Universitat de les Illes Balears

Escola Politècnica Superior

Memòria del Treball de Fi de Grau

Creación de un entorno de pago seguro y sistema de almacenamiento de datos críticos para empresas que gestionan datos de tarjetas de pago

Endian Lin

Enginyeria Informàtica

Any acadèmic 2016-17

DNI de l'alumne: X5814135B

Treball tutelat per Isaac Lera Castro
Departament de Arquitectura y Tecnología de Computadores

S'autoritza la Universitat a incloure aquest treball en el Repositori Institucional per a la seva consulta en accés obert i difusió en línia, amb finalitats exclusivament acadèmiques i d'investigació	Autor		Tutor	
	Sí	No	Sí	No
	X		X	

Paraules clau del treball:
PCI DSS, entorno de pago, almacenamiento de datos.

ÍNDICE GENERAL

ACRÓNIMOS

PCI DSS Las Normas de Seguridad de Datos de la Industria de Tarjetas de Pago

CDE Entorno de Datos de las Tarjetas de Pago

CSS Cascading Style Sheets

IIS Administrador de Servicios de Información de Internet

XML Lenguaje de Marcado Extensible

AES Advanced Encryption Standard

SHA Secure Hash Algorithm

RESUMEN

Las normas de seguridad de datos de la industria de tarjetas de pago son una normativa que [1] proporcionan una referencia de requisitos técnicos y operativos desarrollados para proteger los datos de cuentas, que se aplican a todas las entidades que participan en el procesamiento de los datos de las tarjetas de pago, es decir, las que almacenan, procesan o transmiten datos de las tarjetas de pago y/o datos confidenciales de autenticación.

Si aplicamos esta normativa mejoramos la seguridad del servicio de pago electrónico, pero no es fácil de aplicarla. Según la normativa, todos los componentes del sistema incluidos en el entorno de datos de las tarjetas de pago o conectados a él (personas, procesos y tecnologías) han de cumplir sus requisitos[1]. Por lo tanto, el coste de aplicar la normativa en una red empresarial es muy elevado.

Juniper es una empresa que ofrece soportes tecnológicos y soluciones a las empresas turísticas. Ésta ha proporcionado una solución para resolver el coste de aplicación de la normativa en un entorno de pago para sus clientes.

Esta solución consiste en independizar el entorno de pago del resto de la red para reducir el número de componentes conectados a él, con el objetivo de reducir el coste de aplicación de la normativa, y después realizar dicha aplicación sobre este entorno independizado. Finalmente obtendríamos un entorno más pequeño, independiente y seguro para procesar, transmitir o almacenar los datos de las tarjetas de pago. Otra ventaja de separar el entorno de pago es que, una vez que el inspector de sistema de Juniper afirma que el entorno haya sido aprobado por la normativa, dicho entorno puede servir a todas aquellas empresas que procesan datos de las tarjetas de pago. Para eso simplemente autorizamos, identificamos y autenticamos la empresa que se conecta y leemos sus datos.

Nuestro objetivo es crear un entorno de pago que cumpla los requisitos de la normativa, capaz de procesar los datos de las tarjetas de pago. Para conseguir este objetivo hemos de realizar dos tareas con el siguiente orden:

1. Crear el entorno independizado que procesa datos de la tarjeta de pago.

2. Aplicar la normativa para aprobar el entorno de pago previamente creado.

En este trabajo solo vamos a realizar el primer proyecto, es decir, solamente vamos a crear el entorno que procesa datos de las tarjetas de pago sin aplicar la totalidad de la normativa (exceptuando algunos requisitos de desarrollo que se pueden ir desarrollando y aplicando conjuntamente durante el desarrollo del proyecto).

En definitiva, este proyecto se expone de manera didáctica los pasos a seguir para separar el entorno de pago de la red, construirlo en una red independiente y proponer al entorno un sistema que gestiona y procesa datos de la tarjeta de pago.

CAPÍTULO 1

INTRODUCCIÓN

Internet es un conjunto descentralizado de redes de comunicación interconectadas que permite difundir la información entre sus usuarios, es flexible, abierta, informal y sinérgica[2]. Éste introduce muchas nuevas formas de interactuar con el mundo, una de ellas es la compra y venta en línea. Actualmente, la compra y venta de productos y servicios en línea ocupan una importancia muy significativa en la vida cotidiana de los ciudadanos, ya que solamente al tener acceso al internet, uno puede realizar una compra de un producto o servicio en una tienda online desde cualquier lugar. El aumento de pago electrónico también ha supuesto un aumento de los delitos informáticos sobre los datos financieros. Así pues, la seguridad financiera es una condición crítica tanto para el comprador como para el vendedor. Cada vez que un cliente hace una compra en línea, éste puede ser recesivo a confiar que sus datos financieros sean protegidos y no accedidos por otros [2]. A su vez, el vendedor puede perder clientes por no cumplir los requisitos de la seguridad financiera.

Existe una normativa diseñada por el organismo *PCI Security Standards Council* [3] que se llama *Las normas de seguridad de datos de la industria de tarjetas de pago* (PCI DSS) [1], que sirve para proteger los datos de las tarjetas de pago con el fin de aumentar la seguridad del pago online.

En la fig.1.1 se representa la tabla con los 12 requisitos generales que proporciona la normativa para proteger los datos de cuentas y se clasifican en 6 agrupaciones:

1. La primera agrupación contiene dos requisitos que están orientados para obtener redes y sistemas seguros, para eso requiere configuraciones correctas del firewall y de los valores de los parámetros del sistema.

2. La segunda agrupación está creada para proteger los datos del titular de la tarjeta que contiene dos requisitos de desarrollo. Estos dos requisitos sirven para proteger los datos del titular que están en transmisión o almacenados.

3. La tercera agrupación sirve para mantener todos los sistemas y aplicaciones en estado seguro, con el fin de evitar las vulnerabilidades de seguridad. Para eso hay

que tener el antivirus actualizado y utilizar aplicaciones seguras.

4. La cuarta agrupación controla los accesos a los datos de las tarjetas. Hay que Evitar los accesos no autorizados y establecer sistema de identificación y autenticación.

5. La quinta agrupación supervisa y evalúa las redes periódicamente para asegurar que sus componentes y recursos están funcionando de forma segura y correcta. También hay que probar periódicamente los sistemas y procesos de seguridad.

6. La última agrupación trata de mantener una política de seguridad de información para todo el personal.

Cada uno de estos 12 requisitos se detalla en muchos subrequisitos y que los podemos encontrar en el documento [1].

Desarrolle y mantenga redes y sistemas seguros.	**1.**	Instale y mantenga una configuración de firewall para proteger los datos del titular de la tarjeta.
	2.	No usar los valores predeterminados suministrados por el proveedor para las contraseñas del sistema y otros parámetros de seguridad.
Proteger los datos del titular de la tarjeta	**3.**	Proteja los datos del titular de la tarjeta que fueron almacenados
	4.	Cifrar la transmisión de los datos del titular de la tarjeta en las redes públicas abiertas.
Mantener un programa de administración de vulnerabilidad	**5.**	Proteger todos los sistemas contra malware y actualizar los programas o software antivirus regularmente.
	6.	Desarrollar y mantener sistemas y aplicaciones seguros
Implementar medidas sólidas de control de acceso	**7.**	Restringir el acceso a los datos del titular de la tarjeta según la necesidad de saber que tenga la empresa.
	8.	Identificar y autenticar el acceso a los componentes del sistema.
	9.	Restringir el acceso físico a los datos del titular de la tarjeta.
Supervisar y evaluar las redes con regularidad	**10.**	Rastree y supervise todos los accesos a los recursos de red y a los datos del titular de la tarjeta
	11.	Probar periódicamente los sistemas y procesos de seguridad.
Mantener una política de seguridad de información	**12.**	Mantener una política que aborde la seguridad de la información para todo el personal

Figura 1.1: descripción general de los 12 requisitos de PCI DSS

Sin embargo, el nivel de dificultad de aplicar esta normativa al *entorno de datos de las tarjetas de pago*(CDE) que está instalado en la red empresarial puede ser elevado, porque según dicha normativa (pág. 10 del documento), el alcance de sus requisitos engloba todos los componentes de la red (personas, procesos y tecnologías). Entonces, la motivación de realizar este proyecto es diseñar una solución que minimiza el costo para las empresas que quieran aplicar la normativa PCI DSS.

Juniper [4], donde realicé la práctica de carrera, es una empresa que ofrece soportes tecnológicos y soluciones para las empresas turísticas. Ésta ha proporcionado a sus clientes una solución que resuelve el problema anteriormente citado. La solución se basa en el concepto de *segmentación de la red*, es una metodología que consiste en separar el CDE del resto de la red de la empresa y crear una nueva red aislada e independiente, donde volvemos a instalar el CDE. Al aplicar esta solución el CDE deja de ser una parte de la red de la empresa y se convierte en un entorno independiente con su propio sistema de pago online y con menos componentes. Con lo cual, implica

que este nuevo sistema puede ser integrado y utilizado por más de una empresa, así pues, resolvería el problema de aplicar la normativa a múltiples empresas.

