



**INFORME DEL SISTEMA**

**PARA EXAMEN DE DESARROLLO EN JAVA DEJ4501**

Integrantes: Francisco Benítez

Daniel Carrasco

Javier Salgado

Profesor: Juan Abarca

Fecha de entrega: 13/12/2017

**ÍNDICE**

|  |  |
| --- | --- |
| **Tema** | **Página(s)** |
| Introducción | 3 |
| Descripción del proyecto | 4-6 |
| Código de fuente documentado | 7 |
| Conclusión | 8 |
| Bibliografía | 9 |

**INTRODUCCIÓN**

El siguiente informe presenta la realización del proyecto de sistema de pago para un conjunto de peajes (Se profundizará en la descripción), en el cual debemos aplicar aprendizajes adquiridos durante el semestre, los cuales a través de una plantilla web nos permiten integrar: los datos ingresados por el usuario, combinados con algoritmos creados por nuestro sistema y conexión a base de datos. Esto permite desarrollar sistemas de soluciones a diversos tipos de problemas.

Cabe recalcar la persistencia de datos, por lo que, además de proveer un resultado a partir de parámetros ingresados por el usuario, esta información es almacenada permanentemente en la base de datos, para su posterior uso, que hace más eficiente el proyecto mismo y con esto la interacción del usuario. Por otra parte una base de datos es una mina de recursos, el recurso de la información que permite leer tendencias y hacer proyecciones, para implementar decisiones a nivel empresa.

Falta mencionar el uso de log, que registrará las acciones realizadas por los usuarios, para el análisis del proyecto mismo, y así facilitar la evolución del mismo. Idealmente para excepciones que se presenten y que no hayan sido mencionadas por los requisitos del cliente, y tampoco en el diseño por parte nuestra. Esto permite un soporte al cliente de nuestro proyecto, ante cualquier eventualidad. Ahora en caso de errores, el log también ayudará a identificar la falla, pero no es el objetivo del log, ya que un proyecto profesional no debe albergar cabida a errores, denotando estos como elementos distintos de las excepciones.

**DESCRIPCIÓN DEL PROYECTO**

La cronología de las partes en orden fueron, crear las vistas(JSP), el modelo, luego los servlet, posteriormente el modelo DAO y las query de consultas. Entre los primeros pasos fuimos modificando la base de datos en MYSQL.

Y el detalle para los requerimientos solicitados, usamos el diseño MVC:

**MODELO**

En el modelo creamos las clases para lo explícito: carretera, empresa de cliente, tipo de pedido, opción de retiro; por otra parte para lo implícito las clases:

Pedido, con atributos:

id (primary key en base de datos para tabla Pedido); atributo interno.

nombre del comprador y empresa (con rut, nombre y dirección); datos cliente.

tipo de pago, tipo de retiro, cantidades de cada autopista; atributos para cálculos.

con métodos: “getters and setters”

con método: calcular Total.

Voucher, con métodos: “getters and setters”

con atributos: pedido y total, la clase descrita anteriormente y el monto total a pagar.

Conexión a base de datos llamada peaje.sql, que contiene las tablas relacionadas según la lógica del problema, con tablas:

Carreteras, cuyas columnas son id, nombre y valor

Empresa, cuyas columnas son rut,nombre,direccion e idPersona

Opciones de pago, cuyas columnas son id y tipo

Opciones de retiro, cuyas columnas son id y tipo

Pedido , cuyas columnas son numeroPedido, cantidad,

opcionPago,opcionRetiro,rutEmpresa, idCarretera.

Persona, cuyas columnas son id,nombre, idEmpresa.

**VISTA**

En la vista creamos los JSP para lo explícito de los requerimientos:

La página principal, con formularios para el ingreso de los datos del pedido, un seleccionador de carreteras junto a un botón agregar a la sección que muestra las carreteras y así ingresar la cantidad que desee en tal carretera. “Check Botton” para opciones de pago, y otro para opciones de retiro. Botón para hacer el pedido y así ser redirigido al voucher. Y aparte los botones del index, y vinculo buscar pedidos anteriores.

voucher, el cual muestra el número asignado de pedido, un cuadro con el detalle de la compra, el total a pagar en forma de texto y el botón para confirmar la compra. Y aparte los botones del index, y vinculo buscar pedidos anteriores.

mostrar pedidos, el cual despliega todos los pedidos ya hechos almacenados en la base de datos. Con un redireccionamiento a los formularios llenados en caso de escoger alguno antiguo. Y aparte los botones del index.

