UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO.

FACULTAD DE INGENIERÍA.

“DESARROLLO DE UNA PASARELA DE PAGOS PARA UNA EMPRESA DEL SECTOR PRIVADO”.

INFORME DE TRABAJO PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN.

PRESENTA

JOSUÉ DANIEL DURÁN CORTÉS.

DIRECTORA: M.I NORMA ELVA CHÁVEZ RODRÍGUEZ.

CIUDAD UNIVERSITARIA JUNIO 2015

AGRADECIMIENTOS

A mis padres: A quienes le debo todo lo que soy y agradezco todo su apoyo.

A mi hermana Montse q.d.e.p. Que desde donde quiera que esté, siempre he sentido su apoyo e iluminación.

A mi novia Marce: Quien me ha ayudado en las buenas y en las malas y ha sido un apoyo importantísimo durante mi desarrollo profesional.

A mis amigos y compañeros de trabajo, quienes me han motivado a seguir esta carrera hacia el éxito.

A mi amada casa de estudios y mi facultad, de las cuales me siento muy orgulloso de pertenecer.

ÍNDICE DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

1 INTRODUCCIÓN

En este trabajo expongo el desarrollo que se tuvo que llevar a cabo para la creación de una pasarela de pagos en línea.

Dicho proyecto estuvo dividido en dos etapas, en cada una explico los objetivos y alcances que se tuvieron que lograr. Las metodologías, herramientas y tecnologías utilizadas así como mis aportaciones a dicho proyecto.

En la sección de antecedentes y marco teórico explico de manera detallada todos los conceptos previos que tiene que saber el lector, para entender el tema de una mejor manera.

Cabe destacar que en dicho proyecto, tuve la oportunidad de involucrarme en todas las etapas del desarrollo, desde el diseño de la base de datos, el análisis del sector de negocio, la implementación y las pruebas del mismo. Por lo que pude aplicar muchos de los conceptos que aprendí durante la carrera y de mi experiencia laboral previa.

Por ultimo presento mis conclusiones sobre mi experiencia profesional en el proyecto, exponiendo los logros y dificultades que se tuvieron durante el desarrollo de éste.

2 ANTECEDENTES

2.1 TRANSACCIONES ELECTRÓNICAS ONLINE.

De acuerdo con Fonseca, Pérez, y Faurés (2013), una **transacción** para la economía y el comercio, es una operación de compraventa, es decir, el traspaso de efectivo desde una cuenta bancaria hacia otra. El comercio electrónico, también conocido como **e-commerce** (electronic commerce) opera con transacciones a través de medios electrónicos como internet. Este tipo de comercio ha venido a revolucionar las compras tradicionales, abriendo el panorama a no depender de un lugar físico como un punto de venta, la diferencias de horarios de trabajo o incluso ayudar a las personas con dificultades de movilidad y desplazamiento. De acuerdo con un informe publicado por la eMarketer Inc.; empresa especializada en estudios en profundidad del marketing digital, proyecta las estadísticas mundiales de las ventas de comercio electrónico en un periodo de 2012 a 2016, como se muestra en la siguiente tabla.

B2C Ecommerce Sales Worldwide, by Region, 2011-2016
billions

	2011	2012	2013	2014	2015	2016
North America	$327.77	$373.03	$419.53	$469.49	$523.09	$580.24
Asia-Pacific	$237.86	$315.91	$388.75	$501.68	$606.54	$707.60
Western Europe	$218.27	$255.59	$291.47	$326.13	$358.31	$387.94
Central & Eastern Europe	$30.89	$40.17	$48.56	$57.96	$64.35	$68.88
Latin America	$28.33	$37.66	$45.98	$55.95	$63.03	$69.60
Middle East & Africa	$14.41	$20.61	$27.00	$33.75	$39.56	$45.49
Worldwide	**$856.97**	**$1,042.98**	**$1,221.29**	**$1,444.97**	**$1,654.88**	**$1,859.75**

Note: includes travel, digital downloads and event tickets purchased via any digital channel (including online, mobile and tablet); excludes gambling; numbers may not add up to total due to rounding
Source: eMarketer, June 2013

159668 www.eMarketer.com

Ilustración 1 Estadísticas mundiales de ventas en ecommerce del 2012 al 2016.

Como se puede apreciar, las cifras en valores monetarios son exorbitantes. El comercio electrónico podría superar los 1,2 trillones de dólares (valores monetarios americanos).

3 MARCO TEÓRICO

3.1 BANCA ELECTRÓNICA.

La Banca Electrónica (**E-Banking**) surge con el desarrollo del comercio electrónico, es el reflejo del banco tradicional pero desplegado a través de internet. Su uso permite un rápido y cómodo acceso a servicios bancarios como: revisar su saldo bancario, transferir dinero entre cuentas y pagar sus cuentas. Este intercambio de información financiera hacia los bancos electrónicos constituye lo que son las transacciones bancarias online. El banco virtual tiene ventajas sobre el tradicional pues permite: un amplio marco geográfico, rapidez y simplicidad en las transacciones, un mayor control sobre las cuentas, mejor servicio al cliente, no requiere de presencia física y los servicios están disponibles todo el tiempo que se requiera sin importar hora o lugar donde se encuentre el cliente (Fonseca et al, 2013).

3.2 PASARELAS DE PAGOS.

Son sistemas de pago electrónico en línea que permiten la realización de pagos y transferencias entre tiendas electrónicas y entidades bancarias de manera segura. Una de sus labores es cifrar la información confidencial como: número de tarjeta, nombre de usuario, csv, etc; que se requiere para efectuar transacciones bancarias por internet.

Las pasarelas de pago se integran a la tienda virtual (E-commerce) y almacenan información del banco que maneja las cuentas de compradores (banco emisor) y vendedores (banco adquiriente). En el proceso de pago con tarjeta, la pasarela de pago valida la veracidad de la tarjeta, organizando y gestionando la transferencia del dinero de la cuenta del comprador a la cuenta del vendedor.

Erróneamente son llamadas terminales de punto de venta virtual (TPV), pero no son realmente terminales de punto de venta porque estas últimas sí pertenecen al banco y el vendedor debe tener una cuenta con el banco en donde esté implementado el TPV virtual. Para el proceso de pago o transferencias bancarias es necesario el uso de tarjetas de crédito o débito como medios electrónicos de pago (Fonseca et al, 2013).

3.2.1 Proceso de pago.

El proceso de pago utilizando las pasarelas básicamente consta de estas fases:

1. El cliente accede a un sitio de comercio electrónico y elige los artículos a comprar (carrito de compras).
2. El e-commerce calcula el importe a cobrar y cuando el cliente está listo para pagar, es dirigido a la pasarela de pago, la cual le muestra el monto a pagar y los datos a introducir como el número de tarjeta y su número de verificación (CSV).
3. La pasarela se encarga de cifrar la información del carrito de compras y los datos del cliente, viajando estos de manera segura hacia el banco.
4. La pasarela comprueba con el procesador de pagos que la tarjeta sea válida (que no sea robada o que esté caducada) y que el cliente tenga los fondos suficientes para comprar los artículos.
5. Si el proceso se lleva a cabo de manera exitosa, se ingresa el dinero en la cuenta del vendedor, la misma debe pertenecer al banco en cuestión o este debe tener relación con el banco que posee la cuenta del vendedor.
6. La pasarela le hace saber al comercio y al cliente el resultado de la transacción (si el pago fue exitoso o no).

(Fonseca et al, 2013).

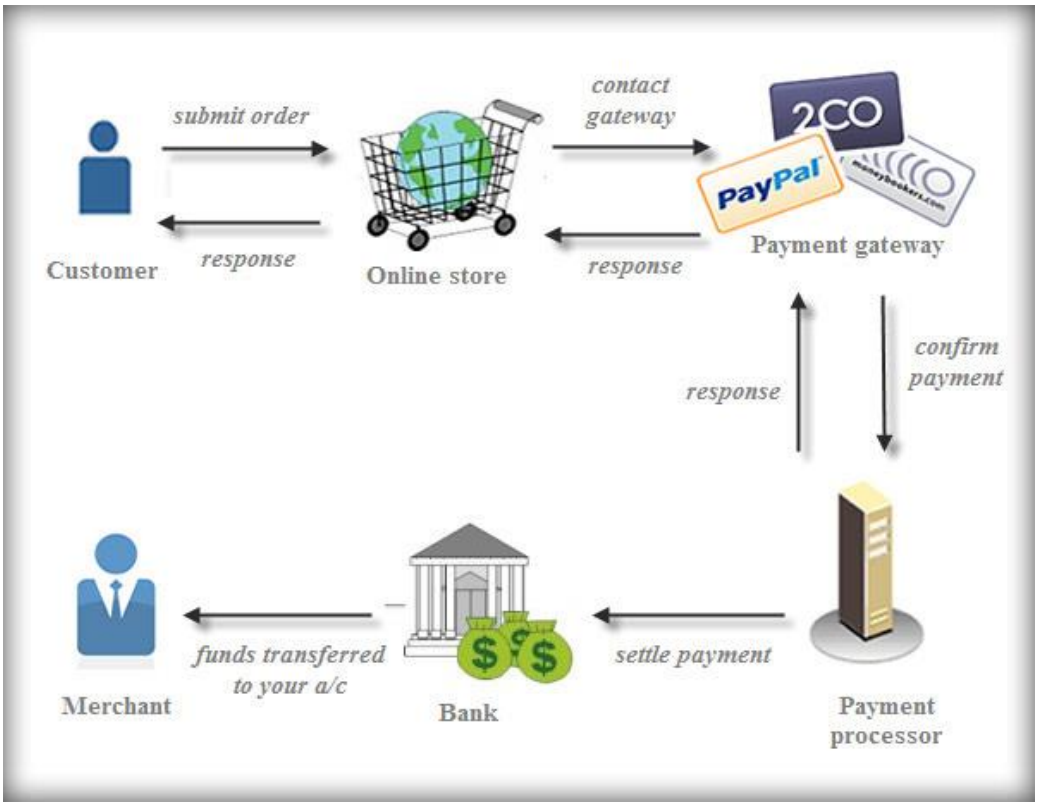


Ilustración 2 Proceso de pago de una pasarela de pagos.

De acuerdo con el Banco de México, los conceptos de banco emisor, banco adquiriente y procesador de pagos, se definen de la siguiente manera:

3.2.2 Banco Emisor.

Es el banco que emite la tarjeta asociada a un contrato de crédito o a una cuenta corriente.

3.2.3 Banco Adquirente.

Es el banco que lleva al comercio una cuenta en la que depositará los importes de las compras con tarjetas, y se encarga de administrar la terminal electrónica instalada en el comercio para procesar las compras en línea.

3.2.4 Procesadores de pagos (Payment processor).

Son empresas que proporcionan los servicios de comunicación entre los bancos emisores y los adquirentes; y las marcas, quienes establecen los estándares operativos y financieros que los bancos emisores deben cumplir para poder llevar la marca en la tarjeta. Cualquier banco mexicano puede emitir tarjetas y proporcionar servicios a los comercios para que acepten pagos. **Prosa** y **E-global** son los procesadores que operan en el país, cada uno de ellos da servicios de comunicación y proceso de operaciones de pago con tarjetas a varios bancos emisores y adquirentes.

3.2.5 Pasarelas de pagos más utilizadas.

Hoy en día existen una gran variedad de pasarelas de pagos, diferenciándose unas de otras por el costo que cobran por transacciones y los países en los cuales operan.

Paypal es la pasarela de pagos más conocida a nivel mundial y pertenece a los Estados Unidos (EE.UU) y opera en más de 50 países. Su servicio permite la transferencia de dinero entre usuarios que tengan correo electrónico; es una alternativa a los tradicionales cheques. Con el uso de esta pasarela se pueden realizar peticiones de pago en comercio electrónico de sitios web de terceros, lo cual implica un sistema de validación de pagos online portable y adaptable. Los métodos de transferencia y pago por medio del correo electrónico tienen asociado una tarjeta de crédito (tales como Visa, Mastercard, American Express y Diners de compradores de EE.UU y Europa). Estos procesos son rápidos y seguros, debido al protocolo de seguridad SSL.

2Checkout (2CO) es la segunda pasarela a nivel mundial, por su uso, calidad, seguridad y prestigio. Esta pasarela trabaja con el 95% de los países del mundo. El pago se realiza al momento de registrarse como usuario. Con esta pasarela se puede vender a todo el mundo y recibir dinero en cualquier banco. Acepta tarjetas

de crédito como: Visa, Mastercard, American Express y Diners. El sistema realiza en tiempo real la verificación de la tarjeta de crédito. La información del comprador viaja cifrada con un nivel de seguridad de 128 bits (protocolo SSL), y se utilizan certificados digitales que garantizan la autenticidad de las partes implicadas y con ello la seguridad de la transacción (Fonseca et al, 2013).

3.3 SEGURIDAD EN LAS TRANSACCIONES ELECTRÓNICAS.

Cuando se realizan transacciones bancarias en línea se debe tener un estricto control de los mecanismos de seguridad que protegen el sistema de ataques a la autenticidad, confidencialidad, integridad, disponibilidad y el no repudio de la información (son los llamados pilares de la seguridad informática) (Fonseca et al, 2013).

3.3.1 Criptografía.

La criptografía es el arte o ciencia de cifrar y descifrar información por medio de técnicas especiales y es empleada comúnmente para permitir un intercambio de mensajes que sólo puedan ser leídos por personas a las que van dirigidos y que poseen las herramientas para descifrarlos. Entre las principales aplicaciones de la criptografía están: el cifrado y la firma electrónica; ambas muy importantes en el comercio electrónico y de cualquier transacción segura que se realice en línea.

Para mantener la información a salvo de todos, a excepción del emisor y el receptor autorizados de la misma y que permiten garantizar que el pago se ha realizado, se emplean las técnicas de cifrado.

Las técnicas de cifrado tratan de asegurar que:

• Sólo el receptor debe ser capaz de acceder a los datos en claro (**confidencialidad**).

• Nadie ha podido añadir, quitar o cambiar los datos originales del mensaje, o los que puedan acompañarlo (**integridad**).

• El mensaje o los datos provienen de quien dice ser (**autenticación**).

Para lograrlo se emplean los algoritmos de cifrado simétricos y asimétricos. Para descifrar los mensajes mediante los algoritmos simétricos, el receptor tiene que aplicar sobre el mensaje cifrado la misma clave que empleó el emisor para cifrar el mensaje original. Los algoritmos simétricos conocidos son: DES, Triple DES, IDEA, RC2, RC4 y RC5. Los algoritmos asimétricos (o de clave pública como también se conocen) se basan en el uso de dos claves diferentes. Una clave puede descifrar lo que la otra ha cifrado. Las claves pública y privada tienen características matemáticas especiales, de tal forma que se generan siempre a la vez y por parejas. Ejemplos de algoritmos asimétricos: Diffie-Hellman, El Gamal, RSA (Fonseca et al, 2013).

3.3.2 Protocolo HTTP.

El Protocolo de Transferencia de HiperTexto (Hypertext Transfer Protocol) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. La especificación completa del protocolo HTTP 1/0 está recogida en el RFC 1945. Fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como el World Wide Web (WWW).

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de

los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL.

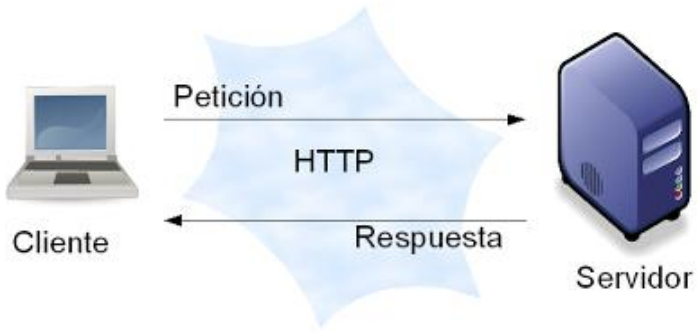


Ilustración 3 Arquitectura cliente servidor.

3.3.3 Etapas de una transacción HTTP.

Cada vez que un cliente realiza una petición a un servidor, se ejecutan los siguientes pasos:

- Un usuario accede a una URL, seleccionando un enlace de un documento HTML o escribiéndola directamente en el campo Location del cliente Web.
- El cliente Web descodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.

- Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente.
- Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, HEAD,…), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada (casi siempre HTTP/1.0) y un conjunto variable de información, que incluye datos sobre las capacidades del browser, datos opcionales para el servidor,…
- El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.
- Se cierra la conexión TCP.

