



Escuela Politécnica Nacional
Facultad de Ingeniería en Sistemas



MANUAL TÉCNICO DEL SISTEMA DE REGISTRO DE MATRICULACIÓN VEHICULAR

Autores:

Matías Lamiña

Sebastián Arévalo

Rodrigo Montero

Mateo Guamaní

Contenido

| | |
|---|----|
| PRESENTACIÓN | 3 |
| OBJETIVO | 4 |
| FINALIDAD DEL MANUAL | 4 |
| 1. INTRODUCCIÓN | 5 |
| 1.1 Propósito del Documento | 5 |
| 1.2 Alcance del Sistema | 5 |
| 1.3 Audiencia Objetivo | 5 |
| 2. ASPECTOS TÉCNICOS | 6 |
| 2.1. HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO | 6 |
| 2.1.1 Zinjal | 6 |
| 2.1.2 GitHub | 6 |
| 3. ARQUITECTURA DEL SISTEMA | 7 |
| 3.1 Patrón de Diseño | 7 |
| 3.1.1 Componentes Principales | 7 |
| 3.1.2 Diagramas de flujo | 8 |
| 4. ESPECIFICACIONES TÉCNICAS | 9 |
| 4.1 Requisitos del Sistema | 9 |
| 4.2 Limitaciones del Sistema | 9 |
| 4.3 MODIFICACIÓN LOCAL | 9 |
| 5. ESTRUCTURA DE DATOS | 13 |
| 5.1 Módulos y funciones | 14 |
| BIBLIOGRAFÍA | 17 |

PRESENTACIÓN

El siguiente manual se ha desarrollado con la finalidad de dar a conocer la información necesaria para realizar mantenimiento, instalación y exploración del Programa de Registro de Matriculación Vehicular, el cual consta de diferentes actividades para el mejoramiento de los procesos de la institución encargada del registro vehicular.

El manual ofrece la información necesaria del desarrollo del programa para que la persona que esté a cargo y desee modificar el programa lo haga de una manera apropiada, dando a conocer la estructura del desarrollo del aplicativo.

OBJETIVO

Dar a conocer el uso adecuado del programa en aspectos técnicos de manera descriptiva e ilustrada sobre los componentes y funcionalidades que conforman el buen funcionamiento del sistema de información.

FINALIDAD DEL MANUAL

La finalidad de este manual técnico es instruir a la persona que quiera administrar, editar o configurar el programa de Registro de Matriculación Vehicular usando las debidas herramientas.

1. INTRODUCCIÓN

1.1 Propósito del Documento

Este manual técnico describe la arquitectura, implementación y funcionamiento interno del Sistema de Matriculación Vehicular, proporcionando la información necesaria para desarrolladores, mantenedores y administradores del sistema.

1.2 Alcance del Sistema

El sistema permite:

- Gestión de administradores con autenticación
- Registro completo de vehículos y propietarios
- Cálculo automático de tarifas e impuestos
- Generación de comprobantes oficiales
- Búsqueda y consulta de registros
- Persistencia de datos en archivos de texto

1.3 Audiencia Objetivo

- Desarrolladores: Para mantenimiento y evolución del código
- -Administradores de Sistema: Para configuración y operación
- Testers: Para pruebas y validación
- Auditores: Para revisión de cumplimiento



2. ASPECTOS TÉCNICOS

El aplicativo tiene la finalidad para el mejoramiento de los procesos de la entidad rectora del transporte y la matriculación. Se recomienda que el siguiente manual sea manipulado únicamente por la persona que quiera administrar, editar o configurar el programa para velar por la seguridad de los datos que se almacenan de manera segura.

2.1. HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO

Para el desarrollo de este programa se decidió utilizar las siguientes herramientas.

2.1.1 Zinjal

Zinjal es un Entorno de Desarrollo Integrado (IDE) ligero y multiplataforma, diseñado principalmente para la programación en **C/C++**. Se caracteriza por ser una herramienta amigable para estudiantes de programación, pero lo suficientemente potente para proyectos complejos. Ofrece una interfaz intuitiva con funcionalidades esenciales como el coloreado de sintaxis, el plegado de código, búsqueda y reemplazo avanzados (incluyendo expresiones regulares), y comandos específicos para la edición en C++.

