Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Laboratorio de Comunicación y Redes

Aplicación de Tipo Bóveda para el Almacenamiento y la Recuperación de Datos Asociados a Tarjetas de Crédito Mediante el Uso de Tokenización Siguiendo las Normas PCI

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
Por la Bachiller:

Daniela Alejandra De Vivo Martínez
V-24.699.188

Para optar por el título de Licenciado en Computación

Tutores: Prof. Eric Gamess y Prof. Antonio Russoniello

Caracas, Julio 2017

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Laboratorio de Comunicación y Redes

ACTA DE VEREDICTO

Quienes suscriben, miembros del jurado designado por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por la Bachiller Daniela De Vivo, C.I. V-24.699.188, con el título "**Aplicación de Tipo Bóveda para el Almacenamiento y la Recuperación de Datos Asociados a Tarjetas de Crédito Mediante el Uso de Tokenización Siguiendo las Normas PCI**", a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el nombrado trabajo por cada uno de los miembros del jurado, éste fijó el día viernes 21 de julio de 2017, para que su autora lo defendiera en forma pública, en la Sala de Internet II, Laboratorio LACORE, de la Escuela de Computación, mediante una exposición oral de su contenido, luego de la cual respondió satisfactoriamente a las preguntas que le fueron formuladas por el jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela.

Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

Es de aclarar que el Profesor Eric Gamess se encuentra de permiso fuera del país. Por esta razón, el Profesor Robinson Rivas, director de la Escuela de Computación, firma el presente documento en su lugar. Adicionalmente, el profesor Darwing Hernández (Jurado Suplente) sustituyó al profesor Dedaniel Urribarri (Jurado Principal) ya que este último se encuentra fuera del país por un periodo de dos semanas.

En fe de lo cual se levanta el presente acta en Caracas a los 21 días del mes de julio del año 2017, dejando constancia que el Profesor Antonio Russoniello actuó como coordinador del jurado.

prof Robinson Rivas M.I.

por el Prof. Eric Gamess
Tutor

Prof. Antonio Russoniello
Tutor

REPUBLICA DE VENEZUELA
Facultad de Ciencias
Escuela de Computación
Universidad Central - Caracas

Prof. Roger Bello
Jurado Principal

Prof. Darwing Hernández
Jurado Suplente

Agradecimientos

Gracias a los profesores Eric Gamess y Antonio Russoniello por apoyarme y guiarme durante el desarrollo de esta investigación.

Gracias a Eneida Colmenares y David Centeno por la guía y el apoyo, a nivel técnico y personal, durante el desarrollo de este sistema.

Gracias a Gabriel Castro, Joaquín Molina y Diego Oliveros por ser, más que unos compañeros, unos amigos y mi principal apoyo y compañía durante la carrera.

Gracias a Gabriel Da Silva y Lolyanni García por ser unos grandes amigos y un apoyo incondicional durante esta etapa final.

Gracias mi tía, Mónica Tome, por ayudarme tanto durante este proceso.

Gracias a mis primos, Alejandro García, Rodrigo García y Mónica Royo, por ser siempre mi guía, mi ejemplo a seguir y unos hermanos para mí.

Gracias a mi novio, Saúl Narváez, por apoyarme y motivarme durante este proceso, por acompañarme tantas largas noches de trabajo y por estar mi lado incluso en los días difíciles.

Gracias mi mamá Mary Luz Martínez y mi madrina Olga León por siempre creer en mí y estar a mi lado de manera incondicional, no solo durante este proceso sino durante toda la carrera, por darme un empujón cuando hizo falta y por apoyarme tanto en los momentos buenos como en los malos. Las quiero mucho.

Dedicado mi papá Enrique De Vivo y a mi tía Marisa Martínez, quienes siempre me cuidan y acompañan desde el cielo, sé que estarían muy felices y orgullosos en este momento.

Resumen

Título:
Aplicación de Tipo Bóveda para el Almacenamiento y la Recuperación de Datos Asociados a Tarjetas de Crédito Mediante el Uso de la Tokenización Siguiendo las Normas PCI.

Autor:
Daniela De Vivo.

Tutores:
Prof. Eric Gamess, Prof. Antonio Russoniello

Los avances tecnológicos han llevado a que en la actualidad exista un gran porcentaje de comercios que realicen sus ventas o sus cobros mediante Internet; sin embargo, en comercios donde los pagos deben realizarse de manera rápida o que se realizan muy repetitivamente, el mecanismo de introducir, de manera manual, los datos de las tarjetas, puede tener efectos negativos, lo que hace que surja la necesidad de buscar una solución que sea más rápida pero igual de segura y eficiente. El presente Trabajo Especial de Grado consiste en la propuesta y desarrollo de un sistema de bóveda que sirva para el almacenamiento y la recuperación, mediante un token, de los datos de una tarjeta de crédito, tomando en cuenta las normas PCI para mantener los niveles de seguridad, y teniendo como fin agilizar los procesos de pago. La propuesta constó de una primera etapa de investigación del problema, dónde se realizó el diseño de la solución. El sistema consta de diferentes módulos que fueron desarrollados en forma de web services, y además permite la interacción con aplicaciones externas en varios tipos de formatos de mensajes, específicamente: XML, JSON y ISO 8583. Al finalizar el desarrollo del sistema, se realizó también una aplicación de tipo cliente que sirvió para realizar las pruebas necesarias. Una vez terminado el sistema y terminadas las pruebas, se pudo comprobar que todos los objetivos planteados al principio de este trabajo fueron logrados de manera satisfactoria.

Palabras clave:
Token, Web Service, Tarjeta de Crédito, JSON, XML, ISO 8583, Payment Card Industry Data Security Standard, Pago, Bóveda.

Tabla de Contenido

1. Introducción 15
1.1 Justificación y Planteamiento del Problema 15
1.2 Objetivo General 16
1.3 Objetivos Específicos 16
1.4 Alcance del Problema 17
1.5 Distribución del Documento 17

2. Marco Teórico 19
2.1 Cifrado y Seguridad de Datos 19
2.1.1 Cifrado 19
2.1.2 Tipos de Cifrado según sus Claves 20
2.1.3 Tipos de Cifrado según Algoritmos 21
2.1.4 DES 21
2.1.5 Triple DES 21
2.1.6 AES (Advanced Encryption Standard) 22
2.1.7 PCI DSS (Payment Card Industry Data Security Standard) 23
2.1.8 P2PE (Point-to-Point Encryption) 24
2.2 Autenticación 25
2.1.1 Sistemas de autenticación Según Factores 25
2.1.2 Mecanismos más Utilizados para la Autenticación 26
2.3 Tokenización 28
2.4 Pagos Electrónicos 30
2.4.1 SET (Secure Electronic Transaction) 31
2.4.2 123Pago 32
2.5 Bóveda de Tarjetas 33
2.5.1 Formato de Mensajes 34
2.6 Herramientas de Software 35
2.6.1 Java 36
2.6.2 Java EE (Enterprise Edition) 36
2.6.3 MySQL 37
2.6.4 Cipher y Key Generator 37
2.6.5 Eclipse IDE 37
2.6.6 Servicios RESTful 38
2.6.7 Apache Tomcat 38
2.6.8 Glassfish 38
2.6.9 HTTP y POST 38
2.6.10 Jersey 39
2.6.11 JPOS 39
2.7 Herramientas de Hardware 39
2.8 Trabajos Relacionados 40
2.8.1 VTS (Visa Tokenization Service) 40
2.8.2 MasterCard 40
2.8.3 Conclusiones 41
3.1 Scrum 43

3.2 Implementación de la Metodología ... 43
3.3 Pruebas y Resultados ... 44
4. Marco Aplicativo ... 45
4.1 Análisis Preliminar ... 45
4.2 Diagrama de Casos de Uso ... 45
4.3 Diseño de la Solución ... 46
4.4 Flujo del Sistema ... 47
4.5 Desarrollo de la Solución ... 48
4.5.1 Ambiente y Herramientas ... 48
4.5.2 Base de Datos ... 48
4.5.3 Servicio de Tokenización ... 49
4.5.4 Servicio de Autenticación y Administración de Información ... 52
4.5.5 Servicio Traductor ... 58
4.6 Aspectos de Seguridad ... 60
5. Pruebas y Resultados ... 63
5.1 Seguridad ... 63
5.1.1 PCI ... 63
5.1.2 Generador de Llaves ... 65
5.1.3 Token en Cabecera ... 67
5.2 Funcionamiento ... 68
5.2.1 Servicio de Tokenización ... 68
5.2.2 Administración de Comercios ... 68
6.2.3 Administración de Token ... 69
5.2.4 Servicio de Traducción ... 72
5.3 Tiempos de Respuesta ... 73
6. Conclusiones ... 77
6.1 Limitaciones ... 78
6.2Trabajos Futuros ... 78
Referencias ... 79
Anexo A: Formato Personalizado ISO 8583 ... 81
Anexo B: Invocación de Pago ... 85

Índice de Figuras

Figura 2.1: Esquema Secuencial del Funcionamiento de AES 23
Figura 2.2: Tarjeta RFID 27
Figura 2.3: Código QR 28
Figura 2.4: Flujo de un Pago con Tarjeta 30
Figura 2.5: Botón de Pago 123Pago 33
Figura 2.6: Icono Taquilla Virtual 123Pago 33
Figura 2.7: Ejemplo de un Mensaje XML 35
Figura 2.8: Ejemplo de mensaje JSON 35
Figura 4.1: Diagrama de Casos de Uso del Sistema 46
Figura 4.2: Arquitectura del Sistema 47
Figura 4.3: Diseño de la Base de Datos 49
Figura 4.4: Especificación de Formato que Consume y Produce el Servicio 50
Figura 4.5: Ruta del Servicio y del Método 50
Figura 4.6: Ruta de la Aplicación 50
Figura 4.7: JSON que Recibirá el Servicio 51
Figura 4.8: Parámetros para Generar la Llave 51
Figura 4.9: JSON que Responderá el Servicio 52
Figura 4.10: Ruta de Acceso al Servicio de Autenticación y Administración 52
Figura 4.11: JSON para Agregar Comercio 53
Figura 4.12: JSON de Respuesta al Agregar un Comercio 53
Figura 4.13: JSON para Eliminar Comercio 53
Figura 4.14: JSON de Respuesta al Eliminar un Comercio 53
Figura 4.15: JSON para Asociar Comercio a una Bóveda 54
Figura 4.16: JSON Respuesta de Asociar Comercio a una Bóveda 54
Figura 4.17: JSON para Generar un Token 55
Figura 4.18: Ejemplo de Conexión con Otro Servicio Web 55
Figura 4.19: JSON que se Recibe para Recuperar un Token 56
Figura 4.20: JSON que se Enviará al Sistema de Pago 58
Figura 4.21: Función que Convierte XML en JSON 59

Figura 4.22: Ejemplo de Invocación al Sistema en XML 59
Figura 4.23: Ejemplo de un Mensaje ISO 8583 60
Figura 4.24: JSON Respuesta de Agregar o Asociar Comercio 61
Figura 4:25: Estructura de Tabla para Autenticación 61
Figura 5.1: Comparativa de Tokens 65
Figura 5.2: Llaves para Distintas Bóvedas 66
Figura 5.3: Resultado Token de Cabecera con un Dígito Diferente 67
Figura 5.4: Error de Comercio Duplicado 69
Figura 5.5: Status de una Bóveda Activa vs. una Desactiva 69
Figura 5.6: Dos Comercios Diferentes Asociados a una Misma Bóveda 69
Figura 5.7: Diferentes Tarjetas en una Misma Bóveda 70
Figura 5.8: Invocación al Servicio del Sistema de Pago Mediante el Probador 71
Figura 5.9: Invocación Mediante el Servicio de Autorización y Administración de Información. 72
Figura 5.10: Gráfico de Tiempo de Respuesta Promedio de un Pago 74
Figura 5.11: Promedio de Tiempo de Respuesta por Petición 75

Índice de Tablas

Tabla 2.1: Resumen de Algoritmos de Cifrado 22
Tabla 2.2: Ejemplos de Mecanismos de Tokenización..... 29
Tabla 5.1: Cuadro de Verificación de Normas PCI 64

1. Introducción

Actualmente, las empresas se encuentran en una constante evolución y actualización tecnológica que les permita mejorar, incrementar y facilitar sus ventas, logrando así un mayor número de clientes. Esto ha llevado a que muchos comercios, quizás la mayoría, desarrollen sitios web y aplicaciones móviles que permitan a los clientes interactuar con sus servicios y realizar pagos y compras a través de los mismos.

Para este tipo de transacciones, el medio de pago más común y sencillo a utilizar son las tarjetas de crédito, sin embargo, el mecanismo de ingreso de los datos de estas es un poco tedioso y puede volverse repetitivo; lo que hace concluir que la solución inmediata a ese problema es dejar almacenados los datos de las tarjetas y que el cliente pueda utilizarlas, simulando una especie de billetera electrónica específica para el comercio. A pesar de que esto parezca una solución simple, para lograrlo se deberían almacenar todos los datos de la tarjeta que son necesarios para la transacción, pero, esto llevaría a almacenar, de manera inapropiada, información que es sensible lo que podría volverse sumamente riesgoso.

Este problema ha hecho que la empresa 123Pago, dedicada a ofrecerle facilidades de pago a sitios web y aplicaciones móviles, se interese en crear una solución, de tipo bóveda, que permita almacenar de manera segura los datos de las tarjetas de crédito, proveyendo a las aplicaciones información no sensible que pueda ser asociada con los datos reales, pero manteniendo estos de manera protegida hasta el momento de ser enviados al canal de pago. Para poder cumplir todos estos requerimientos se deben tomar en cuenta muchos aspectos, tales como: mecanismo de acceso a las tarjetas, estructuras donde serán almacenadas, seguridad en todo el procedimiento, mecanismos de autenticación entre las aplicaciones y la bóveda, protocolos de comunicación, entre otras cosas.

El presente Trabajo Especial de Grado tiene como fin explicar de manera detallada todo lo referente a dicha solución, es decir, desde la investigación realizada para adquirir los conocimientos necesarios, pasando por el diseño de la misma, hasta el desarrollo final y sus conclusiones.

1.1 Justificación y Planteamiento del Problema

La empresa 123Pago es una empresa venezolana que tiene como fin ofrecer a distintos comercios, que tienen sitios web o aplicaciones móviles, soluciones para procesar sus pagos en línea con tarjetas de crédito, facilitando así que sus clientes utilicen sus servicios o adquieran sus productos. Hasta el momento, el procedimiento que manejaban para llevar esto a cabo, consistía en colocar un botón de pago que sirviera de acceso a un formulario donde se capturaban, de manera segura, los datos de las tarjetas y eran enviadas al canal de pago, para posteriormente devolver a la aplicación el resultado del pago, pero, como ya se mencionó, el ingreso repetitivo de los datos de las tarjetas en este formulario se vuelve poco práctico, lo que

limita a que este botón de pago solo sirva para comercios donde las transacciones pueden hacerse de manera no inmediata, es decir, contando con que los clientes poseen tiempo suficiente, al momento de pagar, y que además poseen siempre sus tarjetas de crédito físicas como para poder dar toda la información.

Por otro lado, el manejo de los datos que se necesitan en una transacción de pago es muy delicado, ya que se deben cuidar muchos aspectos de seguridad, lo que hace que no se pueda tomar como opción el almacenar la información de las tarjetas en una base de datos perteneciente a cada aplicación. En este caso, para asegurar la integridad de los datos deben seguirse las normas de PCI DSS (Payment Card Industry Data Security Standard).

Adicionalmente, si se desean almacenar los datos de manera segura, también se debe garantizar que el acceso a ellos sea igual, por lo que hay que proveer mecanismos de autenticación entre las aplicaciones y la bóveda, y un mecanismo que le provea a la aplicación un dato que pueda manejar, sin ser información sensible, pero que sirva para asociarlo con los datos completos de la tarjeta. Para el mecanismo de acceso lo más eficiente y seguro es utilizar la tokenización del número de tarjeta, ya que genera un identificador único y seguro.

La necesidad de poder ofrecer a los comercios una solución que sea mucho más práctica y rápida, pero que tome en cuenta todos los detalles necesarios, llevó a que la empresa se interesara en desarrollar una bóveda de tarjetas de crédito que, cumpliendo todos los requisitos de seguridad necesarios, pueda almacenar los datos de las mismas, pudiendo así poder ofrecerle servicios de pago a aplicaciones que necesitan que estos se hagan de manera rápida.

1.2 Objetivo General

Desarrollar una aplicación que funcione como bóveda para el almacenamiento de los datos de tarjetas de crédito, utilizando la tokenización y garantizando el cumplimiento de las normas PCI.

1.3 Objetivos Específicos

- Desarrollar el modelo de la base de datos.
- Desarrollar un mecanismo de cifrado.
- Desarrollar funciones de tokenización y des-tokenización.
- Investigar e implementar mecanismos de autenticación entre aplicaciones.
- Desarrollar un servicio web que administre y traduzca las peticiones de las aplicaciones (que pueden ser recibidas en distintos lenguajes).
- Desarrollar un servicio que una vez obtenidos los datos en claro sea capaz de interactuar con el canal de pago y enviar respuesta a la aplicación.

1.4 Alcance del Problema

Se realizará una aplicación de tipo bóveda que sea capaz de tokenizar tarjetas al momento de agregarlas y de recuperarlas y medisnte des-tokenización, al momento de realizar el pago. Adicionalmente, se debe poder comunicar con distintas aplicaciones de los comercios y con el canal de pago, que, para este trabajo, será 123Pago. En esta aplicación se deben cumplir las normas PCI en todos los pasos. La aplicación estará compuesta por varios módulos en forma de servicios web. La aplicación y su base de datos serán instaladas en los servidores de 123Pago.

1.5 Distribución del Documento

El presente documento describe los procesos de investigación y desarrollo utilizados para cumplir con los objetivos antes descritos. El documento está dividido en 5 capítulos: el presente (Capítulo 1), contiene la justificación y planteamiento del problema, junto con los objetivos (general y específicos) de la investigación. El Capítulo 2, detalla los basamentos teóricos necesarios para comprender el resto del documento, así como la metodología usada durante el desarrollo del sistema. En el Capítulo 3, se presenta de forma detallada el proceso de diseño y desarrollo del sistema. El Capítulo 4, describe los resultados obtenidos en las pruebas realizadas, y las especifica. Finalmente, en el Capítulo 5, se muestran las conclusiones del trabajo realizado, describiendo los aportes realizados, junto al planteamiento de posibles trabajos futuros.

2. Marco Teórico

Este capítulo tiene como fin describir, de manera explicativa y detallada, las bases teóricas sobre las cuales se realizó la investigación. Se describirá qué es el cifrado y la seguridad de datos, haciendo énfasis en el tipo de cifrado AES y las normas PCI, para luego llegar a la tokenización de tarjetas. Luego, se hará una breve explicación sobre los datos de una tarjeta de crédito que son necesarios para realizar un pago y cómo es el flujo del mismo. Posteriormente, se describirá en qué consiste una bóveda de tarjetas de crédito, cuáles son los aspectos que hay que tomar en cuenta, qué debe cumplir su estructura y que tipo de peticiones puede recibir. Por último, se hablará un poco sobre las tecnologías utilizadas en el desarrollo.

2.1 Cifrado y Seguridad de Datos

En cualquier tipo de comunicación establecida entre dos o más entes existe el riesgo de que esta misma sea interferida o percibida por terceros que no se encuentran involucrados. Este riesgo puede ser pasado por alto si lo que se desea transmitir puede ser conocido públicamente, pero de no ser así, deben tomarse medidas para proteger los mensajes intercambiados.

A raíz de esto se percibe la necesidad de tomar en cuenta diversos mecanismos de seguridad que garanticen que las comunicaciones serán íntegras, confiables y exclusivas. La prioridad en este escenario es proteger los mensajes para que no sean modificados ni puedan ser accedidos por entes no deseados.