En base a lo anterior, nuestro objetivo principal del proyecto final es crear y ofrecer un entorno seguro de pago que cumple con la normativa PCI DSS para que los proveedores puedan integrarse a este sistema y deja de utilizar su propio entorno de pago. Sin embargo, para que cumpla con la normativa hemos de aplicar todos sus requisitos en todos los componentes. Aunque el tamaño de la red haya sido reducido, esto puede generar un exceso de carga de trabajo. Teniendo en cuenta lo anterior, dividimos el proyecto en dos subproyectos:

1. Crear el entorno de pago que procesa los datos de la tarjeta de pago.

2. Aplicar la normativa para aprobar el entorno de pago previamente creado.

El primer proyecto consiste en crear el entorno sin aplicar todos los requisitos de la normativa, excepto los requisitos número 3, 4 y 8 (fig.1.1). Es decir, en este proyecto no solo creamos el entorno sino también aplicamos los requisitos 3, 4 y 8 porque son los requisitos de desarrollo que afectan directamente al funcionamiento de los componentes del proyecto. Por lo tanto, los podemos desarrollar y aplicar durante el proceso de desarrollo del primer proyecto. Así pues, nuestro objetivo sobre este proyecto es crear un entorno de pago independiente que gestiona datos de la tarjetas de pago, y que permite reducir el costo del segundo proyecto. Mientras el segundo proyecto se dedica a aplicar todos los requisitos restantes de la normativa sobre el entorno creado por el proyecto anterior.

Sin embargo, no vamos a realizar el segundo proyecto y lo dejamos en manos de otros compañeros de Juniper por dos motivos:

1. La limitación del tiempo. Porque el primer proyecto requiere realizar muchas tareas y que las especificamos más adelante.

2. Existe un departamento de seguridad de sistemas que se encarga de realizar el segundo proyecto.

El prototipo del proyecto se instala en una infraestructura simple mediante una máquina virtual, y como lenguaje de programación escogemos el que se llama *Visual Basic*. El proyecto se ha planteado las siguientes tareas secuenciales:

1. Diseño del entorno de pago.

    a) Consultar la normativa recopilar informaciones relativas.

    b) Crear un diagrama de flujo del funcionamiento del entorno.

    c) Identificar y analizar los componentes del entorno.

2. Desconexión del entorno de pago de una empresa de prueba.

3. Creación de una red pública en un servidor virtualizado.

4. Creación de una página que gestiona y procesa datos de la tarjeta de pago, además, es de estilo dinámico y multi-idioma.

5. Creación de base de datos para almacenar los datos de la tarjeta.
6. Tokenización de número de la tarjeta para proteger los datos almacenados.
7. Creación de un método de encriptación para cifrar los datos.
8. Creación de un sistema de identificación y autenticación para controlar el acceso al entorno.
9. Creación de un sistema de gestión de datos.
10. Creación de dos módulos para transmisión y recepción de datos.
11. Establecimiento de la comunicación entre las dos redes.
12. Otras tareas menos relevantes como traducciones de idiomas, control de errores, pruebas de funciones del sistema, etc.

El cronograma del proyecto se divide en 4 fases. La primera fase es la consulta de la normativa, que servirá para conocer los requisitos que se aplicarán en nuestro proyecto. En base a estos requisitos, diseñamos el entorno de pago y sus componentes. Una vez acabado el diseño entramos a la fase de desarrollo. Finalmente, testeamos el funcionamiento del entorno de pago online y corregimos los errores encontrados.

Una vez acabado el proyecto, el único componente visible para el cliente es nuestra página de pago, a través de la cual, el cliente podrá realizar el pago o la cancelación del producto. A su vez, el proveedor que se integra a nuestro sistema de pago podrá realizar el cobro más seguro y almacenar los datos de pago de los clientes en un lugar más seguro.

En este documento, hablaremos en el capítulo 2 el diseño de la solución, donde aparecerán todos los componentes de nuestro sistema de pago. En el capítulo 3 describiremos los pasos para segmentar el entorno de pago del proveedor y construirlo en una nueva red. Dedicaremos en el capítulo 4 a montar un sistema de gestión de los datos de las tarjetas de pago en el nuevo entorno de pago, que tendrá la capacidad de realizar el cobro y el almacenamiento estos datos. En el capítulo 5 mostraremos algunos ejemplos de la implementación del proyecto y los resultados de los componentes del proyecto desarrollado. Y finalmente en el capítulo 6 sacaremos las conclusiones del proyecto.

CAPÍTULO 2

DISEÑO

El problema que dificulta a la hora de aplicar la normativa en una red de una empresa es el hecho de que todos los componentes de la misma que se conectan al CDE han de cumplir los requisitos de la normativa. Es decir, el alcance de los requisitos de la normativa es muy extenso. Para resolver este problema hemos decidido segmentar la red, que consiste en separar el CDE del resto de la red y convertirlo en una otra red independiente.

Una vez quedan determinados el problema y la solución, entramos a la fase de diseño. En esta fase descubriremos todos los componentes necesarios para la creación de la solución. Analizando los componentes para concordarlos con los requisitos de la normativa y evitar problemas de desarrollo.

2.1 Arquitectura del entorno de pago

En esta sección vamos a diseñar el aspecto general de nuestro entorno de pago. Para poder diseñarlo hemos de crear un diagrama de flujo sobre el cual, deducimos todas las funcionalidades del nuevo entorno y sus componentes.

El diagrama de flujo 2.1 explica básicamente lo siguiente: un cliente entra a la página de pago (entorno PCI DSS) a través de una página de una empresa acreditada (entorno no PCI DSS), con la intención de realizar un pago. Tiene posibilidad de auto relleno de datos de la tarjeta si es un cliente registrado. Si éste desea efectuar el pago, enviamos la petición de pago al banco online con un límite de intentos. Finalmente enviamos el mensaje de pago exitoso a la empresa para que asigne el producto al cliente, o dejamos al cliente intentar otra vez en caso de pago fracasado. Si se produce un rechazo de pago por parte del cliente o un exceso de intentos, se redirige a la página de la empresa proveedora.

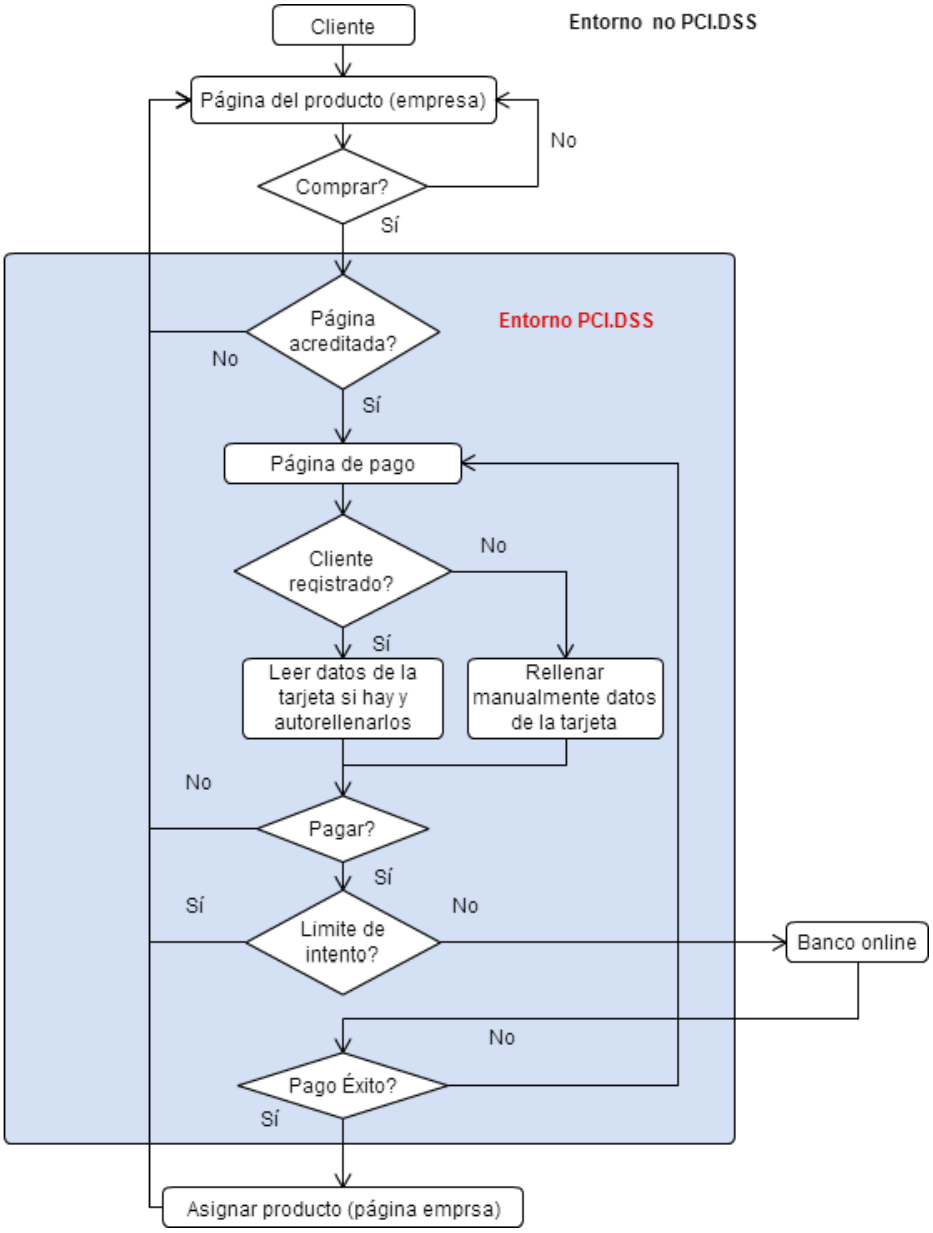

Figura 2.1: Diagrama de flujo del proyecto

Gracias al diagrama de flujo, descubrimos los siguientes puntos claves para diseñar la arquitectura de nuestro entorno:

- La empresa necesita dos módulos para establecer la comunicación con nuestro entorno de pago: uno para enviar la petición de pago, y otro para recibir la respuesta de pago.
- Existe un banco online que interviene para procesar el pago.
- La página de pago es la interfaz de la comunicación entre los dos entornos.