Este proceso se repite en cada acceso al servidor HTTP. Por ejemplo, si se recoge un documento HTML en cuyo interior están insertadas cuatro imágenes, el proceso anterior se repite cinco veces, una para el documento HTML y cuatro para las imágenes. (El protocolo http, NEO).

3.4 PROTOCOLOS DE SEGURIDAD.

El protocolo **Secure Socket Layer (SSL)** facilita la autenticación y privacidad de la información en internet mediante el uso de la criptografía. Sólo el servidor es autenticado (es decir, se garantiza su identidad) mientras que el cliente se mantiene sin autenticar; la autenticación mutua requiere un despliegue de infraestructura de claves públicas para los clientes. El SSL se ejecuta en una capa entre los protocolos de aplicación como el Hypertext Transfer Protocol (HTTP). Proporciona sus servicios de seguridad utilizando la criptografía de llave pública y privada. Para el intercambio de los datos entre el servidor y el cliente, utiliza algoritmos de cifrado simétrico. Para la autenticación, usa el algoritmo de cifrado de clave pública (RSA).

El protocolo **HTTPS** es la versión segura de HTTP. Fue desarrollado por Enterprise Integration Technologies (EIT). Permite el cifrado y autenticación digital igual que SSL. La diferencia consiste, en que HTTPS es un protocolo de nivel de aplicación, es decir, que extiende el protocolo HTTP por debajo. HTTPS es usado para asegurar páginas World Wide Web (www, para aplicaciones de e-commerce, utilizando certificados de clave pública para verificar la identidad de los participantes (Fonseca et al, 2013).

3.5 SISTEMAS ANTIFRAUDE.

3.5.1 ¿Qué se entiende por fraude?

El término fraude, de acuerdo con la Chartered Institute of Management Accountants (2009), esencialmente implica el uso del engaño para deshonestamente obtener un beneficio personal y crear una pérdida a otro. El **e-crime**, es el fraude cometido por las personas que utilizan las computadoras y la tecnología para cometer delitos, como: suplantación de identidad, spam, piratería; hacking, fraudes de ingeniería social, etc.

3.5.2 Fraude en transacciones electrónicas.

En los últimos años, las industrias de las tarjetas de crédito, han tenido que utilizar herramientas como las redes neuronales, para prevenir y detectar fraudes electrónicos. Estas redes neuronales, funcionan como radares para detectar si una transacción puede llegar a ser fraudulenta o no. Por ejemplo si se realizara una transacción con la misma tarjeta desde dos lugares distintos con una distancia significativa y tiempo corto, el radar detectaría que podría existir un riesgo en dicha transacción, este tipo de herramientas trabajan con estadísticas y modelos matemáticos complejos, para conocer la frecuencia de compra del usuario, identificar la dirección IP de la computadora donde proviene dicha transacción, saber si esa tarjeta ya se ha utilizada para cometer otros fraudes, etc. A este tipo de

herramientas, se les suele llamar sistemas o plataformas antifraude y comúnmente se configuran a través de un **BRMS**.

3.5.3 BRMS (Sistema de Gestión de Reglas de Negocio).

Un BRMS es una herramienta de gestión empresarial, que permite representar eficazmente la lógica empresarial mediante reglas de negocio y facilita la gestión de las mismas de forma ágil y accesible a los usuarios de negocio. Este tipo de herramienta permite que el mantenimiento de las reglas de negocio pueda quedar en manos de los usuarios de negocio sin que sea necesario conocimientos técnicos y de programación (decide, 2013). Las reglas pueden ser creadas, modificadas y probadas rápidamente y después ser enviadas a producción cuando estén listas. Esto permite que las instituciones puedan reaccionar de manera rápida contra los fraudes. Otra de las ventajas que permiten los BRMS a las empresas, es que éstas ya no tienen que tener necesariamente diferentes soluciones para detectar los fraudes, el BRMS las unifica. Entre los BRMS más conocidos encontramos a Cybesource con su solución de administración de fraudes.

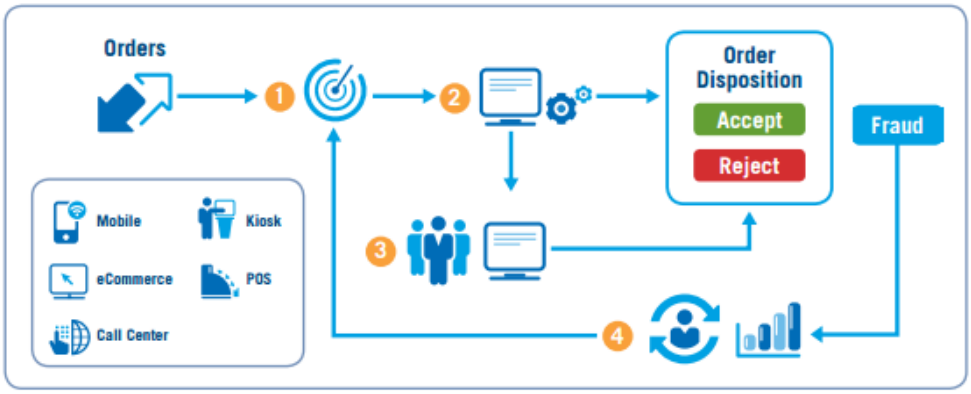


Ilustración 4 Proceso del BRMS de Cybersource.

1. Radar de detección de fraude.
2. Configuración de reglas de negocio.
3. Gestión de revisión de casos de fraude.
4. Sincronización y reportes.

3.6 ISO 8583.

De acuerdo con Suman. K (2010) el **ISO 8583** es el estándar de la International Organization for Standardization (ISO), para transacciones financieras con mensajes originados en una tarjeta; es el estándar para sistemas que intercambian transacciones electrónicas realizadas por los usuarios de tarjetas de crédito.

Cuando se genera una transacción, normalmente proviene de un punto de venta (POS) o un cajero automático y esta viaja a través de una red hacia el sistema que emitió la tarjeta para obtener la autorización financiera correspondiente. La transacción contiene información como el número de tarjeta, csv, número de comercio, el importe, etc. Las transacciones pueden ser compras, retiros, depósitos, reintegros, reversas, consultas de saldo, pagos y transferencias entre cuentas.

El ISO 8583 establece un formato de mensaje y un flujo de comunicación para que diferentes sistemas puedan intercambiar estas transacciones. En general todas las redes de tarjetas se basan en el ISO 8583 para poder transaccionar. Aunque el ISO 8583 define un standard común, normalmente cada red o sistema adapta el standard para su propio uso con campos modificados a sus necesidades particulares.

Un mensaje ISO 8583 consta de manera general de los siguientes elementos, un **Header** (cabecera), **Application Data** (datos de la aplicación) y un **Trailer**. El header y el trailer envuelven la application data y son utilizados para ruteo e integridad del mensaje. La application data está formada por un Message Type Indicator (MTI), BIT MAP (indica cuáles elementos están presentes) y el ISO Data Element (los campos del mensaje).

Application Data.

# Campo	Descripción
0	MTI Message Type Indicator.
1 - Bitmap	64 (o 128) bits, indica la presencia o ausencia de otros campos.
2 - 128	Otros campos especificados en el bitmap.

Header Message Type Indicator (MTI).

Este es un campo numérico de 4 dígitos que clasifica la función de alto nivel del mensaje. Un MTI incluye la versión ISO 8583, la clase (Message Class), la función (Message Function) y el origen del mensaje (Message Origin).

Posición 1. Indica la versión del estándar ISO 8583.

0xxx	ISO 8583:1987 version
1xxx	ISO 8583:1993 version
2xxx	Reservado para uso del ISO
3xxx	Reservado para uso del ISO
4xxx	Reservado para uso del ISO
5xxx	Reservado para uso del ISO
6xxx	Reservado para uso del ISO
7xxx	Reservado para uso del ISO
8xxx	Reservado para uso nacional
9xxx	Reservado para uso privado

Posición 2. Especifica la clase del mensaje.

Posición	Significado	Uso
x1xx	Autorización	Determina si existen fondos disponibles, obtiene una aprobación pero no se toma en cuenta para la conciliación, Dual Message System (DMS) system, espera el intercambio de archivos para imputar la cuenta.
x2xx	Financiero	Determina si existen fondos disponibles, obtiene una aprobación e imputa directamente a la cuenta , Single Message System (SMS), no se intercambian archivos.
x3xx	Manejo de Archivos	Usado por hot-card, TMS y otros cambios.
x4xx	Reverso	Reversa la acción de una autorización previa.
x5xx	Conciliación	Transmite información de cierre.
x6xx	Administrativo	Transmite información de falla en los mensajes.
x7xx	Fee Collection	
x8xx	Manejo de Red	Usado para intercambio seguro de claves, logon, echo test y otras funciones de red
x9xx	Reservado por la ISO	

Posición 3. Indica la función del mensaje.

Posición	Significado
xx0x	Petición (Request)
xx1x	Respuesta a la Petición (Request Response)
xx2x	Aviso (Advice)
xx3x	Respuesta al Aviso (Advice Response)
xx4x	Notificación
xx8x	Confirmación de respuesta (Response acknowledgment)
xx9x	No Confirmación (Negative acknowledgment)

Posición 4. Define el origen el mensaje.

Posición	Significado
xxx0	Comprador
xxx1	Comprador Repetición
xxx2	Emisor

xxx3	Emisor Repetición
xxx4	Otros
xxx5	Otros Repetición

Bitmap.

Es un campo o subcampos dentro del mensaje, que se utiliza para indicar los elementos que se encuentran en el mensaje. La presencia de un elemento de datos en un mensaje específico se indica mediante un uno (1) en la posición asignada; un cero (0) indica la ausencia de un elemento de datos en la posición asignada.

Cada transacción de aplicación incluye un (1) mapa de bits. Un mapa de bits consta de 64 bits de numerados desde la izquierda comenzando con el bit 1. El primer bit del mapa de bits representa un mapa de bits secundario. Si el mensaje ISO no soporta el procesamiento de mapa de bits secundario, entonces el primer bit del mapa de bits es '0'.

Bit 1	**Bit 2**	**Bit 3**	**Bit 4**	**…**	**Bit 64**
Campo 1 Bit map secundario. ‘1’ si esta presente ‘0 sino.’	Campo 2 Número de cuenta principal	Campo 3 Código de procesamiento.	Campo 4 Cargo, Transferencia		Campo 64 Mensaje de codigo de autorizacion.

3.7 CERTIFICACIÓN PCI DSS.

El Estándar de Seguridad de Datos para la Industria de Tarjetas de Pago (PCI DSS), es un conjunto de políticas y procedimientos destinados a optimizar la seguridad de las transacciones de las tarjetas de crédito, débito y efectivo y proteger a los titulares de las trajetas contra el mal uso de su información personal. El PCI DSS fue creado conjuntamente en 2004 por cuatro de las mayores compañías de tarjetas de crédito: Visa, MasterCard, Discover and American Express.

El PCI DSS específica 6 grandes objetivos:

1. Se debe mantener una red segura para que las transacciones puedan llevarse a cabo. Este requisito involucra el uso de firewalls lo suficientemente robustos y eficaces sin causar molestias indebidas a los titulares de las tarjetas o a los vendedores.
2. Si se almacena la información personal del tarjetahabiente, ésta se debe proteger de manera segura. Cuando los datos de los tarjetahabientes se transmiten a través de redes públicas, la información debe ir cifrada de manera efectiva. El cifrado digital es muy importante en todas las formas de transacciones con tarjetas de crédito, pero en particular en el comercio electrónico.
3. Los sistemas deben estar protegidos contra las actividades maliciosas de los hackers, mediante software actualizado como anti-virus, anti-spyware, entre otras soluciones anti-malware. Todas las aplicaciones deben estar libres de errores y vulnerabilidades que podrían abrir la puerta a la explotación, donde los datos de los titulares de las tarjetas puedan ser robados o modificados. Los parches de seguridad de los proveedores de software deben ser regularmente instalados, para asegurar el más alto nivel en el manejo de vulnerabilidades.
4. El acceso a los sistemas de información y operaciones debe ser restringido y controlado. Los tarjetahabientes no deben proporcionar información a las

empresas a no ser que estas deban conocerla para llevar a cabo la transacción. Cada persona que use una computadora en el sistema se le debe asignar un identificador único. La información de los titulares de las tarjetas debe ser protegida física y electrónicamente. Por ejemplo: controles de acceso, evitar la duplicación innecesaria de documentos, control en la eliminación de documentos, etc.

5. Las redes deben ser monitoreadas y probadas constantemente para garantizar que todas las medidas y procedimientos de seguridad estén funcionando correctamente.
6. Las políticas de seguridad deben ser definidas, mantenidas y acatadas todo el tiempo por todas las entidades participantes. Medidas de ejecución como auditorías y sanciones por incumplimiento deben de ser necesarias.

Traducido de Rouse M. (2009).

3.8 WEB SERVICES.

Un servicio web es un conjunto de estándares y protocolos que sirve como medio de comunicación entre aplicaciones web. El protocolo en que típicamente se trabaja es HTTP y HTTPS. Un web service usa un sistema de mensajería estándar como XML o JSON y no está ligado a ningún sistema operativo o lenguaje de programación. En términos más técnicos, un web service, es un sistema de software distribuido cuyos componentes pueden ser mostrados y ejecutados en distintos dispositivos. El web service recibe las peticiones del cliente y genera respuestas, como se puede apreciar en el siguiente esquema:

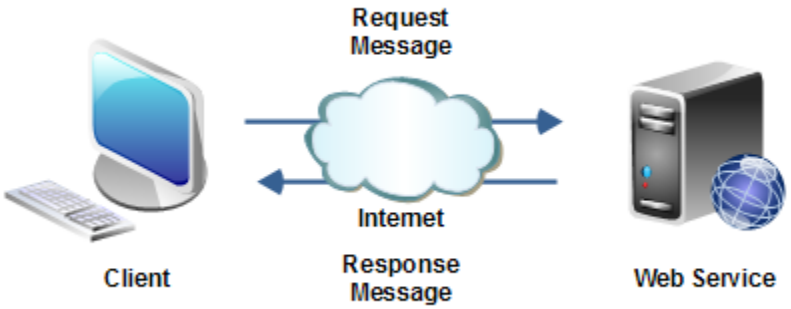


Ilustración 5. Flujo básico de un web service.

Los web services por lo regular pueden estar basados en dos metodologías: SOAP o REST.

3.9 SOAP (SIMPLE OBJECT ACCESS PROTOCOL).

Es el protocolo estándar de los Web Services. Este protocolo está basado en XML y no se encuentra casado a ninguna plataforma o lenguaje de programación. También es el protocolo más aceptado por la mayoría de las plataformas.

Si bien SOAP es un protocolo, éste no es exactamente un protocolo de comunicación entre mensajes como lo es el HTTP. SOAP básicamente son documentos XML y se necesita otro protocolo para la transmisión de estos documentos como el protocolo HTTP o cualquier otro capaz de transmitir textos. Los mensajes SOAP están compuestos por un tag principal llamado **Envelope**, que está dividido en una cabecera o **Header** y en un cuerpo o **Body**. (Web Services con C#. Principios de Web Services).

