofrece un mecanismo para identificar una base de datos de forma que el driver adecuado la reconozca y establezca una conexión con ella. Al igual que las URLs generales sirven para identificar de un modo único un recurso en Internet, en este caso una

URL de JDBC localizará una base de datos determinada. La sintaxis estándar de las URLs de JDBC es la siguiente:

jdbc:<subprotocolo>:<subnombre>

Siempre debe comenzar por jdbc pero es el fabricante del driver quién debe especificar la sintaxis exacta de la URL para su driver JDBC. El subprotocolo suele ser el nombre del driver o del mecanismo de conectividad con la base de datos y el subnombre es una forma de identificar a la base de datos concreta.

La forma común de establecer una conexión con una base de datos es llamar al método getConnection() de la clase DriverManager, a este método se le debe pasar como parámetro la URL de JDBC que identifica a la base de datos con la que queremos realizar la conexión.
La ejecución de este método devolverá un objeto Connection que representará la conexión con la base de datos.

La siguiente imagen nos muestra el proceso en forma sintetizada de hacer una conexión a una base de datos a través de un objeto de la clase DriverManager.

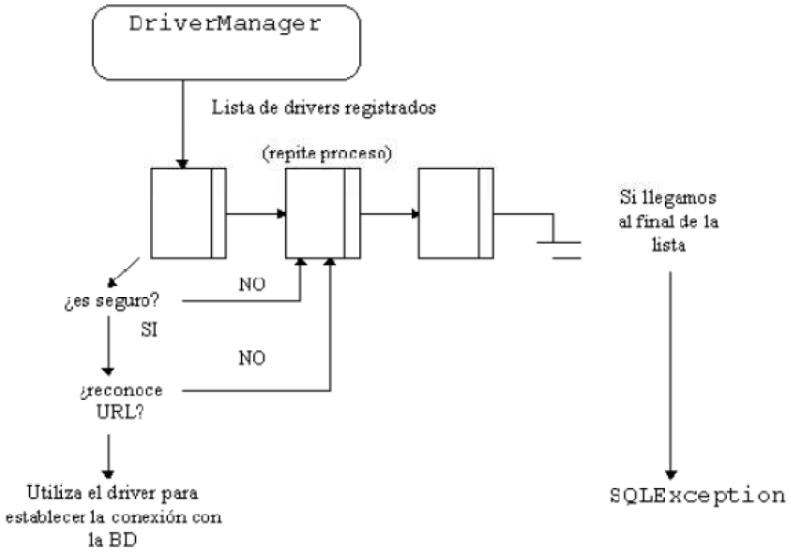


Fig. 2.5.1. Conexión a una base de datos con la clase DriveManager.

2.6 Lenguaje Unificado de Modelado UML.

Conceptos generales.

Como el sucesor de una oleada de métodos de análisis y diseño orientados a objetos (OOA&D) surgido a finales de la década de 1980 y a principios de la década de 1990 unificando los métodos de Booch, Rumbaugh (OMT) y Jacobson.
Pero hay que destacar que como es un lenguaje de modelado, y no un método. El lenguaje de modelado es una notación, principalmente gráfica de donde se valen los métodos para expresar los diseños, y el proceso es la orientación sobre los pasos a seguir para hacer el diseño.

NOTACIONES Y METAMODELOS.

En la condición actual UML define una notación y un metamodelo.
La **notación** es el material gráfico que se ve en los modelos; es la sintaxis del lenguaje de modelado. Por ejemplo, la notación de un diagrama de clases define cómo se representan conceptos y temas como clase, asociación y multiplicidad.

Para algunas personas, las definiciones por sí solas no bastan, llevando a la necesidad de hacer una especificación más rigurosa, independiente de algún lenguaje de programación y utilizando algún método Orientado a Objetos (OO), porque su notación es más intuitiva que formal. Lo anterior nos lleva al uso de un **metamodelo**: un diagrama, usualmente un diagrama de clases que defina la notación.