2.1.1 Cifrado

El cifrado consiste en la transformación de un mensaje bit a bit, sin importar su estructura lingüística, para volverlo incomprensible. Este procedimiento se realiza mediante algoritmos con clave, que pueden ser de diversos tipos, y que se utilizan tanto para cifrar (en el emisor del mensaje), como para descifrar (en el receptor del mensaje), de esta manera, si algún intruso tuviera acceso al mensaje, ya cifrado, no podría saber cuál es el contenido verdadero pues no posee la llave.

El cifrado de mensajes se enfoca principalmente en satisfacer los siguientes objetivos:

- **Confidencialidad:** busca que la información no pueda ser entendida por alguien que no es un punto de la comunicación, es decir, un mensaje sólo podrá ser comprensible para el emisor y el receptor (o los receptores, ya que el mensaje podría ser recibido por varios entes).
- **Integridad:** garantiza que la información no pueda ser modificada ni en el almacenamiento de la misma ni en el tránsito de un punto a otro, y en el caso de que se altere algún mensaje, la modificación del mismo será detectada.

- **No repudio:** garantiza que ningún emisor pueda negar haber sido quien enviará un mensaje firmado digitalmente a un receptor.
- **Autenticidad:** busca lograr que en la comunicación el emisor y el receptor pueden verificarse entre sí, de manera que se sepa que tanto el origen como el destino de la información son legítimos y son los deseados.

2.1.2 Tipos de Cifrado según sus Claves

Como se ha mencionado, el mensaje se cifra en el emisor y se descifra en el receptor. Este flujo es igual para cualquier esquema de cifrado, lo que puede variar son las claves, por lo que existen los siguientes tipos:

- **Cifrado simétrico**: también es conocido como cifrado de clave secreta o cifrado se una sola clave. Es un método criptográfico que utiliza la misma clave para cifrar en el emisor como para descifrar en el receptor [1]. En este caso, las partes involucradas en la comunicación deben ponerse de acuerdo en la clave que se utilizará. La seguridad del cifrado radica en el nivel de seguridad que le proveen las claves, por lo que este esquema posee un inconveniente, y es que los participantes de la comunicación deben intercambiar la clave y los mensajes necesarios para llegar al acuerdo, lo que hace que la seguridad de la clave dependa de la seguridad del medio en el que se envió la misma.
- **Cifrado asimétrico:** también conocido como cifrado de clave pública o cifrado de dos claves. Es un método criptográfico en el que existen dos claves, ambas pertenecientes al receptor del mensaje, donde una es pública y puede ser entregada a cualquier destinatario, y la otra es privada y debe ser guardada por el propietario para que nadie pueda accederla [1]. Cada pareja de claves puede ser generada una sola vez, por lo cual, no existen duplicados en las combinaciones. Se utiliza la clave pública para cifrar el mensaje y la clave privada para descifrarlo, de manera que solo el receptor podrá tener acceso al mensaje. Este método soluciona el problema del cifrado simétrico, ya que los participantes no deben intercambiar mensajes para acordar la clave a utilizar.
- **Cifrado híbrido:** es un método criptográfico que implementa las ventajas tanto del cifrado simétrico como del asimétrico [2]. En este método se deben generar una clave pública y una clave privada en el receptor. Se establece un canal seguro mediante el sistema de cifrado asimétrico para poder intercambiar una clave simétrica a través de dicho canal.
- **Cifrado por curva elíptica:** es una variante del cifrado asimétrico basada en las matemáticas de las curvas elípticas [3]. Nace como una alternativa de los sistemas de cifrado de clave pública, ya que disminuye el tamaño de las claves y aumenta el abanico de posibilidades. Mediante la solución de problemas matemáticos de alta complejidad, como el logaritmo discreto, se les asignan valores a las claves.

2.1.3 Tipos de Cifrado según Algoritmos

Adicional a las claves, los sistemas de cifrado utilizan diferentes tipos de algoritmos tanto para cifrar como para descifrar los mensajes, y según la forma en que estos operan se pueden definir distintos tipos de cifrado:

- **Cifrado por bloques:** es un método criptográfico que utiliza algoritmos que operan en grupos de bits de longitud fija, a los que se les denomina bloques, y aplicándoles una transformación invariante. La manera de operar de estos algoritmos es recibir una entrada de texto claro y producir un bloque del mismo tamaño, dónde la transformación se hará mediante una segunda entrada que recibe el algoritmo, que es la clave secreta. El descifrado funciona de la manera inversa, es decir, se ingresan bloques de texto cifrado y la clave secreta, y se producen bloques de texto claro.
- **Cifrado por flujo:** es un método criptográfico que utiliza algoritmos que operan cifrando bit a bit. Se basan en utilizar claves muy largas que sirvan tanto para cifrar como para descifrar. Estas claves pueden ser predeterminadas o pueden generarse por el usuario mediante un generador de claves. Estos algoritmos de cifrado pueden realizar el cifrado incrementalmente, convirtiendo el texto en claro en texto cifrado bit a bit. Esto se logra construyendo un generador de flujo de clave. Un flujo de clave es una secuencia de bits de tamaño arbitrario que puede emplearse para oscurecer los contenidos de un flujo de datos combinando el flujo de clave con el flujo de datos.

2.1.4 DES

Data Encryption Standard (DES) es un algoritmo de cifrado, es decir, un método para cifrar información. La estructura básica del algoritmo es la siguiente: hay 16 fases idénticas de proceso, denominadas rondas. También hay una permutación inicial y una final, que son funciones inversas entre sí. Estas permutaciones no son criptográficamente significativas, pero se incluyeron para facilitar la carga y descarga de bloques. Antes de las rondas, el bloque es dividido en dos mitades de 32 bits y procesadas alternativamente. Este algoritmo posee una clave de 56 bits.

Hoy en día, DES se considera inseguro para muchas aplicaciones. Existen resultados analíticos que demuestran debilidades teóricas en su cifrado, aunque son inviables en la práctica. Se cree que el algoritmo es seguro en la práctica en su variante de Triple DES, aunque existan ataques teóricos.

2.1.5 Triple DES

Triple DES es el algoritmo que hace triple cifrado del DES y fue desarrollado por IBM en 1998 a raíz de que se descubrió que una clave de 56 bits no era suficiente para evitar un ataque de fuerza bruta, y nace con el fin de agrandar el largo de la clave sin necesidad de cambiar de algoritmo de cifrado. Este método de cifrado dobla la longitud a de la clave a 112 bits, pero en

cambio se triplica el número de operaciones de cifrado, haciendo este método muchísimo más seguro que el DES. La longitud de la clave usada será de 168 bits (3 x 56 bits, que es equivalente a las tres veces de DES), aunque como se ha dicho su eficacia solo sea de 112 bits. Este método es utilizado por la mayoría de las tarjetas de crédito y otros medios de pago electrónicos.

2.1.6 AES

AES es un estándar para el cifrado por bloques y actualmente es uno de los algoritmos más populares y utilizados. Es creado en el 2001 por Rijmen-Daemen en Bélgica y nace con el fin de lograr un algoritmo de cifrado que fuera seguro y eficiente, ya que DES generaba problemas de seguridad y Triple DES problemas de velocidad. En la Tabla 2.1 podemos ver un resumen de la comparación entre estos algoritmos.

Este algoritmo posee claves de 128, 192 y 256 bits y maneja bloques de 128 bits de datos y posee un diseño sencillo pero resistente a diversos ataques. Como podemos ver en la Figura 2.1 su funcionamiento consta de: procesar la data en bloques de 4 columnas de 4 bytes, esto es denominado la matriz de estado, posee 10, 12 o 14 rounds y utiliza el operador "exclusive or" (XOR) para mezclar las columnas entre la clave y la matriz de estado.

Algoritmo	***Tamaño de clave (bits)***	***Tamaño de bloque (bits)***	***Número de etapas***	***Aplicaciones***
DES	56	64	16	SET, Kerberos
Triple DES	112 o 168	64	48	Tarjetas de crédito y pagos electrónicos
AES	128, 192 y 256	128	10, 12 o 14	Diseñado para sustituir DES y Triples DES.

Tabla 2.1: Resumen de Algoritmos de Cifrado

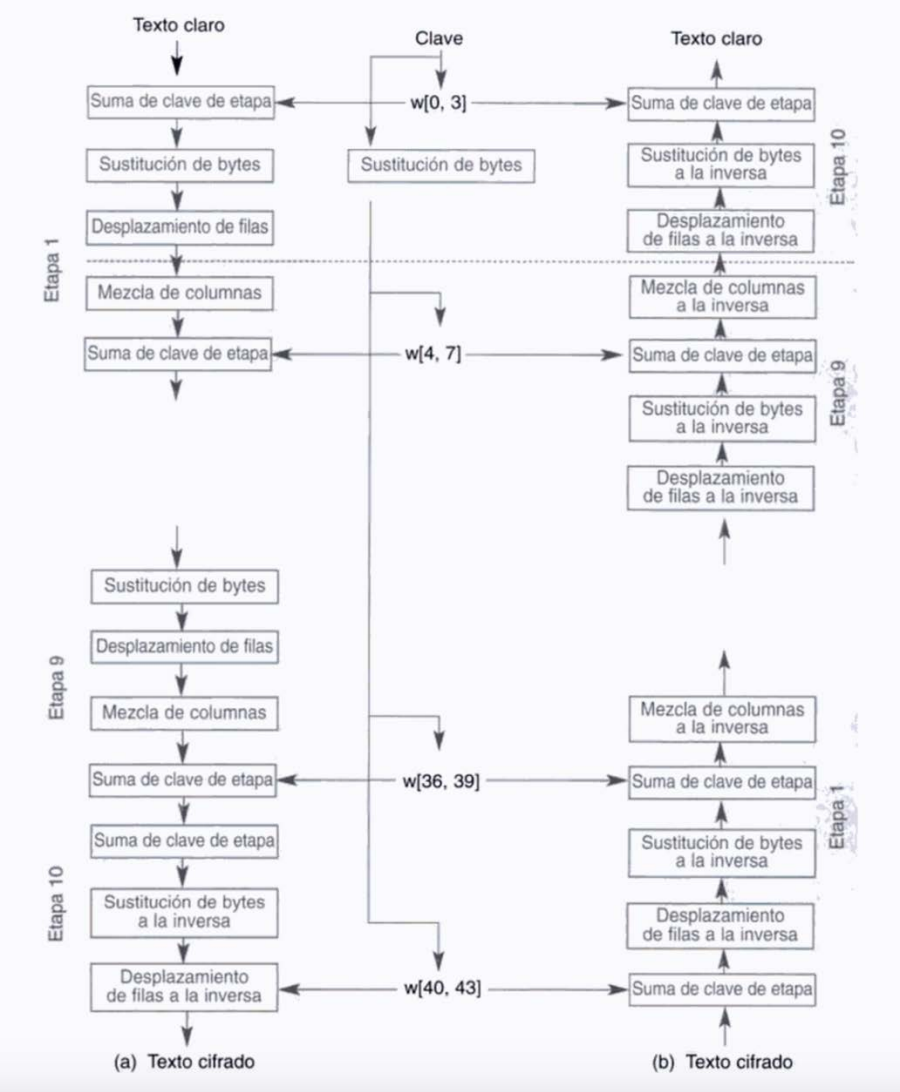

Figura 2.1: Esquema Secuencial del Funcionamiento de AES

2.1.7 PCI DSS

El PCI DSS es un estándar para la seguridad de datos en la industria de las tarjetas de pago. Surge como necesidad de estandarizar las normas de seguridad en el procesamiento, almacenamiento y transmisión de las transacciones de pagos que implican tarjetas, y fue desarrollado por un comité integrado por las compañías de tarjetas, tanto de crédito como de débito, más importantes, el PCI SCC (Payment Card Industry Standards Cuncil). Las compañías que procesan, guardan o trasmiten datos de tarjetas deben cumplir con el estándar o arriesgan la pérdida de sus permisos para procesar las tarjetas de crédito y débito (pérdida de franquicias), enfrentar auditorías rigurosas o pagos de multas. Los comerciantes y proveedores de servicios de tarjetas de crédito y débito, deben validar su cumplimiento al estándar en forma periódica.

La versión actual de la normatividad especifica 12 requisitos para el cumplimiento, organizados en 6 secciones relacionadas lógicamente, que son llamadas "objetivos de control" [4]. Los objetivos de control y sus requisitos son los siguientes:

- **Desarrollar y mantener una red segura**
  - Requisito 1: Instalar y mantener una configuración de cortafuegos para proteger los datos de los propietarios de tarjetas.
  - Requisito 2: No usar contraseñas del sistema y otros parámetros de seguridad predeterminados provistos por los proveedores.
- **Proteger los datos de los propietarios de tarjetas**
  - Requisito 3: Proteger los datos almacenados de los propietarios de tarjetas.
  - Requisito 4: Cifrar los datos de los propietarios de tarjetas e información confidencial transmitida a través de redes públicas abiertas.
- **Mantener un programa de gestión de vulnerabilidades**
  - Requisito 5: Usar y actualizar regularmente un software antivirus.
  - Requisito 6: Desarrollar y mantener sistemas y aplicaciones seguras.
- **Implementar medidas sólidas de control de acceso**
  - Requisito 7: Restringir el acceso a los datos tomando como base la necesidad del funcionario de conocer la información.
  - Requisito 8: Asignar una identificación única a cada persona que tenga acceso a un punto del sistema.
  - Requisito 9: Restringir el acceso físico a los datos de los propietarios de tarjetas.
- **Monitorizar y probar regularmente las redes**
  - Requisito 10: Rastrear y monitorizar todo el acceso a los recursos de la red y datos de los propietarios de tarjetas.
  - Requisito 11: Probar regularmente los sistemas y procesos de seguridad.
- **Mantener una política de seguridad de la información**
  - Requisito 12: Mantener una política que contemple la seguridad de la información

2.1.8 P2PE

El cifrado punto a punto (Point-to-Point Encryption) es un estándar establecido por el PCI. Tiene como objetivo proporcionar una solución de seguridad de pago que convierte instantáneamente las tarjetas (crédito y débito) de datos e información, en código indescifrable, a partir del momento se la tarjeta, para así prevenir la falsificación y el fraude [5]. Básicamente está diseñado para maximizar la seguridad de las transacciones con tarjetas de pago en un entorno cada vez más complejo.

El estándar P2PE define los requisitos que una "solución" debe cumplir para ser aceptada como una solución PCI válida. Una "solución" es un conjunto completo de hardware, software, puerta de entrada, des/cifrado, manejo de dispositivos, etc. Sólo "soluciones" pueden ser validadas; piezas individuales de hardware, tales como lectores de tarjetas no pueden ser validados.

La determinación de una solución cumpla con la norma P2PE o que no la cumpla es la responsabilidad de un asesor de seguridad P2PE calificado. Algunos de los requerimientos incluyen:

- Administración segura del cifrado y descifrado en los dispositivos.
- Seguridad de alto nivel en el ambiente donde se utilizará la información descifrada.
- El uso de metodologías de cifrado seguras y también de las operaciones de claves criptográficas, incluyendo generación de claves, distribución, carga/inyección, administración y el uso.

2.2 Autenticación

La autenticación es el proceso de verificación de la identidad de los participantes en una comunicación, adicionalmente, es una petición a conectarse [6]. Es simplemente un modo de asegurar que los usuarios son quienes dicen ser.

Este proceso va de la mano con el cifrado ya que deben trabajar de manera conjunta para lograr un resultado óptimo, debido a que el cifrado ayuda a la autenticación asegurándole que los mensajes que se envían entre emisor y receptor no fueron interferidos ni entendidos por terceros, mientras que, la autenticación ayuda al cifrado asegurando que el origen y el destino, de los mensajes protegidos, son quienes dicen ser.

Un método de autenticación debe cumplir diversas funciones, pero la prioridad del mismo es que cumpla las siguientes características:

- **Debe ser fiable**, con una probabilidad de acierto muy alta, es decir, el método de autenticación no debería fallar casi nunca en un esquema normal de comunicación.
- **Debe poder identificar y soportar ataques** que intenten vulnerarla, con diferentes técnicas y mecanismos, de manera que el método sea estable.
- **Ser usable y comprensible** por los usuarios finales, pues si la complejidad del sistema es muy elevada, no será usable y los errores se cometerán en los participantes de la comunicación

La autenticación consiste en, un usuario que solicita acceso a un sistema, el sistema le solicita credenciales que validen su identidad, el usuario ingresa tales credenciales y el sistema los verifica para generarle o negarle el acceso

2.1.1 Sistemas de Autenticación Según Factores

Los métodos de autenticación pueden ser basados en diferentes factores, es decir, la misma puede realizarse mediante el suministro de diferentes tipos de información. Principalmente la autenticación se divide en los siguientes tipos:

- **Sistemas basados en algo conocido**: este tipo de sistemas es uno de los más utilizados, ya que realizan la autenticación mediante el uso de un factor conocido por el usuario a autenticarse, por ejemplo, el uso de una contraseña. En todos los esquemas de autenticación basados en contraseñas, se cumple el mismo protocolo: las entidades que participan en la autenticación acuerdan una clave, que debe ser secreta si desean que la autenticación sea fiable. Cuando una de las partes desea autenticarse ante la otra se limita a mostrarle su conocimiento de esa clave común, y si ésta es correcta se otorga el acceso al sistema. Lo habitual es que existan unos roles preestablecidos, con una entidad activa que desea autenticarse y otra pasiva que admite o rechaza a la anterior. Este esquema es muy frágil: basta con que una de las partes no mantenga la contraseña en secreto para que toda la seguridad del modelo se pierda.
- **Sistemas basados en algo poseído**: en este tipo de sistemas la autenticación se hace mediante un objeto que ya posee el usuario, por ejemplo, una tarjeta inteligente o una tarjeta de coordenadas. La tarjeta inteligente, que es la más popular, es un dispositivo de seguridad del tamaño de una tarjeta de crédito, resistente a la adulteración, que ofrece funciones para un almacenamiento seguro de información y también para el procesamiento de la misma en base a tecnología de integración a gran escala. En la práctica, las tarjetas inteligentes poseen un chip empotrado en la propia tarjeta que puede implementar un sistema de archivos cifrado y funciones criptográficas, y además puede detectar activamente intentos no válidos de acceso a la información almacenada.
- **Sistemas basados en biométrica**: este tipo de sistema se basa en una característica física del usuario o en un acto que el mismo no realiza conscientemente. El reconocimiento de formas, la inteligencia artificial y el aprendizaje son las ramas de la informática que desempeñan el papel más importante en los sistemas de identificación biométricos; aquí la criptografía se limita a un uso secundario, como el cifrado de una base de datos de patrones retinales, o la transmisión de una huella dactilar entre un dispositivo analizador y una base de datos. La autenticación basada en características físicas existe desde que existe el hombre y, sin darnos cuenta, es la que más utiliza cualquiera de nosotros en su vida cotidiana: a diario identificamos a personas por los rasgos de su cara o por su voz. Obviamente aquí el agente reconocedor lo tiene fácil porque es una persona, pero en el modelo aplicable a redes el agente ha de ser un dispositivo que, basándose en características del sujeto a identificar, le permita o deniegue el acceso a un determinado recurso.

En un método de autenticación se puede utilizar también la mezcla de varios sistemas de autenticación, por ejemplo, el de algo conocido con el de algo poseído, como sería utilizar una tarjeta de débito e ingresar el PIN de autenticación.