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <miCabecera>miValor</miCabecera>
  </soap:Header>
  <soap:Body>
    <MiMetodo>
      <MiParametro>miValor</MiParametro>
    </MiMetodo>
  </soap:Body>
</soap:Envelope>
```

Ilustración 6 Estructura básica de un mensaje SOAP.

3.10 REST (REPRESENTATIONAL STATE TRANSFER).

Es un conjunto de principios, o maneras de hacer las cosas, que define la interacción entre distintos componentes. El protocolo más usado que cumple esta definición, es el protocolo HTTP.

Esto quiere decir, por extensión, que toda aplicación web bajo el protocolo HTTP es a su vez una aplicación REST. Sin embargo, como se explica posteriormente, eso no implica en absoluto que todas las aplicaciones web sean servicios web RESTful, ya que estas tienen que cumplir una serie de requisitos para ser consideradas tales. (Fernández. A., 2013).

3.10.1 Reglas de la arquitectura REST.

En palabras de (Fernández , 2013).

- **Arquitectura cliente-servidor**: consiste en una separación clara y concisa entre los 2 agentes básicos en un intercambio de información: el cliente y el servidor. Estos 2 agentes deben ser independientes entre sí, lo que permite una flexibilidad muy alta en todos los sentidos.

- **Stateless**: esto significa que nuestro servidor no tiene porqué almacenar datos del cliente para mantener un estado del mismo. Esta limitación es sujeto de mucho debate en la industria, incluso ya empiezan a usarse tecnologías relacionadas que implementan el estado dentro de la arquitectura, como WebSockets.

- **Cacheable**: esta norma implica que el servidor que sirve las peticiones del cliente debe definir algún modo de cachear dichas peticiones, para aumentar el rendimiento, escalabilidad, etc. Una vez más, HTTP implementa esto con la cabecera “Cache-control”, que dispone de varios parámetros para controlar la cacheabilidad de las respuestas.

- **Sistema por capas**: nuestro sistema no debe forzar al cliente a saber por qué capas se tramita la información, lo que permite que el cliente conserve su independencia con respecto a dichas capas.

- **Interfaz uniforme**: esta regla simplifica el protocolo y aumenta la escalabilidad y rendimiento del sistema. No queremos que la interfaz de comunicación entre un cliente y el servidor dependa del servidor al que estamos haciendo las peticiones, ni mucho menos del cliente, por lo que esta regla nos garantiza que no importa quién haga las peticiones ni quien las reciba, siempre y cuando ambos cumplan una interfaz definida de antemano.

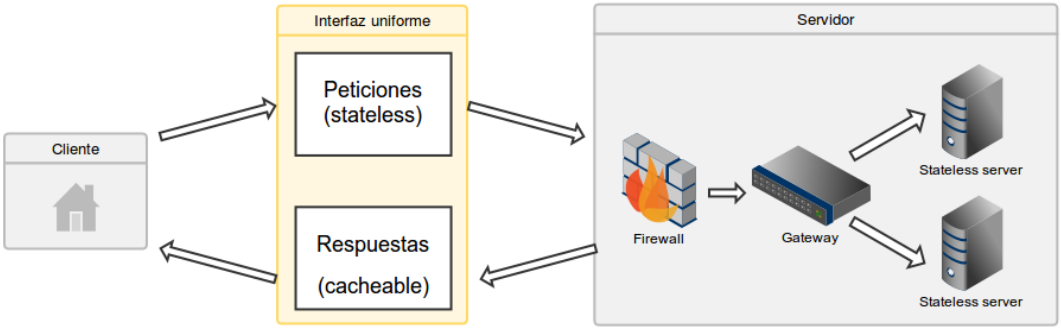


Ilustración 7 . Esquema de arquitectura REST.

3.10.2 ¿Cómo se consigue un web service 100% REST ?

Para que una aplicación o servicio web sea considerado REST al 100%, principalmente se necesitan 4 principios:

- **Identificación de recursos**: cualquier aplicación REST debe ser capaz de identificar sus recursos de manera uniforme. HTTP implementa esto usando las llamadas **URIs** (Uniform resource identifier). Esta es la URL que usamos tradicionalmente, y aunque hay una diferencia sutil entre URLs y URIs, se puede decir que toda URL es una URI a su vez. Esta identificación del recurso no incluye la representación del mismo, cosa que se verá a continuación.
- **Recursos y representaciones**: ya que todo recurso debe tener una identificación (URI), REST define también la manera en que podemos representar las interacciones con el recurso, ya sea para editarlo o borrarlo, directamente del servidor. La forma en que se pueden realizar estas acciones depende de la programación del sistema, pero HTTP define distintas cabeceras de tipos, y un contenido en la respuesta, por lo que nuestras aplicaciones pueden enviar el contenido en el formato que quieran, siempre y cuando este contenido contenga la información

necesaria para poder operar con el objeto en el caso de que tengamos permiso para hacerlo.

- **Mensajes auto descriptivos**: cuando hacemos peticiones a un servidor, éste debería devolver una respuesta que nos permita entender sin lugar a duda cual ha sido el resultado de la operación, así como si dicha operación es cacheable, si ha habido algún error, etc. HTTP lleva a cabo esto a través del estado y cabeceras. El uso de estas cabeceras y estados depende del desarrollo del sistema. Esto es importante tenerlo en cuenta, ya que, desgraciadamente, un gran número de aplicaciones y servicios web no respetan esta regla (por lo tanto no pueden ser considerados REST).
- **HATEOAS**: los web services deben incluir en las respuestas del servidor toda aquella información que necesita el cliente para seguir operando con este servicio web. En otras palabras, el cliente *no* tiene porqué saber que cuando obtenemos, por ejemplo, un objeto cualquiera, tenemos además las opciones de modificarlo, o eliminarlo. El servidor debe enlazar a estas operaciones en la respuesta a dicha petición. De esta manera, lo único que necesita saber un cliente sobre una aplicación REST, es el punto de entrada (*endpoint*). Además nos garantiza más independencia entre el cliente y el servidor. Desgraciadamente, este último principio no se implementa en la mayoría de APIs que usamos hoy en día, por lo que, siendo estrictos, podríamos decir que la mayoría de servicios web no son 100% RESTful. (Fernández , 2013).

3.10.3 Servicios web RESTful.

En palabras de (Fernández, 2013) un servicio web RESTful contiene lo siguiente:

- **URI del recurso.** Por ejemplo: *http://api.servicio.com/recursos/casas/1* (esto nos daría acceso al recurso "Casa" con el ID "1″)
- **El tipo de la representación de dicho recurso**. Por ejemplo, podemos devolver en nuestra cabecera "Content-type: application/json", por lo que el cliente sabrá que el contenido de la respuesta es una cadena en formato JSON, y podrá procesarla como prefiera. El tipo es arbitrario, siendo los más comunes **JSON, XML** y **TXT**.
- **Operaciones soportadas**: HTTP define varios tipos de operaciones (verbos), que pueden ser **GET, PUT, POST, DELETE, PURGE**, entre otros. Es importante saber para qué están pensados cada verbo, de modo que sean utilizados correctamente por los clientes.
- **Hipervínculos**: por último, nuestra respuesta puede incluir hipervínculos hacia otras acciones que podamos realizar sobre los recursos. Normalmente se incluyen en el mismo contenido de la respuesta, así si por ejemplo, nuestra respuesta es un objeto en JSON, podemos añadir una propiedad más con los hipervínculos a las acciones que admite el objeto.

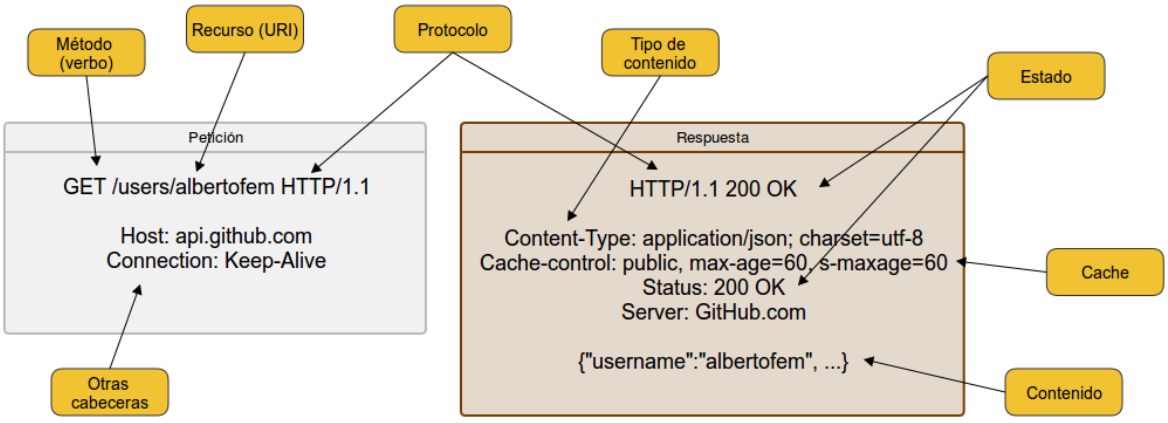


Ilustración 8 . Ejemplo de una petición a un servicio web de Github.

3.10.4 ¿Para qué sirven los web services?

El uso de los web services facilita la reutilización de funciones de una aplicación en distintas plataformas o lenguajes ya sea para un uso personal en distintos proyectos, para comercializarlos o adquirir prestaciones de terceros. Se puede referenciar funciones que se estén ejecutando en otra computadora o servidor sin importarnos en qué están programados ni en qué plataforma están corriendo.

Uno de los ejemplos más comunes del uso de los web services se encuentra en los sitios web de comercio electrónico, los cuales hacen uso de un Web Service para validar los datos de las tarjetas de crédito de sus clientes. Normalmente este Web Service es provisto por algún banco o entidad financiera que actúa como intermediario entre el comercio y las tarjetas de crédito. (Web Services con C#. Principios de Web Services).

3.11 LARAVEL.

Laravel es un framework de código abierto para desarrollar aplicaciones y servicios web con PHP 5. Su filosofía es desarrollar código PHP de forma elegante y simple. Fue creado en 2011 por Taylor Otwell y tiene una gran influencia de frameworks como Ruby on Rails, Sinatra y ASP.NET MVC.

El framework intenta aprovechar lo mejor de otros frameworks y además aprovechar las características de las últimas versiones de PHP 5. Gran parte de Laravel está formado por dependencias, especialmente de Symfony, esto implica que el desarrollo de Laravel dependa también del desarrollo de sus dependencias.

Laravel posee Rutas, Modelos, Plantillas, Vistas y Controladores, además de un motor propio para el manejo de plantillas llamado Blade.

Características básicas.

- Sistema de ruteo, soporta RESTful.
- Blade, motor de plantillas.
- Peticiones Fluent.
- Eloquent ORM.
- Composer como gestor de dependencias.
- Soporte para caché
- Soporte para MVC
- Manejo de componentes de Symfony

(Laravel PHP framework, 2014).

.

4 PROYECTO PASARELA DE PAGOS

4.1 PRIMERA ETAPA.

4.1.1 Problemática.

Crear un servicio para conectar con el procesador de pagos de la empresa, el cual podía transaccionar solamente con los medios de pago (tarjetas de crédito), que pertenecen a la marca de la empresa.

Este tipo de tarjetas no transaccionan de la misma manera que las tarjetas de crédito y débito comunes, como: Visa, Mastercard o American Express. El procesador de pagos que las trabaja opera de manera independiente, ocupando el estándar para transacciones electrónicas ISO 8583.

