UNSIS

Universidad de la Sierra Sur

Docendo Discimus

Miahuatlán de Porfirio Díaz

Paradigmas de Programación

Proyecto: Agencia Speed Agency

Profesor: M.T.E Everardo Pacheco

Alumnos: Eleazar Jarquin Ramos

Giovany Herrera Lopez

America Yaridsaida Villalobos Rodriguez

Fecha 30/01/2025

índice

Introducción:

El presente proyecto en java tiene como objetivo el desarrollo de un sistema integral de gestión para una agencia de autos que incluye un inventario de vehículos, un control de empleados y la realización de ventas. Este sistema esta diseñado para optimizar la administración de los vehículos en inventario, gestionar eficientemente los datos de los empleados y facilitar las transacciones de ventas de manera ágil y organizada.

Este proyecto se enmarca dentro de los temas estudiados en la materia de Paradigmas de Programación Orientado a Objetos 2, donde se han aplicado diversos conceptos fundamentales

Este proyecto no solo mejora la operatividad de la agencia de autos, sin1o que también representa la aplicación practica de los principios de la programación orientada a objetos, con el fin de crear un sistema robusto, eficiente y escalable.

Desarrollo

Descripción de la Aplicación

“ El sistema de gestión para una agencia de autos permite que un encargado en este caso un administrador tenga acceso al sistema y a si mismo crear cuentas para otros usuarios para puedan acceder al punto de venta que es donde se realizan las ventas, recursos humanos que es donde el usuario encargado puede consultar , dar de alta, actualizar, eliminar y crear una nueva cuenta buscar empleados por id y nombre. El que tiene cuenta de contador tiene acceso a registrar compra, Registro de gastos y registrar deuda, y el administrador que tiene acceso a todo el programa esto dependiendo al rol de usuarios. A través de una interfaz gráfica los usuarios pueden interactuar con la base de datos para realizar estas operaciones de manera sencilla y eficiente”

Arquitectura Modelo-Vista-Controlador (MVC)

Modelo:

El paquete `Modelo` es responsable de gestionar la comunicación entre la aplicación Java y la base de datos PostgreSQL, permitiendo realizar operaciones como consultas y actualizaciones sobre las tablas relevantes para el sistema de inventario de vehículos, empleados y otros aspectos de la agencia. A continuación, se describe la funcionalidad de cada clase dentro de este paquete:

Clase `Conexión`

Esta clase es la responsable de gestionar la conexión con la base de datos. Contiene los métodos necesarios para establecer, mantener y cerrar la conexión. Utiliza parámetros configurados para autenticar y realizar las consultas a la base de datos, evitando la necesidad de ingresar credenciales de forma manual. Además, la clase maneja errores relacionados con la conexión de manera eficiente.

Clase `Querys`

La clase `Querys` es la encargada de realizar las consultas a la base de datos para obtener información relevante de las tablas. Incluye una serie de métodos que permiten buscar, obtener y filtrar registros de la base de datos según criterios específicos. Algunas de las funcionalidades de esta clase incluyen:

- **`buscaCliente(String id)`**: Realiza una consulta para obtener información de un cliente según su ID.
- **`buscaModelo()`**: Busca vehículos cuyo modelo coincida parcialmente con el valor proporcionado.
- **`buscaBrand()`**: Filtra los vehículos por la marca proporcionada.
- **`queryGastos()`**: Recupera las descripciones de los gastos registrados.
- **`queryDeudas()`**: Obtiene las descripciones de las deudas registradas.
- **`buscaId()`**: Busca vehículos cuyo ID coincida parcialmente con el proporcionado.
- **`buscaNombre()`**: Filtra vehículos por su nombre.
- **`queryInventary()`**: Obtiene todos los registros de la tabla de vehículos.
- **`queryUpdateI(String id)`**: Obtiene los detalles de un vehículo específico para actualizarlo.
- **`queryusers(String tabla, String user, String passw)`**: Verifica si las credenciales del usuario son correctas en la base de datos.
- **`llenaActulizarE(String id)`**: Recupera los datos de un empleado para ser mostrados en la interfaz de actualización.
- **`consultaEmpleados()`**: Obtiene los IDs y nombres de los empleados registrados.
- **`getShopPromo()`**: Obtiene los nombres de los vehículos disponibles.
- **`endRegister()`**: Recupera el último ID de registro de los registros de la base de datos.