- Cuando nuestro entorno de pago recibe datos desde el entorno no PCI DSS es necesario hacer una acreditación de datos.
- Es necesario tener una base de datos para almacenar los datos de las tarjetas de pago.

Finalmente, toda esta información se puede representar en la fig.2.2.

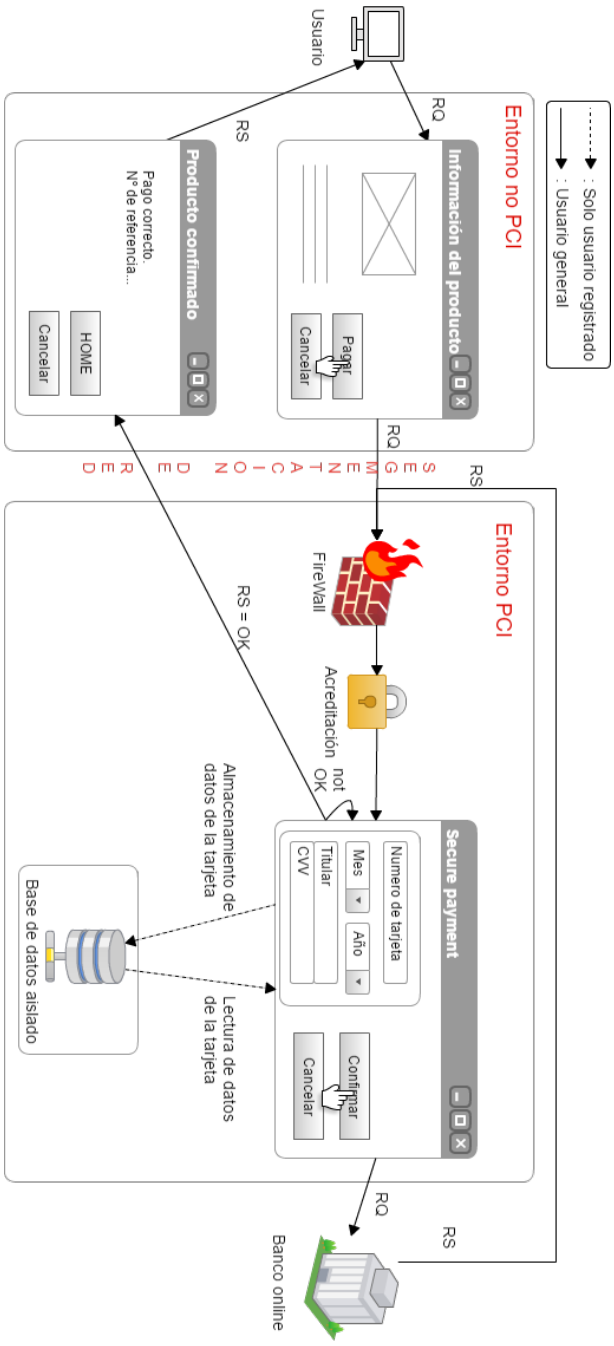

Figura 2.2: Arquitectura del proyecto

2.2 Componentes del entorno PCI DSS

La arquitectura del entorno de pago que hemos diseñado está formada por los componentes de dos entornos, y para distinguirlos, separamos los componentes en dos clases: los del entorno PCI DSS y los del no PCI DSS. En esta sección vamos a diseñar los componentes PCI DSS, cada uno de ellos tiene que cumplir los requisitos de la normativa.

2.2.1 Las peticiones

Durante el proceso de gestión de datos, la página de pago puede enviar o recibir varias peticiones y son las siguientes:

1. Petición de intención de pago. Es la petición que envía la empresa proveedora cuando su cliente haya seleccionado un producto y desea realizar el pago.

2. Petición de confirmación de pago. Esta petición se genera cuando todos los datos necesarios de pago hayan sido rellenados y el cliente decide realizar el pago ya.

3. Petición de escritura y lectura de datos de la tarjeta de pago. La petición de escritura se genera cuando un cliente registrado ha realizado el pago, pero aún no tiene datos de la tarjeta almacenados en la base de datos. En este caso, realizamos la escritura para almacenarlos. La de lectura se genera cuando un usuario registrado entra a la página de pago con los datos de la tarjeta almacenados en la base de datos, de esta manera, autorellenamos sus datos de la tarjeta de pago.

4. Petición de eliminación de datos de las tarjetas. Cuando los datos de las tarjetas almacenados en la base de datos han expirado el tiempo de almacenamiento establecido por la normativa, hemos de generar esta petición para eliminarlos.

5. Petición de cancelación del producto. Se genera cuando el cliente deja de comprar el producto.

Intención de pago

La petición que establece la comunicación entre la empresa proveedora y el entorno PCI DSS. Esta petición ha de poder identificar la página de la empresa proveedora, el idioma que utiliza dicha página, el cliente y el producto seleccionado.

Esta petición ha de pasar por un proceso de acreditación, ya que necesitamos saber si la petición ha sido realmente transmitida por la empresa o es un fraude. Por lo tanto, además de los datos previamente dicho, esta petición ha de contener una información que pueda justificar y autorizar el acceso.

Confirmación de pago

La petición que establece la comunicación entre el entorno PCI DSS y el banco. En este proyecto no realizamos el pago sino, una simulación de pago. A pesar de eso, hemos de ser conscientes de que, si lo hiciéramos, según la normativa (requisito 4), hemos de cifrar el número de la tarjeta en el momento de la transmisión. Además, hemos de reconocer los datos necesarios de acuerdo con la configuración de peticiones de

cada banco, ya que los bancos también necesitan realizar una acreditación de nuestros datos.

Escritura y lectura de datos de la tarjeta de pago

Estas peticiones las ejecutamos siempre y cuando el cliente fuese un cliente registrado. Leemos los datos de su tarjeta previamente almacenados y auto rellenamos los campos una vez que el cliente haya sido identificado y accedido a la página de pago. Y lanzamos la petición de escritura cuando el cliente registrado no tiene guardado los datos de la tarjeta después de la compra.

Estas peticiones ejecutan las instrucciones de base de datos para una escritura o lectura de datos. Según la normativa (requisito 3), solo se puede almacenar los siguientes datos de la tarjeta: el nombre del titular, fecha de vencimiento, el tipo de la tarjeta y el número de la tarjeta cifrado con una criptografía sólida. No se puede almacenar el código de verificación de tarjeta (CVV).

Eliminación de datos de la tarjeta

Según la normativa (requisito 3), hemos de tener un control trimestral para identificar y eliminar, de manera segura, los datos de las tarjetas de pago expirados. Entonces, cuando almacenemos los datos de la tarjeta también será necesario almacenar la fecha de creación.

Cancelación del producto

La petición lanzada por el cliente cuando éste no desea continuar con el proceso de compra. En este caso simplemente le redirigimos a la página de la empresa. Y finalmente, enviamos un mensaje a la empresa diciéndole que, este producto ya puede volver a estar en estado disponible para otros clientes.

2.2.2 FireWall

El FireWall realiza la función de filtrar o bloquear las peticiones no permitidas para evitar accesos no autorizados. Según la normativa (requisito 1), donde dice que tenemos que instalar y configurar el Firewall para la protección de datos de las tarjetas de pago. Sin embargo, en este proyecto no haremos la configuración del FireWall por dos motivos: 1) Es un proyecto prototipo y la red donde se instala es cambiante. 2) Hay un departamento que se encarga de la configuración del mismo.

2.2.3 Acreditación de petición

La acreditación desempeña una función similar al Firewall pero en otro nivel de seguridad. Hay que tener en cuenta que aunque la petición fuese transmitida a partir de una red autorizada tampoco podemos fiar en ella, ya que la red puede sufrir un ataque y se convierte en una herramienta del atacante. Entonces, tenemos que verificar de alguna manera que la petición que entra desde la otra red sea correcta, confiable, completa y autorizada.

2.2.4 Página de pago

La interfaz de la comunicación entre el entorno no PCI DSS y el entorno PCI DSS. Además, actúa como un gestor que gestiona los datos de las tarjetas según las peticiones y las respuestas que recibe. Siendo ésta una página web, aparte de tener funcionalidades que satisfacen las peticiones 2.2.1 también tiene que tener las siguientes características:

- Control de errores. Evita el abuso de intentos de pago, expulsa las peticiones fraudulentas y también asegura que los datos introducidos de la tarjeta sean correctos.

- Cascading Style Sheets (CSS)[5] dinámico. CSS es un lenguaje usado para definir la presentación de documentos estructurados y es muy utilizado en documentos web. En nuestro caso, la página de pago ha de utilizar CSS que define la página de la empresa para adaptar a su estilo de presentación. Dado que nuestro proyecto está diseñado para múltiples empresas y cada una de ellas tiene su propio estilo de página. Por lo tanto, el CSS que utiliza la página de pago ha de ser dinámico. En la petición de *Intención de pago* contiene la información de la empresa, y a través de la cual podemos identificar la empresa y escoger su CSS.

- Multi-idioma. Igual que el estilo de página, la página de pago ha de adaptar el idioma al de la página de la empresa. Llegamos a conocer el idioma que utiliza esta última a través de la petición que nos envía, y mostramos nuestra página en ese idioma.

2.2.5 Base de datos aislada

Una base de datos aislada de internet y protegida con un FireWall para almacenar los datos de las tarjetas de pago, con acceso muy restringido (requisito 8). En este proyecto no vamos a crear la base de datos en un lugar aislada de internet, sino que la creamos en nuestra red, ya que es un proyecto prototipo.

2.2.6 Datos de las tarjetas de pago

Estos datos son muy sensibles y confidenciales que requieren una protección segura a la hora de transmitir, procesar o almacenar estos datos. Entonces, en el momento de realizar cualquiera de las tres acciones hemos de cifrarlos según la normativa (requisito 3).