2.1.2 Mecanismos más utilizados para la Autenticación

Como ya se mencionó anteriormente, la autenticación consta de un usuario que solicita acceso a un sistema o una información, mediante unas credenciales que validen si tiene o no el permiso de hacerlo. Por lo general, la autenticación es un esquema de usuario – sistema, pero pueden existir otros casos donde, por ejemplo, la autenticación deba hacerse entre dos

aplicaciones o sistemas, para validar que ambas son quienes dicen ser. Este escenario es un poco diferente porque entre ambas deben acordar que mecanismo se utilizará para realizar dicha autenticación, y el procedimiento seleccionado deberá realizarse cuando se desea comenzar un intercambio de información entre ambas.

Planteados todos los escenarios donde puede realizarse una autenticación, los mecanismos más utilizados para realizarla son [7]:

- **Identificador y contraseña:** es el sistema más utilizado, ya que suele ser bastante simple pero seguro a la vez, tomando en cuenta que su nivel de seguridad siempre dependerá de la complejidad de la contraseña y la no propagación de la misma por el usuario. Este es esquema más sencillo, es un sistema basado en algo conocido, donde se asocia un identificador de usuario a una contraseña y el acceso se otorga si ambas son válidas y coinciden.
- **Identificador y contraseña OTP (One-Time Password):** este esquema funciona igual que el anterior, pero en este caso la contraseña solo puede ser utilizada por un tiempo limitado y solo se puede acceder con ella una vez. Por lo general, en este esquema, las contraseñas son generadas por el sistema y entregadas al usuario, sin embargo, también pueden ser elegidas por el mismo.
- **Certificados PKI (Public Key Infrastructure):** son certificados digitales conformados por combinaciones de hardware, software y protocolos de seguridad, que permiten a los usuarios autenticarse frente a otros usuarios y poder acceder a su información para realizar otras acciones, como, por ejemplo, acceder a una clave pública de un usuario para cifrar/descifrar mensajes que se le desee enviar [8].
- **Biométrica:** basándonos en la información ya explicada anteriormente, se sabe que la autenticación por biométrica consiste en la verificación mediante un elemento físico del usuario o un gesto que realiza de manera involuntaria, este mecanismo solo funciona en esquemas usuario – sistema o en esquemas donde las aplicaciones a autenticarse entre sí deben ser manejadas por el usuario, por lo que en realidad se volvería una autenticación entre dos usuarios y no entre dos aplicaciones.
- **RFID (Radio Frequency Identification):** este mecanismo utiliza tarjetas, que por lo general funcionan como etiquetas como podemos observar en la Figura 2.2, que poseen antenas que permiten enviar y recibir información, y que adicionalmente lleva un número de identificación que permite realizar la autenticación.

Figura 2.2: Tarjeta RFID

- **Código QR (Quick Response):** si bien los códigos QR no son un método de autenticación robusto como para un login de un sistema de información sensible, se utilizan para darle al usuario, mediante el escaneo del mismo, acceso a algún recurso o información particular. Se suele utilizar más como un extra que pueda proveer mayor seguridad que como un mecanismo de autenticación en general. Por ejemplo, se puede utilizar un código QR y un lector para abrir la puerta de un estacionamiento, o para mostrar el precio de un producto. El aspecto de un código QR es el que podemos observar en la Figura 2.3.

Figura 2.3: Código QR

- **Firma digital:** es una herramienta compuesta por un conjunto de datos que son asociados a un mensaje para asegurar la autenticidad e integridad del mismo. Las firmas digitales utilizan criptografía asimétrica [9].
- **Token:** es un elemento que un sistema le entrega a un usuario autorizado para que el mismo se autentique. El token es incomprensible para cualquier otro sistema excepto para el que lo generó, este sistema si puede comprenderlo y traducirlo para permitir o denegar el acceso de un usuario. Actualmente es muy utilizado en diversos sistemas ya que es una manera más segura de darle una "llave" al usuario sin que exista un riesgo muy elevado.

2.3 Tokenización

La tokenización es un proceso mediante el cual un número de cuenta o tarjeta es reemplazado por un valor llamado **token** [10]. Cada token es asociado a un número de tarjeta, por lo que existe también un proceso de des-tokenización que consistirá en, mediante el suministro del token, recuperar los valores originales de la tarjeta o cuenta. La seguridad de este sistema se basa en la complejidad de obtener el número real de la tarjeta o cuenta si solo se posee conocimiento del token.

El origen y la motivación de la tokenización se basa en que almacenar la información de las tarjetas/cuentas y de sus respectivos titulares es muy delicado, lo que lleva a que se deban tomar en cuenta muchos aspectos de seguridad y esto puede volver muy complejo y engorroso el sistema. Al tokenizar esta información, el almacenamiento del mismo es un proceso seguro, ya que el token por sí solo, sin el algoritmo de des-tokenización, no revela ninguna información sensible, por lo que hace mucho más sencillo el almacenamiento de la información y el cumplimiento de las normas PCI.

A continuación se indican las relaciones dadas entre la tokenización y las normas PCI es la siguiente [10]:

- El hecho de utilizar una solución que realice tokenización no implica que se deba obviar el cumplimiento de las normas PCI, el mismo debe mantenerse, sin embargo, será mucho más sencillo para las aplicaciones lograr cumplir los requerimientos necesarios de la normativa.
- Para verificar que una solución de tokenización es efectiva se debe asegurar que el número de cuenta/tarjeta no es recuperable en un ambiente donde no se puede garantizar la seguridad.
- Los sistemas de tokenización y almacenamiento deben estar protegidos mediante diversas medidas de seguridad.
- Si se va a utilizar una solución de tokenización se debe garantizar la veracidad de la misma y de su proveedor.

Si se considera la tokenización de tarjetas, cuya cantidad de dígitos puede ser de 16 o 20, existen 3 maneras principales de tokenizar este número: generando un token númerico completo, generando un token alfanumérico completo o tokenizando solo los 8 o 12 dígitos intermedios y dejando intactos los 4 externos. En la Tabla 2.2 se puede observar un ejemplo de un número de tarjeta y el token que genera cada mecanismo.

Mecanismo de Tokenización	***Número de Tarjeta***	***Token***
Token Numérico	3478 9100 4879 0810	982736281928171927
Token Alfanumérico	5470 0048 9876 2983	7aF1Zx118523mw4cwl5x2
128, 192 y 256	7863 2190 1284 9021	7863x18523mw4cw9021

Tabla 2.2: Ejemplos de Mecanismos de Tokenización

La tokenización es básicamente un proceso de autenticación mediante algo conocido, que en este caso es el token, ya que el usuario o comercio lo utiliza para que se le dé acceso a una información que no es de conocimiento público.

El proceso básico de tokenización consiste en: el sistema de autenticación es invocado mediante una aplicación, esta misma le otorga un número de tarjeta y la información de autenticación que necesite el sistema; el sistema se encarga de validar estos datos y de registrar o recuperar, si son válidos, de la bóveda de almacenamiento, que cumple normas PCI, la información correspondiente al token; el sistema envía la información a dónde haya indicado el solicitante.

Por último, el sistema de tokenización también debe cumplir un conjunto de requisitos establecidos por PCI [10], estos son los siguientes:

1. El sistema no proporciona la información referente al token a ninguna red o aplicación que no esté dentro de las definidas por el comercio.
2. Todos los componentes de tokenización se encuentran en redes internas y seguras.
3. Solo se permiten comunicaciones de confianza y verificadas, tanto interna como externamente del sistema.
4. La solución de autenticación garantiza, mediante mecanismos de seguridad, que los datos de las tarjetas serán almacenados de manera segura.
5. La solución de tokenización posee control de acceso.
6. Los componentes del sistema están diseñados bajo estándares de seguridad.
7. La solución debe poder eliminar los datos de manera segura e irrecuperable
8. La solución debe poder identificar cualquier actividad sospechosa.

2.4 Pagos Electrónicos

La sociedad moderna ha ido dependiendo cada vez más de las transacciones de pago, principalmente de transferencias de crédito, débitos directos o pagos con tarjetas. Los pagos en efectivo están disminuyendo y en cambio, aumentan los pagos a través de los sistemas basados en cuentas bancarias.

La mayoría de las transacciones de pago en la actualidad se hacen con tarjetas, bien sea de débito o crédito, y con la presencia o no de las mismas. En toda transacción se ven involucrados un gran número de entes, tanto a nivel de hardware como de software, y se debe mantener el nivel de seguridad y de integridad de datos necesario durante toda la transacción [11]. Dentro del sistema general encargado de realizar la transacción se pueden destacar los siguientes componentes: la tarjeta (bien sea para una transacción por banda, por chip o de tarjeta no presente), el dispositivo para la lectura de la tarjeta, el sistema del ente beneficiario del pago y el banco.

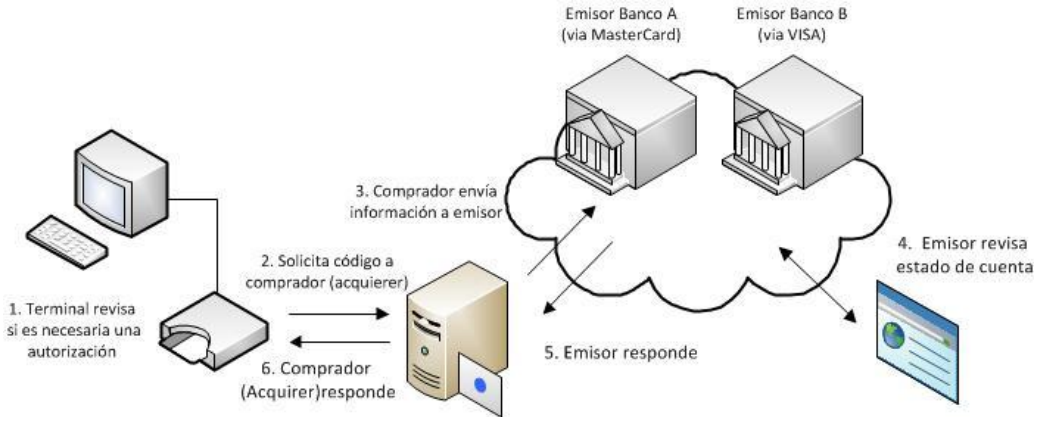

Figura 2.4: Flujo de un Pago con Tarjeta

En la Figura 2.4 se puede apreciar el flujo normal de una transacción de pago, que consiste, básicamente, en: insertar la tarjeta en un terminal de punto de venta para que el mismo pueda captar su información o ingresar manualmente la información de la misma, enviar la información al sistema del beneficiario de la transacción, este sistema se encarga de enviarlo a un host de pago o directamente al emisor de la tarjeta para que este haga la verificación con el banco del estado de cuenta.

Cada día son más los comercios y empresas que desean ofrecer sus servicios mediante el aprovechamiento de la evolución tecnológica, es decir, a través de aplicaciones web y móviles. Bajo esta demanda se ha ido evolucionando también el mundo de las transacciones de pago llegando a un punto donde, en este momento, lo pagos electrónicos son la manera más rápida y sencilla de efectuar un pago.

La aparición de los pagos electrónicos provocó un cambio radical en el mundo transaccional, ya que las formas tradicionales de pago fueron adaptadas de manera virtual, manteniendo su mismo funcionamiento, y agregándole como un beneficio adicional la alta disponibilidad, casi permanente, sin necesidad de un personal que lo maneje y la reducción de costos que implican los pagos físicos, tales como impresión de recibos o facturas [12].

Las transacciones de pago electrónico se pueden dividir en cuatro grandes categorías [13]:

- **Cajeros electrónicos:** son de sistemas en los cuales los clientes abren unas cuentas con todos sus datos en unas entidades de Internet. Estas entidades les proporcionan algún código alfanumérico asociado a su identidad que les permita comprar en los vendedores que estén asociados a las mismas.
- **Dinero electrónico:** es el dinero creado, cambiado y gastado de forma electrónica. Este dinero tiene un equivalente directo en el mundo real: la moneda. El dinero electrónico se usará para pequeños pagos. Este dinero puede ser online, si se necesita interactuar con el banco u offline si no se necesita esta interacción.
- **Cheques electrónicos:** Los métodos para transferir cheques electrónicos a través de Internet no están tan desarrollados de una forma diferente a la transferencia de fondos. Los cheques electrónicos podrían consistir algo tan simple como enviar un email a un vendedor autorizándole a sacar dinero de la cuenta, con certificados y firmas digitales asociados.
- **Tarjetas de crédito***:* Los sistemas de tarjetas de crédito en Internet funcionan de forma muy similar a como lo hacen hoy en día. El cliente puede usar si lo desea su tarjeta de crédito actual para comprar productos en una tienda virtual. La principal novedad consiste en el desarrollo del estándar de encriptación SET (Secure Electronic Transaction) por parte de las más importantes compañías de tarjetas de crédito.

2.4.1 SET

SET es una especificación global, desarrollada por Visa, MasterCard y otras compañías [14], y utiliza certificados digitales para proteger las transacciones de tarjetas de crédito que se realizan a través de Internet.

La especificación de SET enlista los siguientes requerimientos de negocio para procesamiento de pago seguro con tarjeta de crédito a través de Internet y otras redes [15]:

- Proveer confidencialidad de pago e información de órdenes de compra
- Asegurar la integridad de la totalidad de los datos que se transmiten
- Proveer autenticación de que el portador de una tarjeta es un usuario legítimo de una cuenta de tarjeta de crédito
- Proveer autenticación de que el comerciante puede aceptar transacciones con tarjetas de crédito a través de su relación con una institución financiera
- Asegurar el uso de las mejores prácticas de seguridad y de técnicas de diseño de sistemas para proteger los involucrados legítimos en la transacción de comercio electrónico
- Crear un protocolo que no dependa de mecanismos de seguridad de transporte ni que prevenga su uso
- Facilitar y promover la interoperabilidad entre proveedores de software y redes.

2.4.2 123Pago

Es la plataforma de pagos por Internet líder del mercado. Esta plataforma, además de contener una pasarela de pagos (botón de pago), brinda un servicio integral de recaudación y cobranza a empresas pequeñas, medianas y grandes [16].

En alianza con los principales bancos del país, brinda una herramienta de recaudación de pagos por medio de canales electrónicos fácil, rápida, segura y empleando múltiples medios de pago.

123Pago es una empresa venezolana fundada en 2010 con el objetivo de ofrecer la mejor solución de transacciones de pago en línea. Sus fundadores David Centeno y José Ramón Campos, son empresarios con más de 25 años de experiencia profesional en el área bancaria. Esta empresa tiene como objetivo lograr un sistema completo, seguro y de alta disponibilidad de transacciones de pago.

Dentro de los productos más importantes de 123Pago se pueden destacar:

- **Botón de Pago:** consiste en una pasarela de pagos web para un sitio virtual con múltiples medios y modalidades de pago convenientes para su cliente. Adicionalmente posee un portal administrativo para empresas, a través del cual una empresa podrá hacerle seguimiento a cada una de los pagos de sus clientes, en tiempo real si así lo requiere. Este botón (observado en la Figura 2.5) es integrado en diversas páginas web pero siempre mantiene el mismo aspecto.

Figura 2.5: Botón de Pago 123Pago

- **Taquilla Virtual:** Especialmente diseñado para empresas de servicio, que requieren cobrarles a sus clientes de forma eficiente al tiempo que reduce sus índices de morosidad, se creó esta solución que se adapta a un sin número de actividades comerciales y benéficas. Desde clubes, colegios, condominios hasta corporaciones que requieren presentar y cobrar sus servicios regularmente, 123Pago Taquilla Virtual (Figura 2.6), se configura y adapta a los requerimientos y necesidades. Los usuarios de los clientes de la empresa podrán ver en detalle sus cuentas, su historial de pago, además de seleccionar entre múltiples medios y modalidades de pago para realizar el pago de sus facturas.

Figura 2.6: Icono Taquilla Virtual 123Pago

123Pago, como canal de pago, posee conexiones con diferentes bancos, y para la comunicación con los mismos utiliza servicios web. Un servicio web es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de computadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos.

2.5 Bóveda de Tarjetas

Una bóveda digital de tarjetas de crédito es un sistema donde se almacenan, de manera tokenizada, toda la información referente a cada tarjeta. Las bóvedas deben cumplir diversos mecanismos de seguridad, incluyendo la normativa PCI, al manejar información sensible. El mecanismo más importante de seguridad es que la información completa de una tarjeta solo se pueda ser obtenida mediante un token proveniente de un solicitante válido y confiable.

Su objetivo principal es el de almacenar de manera segura información de tarjetas de crédito para facilitar aplicaciones de comerciantes a utilizarlas.

El sistema de bóveda como tal es una aplicación que cuenta con diferentes módulos, y que debe abarcar en su totalidad, la recepción de peticiones, validación de emisores, tokenización y destokenización, verificación de datos de autenticación, verificación de validez de token y envío de tarjeta destokenizada a un receptor seguro y autorizado.

2.5.1 Formato de Mensajes

Para el intercambio de mensajes entre la bóveda y cualquier otra aplicación se debe definir el formato de los mensajes, tanto de petición como de respuesta. Para ellos los tipos de mensajes más comunes a utilizar serán los siguientes:

- **ISO 8535:** es el estándar para transacciones financieras con mensajes originados en una tarjeta y especificaciones de los mensajes de intercambio. Desarrolado por la International Organization for Standardization para sistemas que intercambian transacciones electrónicas realizadas por poseedores de tarjetas de crédito [17].

  Las transacciones con tarjetas de crédito suelen cumplir, todas, el mismo esquema, por lo que el ISO 8583 busca definir un formato de mensaje y flujo de comunicación para que diferentes sistemas puedan intercambiar información de transacciones sin importar sus especificaciones.

  El ISO 8583 define mensajes para compras, extracciones, depósitos, reintegros, reversos, consultas de saldo, transferencias entre cuentas, conciliaciones totales, intercambio seguro de claves y varias funciones administrativas adicionales.

  Todos los mensajes ISO 8535 están compuestos por tres partes principales: Message Type Indicator (MTI), que se encarga de indicar el tipo de mensaje que se está enviando o recibiendo; uno o más bitmaps, para indicar qué elementos están presentes en el mensaje; y por último el campo de “data elements”, que contiene los campos del mensaje.

- **XML (Extensible Markup Language):** un mensaje XML es una jerarquía de etiquetas XML que constituyen la representación de información estructurada en la web (todos documentos), de modo que esta información pueda ser almacenada, transmitida, procesada, visualizada e impresa, por diversos tipos de aplicaciones y dispositivos [18]. La estructura de este tipo de mensajes es la observada en la Figura 2.7.

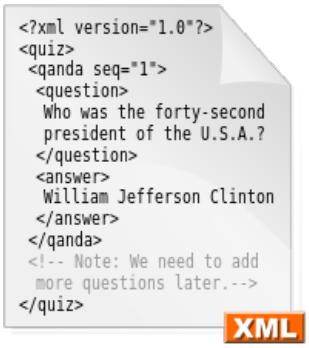

Figura 2.7: Ejemplo de un Mensaje XML

- **JSON (JavaScript Object Notation**): es un formato ligero de intercambio de datos que es fácil de leer, escribir, generar e interpretar tanto por humanos como por computadores [19]. JSON es un formato de texto que es completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos. En la Figura 2.8 se puede observar un ejemplo de un mensaje de tipo JSON.

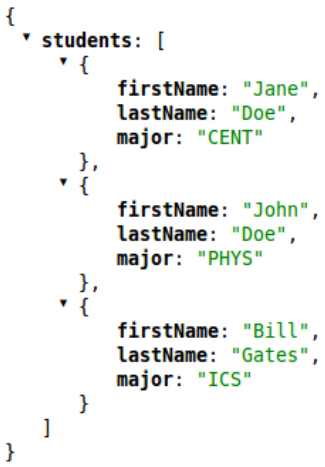

Figura 2.8: Ejemplo de mensaje JSON

JSON está compuesto por dos estructuras principales: la primera, una combinación de parámetro-valor, que representa un objeto o registro; la segunda, una lista ordenada de valores, que se conoce en la mayoría de lenguajes como un arreglo.