Clase `Updates`

La clase `Updates` se encarga de actualizar registros en la base de datos. Los métodos de esta clase permiten modificar la información de empleados y vehículos. Algunos de los métodos destacados incluyen:

- `contactoEmpleado(String id, String ubicacion, String email, String telefono)`: Actualiza la información de contacto de un empleado.
- `actualizarEmpleado(String idocurp, String nombre, String apellidoP, String apellidoM, String fechaN, String fechaI, String estado, String telefono, String mail, String ubi, String horaEntrada, String horaSalida, String horaComida)`: Actualiza los datos de un empleado en la base de datos.
- `queryUpdateInventario(int id, String tipo, String descripcion, int cantidad, String marca, String modelo, String num_serie, Double precio, String nombre)`: Actualiza los datos de un vehículo en el inventario, permitiendo modificar detalles como tipo, descripción, cantidad, marca, modelo, número de serie y precio.

Clase `Delets`

La clase `Delets` se encarga de eliminar registros de la base de datos. Sus métodos permiten borrar entradas de diversas tablas, asegurando que la información obsoleta o innecesaria sea retirada del sistema. Los métodos clave de esta clase incluyen:

- **`deleteEmpleado(String id)`**: Elimina un empleado de la base de datos mediante su ID.
- **`deleteAutomovil(String id)`**: Elimina un vehículo del inventario mediante su ID.
- **`deleteContactoEmpleado(String id)`**: Elimina los datos de contacto de un empleado según su ID.
- **`deleteCliente(String id)`**: Elimina a un cliente del sistema a través de su ID.

La clase maneja la eliminación de registros de manera eficiente, asegurándose de que la integridad de los datos sea mantenida durante el proceso.

Clase `Inserts`

La clase `Inserts` es responsable de insertar nuevos registros en las tablas de la base de datos. Permite agregar información relevante para los empleados, vehículos y otros aspectos del sistema, facilitando la expansión de los datos en el sistema. Algunos de los métodos más importantes de esta clase son:

- `insertEmpleado(String idocurp, String nombre, String apellidoP, String apellidoM, String fechaN, String fechaI, String estado, String telefono, String mail, String ubi, String horaEntrada, String horaSalida, String horaComida)`: Inserta un nuevo empleado en la base de datos con los detalles proporcionados.
- `insertAutomovil(String tipo, String descripcion, int cantidad, String marca, String modelo, String num_serie, Double precio, String nombre)`: Inserta un nuevo vehículo en el inventario.
- `insertCliente(String nombre, String telefono, String email)`: Inserta un nuevo cliente en el sistema con su nombre, teléfono y correo electrónico.
- `insertContactoEmpleado(String id, String ubicacion, String email, String telefono)`: Inserta los datos de contacto de un empleado en la base de datos.

Los métodos de la clase `Inserts` aseguran que los nuevos registros se inserten de manera correcta y eficiente.

Vista:

Interfaz Gráfica de Usuario

El sistema cuenta con una interfaz gráfica diseñada en **NetBeans** utilizando **JFrame** y **JDialog**, permitiendo una navegación intuitiva y organizada. A continuación, se detallan las principales secciones de la interfaz:

Pantalla de Inicio de Sesión

Al abrir el programa, se muestra una ventana donde los usuarios deben ingresar su **nombre de usuario** y **contraseña** para acceder al sistema. Dependiendo de los permisos del usuario, se habilitan diferentes módulos.

Módulos del Sistema

1. **Contabilidad**
    - Permite llevar un registro detallado de los ingresos y egresos del negocio.
    - Se pueden consultar y registrar **compras**, **ventas**, **deudas** y **gastos**.
2. **Punto de Venta**
    - Se encarga de la gestión de ventas.
    - Permite registrar la venta de vehículos a clientes y actualizar el inventario.
3. **Gestión de Empleados (CRUD)**
    - Se pueden **crear**, **leer**, **actualizar** y **eliminar** empleados.
    - Cada empleado tiene un perfil con datos personales, fecha de ingreso, horarios y estado dentro de la empresa.
4. **Gestión de Vehículos (CRUD)**
    - Permite administrar el inventario de automóviles.
    - Se pueden agregar nuevos vehículos, modificar información, eliminar registros o consultar detalles de los existentes.
5. **Administración de Usuarios**
    - El sistema permite la creación de **nuevas cuentas de usuario** con distintos niveles de acceso.
    - Solo los administradores pueden gestionar las cuentas de otros usuarios.