2.3 Componentes del entorno no PCI DSS

Los componentes del entorno no PCI DSS son normalmente los que no podemos modificar, como por ejemplo el FireWall del banco, la página del banco, etc. No obstante, tenemos que modificar la red de la empresa por dos motivos: 1) Tenemos que separar el CDE del resto de la red. 2) Queremos que las dos redes se establezcan una comunicación entre sí.

2.3.1 Página de compra de la empresa

En esta página, normalmente existe un formulario que pide al cliente los datos de la tarjeta de pago. Si queremos separar el entorno de pago de esta red, hemos de eliminar este formulario para que no aparezcan estos datos en la red de la empresa.

2.3.2 Módulo de envío de petición

Una vez separado el CDE de la red de la empresa, el lugar donde estaba instalado el formulario que recopilaba datos de la tarjeta de pago, será substituido por el módulo de envío de petición. Este módulo recopila y genera toda la información necesaria para construir la petición de *intención de pago* (Id de la empresa, Id del cliente, Id del producto, etc.). Finalmente envía la petición a nuestra red.

2.3.3 Módulo de recepción de estado de pago

El módulo que instalamos en la red de la empresa y realiza la última tarea de nuestro sistema de pago online: recibir nuestra respuesta del estado de pago y comunicar a la empresa si es un pago con éxito, o que el producto ha sido cancelado por el cliente. Entonces la empresa depende de la respuesta que le transmitamos, hará asignación o liberación del producto.

CAPÍTULO 3

PREPARACIÓN DEL ENTORNO

Este capítulo detalla la fase de desarrollo de nuestro entorno sin sistema de gestion de pago. Es decir, utilizamos todo lo necesario que hemos diseñado en el capítulo anterior para crear el cuerpo del entorno, con todos sus componentes necesarios pero sin funcionamiento. Para que éste funcionase es necesario un sistema de gestión de datos, que lo veremos con más detalle en el capítulo siguiente.

3.1 Desconexión de CDE

Una empresa que vende productos o servicios en línea, suele tener un formulario que recopila datos de las tarjetas de pago instalado en la página de pago. En nuestra propuesta, el servicio de pago de dicha empresa es eliminado para que en esta red no aparezcan los datos de las tarjetas de pago y como consecuencia, el CDE se quedaría desconectado. De esta manera, la empresa deja de tener su sistema de gestión de datos de pago y procederá a utilizar el entorno que vamos a construir. Para desconectar el CDE hemos de seguir los siguientes pasos:

1. Obtener permiso para acceder al código fuente de la página de pago de la empresa.

2. Encontrar el formulario que recopila datos de las tarjetas de pago y eliminarlo.

3.2 Creación de red

Una vez tengamos el CDE desconectado de la red de empresa,pasamos al siguiente paso que es crear una red pública e independiente, sobre la cual instalamos los componentes del entorno de pago.

Para simular una red independiente, vamos a construir un servidor en una máquina virtual. Utilizamos VMware para instalar el sistema operativo *Windows7 Ultimate*, y

activamos el Administrador de Servicios de Información de Internet (IIS) para que este equipo funcionase como un servidor web.

Finalmente este equipo virtualizado y la red interna formada por el mismo dan forma al servidor de nuestro entorno prototipo.

3.3 Construcción de componentes

En esta sección vamos a crear los dos componentes más relevantes de nuestro proyecto, que son la página web que gestiona los datos de la tarjeta de pago, y una base de datos para almacenar estos datos.

3.3.1 Página de pago

Esta página de pago gestiona los datos de la tarjeta de pago y está publicada en el servidor web. Sin embargo, en esta subsección solo creamos el estilo de la página y no entrar en detalle en su sistema de gestión de datos.

En el capítulo de diseño hemos diseñado la página de pago con la capacidad de adaptarse a diferentes estilos e idiomas de distintas empresas, para que ésta pueda ser utilizada por más de una empresa. A continuación veremos los procesos de creación de una página con estilo dinámico y multi-idioma.

Estilo CSS dinámico

El diseño de la página es igual para todas las empresas integradas que contiene los siguientes componentes: una cabecera para colocar logos de distintas empresas; un cuerpo para describir la información del producto y colocar el formulario para rellenar datos de la tarjeta de pago; un pié de página para otras informaciones. Nuestro diseño de la página es sencillo, porque diseñar página no es nuestro objetivo sino una mejora del proyecto. Por lo tanto, para una página de pago, con este diseño ya es suficiente .

Para conseguir esto, manipulamos el fichero CSS de la página y reservamos estas 3 secciones con la instrucción *div*. Podemos tomar la figura 3.1 como referencia para ilustrar el diseño.

Por otro lado, necesitamos que la página adapte al estilo de distintas empresas. Por ejemplo, colocar su logo en la sección cabecera, cambiar el color de los elementos (Letras, botones, etc.) al color principal de la empresa, etc. Podemos observar la figura 3.2 para contrastar con la figura 3.1. para conseguir lo anteriormente dicho hemos de identificar la empresa que desea conectarse y después encontrar su fichero CSS para poder pintar la página. Los pasos concretos son:

1. Leemos el id de la empresa que nos envía por la petición de *intención de pago* antes de pintar la página para identificarla.
2. Una vez identificada hay dos opciones a seguir:
   a) Descargar el logo y el fichero CSS desde su página si la dirección de descarga es fija.
   b) Si la dirección de descarga del logo y el fichero CSS no es fija, utilizar el logo y el fichero CSS local.

3. Pintar la página con el logo y el fichero CSS obtenidos.

En este proyecto se ha escogido la opción *a* para la descarga del logo y la opción *b* para conseguir el CSS, porque la dirección que permite la descarga del fichero CSS no es fija. La ventaja de la segunda opción, la b, es que el contenido está preparado en nuestro local para nuestra página, siendo el proceso de servicio más ligero. En cambio, el inconveniente es la poca flexibilidad, ya que, si la página de la empresa cambia de estilo, tenemos que cambiar también nuestro CSS.

Multi-idioma

Otro requisito para que nuestra página se convierte en una página universal es el multi-idioma. Hay que tener todos los idiomas que proporciona cada proveedor, para asegurar la correcta lectura de los clientes de cualquier proveedor integrado. Las figuras 3.1 y 3.2 representan dos páginas de dos proveedores en dos idiomas, uno en castellano y otro en chino.

Para conseguir lo anteriormente dicho, la página ha de estar construida por palabras y frases cortas, cada una de ellas se representa como una variable y tiene traducciones en distintos idiomas. De esta manera, cuando queremos que la página pinte en un idioma, simplemente leer la traducción correspondiente. Para que la página tuviese esta función seguiríamos los siguientes pasos:

1. Preparar un fichero de Lenguaje de Marcado Extensible (XML) [6] con todas las palabras y frases traducidas en múltiples idiomas.

2. Leer el id del idioma que nos envía a través de la petición de intención de pago antes de pintar la página.

3. Leer las traducciones del fichero previamente preparado y pintarlas a la página.

Con esto acabamos de crear el aspecto de la página y ya la podemos instalar en el servidor.

3.3.2 Base de datos

El otro componente más importante de nuestro proyecto es la Base de datos. Según la normativa, la tenemos que tener en un servidor aislado y protegido, para almacenar los datos de la tarjeta. Sin embargo, en este proyecto prototipo la instalamos en nuestra red.

Para crear una base de datos utilizamos un sistema de gestión de base de datos llamado *MySql*. Lo descargamos desde su página web y luego instalarlo siguiendo las instrucciones de su instalación. Configuramos las credenciales de acceso (requisito 8) y ya tenemos una base de datos instalada en nuestro servidor.

3.4 Establecimiento de comunicación

Hasta aquí hemos creado una red independiente, sobre la cual hemos instalado un entorno prototipo. En esta sección vamos a crear un puente de comunicación entre la empresa y nuestro entorno de pago, ya que necesitamos la información de la empresa