2.1.2 GitHub

El desarrollo del Sistema de Matriculación Vehicular se llevó a cabo utilizando **metodologías ágiles de desarrollo colaborativo** mediante la plataforma GitHub. Este enfoque permitió la implementación de un **flujo de trabajo distribuido** donde múltiples desarrolladores contribuyeron simultáneamente al proyecto, manteniendo la **integridad del código fuente** y la **trazabilidad de cambios** a través de un sistema de control de versiones robusto.

3. ARQUITECTURA DEL SISTEMA

| CAPA DE PRESENTACIÓN | |
|----------------------|---------------------|
| (menuprincipal.c) | Interfaz de Usuario |
| (main.cpp) | Menú Principal |

| CAPA DE LÓGICA | |
|------------------|-----------------------|
| Validaciones | Cálculos de Matrícula |
| (libfunciones.c) | (main.cpp) |

| CAPA DE DATOS | |
|----------------------|-----------------------|
| Gestión Admin | Cálculos de Matrícula |
| (login_de_usuario.c) | (.txt files) |

3.1 Patrón de Diseño

El sistema implementa un **patrón de diseño modular** que divide las responsabilidades en componentes especializados y cohesivos. Esta aproximación arquitectónica permite que cada módulo tenga una **responsabilidad única y bien definida**, siguiendo el principio de **Single Responsibility Principle (SRP)**.

3.1.1 Componentes Principales

| Componente | Archivo | Funcionalidad |
|--------------------|----------------------|---------------------------------------|
| Interfaz Principal | main.cpp | Coordinación general, flujo principal |
| Menú de Usuario | menuprincipal.c | Presentación e interacción |
| Validaciones | libfunciones.c/.h | Validación de datos de entrada |
| Autenticación | `login_de_usuario.c` | Gestión de administradores |