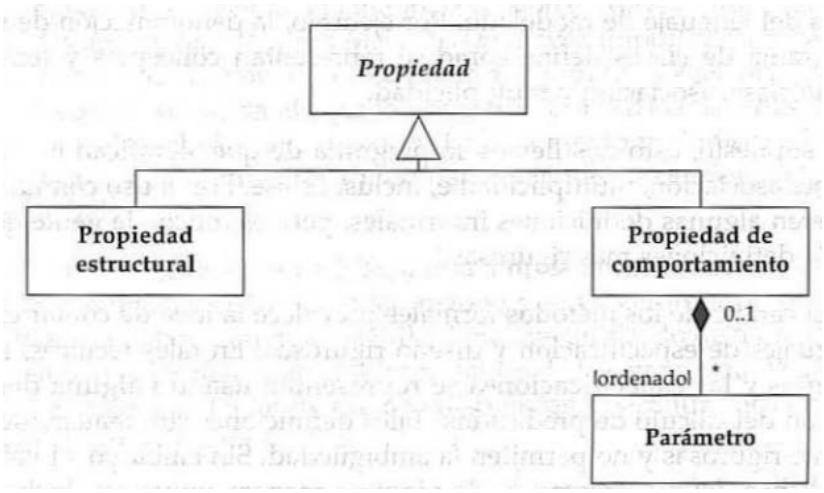


Figura 2.6.1.1 Ejemplo de un metamodelo en UML 1.1

NECESIDAD DE ANALIZAR Y DISEÑAR

- **Aprendizaje del paradigma Orientado a Objetos**: Las técnicas de UML en sí fueron diseñadas para ayudar a los usuarios a hacer un buen desarrollo de OO, como las tarjetas CRC, los diagramas de interacción, los diagramas de clases y el concepto de patrones.
- **Comunicación con los expertos del dominio:** UML permite una buena comunicación, de manera neutral con los clientes que no están familiarizados en el aspecto de desarrollo, pero que necesitan conocer los avances. Por lo que junto a una comprensión adecuada del mundo del usuario, es la clave para el desarrollo de un buen software.

- **Comprensión del panorama general:** Las técnicas de diseño de UML, permiten adquirir una visión completa del sistema en poco tiempo, debido a que uno como consultor o desarrollador no trabaja solamente en un sistema, o desde el comienzo del mismo, y puede resultar fácil perderse cuando se trata de un proyecto grande.

Diagramas utilizados para la elaboración del SIGEB.

Los casos de uso.

Un caso de uso es, en esencia, una interacción típica entre un usuario y un sistema de cómputo, pero que poseen ciertas propiedades como lo son:

- El caso de uso capta alguna función visible para el usuario.
- El caso de uso puede ser pequeño o grande.
- El caso de uso logra un objetivo discreto para el usuario.

Las interacciones con el sistema permiten decir tanto situaciones habituales como “ingresar al sistema” o “cambiar el tamaño de letra”, como objetivos concretos del usuario por ejemplo “garantizar el formateo de un documento” y “garantizar la alta disponibilidad del sistema.”
En cuanto a los diagramas de casos de uso como representación gráfica de los casos de uso definido por Jacobson en 1994 se analizará mediante un ejemplo los componentes del mismo.