Para esta etapa del proyecto, se tuvo que desarrollar un servicio RESTful ocupando el lenguaje de programación PHP. Cuando llegué al proyecto, dos compañeros ya habían empezado a desarrollar dicho framework, el cual se había diseñado bajo el patrón de diseño MVC (Modelo Vista Controlador), el framework era básico pero funcional, y contaba con conexiones a bases de datos como Mysql y DB2, además de métodos para intercambiar información entre los modelos, las vistas, y los controladores.

4.1.2 Diseño de API RESTful.

El API que se tuvo que desarrollar para el gateway de pagos, de la primera etapa, debía cumplir con las siguientes buenas prácticas:

1. Uso de SSL, en todos los casos.
2. Realizar una buena documentación del API, entendible para el desarrollador o la persona que la implemente.
3. Versionamiento a través de la URL y no de los encabezados.
4. Uso de parámetros de consulta para filtros avanzados de búsqueda y ordenamiento.
5. Limitar los campos de las consultas devueltos por el API.
6. Basarse en los métodos de HTTP (POST, GET, PATCH & PUT).
7. Uso de json como formato de respuesta.
8. Utilizar el estilo de escritura camelCase.
9. Paginado utilizando encabezados link.
10. Uso de autenticación basada en tokens.
11. Incluir encabezados de respuesta para cache.
12. Utilizar o hacer analogía a los códigos de respuesta de HTTP.
13. Utilizar estándares web que tengan sentido.
14. Debía ser amigable para el desarrollador y ser explotable desde la barra de dirección del navegador.
15. Debía ser sencilla, intuitiva y consistente.
16. Debía ser eficiente.

4.1.2.1 Ruteo de peticiones.

Una de las primeras actividades que tuve en el proyecto fue el desarrollo del ruteo de peticiones para el servicio RESTful que se estaba desarrollando. El ruteo de peticiones es una parte esencial en los web services, los cuales deben seguir ciertos principios.

4.1.2.2 Los principios fundamentales de REST:

-Separar el API en recursos lógicos.
-Los recursos son manipulados a través de peticiones HTTP en los cuales los métodos: GET, POST, PUT, PATCH y DELETE; tienen un significado específico.
-Los recursos deben ser identificados con sustantivos y no con verbos. Por ejemplo un recurso puede ser alumno, maestro o grupo.
-Los recursos deben de hacer sentido al consumidor del API.
-Los recursos abstraen y aíslan detalles de implementación.
-Una vez que ya se tienen identificados los recursos, se necesita identificar las acciones que aplican a ellos y cómo se relacionan con el API.

Método	Acción
GET	Consultar uno o más recursos.
POST	Creación de un nuevo recurso.
PUT	Actualizar un recurso.
PATCH	Actualización parcial del recurso.
DELETE	Eliminación del recurso.

-Mantener la estructura de la URL limpia y clara.
-Por conveniencia, se debe utilizar palabras en plural y en inglés para nombrar a los recursos.
Para desarrollar el ruteo de peticiones tuve que realizar lo siguiente:
Segmenté la URL que proviene del navegador, para obtener la URI (identificador del recurso), así como los parámetros de consulta, para poder redirigir al controlador destino y éste a su vez hiciera uso de los modelos y vistas correspondientes.

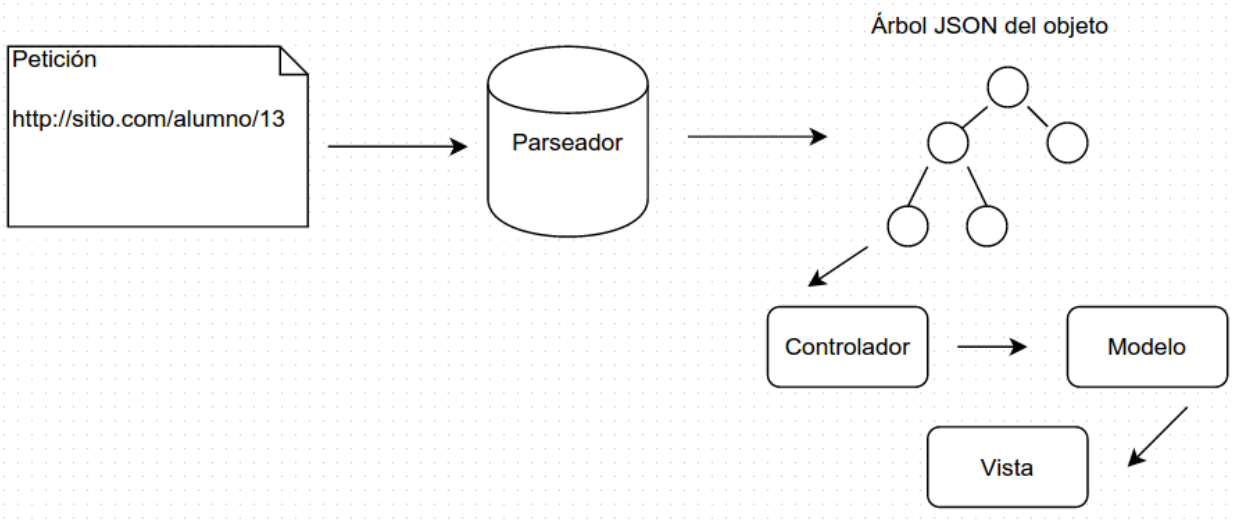


Ilustración 9 Esquema general del parseador de peticiones.

4.1.2.3 Tokenización.

Con el fin de dar acceso y proteger la información confidencial, que se almacena, se transmite y se procesa en un sistema, existen diversas técnicas que permiten proteger la información del personal no autorizado, algunos de estos métodos son:

-Uso de técnicas hash.
-Cifrado simétrico y asimétrico.
-Truncamiento.
-Transposición.
-Enmascaramiento.
-Tokenización.

El uso de estos mecanismos depende de cada organización, se puede utilizar uno o varios en conjunto.

La **tokenización** o uso de tokens: son valores únicos asociados a datos confidenciales, que son empleados como reemplazo de estos últimos, con la característica diferencial de que el token no permite inferir el dato confidencial con

el que se relaciona, minimizando de esta manera el riesgo de almacenamiento inseguro y la necesidad de implementación de controles asociados a datos confidenciales, ya que el token en sí no es catalogado como un dato confidencial.

El uso de esta técnica es bastante amplio, lo encontramos en la protección de información de carácter personal (LOPD), datos de seguridad social, información clínica, patentes y datos de tarjetas de pago a los que afecta la normativa Payment Card Industry Data Security Standard (**PCI DSS**), entre otros.

La arquitectura básica de un sistema de tokenización abarca tres componentes principales, una base de datos maestra centralizada, un servicio de cifrado/tokenización e interfaces de comunicación.

BD maestra.

Los datos confidenciales que deben ser centralizados en una única base de datos (BD). Cada dato a ser protegido debe estar relacionado con su token correspondiente, para garantizar su referencia única. La base de datos debe cumplir como mínimo con los siguientes controles de seguridad:

-**Aislamiento**. La base de datos maestra requiere ser aislada del resto de componentes del sistema.
-**Cifrado**. La BD debe contener información confidencial cifrada, cubriendo todos los controles relacionados con gestión de claves, custodia y copia de seguridad
-**Alcance**. El token y su referencia al dato confidencial deben ser válidos únicamente para un entorno limitado.
-**Autenticación, Autorización y Registro**. Únicamente el personal aprobado por la organización debe poder tener acceso a la base de datos maestra centralizada y su autenticación debe ser robusta, para prevenir accesos no autorizados.
-**Disponibilidad**. Se debe disponer de una estrategia de disponibilidad que garantice la continuidad de la operación en caso de problemas.

Servicio de cifrado/tokenización.

El proceso de tokenización consta de dos partes:

-Cifrado/descifrado de la información confidencial almacenada en la base de datos maestra centralizada.
-Asignación de un token a dicho dato cifrado.

Se requiere de un servicio que provea el cifrado/descifrado de la información almacenada en la base de datos y que gestione la asignación y referencia de tokens uno-a-uno con dicha información en los procesos de entrada y salida de datos confidenciales.

Interfaces de comunicación.

Son cualquier conexión entre componentes internos (componente de cifrado, componente de tokenización y llamadas a la base de datos) y aplicaciones externas como API's, Web Services, etc. Cualquier entrada o salida desde o hacia estas interfaces deben ser registradas, definiendo controles de acceso específico, seguridad en el transporte y acciones en caso de eventos sospechosos.

Resumen de (Acosta, 2010).

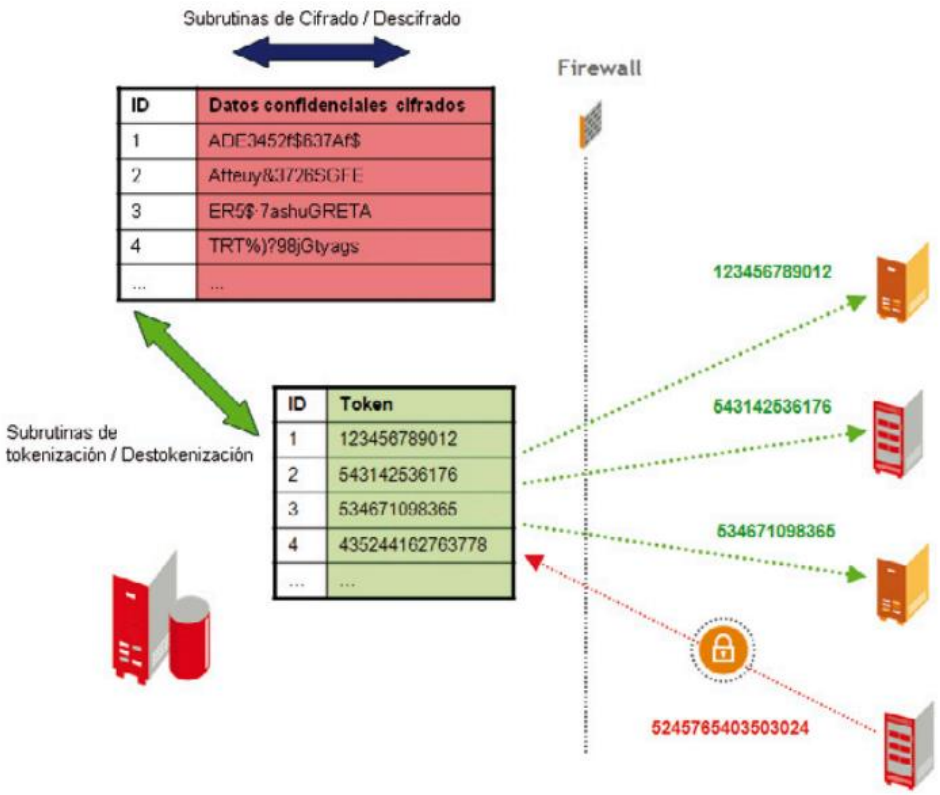


Ilustración 10 Esquema de tokenización.

.

Como se puede apreciar en la figura anterior la base de datos se encuentra aislada del resto de los sistemas y mantiene una estructura de pares “dato confidencial–token”, donde “dato confidencial” se almacena cifrado.

En esta parte del proyecto, ayude con el diseño e implementación del método de seguridad para la tokenización del sistema, el cual hace uso de algoritmos de cifrado y generación de tokens en un tiempo determinado.

Para hacer uso del API RESTful, el cliente que se conecta, debe tener acceso a la VPN de la empresa, además las peticiones hacia el servicio deben de ir en cierto formato, con la información del cliente cifrada mediante un script que se le da al cliente y sus identificadores únicos para transaccionar. El proceso se explica en el siguiente diagrama.

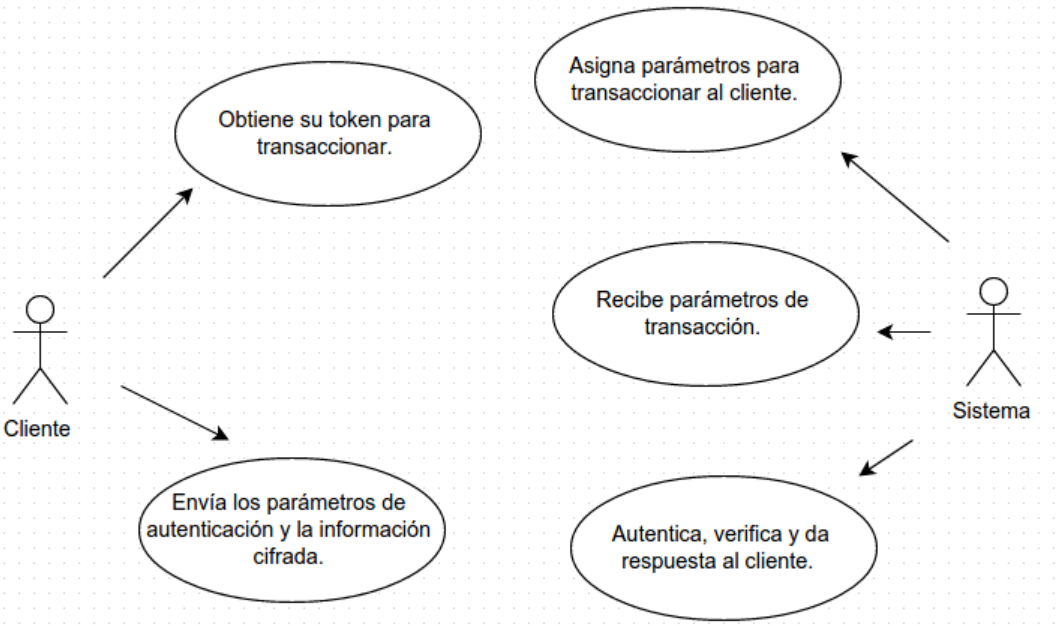


Ilustración 11. Diagrama de caso de uso, del proceso de tokenización.

4.1.2.4 Flujos de transacción.

La conexión al procesador de pagos, se realizó mediante dos procesos, el primero consistió en la conexión a un web service que valida el nip y el número de las tarjetas de la marca de la empresa. El otro web service al que se tuvo que conectar es la entidad financiera que utiliza el ISO 8583 para transaccionar. A continuación describo mi colaboración en ambos desarrollos.

4.1.2.5 Conexión al web service del nip.

En el flujo del servicio RESTful que se implementó, antes de poder transaccionar con el procesador de pagos, se tiene que validar que el número y nip de la tarjeta sean válidos. Para poder lograr esto, se tuvo que consumir un web service que se encarga de dichas validaciones.

Mi labor en esta parte del proyecto consistió en lograr la conexión con el web service del nip y la tarjeta. Este web service está desarrollado bajo SOAP (Simple Object Access Protocol), el cual es un protocolo de transferencia de datos a través de XML.

Para lograr la conexión con el web service, primero tuve que aprender cómo se realizaba una conexión SOAP con PHP, con ello encontré el concepto de WSDL (Web Services Description Language), el cual es un formato estándar escrito en XML, que sirve para describir un web service, obtener su ubicación y saber que operaciones o métodos permite.