2.6 Herramientas de Software

Para poder satisfacer todos los requerimientos mencionados anteriormente y lograr el desarrollo de un sistema de bóveda de tarjetas, se debe hacer uso de distintas herramientas que faciliten todo el proceso.

Como objetivo principal, este sistema debe poder ser utilizado por diversas aplicaciones, de diversos tipos y que corran sobre diversas plataformas, de manera que pueda expandirse, en su mayoría, la cantidad de clientes y comercios que la utilicen.

Tomando en cuenta los avances tecnológicos, y la gran dependencia que existe actualmente al uso de Internet, se requiere también un enfoque basado en estos aspectos, es decir, que la base de la interacción entre el sistema y el exterior, sea mediante Internet.

A raíz de esto se deben tomar en cuenta muchísimos aspectos a nivel de software y hacer un estudio y selección de cuales herramientas facilitan y cumplen todos los requerimientos mencionados.

2.6.1 Java

La selección de un lenguaje para el desarrollo es fundamental para el proyecto, por lo que se debe buscar uno que satisfaga e incluso mejore las necesidades establecidas.

Java es un lenguaje de propósito general, concurrente, basado en clases y orientado a objetos [20]. Es un lenguaje estática y fuertemente tipado, es decir, que cada variable debe tener asignado un tipo de dato y que ese tipo de dato solo puede ser utilizado, como otro tipo, mediante una conversión. Adicionalmente, posee otras ventajas como que es seguro, multi-hilo, robusto y dinámico. Las principales ventajas de este lenguaje, que fueron una fuerte motivación a utilizarlo para este tipo de sistema, es que es multiplataforma (es decir, que el mismo código funciona en diversas plataformas) y que provee la posibilidad de desarrollar aplicaciones web dinámicas. Otra ventaja enorme que favoreció la elección este lenguaje es la amplia gama de librerías que posee, lo que puede agilizar muchísimo el trabajo de desarrollo.

Java también cuenta con un API (Application Programming Interface) llamado **JDBC** (Java Database Connectivity) que es un estándar para la conexión entre el lenguaje Java y diversas plataformas de bases de datos de tipo SQL y algunas otras fuentes de datos [21].

2.6.2 Java EE (Enterprise Edition)

Java EE es una plataforma que agiliza y facilita el desarrollo de aplicaciones empresariales. Tiene como objetivo proporcionar a los desarrolladores un conjunto de APIs con el fin de reducir el tiempo y la complejidad del desarrollo y de mejorar el rendimiento de las aplicaciones [22]. Esta herramienta es de gran utilidad para el desarrollo de aplicaciones y servicios web ya que ofrece APIs para sokcets web, mensajería XML, procesamiento de JSON y soporte para otros lenguajes que puedan ser integrados en la aplicación.

2.6.3 MySQL

MySQL es un sistema manejador de base de datos relacionales, de código abierto [23]. Este sistema cuenta con un alto rendimiento y una alta disponibilidad, lo que hace que sea una buena opción para integrar en aplicaciones donde se realizan muchas transacciones sobre la base de datos y se necesitan cortos tiempos de respuesta.

Además de lo mencionado anteriormente, MySQL es un sistema que es soportado por una gran cantidad de plataformas y sistemas operativos, y no necesita un hardware muy fuerte o robusto para poder ser utilizado.

2.6.4 Cipher y Key Generator

Para un sistema de tipo bóveda se necesita utilizar diversos mecanismos de cifrado que permitan la tokenización y el almacenamiento seguro de la información. Si se desea un alto nivel de seguridad en el cifrado que se realice, se necesitará también claves de alta complejidad.

Como se mencionó anteriormente utilizar Java sobre una plataforma como Java EE provee la ventaja de tener una gran cantidad de librerías y APIs a disposición. En este caso específico java provee una librería de gran utilidad llamada **javax.crypto** que cuenta con un conjunto de clases que facilitan el desarrollo de cualquier esquema criptográfico y, dentro de esta librería, se puede destacar las siguientes clases como las más utilizadas:

- **Cipher**: esta clase provee funcionalidades criptográficas para cifrar y descifrar información, y soporta los mecanismos DES, AES y RSA [24].
- **Key Generator**: esta clase provee la funcionalidad de generar claves simétricas aleatorias y seguras. Estas claves tienen dos maneras de inicializarse, y es que pueden ser generadas en base al algoritmo para el que serán utilizadas o simplemente de forma aleatoria [25].

2.6.5 Eclipse IDE

Eclipse es una plataforma de herramientas, que permiten y facilitan el desarrollo de aplicaciones en el lenguaje Java (principalmente), y que pueden ser utilizadas mediante una interfaz de usuario que puede ser ejecutado en distintos sistemas operativos [26].

Esta plataforma también soporta otros lenguajes, como por ejemplo PHP, C++ y JavaScript, y permite la creación de distintos tipos de proyectos tales como: proyectos web dinámicos, proyectos basados en Maven, aplicaciones de Java, servlets, entre otros, y soporta diferentes plataformas de desarrollo entre las que se encuentran Java EE y Eclipse Android (para el desarrollo de aplicaciones móviles).

2.6.6 Servicios RESTful

REST (Representational State Transfer) se refiere a un conjunto de principios que forman un estilo de arquitectura de la web. En este estilo arquitectónico los datos y la funcionalidad se consideran recursos y se accede mediante URI (Uniform Resource Identifiers) [27]. El estilo arquitectónico REST limita una arquitectura a una arquitectura cliente-servidor y está diseñado para utilizar un protocolo de comunicación sin estado, normalmente HTTP.

En este tipo de servicios todos los recursos se manejan utilizando un mismo conjunto de operaciones, que son GET, POST, PUT y DELETE, y se utilizan para las consultas y respuestas, mediante HTTP, entre un cliente y un servidor. Por último, los mensajes pueden ser escritos y accedidos en diferentes formatos, sin embargo, los más utilizados son mensajes XML y mensajes de tipo JSON.

2.6.7 Apache Tomcat

Apache Tomcat es un contenedor de servlets de Java de código abierto [28], y fue creado principalmente para correr aplicaciones web de tipo JSP (JavaSource Pages). Se puede instalar directamente sobre distintos sistemas operativos tales como Windows y Linux, pero también puede ser utilizado directamente desde un IDE como, por ejemplo, Eclipse. El aspecto más llamativo de Tomcat es el hecho de que utiliza muchas especificaciones de la plataforma Java EE, por lo que también provee un servidor web HTTP donde se puede correr aplicaciones de este tipo escritas en Java.

2.6.8 Glassfish

Glassfish es un servidor de código abierto encargado de soportar aplicaciones compatibles con la plataforma Java EE [29]. Este servidor también puede ser utilizado en varios sistemas operativos, sin embargo, lo más común es que sea utilizado en IDEs tales como Eclipse y NetBeans.

2.6.9 HTTP y POST

HTTP (Hypertext Transfer Protocol) es un protocolo sin estado para la transferencia de datos e información en la red de Internet y es conocido mayormente por realizar la comunicación entre servidores web y navegadores web [30]. HTTP maneja un esquema de cliente-servidor, es decir, un intercambio de mensajes que consta de peticiones y respuestas. Para la transmisión de estos mensajes posee diversos métodos, dentro de los que se puede resaltar el método POST.

El método de HTTP, POST, es utilizado mayormente para enviar datos provenientes de un formulario, desde una aplicación hacia un servidor. En este método los parámetros de la petición no son almacenados en ningún lugar, y son enviados en el cuerpo de la petición. Este

método trae como ventaja que los parámetros no son visibles al momento de ser enviados, lo que lo vuelve un poco más seguro.

2.6.10 Jersey

Jersey es un framework de código abierto para el desarrollo de servicios web de tipo RESTful, que tiene como fin dar soporte a las aplicaciones web de tipo JAX-RS. Además de ser un framework, posee un API propio que extiende le conjunto de herramientas de JAX-RS [31].

2.6.11 JPOS

JPOS es una librería de código abierto que provee facilidades para la compatibilidad entre la mensajería ISO 8583 y las aplicaciones de uso cotidiano que no utilizan ese tipo de mensajes [32]. Tiene como fin ayudar a la construcción de soluciones financiaras que realicen intercambio de mensajes para transacciones de pago con tarjetas.

A pesar de ser una librería con una amplia gama de funcionalidades, requiere de un desarrollo adicional para la integración y el uso adecuado de la misma.

2.7 Herramientas de Hardware

El sistema de bóvedas planteado durante el documento tiene como uno de los objetivos principales poder utilizarse en la mayor cantidad de plataformas posibles y poder ser consumido por la mayor variedad de aplicaciones posibles, por lo que a nivel de hardware no se existirán grandes requerimientos al momento de ser cliente del sistema, simplemente bastará con tener una aplicación, desarrollada en cualquier lenguaje, corriendo sobre cualquier sistema operativo, que sea capaz de invocar servicios web, o tener una herramienta de probador de servicios web.

Sin embargo, en el lado de servidor, los requerimientos de hardware son un poco más complicados, ya que, a nivel funcional, solo se necesita de un servidor que soporte servicios web y un servidor de base de datos MySQL pero, como deben tomarse en cuenta todos los factores de seguridad por manejarse información sensible, el ambiente donde se encuentren estos servidores debe ser seguro a nivel propio. Por lo tanto, el hardware debe poder soportar no solo los servidores, sino firewalls, VPNs (Virtual Private Network), entre otras cosas. A continuación, se realizará una breve descripción del servidor que se utilizará para el sistema (este servidor contiene otras aplicaciones, no es dedicado):

<u>Servidor de Aplicaciones</u>

- IBM SystemX 3250 M4
- Particionado en 3 máquinas virtuales con CentOS 7.0 y cada partición tiene CentOS 6.8.
- Procesador Intel Xeon E3-1200
- Memoria 32 GB DDR-3

- Cache de 8 MB L3

Servidor de la base de datos

- IBM SystemX 3200 M3
- Intel Xeon 3400 Series
- Memoria de 32 GB ECC DDR-3
- Mecanismos de cifrado para mayor seguridad de la información

Existe una interfaz de red para conectar las máquinas virtuales con el exterior y una segunda interfaz de red para configurar una VPN llamada InterCluster que está conformada por un switch FortiNet. La interfaz de la red para la base de datos es independiente de la interfaz de acceso a la red pública y privada. La base de datos está conectada a una VLAN privada. Los servidores de aplicaciones están en una DMZ (zona desmilitarizada, zona segura entre red interna y red externa). Por último, el acceso a ambos servidores está controlado por un firewall llamado Fortinet 60C, y por otras métricas de seguridad.

2.8 Trabajos Relacionados

2.8.1 VTS (Visa Tokenization Service)

El servicio de tokenización de Visa es un sistema de seguridad que se encarga de reemplazar un número de cuenta por un identificador único y seguro, un token [33]. Este token permite a los clientes de billeteras digitales y a los comerciantes ofrecer una manera segura de realizar pagos en internet. Visa también ofrece el Programa de habilitación digital de Visa (VDEP) [34] que permite a los comerciantes utilizar soluciones de pago con VTS mediante métodos de pago como: (1) Android Pay, (2) Samsung Pay y (3) Visa Checkout. Para el aprovisionamiento de tokens, un consumidor inscribe su cuenta Visa con un proveedor de pago digital (en su mayoría, carteras móviles), este proveedor solicitará un token de VTS y Visa podría compartir este token con el banco emisor. Para el uso de tokens, el proveedor de pago digital pasa el token como parte de una solicitud de autorización, Visa recibe el token y lo envía junto con la información de la cuenta o tarjeta al emisor, el emisor acepta o rechaza la solicitud y envía su respuesta a Visa, que envía la respuesta al proveedor.

2.8.2 MasterCard

Como Visa, MasterCard también tiene su sistema de tokenización. MasterCard ofrece dos soluciones: (1) Tokenización de Pagos y (2) Gateway para Servicios de Pago y Tokenización de Tarjetas [35]. Ambos tienen el mismo objetivo, transformar información sensible en tokens. En la primera solución, MasterCard asocia una referencia de transacción única una vez que se realiza una transacción exitosa, y luego esta referencia de autorización se utiliza en lugar del número de tarjeta, y el comerciante sólo necesita capturar el código de seguridad de la tarjeta (CVV) y la fecha de caducidad. Cada referencia tiene una fecha de vencimiento preestablecida de 13 meses. Por otro lado, en la Tokenización de Tarjetas, un comerciante recibirá un token en

respuesta a una solicitud de autorización. Las fichas se utilizarán en lugar del número de tarjeta, y el comerciante sólo tendrá que pedir el código de seguridad de la tarjeta y la fecha de caducidad. Cada token tiene una fecha de vencimiento preestablecida de 48 meses.

2.8.3 Conclusiones

Las soluciones anteriores tienen algunos problemas. En primer lugar, la tokenización realizada por Visa o MasterCard obliga a cada transacción a pasar a través de un paso adicional para ir a su sistema y esperar a que se realice la destokenización. Adicionalmente, estos sistemas de tokenización no permiten poseer tokens compartidos entre varios comercios. Otro problema es que empresas como Visa y MasterCard no aceptan todos los servicios de pago. Este tipo de solución, cuando el token está asociado a la compañía de tarjetas, es más utilizado en billeteras digitales.

3. Marco Metodológico

En este capítulo se definirá una forma estructurada y organizada que permita desarrollar una solución que cumpla con todas las características mencionadas anteriormente y que satisfaga todos los objetivos propuestos.

Para dicha metodología se utilizarán algunos aspectos abstraídos de la metodología Scrum, adaptándolos a la situación, principalmente porque el desarrollo será realizado por una sola persona y no por un equipo como la misma plantea, y deben tomarse en cuenta otros aspectos que sí son necesarios para este proyecto.

3.1 Scrum

Scrum es un proceso metodológico que tiene como fin agilizar y mejorar el desarrollo de un proyecto en equipo, partiendo de la primitiva de que el enfoque principal se debe hacer en la funcionalidad que más beneficios aporta al proyecto.

Entre los beneficios que aporta se encuentran: la flexibilidad de adaptarse a los cambios que puedan producirse en el proyecto, ya que propone el solapamiento de procesos; permite que el cliente pueda utilizar ciertos aspectos de la aplicación sin necesidad de que se encuentre terminado el desarrollo; ayuda a predecir los tiempos, ya que se maneja con las denominadas "iteraciones", que consisten en entregas parciales y regulares; entre otras cosas

De esta metodología se adaptarán varios aspectos a este desarrollo:

- Se permitirá el solapamiento de actividades, es decir, si es necesario modificar un proceso, así se haya considerado finalizado o se encuentre en otro punto del desarrollo, se modificará.
- Se realizarán primero los aspectos más importantes y críticos del sistema, de manera que se pueda garantizar desde el principio que funcionan de manera correcta.
- Se utilizará el mecanismo de iteraciones, es decir, entregas constantes de bloques pequeños de la aplicación, en cortos periodos de tiempo.
- Cada iteración debe ser funcional, de manera que el desarrollo vaya siendo incrementado sin afectar su funcionamiento
- Existirá únicamente dos roles en el desarrollo, el primero es el de "product owner" (poseedor del proyecto) que a la vez realizará el desarrollo, y el segundo es del de "stakeholder", que es el encargado de recibir y supervisar las iteraciones.

3.2 Implementación de la Metodología

Antes de utilizar la implementación de Scrum mencionada anteriormente, existen unos prerrequisitos que deben ser cumplidos para poder comenzar el desarrollo del proyecto:

- Se debe establecer los requerimientos necesarios para comenzar el desarrollo y preparar los ambientes y herramientas a utilizar.
- Se debe haber definido el diseño de la arquitectura del sistema y los módulos que por los que será compuesto.
- Se deben establecer los tiempos deseados y hacer una planificación en función a eso.

3.3 Pruebas y Resultados

Debido a que la información con la que se tratará en este proyecto es sensible, se deben realizar diversas pruebas que verifiquen su buen uso y protección, tanto para asegurar que el sistema funciona de la manera que se espera, como para asegurar que no se está tratando la información de manera inapropiada.

Para este sistema lo que se tomará en cuenta al momento de hacer las pruebas es:

- Si se está almacenando la información en la base de datos y cómo se está haciendo.
- Si se está realizando bien la tokenización y la des-tokenización.
- Si se está realizando bien la autenticación entre las aplicaciones y el sistema.
- Verificar que puede agregarse un nuevo comercio y asignarle al mismo su propia bóveda.
- Comprobar que se pueda realizar de manera exitosa el añadir una nueva tarjeta para que sea tokenizada.
- Verificar que puede realizarse un pago nada más proveyendo el token.
- Validar que el sistema rechaza los tokens inválidos y las peticiones de comercios que no poseen bóveda.
- Comprobar que se generan llaves diferentes para cada bóveda.
- Comprobar que los tokens son únicos.
- Verificar que se estén cumpliendo las normas PCI.
- Comprobar que un comercio puede ser asociado a una bóveda existente.
- Asegurarse de que todos los servicios son accesibles y consumibles.
- Garantizar respuesta de todos los servicios.

4. Marco Aplicativo

Una vez planteado el problema y realizada la investigación teórica se tienen todas las bases necesarias para comenzar el desarrollo del sistema. Este capítulo tiene como fin describir y mostrar de manera detallada todos los pasos que fueron realizados al momento de diseñar la solución, definir el flujo del sistema, hacer uso de las herramientas y realizar el desarrollo.

4.1 Análisis Preliminar

Para realizar el diseño y desarrollo del sistema se debieron tomar en cuenta varios aspectos importantes. Primero, como se mencionó anteriormente, se necesitaba que este sistema tenga una respuesta rápida y sea de alta disponibilidad. Segundo, el diseño debía permitir el cumplimiento de las normas PCI. Tercero, debía ser compatible y consumible por la mayoría de aplicaciones posibles. Cuarto, debía ser sencillo de utilizar para un usuario. Por último, debía poder comunicarse, además de con las aplicaciones, con sistemas de pago.

4.2 Diagrama de Casos de Uso

En la Figura 4.1 se puede observar el diagrama de casos de uso basado en los requerimientos del sistema desarrollado, este diagrama es de nivel 1.

Para el esquema del sistema en general se tendrán dos posibles usuarios: administrador, que puede ser un administrador de la empresa de pagos o del comercio que está utilizando la bóveda (esto es decisión de la empresa de pago que provee el servicio de bóveda), y un cliente, que es el que está utilizando la aplicación del comercio. En este escenario, únicamente el administrador podrá agregar un nuevo comercio y generarle una bóveda personal, asociar un comercio a una bóveda existente o eliminar un comercio y desactivar su bóveda, en caso de ser única.

Un cliente de la bóveda solo podrá tener opción de agregar una nueva tarjeta, lo que significa generar un token asociado a la tarjeta y, de invocar un pago, mediante la selección y el envío de dicho token generado, para que puedan ser recuperados los datos de la tarjeta y enviados al medio de pago. El caso de administrar un token tiene la responsabilidad de, en base los datos recibidos, enviarlos al método correspondiente y realizar los datos correspondientes.

Como se tiene previsto que este sistema será consumido por una aplicación de un comercio, se asume que el manejo de la sesión lo realizará dicha aplicación, por eso no existe un caso de iniciar/cerrar sesión.

