

Informe de Laboratorio 2

José Miguel García¹

Soledad García¹

Myriam Delgado¹

1. INTRODUCCIÓN

Actualmente, se tiene la necesidad de realizar un sistema para varias universidades el cual gestione los pagos de las matrículas de sus estudiantes.

El sistema será utilizado por el estudiante como una pasarela de pago para realizar su matrícula asociada a una institución universitaria determinada. Al finalizar la transacción, si esta es exitosa se obtendrá un comprobante de pago acompañado de un código QR.

Para cada crear la pasarela de pagos es necesaria la siguiente información:

- Identificación: Esta será la cédula o número de identificación del estudiante.
- Nombres y apellidos: Nombres completos del estudiante.
- Institución: Institución educativa universitaria en la que se desea realizar el pago.
- Cantidad: Monto total del pago a realizar.
- Número de la tarjeta de crédito.
- Fecha de vencimiento de la tarjeta de crédito.
- Código de verificación de la tarjeta de crédito.

Este documento detalla el sistema creado para satisfacer la necesidad descrita anteriormente el cual corresponde al Laboratorio 2 del curso de Profundización en la Arquitectura de Software, teniendo la siguiente estructura: En la **sección 2** se detallan los objetivos que se desean cumplir con el desarrollo de dicho sistema, en la **sección 3** se describen las herramientas empleadas en este desarrollo, en la **sección 4** se ilustra la arquitectura empleada para este desarrollo, en la **sección 5** se da explicación del desarrollo, anotaciones utilizadas y se ilustran algunos pantallazos del sistema, en la **sección 6** se busca dar a conocer algunas mejoras que se desearían implementar en un futuro, en la **sección 7** se especifican las conclusiones y finalmente, en la **sección 8** se incluye la bibliografía.

2. OBJETIVOS

Objetivo General

Crear una aplicación Web que permita simular una pasarela de pago para los estudiantes de las diferentes universidades.

¹ Estudiante de Maestría en Ingeniería, Grupo Investigación ITOS. Universidad de Antioquia

Objetivos específicos

- Comprender el uso de los recursos JDBC y pool de conexiones para el manejo de las conexiones y concurrencia en una base de datos de una pasarela de pagos para una universidad.
- Implementar las anotaciones JSP para simplificar la persistencia y el mapeo de una base de datos objeto-relacional de una pasarela de pago.
- Crear validaciones en la vista y respaldo de las transacciones realizadas usando códigos QR y captchas con tecnologías de google.
- Aplicar el uso de EJB-Session Beans: Stateful, Stateless y Singleton para la seguridad de una aplicación transaccional.
- Utilizar Servlets para la lógica del negocio y JSF para la vista de la pasarela de pago.
- Implementar los objetos request y response que nos proporcionan los Servlets.
- Incluir JSF con primefaces en la creación de una pasarela de pago web.
- Aplicar una arquitectura básica de 3 o N capas para el desarrollo de la aplicación web.
- Aplicar el uso de los patrones de diseño DTO y DAO.

3. HERRAMIENTAS EMPLEADAS

Las herramientas utilizadas en este sistema son las siguientes:

- **PrimeFaces**: Librería de Java Server Faces (JSF) que permite dar forma a la vista o el componente visual de la pasarela de pagos mediante etiquetas XHTML.
- **MariaDB**: Base de datos para guardar la información de la pasarela de pago(es la versión nuevo de MySQL). Se utilizó la versión 10.1.11.
- **NetBeans**: IDE seleccionado que facilitó el desarrollo del software. Se utilizó la versión 8.1.
- **GlassFish**: Servidor de aplicaciones para desplegar el sitio web. Se utilizó la versión 4.1.1.
- **Sequel Pro**: Programa para la gestión de bases de datos MySQL.
- **Java EE**: Plataforma seleccionado para el desarrollo del software.
- **Java Servlets**: Servlets utilizados para responder a las solicitudes de la vista.
- **Enterprise JavaBeans**: Tecnología utilizada para crear y reutilizar el componente de pago el cual maneja las transacciones entre el estudiante y la universidad.
- **Web Browser**: Permite visualizar la interfaz web gráfica de la aplicación.
- **LucidChart**: Herramienta web para realizar diagramas UML.
- **reCaptcha de Google**: Se utilizó una clave API de Google para generar el código recaptcha.