3.1.2 Diagramas de flujo.

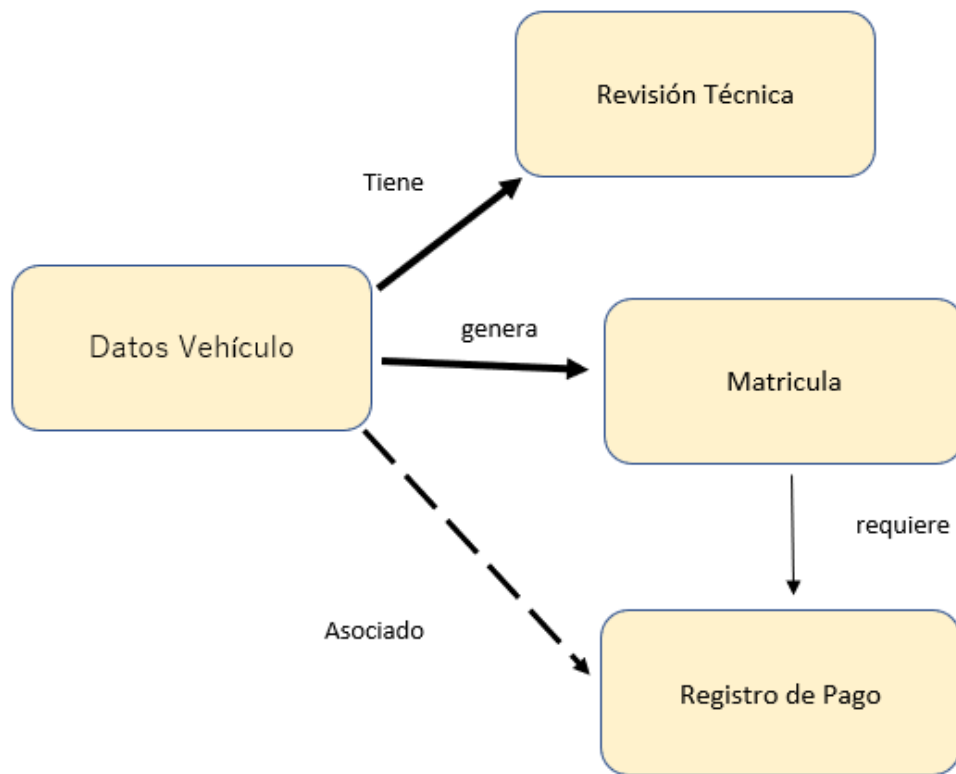


Figura 1.1 Diagrama de Estructuras de Datos

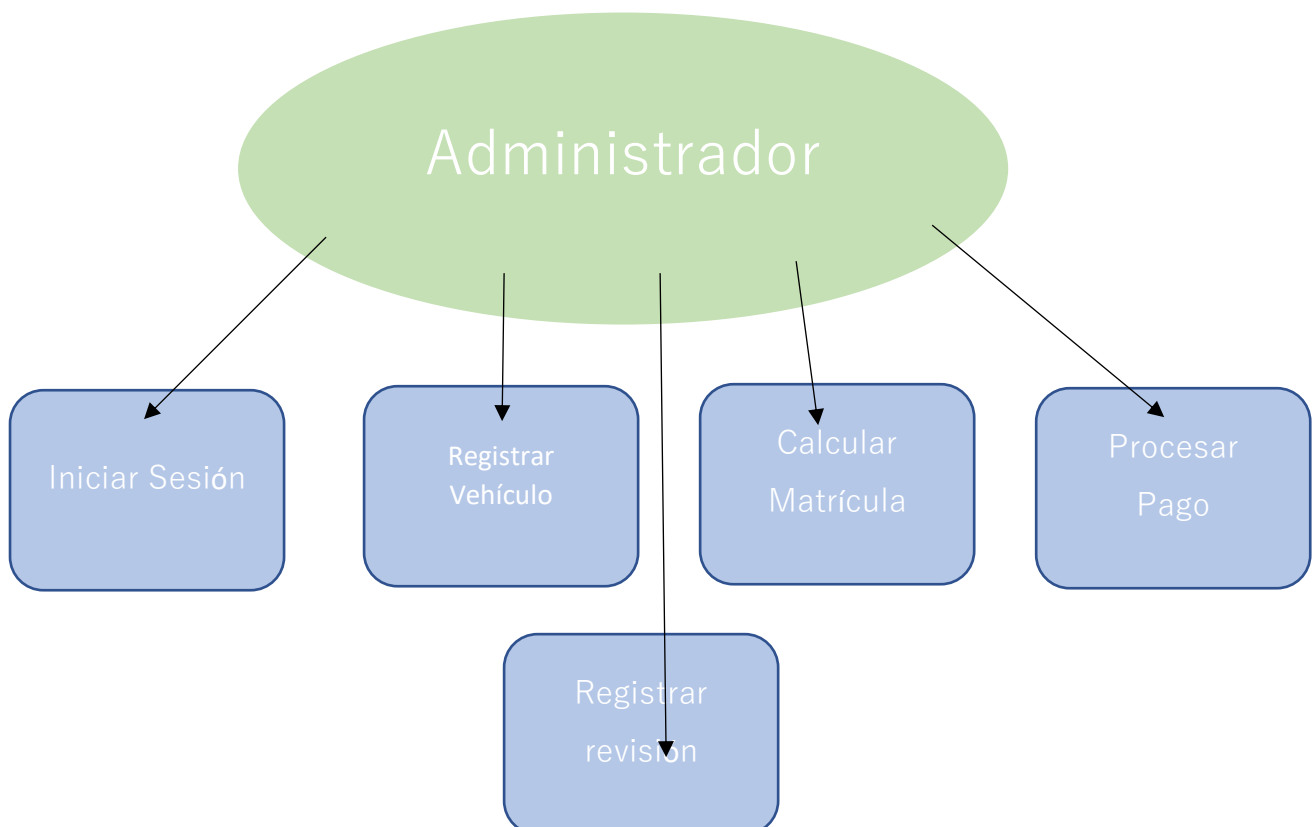


Figura 1.2 Diagrama de usos del sistema

4. ESPECIFICACIONES TÉCNICAS

Aquí se encuentran los requisitos para el uso del programa, recordando como es su instalación, requisitos para su funcionamiento, además de sus limitaciones.