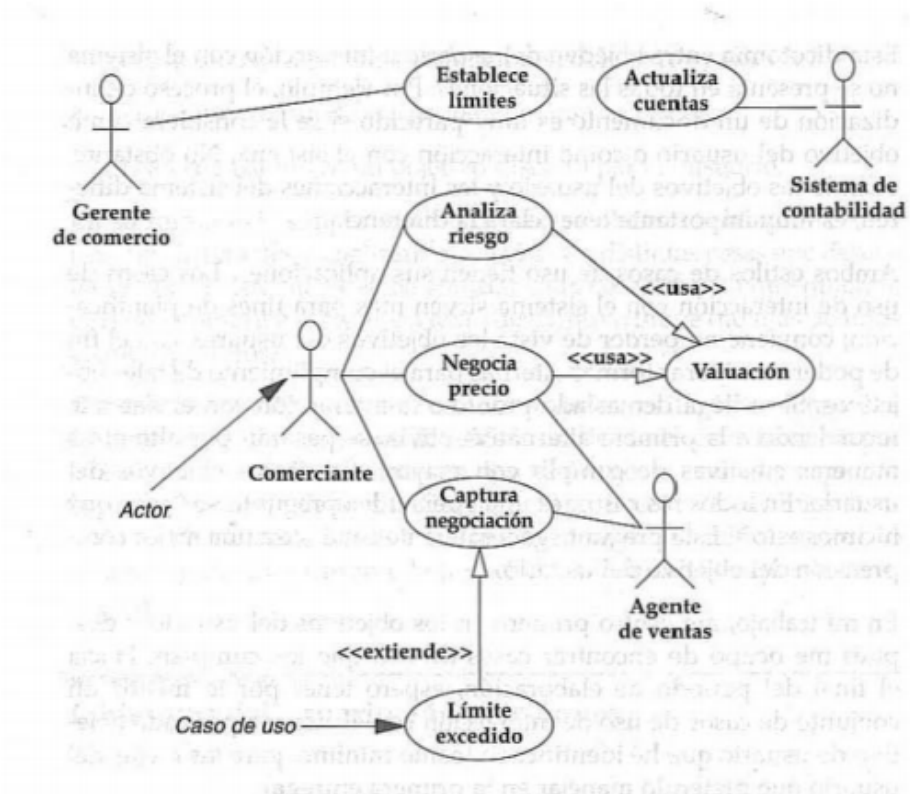


Figura 2.6.2.1 Ejemplo de un diagrama de casos de uso.

- **Actores:** Se utiliza el término actor para llamar así al usuario, cuando desempeña ese papel con respecto al sistema, un actor puede ser también un sistema externo que utilice la información del sistema actual.

- **Uses y extends:** Además entre los vínculos entre los actores y los casos de uso, hay otros tipos de vínculos; la relación extends que se usa cuando se tiene un caso de uso que es similar a otro, pero que hace un poco más, y la relación uses que ocurre cuando se tiene una porción de comportamiento que es similar en más de un caso de uso y no se quiere copiar la descripción de tal caso de uso.

DIAGRAMAS DE CLASES.

El diagrama de clase describe los tipos de objetos que hay en el sistema y las diversas clases de relaciones estáticas que existen entre ellos, encontrando dos tipos fundamentales de relaciones estáticas:

- Asociaciones.
- Subtipos.

También muestran los atributos y operaciones de una clase y las restricciones a las que se ven sujetos, según la forma en la que se conecten los objetos.

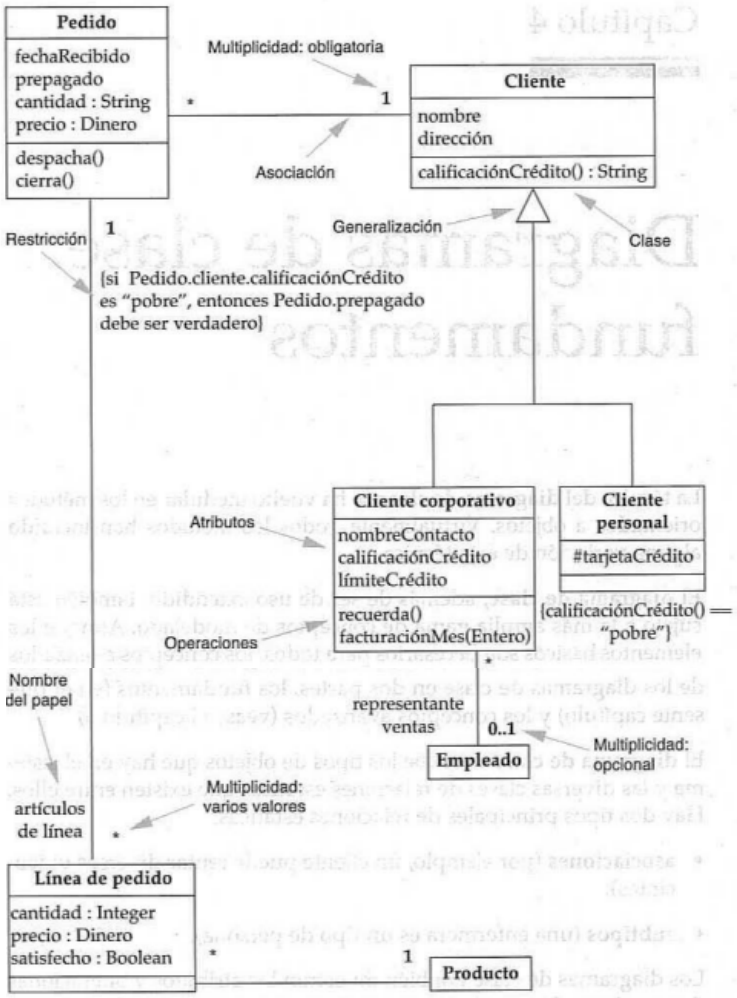


Figura 2.6.2.2 Ejemplo de un diagrama de clases

Y al hacerlo independiente a algún lenguaje de programación se estandarizan terminologías que puedan resultar hasta antagónicas entre los diversos métodos OO.
Muchos de los términos utilizados aquí como asociaciones, herencias, atributos y métodos se explicaron en el apartado de Programación Orientada a Objetos.