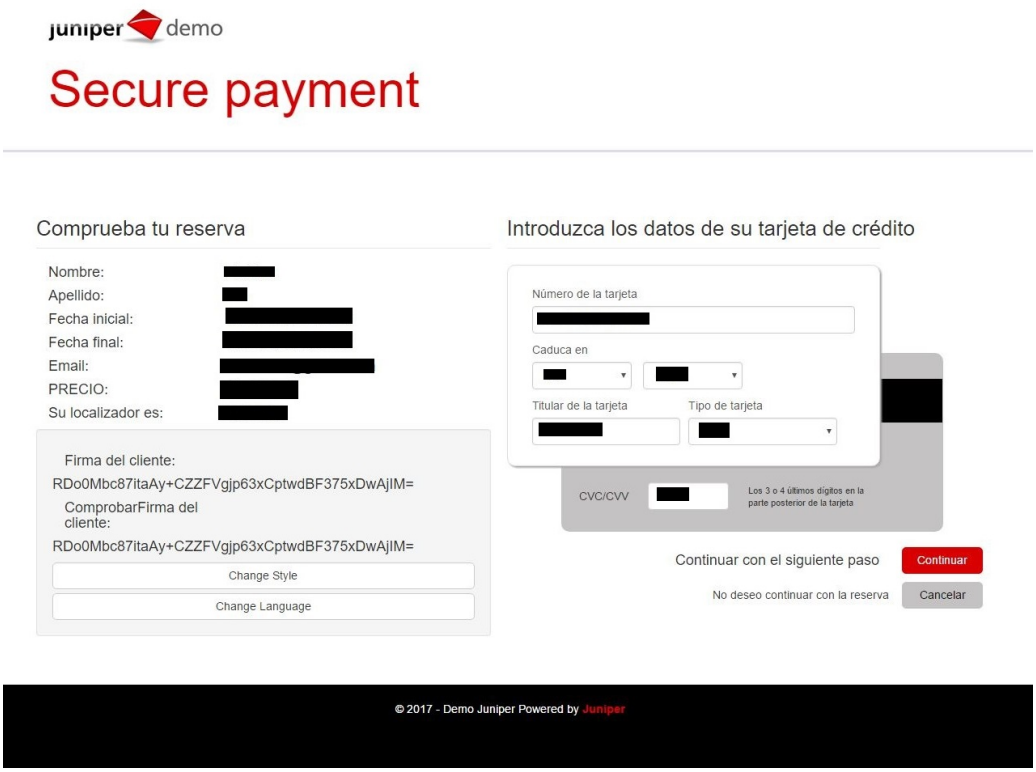

Figura 3.1: Página de pago para empresa Juniper

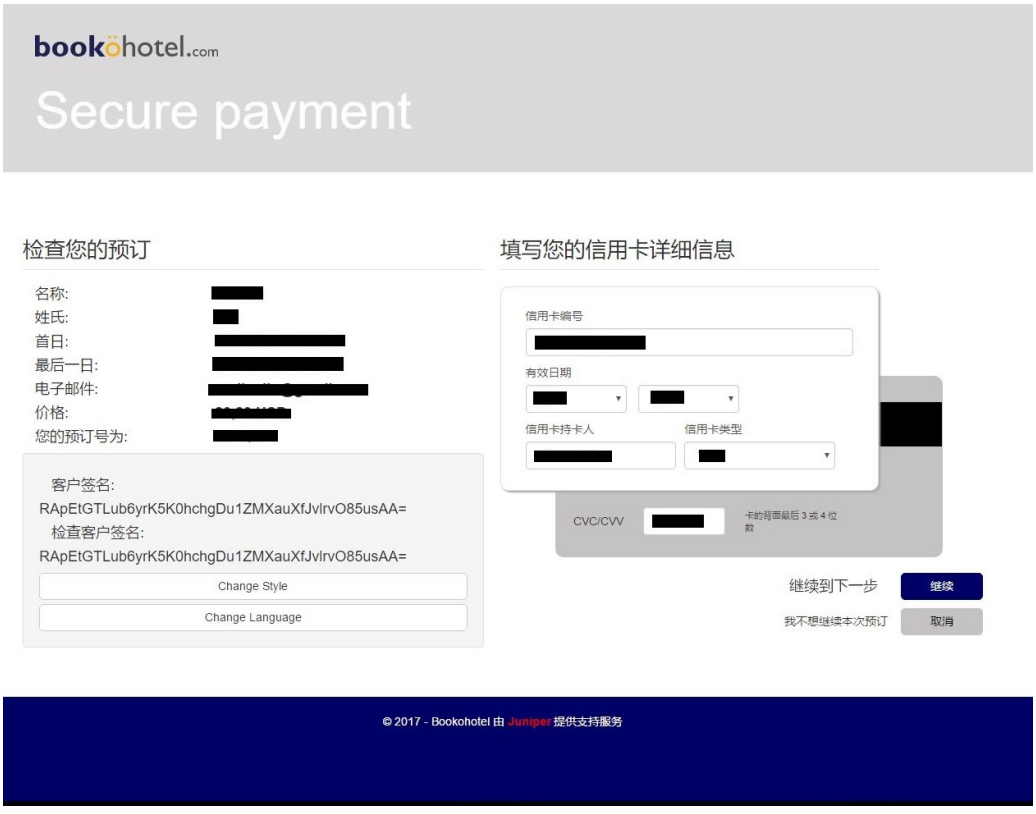

Figura 3.2: Página de pago para empresa BookOhotel

para llevar a cabo el pago, y tenemos que enviarle la información sobre el estado del pago.

3.4.1 Configuración de la página origen

Para que la empresa pueda comunicar con nosotros tenemos que crearle dos módulos: uno es el emisor y otro el receptor. El módulo emisor realizará función de envío de los datos necesarios con nuestros requisitos de seguridad y el receptor, hará función de recibir nuestras respuestas sobre el estado de pago. Seguimos los siguientes pasos para conseguir este objetivo:

1. Implementar una función que genera una firma digital 3.4.2 para la acreditación de acceso.
2. Crear un módulo que envía petición de intención de pago y que realiza las siguientes tareas:
   2.1. Recopila información de identificación (identificadores de la empresa, del cliente, del producto y del idioma).
   2.2. Genera una firma digital con una clave privada para la empresa y nosotros.
   2.3. Envía la firma digital junto con la información previamente recopilada al nuestro entorno de pago.
3. Crear un módulo receptor que hace las siguientes tareas:
   3.1. Recibe e identifica el mensaje enviado por la página de pago, para saber si es un pago con éxito o una cancelación del producto.
   3.2. Cancelar el producto en caso de que el cliente decida no comprarlo o asignar el producto si se ha efectuado el pago.

3.4.2 Acreditación de petición

Cuando detectamos un acceso a la página de pago, lo primero que hacemos es verificar la autorización de la petición y la integridad de los datos. Esto se consigue mediante la Firma Digital [7], evitando problemas como accesos no autorizados y datos incompletos.

La firma digital

La firma digital es un signo único para la identificación y autenticación de una persona, una entidad o un fichero, que puede ser representada en forma de dibujos, imagenes o conjunto caracteres. En nuestro caso la necesitamos para asegurar la autorización de acceso de los cliente de las empresas proveedoras a nuestra página y la integridad de los datos que nos transmiten. Para ello vamos a generar una firma digital para ambos lados (una para el proveedor y otra para nosotros) con el mismo parámetros de entrada y una clave privada. La firma ha de ser única, impredecible e irreversible. De esta manera, cuando nos llega la petición comparamos las dos firmas. Si estas son idénticas confirmamos la autorización de la petición y la integridad de los datos. En caso contrario, rechazamos la petición.

Para una mayor confidencialidad, antes de generar la firma hemos de cifrar los datos de la petición utilizando una criptografía sólida, con una clave privada que solo la conocen entre las dos redes. En este caso utilizamos el algoritmo Advanced Encryption Standard (AES) de 128 bits para la encriptación de datos.

Una vez que los datos queden encriptados por el algoritmo AES, utilizamos este resultado como parámetro de entrada para la generación de la firma. Para esto utilizamos el algoritmo Secure Hash Algorithm (SHA) de 256 bits. Ambos algoritmos son mencionados y aprobados por el articulo [7].

Pasos de generación de la firma digital:

1. Recopilar los datos que serán enviados.
2. Preparar una clave de 128 bits para el algoritmo AES de 128 bits.
3. Encriptar los datos con la clave utilizando el algoritmo AES.
4. El resultado cifrado se pasa por el algoritmo SHA de 256 bits, y así obtener el resultado en cadena de caracteres de una longitud de 256 bits.

CAPÍTULO 4

SISTEMA DE GESTIÓN DE DATOS

En el capítulo anterior hemos creado el entorno prototipo en una red independiente, pero aún no tiene la capacidad de gestionar los datos de las tarjetas de pago. En este capítulo vamos a proporcionar a nuestra página de pago un sistemas de gestión de estos datos, para que dicha página pueda realizar de manera segura, el pago, el almacenamiento de datos y otras operaciones.

4.1 Acreditación e identificación de datos

La acreditación de petición es el primer paso para permitir el acceso de un cliente que entra al nuestro entorno con la intención de pago, tal como hemos comentado en el capítulo anterior, esto se consigue mediante una firma digital. Por lo tanto, antes de mandar el cliente al nuestro entorno la empresa proveedora tiene que generar una firma digital, siguiendo los pasos que hemos comentado en el capítulo anterior.

Cuando recibimos los datos y la firma digital que nos envía la empresa proveedora, tenemos que identificar todos estos datos y crear otra firma digital con dichos datos, para verificar la integridad de estos datos y la autorización del acceso. Los pasos concretos se detallan a continuación:

1. Identificar los datos que nos llegan:

    - El id del producto. Nos sirve para indicar a la empresa sobre el estado de pago del producto.
    - El id de la empresa. Nos sirve para poder pintar la página de pago a su estilo y reconocer la pasarela de pago que utiliza.
    - El id del cliente para identificar si es un cliente registrado o no, en caso afirmativo, éste disfrutará del servicio de autorrelleno de datos de la tarjeta.
    - El idioma que utiliza el cliente para que la página de pago muestre su contenido en ese idioma.

- La firma digital generada por la empresa, nos servirá para comparar con la que vamos a generar.

2. Utilizar estos datos para generar una nueva firma digital, con la misma clave privada que ha utilizado la empresa proveedora.

3. Comparar las dos firmas, en caso de que sean iguales, permitimos su acceso, y en caso contrario, rechazamos su entrada.

Con todo esto acabamos de acreditar e identificar los datos, finalmente guardamos estos datos en las variables de sesión para no perderlos durante el proceso de pago.

4.2 Acciones de pago

El cliente que haya entrado a nuestra página de pago ha de poder realizar el pago del producto o la cancelación del mismo.

4.2.1 Confirmación

La primera acción es realizar el pago, pero en este proyecto solo hacemos una simulación de pago. Cuando el cliente elige la opción de pagar, lo que hacemos es generar una señal tipo ”ok"para indicar al sistema de pago que el abono ha sido efectuado correctamente, y después, la señal será enviada a la página del proveedor y será recibido por el módulo receptor que habíamos creado. Al recibir el mensaje, la empresa asigna el producto a su cliente. No obstante, hay que tener en cuenta los siguientes puntos durante el proceso de pago:

1. Todos los datos de la tarjeta han sido introducidos en formato correcto.

2. La fecha de caducidad de la tarjeta no puede estar expirado.

3. Cifrar el número de la tarjeta antes de enviarlo a la pasarela de pago (requisito 4). No podemos realizar esta tarea por dos motivos: 1) Es una simulación de pago y no enviamos datos de la tarjeta a ningún sitio. 2) Cada banco puede tener su requisito de encriptación. Aún así dejaremos un método de encriptación preparado que lo veremos en la siguiente sección.