4.1 Requisitos del Sistema

Hardware Mínimo:

- Procesador: Intel Pentium 4 o equivalente (1GHz)
- Memoria RAM: 512 MB
- Almacenamiento: 50 MB libres
- Sistema Operativo: Windows 8 o superior

Software Requerido:

Compilador: Zinjal

4.2 Limitaciones del Sistema

| Característica | Valor/Rango | Definición/Validación |
|---------------------|-----------------|----------------------------|
| Registros máximos | 135 vehículos | Definido por MAX_REGISTROS |
| Longitud nombre | 50 caracteres | Buffer de la estructura |
| Longitud contraseña | 6-20 caracteres | Validación de seguridad |
| Rango avalúo | \$1 - \$900,000 | Validación de negocio |
| Rango cilindraje | 1 - 110,000 cc | Límites técnicos |
| Años válidos | 1900 - 2025 | Validación temporal |

4.3 MODIFICACIÓN LOCAL

Si el desarrollador quiere realizar modificaciones del programa de manera local, deberá tener instalado Zinjal, con dicha herramienta podrá iniciar el programa.

Para lo cual debe de dirigirse al sitio:

<https://zinjai.sourceforge.net/index.php?page=descargas.php>

Figura 2: Sitio web de Zinjal



Descargar programa:



Contenido
Enlaces descarga |
Instalación: Windows |
Instalación: GNU/Linux |
Instalación: Mac OS X |
Software adicional |
Código fuente |
Versiones de prueba |
Detalles de los archivos |

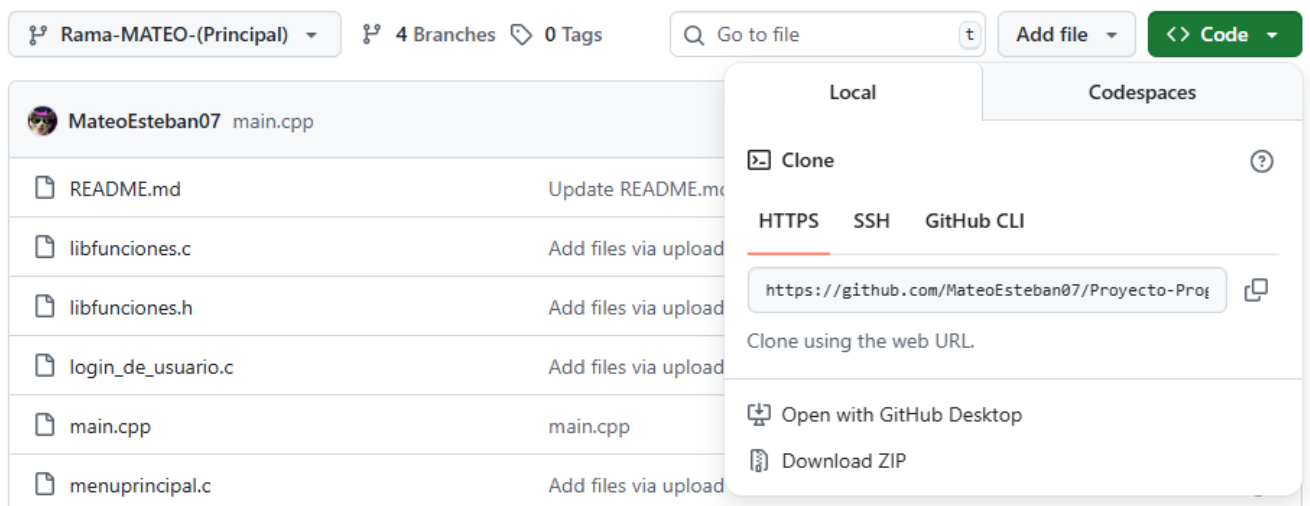
Fuente por Autores

Debe elegir de acuerdo al sistema operativo que use. Una vez haya instalado correctamente Zinjal en su ordenador procederá a dirigirse al repositorio de GitHub.


<https://github.com/MateoEsteban07/Proyecto-Programacion-1>

Aquí encontrara los archivos necesarios para el funcionamiento del programa:

Figura 3: Repositorio del programa en GitHub



Fuente por Autores

Presione en  Download ZIP se descargará un archivo el cual deberá de descomprimir.

Posteriormente diríjase a Zinjal y ábralo.