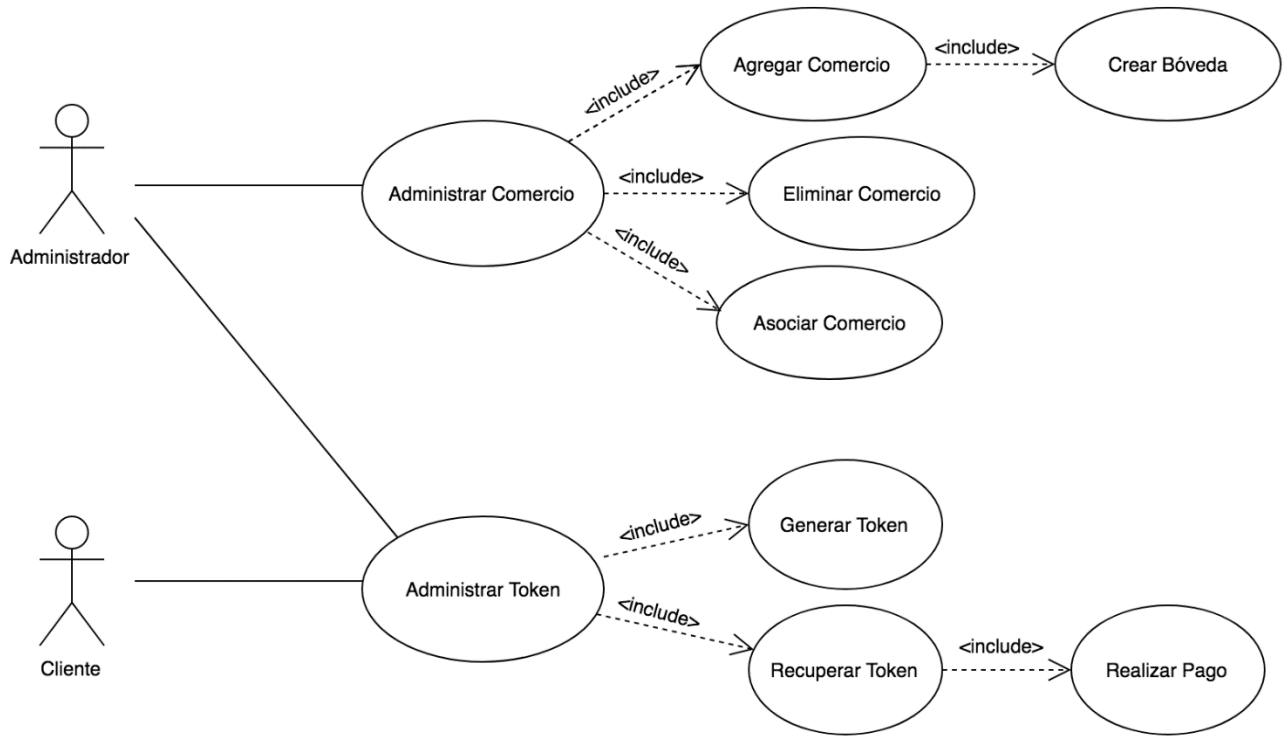

Figura 4.1: Diagrama de Casos de Uso del Sistema

4.3 Diseño de la Solución

En la Figura 4.2 se puede apreciar la arquitectura propuesta para la solución del sistema a desarrollar. Dicha solución consta de 3 web services principales, ubicados todos dentro del mismo servidor, y una base de datos a la cual podrán acceder los mismos. También se puede observar que la aplicación podrá acceder al sistema utilizando mensajes de tipo JSON, XML y ISO 8583. Estos mensajes serán de tipo petición/respuesta y abarcarán toda la información necesaria en la comunicación.

Uno de los servicios del sistema, el de autenticación y administración de la información, se comunicará de manera directa y bidireccional con el sistema de pago de 123Pago, quien se encargará de procesar el pago y posteriormente devolverle la respuesta al servicio y este a la aplicación del cliente, por lo que no existe una interacción directa entre la aplicación del cliente y el sistema de pago.

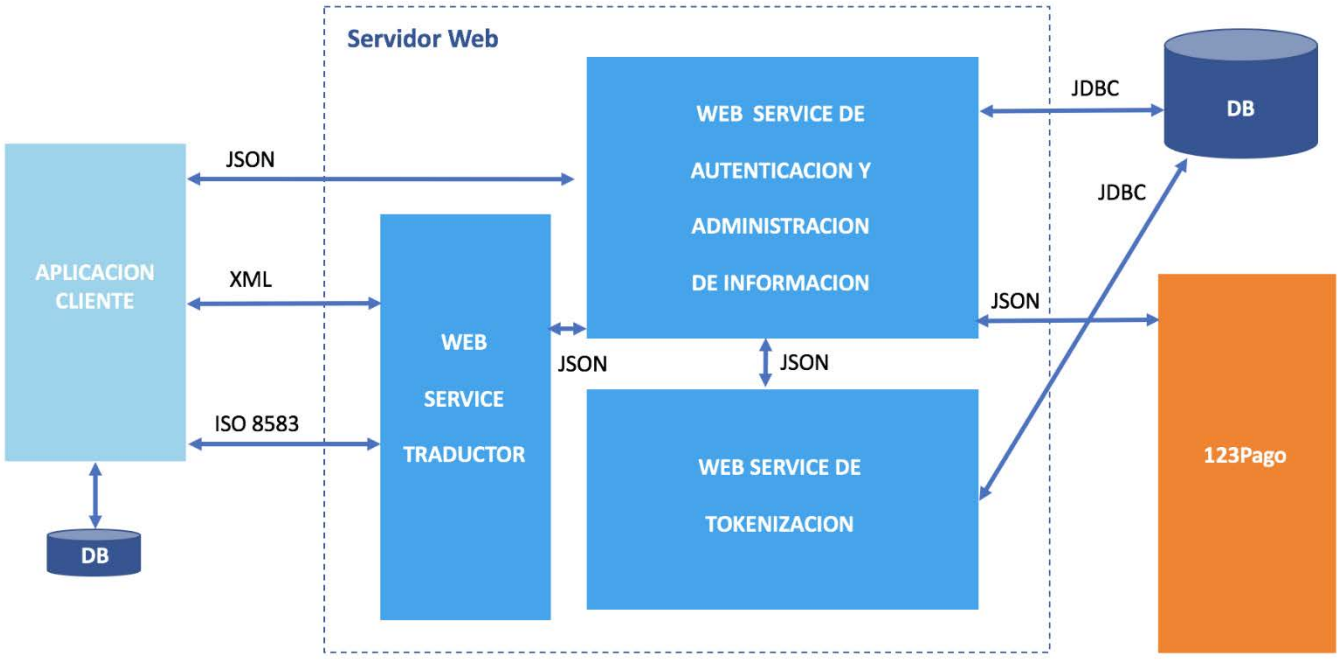

Figura 4.2: Arquitectura del Sistema

Entre la aplicación del cliente y el sistema de bóveda existirá un mecanismo de autenticación que será detallado más adelante, pero que verificará que la aplicación que está consumiendo la bóveda es una aplicación válida.

Por último, la aplicación del cliente que invoca el sistema de bóveda debe poseer una base de datos o alguna manera de almacenar el token que será generado. Como la bóveda garantizará que la aplicación a la que se le entrega el token es una aplicación autorizada, el poseedor de esta aplicación debe encargarse de tener sus propios mecanismos de autenticación, acceso y seguridad ya que el token quedará bajo su responsabilidad.

4.4 Flujo del Sistema

El flujo general que debe cumplir el sistema, a nivel funcional, es el siguiente:

1. El administrador del comercio lo registra mediante el servicio de autenticación y administración y se le devuelve que el registro fue exitoso y el id de su bóveda.
2. Se habilita una opción de pago con bóveda en la aplicación del comercio y el usuario de la aplicación añade una tarjeta de crédito y se genera un token, mediante el servicio de tokenización.
3. El sistema devuelve un token, que no es conocido por él, pero se almacena en la aplicación y en el siguiente paso, le sale el mismo (que otorga un poco de información con respecto a la tarjeta) como una opción de pago.

4. El usuario selecciona la tarjeta, y esto invoca al servicio de administración de información, que se encarga de realizar las validaciones, la destokenización y de enviar la información al sistema de pago.
5. El sistema devuelve el resultado de la transacción al servicio y el servicio se lo envía a la aplicación del cliente.

4.5 Desarrollo de la Solución

Una vez definido el flujo y el diseño de la solución se comenzó con el desarrollo del sistema, el cual será explicado de forma detallada.

4.5.1 Ambiente y Herramientas

Como se mencionó en el Capítulo 2, Java es un lenguaje que permite el desarrollo de aplicaciones web dinámicas y es multiplataforma, además de otros beneficios que ofrece, por lo cual se decidió que era la opción ideal para utilizar en esta solución

En la Figura 3.2 se observó que la arquitectura propuesta está compuesta por distintos servicios web. Por lo tanto, estos servicios, que como ya se mencionó serán desarrollados en Java, serán del tipo RESTful y se comunicarán mediante mensajes de tipo JSON. El desarrollo de estos servicios se hizo sobre el framework Jersey.

El desarrollo se realizó en el IDE Eclipse y los servicios fueron desplegados y probados sobre dos tipos de servidores: un servidor de Apache Tomcat 7.0 y un servidor Glassfish 3.1. Para la base de datos se utilizó una base de datos relacional mediante el sistema manejador MySQL.

El sistema soporta la interacción con las aplicaciones cliente en tres tipos de formatos: JSON, XML y ISO 8583. Sin embargo, la comunicación entre el sistema de bóveda y el sistema de pago siempre se realizará en formato JSON.

4.5.2 Base de Datos

La base de datos se desarrolló con una estructura simple y sencilla pero que fuera capaz de almacenar todos los datos que son necesarios para el sistema de tokenización de tarjetas. La estructura de la base de datos es la observada en la Figura 4.3:

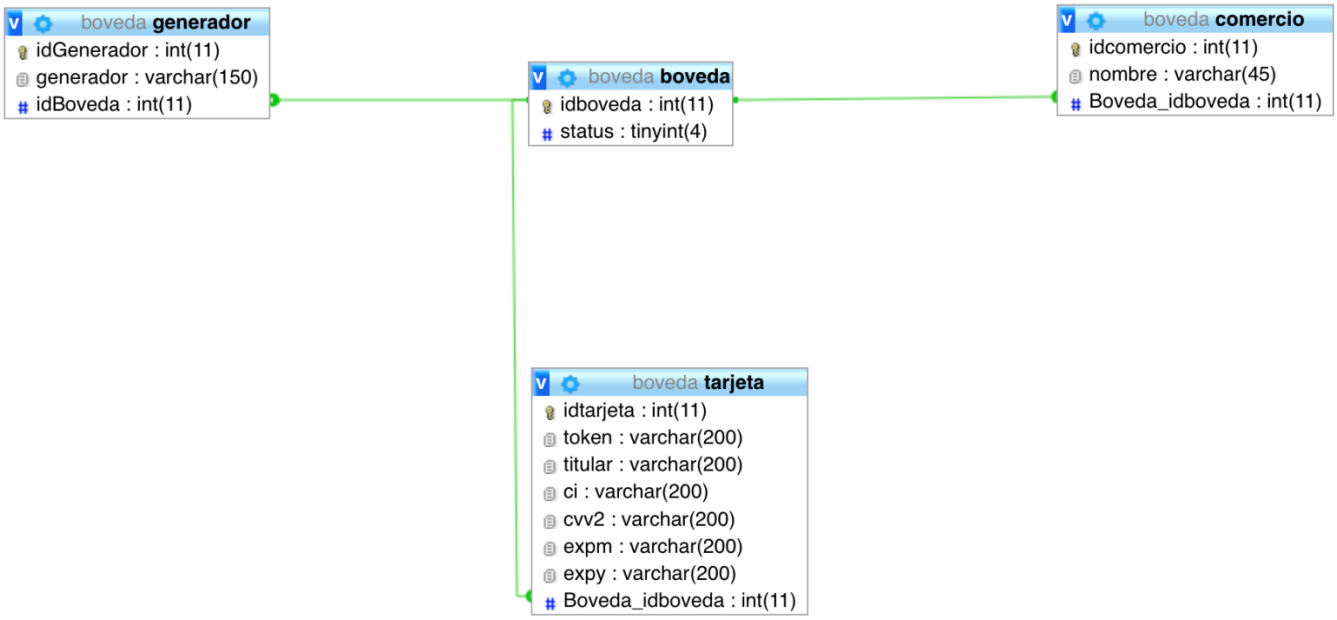

Figura 4.3: Diseño de la Base de Datos

La estructura consiste en:

- Una tabla principal, que es la tabla de bóveda, que contiene una lista de ids para las bóvedas existentes y un status, que se encontrará en 0 cuando la bóveda este inactiva y en 1 cuando este activa.
- Una tabla de comercio, donde solo se amacena el id del comercio, su nombre y el id de la bóveda a la que se encuentra asociado.
- Una tabla llamada generador, donde se almacena de manera segura la llave asociada a cada bóveda, por lo que está compuesta por un id de generador, un generador que será la llave y un id de la bóveda a la que está asociado.
- Una tabla de tarjeta, donde es almacenada, de manera cifrada, toda la información asociada a una tarjeta, más un id de la bóveda a la que pertenece.

4.5.3 Servicio de Tokenización

Este servicio tiene como fin generar un token asociado a los datos de una tarjeta. Sin embargo, también se utiliza. Su estructura debe ser bastante simple, y por mantener un nivel de seguridad, solo tiene acceso a la base de datos para insertar el registro de un nuevo token generado. Este servicio cuenta con una clase principal que maneja las peticiones, una clase que realiza la encripción, una clase de configuración del servicio y una clase de conexión con la base de datos mediante el API de JDBC. Para el desarrollo de este servicio se utilizaron librerías tales como: *javax.crypto.*, javax.ws.rs.*, org.json.*,* entre otras.

Como se puede observar en la Figura 4.4, el servicio consumirá y producirá información en formato JSON, así que se debe especificar de esta manera.

```
@Consumes({MediaType.APPLICATION_JSON})
@Produces({MediaType.APPLICATION_JSON})
```

Figura 4.4: Especificación de Formato que Consume y Produce el Servicio

Posteriormente, para continuar el desarrollo lo primero que se realizó fue definir la ruta del servicio y el método que se utilizaría para enviar y recibir información.

```
@Path("ServicioToken")
public class ServicioToken {

    @POST
    @Path("generar")
    @Consumes({MediaType.APPLICATION_JSON})
    @Produces({MediaType.APPLICATION_JSON})
    public JSONObject generarToken(JSONObject json){
```

Figura 4.5: Ruta del Servicio y del Método

```
@javax.ws.rs.ApplicationPath("rest")
public class ApplicationConfig extends Application {
```

Figura 4.6: Ruta de la Aplicación

En la Figura 4.5 se observa la ruta del servicio, dónde para acceder al servicio general se debe agregar a la URL el texto */ServicioToken* y, para acceder al método **generarToken** debe agregarse */generar.* Aquí también se puede apreciar que el método de HTTP que se utilizará para la comunicación es el método *POST*.

Por otro lado, en la Figura 4.6 se observa que se agrega una ruta de aplicación */rest*, esto se hace debido a que al ser un servicio web de tipo REST se deben agregar algunas configuraciones a la clase **Application** de Java.

Finalmente, la URL para acceder a este servicio es la siguiente, dónde *<servidor>* es la dirección IP o el nombre del servidor (con la especificación del puerto si es necesario) y *TokenWS* es el nombre el proyecto que contiene el servicio web:

URL = *http:// <servidor>/ TokenWS/rest/ServicioToken/generar.*

En el sistema de bóveda este servicio será invocado por otros servicios internos, sin embargo, se creó de igual manera una ruta independiente para el mismo por si se desea reutilizar en otro proyecto o con otra finalidad.

A nivel funcional, este servicio será invocado mediante la URL mencionada, y recibirá un mensaje de tipo JSON como el de la Figura 4.7, por lo que el primer paso en el desarrolló fue

hacer que la aplicación pueda entender y desglosar el JSON para tener la información necesaria. Este JSON está compuesto por todos los datos de la tarjeta (*número*, *cvv2*, *expm* que es mes de expiración y *expy* que es el año de expiración), los datos del titular (nombre de *titular* y *cedula* de identidad) y el número del comercio al que pertenece el usuario, representado en el mensaje como *nocom*.

```
{
    "titular": "Daniela",
    "cedula": "24699188",
    "numero": "7887788778877887",
    "cvv2": "555",
    "expm": "05",
    "expy": "2018",
    "nocom": "11"
}
```

Figura 4.7: JSON que Recibirá el Servicio

Una vez obtenidos estos datos el servicio recuperará el id de la bóveda a la que pertenece el comercio indicado, mediante el id que fue suministrado. Posteriormente, tomará el número de tarjeta y lo dividirá en 3 partes que son: los primeros 4 dígitos, los 8 intermediarios y los últimos 4. Un punto importante es que al ser una bóveda de tarjetas de crédito únicamente, se asume seguro que el número de tarjeta siempre tendrá 16 dígitos.

Los 8 dígitos intermediarios que se obtuvieron de la división del número de tarjeta serán encriptados mediante el uso de los métodos de la clase de encriptación y posteriormente concatenados nuevamente a los 4 dígitos iniciales y 4 dígitos finales, generando así el **token**.

La clase de encriptación se apoyará en la clase *javax.crypto.KeyGenerator* para generar claves aleatorias, especificándole como se puede apreciar en la Figura 4.8 el tipo de algoritmo de cifrado que se utilizará, que en este caso fue **AES**, la codificación que quiere utilizarse, que fue **UTF-8,** y que después utilizará el algoritmo de encriptación al momento de ejecutrarse y el tamaño de la clave que en este caso es de **128 bits**.

```
private final String ALGORITMO = "AES";
private final int LONGITUD = 128;
private final String CODIFICACION = "UTF-8"
```

Figura 4.8: Parámetros para Generar la Llave

Esta clave solo se generará la primera vez que se inserte una tarjeta en la bóveda, el resto de las veces dicha clave se recuperará de la base de datos y se invocará a la clase de encriptación con la clave como parámetro.

Para la encriptación de los dígitos intermediarios la clase lo hará apoyándose en la clase *javax.crypto.cipher*, con la clave ya obtenida y sabiendo que el algoritmo que se utilizará es AES.

Teniendo ya el **token** generado, el servicio utilizará la misma clave para encriptar el resto de los datos (que si serán encriptados en su totalidad sin ser subdivididos) almacenará todos los datos en la base de datos y finalizará devolviendo el **token** generado en formato JSON, como el ejemplo mostrado e la Figura 4.9.

Figura 4.9: JSON que Responderá el Servicio

Los 8 dígitos intermediarios que fueron encriptados se vuelven 32 dígitos alfanuméricos, y se decidió que los textos encriptados fueran alfanuméricos con el fin de ampliar la gama de compatibilidad.

4.5.4 Servicio de Autenticación y Administración de Información

Este servicio contiene toda la lógica y administración del sistema de bóveda en sí, y esta subdividido en dos grandes partes.

Primero, al igual que en el servicio anterior se debe realizar la configuración de la aplicación, la configuración de la conexión a la base de datos mediante JDBC y se debe definir la ruta general del servicio que será la observada en la Figura 4.10.

```
@Path("Autenticacion")
```

Figura 4.10: Ruta de Acceso al Servicio de Autenticación y Administración

Este servicio en general, utilizará algunas librerías similares al anterior, pero también incluirá algunas otras tales como *java.io* y *java.net*, para la conexión, lectura y escritura de datos con otro servicio web.

La primera parte de este servicio consta de la administración de la información de los comercios que poseen bóveda, es decir, es donde se van a crear, asociar o eliminar comercios de las bóvedas. En el planteamiento de la solución se asumió que cada comercio tendrá una bóveda particular, sin embargo, se desea dejar abierta la opción a que a una bóveda pueda asociarse más de un comercio.

Para cumplir estos requerimientos, esta sección tendrá tres opciones, dónde la opción que el cliente desea ejecutar debe ser indicada en el mensaje de petición.