4.2.2 Cancelación

Cuando el cliente decide cancelar el producto, enviamos esta información al módulo receptor que está instalado en la red de la empresa proveedora. Al recibir este mensaje, la empresa libera el producto y este último estará disponible para otros clientes.

4.3 Almacenamiento de datos

Aparte de poder realizar un pago, la página también dispone de la posibilidad de almacenar los datos de la tarjeta de pago del cliente registrado, para facilitar la próxima compra, siempre y cuando el cliente nos lo autoriza.

4.3.1 Configuración de Base de datos

Una correcta configuración de base de datos nos garantiza el funcionamiento, el rendimiento y la seguridad a la hora de procesar los datos. Conectamos a la base de datos y creamos una tabla para almacenar y extraer los datos de la tarjeta con los siguientes atributos:

- Token, es una cadena de caracteres única que sirve de identificador de los datos de la tarjeta.
- CardNum, el número de la tarjeta de pago, está cifrado con una clave privada sólida.
- CardHolder, el nombre del titular de la tarjeta.
- DateExpiration, nos permite conocer la fecha de vencimiento de la tarjeta.
- Tipo, para saber el tipo de la tarjeta.
- DateCreation, la fecha de almacenamiento de estos datos en la base de datos, sirve para para eliminar dichos datos una vez pasado el tiempo límite de almacenamiento en la base de datos.

A continuación, vamos a crear otra tabla para almacenar las claves que cifran y descifran los datos de las tarjetas de pago, porque según el requisito número 3 de la normativa, cada uno de estos datos ha de estar cifrado con una clave privada y distinta, la llamamos clave de datos. Esta tabla hay que crearla en una base de datos independiente, aislada y encriptada por una clave sólida, de manera que, cuando tengamos que extraer los datos de la tarjeta, lo primero que vamos a hacer es buscar en dicha base de datos (lo cual implica una identificación y autenticación de acceso) la clave de datos correspondiente, la desciframos y después extraer los datos de la tarjeta utilizando la clave encontrada. No obstante, para un proyecto prototipo lo creamos en la misma base de datos. La tabla contiene los siguientes atributos:

- Token, lo mismo que antes, sirve de identificación.
- Clave de datos, que sirve para cifrar y descifrar los datos de las tarjetas de pago almacenados en la base de datos.
- Fecha de creación, servirá para eliminarlos una vez expirado el tiempo permitido de almacenamiento.

Con estos acabamos la fase de preparación de base de datos. Las dos tablas se pueden presentar en las figuras 4.1 y 4.2.

token	cardNum	cardHolder	dateExpiration	tipo	dateCreation
Rwv7R//xRJo7e2VirqX0EpTUEq1XOo44RtYk8qvv9xU=4242	BLOB	Endian	2017-12-00	VisaCard	2017-06-19
NULL	NULL	NULL	NULL	NULL	NULL

Figura 4.1: Datos de la tarjeta en la base de datos

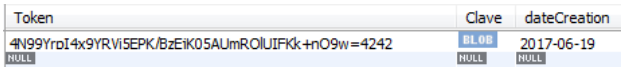

Figura 4.2: Clave de datos en la base de datos

4.3.2 Tokenización de número de tarjeta de pago

Los datos de la tarjeta están almacenados teóricamente en una base de datos aislada, para poder sacarlos hay que tener de cada uno de ellos un identificador. El número de la tarjeta sirve de identificador, sin embargo, según la normativa, este número ha de estar encriptado si lo almacenamos. Por lo tanto, hemos de crear un identificador para cada tarjeta de pago. Una solución es la "Tokenización de número de la tarjeta"[8].

Lo que hace una tokenización es generar una cadena única de caracteres llamada "token", que esconde total o parcialmente los contenidos de datos. De esta manera, esta cadena de caracteres no tiene ningún sentido literalmente dicho.

Esta técnica la aplicamos en el número de la tarjeta y conservamos los últimos 4 dígitos de la tarjeta, la ventaja de este formato es que, facilita la comunicación con los titulares de la tarjeta para que éstos sepan cuál es la tarjeta que estamos tratando. Con este token tenemos un identificador para determinar la tarjeta de pago a la hora de realizar operaciones como lectura y escritura de datos.

Para generar un token seguimos los siguientes pasos:

1. Preparamos la función de Hash256 que habíamos empleado anteriormente.

2. Pasar por la función Hash256 el número de la tarjeta exceptuando los 4 últimos dígitos.

3. Concatenar los 4 últimos dígitos con la cadena generada por la función.

Finalmente, la cadena de TOKEN quedaría como se representa la figura 4.3.

Figura 4.3: Token

4.3.3 Encriptación de datos de la tarjeta

Antes de almacenar los datos de la tarjeta en la base de datos, tenemos que encriptar el número de la tarjeta con una clave única y sólida. Podemos utilizar el algoritmo AES 128 bits según el documento [9] para la encriptación de datos de la tarjeta de pago. Seguimos los siguientes pasos:

1. Generamos una clave única y sólida para la encriptación del número. Para asegurar que la clave sea única y sólida, reutilizamos el algoritmo de SHA de 256 bits. Seguimos los pasos en detalle:

1.1. Extraemos los 4 últimos dígitos del número de la tarjeta y los utilizaremos para crear una clave sólida.

1.2. Añadimos un número aleatorio a estos 4 dígitos para asegurar que la clave generada sea única.

1.3. Pasar los 5 dígitos a la función SHA y obtenemos la clave.

2. Reutilizamos la función AES de 128 bits, pasándole el número de la tarjeta y la clave que hemos generado como parámetro de entrada, nos devolverá un resultado encriptado.

Ahora ya tenemos el número cifrado y preparado para almacenarlo en la base de datos.

4.3.4 Encriptación de clave de datos

La clave de datos que sirve para cifrar y descifrar los datos de la tarjeta de pago ha de estar también encriptada antes de ser almacenada. Seguimos los siguientes pasos para llevar a cabo el proceso de encriptación de la clave de datos.

1. Generamos una clave sólida manualmente para encriptar la clave de datos.

2. Utilizamos la función AES de 128 bits para generar un resultado encriptado, pasándole la clave de datos y la clave que acabamos de crear.

Con esto ya tenemos la clave de datos encriptada y preparada para ser almacenada.

4.3.5 Grabación de datos

Ya tenemos los datos preparados y podemos realizar la grabación de datos. Son dos grabaciones, la primera grabamos los datos de la tarjeta y la segunda, las claves de datos. Las dos grabaciones tienen un mismo procedimiento, por lo tanto, describimos uno para ilustrar:

1. Recogemos todos los valores que deseamos almacenar.

2. Conectamos a la base de datos.

3. Grabar los valores con la instrucción “Insert into"de MySQL.

Las figuras 4.1 y 4.2 representan estos datos una vez queden almacenados en la base de datos.

4.3.6 Lectura

La petición de lectura se ejecuta cuando el cliente sea del tipo registrado y tenga datos previamente guardados en la base de datos. En este caso, dicho cliente tendría un token guardado y con este token hacemos de referencia para buscar sus datos de la tarjeta en la base de datos. Para realizar esta tarea ejecutamos la instrucción de lectura de MySql.

4.3.7 Eliminación

La normativa dice que hemos de eliminar los datos una vez expirado el tiempo máximo de ser almacenados. En este caso, tanto los datos de la tarjeta como las claves de datos, han de ser definitivamente eliminados. Para esto, creamos una instrucción de eliminación de MySql que será ejecutada manualmente para borrar los datos cuando sea necesario.

CAPÍTULO 5

IMPLEMENTACIÓN Y RESULTADOS

Hasta ahora hemos visto el proyecto teóricamente construido, pero en este capítulo veremos los código de programación de algunos componentes del proyecto. Así que ofrecemos una vista técnica sobre la implementación interna del proyecto y también los resultados generados por estos componentes.

5.1 Página de pago

Empezamos con la página de pago siendo ésta uno de los componentes más importantes de nuestro proyecto. Para ver el aspecto de la página se puede consultar la figura 3.1.

En esta sección veremos los códigos de programación que se encargan de llevar a cabo las distintas funciones de nuestra página de pago, tales como la acreditación de datos, auto relleno de datos, etc.

5.1.1 Acreditación de datos

La figura 5.1 representa el código del programa que acredita los datos una vez se ha cargado la página. Y realiza básicamente lo que habíamos comentado en el capítulo anterior, que es identificar los datos que nos llegan, generamos una firma con estos datos y después compararla con la que nos llega.