Por otra parte jsp para mostrar éxito, y otro, para mostrar error, en las pruebas de ingreso de datos.

Y También el html index con accesos a ingreso de pedidos y mostrar pedidos.

**CONTROLADOR**

En el controlador los accesadores a las conexiones clases:

CarreteraDAO con string de conexiones para acciones readAll y read.

EmpresaDAO con string de conexiones para acciones insert, readAll, read.

PedidoDAO con string de conexiones para acciones insert,delete, readAll.

ICarretera para cargar el arraylist de la clase carreteras

IEmpresa para cargar el arraylist de la clase empresas

Ipedido para cargar el arraylist de la clase pedido.

La clase conexión, que permitirá instanciar la conexión a la base de datos.

Y los servlets que controlaran la interacción del modelo y la vista:

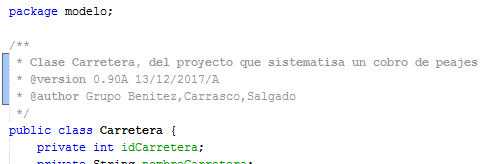
Servlet de consultar o mostrar.

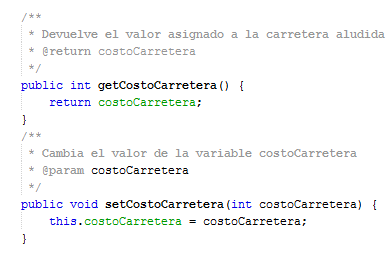
Servlet de ingreso de empresa.

**CÓDIGO FUENTE DOCUMENTADO**

Por motivos de tiempo no hemos documentado el proyecto. Ya que nos hemos enfocado en la funcionalidad del trabajo, dejando para el final este punto. Entendemos la importancia de la documentación para la lectura de otras personas al proyecto, mostrando el tipo de parámetros que se deben ingresar, la descripción de los métodos y detalles del uso de estos.

Al menos hemos de ejemplificar algunas documentaciones que se debieron hacer:





**CONCLUSIÓN**

La arquitectura MVC permitió no perdernos en la integración de la información existente y a escribir en la base de datos, más los parámetros ingresados por el usuario a través de la interfaz web, junto con los algoritmos mismos de las clases explícitas. Esto ayudaría también, la lectura del código para otra persona, con los conocimientos informáticos pertinentes, pueda interactuar con el proyecto completo.

La consistencia de la base de datos es primordial, y es lo que nos tuvo frenados unos cuantos días, ya que no encontrábamos el error porque no interactuábamos con bases de datos “complejas” desde el semestre pasado. Llegó el momento en que la hicimos todo de nuevo y funcionó, detectando que las relaciones que hicimos contenían errores. Sin la base de datos creada en forma correcta, no podemos hacer nada significativo más que mostrar valores que se borrarían al cerrar la ventana. Estimamos ardua la dificultad de este tipo, por combinar conocimientos de base de datos, junto a los de Java. Y la interacción de uno al otro, y viceversa.

Al superar las dificultades de la base de datos, se obtiene una valiosa base de datos, que permite hacer procesos mas eficientes y tener una mina de recursos (información), que será de gran utilidad para el cliente que solicita la solución.

En cuanto a la vista o interfaz web, el diseño no fue nuestro centro, ya que la funcionalidad es primordial, esta funcionalidad tampoco a la ligera, sino un proyecto que cumpla los requerimientos del cliente y además “predisponerse a errores capa 8”, aprovechando las funcionalidades del try catch, para depurar errores de ingreso.

La herramienta Github, permitió el trabajo en equipo de manera remota. Así superamos dificultades de compatibilidad de tiempos, ya que trabajamos aparte de cursar la carrera.

DEJ nos permitirá implementar proyectos o soluciones desde pequeños emprendimientos en adelante. Agradecemos todo lo enseñado.

**BIBLIOGRAFÍA**

Usamos los proyectos hechos durante el semestre

Actividad universidad

Mantenedor Alumnos

Ejemplo PrePrueba

API de Logging para registrar la actividad del usuario y excepciones ocurridas en el sistema.

<https://docs.oracle.com/javase/7/docs/api/java/util/logging/Level.html>

<http://www.oracle.com/technetwork/java/index.html>