La **opción 1** es para agregar un comercio y crearle su propia bóveda. Este escenario es bastante simple ya que el cliente solo tiene que enviar su nombre de comercio y la opción 1, así que la petición sería algo similar a la observada en la Figura 4.11.

```
{
    "opcion" : "1",
    "nombre" :"pruebacom"
}
```


Figura 4.11: JSON para Agregar Comercio

En este caso el servicio se encarga de agregar una nueva bóveda, colocarla como activa, agregar un nuevo comercio y asociar el id de esta nueva bóveda a dicho comercio. Después de realizar esto el servicio devuelve un JSON como el observado en la Figura 4.12, que contiene el id de comercio que se le generó para este sistema y el id de la bóveda que va a utilizar.


```
{
    "idComercio" : "5",
    "idBoveda" : "7"
}
```


Figura 4.12: JSON de Respuesta al Agregar un Comercio

La **opción 2** consiste en eliminar un comercio y desactivar su bóveda, siempre y cuando sea una bóveda particular, de no ser el caso la misma quedará activa. Los datos del comercio se eliminarán de la base de datos, pero los datos de la bóveda se mantendrán por si se desea, en un periodo corto de tiempo, ser asociada a otro comercio. En este escenario el servicio solo responde con un mensaje de éxito o error en el proceso de eliminación. Los formatos de petición y respuesta serían los observados en las Figuras 4.13 y 4.14, respectivamente.


```
{
    "opcion" : "2",
    "nombre" :"pruebacom",
    "id" : "4"
}
```


Figura 4.13: JSON para Eliminar Comercio


```
{
    "status" : "Comercio eliminado de la boveda exitosamente"
}
```


Figura 4.14: JSON de Respuesta al Eliminar un Comercio

Por último, la **opción 3** permite al cliente asociar un comercio a una bóveda que ya se encuentra creada. Para esto el cliente solo tiene que enviarle al servicio el número de opción, el nombre del comercio y el id de la bóveda a la que se quiere asociar, como se puede apreciar en la Figura 4.15.

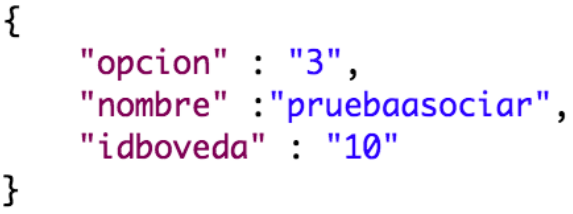

Figura 4.15: JSON para Asociar Comercio a una Bóveda

En este caso, el servicio creará el comercio en la base de datos, con el id de bóveda indicado y, en caso de que la bóveda estuviera inactiva, cambiaría su *status a 1*. La respuesta sería la observada en la Figura 4.16, que consiste únicamente en el id de comercio que fue creado.

Figura 4.16: JSON Respuesta de Asociar Comercio a una Bóveda

En esta sección del servicio el método de comunicación HTTP será POST, y la URL que se deberá agregar es **/modificarComercio,** que será la misma para las tres opciones, ya que la opción será especificada en el JSON. Asumiendo que *<servidor>* cumple las mismas condiciones que en el servicio anterior y que **/AuthenticationWS** es la ruta para el nombre del proyecto, la manera de acceder al servicio sería la siguiente:

URL = *http://<servidor>/ AuthenticationWS/rest/modificarComercio*

La segunda parte de este servicio consiste en la administración del **token**. Este servicio es el encargado de interactuar directamente con las aplicaciones clientes y de recibir sus peticiones, bien sea para la administración de comercios o para la administración de token. En esta parte también existen dos opciones, la primera que consiste en invocar al servicio de tokenización y la segunda que consiste en recuperar los datos de una tarjeta, mediante un token recibido y enviarlos al medio de pago. En este caso la opción también será enviada dentro del JSON.

La **opción 1** es muy sencilla ya que solo consiste en la invocación del servicio de tokenización, basado en los parámetros que envía el usuario mediante un JSON, que se puede observar en la Figura 4.17, que es el mismo que se envía al servicio de tokenización, pero con el parámetro adicional que indica la opción.

```
{
    "opcion" : "1",
    "titular": "Daniela",
    "cedula": "24699188",
    "numero": "7887788778877887",
    "cvv2": "555",
    "expm": "05",
    "expy": "2018",
    "nocom": "1"
}
```


Figura 4.17: JSON para Generar un Token

Para realizar la conexión con otros servicios web se hará uso de las librerías mencionadas anteriormente, por lo que todas las conexiones a otros servicios se verán similares a la mostrada en la Figura 4.18

```
URL url = new URL("http://localhost:8080/TokenWS/rest/ServicioToken/generar");
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setDoOutput(true);
conn.setRequestMethod("POST");
conn.setRequestProperty("Content-Type", "application/json");

String input = "{\"titular\":" + req.optString("titular") + ",\"cedula\":" + req.optString("cedula") +

OutputStream os = conn.getOutputStream();
os.write(input.getBytes());
os.flush();

BufferedReader br = new BufferedReader(new InputStreamReader(
        (conn.getInputStream())));

String output;
while ((output = br.readLine()) != null) {
    out = new JSONObject(output);
}
conn.disconnect();
```

Figura 4.18: Ejemplo de Conexión con Otro Servicio Web

En esta conexión se define el **URL** que se consumirá, se define que se espera una respuesta, que el método para la petición es el método **POST** y que el tipo de contenido será **JSON**. Posteriormente se crea un **String** para armar el JSON, que al haber indicado que sería el tipo de dato el servicio que se está consumiendo lo leerá como tal, se crea un **stream** de entrada y uno de salida para escribir y leer datos, y por último un buffer para leer la respuesta. Dicha respuesta se asigna a un JSON y finalmente se cierra la conexión.

El formato de la respuesta del servicio será exactamente el mismo de la Figura 4.9 en el servicio de tokenización. La llamada al servicio de tokenización se hace desde este servicio (en vez de hacerla directamente desde el cliente) ya que aquí se agrega una capa de verificación en la base de datos de que el comercio efectivamente posee una bóveda.

La **opción 2** si es un poco más extensa, ya que es la que recuperará el **token**, realizará la destokenización y enviará los datos al sistema de pago. En este caso la petición del cliente debe contener, la opción, el token y un conjunto de datos que deben ser generados por el comercio y que se necesitan para el momento de realizar el pago, la petición será como la observada en la Figura 4.19.

En esta petición **emailCliente** es simplemente el email del usuario de la aplicación cliente, el **nai** es el número identificador de la transacción, el **idCobro** es un id específico para los cobros, el **nbproveedor** es un identificador el tipo de permisos que tiene el comercio sobre el sistema de pago, el concepto es la razón de la transacción, el monto es el total, el **idpromoción** es en caso de que exista el sistema de pago lo identificará y hará las modificaciones necesarias sobre el monto, la **ipCliente** es la dirección IP de la aplicación que está consumiendo el servicio, el **nombreMdp** corresponde el nombre del medio de pago (por ej. tarjeta de crédito), y el **oneTimePwd** irá en 0 en caso de que no haya restricciones de tiempo y en 1 en caso de que sí (de haber restricciones de tiempo son netamente manejadas por el sistema de pago).

Todos estos atributos adicionales son generados por el comercio y administrados por el sistema de pago, en este caso el sistema de bóveda solo se encarga de transmitirlos entre ambos, para poder agregar los parámetros que sí dependen de él.


```
{
    "opcion" : "2",
    "token": "788747e5af1743da862d7324433f01a86db47887",
    "nocom" : "11",
    "emailCliente":"devivodany@gmail.com",
    "nai":"12345678",
    "idCobro":"1",
    "nbproveedor":"2",
    "concepto":"monto",
    "monto":"250",
    "idpromocion":"1",
    "ipCliente":"8.8.8.8",
    "nombreMdp":"prueba",
    "oneTimePwd":"0"
}
```


Figura 4.19: JSON que se Recibe para Recuperar un Token

De esta petición el servicio solo tomará el campo de **opción**, el de **token** y el de **nocom** (número de comercio), del resto todos los campos se mantendrán igual hasta el momento de ser enviado al sistema de pago.

Se tomará el token y se enviará al método de destokenización. Este método no es accesible mediante una URL directamente por medidas de seguridad, es un método interno del servicio y sólo será invocado después de verificar que el comercio es válido y es quien dice ser.

El método de desencriptación funciona a la inversa que el de tokenización, es decir, una vez obtenido el token se realiza la búsqueda en la base de datos en la tabla de tarjetas con el mismo, ya que, cada token es único e irrepetible. Se recupera de la base de datos toda la información asociada a la tarjeta a la que corresponde el token y se recupera el id de la bóveda a la que la tarjeta pertenece, y con este id, se recupera de la tabla generador la clave para desencriptar.

Se invoca a la clase de encriptación, que también estará contenida en este servicio, y pasando llave recuperada como parámetro se invoca el método de desencriptación. Todos los datos de la tarjeta se desencriptan de manera completa, exceptuando el token, al que de nuevo se le debe realizar una división donde, los primeros 4 dígitos quedan intactos, se desencriptan los 32 dígitos intermedios para obtener los 8 dígitos originales, y los últimos 4 dígitos también quedan intactos. Posteriormente, se concatenan las 4 divisiones del token y se obtiene el número original de tarjeta.

Estos parámetros son devueltos al método de recuperar y con ellos se rellenan los campos restantes en el JSON que debe ser enviado al sistema de pago. En la Figura 4.20 se puede observar como queda dicho JSON.

En este JSON los campos del comercio son los enviados por el cliente al momento de realizar petición y los datos de la tarjeta y del titular son los recuperados de la bóveda de datos. En este caso, el cliente siempre será el mismo que el TH (Tarjeta habiente) ya que no se permitirá que un cliente almacene una tarjeta en la bóveda que pertenezca a otro titular.

Una vez se tenga la respuesta del sistema de pago se devolverá la misma, en el mismo formato, a la aplicación del cliente. Esto es debido a que, al sistema de bóveda, le es indiferente si el pago fue aprobado o rechazado, el solo se encarga de recuperar los datos de una tarjeta y asegurarse que los mismos sean utilizados de manera segura.

En esta sección del servicio el método de comunicación HTTP también será POST, y la URL que se deberá agregar es **/adminToken,** que será la misma para las dos opciones, ya que la opción será especificada en el JSON. Asumiendo que *<servidor>* cumple las mismas condiciones mencionadas y que **/AuthenticationWS** es la ruta para el nombre del proyecto, la manera de acceder al servicio sería la siguiente:

URL = *http://<servidor>/ AuthenticationWS/rest/adminToken*

```
{
    "cedulaCliente":"24699188",
    "cedulaTH":"24699188",
    "nombreCliente":"Daniela",
    "nombreTH":"Daniela",
    "emailCliente":"devivodany@gmail.com",
    "nai":"12345678",
    "idCobro":"1",
    "direccion":"bello monte",
    "nuTarjeta":"7887788778877887",
    "cvv2":"555",
    "mesVencimientoTarj":"5",
    "anioVencimientoTarj":"2018",
    "nbproveedor":"2",
    "concepto":"monto",
    "monto":"250",
    "idpromocion":"1",
    "ipCliente":"8.8.8.8",
    "nombreMdp":"prueba",
    "oneTimePwd":"12"
}
```


Figura 4.20: JSON que se Enviará al Sistema de Pago

4.5.5 Servicio Traductor

Este tercer y último servicio tiene como objetivo principal hacer que el sistema de bóveda pueda interactuar con aplicaciones cliente que no manejen el formato de mensajes JSON.

Como se mencionó en el Capítulo 2, los formatos más comunes para este tipo de transacciones son JSON, XML y ISO 8583, así que este servicio recibirá peticiones en los formatos que no son JSON para traducirlas a formato que sí es JSON y así se pueda invocar el Servicio de Autenticación y Administración de la Información. Este servicio será divido en dos partes, una para XML y una para ISO 8583, así que existirán dos URLs diferentes.

La traducción de **XML** a JSON es la más sencilla, ya que la librería **org.json.*** provee una clase de XML que tiene una función de *XML.toJSON( ),* como se puede observar en la Figura 4.21, la que como único parámetro requiere el XML en forma de **String** (así es recibido en el servicio). La única diferencia es que en el XML se debe agregar un campo adicional que indique a qué módulo del servicio desea acceder (el de administración de comercios o el de administración de token), este campo sería de tipo Int e irá en 0 cuando se quiera acceder a la administración de comercios y en 1 cuando se desee que sea a la de token. La Figura 4.22 es un ejemplo de una petición en XML.

```
jsonx = XML.toJSONObject(xml);
```

Figura 4.21: Función que Convierte XML en JSON

```
<site>1</site>
<opcion>2</opcion>
<token>788796b7ae9156abe98d1c8495d5bc598c917887</token>
```

Figura 4.22: Ejemplo de Invocación al Sistema en XML

Cuando el Servicio de Autenticación y Administración finaliza lo que se le haya solicitado, devuelve la respuesta a este servicio, para que el mismo vuelva a traducirla a XML y la envía a la aplicación del cliente.

En esta sección del servicio el método de comunicación HTTP será POST, y la URL que se deberá agregar es **/xml2json,** que será la misma para las tres opciones, ya que la opción será especificada en el JSON. Asumiendo que *<servidor>* cumple las mismas condiciones que en el servicio anterior y que **/TranslateWS** es la ruta para el nombre del proyecto, la manera de acceder al servicio sería la siguiente:

URL = *http://<servidor>/TranslateWS/rest/Translate/xml2json*

La segunda parte de este servicio es la encargada de recibir las peticiones en **ISO 8583** y transformarlas en formato JSON.

Lo primero que se tomó en cuenta en esta parte es el hecho de que el formato ISO 8583 no puede adaptarse a todo tipo de peticiones, sino únicamente las de tipos transaccional, por lo que para este caso las peticiones en ISO solo serán a la parte de administración de token en el Servicio de Autenticación y Autorización, por ende, solo podrán hacerla aplicaciones cliente y la configuración de comercio deben ser solicitadas a la empresa o realizadas por otra aplicación.

Para esta traducción se utilizó una librería llamada **org.jpos.iso.*.** Esta librería permite la facilidad de, mediante un archivo *.xml* que indique el formato del mensaje **ISO 8583**, leer el ISO 8583 recibido por el servicio e ir extrayendo uno a uno cada campo en formato **String**, lo que hace que se pueda ir llenando el mensaje **JSON** de manera muy sencilla.

El archivo de configuración se realizó adaptando los campos que se necesitan a los campos originales que pudieran ser más parecidos, es decir, se utilizarán los campos necesarios del formato original, y se dará un manual al cliente de cómo adaptar su información a este formato.

Por ende, este servicio recibe un mensaje en formato ISO 8583, tal como el representado en la Figura 4.23, lo traduce a JSON mediante este servicio, el mismo servicio hace la llamada a otro servicio, pasándole el JSON creado, el servicio externo lo recibe, ejecuta las operaciones indicadas en la petición y genera una respuesta que también se encuentra en formato JSON,

esta respuesta es devuelta al servicio de traducción, que se encarga de, mediante el mismo archivo de configuración .xml, crear ahora el mensaje ISO 8583 en base a la respuesta JSON recibida. El mensaje final tendrá la misma estructura del mensaje recibido.

```
0200B220000000100000000000000080000020123400000001000001107221801234560 6A5DFGR021ABCDEFGHIJ1234567890
```

Figura 4.23: Ejemplo de un Mensaje ISO 8583

En esta sección del servicio el método de comunicación HTTP será POST, y la URL que se deberá agregar es **/iso2json,** que será la misma para las tres opciones, ya que la opción será especificada en el JSON. Asumiendo que *<servidor>* cumple las mismas condiciones que en el servicio anterior y que **/TranslateWS** es la ruta para el nombre del proyecto, la manera de acceder al servicio sería la siguiente:

URL = *http:// <servidor>/TranslateWS/rest/Translate/iso2json*

4.6 Aspectos de Seguridad

La seguridad en este sistema es un punto muy delicado que será abarcado mediante tres maneras principales y en conjunto de validaciones en el desarrollo que ya fueron mencionadas.

La primera es el ambiente donde se encontrará ubicado el sistema, que como se explicó en el Capítulo 2, tiene sus propios mecanismos de seguridad y control de acceso, por lo que la información sensible se encontrará en un entorno seguro.

La segunda es que el diseño de este sistema está realizado para poder ser utilizado con un certificado SSL, que se obtendrá al momento de que el sistema se encuentre en producción, lo que significa que en vez de utilizar el protocolo HTTP para el intercambio de información, se utilizará el protocolo HTTPS.

La tercera es un mecanismo de tokenización que se agregó al Servicio de Autenticación y Administración de información, que tiene como objetivo, que al momento de que un comercio sea asociado a una bóveda o que se le sea creada su propia, se le generará un token aleatorio que será devuelto en la respuesta conjunto al i**dComercio** y al **idBóveda**, por lo que el formato de la respuesta de este servicio será el mostrado en la Figura 4.24. La aplicación cliente debe enviar este token en la cabacera de HTTP en todas las peticiones que realice. El token será generado a raíz de la encriptación del nombre de comercio utilizando una llave aleatoria y el mismo mecanismo de cifrado del Servicio de Tokenización. Estas llaves serán almacenadas en una tabla nueva que se encontrará en la misma base de datos pero que no estará relacionada con ninguna de las otras tablas y tendrá la estructura mostrada en la Figura 4.25.

```
{
    "idComercio" : "2",
    "idBoveda" : "5",
    "token": "96b7ae9156abe98d1c8495d5bc598c"
}
```


Figura 4.24: JSON Respuesta de Agregar o Asociar Comercio

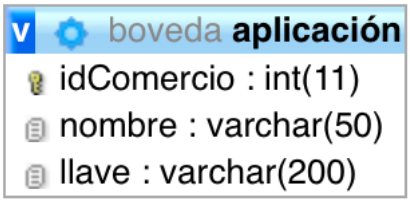

Figura 4:25: Estructura de Tabla para Autenticación

En todas las peticiones que reciba el servicio, se obtendrá el token de la cabecera, se descifrará con la llave y si el resultado coincide con el nombre del comercio se permitirá el acceso, en caso contrario se negará.

Se decidió usar un token de identificación fijo y no uno variable por sesión debido a que las sesiones en este sistema son muy cortas, en casi todos los casos las peticiones serán transacciones de cortos tiempos y poco intercambio de mensajes.

5. Pruebas y Resultados

El presente capítulo tiene como objetivo exponer el conjunto de pruebas que se le realizaron al sistema para comprobar su funcionamiento y cuáles fueron los resultados de las mismas.

Para este sistema, como se explicó anteriormente, se asumirá que los datos ingresados son siempre válidos, es decir, a nivel gramatical, por lo que no existirán validaciones de formato ni de casos borde de las peticiones recibidas.

En este caso, los objetivos principales que debe cumplir el sistema son dos: ser seguro y ser efectivo, por lo que las pruebas realizadas se realizarán con ese enfoque.

5.1 Seguridad

5.1.1 PCI

Uno de los objetivos más importantes que se planteó al comienzo del proyecto fue el de mantener el cumplimiento de las normas PCI, por lo que la Tabla 5.1 muestra el esquema de los requisitos que fueron cumplidos y cómo fueron logrados.

Requisito	*Alcanzado*	*Justificación*
Instalar y mantener una configuración de cortafuegos para proteger los datos de los propietarios de tarjetas.	Sí	Como se indicó en la Sección 2.7, el servidor donde se encontrará el sistema cuenta con un cortafuegos.
No usar contraseñas del sistema y otros parámetros de seguridad predeterminados, provistos por los proveedores.	Sí	Todas las contraseñas y parámetros se seguridad, tanto del servidor como del sistema, fueron personalizadas con un alto nivel de complejidad.
Proteger los datos almacenados de los propietarios de tarjetas.	Sí	Nuevamente, como ya se mencionó, el servidor de base de datos tiene sus propias medidas de seguridad y garantiza un ambiente seguro y protegido.
Cifrar los datos de los propietarios de tarjetas e información confidencial transmitida a través de redes públicas abiertas.	Sí	En el modelo actual no se encuentra implementado el certificado seguridad, pero se contempló en el diseño que el mismo será incluído, por lo que se adquirirá e implementará en el momento que el sistema pase a producción. Esto ayudará a garantizar la seguridad de los mensajes en redes públicas.