4. ARQUITECTURA

La arquitectura de la aplicación se ha diseñado teniendo en cuenta 3 capas: Lógica de la Aplicación, Lógica del Negocio y Modelo. En la Figura 1, se ilustra la arquitectura de la aplicación por paquetes de acuerdo a estas capas.

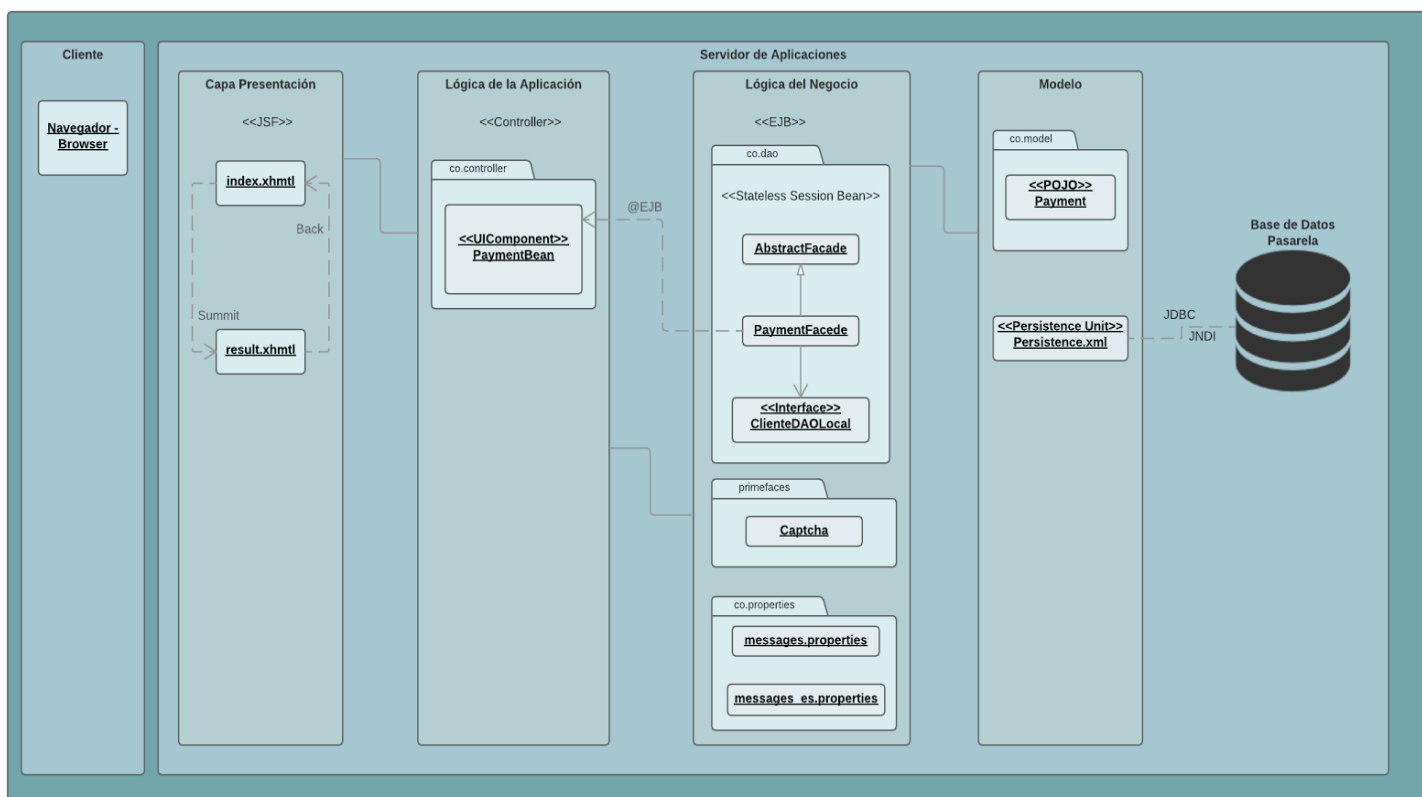


Figura 1. Arquitectura por capas del sistema

De acuerdo a la Figura 1, el **cliente o navegador** hace una petición http a la **Capa de Presentación**, esta capa cuenta con 2 archivos index.xhtml y result.xhtml, ambos son componentes JSF; entre estos componentes hay una interacción en donde *index* valida la información ingresada por el usuario, y si esta es correcta muestra el resultado en *result* utilizando *submit*.