Clase Login.java

Esta clase es una clase que extiende la clase Jframe, aquí se tendra la ventana para que diferentes

usuarios puedan logearse y hacer uso de la aplicación.

Contiene un pequeño y básico formulario de introducir usuario y contraseña para iniciar sesión y un

boton el cuál mandara los datos del formulario como usuario y contraseña e iniciara sesión con ese rol en la base de datos, en caso de no encontrarse mandara una alerta.

Clase InicioFrame.java

Esta clase extiende de la clase Jframe, esta ventana solo la podran ver quienes esten logeados con el rol

de administrador lo cual les va a permitir navegar por toda la aplicación, podrán hacer cambios y

actualizaciones en la base de datos ya que tienen todos los permisos de aplicación.

Esta ventana contendra un botón en el cual se podra acceder a cada área de la aplicación.

Clase Principal.java

Esta clase que extiende de la clase JFrame contiene el espacio requerido para el punto de venta a la cuál

podra tener acceso el rol administrador y el rolo usuarioPV; usuarioPV sera el usuario que solo tendra

acceso a esta ventana y no tendra acceso a ninguna otra, en esta ventana el usuario podra hacer ventas

rellenando formularios, podra buscar productos por Modelo o Marca podra terminar su dia y cerrar

sesión, en esta ventana se mostrarán algunos accesos rápidos de productos mas comunes, asi como

también promociones.

Clase GestionEmpleados.java

Esta clase que extiende de la clase JFrame contiene un CRUD en el cuál el área de RRHH podra llevar

un control de los empleados en la empresa, en este espacio podrán consultar empleados, tendran un

botón para ir a dar de alta a algún empleado, para actualizar los datos de algún empleado y para

eliminar empleados en caso de que se requiera, además se tendra una barra buscadora para encontrar

mas rápido a algún empleado en especidfico apartir de su nombre o ID.

Clase RegisterEmpleado.java

Esta clase que extiende la clase JFrame contiene un formulario para dar de alta a algún empleado nuevo

el cuál se incorporara a la empresa rellenando ciertos campos que son importantes para llevcar un buen

control de los datos del empleado, la información recolectada por esta ventana se dividira en tres

secciones las cuales son:

Datos personales.•Información de empresa.

•Información para el puesto

Y dos botones los cuales son:

btnGuardar y btnDescartar; el botón btnGuardar guardara la información introducida en el formulario y

btnDescartar descartara lo introducido lo desechara y regresara a la ventana GestionEmpleados.java.

Clase Conta.java

Esta clase extiende de la clase JFrame y contiene lo necesario para llevar acabo la contabilidad de la

aplicación, a esta área tendra acceso el rol de contador y el de administrador; en esta ventana tendremos

botones para registrar y actualizar gastos y deudas, asi como también agregar algún proveedor y ver el

balance general de la aplicación.

Clase Inventario.java

Esta clase que extiende de la clase JFrame contendra un CRUD en el cuál podremos administrar el

inventario de todos los autos que se tienen, podriamos agregar nuevos vehiculos, consultar los que ya

tengo, actualizar datos de los que ya estan el la BD y eliminar.

También tendremos una barra buscadora la cuál nos servira para encontrar más rápido algún auto

apartir de su nombre, modelo o ID.

Package icons e icons.images

En este paquete estarán contenidos todos los iconos y png’s necesarios para el funcionamiento e

interacción de la aplicación.

Interacción con la Base de Datos

Cada módulo se conecta con una base de datos PostgreSQL, permitiendo el almacenamiento y recuperación de información en tiempo real. La comunicación con la base de datos se realiza mediante consultas SQL desde las clases del paquete **Modelo**.