Usar y actualizar regularmente un software antivirus.	Sí	Este requisito también está previsto en los servidores donde se encontrarán el sistema y la base de datos.
Desarrollar y mantener sistemas y aplicaciones seguras.	Sí	El sistema posee varios mecanismos de seguridad que se irán actualizando y mejorando constantemente.
Restringir el acceso a los datos tomando como base la necesidad del usuario de conocer la información.	Sí	Uno de los principales objetivos del sistema. Mediante la tokenización se controla que los usuarios tengan el menor acceso a la información posible, de hecho, el máximo de conocimiento que tendrá el usuario será el token de una o más tarjetas, token de acceso, su idComercio asignado y su idBoveda asignado.
Asignar una identificación única a cada usuario que tenga acceso a un punto del sistema.	*Sí*	Los usuarios administradores tendrán una identificación única para acceder al servidor donde se encontrará ubicado el sistema. Los usuarios de tipo cliente, que son los comercios, tendrán una identificación única para invocar al sistema.
Restringir el acceso físico a los datos de los propietarios de tarjetas.	Sí	Controlado por el servidor.
Rastrear y monitorizar todo el acceso a los recursos de la red y datos de los propietarios de tarjetas	Sí	Para las transacciones de pago se solicita el IP de la aplicación cliente de manera de crear un registro y tener una verificación (en el sistema de pago). Esta misma información es almacenada en el registro del servidor del sistema de bóveda.
Probar regularmente los sistemas y procesos de seguridad	Sí	Se realizaron pruebas durante todo el desarrollo del sistema, se realizaron pruebas de confirmación en la fase de prueba del sistema y se encuentra planificado realizar pruebas mensuales para mantener controlados los niveles de seguridad.
Mantener una política que contemple la seguridad de la información	Sí	Tokenización.

Tabla 5.1: Cuadro de Verificación de Normas PCI

Como se pudo observar, las normas PCI fueron cumplidas, en aspecto general, en su totalidad. Se tiene como fin, para futuros desarrollos y evoluciones de este sistema, ir mejorando y ampliando la gama de mecanismos de seguridad.

Adicionalmente se tomaron en cuenta varias consideraciones del P2PE (mencionado en el Capítulo 2), de las cuales se tomaron dos como bases fundamentales en el diseño y desarrollo de este sistema. La primera, garantizar que el ambiente donde va a utilizarse la información desencriptada es seguro y no es accesible para cualquiera. La segunda, el uso de metodologías de cifrado seguras, lo que es una de las motivaciones principales para el uso de AES como algoritmo de cifrado.

5.1.2 Generador de Llaves

Uno de los aspectos que garantiza mayor nivel de seguridad en este sistema es el hecho de que cada tarjeta se encuentre asociada a una bóveda (que puede pertenecerle a uno o más comercios) pero, como cada bóveda poseerá su propia llave para cifrar, se garantiza que el token será único, es decir, que una misma tarjeta tokenizada en comercios con bóvedas diferentes generará dos tokens diferentes. Se puede observar un ejemplo en la Figura 5.1.

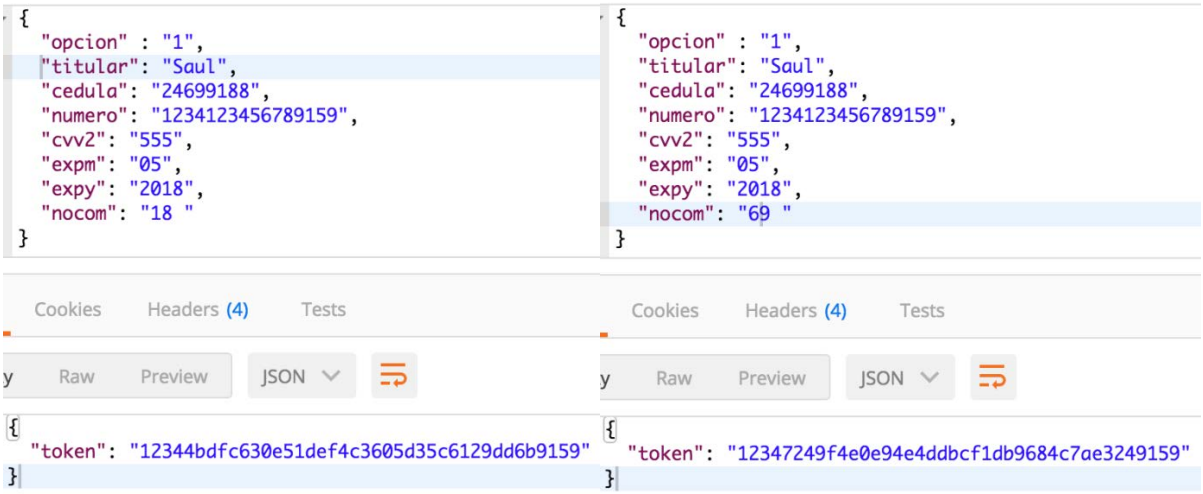

Figura 5.1: Comparativa de Tokens

Como las llaves serán generadas de forma aleatoria, para que sean más seguras, se debe tener un generador de llaves robusto, que genere llaves complejas y diferentes, sin seguir ningún patrón. Para comprobar esto se realizó la inserción de más de 100 comercios y se comprobó, como se puede apreciar en el extracto de la Figura 5.2, que todas las llaves generadas eran diferentes.

idGenerador	generador	idBoveda
31	7b1e20d9443fc49b7856f6490242 4d5e	30
32	6fa2e484c3afdf692a8daa9f804eefb5	31
33	478b75f9ba60c17a0b1146d21b127a88	32
34	1bb351c1696adb0c0d9bffbd4a758cd8	33
35	b8021530f09d01cf22e37765bb7a51e8	34
36	b452d002cbd85b4ae1cf73ce92df91c5	35
37	9c0cffaefe16943cc1fd581f1e907ad6	36
38	9e3636b39a4303a45014cbe270051352	37
39	b9ca693128ac2108573ed57c16fe03ed	38
40	12d5424f2d61f3ebe4967f1745182735	39
41	a1f620fd7da5f5801ed34babbce8b2c1	40
42	9363d07db0e5e2bd8308a28d0e2eebd0	41
43	a66baa33b3d1d64ed8f456e4be3912c5	42
44	8bd8d760bea76d05f7d058ada2f7a342	43
45	5f8afa665d61f9779c3581b178315122	44
46	daa559aa6343cfad26d8f8d3e51dd296	45
47	4e20f9d5638fcefc640da238b8a151db	46
48	496f2d90e14070e74f08060fe8f20295	47
49	64922e08a4c70b13e2bcc91d14398333	48
50	42fd7e33cdb90e0a57526e6c360d57ec	49
51	4bdb27bcd1cc8a87ee313b9d227ac571	50
52	44bf5e402e412681bc487d5445a0d7dd	51
53	30a8f202e684130645bde632b6e1d648	52
54	1a174cd821ef1d7d0e7b432f76ad3fa6	53
55	1e669db424602db83b46688b129a7812	54
56	bf54566feeba8c490643abd026f89bfd	55
57	43a0c34cd0b4ec3cb1e3666449b95360	56

Figura 5.2: Llaves para Distintas Bóvedas

5.1.3 Token en Cabecera

El mecanismo de asociar un token para cada comercio y que el mismo sea enviado en la cabecera de todas las peticiones que se realicen al servicio añade un poco de seguridad extra al intercambio de mensajes, que como ya se mencionó, estará previsto que utilice el protocolo HTTPS.

Se realizaron pruebas donde de intento de acceso a los servicios de administración de token, con un token de cabecera modificado (modificaciones completas o parciales) o ausente, y en todos los casos el acceso fue negado, mientras que si se utiliza el token correcto se puede acceder a los métodos y opciones del servicio. En la Figura 5.3 se puede observar la diferencia entre el acceso con un token válido, que sí genera respuesta y uno inválido que niega el acceso.

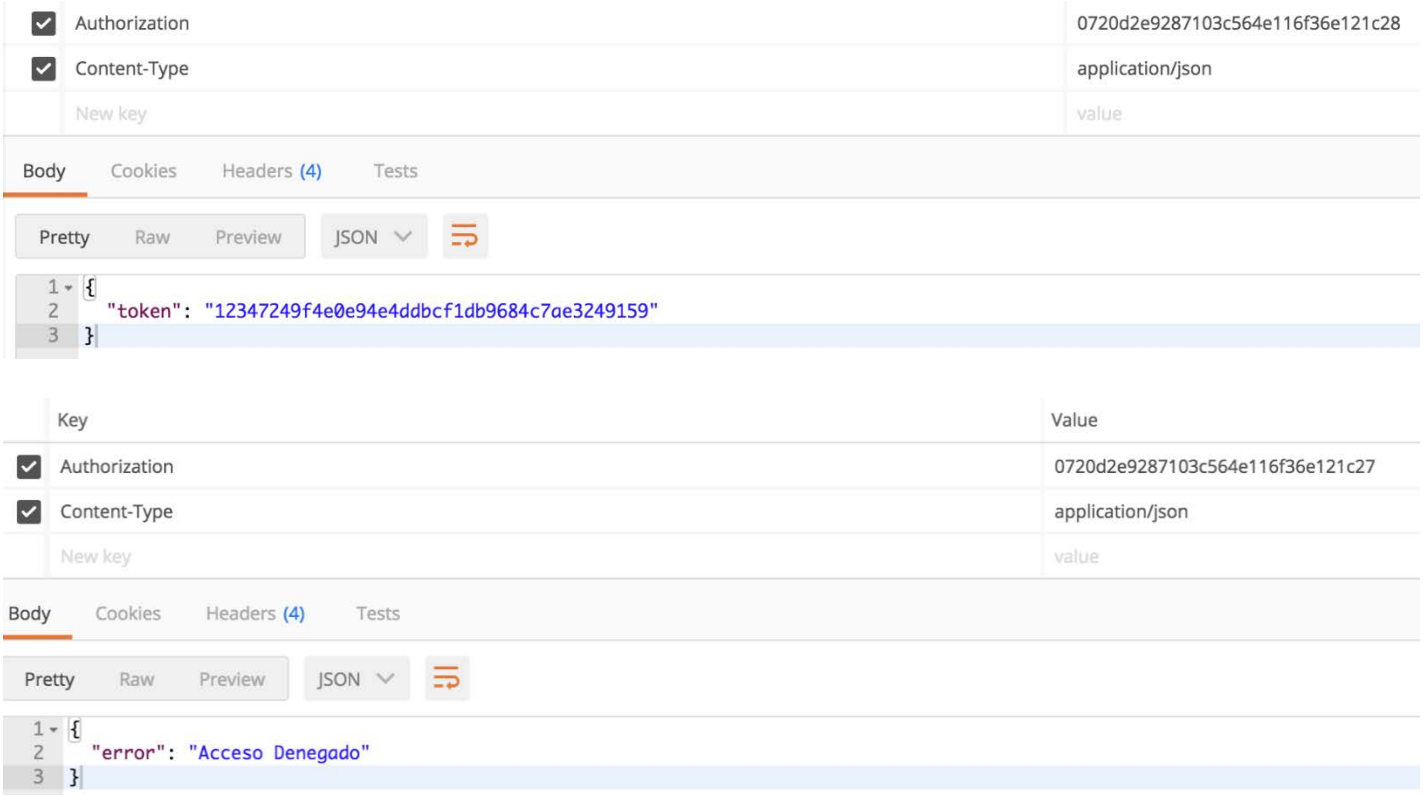

Figura 5.3: Resultado Token de Cabecera con un Dígito Diferente

5.1.4 Validaciones Adicionales

A nivel de código se desarrollaron y añadieron algunas restricciones adicionales a las ya mencionadas, casi todas como un paso extra para la verificación y el acceso a los datos, entre estas restricciones se puede destacar:

- Siempre se verifica que el comercio posee una bóveda antes de permitirle generar o recuperar un token.

- Se valida que no se estén generando llaves diferentes para una misma bóveda.
- Al momento de recuperar un token se valida que el mismo se encuentre en la bóveda del comercio que lo solicita.
- Al momento de asociar un comercio se garantiza que la bóveda se encontrará activa.

Mediante el transcurso de las diferentes pruebas se comprobó que estas restricciones se estaban cumpliendo siempre.

5.2 Funcionamiento

Otro aspecto fundamental del sistema es que debe cumplir de manera adecuada sus funciones siempre, ya que un pequeño error podría generar un problema mayor. Por este motivo, en la fase de pruebas y resultados, se realizaron numerosas operaciones y peticiones a los servicios para comprobar su funcionamiento.

5.2.1 Servicio de Tokenización

Este servicio fue el más probado, desde hacerlo directamente, hasta hacerlo mediante los otros servicios, ya que es la base de todo el sistema de bóveda.

Una vez terminado el Servicio de Tokenización, y su respectivo método de des-tokenización, se realizaron numerosas pruebas (más de 200) donde se probaba que del número de tarjeta se conseguía un token, mediante métodos de encriptación, y que de ese mismo token se podía recuperar el número de tarjeta. Este método fue 100% efectivo en todos los casos, siempre que un token fue generado, pudo ser recuperada la información de la tarjeta asociada al mismo.

También mostró la misma garantía cuando fue consumido desde otro servicio y no directamente de un cliente.

5.2.2 Administración de Comercios

En este punto era de gran importancia demostrar que las tres posibles opciones tenían un buen funcionamiento. Estas secciones son relativamente sencillas por lo que se pudo hacer gran cantidad de pruebas.

Para agregar un comercio se debía asegurar que se creaba el comercio y la bóveda con el id que indica la respuesta, y que se generaba y almacenaba el token de autenticación de comercio. Este método, después de varias modificaciones, logro hacerse efectivo en todos los casos. El único error que pudo arrojar fue el observado en la Figura 5.4, al intentar agregar un comercio que ya estaba agregado.

```
{
  "error": "Comercio Duplicado"
}
```


Figura 5.4: Error de Comercio Duplicado

Para eliminar un comercio debía verificarse que, en caso de poseer una bóveda única, la misma cambiaría su status a 0, como se puede observar en la Figura 5.5, que significa que no se encuentra activa. Este método también cumple su función al 100%.

idboveda	status
1	1
2	0

Figura 5.5: Status de una Bóveda Activa vs. una Desactiva

Por último, en el método de asociar una bóveda a un cliente lo más importante de verificar era que se logrará, a nivel de registros, que dos o más comercios tuvieran el mismo id de bóveda a la que pertenece. Este método falla únicamente cuando se desea asociar un comercio existente que posee bóveda personal a otra bóveda, ya que en este caso se deben realizar verificaciones adicionales como, por ejemplo, si se migrarán los datos de la bóveda anterior a la que se está asociando; este caso quedará para futuras investigaciones. En la Figura 5.6 se puede apreciar dos comercios diferentes asociados a una misma bóveda.

idcomercio	nombre	Boveda_idboveda
12	prueba	1
13	Daniela	1

Figura 5.6: Dos Comercios Diferentes Asociados a una Misma Bóveda

6.2.3 Administración de Token

Este servicio es uno de los puntos más críticos del sistema, del cual se debe obtener una respuesta segura siempre. Para ambos métodos pertenecientes a él, es decir, el de generar un token y el de recuperar un token, se realizó una comprobación de token de cabecera, que como se mencionó anteriormente, resultó ser muy efectiva en todos los casos.

Para la sección de generar un token, como consiste simplemente en consumir el servicio de Tokenización, de una manera más controlada, el patrón para realizar pruebas fue el mismo que para el Servicio de Tokenización, una carga masiva de tarjetas de crédito y probar que siempre se generarían tokens diferentes, a pesar de que las tarjetas se encuentren en la misma bóveda, como se puede observar en la Figura 5.7.

47	12347249f4e0e94e4ddbcf1db9684c7ae3249153
48	3234528a16c7e36fc877755d42e19d408cf29153
49	7834528a16c7e36fc877755d42e19d408cf29153
50	5634528a16c7e36fc877755d42e19d408cf29153

Figura 5.7: Diferentes Tarjetas en una Misma Bóveda

El ejemplo propuesto en la Figura 5.7 tiene como fin resaltar, que números de tarjetas similares (es decir, diferentes únicamente por uno o dos dígitos) tendrán un token asociado muy similar también, sin embargo, se pudo comprobar mediante las pruebas realizadas, que la única manera de que existan dos tokens iguales es que hayan sido generados con la información de la misma tarjeta y pertenecientes al mismo comercio.

El rendimiento de esta sección a lo largo de las pruebas fue bastante satisfactorio, los errores producidos siempre fueron por un dato mal colocado y, como se ha mencionado, para este sistema se asumió que la aplicación cliente que lo consuma será la encargada de verificar los datos enviados. Por otro lado, para la recuperación del token el proceso fue un poco más extenso ya que es el servicio más completo.

Lo primero que se realizó fue una nueva comprobación de que, al recibir una petición con un token de tarjeta, se podían recuperar los datos de la misma para ser utilizados posteriormente en el pago. Al igual que las pruebas realizadas al momento de crear le método de tokenización, como se garantiza que el token es único, la recuperación y des-tokenización es efectiva en todos los casos probados, siempre y cuando no exista alteración del token, por lo que se puede garantizar un buen funcionamiento. El resto de los datos que deben ser enviados al sistema de pago por porte del sistema de bóveda, son enviados por el cliente al momento de realizar la petición, nuevamente, estos datos se asumirán válidos así que no se realizarán verificaciones adicionales en este paso.

Lo siguiente es la comprobación de la invocación del servicio de pago. Para esto se verificó que la invocación al sistema de pago mediante la bóveda generara el mismo resultado que introducir los datos manualmente en un probador, y se probaron transacciones aprobadas y rechazadas para comprobar que la respuesta se recibe de igual manera en ambos casos.

Se realizaron alrededor de 40 transacciones en las que se comparó el resultado del pago ingresando los datos completos en una herramienta para probar el servicio de pago, como se puede observar en la Figura 5.8, se contraste con el hecho de ingresar únicamente los datos del comercio y el token asociado en la tarjeta en el sistema de bóveda, como se puede observar en la Figura 5.9. Como puede apreciarse la respuesta del sistema de pago coincidió, esto ocurrió en todos los casos (siempre y cuando los datos con los que se invoca hayan sido similares.

Figura 5.8: Invocación al Servicio del Sistema de Pago Mediante el Probador

Figura 5.9: Invocación Mediante el Servicio de Autorización y Administración de Información

Al comprobar que tanto la recuperación del token, como la invocación del sistema de pago se son ejecutadas de manera correcta e incluso todas las validaciones propuestas anteriormente también, se puede garantizar que el servicio de recuperación de token para realizar un pago con tarjeta de crédito se ejecutará de forma correcta, siempre que los datos ingresados sean válidos.

5.2.4 Servicio de Traducción

Para este servicio las pruebas consistieron en realizar distintas peticiones, previamente hechas en JSON y comprobar que los resultados fueran los mismos.

En el caso de la traducción de XML, es 100% efectiva ya que solo se encarga de convertir el mensaje XML yen JSON y viceversa. Se realizaron 25 peticiones mediante XML y en los 25 casos la traducción se realizó correctamente y la invocación/respuesta al servicio también.

Por otro lado, en la traducción del ISO 8583, como el formato es mucho más estricto existen más posibilidades de una falla al momento de construirlo (por parte del cliente) en el formato especificado lo que hace que, como el sistema asume que los datos son válidos, pueda haber algún parámetro incorrecto. Las aplicaciones cliente tendrán el formato de configuración del ISO para las peticiones, sin embargo, es probable que existan algunas fallas, de hecho, se realizaron prueba con aplicaciones de tipo cliente, dónde no se conocieron las maneras que utilizaban para crear los mensajes ISO, y en estas pruebas 3 de cada 10 peticiones tenía un error de formato. Todas las pruebas fueron realizadas en un servidor Apache Tomcat.