El programa que genera la firma digital se representa en la figura 5.2. Sobre ésta podemos observar que, existen 4 parámetros de entrada (StrToHash, IV, key y channel) y utilizamos el vector inicial (IV), la clave privada (key) y el canal para crear un encriptor del tipo AES. Después formamos un generador de firma digital con el encriptor y la función Hash de 256 bits, mediante el cual, pasamos los datos (StrToHash) para convertirlos en una firma digital.

```
Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    If Session.Item("sign") Is Nothing Then
        Session("sign") = Request.Form("sign")
    End If
    If Session.Item("bookingID") Is Nothing Then
        Session("bookingID") = bookingID
    End If
    If Session.Item("customerID") Is Nothing Then
        Session("customerID") = customerID
        Session("customerIDForSign") = customerID
    End If
    If Session.Item("userID") Is Nothing Then
        Session("userID") = userID
    End If
    If Session.Item("intentos") Is Nothing Then
        Session("intentos") = 1
    End If
    If Not CStr(Session("bookingID")).Equals("0") Then 'Petición detectada
        res = New BookingEngine.reserva.reserva(Convert.ToInt32(CStr(Session("bookingID"))))
        sign = Server.UrlDecode(CStr(Session("sign"))) 'Firma entrante
        'Generar firma para comparar
        Dim strToHash As String
        strToHash = Session("bookingID") & Session("customerIDForSign") & Session("langForSign")
        If Not String.IsNullOrEmpty(CStr(Session("userID"))) Then
            isUser = True
            strToHash = strToHash & CStr(Session("userID"))
        End If
        mySign = generateSignatureAES(strToHash, IV, PRIVATE_KEY, CHANNEL) 'Generamos la firma
        If Not sign.Equals(mySign) Then 'Comparamos la firma
            errorFirma = True
        Else 'Acceso autorizado
```

Figura 5.1: Código de acreditación de datos

```
Protected Function generateSignatureAES(ByVal StrToHash As String, ByVal IV As String,
                                        ByVal key As String, ByVal channel As String) As String
    Dim keyAes() As Byte
    Dim channelByteArray As Byte() = Encoding.UTF8.GetBytes(channel)
    ' Create an Aes object
    ' with the specified key and IV.
    Dim aesAlg As New AesCryptoServiceProvider()
    aesAlg.Key = Convert.FromBase64String(key)
    aesAlg.IV = Encoding.UTF8.GetBytes(IV)
    aesAlg.Mode = CipherMode.CBC
    aesAlg.Padding = PaddingMode.Zeros
    ' Create a decrytor to perform the stream transform.
    Dim encryptor As ICryptoTransform = aesAlg.CreateEncryptor()

    'Encript the key with channel
    keyAes = encryptor.TransformFinalBlock(channelByteArray, 0, channelByteArray.Length)
    Dim strDs_Signature As String
    Using hmac As System.Security.Cryptography.HMACSHA256 = New HMACSHA256(keyAes)
        strDs_Signature = Convert.ToBase64String(hmac.ComputeHash(Encoding.UTF8.GetBytes(StrToHash)))
    End Using
    Return strDs_Signature
End Function
```

Figura 5.2: Código de generación de la firma digital

Adjuntamos también la figura 5.3 que representa el caso en el que un cliente entra con la firma errónea y denegamos su acceso.

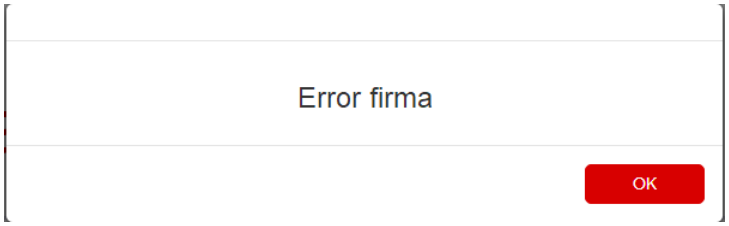

Figura 5.3: Mensaje de error de la firma

5.1.2 Auto relleno de datos

Para que el formulario de la tarjeta de pago auto rellene los datos de la tarjeta del cliente, solo se los tenemos que pedir a la base de datos siempre y cuando el usuario tuviese datos previamente guardados en dicha base de datos. En la figura 5.7 se muestra el programa que extrae datos de la tarjeta de pago.

5.2 Base de datos

A diferencia de la página de pago, la base de datos necesita menos códigos ya que tenemos el gestor de base de datos que nos ayuda a ajustar su configuración. Sin embargo, para realizar operaciones con los datos almacenados, hemos de ejecutar instrucciones correspodientes.

En esta sección veremos la configuración de las tablas que almacenan datos y un ejemplo en código que nos permite extraer los datos de las tarjetas.

5.2.1 Configuración de tablas

En este proyecto necesitamos que la base de datos contenga dos tablas para almacenar dos tipos de datos: una para almacenar los datos de la tarjeta, y otra para las claves que cifran estos datos.

Las siguientes dos figuras 5.4 y 5.5 nos enseñan las configuraciones de ambas tablas según la descripción en el apartado 4.3.1.

Columns in table

Column	Type	Nullable	Indexes
token	varchar(50)	NO	PRIMARY, token_UNIQUE
cardNum	tinyblob	YES	
cardHolder	varchar(45)	YES	
dateExpiration	date	YES	
tipo	varchar(45)	YES	
dateCreation	date	YES	

Figura 5.4: Configuración de la tabla de datos de la tarjeta

Columns in table

Column	Type	Nullable	Indexes
Token	varchar(50)	NO	PRIMARY, Token_UNIQUE
Clave	tinyblob	YES	
dateCreation	date	YES	

Figura 5.5: Configuración de la tabla de claves que cifran datos

5.2.2 Obtención de datos

Para poder obtener datos de la tarjeta desde la base de datos, tenemos que extraer primero la clave de datos para poder descifrar estos datos. La figura 5.6 representa el programa que extrae esta clave, que consisten en 4 pasos:

1. Establecer la conexión con el servidor de la base de datos.
2. Ejecutar el comando con la instrucción *SELECT * FROM claves WHERE token=token*, lo cual significa que queremos sacar la clave con el identificador igual al que hemos introducido.
3. Leemos el valor de la clave con la instrucción *getValue(1)*.
4. Desciframos la clave obtenida con la clave general y guardarla.

```
Private Function getKeyOfKey(ByVal token As String) As String
    Dim conn As New MySqlConnection
    Dim result As String = String.Empty
    Try
        conn.ConnectionString = myConnectionString_KEY
        conn.Open()
        Dim sqlquery As String = "SELECT * FROM claves  where token='" & token & "';"
        Dim data As MySqlDataReader
        Dim adapter As New MySqlDataAdapter
        Dim command As New MySqlCommand
        command.CommandText = sqlquery
        command.Connection = conn
        adapter.SelectCommand = command
        data = command.ExecuteReader
        While data.Read()
            result = DecryptStringFromBytes_Aes(data.GetValue(1), KEY_KEY)
        End While

    Catch ex As MySql.Data.MySqlClient.MySqlException
        Console.Write(ex.Message)
    Finally
        conn.Close()
    End Try
    Return result
End Function
```

Figura 5.6: Código de extraer claves para descifrar los datos de la tarjeta

El procedimiento para sacar los datos de la tarjeta es similar al del programa anterior. Cuando ejecutamos el programa que representa en la figura 5.7, lo primero que hace es ejecutar el programa anterior para sacar la clave, que nos servirá después para descifrar el número de la tarjeta.

Una vez obtenida la clave conectamos al servidor de la base de datos y ejecutamos la instrucción *Select*. A diferencia de lo anterior, esta vez tenemos más informaciones. La instrucción *Select* nos extrae todos los datos de la tarjeta conjuntamente en una cadena de caracteres, por lo que hemos de separarla y distinguir los diferentes tipos de datos, tales como el número de la tarjeta, la fecha de caducidad, etc. El número de la tarjeta está cifrado, y hemos de descifrarlo con la clave anteriormente obtenida. Finalmente guardamos los resultados separándolos con un separador.

```
Private Function getCardByToken(ByVal token As String) As String
    Dim conn As New MySqlConnection
    Dim result As String = String.Empty
    Dim key As String = getKeyOfKey(token)
    Try
        conn.ConnectionString = myConnectionString_PCI
        conn.Open()
        Dim sqlquery As String = "SELECT * FROM data where token='" & token & "';"
        Dim data As MySqlDataReader
        Dim adapter As New MySqlDataAdapter
        Dim command As New MySqlCommand
        command.CommandText = sqlquery
        command.Connection = conn
        adapter.SelectCommand = command
        data = command.ExecuteReader
        While data.Read()
            For i As Integer = 0 To 5
                Select Case i
                    Case 1
                        result = result &
                            DecryptStringFromBytes_Aes(data.GetValue(i), key) &
                            separador
                    Case 3
                        result = result & CType(data.GetMySqlDateTime(i).Month, String) &
                            separador & CType(data.GetMySqlDateTime(i).Year, String) &
                            separador
                    Case 5
                        result = result & CType(data.GetMySqlDateTime(i), String) &
                            separador
                    Case Else
                        result = result & data.GetString(i) & separador
                End Select
            Next
        End While
    Catch ex As MySql.Data.MySqlClient.MySqlException
        Console.Write(ex.Message)
    Finally
        conn.Close()
    End Try
    Return result
End Function
```

Figura 5.7: Código de extraer datos de la base de datos

5.3 Módulo emisor y receptor

Vamos a ver en esta sección los dos módulos que hemos creado para la red de la empresa, que son el módulo emisor y el receptor.