Una conexión SOAP mediante PHP se realiza de la siguiente manera:

$clienteSoap = new SoapClient("IP/algun.wsdl");

Esta instrucción inicializa el constructor de la clase SoapClient, para poder realizar la llamada al servicio. Se pasa como argumento la dirección del wsdl del servicio a conectar.

$respuesta = $clienteSoap->_soapCall("Metodo", array("parametros"));

Con el método _soapCall, se hace una llamada a un método específico del web service, pasando sus argumentos, (si es que los requiere), como un arreglo de parámetros.

El web service SOAP devuelve un código de respuesta, que permite saber si la tarjeta es válida o no, o si ésta se bloqueó por número de intentos fallidos. Posteriormente si la tarjeta es válida, se continúa con el flujo del sistema hacia la conexión con el procesador de pagos.

4.1.2.6 Conexión al procesador de pagos.

Si la tarjeta es validada correctamente, por el servicio SOAP, continúa el proceso de transacción hacia el procesador de pagos. Dicho sistema trabaja, como la mayoría de los procesadores de pagos, con el estándar ISO 8583, éste estándar se utiliza para las transacciones financieras a través de una red, lo ocupan: POS (Puntos de Venta), cajeros automáticos (ATM), y el comercio electrónico (e-commerce).

La conexión a este sistema, la realizamos un compañero y yo. Yo me encargue de generar los scripts para poder enviar e interpretar la mensajería del ISO 8583, mi compañero realizó la conexión al web service mediante sockets.

Para poder realizar una transacción mediante el ISO 8583, se debe generar un mensaje de **requerimiento de autorización financiera**, el cual es un bitmap, con todos los campos necesarios para transaccionar como: la versión del ISO que se va a utilizar, el tipo de mensaje, el importe, el identificador de la terminal; el número de la tarjeta y su fecha de expiración, entre otros. Posteriormente se debe interpretar la **respuesta al requerimiento de autorización financiera**, el cual también es un bitmap, que regresa el procesador de pagos, la cual indica si la transacción fue aprobada o no.

Con el ISO 8583, también se pueden realizar reversas, es decir, la cancelación de transacciones, dichos procedimientos trabajan de manera similar a una transacción, primero se envía un requerimiento de reversa y se recibe una respuesta. En la siguiente figura, se representa de manera general, el flujo de transacción completo del servicio web.

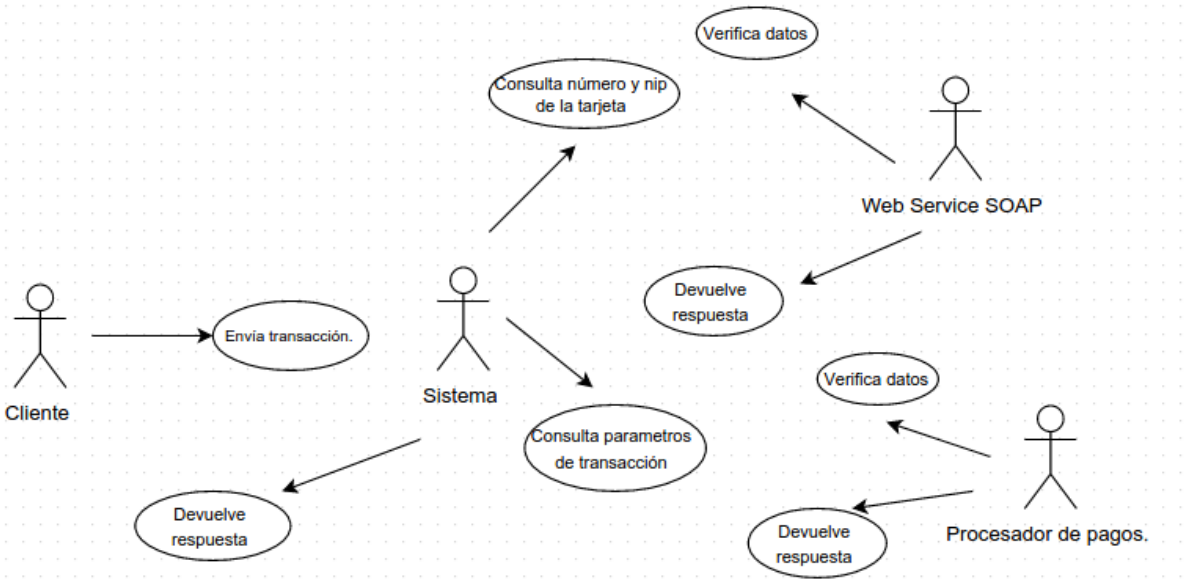


Ilustración 12 Diagrama de caso de uso del flujo de transacción del sistema.

4.2 SEGUNDA ETAPA.

4.2.1 Problemática.

En esta etapa del proyecto el objetivo principal consistió en crear una pasarela de pagos, que permitiera recibir como medios de pago las tarjetas de la empresa, las tarjetas de débito y crédito de marcas comerciales como Visa, Mastercard o American Express; recibir pagos de tiendas de conveniencia como Oxxo, Seven Eleven , etc.

Parte de los requerimientos de esta nueva etapa, fue la creación de cuentas para dos tipos de clientes, una **cuenta de tipo personal**, en la cual se registran clientes que sólo necesitan realizar pagos a través de la pasarela, la otra cuenta es de **tipo empresa**, en ella se registran las empresas o negocios que deseen ofrecer la pasarela de pagos, para que sus clientes puedan transaccionar a través de este medio.

Los principales retos en esta nueva etapa fueron los siguientes:

-Conocer los procesos de operación de negocio y técnicos, con los que trabajan las pasarelas de pagos como **Paypal, Netpay, Skrill**, etc.

-Implementar una plataforma antifraude como **Cybersource**, para garantizar la seguridad en las transacciones.

-Obtención de la certificación **PCI DSS**, la cual permite manejar y almacenar datos personales sensibles de los clientes, siguiendo ciertas normas de seguridad.

-Conexión con el procesador de pagos **PROSA**, para procesar pagos con tarjetas de crédito y débito de las marcas comerciales.

-Rediseño total a la base de datos y del web service que se había desarrollado en la primera etapa.

-Migración del sistema anterior a un framework en PHP (Laravel), que permitiera una mayor escalabilidad, rendimiento y sobre todo acelerará los tiempos de desarrollo.

-Mejoras de los ambientes de desarrollo, ocupando control de versiones (GIT) y adaptándose a metodologías ágiles, como Scrum.

4.2.2 Flujo general de transacción.

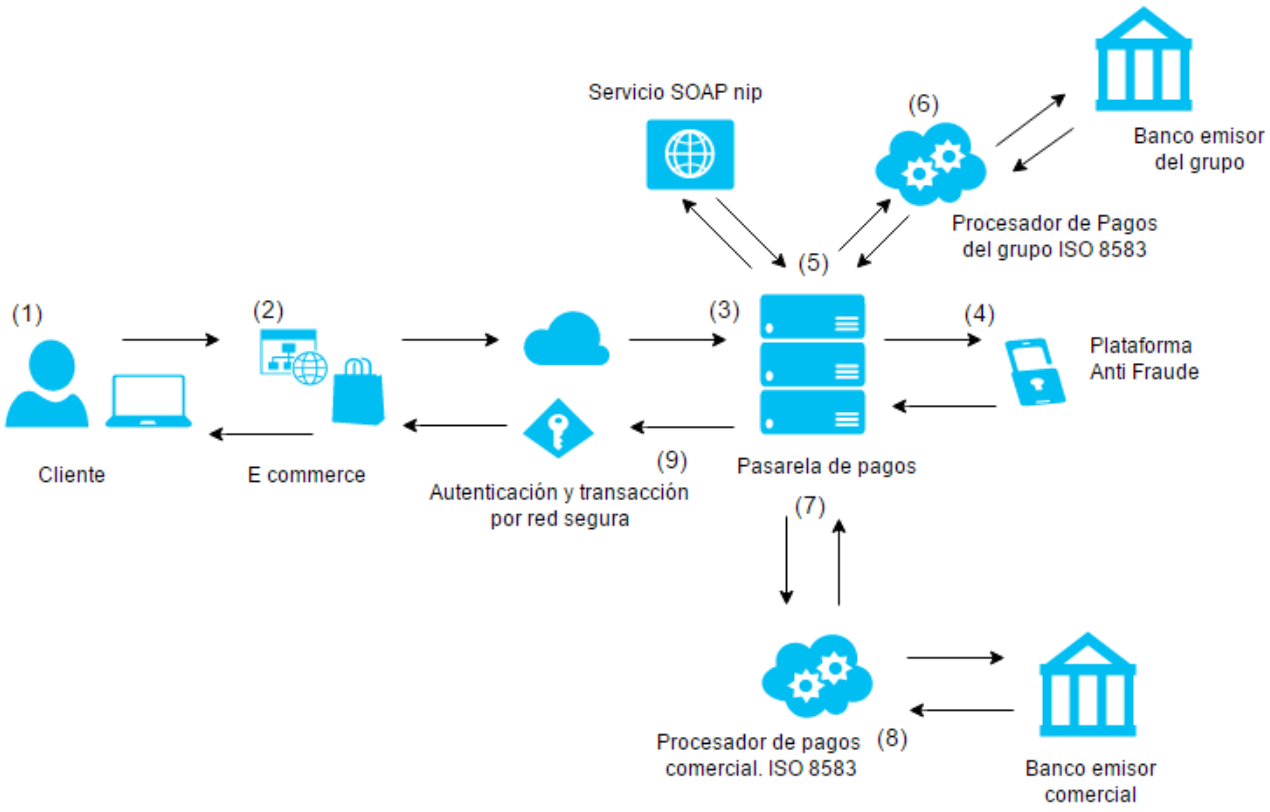


Ilustración 13 Flujo general de transacción de la plataforma de pagos.

1. El cliente con cuenta de tipo persona ingresa al e-commerce de la cuenta tipo empresa, llena su carrito de compras, elige su método de pago; se autentica, ingresa los datos de su tarjeta y envía solicitud de pago.
2. Desde el e-commerce se envía el carrito de compras y los parámetros de autenticación.
3. La pasarela de pagos, recibe la orden de compra, auténtica la orden de compra y verifica el método de pago.
4. La pasarela de pagos verifica con la plataforma antifraude, si la tarjeta es segura.
5. Si el método de pago es con tarjeta del grupo, la pasarela de pagos consume servicio de validación del nip y posteriormente se envía solicitud al procesador de pagos del grupo.
6. El procesador de pagos del grupo envía la solicitud de compra al banco emisor del grupo.
7. Si el método de pago es con tarjeta comercial, se envía solicitud de pago al procesador de pagos comercial (PROSA).
8. El procesador de pagos comercial envía la solicitud de compra al banco emisor.
9. Se regresan respuestas de la transacción.

4.2.3 Flujo general del registro de clientes.

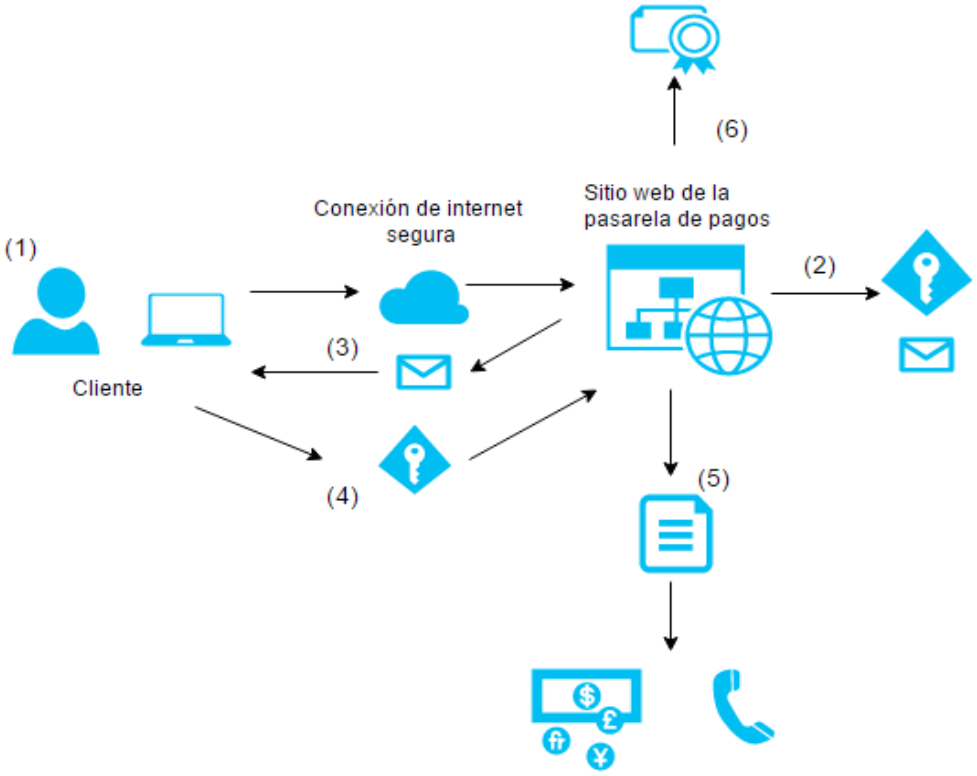


Ilustración 14 Flujo general del registro y alta de clientes.

1. El cliente entra al sitio público de la pasarela de pagos y selecciona un tipo de cuenta (personal | empresa).
2. El cliente ingresa correo, contraseña y dos preguntas de seguridad para recuperación de su cuenta.
3. Se envía un correo de verificación al correo del cliente. Éste da click en el link de verificación y es redirigido al login.
4. El cliente ingresa al sistema.
5. El cliente completa su registro, ingresando sus datos comerciales y de contacto.
6. La cuenta es activada. El cliente ya puede transaccionar.

4.2.4 Rediseño de la base de datos.

La base de datos de la primera etapa, tuvo que cambiar por completo debido a los nuevos requerimientos. A continuación menciono los cambios más importantes que se tuvieron que diseñar:

-Se adaptó el registro de cuentas tipo personal y empresa.

-De la cuenta tipo empresa se dio la posibilidad que una cuenta de este tipo, pudiera tener usuarios asociados, llamados operadores, a los cuales el titular de la cuenta puede asignar permisos y privilegios para acceder a los recursos y así ayudarlo a administrar la cuenta.

-Las cuentas de tipo empresa, se catalogan por su **sector de negocio** como: retail, travel, services, etc.

-Las cuentas de tipo empresa, pueden elegir entre los diferentes métodos de pagos, para ofrecer a sus clientes.

-Ambos tipos de cuenta pueden asociar sus tarjetas de crédito y débito a la cuenta.

-El sistema de comisiones se modeló, cumpliendo las necesidades del negocio.

-Las órdenes de compra se catalogaron como transacción de tarjetas comerciales y tarjetas de la empresa. Separando sus flujos, dependiendo de los servicios que se consulta.

-Se reestructuraron los catálogos de códigos de respuesta del sistema y de los servicios externos que se consumen.

-Los usuarios titulares de la cuenta tienen que asociar preguntas de seguridad al momento de registrarse, para poder recuperar sus datos de acceso como password y username.

-Se mejoró el registro de logs e historiales de todo el sistema.

Participe activamente en el diseño de la nueva base de datos, aplicando todos los conocimientos que aprendí en mi curso de base de datos de la facultad y de mi trabajo anterior. Puse en práctica el correcto modelado de una base de datos como presento a continuación.

4.2.4.1 Metodología empleada para la base de datos.

Un consejo invaluable de mis estudios en la facultad, es el hecho de pensar antes de actuar al resolver un problema. Esto lo vi reflejado en el diseño de la base de datos, el cual es muy importante ya que si se tiene un buen diseño, las ventajas pueden ser las siguientes:

-El sistema se vuelve flexible a evolucionar y adaptarse a las futuras necesidades de la organización.
-El modelo representa fielmente las operaciones de la organización.
-El modelo es independiente de la implementación física.
-La fase de desarrollo se agiliza y se evitan dolores de cabeza futuros.

En el diseño de bases de datos se debe analizar a detalle los requisitos del sistema a modelar, es decir la abstracción del problema del mundo real. Para esto antes se tuvo que investigar a fondo, las reglas del negocio de una pasarela de pagos. Una vez que se tienen bien estudiadas y definidas las reglas del negocio, se procede a utilizar el **modelo entidad-relación** para crear el modelo conceptual, que a continuación describo.

