UNIVERSIDAD MARIANO GALVEZ

SEDE: MAZATENANGO

CARRERA: INGIENRIA EN SISTEMAS

CURSO: programación III

CATEDRATICO: ING. Miguel Lemus

CICLO: V

SEMESTRE: 1ro.

Manual Tenico.

NOMBRE: HEBER ABIMAEL SIMON GREORIO

CARNE: 3090-23-6405

FECHA DE ENTREGA:15/06/205

Manual Técnico del Sistema de Gestión Vehicular

Introducción

Este manual técnico proporciona una guía detallada sobre la arquitectura, funcionamiento y uso del Sistema de Gestión Vehicular, una aplicación diseñada para la administración eficiente de información relacionada con vehículos, multas y traspasos. El sistema emplea estructuras de datos avanzadas como Árboles Binarios de Búsqueda (ABB) y Árboles AVL para optimizar el almacenamiento y la recuperación de datos vehiculares, así como listas doblemente enlazadas y listas circulares para la gestión de multas y traspasos, respectivamente.

El objetivo de este manual es servir como un recurso completo para técnicos, desarrolladores y administradores del sistema, facilitando la comprensión de su lógica interna, la resolución de problemas y la extensión de sus funcionalidades. Se abordarán los componentes clave, los flujos de interacción del usuario y las consideraciones para la manipulación de datos.

Este proyecto ha sido desarrollado en el entorno de desarrollo integrado (IDE) Apache NetBeans, haciendo uso extensivo de sus funcionalidades para la creación de interfaces gráficas de usuario (GUI). La implementación se basa en la creación de JFrames para las ventanas principales y secundarias de la aplicación.

Componentes de la Interfaz de Usuario (GUI)

Para la visualización y gestión de datos, se emplean JTables, permitiendo presentar la información de manera estructurada y tabular. La interacción del usuario se facilita mediante diversos componentes Swing, tales como:

- **JComboBox: Para la selección de opciones predefinidas, como departamentos.**
- **JLabel: Para la visualización de texto estático y etiquetas informativas.**
- **JTextField: Para la entrada de texto por parte del usuario, como placas de vehículos, DPIs, nombres, etc.**
- **JButton: Para disparar acciones y eventos, como la carga de archivos, la búsqueda, modificación, eliminación y visualización de datos.**

Estructuras de Datos

El sistema incorpora estructuras de datos avanzadas para la manipulación eficiente de la información:

1. Árboles Binarios de Búsqueda (ABB)

- **Descripción: Utilizados para la gestión de vehículos, permitiendo operaciones de inserción, búsqueda y eliminación con un rendimiento optimizado.**
- **Implementación: Clases ABB.java y NodoABB.java.**
- **Operaciones:**
    - **Inserción de vehículos.**
    - **Búsqueda de vehículos por placa.**
    - **Eliminación de vehículos.**
    - **Modificación.**

2. Árboles AVL

- **Descripción: Una variante auto-balanceada de los árboles binarios de búsqueda, que garantiza un rendimiento logarítmico para las operaciones, incluso en el peor de los casos.**
- **Implementación: Clases AVL.java y NodoAVL.java.**
- **Operaciones:**
    - **Inserción de vehículos.**
    - **Búsqueda de vehículos por placa.**
    - **Eliminación de vehículos.**
    - **Modificación de vehículos.**

3. Listas Doblemente Enlazadas

- **Descripción: Empleadas para la administración de multas, facilitando la inserción, búsqueda, modificación y eliminación de registros de forma eficiente.**
- **Implementación: Clase ListaDoblementeEnlazadaMultas.java.**
- **Operaciones:**
    - **Agregar, modificar y eliminar multas.**
    - **Búsqueda de multas por ID.**

4. Listas Circulares

- **Descripción: Utilizadas para la gestión de traspasos de vehículos, permitiendo recorrer la lista de manera continua y eficiente.**
- **Implementación: Clase ListaTraspasos.java.**
- **Operaciones:**
    - **Agregar, modificar y eliminar traspasos.**
    - **Búsqueda de traspasos por ID.**

Uso del Sistema

3.1. Inicio de Sesión (Login)

La aplicación inicia con una ventana de autenticación para garantizar el acceso seguro al sistema.