5.3.1 Módulo emisor

Lo que hace este módulo es recopilar información, crear la firma digital y después mandarla a nuestra página de pago. Podemos consultar el apartado 4.1 para conocer los datos que necesitamos y después creamos un *form* automático de tipo "POST"para enviarlos. Todo esto es lo que hace el código que representa en la figura 5.8.

```
Protected Sub booking_include_ctlBookingPayment_include_secureTPV_Load(sender As Object,
                                                                       e As EventArgs) Handles Me.Load
    If Not IsNothing(Me.booking) Then
        Dim customerID As String = frontLibrary.config.settings.get(Of String)("$juniperTPV_customerID")
        Dim strToHash As String = CStr(booking.id) & customerID & i18n.languages.currentCulture
        If webUser.isLogin Then
            strToHash = strToHash & CStr(Usuario.idClienteFinal)
        End If
        Dim sign As String = Server.UrlEncode(generateSignatureAES(strToHash, IV_AES, KEY, CHANNEL))

        'mandamos datos por POST
        Dim url As String = "https://payment.juniper.es/"
        Dim sb As New StringBuilder()
        sb.AppendLine("<body onload='document.forms[0].submit()'>")
        sb.AppendFormat("<form action='{0}' method='post'>", url)
        sb.AppendFormat("<input type='hidden' name='sign' value='{0}'>", sign)
        sb.AppendFormat("<input type='hidden' name='bookingID' value='{0}'>", CStr(booking.id))
        sb.AppendFormat("<input type='hidden' name='customerID' value='{0}'>", customerID)
        sb.AppendFormat("<input type='hidden' name='lang' value='{0}'>", i18n.languages.currentCulture)
        If webUser.isLogin Then
            sb.AppendFormat("<input type='hidden' name='user' value='{0}'>", Usuario.idClienteFinal)
        End If
        sb.Append("</form>")
        sb.AppendLine("<script>$(document).ready(function () {$jbe.loading.show();});</script>")
        sb.AppendLine("</body>")
        Response.Write(sb.ToString())
        Response.End()
    End If
End Sub
```

Figura 5.8: Código del módulo emisor

5.3.2 Módulo receptor

La figura 5.9 representa el código del módulo receptor de nuestro proyecto que hace básicamente lo siguiente:

1. Recibir el ID del producto con el cual el cliente efectuó el pago.
2. Cambiar el estado del producto (bajo el control de error).
3. Registrar el historial de la transacción.
4. Redirigir a la página de confirmación a través de un *form*.

```
Private Sub Page_Load(ByVal sender As System.Object,
                      ByVal e As System.EventArgs) Handles MyBase.Load
    Dim frmTPV As String = String.Empty
    If Not IsNothing(Request.QueryString("bookingID")) Then
        Dim bookingID As Integer = Convert.ToInt32(Request.QueryString("bookingID"))
        Me.booking = New BookingEngine.reserva.reserva(bookingID)

        Dim result As String = String.Empty
        If TPV.utilities.canChangeStatus(Me.booking,
                                         reserva.estadoReserva.estadoCodigo.Pag) Then
            result = TPV.utilities.changeBookingStatus(Me.booking,
                                              reserva.estadoReserva.estadoCodigo.Pag)
        End If
        If Not String.IsNullOrEmpty(result) Then
            'TPV KO
            TPV.utilities.sendMail(BookingEngine.config.mailResponsable,
                    "errortpv@ejuniper.com",
                    canal.Recuperar(Me.booking.id_Canal).NombreCanal &
                    ": Error al pasar a pagada la reserva con localizador: " &
                    Me.booking.localizador,
                    canal.Recuperar(Me.booking.id_Canal).NombreCanal &
                    ": Ha ocurrido un error al pasar a pagada la reserva con localizador: " &
                    Me.booking.localizador & "<br/> Error: " & result,
                    canal.Recuperar(Me.booking.id_Canal))

            Me.changeToSelectedStatus(Me.booking, "Quo")
            Me.rsp = New TPV.response(TPV.response.codes.KO,
                    "Error-23: Ha ocurrido un error al pasar a pagada la reserva. ",
                    TPV.response.errorTypeENUM.juniper)
        Else
            'Status OK
            Me.rsp = New TPV.response(TPV.response.codes.OK,
                        frontLibrary.i18n.get("msgPaymentT"))
        End If
        Dim idLog As Integer = Nothing
        If Me.rsp.ID = TPV.response.codes.KO Then
            idLog = TPV.utilities.saveOnLog(LogBBDD.proveedor.Todos, LogBBDD.nivel.Urgente,
                                            False, "TPV", "Secure TPV:" & rsp.msg)
            rsp.msg = "TPV - LOG ID: " & idLog & " -->" & rsp.msg
        End If
        TPV.utilities.saveOnBookingHistory(New ToolsLibrary.StringEsEn(Me.rsp.msg, Me.rsp.msg),
                                           Me.booking)
        'liberar semaforo antes de salir
        liberaSemaforo()
        Dim url As String = "/booking/bookingDetails.aspx?localizador=" &
            Me.booking.localizador & "&mail=" &
            Me.booking.email & "&status=" & Me.rsp.ID & "&from=" & Me.rsp.errorType
        frmTPV = "<form name='volver' id='volver' action='" & url &
            "' method='post' target='_parent' />" &
    "<script type='text/javascript'>if (navigator.appName.indexOf('Explorer') == -1) " &
    "{window.parent.location.href='" &
                url & "';}else{document.volver.submit();}</script>"
        Response.Write(frmTPV)
    End If
End Sub
```

Figura 5.9: Código del módulo receptor

CAPÍTULO 6

CONCLUSIÓN

Las empresas que venden productos o servicios online necesitan que sus sistemas de pago sean seguros y aprobados por la normativa PCI DSS, al mismo tiempo, éstas no quieren invertir sus dineros en la aplicación de la normativa, porque dicha normativa requiere el cumplimiento de sus requisitos en todos los componentes de la red que se conectan directamente o indirectamente al sistema de pago, además para que sus sistemas sean aprobados por la normativa hay que realizar una evaluación y mantenimiento periódicamente para garantizar el cumplimiento de requisitos de estos componentes.

Juniper ha diseñado una solución que consiste en instalar un sistema de pago en un entorno independiente, capaz de gestionar los datos de la tarjeta de pago para múltiples empresas. El entorno independiente tendrá menos componentes, que facilitará tanto a la aplicación de la normativa como la evaluación y mantenimiento del sistema. Sin embargo, para lograr esta solución se necesita dos procesos: el primero es crear el entorno con un sistema de pago y aplicar los requisitos número 3, 4 y 8 (que son los requisitos que afectan directamente al funcionamiento de los componentes); mientras el segundo es aplicar los requisitos restantes de la normativa sobre este entorno.

Este proyecto corresponde al primero proceso, cuyo objetivo es construir un entorno de pago para facilitar la realización del segundo proceso. Para ello hemos realizado las siguientes tareas ya mencionadas en el capítulo 1:

1. Diseño del entorno de pago.
2. Desconexión del entorno de pago de una empresa de prueba.
3. Creación de una red pública en un servidor virtualizado.
4. Creación de una página que gestiona y procesa datos de la tarjeta de pago, además, es de estilo dinámico y multi-idioma.
5. Creación de base de datos para almacenar los datos de la tarjeta.
6. Tokenización de número de la tarjeta para proteger los datos almacenados.

7. Creación de un método de encriptación para cifrar los datos.
8. Creación de un sistema de identificación y autenticación para controlar el acceso al entorno.
9. Creación de un sistema de gestión de datos.
10. Creación de dos módulos para transmisión y recepción de datos.
11. Establecimiento de la comunicación entre las dos redes.
12. Otras tareas menos relevantes como traducciones de idiomas, control de errores, pruebas de funciones del sistema, etc.

Los requisitos número 3 y 8 de la normativa han sido desarrollados y aplicados, que están demostrados en los capítulos 4 y 5. En cambio, no se ha podido aplicar el requisito número 4 de la normativa, que es cifrar los datos de la tarjeta en transmisión, porque desconocemos el método de encriptación que utilizan los bancos.

Existen dos puntos mejorables en este proyecto: el estilo de la página y la opción de elegir la tarjeta de pago cuando un cliente tiene almacenada más de una tarjeta. Ninguno de los dos anteriores es el objetivo del proyecto, pero para satisfacer en la medida de lo posible a las necesidades del cliente y de las empresas que se conectan, valdría la pena mejorarlos.

Es un proyecto de trabajo en equipo. Recordamos que existían dos módulos (emisor y receptor) que teníamos que instalar en la página de la empresa, esto implica que hay que pedir información a los desarrolladores de esta página. No es nada fácil trabajar en equipo en una empresa relativamente grande, ya que cada uno trabaja para su departamento y no siempre se encuentra disponible. Mi experiencia no fue mala, pero tampoco satisfactoria, porque el tiempo de respuesta de los compañeros muchas veces se demora bastante, pero gracias a esta experiencia aprendí cómo organizar y aprovechar mejor el tiempo.

En definitiva, este proyecto no tan solo ha sido un proyecto de programación sino que se han realizado una serie de tareas, tales como la consulta de la normativa, análisis de los requisitos, el diseño de soluciones para cumplir dichos requisitos, y finalmente programación de la solución.

BIBLIOGRAFÍA

[1] P. S. S. Council. Normas de seguridad de datos de la pci. [Online]. Available: https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2_3_es-LA.pdf (document), 1, 1

[2] M. S. I. A. B. A. S. Najma Imtiaz Ali, Suhaila Samsuri, "Online shopping satisfaction in malaysia: A framework for security, trust and cybercrime," *2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*, pp. 194 – 198, 11 2016. 1

[3] P. S. S. Council. About us. [Online]. Available: https://es.pcisecuritystandards.org/about_us/ 1

[4] Juniper. Juniper innovation travel technology. [Online]. Available: https://www.ejuniper.com 1

[5] W3C. Css snapshot 2017. [Online]. Available: https://www.w3.org/TR/css-2017/ 2.2.4

[6] Wikipedia. Extensible markup language. [Online]. Available: https://es.wikipedia.org/wiki/Extensible_Markup_Language 1

[7] W. Tofan, Hermawan; Rini Wisnu, "Implementation aes with digital signature for secure web-based electronic archive," *International Conference on Information Technology and Electrical Engineering (ICITEE)*, pp. 1 – 6, 08 2016. 3.4.2, 3.4.2

[8] T. T. P. S. S. C. Scoping SIG. Pci dss tokenization guidelines. [Online]. Available: https://www.pcisecuritystandards.org/documents/Tokenization_Guidelines_Info_Supplement.pdf 4.3.2

[9] P. S. S. Council. Glosario de términos, abreviaturas y acrónimos de la pci. [Online]. Available: https://www.pcisecuritystandards.org/documents/PCI_DSS_Glossary_v3-2_3_es-LA.pdf 4.3.3