DIAGRAMAS DE INTERACCIÓN.

Son modelos que describen la manera en que colaboran grupos de objetos para cierto comportamiento.
Habitualmente un diagrama de interacción capta el comportamiento de un solo caso de uso. El diagrama muestra cierto número de ejemplos de objetos y los mensajes que se pasa entre los objetos dentro del caso de uso.
Hay dos tipos de diagramas de interacción: los diagramas de secuencia y los diagramas de colaboración.

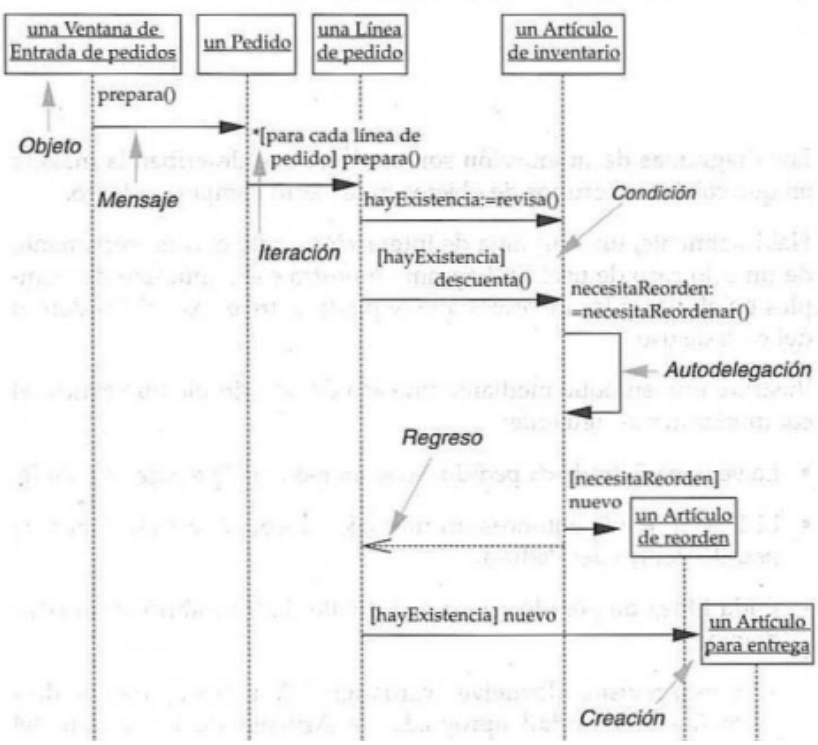


Figura 2.6.2.3 Ejemplo de un diagrama de secuencia

La línea vertical se llama línea de vida del objeto y representa la vida del objeto durante la interacción, cada mensaje se representa mediante una flecha entre las líneas de vida de los objetos, el orden en que se dan estos mensajes es de arriba hacia abajo, una autodelegación es cuando un objeto se manda un mensaje a sí mismo. Un marcador de iteración es cuando se envía un mensaje muchas veces a varios objetos receptores, y finalmente un regreso es cuando se envía de vuelta el mismo mensaje mandado y no uno nuevo.

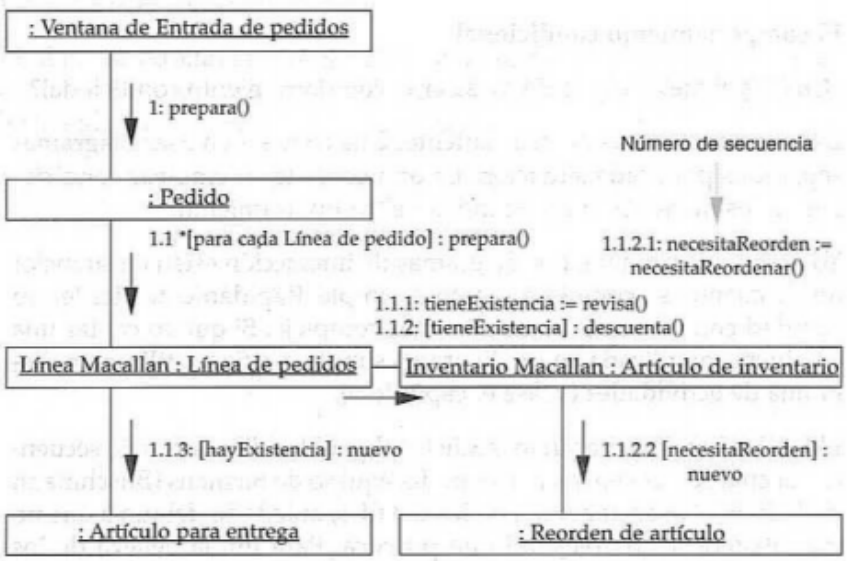


Figura 2.6.2.4 Ejemplo de un diagrama de colaboración.