- **Clase Responsable: Loguear.java**
- **Interfaz de Usuario:**
    - **Campos de texto para "Usuario" y "Contraseña".**
    - **Un botón "Ingresar" para validar las credenciales.**

Flujo de Uso:

1. **Al ejecutar la aplicación, se mostrará la ventana de Loguear.**
2. **El usuario debe ingresar sus credenciales.**
3. **Al hacer clic en "Ingresar", el sistema verificará el usuario y contraseña (actualmente "Admin" y "12345" como se infiere del código).**
4. **Si las credenciales son correctas, la ventana principal (Principal.java) se hará visible y la ventana de login se cerrará.**
5. **Si las credenciales son incorrectas, se mostrará un mensaje de error.**

3.2. Módulo Principal (Gestión de Vehículos - ABB y AVL)

Este es el corazón del sistema donde se gestiona la información de los vehículos utilizando dos estructuras de datos principales: un Árbol Binario de Búsqueda (ABB) y un Árbol AVL.

- **Clase Responsable: Principal.java**
- **Interfaz de Usuario:**
    - **Pestañas/Secciones: Para interactuar con el ABB y el AVL de manera separada o conjunta.**
    - **Carga de Archivos: Un botón para cargar datos de vehículos desde archivos CSV.**
    - **Entrada de Datos: Campos de texto para ingresar los detalles de un nuevo vehículo (Placa, DPI, Nombre, Marca, Modelo, Año, Multas, Traspasos).**
    - **Botones de Acción: "Agregar", "Buscar", "Modificar", "Eliminar" para cada estructura de árbol.**
    - **Visualización de Árboles: Botones para generar representaciones gráficas de los árboles (usando Graphviz).**
    - **Tablas de Datos: Para mostrar los vehículos cargados o resultados de búsquedas.**
    - **Dropdown/ComboBox de Departamento: Para filtrar o seleccionar un departamento al cargar archivos o realizar operaciones.**
    - **Área de Mensajes: Para mostrar retroalimentación al usuario, incluyendo tiempos de ejecución.**

Flujo de Uso:

3.2.1. Carga de Datos

1. **Clic en "Cargar":**
    - **Se abrirá un JFileChooser para que el usuario navegue y seleccione el archivo CSV de vehículos.**
    - **El sistema leerá el archivo y poblará tanto el arbolABB como el arbolAVL con la información de los vehículos.**
    - **Los datos se mostrarán en la JTable correspondiente.**
    - **Se mostrará el tiempo de carga en el área de mensajes.**

3.2.2. Agregar Vehículo

1. **Ingresar Datos: Llenar los campos de texto con la información del nuevo vehículo.**
2. **Clic en "Agregar": El vehículo será agregado tanto al ABB como al AVL, y se actualizarán las tablas y el área de mensajes.**

3.2.3. Buscar Vehículo

1. **Ingresar Placa: Introducir la placa del vehículo en el campo de búsqueda correspondiente.**
2. **Clic en "Buscar": El sistema buscará el vehículo en la estructura de datos seleccionada (ABB o AVL) y mostrará sus detalles.**

3.2.4. Modificar Vehículo

1. **Ingresar Placa: Ingresar la placa del vehículo a modificar.**
2. **Clic en "Modificar": El sistema buscará el vehículo. Si lo encuentra, se habilitarán los campos para que el usuario edite la información.**

3.2.5. Eliminar Vehículo

1. **Ingresar ID/Placa: Introducir el ID o la placa del vehículo a eliminar.**
2. **Clic en "Eliminar": El sistema eliminará el vehículo de la estructura de datos correspondiente y actualizará las tablas.**

3.2.6. Visualización Gráfica de Árboles (ABB y AVL)

1. **Clic en "Ver ABB":**
    - **El sistema generará un archivo .dot que describe la estructura actual del arbolABB.**
    - **Este archivo se pasará a Graphviz para generar una imagen que representará el árbol.**
2. **Clic en "Visualizar AVL":**
    - **El proceso es idéntico al de la visualización del ABB, pero se aplica al arbolAVL.**

3.3. Módulo de Multas

Este módulo gestiona la información de las multas asociadas a los vehículos, utilizando una lista doblemente enlazada para su almacenamiento y manipulación.