4.2.4.2 Modelo entidad-relación.

El Modelo Entidad-Relación (E-R) fue propuesto por Peter Chen en 1976 para la representación conceptual de los problemas del mundo real. Este modelo de datos representa los datos utilizando grafos y símbolos gráficos, además de tablas para la representación de los datos y sus relaciones.

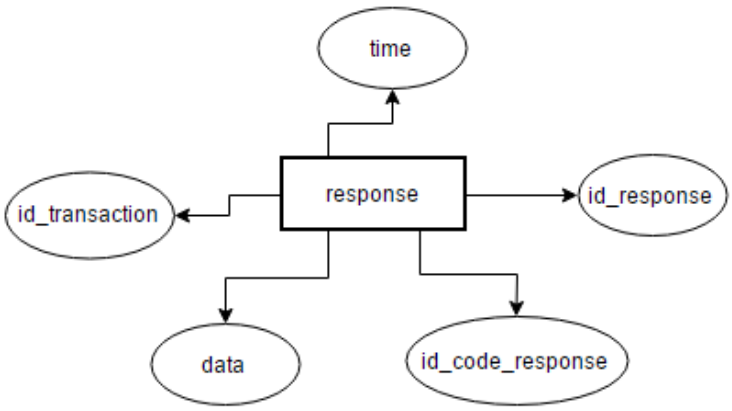
Conceptos básicos del modelo E-R:

1. **Entidad**: Es un objeto del mundo real que tiene interés para la organización. Por ejemplo, la entidad USUARIO del sistema o la entidad CLIENTE de una cuenta. Se representan con rectángulos con el nombre en el interior.

2. Conjunto de Entidades: Es un grupo de entidades del mismo tipo, y no tienen que ser conjuntos disjuntos, es decir, puede haber una entidad que pertenezca a varios conjuntos de entidades a la vez. Por ejemplo, el conjunto de entidades CLIENTE.

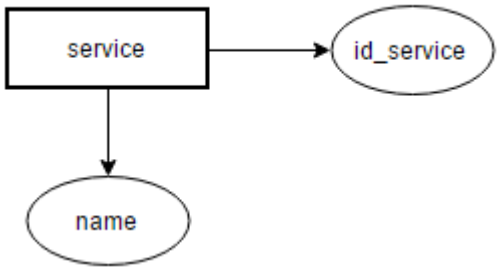
3. Entidad Fuerte: Es una entidad que no depende de otra entidad para su existencia. Por ejemplo, la entidad PAYMENT METHOD es fuerte pues no depende de otra para existir como entidad, mientras que la entidad PHONE es una entidad débil pues necesita a la entidad CLIENT para existir.

5. **Atributos o Campos:** Son las unidades de información que describen propiedades de las entidades. Por ejemplo, la entidad RESPONSE posee los atributos: time, date, id_response, id_transaction, id_code_response y data. Se representan mediante una elipse con el nombre en el interior.



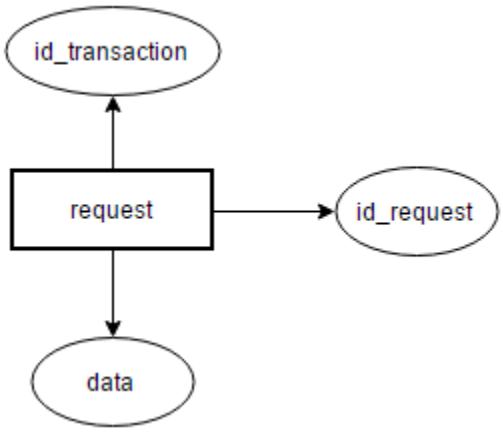
5. Dominio: Es el conjunto de valores permitidos para cada atributo. Por ejemplo, el dominio del atributo id_transaction del ejemplo anterior puede ser el conjunto de cadenas de texto cuyo componentes sean solo números.

6. Identificador o Superclave: Es el conjunto de atributos que identifican de forma única a cada entidad. Por ejemplo en la entidad SERVICE, el atributo que identifica de manera única a la entidad es id_service.



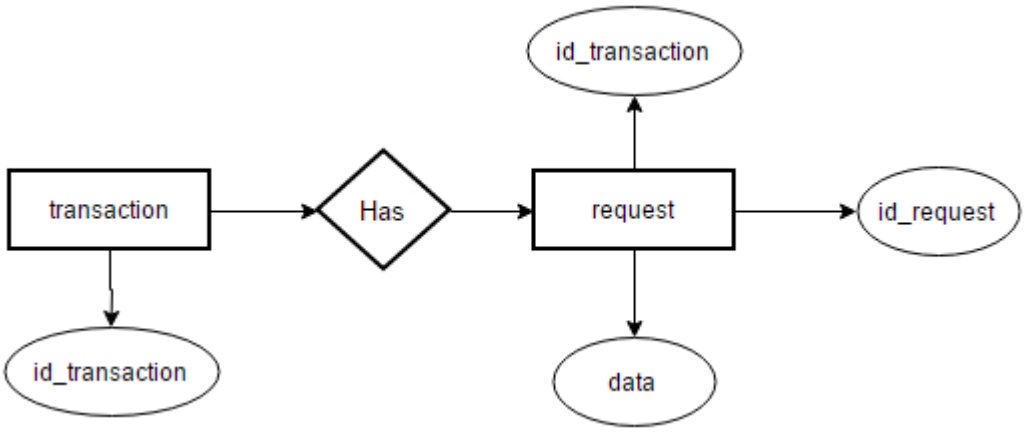
7. Clave Candidata: Es cada una de las superclaves formadas por el mínimo número de campos posibles. En el ejemplo anterior la clave candidata sigue siendo id_service, ya que no se cuenta con otra clave candidata.

8. Clave Primaria o Clave Principal (Primary Key): Es la clave candidata seleccionada por el diseñador de la BD para identificar a cada entidad. Una clave primaria no puede tener valores nulos (vacíos), ha de ser sencilla de crear y no ha de variar con el tiempo. El atributo o conjunto de atributos que forman parte de la clave primaria se representan subrayados. En el siguiente ejemplo, la clave primaria de la entidad REQUEST es id_request.



9. Clave Ajena o Clave Foránea (Foreign Key): Es el atributo o conjunto de atributos de una entidad que constituyen la clave primaria de otra entidad. Las claves foráneas representan las relaciones entre entidades. En el ejemplo anterior, la llave foránea seria id_transaction ya que la entidad REQUEST y TRANSACTION están relacionadas.

10. Relación: Es una asociación entre diferentes entidades. Se representan mediante un rombo con su nombre, un verbo, en su interior. En el siguiente esquema se muestra la relación entre la entidad TRANSACTION y REQUEST



Resumen de (El modelo entidad-relación).

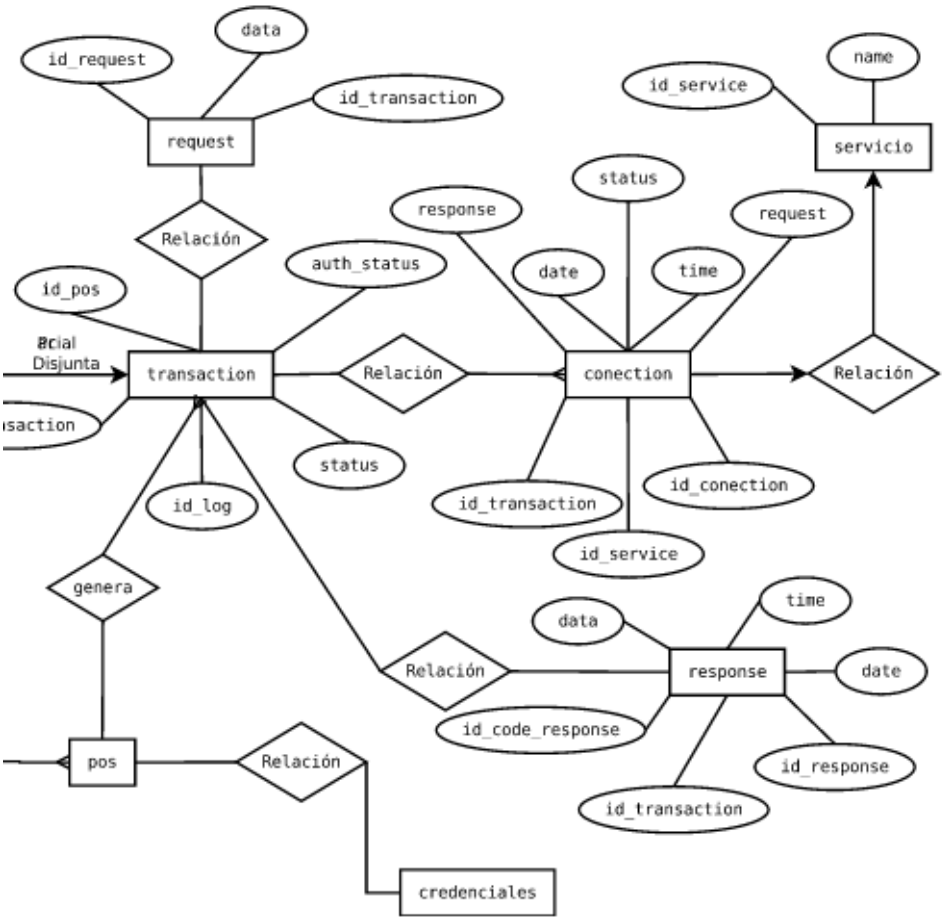


Ilustración 15 Fragmento del diagrama E-R simplificado.

Como podemos ver, el modelo entidad-relación, nos permite obtener un modelo semántico del problema a resolver. Con él podemos describir los niveles conceptual y externo de los datos del sistema además es independiente de cualquier DBMS.

4.2.4.3 Modelo relacional.

Cuando ya se tiene un buen modelo conceptual, se procede a describir el modelo lógico de la base de datos, para ello se puede utilizar el modelo relacional, el cual fue propuesto por Codd en 1970 y hoy en día continúa siendo el más utilizado, debido a su simplicidad y poder.

El modelo relacional comenzó con el uso de la teoría de relaciones en matemáticas y se adaptó para su uso en la teoría de bases de datos. El mismo tipo de desarrollo teórico de la materia, completada con notación formal, definiciones, teoremas y pruebas que usualmente se encuentran en matemáticas, se pueden aplicar a las bases de datos usando este modelo.

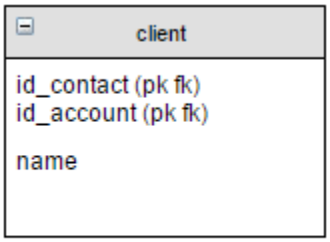
En este modelo, las entidades se representan como relaciones, que se representan físicamente como tablas o arreglos bidimensionales, y los atributos como columnas de dichas tablas. Las relaciones también se pueden representar como relaciones o tablas, pues este modelo considera una relación como un tipo especial de entidad. (Catherine M, 2009).

4.2.4.4 Transformación del modelo E-R al modelo relacional.

El mapeo del modelo E/R a relacional es bastante sencillo, así lo explica (Sotorrio O ,2010) y es ejemplificado con fragmentos del diseño de la base de datos del sistema que se desarrolló.

Transformación de entidades débiles

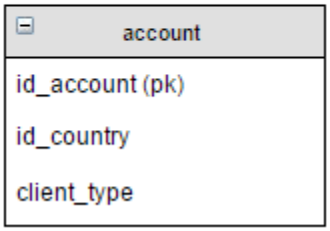
Todas las entidades del modelo E/R se convierten en tablas en el modelo relacional. Las entidades débiles también se transforman en tablas pero su clave primaria se compone de la unión de ésta con la clave de la entidad fuerte a la que pertenece.



Transformación de las relaciones (1:1)

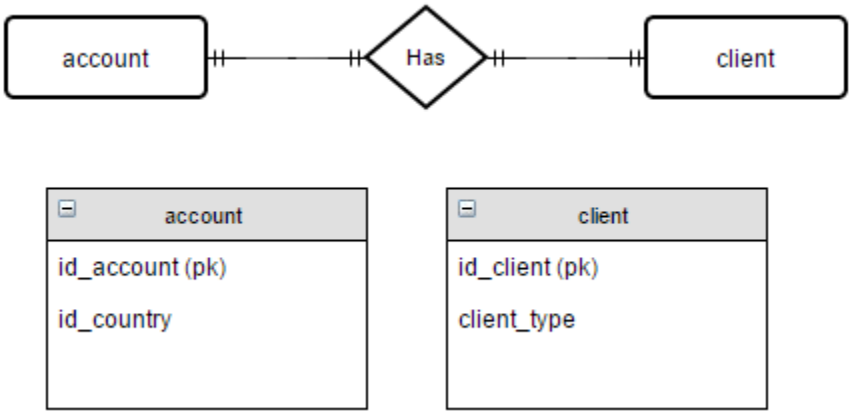
- Mismo Identificador.

Si las dos entidades tienen el mismo identificador se transforman en única tabla que contendrá este identificador como clave primaria y los atributos de ambas entidades.



- Diferente Identificador.

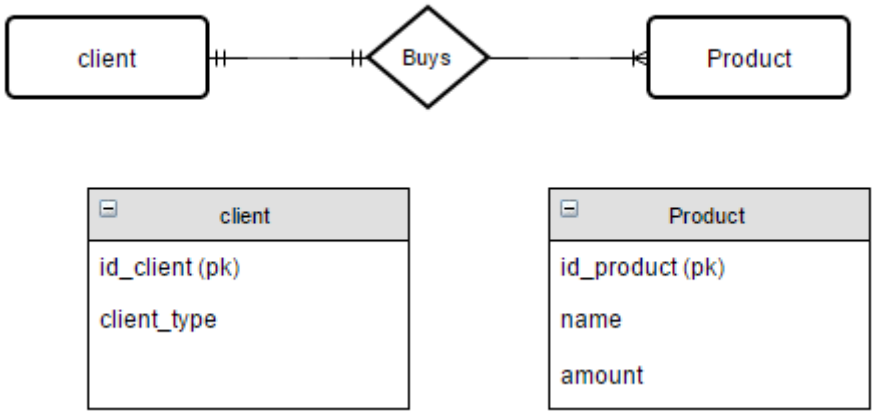
Cuando tienen diferente identificador cada entidad se convierte en una tabla con su identificador como clave primaria y como clave ajena el identificador de la otra entidad.



Transformación de relaciones (1:N).

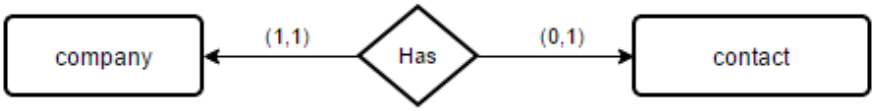
- Cardinalidad mínima uno.

Si en la relación la entidad que participa con cardinalidad máxima igual a uno, lo hace también con cardinalidad mínima igual a uno, cada entidad se transforma en una tabla con su respectiva clave primaria. La tabla, que participa con cardinalidad N, tendrá como clave ajena la clave primaria de la otra tabla, así como los atributos de la relación.



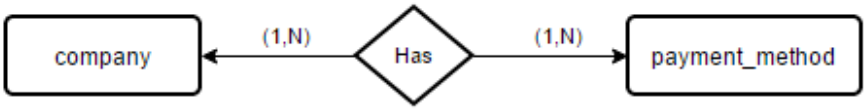
- Cardinalidad mínima cero.

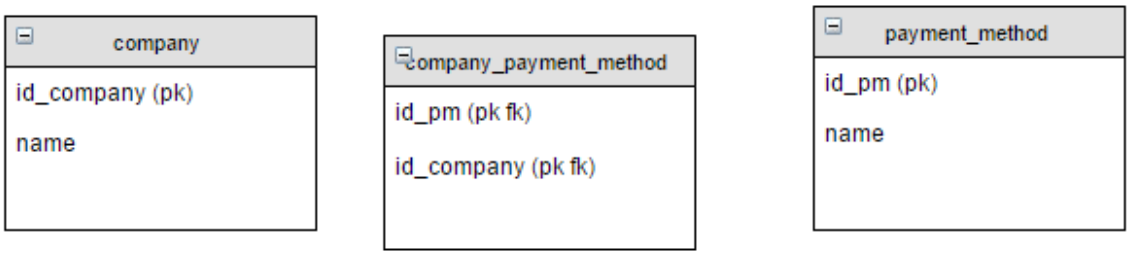
En este caso cada entidad se transforma en una tabla con su respectiva clave primaria. Se añade otra tabla que representa la relación, cuya clave primaria será la clave primaria de la tabla con cardinalidad N. Y tendrá como clave ajena la clave primaria de la tabla con cardinalidad uno.



Transformación de las relaciones (N:N)

Cada entidad se transforma en una tabla con su respectiva clave primaria. Se añade una tabla para la relación con los atributos de esta y como clave primaria la composición de las claves de las otras entidades.





4.2.5 Migración del sistema a Laravel.

Con los nuevos requisitos, para crear la plataforma de pagos, fue necesario buscar una alternativa para agilizar el desarrollo. Yo fui uno de los partidarios en escoger el framework laravel como una buena alternativa, a continuación expongo las ventajas que se tienen al utilizar este framework:

1. Se reduce considerablemente los costos y tiempos de desarrollo y el mantenimiento.
2. La curva de aprendizaje es rápida y la documentación es muy amplia y fácil de entender.
3. Es flexible y adaptable con el patrón de diseño MVC.
4. Es modular y su gestor de dependencias ofrece muchas alternativas robustas y seguras.
5. Cuenta con un gestor de caché.
6. Una de sus grandes características es su ORM Eloquent, el cual mapea la base datos a objetos. Con lo que se agiliza la manipulación de la base de datos.
7. Migraciones. Las cuales permiten tener un control de versiones sobre nuestra base de datos, se puede modificar la base de datos de manera rápida y sencilla.
8. Seeds. Permiten poblar la base de datos ágilmente, esto ayuda mucho cuando estás desarrollando y necesitas tener datos de prueba en la base, para probar tu aplicación.
9. Laravel utiliza el sistema de plantillas Blade, el cual permite tener un código más limpio y ordenado.
10. En materia de seguridad laravel es muy robusto y cuenta con su propio sistema de tokenización.

Las actividades que realicé utilizando laravel fueron las siguientes:

4.2.5.1 Migraciones de la base de datos.

Realicé la migración de las tablas del modelo relacional que se diseñó, las migraciones permiten modificar la base de datos de manera sencilla, sin ocupar directamente sentencias SQL. A continuación presento un ejemplo sencillo de cómo crear una migración en laravel.

Para crear una migración se debe ejecutar el siguiente comando:

php artisan make:migration create_users_table.

Lo que hace el comando anterior es crear un archivo en php, el cual es una clase que hereda de la clase Migration, en dicho archivo debes poner la creación de tablas que forman parte de tu base de datos. Esto se hace a través de un esquema como muestro a continuación:

```
Schema::create('users', function($table)
{
  $table->increments('id');
});
```

La instrucción anterior genera una tabla llamada **users**, instanciando una **closure**, con la variable $table, la cual se va a utilizar para generar los atributos de la tabla. Cuando se crea una migración, se debe poner la instrucción que elimine cada esquema (tabla), que se ha creado:

```
Schema::dropIfExists('users');
```

Esto es necesario para que se pueda hacer un rollback a la migración. Y así volverla a ejecutar.

php artisan migrate:rollback

4.2.5.2 Controladores y modelos del módulo de clientes.

Estuve a cargo de la creación de los controladores y modelos del módulo de clientes. A continuación presento un ejemplo sencillo, de cómo implementar un controlador en laravel.

Ejemplo de un controlador básico.

```
<?php namespace App\Http\Controllers;
use App\Http\Controllers\Controller;
class UserController extends Controller {
  /**
   * Mostrar el perfil de un usuario dado.
   *
   * @param  int  $id
   * @return Response
   */
  public function showProfile($id)
  {
     return view('user.profile', ['user' => User::findOrFail($id)]);
  }

}
```

En el ejemplo podemos apreciar, que un controlador hereda de la clase base Controller y en este caso el controlador UserController tiene un método que se llama showProfile, el cual es invocado desde el ruteador de peticiones del framework. El método showprofile regresa una vista se que llama user.profile, pasando como parámetro una instancia de un usuario.

4.2.5.3 Registro de cuenta.

Realicé dos flujos de registro, para los dos tipos de cuenta (persona y empresa). Uno de estos flujos es el registro express, en éste el cliente entra al portal público del procesador de pagos, y crea una cuenta ingresando su correo, contraseña, y elige una pregunta de seguridad. Cuando el cliente realiza dicho registro, se le envía un correo (el que registró), con un link de verificación, para que pueda verificar su cuenta. Si el cliente accede al link, se le redirige al login para que pueda ingresar al sistema. El cliente debe completar sus datos comerciales y de contacto para que su cuenta quede activada y así pueda empezar a utilizar el sistema. Estos flujos los explico de manera general en los siguientes diagramas:

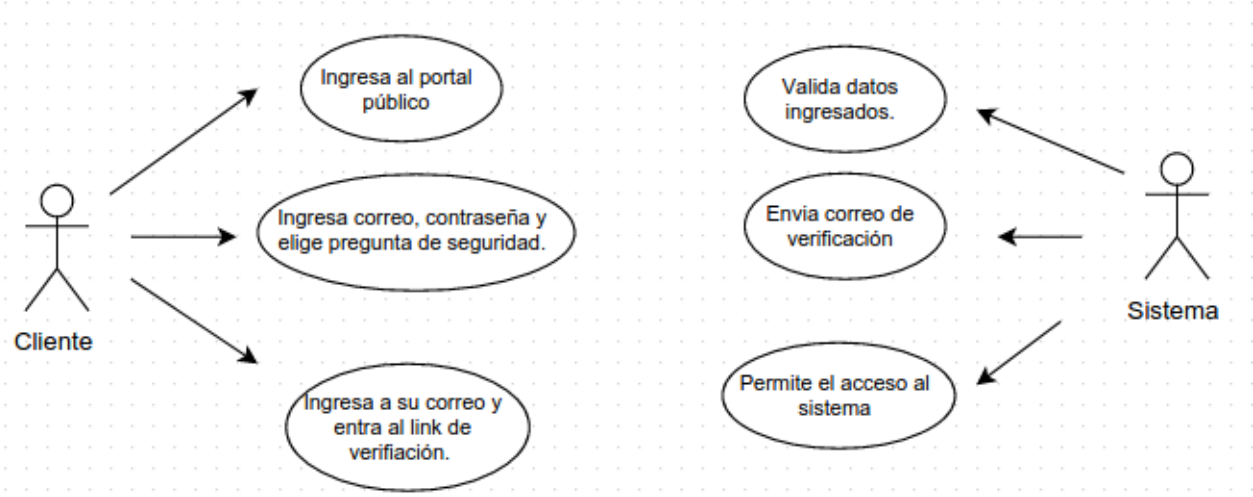


Ilustración 16 Diagrama de caso de uso del registro express de la cuenta.

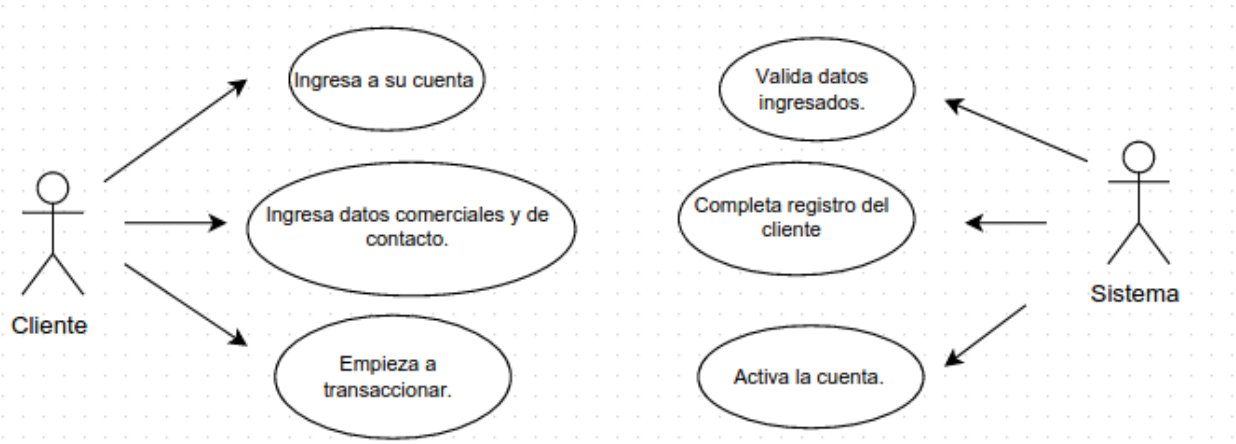


Ilustración 17 Diagrama de caso de uso de la activación de la cuenta.

4.2.5.4 Recuperación de datos de acceso.

Si el cliente olvida o extravía su contraseña, puede sustituir ésta ingresando al login y solicitar el cambio de contraseña, el flujo lo explico en el siguiente diagrama.

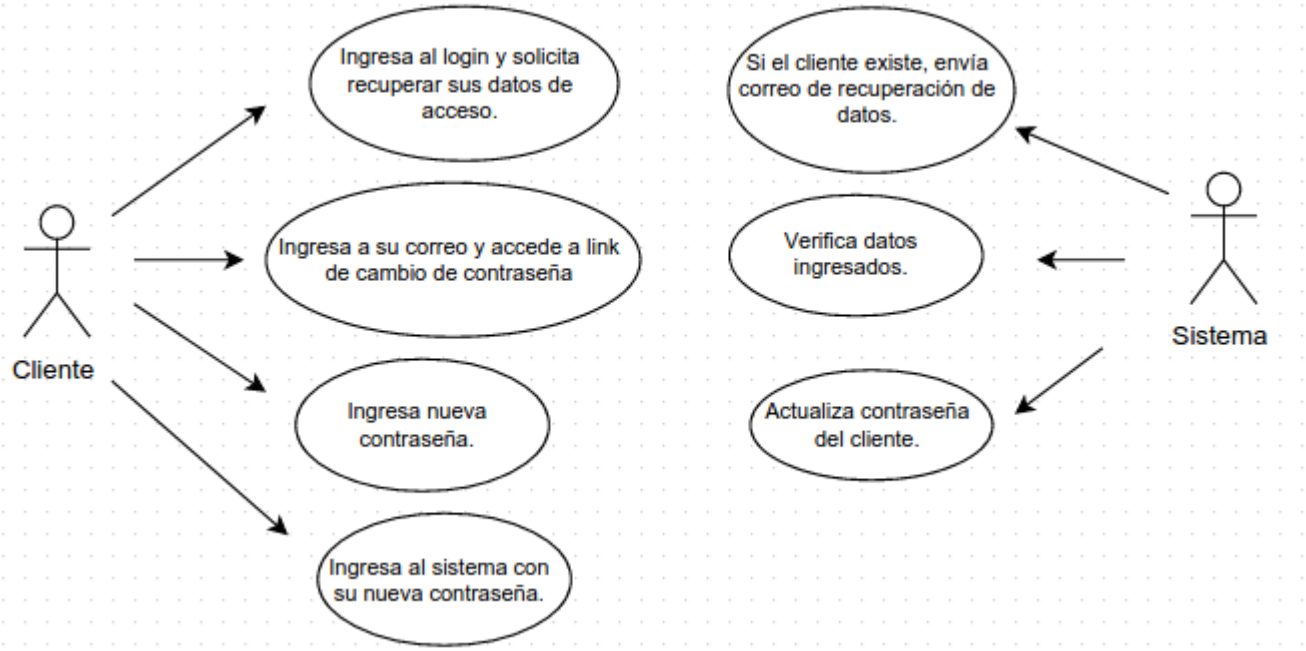


Ilustración 18 Diagrama de caso de uso de la recuperación de contraseña.

4.2.5.5 Reportes de transacciones.

Realicé el desarrollo de los reportes de las transacciones del sistema, esto incluye los filtros de transacciones por día, fecha, cuenta, totales; transacciones rechazadas, aceptadas, reversas, etc.

Este desarrollo se dividió en dos partes, la primera corresponde a las vistas que tiene acceso el propietario de la cuenta, es decir, sus transacciones que ha realizado. La segunda parte corresponde a las vistas a las que puede acceder el usuario comercial, el cual se encarga de gestionar todas las cuentas del sistema, asignar las comisiones respectivas y generar los reportes totales.

DataTables para generar las vistas de transacciones.

Utilice este plug-in para JQuery, el cual es una herramienta flexible y avanzada para manipular la interacción de tablas HTML.

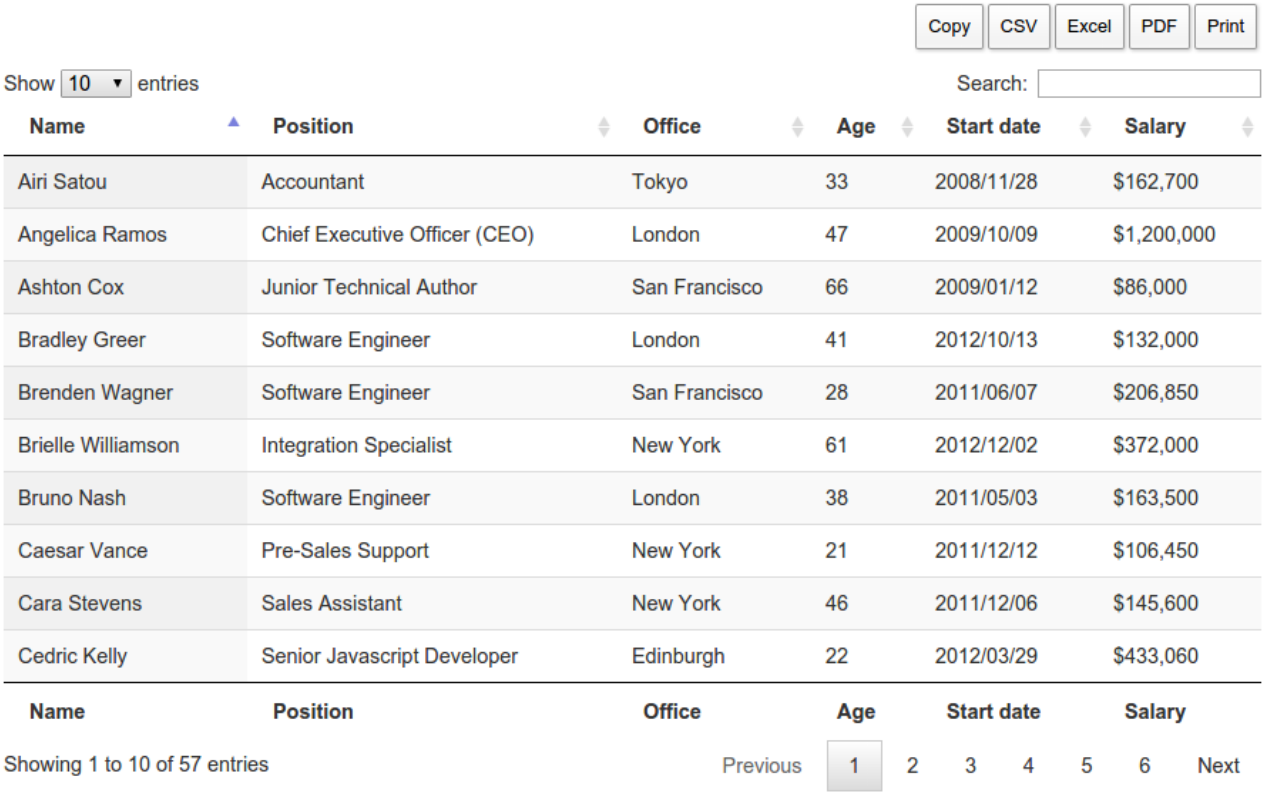


Ilustración 19 Datatable de ejemplo.

Algunas de las versatilidades que te permite esta herramienta son las siguientes:

-Te permite mostrar el total de registros o en segmentos definidos.
-Cuenta con un buscador muy útil, para filtrar la información.
-Puedes exportar tus tablas a formato: csv, excel o pdf con un solo click.
-Permite ordenar las columnas por orden alfabético y de menor a mayor, según sea el caso, de manera muy intuitiva para el usuario.
-Tiene un paginador, para no visualizar toda la información abarcando toda la página.
-Se pueden cambiar los estilos de la tabla, dependiendo de las necesidades.

4.2.6 Certificación PCI DSS.

Las pasarelas de pagos, al transaccionar con tarjetas de crédito y débito, deben cumplir con ciertos parámetros o normas de seguridad, para garantizar que los datos confidenciales de los dueños de las tarjetas, permanezcan seguros y no lleguen a manos de personas no autorizadas. El estándar de seguridad que se tiene que seguir es el PCI DSS.

Durante todo el desarrollo del proyecto, se tuvo que tener en cuenta cumplir cada uno de los puntos de la certificación, como a continuación presento:

1. El área de redes, se encargó de mantener una red segura, para transaccionar.
2. Las contraseñas del sistema, son robustas y no predeterminadas.
3. Se aseguró la protección de los datos almacenados de los propietarios de las tarjetas.
4. Las políticas de seguridad de la empresa se actualizaron para cumplir con la certificación.
5. Se hicieron pruebas de Pent Test.
6. En cada módulo de la aplicación se verificó el código para reducir el riesgo de vulnerabilidades.
7. Se monitoriza todo acceso a los recursos de la red y datos de los tarjetahabientes.
8. El software se mantiene actualizado con los últimos parches de seguridad.

4.2.7 Procesador de pagos.

Como parte de los nuevos requerimientos de esta etapa, se tuvo que lograr la comunicación con el procesador de pagos que maneja las tarjetas comerciales de Visa y MasterCard, este procesador de pagos fue **PROSA** a continuación describo el proceso de autorización de pagos en línea que se tuvo que seguir.

Autorizaciones de pagos en línea.

Una autorización en línea significa que cuando tú envías una orden de pago utilizando una tarjeta de crédito, recibes inmediatamente una confirmación de la disponibilidad de fondos. Si los fondos están disponibles, el banco emisor aplica el cobro a la cuenta del cliente. La mayoría de las tarjetas de crédito son procesadas en línea.

Las autorizaciones en línea expiran con el banco emisor después de un cierto periodo de tiempo si estas no han sido capturadas y establecidas. La mayoría de las autorizaciones caducan dentro de cinco a siete días. El banco emisor establece el periodo de tiempo.

Cuando la autorización expira con el banco emisor, el banco adquirente puede requerir que se le envíe una solicitud autorización que incluya un campo de captura en el mismo mensaje. (Credit Card Services Using the Simple Order API, 2015).

La siguiente figura representa el proceso de autorización en línea.

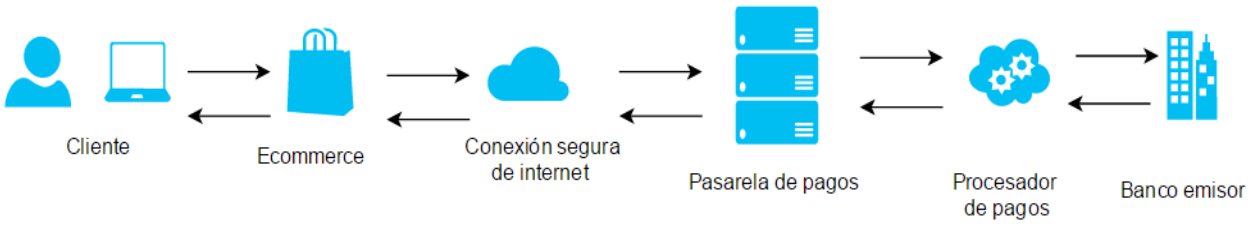


Ilustración 20 Proceso de autorización en línea.

1. El cliente ingresa el número de la tarjeta de crédito, fecha de expiración e información adicional de la tarjeta.
2. El e-commerce envía solicitud de autorización.
3. La pasarela de pagos, valida la información de la orden de compra y envía la petición de autorización al procesador de pagos.
4. El procesador de pagos envía la transacción al banco emisor de la tarjeta.
5. El banco emisor aprueba o declina la solicitud.

4.3 METODOLOGÍA SCRUM.

Durante el desarrollo de la segunda fase del proyecto, sugerí al líder de desarrollo que utilizáramos la metodología ágil Scrum, para agilizar nuestro tiempo de desarrollo y hacer entregables completos en corto plazo, a continuación describo dicha metodología.

De acuerdo con la página web Softeng:

Scrum es una metodología ágil y flexible para gestionar el desarrollo de software, cuyo principal objetivo es maximizar el retorno de la inversión para la empresa (ROI). Se basa en construir primero la funcionalidad de mayor valor para el cliente y en los principios de inspección continua, adaptación, auto-gestión e innovación.

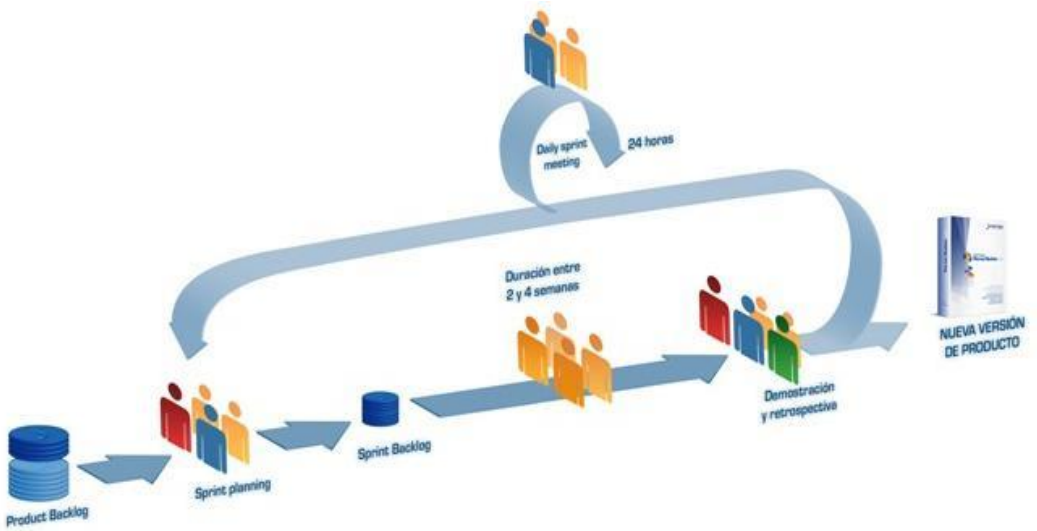


Ilustración 21. Metodología Scrum.

4.3.1 ¿Cuándo se utiliza?

Con la metodología Scrum el cliente se entusiasma y se compromete con el proyecto dado que lo ve crecer iteración a iteración. Asimismo le permite en cualquier momento realinear el software con los objetivos de negocio de su empresa, ya que puede introducir cambios funcionales o de prioridad en el inicio de cada nueva iteración sin ningún problema.

Esta metódica de trabajo promueve la innovación, motivación y compromiso del equipo que forma parte del proyecto, por lo que los profesionales encuentran un ámbito propicio para desarrollar sus capacidades.

4.3.2 Beneficios

-Cumplimento de expectativas: El cliente establece sus expectativas indicando el valor que le aporta cada requisito / historia del proyecto, el equipo los estima y con esta información el Product Owner establece su prioridad. De manera regular, en las demos de Sprint el Product Owner comprueba que efectivamente los requisitos se han cumplido y transmite se feedback al equipo.

-Flexibilidad a cambios: Alta capacidad de reacción ante los cambios de requerimientos generados por necesidades del cliente o evoluciones del mercado. La metodología está diseñada para adaptarse a los cambios de requerimientos que conllevan los proyectos complejos.

-Reducción del Time to Market: El cliente puede empezar a utilizar las funcionalidades más importantes del proyecto antes de que esté finalizado por completo.

-Mayor calidad del software: La metódica de trabajo y la necesidad de obtener una versión funcional después de cada iteración, ayuda a la obtención de un software de calidad superior.

-Mayor productividad: Se consigue entre otras razones, gracias a la eliminación de la burocracia y a la motivación del equipo que proporciona el hecho de que sean autónomos para organizarse.

-Maximiza el retorno de la inversión (ROI): Producción de software únicamente con las prestaciones que aportan mayor valor de negocio gracias a la priorización por retorno de inversión.

-Predicciones de tiempos: Mediante esta metodología se conoce la velocidad media del equipo por sprint (los llamados puntos historia), con lo que consecuentemente, es posible estimar fácilmente para cuando se dispondrá de una determinada funcionalidad que todavía está en el Backlog.

-Reducción de riesgos: El hecho de llevar a cabo las funcionalidades de más valor en primer lugar y de conocer la velocidad con que el equipo avanza en el proyecto, permite despejar riesgos eficazmente de manera anticipada.

4.3.3 Por qué utilizar una metodología ágil.

-Mejoran la calidad del trabajo.
-Ayuda a ofrecer un buen producto.
-Ayuda a tener al cliente más contento.
-El equipo de trabajo trabaja más cómodo.

La elección de una correcta metodología de desarrollo de software, es decisiva en el éxito o fracaso de un proyecto. Anteriormente se venía utilizando metodologías tradicionales, las cuales estaban saturadas de documentación provocando que fueran poco flexibles al cambio.

Hoy en día el escenario del desarrollo de software es dinámico y los requisitos cambian habitualmente, por lo que surge la necesidad de echar mano de metodologías ágiles, ligeras y versátiles.

Los criterios que se tomaron en cuenta para elegir entre una metodología ágil y una tradicional fueron los siguientes:

Criticidad: en general los proyectos críticos como los relativos a salud o plantas nucleares, etc, funcionan mejor con metodologías tradicionales. Nuestro proyecto no contaba con ese nivel de criticidad.

Tamaño del equipo: nuestro equipo estaba formado por 8 integrantes, por lo que se adaptaba perfecto para una metodología ágil.

Capacitación y experiencia del equipo: el equipo contaba con un porcentaje mayor al 35% en cuanto a experiencia.

Estabilidad de los requisitos: La volatilidad de los requisitos era alta, por lo que no se pudo adoptar una metodología tradicional.

Disponibilidad del cliente: el cliente contaba con dicha disponibilidad y estaba a favor de las metodologías ágiles.

4.3.4 Justificación de Scrum.

A continuación presento la justificación que se realizó, para elegir la metodología ágil Scrum para el proyecto.

Los principales puntos para escoger a Scrum frente a otra metodología ágil como Kanban fueron:

-El proyecto era grande con entregas incrementables.
-El backlog del producto estaba bien priorizado.
-El equipo era multidisciplinario.

-Los sprints se adaptaban a los tiempos de entrega.
-Sencillo de entender.
-El equipo era auto-organizado.
-Es rápido sin necesidad de planificaciones iniciales como Pert o diagramas de Gantt.
-El 60% del equipo ya contaba con experiencia previa con Scrum.

4.3.5 Scrum caso práctico.

Nuestros sprints tenían una duración de entre 3 y cuatro semanas. Al final de cada uno de estas iteraciones, entregamos una parte del producto completo y verificado por el equipo y el cliente.

Al principio de cada sprint, realizábamos una planificación del próximo sprint y se fijaba la duración de éste. Diariamente el equipo realizaba una reunión de seguimiento muy breve en la mañana. Para verificar su avance y resolver posibles conflictos. Básicamente cada miembro del equipo trataba de responder las siguientes preguntas.

• ¿Qué he hecho desde la última reunión de sincronización? ¿Pude hacer todo lo que tenía planeado? ¿Cuál fue el problema?

• ¿Qué voy a hacer a partir de este momento?

• ¿Qué impedimentos tengo o voy a tener para cumplir mis compromisos en esta iteración y en el proyecto?

Como recomendación tratábamos de hacer la reunión de pie, para que cada miembro del equipo no se relajara ni se extendiera demás en los detalles necesarios.

El plan del proyecto, consistió en una lista de requisitos priorizados del producto, el cliente se encargaba de priorizar los requisitos balanceando el valor que le aportaban respecto al coste y se dividían en iteraciones y entregas.

A continuación presento las actividades de la metodología Scrum que se siguieron:

Ilustración 22. Actividades del proceso de Scrum

Planificación de la iteración.

En esta etapa del sprint, nos reuníamos con el cliente (representante de la empresa), para establecer los requisitos de la entrega, Una vez que éstos eran definidos y priorizados. El equipo se reunía para estimar juntos el esfuerzo necesario para realizar cada tarea y de acuerdo a las habilidades de cada miembro del equipo se asignaban las actividades a realizar.

En el caso en que hubiera un cambio de contexto considerable para el proyecto, se terminaba el sprint antes de tiempo y se planificaba el siguiente.

Demostración de requisitos completados.

Aquí realizábamos una reunión de máximo 4 horas, para presentarle al cliente los requisitos completados en la iteración. El cliente podía hacer las adaptaciones necesarias de manera objetiva.

Retrospectiva.

Aquí analizábamos cómo fue nuestra manera de trabajar durante la iteración, qué cosas funcionaban bien y cuales debíamos de mejorar. Esto nos benefició porque incremento la productividad y el aprendizaje del equipo.

4.3.6 Herramientas utilizadas.

Lista de requisitos priorizada.

En ella representamos los requisitos y /o expectativas del cliente, respecto a los objetivos y entregas del proyecto.

Listas de tareas de iteración.

En ella despostamos los requisitos y tareas específicas para cada sprint, dividiendo el tablero en tareas pendientes, en ejecución y terminadas.

Gráfico de trabajo pendiente.

Es un gráfico de trabajo pendiente a lo largo del tiempo que muestra la velocidad a la que se está completando los requisitos. Permite extrapolar si el podrá completar el trabajo en el tiempo estimado.

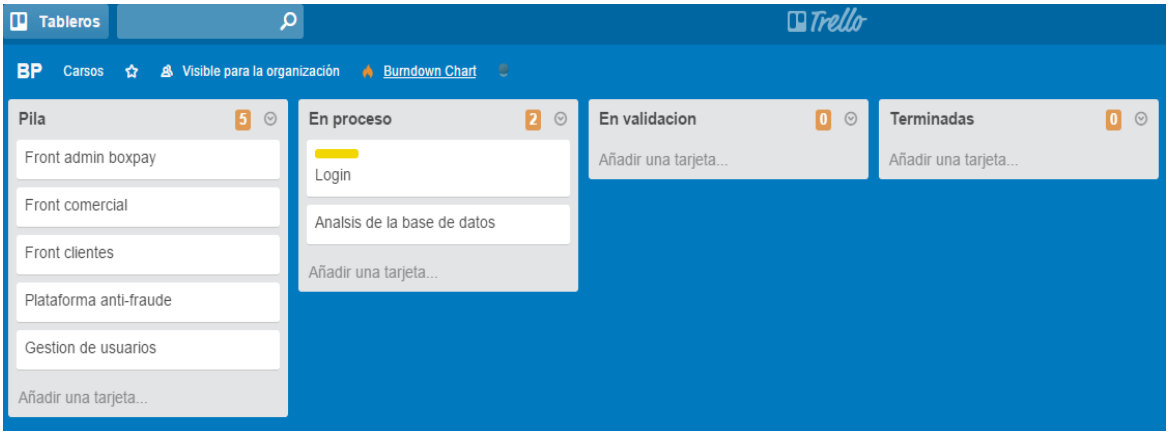


Ilustración 23. Tablero de actividades del sprint

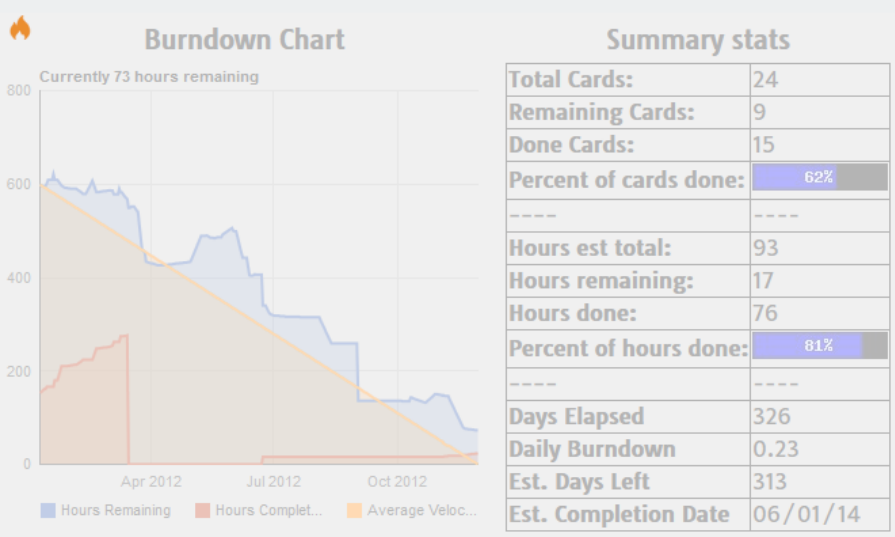


Ilustración 24. Gráfica Burndown

Para estas tres herramientas básicas ocupamos la plataforma **Trello**.

https://trello.com/

5 CONCLUSIONES.

Éste proyecto me permitió aplicar mis conocimientos y habilidades adquiridas durante la carrera, como el diseño de las bases de datos, la programación orientada a objetos, la capacidad de abstracción del mundo real y sobre todo el desarrollo del pensamiento lógico, crítico, objetivo y creativo que nos es fomentado durante nuestra estancia en la máxima casa de estudios.

Un aspecto que me agradó considerablemente, durante el desarrollo de éste proyecto, es que desde el principio tuve la libertad de proponer soluciones fiables, aportar mis conocimientos previos y compartir parte de mi experiencia laboral anterior para resolver los problemas.

Gracias a este proyecto, pude aprender más sobre el sector de negocio de las transacciones financieras en línea. También pude desarrollar y mejorar mis capacidades de interacción con los clientes y entender sus necesidades desde su punto de vista.

Por ultimo quisiera decir, que mi educación en la facultad de ingeniería me ha dado mucha seguridad para desenvolverme en el mundo laboral, con base en lo anterior me gustaría recalcar que el ingeniero en computación debe estar en constante actualización, sobre las nuevas metodologías y tendencias de la industria. Es una carrera donde cada día la revolución tecnológica genera un impacto directo, donde si no te actualizas o simplemente no te gusta mejorar, te rezagas.

Por mi raza hablará el espíritu. José Vasconcelos (1882-1959).

6 REFERENCIAS.

-Fonseca, S. D, Pérez, R. W. & Faurés, M. M. (2013, 29 de agosto). Pasarela de pagos para la seguridad de transacciones bancarias en línea. Editorial online 3ciencias. Recuperado el 30 de abril de 2015, de http://www.3ciencias.com/wp-content/uploads/2013/08/pasarela-de-pagos.pdf.

-B2C Ecommerce Climbs Worldwide, as Emerging Markets Drive Sales Higher. (2013, 27 de junio). eMarketer Inc. Recuperado el 29 de abril de 2015 de http://www.emarketer.com/Article/B2C-Ecommerce-Climbs-Worldwide-Emerging-Markets-Drive-Sales-Higher/1010004.

-Figura 1. Huyen N. (2012, 6 de Diciembre). 10 Safe and Popular Gateways for Online Payment Processing. InstantShift. Recuperado el 30 de abril de 2015 de http://www.instantshift.com/2012/12/06/10-safe-and-popular-gateways-for-online-payment-processing/.

-Sistemas de pago, Banco de México. Recuperado el 10 de abril de 2015 de http://www.banxico.org.mx/divulgacion/sistemas-de-pago/sistemas-pago.html.

-Suman K. (2010, 7 de agosto). Introduction to ISO 8583. Code Project. Recuperado el 29 de abril de 2015 de http://www.codeproject.com/Articles/100084/Introduction-to-ISO.

-Rouse M. (2009, mayo). PCI DSS (Payment Card Industry Data Security Standard). TechTarget. Recuperado el 02 de mayo de 2015 de http://searchfinancialsecurity.techtarget.com/definition/PCI-DSS-Payment-Card-Industry-Data-Security-Standard.

-Fraud risk management. A guide to good practice. (2009, junio).CIMA (Chartered Institute of Management Accountants). Recuperado el 03 de mayo de 2015 de http://www.cimaglobal.com/documents/importeddocuments/cid_techguide_fraud_risk_management_feb09.pdf.pdf.

-Qué es un BRMS y sus ventajas. (2013, 19 de diciembre). decide. Recuperado el 03 de mayo de 2015 de http://www.decidesoluciones.es/que-es-un-brms-y-sus-ventajas/.

-Figura 2. Fraud Management Solutions. A World of Fraud Prevention at Your Fingertips (2013). Cybersource. Recuperado el 03 de mayo de 2015 de https://www.cybersource.com/resources/collateral/Resource_Center/service_briefs/CYBS_fraud_management_solutions.pdf.

Figura 3. http://tutorials.jenkov.com/web-services/message-formats.html

-Figura 2. Arquitectura cliente servidor. http://integraciondemultimedia2012.blogspot.mx/2012/05/paradigma-cliente-servidor.html

-El protocolo HTTP, Networking and Emerging Optimization (NEO). Recuperado el 04 de mayo de 2015 de http://neo.lcc.uma.es/evirtual/cdd/tutorial/aplicacion/http.html.

-Web Services con C#. Principios de Web Services. usr.code. Recuperado el 04 de mayo de 2015 de

http://usershop.redusers.com/media/blfa_files/lpcu104/capitulogratis.pdf.

-Figura 4. Web Services con C#. Principios de Web Services. usr.code. Recuperado el 04 de mayo de 2015 de

http://usershop.redusers.com/media/blfa_files/lpcu104/capitulogratis.pdf.

-Fernandez. A. (2013, 12 noviembre). Servicios web RESTful con HTTP. Parte I: Introducción y bases teóricas. Asociación Desarrolladores Web de España (ADWE). Recuperado el 04 de mayo de 2015 de http://www.adwe.es/general/colaboraciones/servicios-web-restful-con-http-parte-i-introduccion-y-bases-teoricas.

-Figura 5. Fernandez. A. (2013, 12 noviembre). Servicios web RESTful con HTTP. Parte I: Introducción y bases teóricas. Asociación Desarrolladores Web de España

(ADWE). Recuperado el 04 de mayo de 2015 de http://www.adwe.es/general/colaboraciones/servicios-web-restful-con-http-parte-i-introduccion-y-bases-teoricas.

-Figura 6. Fernandez. A. (2013, 12 noviembre). Servicios web RESTful con HTTP. Parte I: Introducción y bases teóricas. Asociación Desarrolladores Web de España (ADWE). Recuperado el 04 de mayo de 2015 de http://www.adwe.es/general/colaboraciones/servicios-web-restful-con-http-parte-i-introduccion-y-bases-teoricas.

-Laravel PHP framework. (2014, 22 de julio). dondaldorios.com. Recuperado el 04 de mayo de 2015 de http://donaldorios.com/2014/07/22/laravel-php-framework/.

Qué es PHP?. php.net. Recuperado el 04 de mayo de 2015 de https://php.net/manual/es/intro-whatis.php.

-Acosta. D. (2010, noviembre). Fundamentos de 'tokenización' y el cumplimiento de PCI DSS. internet security auditors. Recuperado el 05 de mayo de 2015 de http://isecauditors.com/sites/default/files//files/RedSeguridad_49_Tokenizacion.pdf.

-El modelo entidad-relación. Apuntes del módulo profesional: sistemas gestores de bases de datos. Departamento de Informática del I.E.S. Bajo Guadalquivir. Recuperado el 06 de mayo de 2015 de http://www.juntadeandalucia.es/averroes/iesbajoguadalquivir/inf/sgbd/ApuntesSGBD7.pdf.

-Catherine M. (2009). Bases de datos. México: McGRAW-HILL.

-Sotorrío O. (2010, 12 de octubre). Transformación del Modelo E/R al Modelo Relacional. oscarsotorrio.com. Recuperado el 07 de mayo de 2015 de http://oscarsotorrio.com/post/2010/10/12/Transformacion-del-Modelo-ER-al-Modelo-Relacional.aspx

-Credit Card Services Using the Simple Order API. (2015, abril). Cybersource. Recuperado el 09 de mayo de 2015 de

http://apps.cybersource.com/library/documentation/dev_guides/CC_Svcs_SO_API/Credit_Cards_SO_API.pdf.

-Metodología Scrum. Softeng.es. Recuperado el 07 de mayo de 2015 de http://www.softeng.es/es-es/empresa/metodologias-de-trabajo/metodologia-scrum.html.