Una de las características principales de ambos diagramas es su simplicidad. Se pueden ver con facilidad los mensajes con tan solo mirar el diagrama. Sin embargo si se trata de representar algo más que solo un simple proceso secuencial. Sin mucho comportamiento condicional o de ciclos, la técnica puede quedarse corta o insuficiente.
Algunas personas prefieren el diagrama de secuencia por el énfasis que se le pone a la secuencia, y es fácil aprender el orden en el que suceden las cosas, por otra parte las personas que prefieren el diagrama de colaboración lo hacen porque pueden usar la distribución para indicar cómo se conectan estáticamente los objetos.

2.7. Introducción a la seguridad del software.

Concepto de seguridad informática.

La seguridad informática consiste en asegurar que los recursos del sistema de información (material informático o programas) de una organización sean utilizados de la manera que se decidió y que el acceso a la información allí contenida, así como su modificación, sólo sea posible a las personas que se encuentren acreditadas y dentro de los límites de su autorización.

Metas de la seguridad enfocadas al software.

DISPONIBILIDAD.
Este término hace referencia al método de precaución contra posibles daños tanto en la información como en el acceso a la misma: ataques, accidentes o, simplemente, descuidos pueden ser los factores que obligan a diseñar métodos para posibles bloqueos.

CONFIDENCIALIDAD.
La información puede ser accedida únicamente por las personas que tienen autorización para hacerlo. Por ejemplo, cuando decimos que Internet es una Red de redes, estamos diciendo que hay medios que se entrelazan entre sí para lograr una vinculación. Es por ello que la confidencialidad se puede ver amenazada si alguien intercepta los paquetes que viajan de un lado al otro.

AUTENTICIDAD.
Algunos profesionales de la seguridad no incluyen este ítem cuando hablan de los pilares. Particularmente, creemos que no se puede soslayar este concepto, debido al hecho de que integridad nos informa que el archivo, por ejemplo, no ha sido retocado ni editado, y autenticidad nos informa que el archivo en cuestión es el real.

INTEGRIDAD.
Cuando nos referimos a integridad, queremos decir que estamos totalmente seguros de que la información no ha sido borrada, copiada o alterada, no sólo en su trayecto, sino también desde su origen. Por ejemplo, si un atacante modifica información confidencial para provecho propio, o si dicha información está codificada y el mismo atacante, al no poder leerla claramente, la borra.

Concepto de código abierto y código cerrado.

Código abierto (open source en inglés) es el término por el que se le conoce a cierto tipo de software. Este término empezó a utilizarse en 1998 por usuarios de la comunidad del software libre, tratando de usarlo como reemplazo al ambiguo nombre original del software libre (free software).

En inglés, "free software" puede significar diferentes cosas. Por un lado, permite pensar en "software por el que no hay que pagar", y se adapta al término de forma igualmente válida que el significado que se pretende (software que posee ciertas libertades). Lamentablemente, el término no resultó apropiado como reemplazo para el ya tradicional free software, y en la actualidad es utilizado para definir un movimiento nuevo de software, diferente al movimiento del software libre, aunque no completamente incompatible con este, de modo que es posible (como de hecho ocurre) que ambos movimientos trabajen juntos en el desarrollo práctico de proyectos.

El significado obvio del término "código abierto" es "se puede mirar el código fuente", lo cual es un criterio más débil y flexible que el del software libre; un programa de código abierto puede ser software libre, pero también puede serlo un programa semi-libre o incluso uno completamente propietario. El software de código abierto es software para el que su código fuente está disponible públicamente, aunque los términos de licenciamiento específicos varían respecto a lo que se puede hacer con ese código fuente.

Por otro lado código cerrado hace referencia a la invisibilidad del código, es decir, que el código fuente no es revelado para su estudio y pruebas. La mayoría de los programas en el mercado son de código cerrado, ya que de ser abierto expondrían una mayor cantidad de vulnerabilidades de sus sistemas, lo que los haría inseguros.