Controlador:

El Controlador en el programa se encarga de gestionar la lógica de interacción entre la interfaz de usuario y los datos. Actúa como intermediario, asegurando que las acciones realizadas por el usuario, como actualizar información o cargar datos, sean procesadas correctamente y reflejadas en la interfaz.

Gestión de la hora y fecha: A través de la clase Hora, el controlador mantiene actualizada la hora y la fecha en la interfaz de usuario, mostrando la información en tiempo real y realizando un seguimiento constante para garantizar que la hora no se detenga, actualizando la fecha cuando es necesario.

Carga y visualización de imágenes: Con la clase SavePictures, el controlador maneja la carga de imágenes de perfil de los empleados. Si una foto está disponible, se muestra en la interfaz, y si no, se muestra una imagen por defecto.

Actualización de registros en la base de datos: La clase Updates gestiona la actualización de los datos en la base de datos, asegurando que la información sobre empleados, vehículos y contactos se mantenga actualizada según las acciones del usuario en la interfaz.

Clase Conexion.java

Contiene métodos para la conexión y cierre a una BD postgreSQL.

Método conectaagency():

Este método levanta una conexión a una base de datos postgreSQL.

Método closeBd():

Este método cierra la conexión abierta a la base de datos postgreSQL.

Clase Delets.java

Contiene métodos que hacen peticiones de eliminación de registros a la BD postgreSQL.

Método eliminarEmpleado(String id):

Este método recibira un ID y el nombre de alguna tabla y hara una consulta de eliminación a la BD.

Clase Inserts.java

En esta clase tendremos métodos que insertaran contenido a la BD.

Método queryInsertinventario(String tabla,String tipo, String descripcion, int cantidad,String marca,String modelo,String num_serie,Double precio, String nombre):

Este método recibe los datos necesario e inserta un nuevo registro a la tabla automovil en la BD.

querysInsertEmpleados(String idocurp, String nombre, String apellidoP, String apellidoM, String fechaN,String fechaI, String estado, String telefono,String mail, int idarea, String ubi, String horaEntrada,String horaSalida,String horaComida, int idpuesto):

Este método recibe los datos necesarios para insertar o actualizar un nuevo empleado a la BD.

Clase Querys.java

Esta clase solo tendra métodos que consultaran contenido en la BD.

Método queryInventary():

Este método consulta todo lo que hay en la tabla automovil de la BD.*

Método queryUpdateI(int id):

Este método consulta todo lo que contiene la tabla automovil de la BD.Método llenaActualizarE(String id):

Consulta todos los datos de la tabla empleados*.

Método buscaIdEm(String id):

Este método consulta algunas coincidencias según lo que se inserte en las barras buscadoras.

Método consultaEmpleados():

Este método consulta algunos datos reelevantes para mostrar en la tabla de empleados.

Clase Updates.java

Esta clase tendra métodos para actualizar tablas en la BD.

Conclusiones

En el desarrollo de este sistema de gestión de inventarios, ventas y empleados, se ha logrado implementar una estructura eficiente que permite realizar las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) para diferentes entidades, como vehículos, empleados y clientes. A través de una interfaz gráfica intuitiva, los usuarios pueden interactuar fácilmente con el sistema, garantizando una experiencia fluida y accesible.

El uso de patrones de diseño, como el Modelo-Vista-Controlador (MVC), ha sido clave para separar las responsabilidades en diferentes capas, lo que facilita la escalabilidad y el mantenimiento del sistema. El modelo maneja la lógica de negocio y las interacciones con la base de datos, mientras que el controlador asegura que la información fluya correctamente entre la vista y el modelo, gestionando la actualización de los datos y la interfaz de usuario en tiempo real.

Además, la implementación de funcionalidades como la actualización automática de la hora y fecha, la carga de imágenes de perfil y la correcta gestión de registros de empleados y vehículos han permitido mejorar la eficiencia del sistema y la experiencia del usuario.

Este proyecto ha sido un desafío que ha permitido aplicar conceptos de programación orientada a objetos, manejo de bases de datos y diseño de interfaces gráficas. Ha demostrado la importancia de mantener un código limpio y modular para poder realizar modificaciones y mejoras fácilmente en el futuro. Con estos conocimientos, el sistema puede seguir evolucionando para adaptarse a nuevas necesidades o requerimientos.