En esta capa, también, se encuentra el archivo faces-config el cual mapea por medio de beans los recursos que necesitan las páginas JSF para realizar las operaciones, sin este archivo *index* y *result* no podrían acceder a ninguna capa ilustrada en dicha figura.

En la capa de la **Lógica de la Aplicación**, el paquete *co.controller* tiene la clase *PaymentBean*, el cual captura los eventos de la capa de presentación.

La capa de la **Lógica del Negocio** contiene el paquete *co.dao* en donde los componentes de este paquete se encargan de la lógica del negocio, para esto expone una interfaz que permite capturar o identificar la entidad que se necesita para hacer la operación, y así suministrar esta información a la capa del **Modelo** para que esta a su vez la persista en la base de datos mediante una unidad de persistencia que se conecta a dicha base de datos utilizando JDBC y JNDI.

Adicional a lo anterior, en la **Lógica del Negocio** se encuentra, también, el paquete *primefaces* que contiene la clase que gestiona el reCaptcha, y el paquete *co.properties* el cual contiene los mensajes de validación de la aplicación en diferentes idiomas.

5. DESARROLLO

De acuerdo a las anteriores sesiones, creamos una aplicación para simular una pasarela de pagos; usamos el IDE NetBeans en la versión 8.1 con un servidor de base de datos 10.1.11-MariaDB Homebrew, la aplicación corre sobre un servidor de aplicaciones GlassFish Server Open Source Edition 4.1.1 (build 1).

Creamos la base de datos con la ayuda del IDE para almacenar los pagos de los estudiantes. Para el desarrollo del Modelo creamos 1 POJO, *Payment*, mapeado mediante una interfaz local de acceso a los datos, posteriormente se crea su respectivo EJB. En el modelo se crea, también, la unidad de persistencia la cual persiste la base de datos conectándose a esta mediante JDBC (que maneja toda la conexión de la base de datos) y JNDI (que maneja la ubicación de la base de datos).

El POJO creado tomó la información en la base de datos para construirse por medio del IDE, de la misma manera se crearon las interfaces y la lógica de la aplicación, es importante resaltar que no todo se desarrolló de manera automática, la lógica de la captcha, las validaciones, establecer el flujo de navegación para la vista y las paginas web se crearon manualmente desarrollando en Java Server Faces (JSF); la capa de vista incluye 2 páginas: *index.xhtml* y *result.xhtml*.

En el desarrollo de este sistema se utilizaron anotaciones para indicar el procesamiento de algunas clases, variables y métodos, a continuación se listan las anotaciones utilizadas:

@Local Se usa para indicar que la interfaz *PaymentFacadeLocal* es una interfaz local.

@Entity Significa que la clase será una entidad usada para realizar consultas en una base de datos

@XmlRootElement El propósito de esta anotación es asociar el elemento *root* o raíz con una clase, en este caso con la clase *Payment.java*

@NamedQueries(@NamedQuery(name="Nombre de la consulta ",query="consulta en SQL"))
Indica que existirá una consulta en la base de datos de nombre "name" y la consulta en lenguaje de persistencia de java estara contenido en "query"

@Id Con la anotación anterior indicamos que el campo definido será una clave primaria dentro de la base de datos.

@GeneratedValue(strategy = GenerationType.[AUTO|IDENTITY|SEQUENCE|TABLE]) Es usada en los proyectos que tienen claves primarias simples; define la forma de generación de la clave, si necesitamos descargar la responsabilidad en el proveedor de persistencia y la base de datos usamos AUTO, pero si necesitamos que el proveedor de persistencia tenga el control será IDENTITY si necesitamos consecutivos generados por la base de datos usamos SEQUENCE o si son valores que se aseguran por el proveedor de la base de datos usaremos TABLE.

@Basic(optional = "false o true") Indica que un atributo debe ser persistido y un mapeo estándar debe ser utilizado

Como parámetro utilizamos *optional* que permite indicar si el campo será nulo. Por defecto true. En este caso el *id* no debe ser nulo, por lo que se pone *false*.