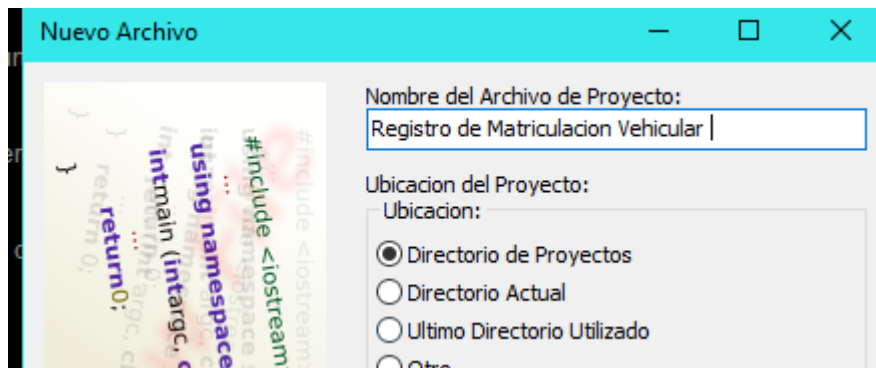
Figura 4: Zinjal



Fuente por Autores

Seleccione la opción crear un nuevo proyecto.

Figura 5: Zinjal, ventana de crear proyecto



Fuente por Autores

Coloque un nombre.

Figura 6: Zinjal subir archivos del programa

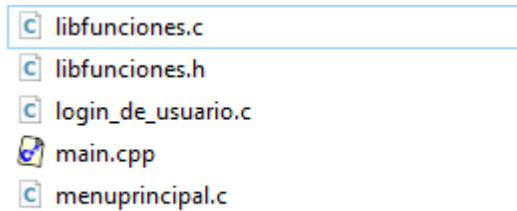


Fuente por Autores

Le aparecerá esta venta, seleccione la opción señalada

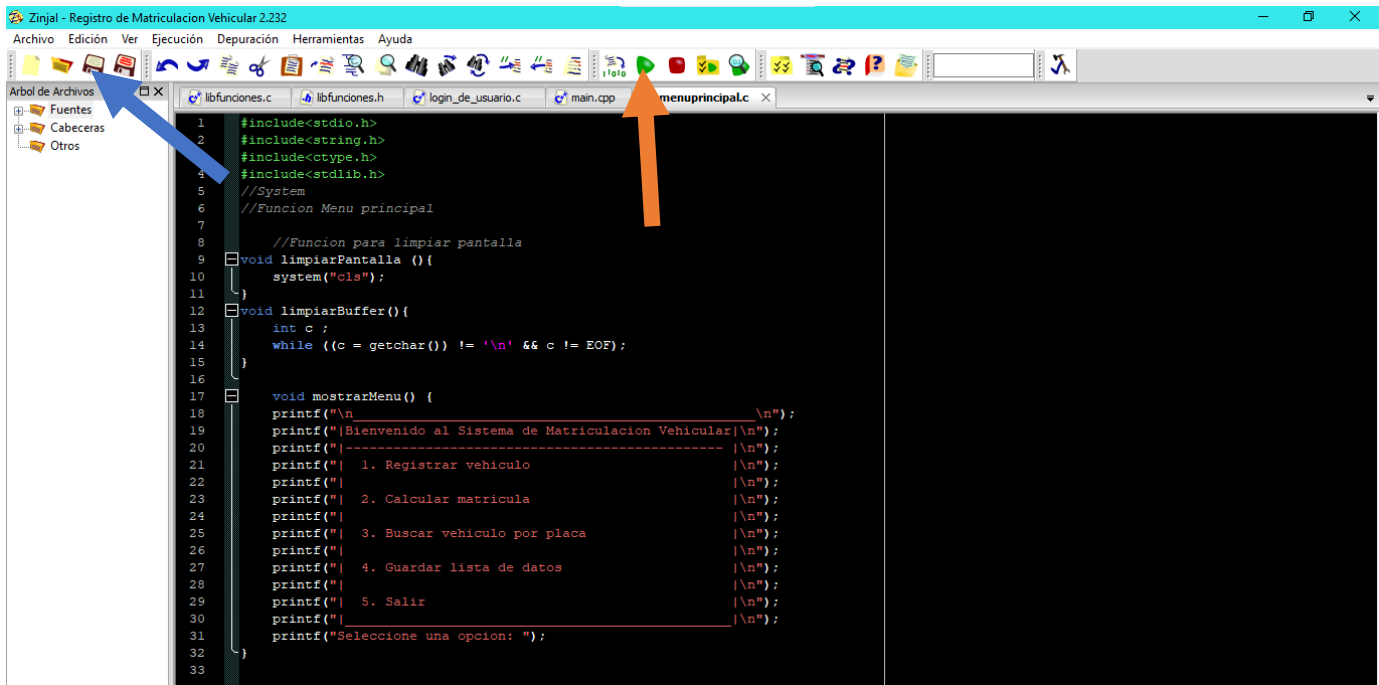
Se mostrará la ventana de seleccionar archivos diríjase a la carpeta donde estén los archivos y seleccione.