- **Clase Responsable: Multas.java y ListaDoblementeEnlazadaMultas.java**
- **Interfaz de Usuario:**

- **Carga de Archivos: Botón "Cargar Multas" para importar datos desde archivos CSV.**
- **Tabla de Multas: Para mostrar las multas.**
- **Campos de Búsqueda/Eliminación/Modificación: Para gestionar multas.**
- **Botón "Crear": Para ingresar nuevas multas.**

Flujo de Uso:

3.3.1. Carga de Multas

1. **Clic en "Cargar Multas":**
    - **Se abrirá un JFileChooser para que el usuario seleccione la carpeta general de departamentos.**
    - **El sistema recorrerá las subcarpetas buscando archivos *_multas.txt.**
    - **Los datos de las multas se cargarán en la lista de multas y se actualizará la tabla.**

3.3.2. Gestión de Multas

- **Agregar: Clic en "Crear" para ingresar los detalles de la nueva multa.**
- **Buscar: Ingresar el ID de la multa y clic en "Buscador".**
- **Modificar: Ingresar el ID de la multa y clic en "Modificar".**
- **Eliminar: Ingresar el ID de la multa y clic en "Eliminador".**

3.4. Módulo de Traspasos

Este módulo se encarga de la gestión de los traspasos de vehículos, utilizando una lista circular para su implementación.

- **Clase Responsable: Traspasos.java y ListaTraspasos.java**
- **Interfaz de Usuario:**
    - **Carga de Archivos: Botón "Cargar Traspasos" para importar datos.**
    - **Tabla de Traspasos: Para mostrar los traspasos.**
    - **Campos de Búsqueda/Eliminación/Modificación: Para gestionar traspasos.**

Flujo de Uso:

3.4.1. Carga de Traspasos

1. **Clic en "Cargar Traspasos":**
    - **Similar a las multas, se solicitará la carpeta de departamentos.**

- **El sistema buscará los archivos *_traspasos.txt en cada subcarpeta.**

3.4.2. Gestión de Traspasos

- **Agregar: Clic en "Ingreso" para introducir los datos del nuevo traspaso.**
- **Buscar: Ingresar el ID de la multa y clic en "Buscador".**
- **Modificar: Ingresar el ID de la multa y clic en "Modificar".**
- **Eliminar: Ingresar el ID de la multa y clic en "Eliminador".**

3.5. Resumen General del Sistema

Este módulo presenta una visión consolidada de los datos del sistema, mostrando estadísticas y resúmenes de los vehículos, multas y traspasos cargados.

- **Clase Responsable: Resumen.java**
- **Interfaz de Usuario:**
    - **Etiquetas y áreas de texto para mostrar información como:**
        - **Número total de vehículos.**
        - **Número total de multas.**
        - **Número total de traspasos.**
        - **Vehículo con más multas.**
        - **Departamento con más vehículos/multas/traspasos.**

Flujo de Uso:

1. **Desde la ventana principal, el usuario seleccionaría la opción para ver el "Resumen General".**
2. **El sistema recopilaría la información de las estructuras de datos y presentaría los datos calculados.**

Recorridos en Árboles Binarios de Búsqueda (ABB) y Árboles AVL

Los árboles ABB y AVL utilizan recorridos específicos para visitar y procesar los nodos de manera ordenada. Estos recorridos son fundamentales para operaciones como impresión, búsqueda, o generación de listas ordenadas.

1. Recorrido InOrden (Inorder)

- **Definición: Visita primero el subárbol izquierdo, luego el nodo actual, y finalmente el subárbol derecho.**
- **Resultado: Devuelve los elementos ordenados ascendentemente según la clave (placa del vehículo).**

- **Uso típico: Obtener una lista ordenada de todos los vehículos.**

2. Recorrido PreOrden (Preorder)

- **Definición: Visita el nodo actual primero, luego el subárbol izquierdo, y finalmente el subárbol derecho.**
- **Resultado: Útil para clonar el árbol o guardar su estructura.**
- **Uso típico: Guardar o visualizar la estructura de los árboles ABB o AVL.**

3. Recorrido PostOrden (Postorder)

- **Definición: Visita primero el subárbol izquierdo, luego el subárbol derecho, y finalmente el nodo actual.**
- **Resultado: Útil para eliminar el árbol o realizar operaciones que requieren post-procesamiento del nodo.**
- **Uso típico: Limpieza o liberación de recursos asociados al árbol.**