@Temporal(TemporalType.DATE)

Se utiliza junto con la anotación @Basic para especificar que un campo fecha debe guardarse con el tipo java.util.Date o java.util.Calendar. Si no se especificara ninguno de estos por defecto se utiliza java.util.Timestamp.

@Size(max = "numero") Se utiliza para indicar y posteriormente evaluar la longitud mínima y máxima de un campo o propiedad. En este caso, solo se ha indicado una longitud máximo así: *max=30*.

@Stateless Define que el bean será de tipo Stateless sesión bean significa que no contendrá datos en mientras se realizan las diferentes peticiones, servirá solo por demanda.


@PersistenceContext(unitName = "Unidad de Persistencia") Expresa la unidad de persistencia de la cual depende, en este caso *unitName* es igual tiene como valor "MatriculaEstudiantePU".

@Override Significa que se sobrescribe un método que hace parte de una clase de orden superior.

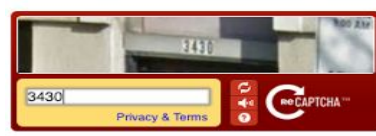
@EJB Indica la dependencia con el bean que se expresa a continuación.

Ahora, en la Figura 2, se ilustra la página inicial para que el estudiante ingrese la información del pago a realizar, en esta página se incluye también un código reCaptcha para mayor seguridad en el pago.

Proporcione la información de la tarjeta

Id pago *	<input type="text" value="12"/>
Nombre *	<input type="text" value="Soledad Garcia"/>
Institución *	<input type="text" value="udea"/>
Monto *	<input type="text" value="4800000"/>
Tarjeta *	<input type="text" value="3"/>
fecha de vencimiento *	<input type="text" value="04/2016"/> 
CVV No *	<input type="text" value="231"/>
Contraseña *	<input type="password" value="....."/>
Repita contraseña *	<input type="password" value="....."/>

Primefaces - Captcha



Check Submit validar

Figura 2. Formulario de Pago

Una vez ingresada esta información el usuario puede visualizar la página de la confirmación (notificando la respuesta del pago) ilustrada en la Figura 3.

Infomación de pago

Id pago 0
Monto \$48.000,00
Tarjeta 9647893756483742
TIPO TARJETA
CVV No 321
fecha de vencimiento 11/2021



Back

Figura 3. Confirmación del pago

Esta aplicación también incluye una versión en inglés ilustrada. La figura 4 ilustra la página en inglés para el formulario de pago.

Please provide your card details

Id Invoice *	<input type="text" value="0"/>
Name *	<input type="text" value="Soledad Garcia"/>
Institution *	<input type="text" value="u de a"/>
Amount *	<input type="text" value="48,000.00"/>
Credit Card *	<input type="text" value="9647893756483742"/>
Expiration date (Month/Year) *	<input type="text" value="11/2021"/>
CVV No *	<input type="text" value="321"/>
Contraseña *	<input type="password"/>
Repita contraseña *	<input type="password"/>

Primefaces - Captcha




[Privacy & Terms](#)

Figura 4. Formulario de Pago (Versión en inglés)

Finalmente, la aplicación también tiene validaciones en los campos que se deben ingresar en el formulario de pago.

Proporcione la información de la tarjeta

Id pago *	<input type="text" value="0"/>	
Nombre *	<input type="text"/>	<input checked="" type="checkbox"/> Un nombre es requerido
Institución *	<input type="text"/>	<input checked="" type="checkbox"/> Una institución es necesaria
Monto *	<input type="text" value="0,00"/>	<input checked="" type="checkbox"/> Monto: Error de validación: el atributo especificado no está entre los valores esperados: 100 y 50.000.
Tarjeta *	<input type="text"/>	<input checked="" type="checkbox"/> Una tarjeta es necesaria
fecha de vencimiento *	<input type="text" value="04/2016"/> 	
CVV No *	<input type="text" value="0"/>	<input checked="" type="checkbox"/> CVV No: Error de validación: el largo es inferior que el mínimo permitido de '3'
Contraseña *	<input type="password"/>	<input checked="" type="checkbox"/> La clave no puede ser vacia
Repita contraseña *	<input type="password"/>	<input checked="" type="checkbox"/> La segunda Clave es necesaria

Primefaces - Captcha

☒ Un nombre es requerido

Una institución es necesaria

Monto: Error de validación: el atributo especificado no está entre los valores esperados: 100 y 50.000.