5.3 Tiempos de Respuesta

Para las funciones de registro y administración de los tokens y de los comercios, los tiempos de respuesta son prácticamente inmediatos. Solo pueden generarse fallas adicionales o esperas más largas si se presenta algún problema en el dispositivo que realiza la solicitud al sistema de bóveda o alguna falla en la red de Internet mediante la cual se está haciendo la misma.

Por otro lado, para la recuperación de un token y la realización de un pago, los tiempos de espera pueden ser mucho más variables ya que no solo dependen de los dispositivos, la red y el sistema, sino que también tiene gran influencia el tiempo de respuesta del banco. Las transacciones bancarias suelen tener un tiempo de respuesta no mayor a 1 segundo, sin embargo, pueden existir fallas o retardos en la transmisión, tanto como al momento de solicitar como al de recibir la respuesta.

Para medir un promedio de los tiempos de respuesta del sistema se realizaron pruebas mediante el sistema de simulación de 123Pago, que se encarga de realizar el proceso de invocación de pago utilizando el sistema de bóveda, como lo haría cualquier aplicación y, si el pago se aprueba, se encarga de realizar el reverso de manera inmediata.

123Pago utiliza, para el procesamiento de pagos, dos (2) bancos diferentes, Banesco y Banplus, cuando el sistema de bóveda invoca a 123Pago para realizar el pago, es el mismo 123Pago que se encarga de decidir por que banco realizar la transacción. Para las pruebas se tomó un universo de 5 transacciones y se probó tanto con tarjetas de créditos pertenecientes al banco que se realiza el pago, como con tarjetas de otros bancos.

La Figura 5.10 muestra los resultados obtenidos, redondeando a un decimal.

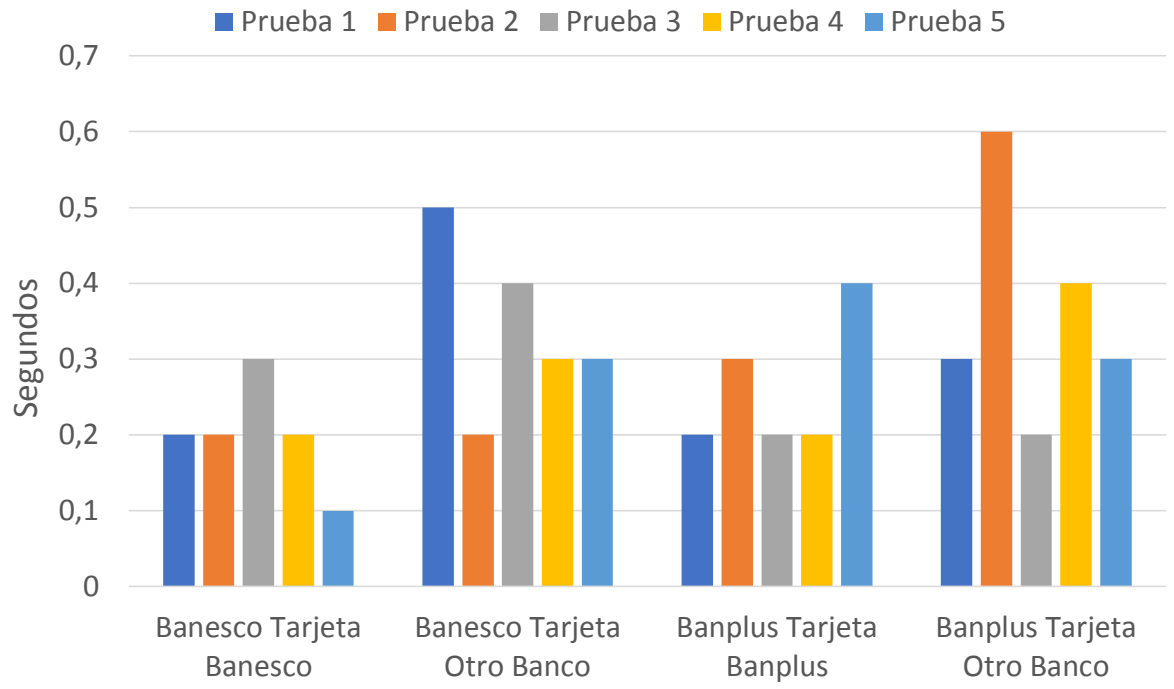

Figura 5.10: Gráfico de Tiempo de Respuesta Promedio de un Pago

Una vez realizadas las pruebas se estimó que el tiempo promedio de la transacción es de unos 0,29 segundos.

A pesar de que el sistema se encontrará en un servidor que posee control de acceso y que se encargará de limitar la cantidad de peticiones que el mismo reciba, se decidió realizar pruebas, mediante el mismo sistema de simulación de 123Pago, permitiendo el acceso ilimitado al sistema de bóveda, para observar los tiempos de respuesta del sistema cuando recibe más de una petición. Para este caso, se ignoró el banco que procesa el pago, y solo se realizaron transacciones con distintas tarjetas de manera simultánea. Se realizaron 7 lotes de pruebas para ver las variaciones de los tiempos de respuesta. El primer lote se realizó una sola petición, en el segundo se realizaron 5 peticiones, en el tercero 10, en el cuarto 15, en el quinto 30, en el sexto 40 y en el séptimo 50.

En la Figura 5.11 se puede observar los resultados obtenidos mediante estas pruebas.

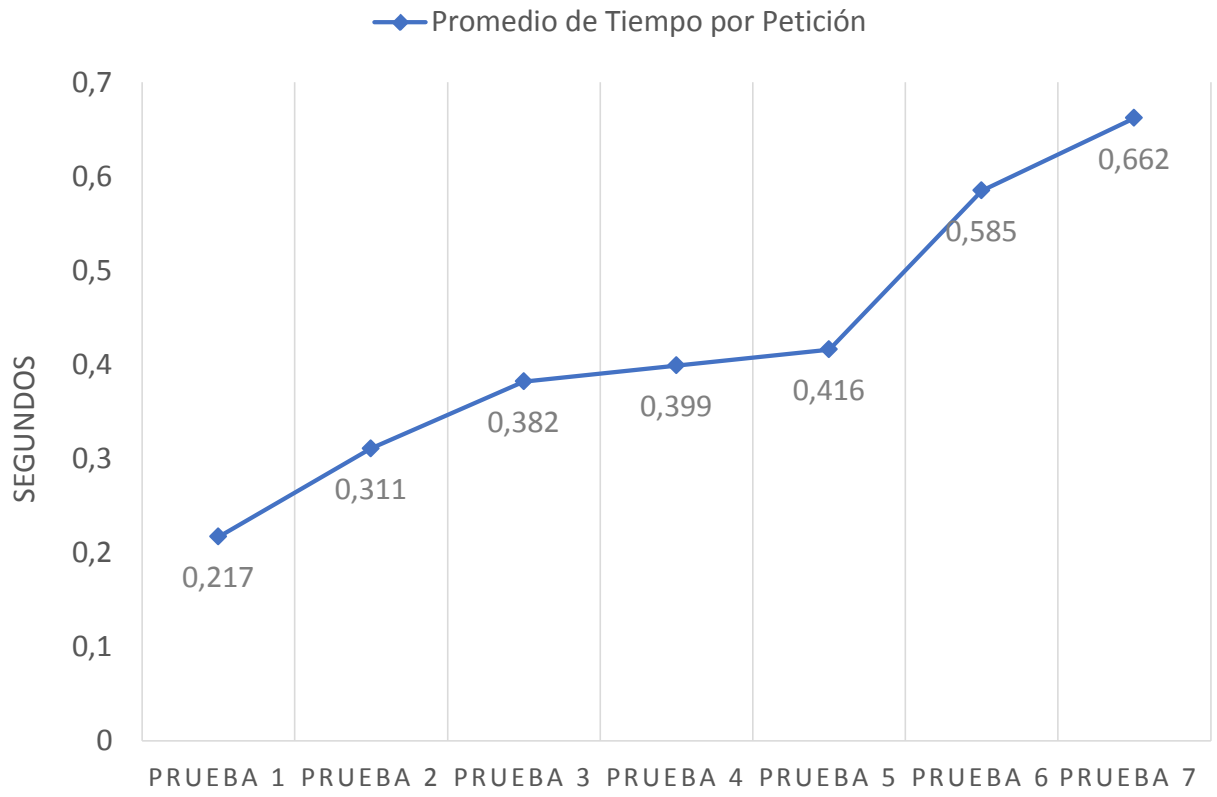

Figura 5.11: Promedio de Tiempo de Respuesta por Petición

A medida que la cantidad de peticiones simultáneas aumenta también aumenta el tiempo de respuesta que tiene cada petición, sin embargo, en todos los casos las peticiones de las transacciones fueron respondidas satisfactoriamente.

6. Conclusiones

La necesidad de los comercios de adaptarse a los avances tecnológicos es cada vez mayor, lo que ha hecho que, en su mayoría, estos busquen innovarse y comercializar sus productos y servicios vía aplicaciones web y móviles. A raíz de esto, ha surgido una gran cantidad de sistemas de pago electrónico que permiten que dicha innovación sea posible, sin embargo, al tener que realizar estos pagos mediante aplicaciones, el ingreso de los datos se produce de forma manual, lo que es repetitivo y poco práctico para comercios que necesiten que los pagos se realicen de manera rápida. Por este motivo, se realizó un trabajo de investigación y diseño que lograra una solución segura, eficiente y compatible con la mayor cantidad de plataformas posibles, para crear un sistema de bóveda que permitiera recuperar, de manera rápida, la información de una tarjeta mediante el uso de un token y, enviar dicha información a un sistema de pago.

Se definió una metodología para el desarrollo de esta solución, la cual utilizaba algunos extractos de la metodología Scrum, con el fin de agilizar el proceso y garantizar el funcionamiento de las partes constitutivas y del conjunto total.

Se desarrolló un sistema en Java, sobre el framework Jersey y con apoyo de diversas librerías, que está compuesto por tres (3) servicios principales, todos del tipo RESTful. Se decidió utilizar el esquema de servicio web para lograr que, cualquier aplicación, independientemente de su plataforma, pueda consumir este sistema. Los servicios fueron divididos en: un servicio de tokenización, para la producción de un token asociada a una tarjeta de crédito; un servicio de administración, para manejar los comercios y las bóvedas existentes en el sistema, controlar las peticiones de generar o recuperar un token y recuperar la información de una tarjeta de crédito para enviarla a un sistema de pago; y por último un servicio que convirtiera en JSON las peticiones recibidas en otro formato.

Adicionalmente, se tomaron diferentes medidas de seguridad en base de las normas PCI, y se tomó en cuenta, durante el diseño del proyecto, que el servidor donde se encontrará ubicado el desarrollo, implemente sus propias medidas de seguridad y que, posteriormente, cuando el proyecto se coloque en producción, se adquirirá un certificado de seguridad.

Posteriormente, se realizaron una serie de pruebas y verificaciones que comprobaran que el funcionamiento del sistema fuera efectuado de forma correcta, demostrando que el sistema cumplía sus funciones.

A raíz de que fueron alcanzados los pasos anteriormente descritos, esto se puede concluir que los objetivos planteados al comienzo del trabajo fueron alcanzados de manera exitosa, logrando así un sistema de tipo bóveda que permita almacenar y recuperar tarjetas de crédito, mediante la tokenización, de manera segura y cumpliendo las normas PCI.

6.1 Limitaciones

Algunas de las limitaciones que se presentaron durante el desarrollo del proyecto fueron:

- Tener que permitir que la captura de los datos de la tarjeta la realice la aplicación cliente, ya que eso amplía los riesgos de seguridad (si el ambiente donde se capturan no es seguro) y reduce la cantidad de validaciones que pueden realizarse al momento de la captura.
- La falta del certificado de seguridad en la fase de desarrollo.

6.2 Trabajos Futuros

Para un futuro se proponen los siguientes trabajos relacionados a este tema:

- Adaptar el sistema de bóveda para tarjetas de débito.
- Adaptar el sistema de bóveda para cuentas bancarias.
- Desarrollar una billetera electrónica mediante la tokenización de tarjetas.
- Asociar un mismo sistema de bóveda a varios sistemas de pago.

Referencias

[1] W. Stalling, Cryptography and Network Security. Editorial Pearson. Sexta Edición. 2014.

[2] H. Nemati, L. Yang. Applied Cryptography for Ciber Security and Defense. 2011.

[3] D. Hankerson, A. Menezes, S. Vanstone. Guide to Elliptic Cryptography. 2008.

[4] PCI DSS – Documentation

[5] P2PE - Documentation

[6] R. Smith. Authentication: From Passwords to Public Keys. Addison-Wesley Professional. 1st edition. October 2001.

[7] Métodos de Autenticación. https://www.evidian.com/pdf/wp-strongauth-es.pdf

[8] R. Sonawane. Digital Certificates: An Introduction into Encryption and Authentication. 2004.

[9] J. Katz. Digital Signatures. 2010.

[10] Tokenization Guidelines - PCI DSS, 2011.

[11] Comité de Sistemas de Pago y Liquidación. Principios Básicos Para los Sistemas de Pago. 2001.

[12] G. Garibaldi, Comercio Electrónico: Conceptos básicos y reflexiones, 1998.

[13] Sistemas de Pago Electrónico http://pyme.net.uy/documentos/sistemas_pago.htm

[14] W. Stalling. The SET Standard. Noviembre, 2000.

[15] PCI DSS - P2PE Requirements.

[16] La Empresa. http://www.123Pago.net.ve/que-es-123Pago

[17] ISO 8583 https://www.iso.org/obp/ui/#!iso:std:31628:en

[18] F. Dosunmu, O. Dosunmu. XML: Extansible Markup Language. 2015.

[19] Formato ISO 8583. https://hithisissuresh.wordpress.com/iso-8583

[20] H. Schildt. Java: The Complete Reference. McGraw-Hill Education. 9th edition. Abril 2014.

[21] JDBC http://www.oracle.com/technetwork/java/javase/jdbc/index.html

[22] D. Coward. Java EE 7: The Big Picture. McGraw-Hill Education. 1st edition. Octubre 2014.

[23] M. Maclaughlin. MySQL Workbench: Data Modeling & Development. McGraw-Hill Education. 1st edition. 2013.

[24] Cipher. https://docs.oracle.com/javase/7/docs/api/javax/crypto/Cipher.html

[25] KeyGenerator https://docs.oracle.com/javase/7/docs/api/javax/crypto/KeyGenerator

[26] Eclipse. https://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2Fint_eclipse.htm

[27] L. Richardson, S. Ruby. Restful Web Services. Primera Edición. 2007.

[28] J. Goodwill, A. Vukotic, Apache Tomcat 7. 2011.

[29] Glassfish. https://glassfish.java.net/public/faq/GF_FAQ_2.html

[30] D. Gourly, B. Totty, HTTP: The definitive guide. Primera Edición. 2004.

[31] Jersey https://jersey.java.net

[32] JPOS http://www.jpos.org/about
[33] Visa Token Service https://developer.visa.com/products/vts
[34] VDEP http://www.visa.com.mx/asociandose-con-nosotros/tecnologia-de- pago/visa-token- service.html
[35] Tokenization http://www.mastercard.com/gateway/implementation_guides/Tokenization.html

Anexo A: Formato Personalizado ISO 8583

El Formato Personalizado de ISO 8583 que se utilizó en el desarrollo es el que se puede observar a continuación. Solo se utilizarán los tipos de mensaje 200 y 201, que son los mensajes de autorización y respuesta.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE isopackager SYSTEM "genericpackager.dtd">
<isopackager>
  <isofield
      id="0"
      length="4"
      name="MESSAGE TYPE INDICATOR"
      class="org.jpos.iso.IFA_NUMERIC"/>
  <isofield
      id="1"
      length="64"
      name="BIT MAP"
      class="org.jpos.iso.IFA_BITMAP"/>
 <isofield
      id="2"
      length="64"
      name="Numero de Tarjeta"
      class="org.jpos.iso.IFA_BITMAP"/>
 <isofield
      id="3"
      length="6"
      name="nai"
      class="org.jpos.iso.IFA_NUMERIC"/>
 <isofield
      id="4"
      length="12"
      name="Monto"
      class="org.jpos.iso.IFA_NUMERIC"/>
 <isofield
      id="11"
      length="6"
      name="idCobro"
      class="org.jpos.iso.IFA_NUMERIC"/>
 <isofield
      id="14"
      length="4"
      name="Mes de Expiracion”
      class="org.jpos.iso.IFA_NUMERIC"/>
  <isofield
      id="15"
      length="4"
      name="Ano de Expiracion"
      class="org.jpos.iso.IFA_NUMERIC"/>
  <isofield
      id="18"
      length="4"
      name="Numero de Comercio"
      class="org.jpos.iso.IFA_NUMERIC"/>
  <isofield
      id="27"
      length="1"
      name="opcion"
      class="org.jpos.iso.IFA_NUMERIC"/>
  <isofield
      id="28"
      length="8"
      name="id de Promocion"
```

```
    class="org.jpos.iso.IFA_NUMERIC"/>
<isofield
    id="32"
    length="11"
    name="nbproveedor”
    class="org.jpos.iso.IFA_NUMERIC"/>
<isofield
    id="37"
    length="12"
    name="respuesta"
    class="org.jpos.iso.IFA_LLCHAR"/>
<isofield
    id="50"
    length="3"
    name="cvv2”
    class="org.jpos.iso.IFA_LLCHAR"/>
<isofield
    id="53"
    length="18"
    name="cedula”
    class="org.jpos.iso.IFA_NUMERIC"/>
<isofield
    id="54"
    length="120"
    name="email”
    class="org.jpos.iso.IFA_LLCHAR"/>
<isofield
    id="66"
    length="1"
    name="onetimepwd”
    class="org.jpos.iso.IFA_NUMERIC"/>
<isofield
    id="94"
    length="7"
    name="concepto”
    class="org.jpos.iso.IFA_LLCHAR"/>
<isofield
    id="95"
    length="42"
    name="direccion"
    class="org.jpos.iso.IFA_LLCHAR"/>
<isofield
    id="98"
    length="24"
    name="ipcliente"
    class="org.jpos.iso.IFA_LLCHAR"/>
<isofield
    id="102"
    length="28"
    name="nombremdp"
    class="org.jpos.iso.IFA_LLCHAR"/>
<isofield
    id="103"
    length="28"
    name="titular "
    class="org.jpos.iso.IFA_LLCHAR"/>
<isofield
    id="104"
    length="64"
    name="token"
    class="org.jpos.iso.IFA_LLCHAR"/>
<isofield
    id="44"
    length="25"
    name="ADITIONAL RESPONSE DATA"
    class="org.jpos.iso.IFA_LLCHAR"/>
<isofield
```

```
        id="105"
        length="999"
        name="RESERVED ISO USE"
        class="org.jpos.iso.IFA_LLLCHAR"/>
</isopackager>
```

Anexo B: Invocación de Pago

A continuación, se muestra un ejemplo de la invocación que exige el sistema de pago:


```
{       "cedulaCliente":"24699188",
        "cedulaTH":"24699188",
        "nombreCliente":"Daniela",
        "nombreTH": "Daniela",
        "emailCliente":"prueba@gmail.com",
        "nai":"1af56",
        "idCobro":"1",
        "direccion":"bello monte",
        "nuTarjeta":"1234123456785678",
        "cvv2":"555",
        "mesVencimientoTarj":"05",
        "anioVencimientoTarj":"2018",
        "nbproveedor":"1",
        "concepto":"pago",
        "monto":"100",
        "idpromocion":"1",
        "ipCliente":"192.168.1.1",
        "nombreMdp":"tdc",
        "oneTimePwd":"123"
}
```