Implementación en tu sistema

- **ABB: Métodos implementados**
    - **obtenerVehiculosOrdenados() — Recorrido InOrden**
    - **obtenerVehiculosPreOrden() — Recorrido PreOrden**
    - **obtenerVehiculosPosOrden() — Recorrido PostOrden**
- **AVL: Métodos implementados**
    - **obtenerTodosLosVehiculos() — Recorrido InOrden**
    - **obtenerVehiculosPreOrden() — Recorrido PreOrden**
    - **obtenerVehiculosPosOrden() — Recorrido PostOrden**

Cada método devuelve una lista de vehículos (List<Vehiculo>) en el orden correspondiente, que luego se utiliza para mostrar los datos en la tabla de interfaz.

Encriptación y Desencriptación en el Sistema

Para garantizar la confidencialidad de los datos sensibles (como placas, DPI, nombres, marcas, modelos, y departamento), se utiliza un sistema de encriptación tipo Cifrado César con desplazamiento fijo.

Método de Encriptación

- **Descripción: Desplaza cada letra y dígito un número fijo de posiciones en el alfabeto y en la secuencia numérica respectivamente.**
- **Características:**

- **Letras se transforman preservando mayúsculas y minúsculas.**
- **Números se desplazan dentro del rango 0-9.**
- **Otros caracteres no se modifican.**

Código simplificado:

java13 lines

Click to close

private String encriptar(String texto, int desplazamiento) {

StringBuilder resultado = new StringBuilder();

...

Método de Desencriptación

- **Descripción: Aplica el desplazamiento inverso para recuperar el texto original.**
- **Implementación: Se usa la propiedad que desencriptar es equivalente a encriptar con desplazamiento inverso.**

Código simplificado:

java3 lines

Click to expand

private String desencriptar(String texto, int desplazamiento) {

return encriptar(texto, 10 - desplazamiento); // Desplazamiento inverso para números

...

Uso

- **Aplicación: Se aplica al momento de almacenar datos para proteger la información.**
- **Interacción: Los botones "Encriptar" y "Desencriptar" en la interfaz aplican estos métodos a las tablas de datos.**

Resumen del Sistema

El **Sistema de Gestión Vehicular** es una herramienta integral diseñada para facilitar la administración de información relacionada con vehículos, multas y traspasos. A través de la implementación de estructuras de datos avanzadas como **Árboles Binarios de Búsqueda (ABB)** y **Árboles AVL**, así como listas doblemente enlazadas y listas circulares, el sistema optimiza el almacenamiento y la recuperación de datos, garantizando un rendimiento eficiente en operaciones críticas.

La interfaz gráfica de usuario (GUI) proporciona una experiencia intuitiva, permitiendo a los usuarios interactuar fácilmente con el sistema. Las funcionalidades de carga, búsqueda, modificación y eliminación de datos, junto con la capacidad de visualizar estructuras de datos de manera gráfica, hacen que el sistema sea versátil y fácil de usar.

Impacto en la Gestión Vehicular

Este sistema no solo mejora la eficiencia en la gestión de datos vehiculares, sino que también contribuye a la transparencia y la seguridad de la información. La implementación de encriptación para datos sensibles asegura que la información personal de los usuarios esté protegida, lo que es fundamental en un entorno donde la privacidad es una preocupación creciente.

Recomendaciones para el Uso y Mantenimiento

1. **Capacitación de Usuarios**: Se recomienda proporcionar capacitación a los usuarios del sistema para maximizar su efectividad. Esto incluye familiarizarse con la interfaz y las funcionalidades disponibles.
2. **Mantenimiento Regular**: Realizar un mantenimiento regular del sistema, incluyendo actualizaciones de software y revisión de la integridad de los datos, es crucial para asegurar un rendimiento óptimo.
3. **Respaldo de Datos**: Implementar un sistema de respaldo de datos para prevenir la pérdida de información crítica. Esto puede incluir copias de seguridad periódicas de los archivos de datos y la base de datos.
4. **Monitoreo de Rendimiento**: Evaluar el rendimiento del sistema de manera continua para identificar áreas de mejora. Esto puede incluir la optimización de algoritmos de búsqueda y la revisión de la estructura de datos utilizada.
5. **Feedback de Usuarios**: Fomentar la retroalimentación de los usuarios para identificar problemas y oportunidades de mejora en la interfaz y las funcionalidades del sistema.