Una tarjeta es necesaria

CVV No: Error de validación: el largo es inferior que el mínimo permitido de '3'

Figura 5. Validación de los campos

En la Figura 5 se ilustra dicha validación en caso que el usuario no ingrese correctamente los datos.

6. POSIBLES MEJORAS

Durante el desarrollo de este software se analizaron posibles mejoras que podrían ser implementadas en un futuro. A continuación, se listan dichas mejoras:

- Exponer esta pasarela a futuras implementaciones, esto podría ser mediante un servicio web o la posibilidad de agregarse como widget o iFrame y así otras aplicaciones que en un futuro se desarrollen y se necesite realizar un pago, podría utilizarse este desarrollo. Por supuesto, para lograr esto se debe estandarizar más esta aplicación para que se pueda integrar en diferentes contextos

7. CONCLUSIONES

- El manejo del pool de conexiones favorece el incremento de la escalabilidad y el rendimiento de la aplicación desarrollada en el laboratorio. Al utilizar el pool de conexiones las arquitecturas son más robustas que en una arquitectura tradicional cliente/servidor. El pool permite tener centralizado y controlado el manejo de las conexiones a la base de datos, ya que el acceso a la misma no se hace desde el cliente, como en una aplicación en dos capas, sino que en este tipo de aplicación el acceso es realizado por el servidor de aplicaciones. El pool es capaz de ofrecer múltiples conexiones lógicas utilizando un reducido número de conexiones reales. Es así como el manejo de un pool de conexiones favorece la escalabilidad y el rendimiento de la aplicación desarrollada en este laboratorio.

- Java Server Faces (JSF) permite construir aplicaciones web que implementan una separación entre el comportamiento y la presentación tradicionalmente ofrecidas por arquitectura UI del lado del cliente. Es posible implementar de manera independiente los componentes web del negocio y se simplifica el trabajo del desarrollador permitiendo la división de tareas por capas, donde los componentes del sistema son visibles disminuyendo la complejidad.
- Al utilizar Frameworks como JSF y PF se pueden realizar desarrollos mucho más ágiles conservando patrones de diseño. JSF permite abstraerse de varios de los problemas más comunes a la hora de implementar una página web. También permite un alto grado de configuración al permitir diseñar y administrar componentes y funcionalidades propias.
- El uso de patrones de diseño facilita el mantenimiento de las aplicaciones en las fases de codificación y de pruebas. El modelo vista controlador MVC es un patrón de diseño que permite que los requisitos no funcionales se cumplan al separar la aplicación en diferentes capas. Actualmente existen muchos frameworks para el desarrollo de aplicaciones basadas en MVC; sin embargo su utilización requiere de un entrenamiento para sacarle el mayor provecho posible.

8. BIBLIOGRAFIA

Botia, D(2016). WebSite Profundización en Arquitecturas de Software. Recuperado el 19 de Marzo de 2015 a partir de: <https://sites.google.com/site/profarquitecturasdesw>

ISOFT(2009). JDBC: acceso a bases de datos. Recuperado el 10 de Marzo de 2015 a partir de: <http://www.vc.ehu.es/jiwotvim/ISOFT2009-2010/Teoria/BloqueIV/JDBC.pdf>

Oracle. Annotations Basics. Recuperado el 19 de Marzo de 2015 a partir de: <https://docs.oracle.com/javase/tutorial/java/annotations/basics.html>

Oracle. The Java EE 6 Tutorial. Recuperado el 19 de Marzo de 2015 a partir de: <http://docs.oracle.com/javaee/6/tutorial/doc/gipjg.html>

Bootstrap community. About Bootstrap. Recuperado el 19 de Marzo de 2015 a partir de: <http://getbootstrap.com/about/>

SequelPro. Portal de SequelPro. Recuperado el 19 de Marzo de 2015 a partir de: <http://www.sequelpro.com/>

Prime Faces. Why PrimeFaces. Recuperado el 19 de Marzo de 2015 a partir de: <http://www.primefaces.org/whyprimefaces>