Figura 7: Zinjal



Fuente por Autores

Figura 8: Zinjal, Programa instalado.



Fuente por Autores

Para que empiece a funcionar asegúrese de guardar el proyecto con la opción señalada con azul (hacer esto por cada archivo).

Y para que funcione presione el botón señalado con naranja.

Figura 9: Programa de Registro de Matriculación Vehicular

```
-----  
Sistema de Administracion  
-----  
0. Salir  
1. Registrar administrador  
2. Iniciar sesion como administrador  
-----  
Seleccione una opcion: _
```

Fuente por Autores

Y listo esta completada la instalación, recuerde que todo archivo generado por el programa se encontrara en la misma carpeta del proyecto, además de que una vez lo cierre y desee abrirlo otra vez no tiene que repetir este proceso, solamente abra Zinjal y en la parte de abajo encontrara el programa.

5. ESTRUCTURA DE DATOS

Figura 10. Estructura Principal: Matrícula

```
typedef struct {  
    char nombre[50];           // Nombre del propietario  
    char placa[12];            // Placa vehicular (formato ABC1234)  
    char anio[10];             // Año de fabricación (como string)  
    char cedula[20];           // Cédula del propietario (10 dígitos)  
    float avaluo;              // Valor comercial del vehículo  
    int cilindraje;            // Cilindraje en centímetros cúbicos  
    int tipo;                  // 0=Eléctrico, 1=Combustible  
    int peso;                  // 0=Liviano, 1=Pesado  
    int revisionesTecnicas;    // Número de revisiones anuales  
} matricula;
```

Fuente por Autores

Figura 11. Estructura de Administrador

```
typedef struct {  
    char nombre_del_registrador[50]; // Nombre del administrador  
    char contrasenia[20];             // Contraseña (mínimo 6 caracteres)  
} PQR; // Persona Que Registra
```

Fuente por Autores

Figura 12. Variables Globales

```
// Variables de control del sistema
int opcion;           // Opción seleccionada en menús
int peso_v;           // Variable temporal para peso
int anioint;          // Año convertido a entero
float valor_a_pagar;   // Total calculado
float impverde;        // Impuesto verde calculado

// Constantes del sistema
#define MAX_REGISTROS 135 // Límite de registros
```

Fuente por Autores

5.1 Módulos y funciones

Funciones

- **'int validarNumeros(const char *cadena)'**: Verificar que una cadena contenga solo dígitos y `cadena` - String a validar además de retornar 1 si es válida, 0 si contiene caracteres no numéricos y $O(n)$ donde n es la longitud de la cadena.

- **validarOracion()**

Verificar que una cadena contenga solo letras y espacios y `oracion` - String para validar además de retornar 1 si es válida, 0 si contiene caracteres inválidos su uso es para validación de nombres de personas.

- **esPlacaValida()**

Validar formato de placa vehicular el formato es ABC1234 (3 letras + 4 números)

Validaciones:

- Longitud exacta de 7 caracteres
- Primeros 3 caracteres: letras mayúsculas
- Últimos 4 caracteres: dígitos numéricos

Retorno: 1 si es válida, 0 si no cumple el formato

- **(menuprincipal.c)**

limpiarPantalla()

Su propósito es limpiar la consola y ejecuta comando `system("cls")`

limpiarBuffer()

Su propósito es limpiar el buffer de entrada y su uso es para prevenir problemas con ``scanf()`` y ``fgets()``

- **limpiarEspaciosExtra()**.-Su propósito es eliminar espacios duplicados en cadenas.
- **limpiarExtremos()**.-Su propósito es eliminar espacios al inicio y final y su uso es la normalización de entradas de usuario

- **Gestión de Administradores**

`void registrar_administrador`

- Validar nombre (solo letras)
- Verificar unicidad
- Validar contraseña (mínimo 6 caracteres)
- Guardar en archivo

login_Administrador()

`int login_Administrador()`

Proceso de autenticación:

1. Leer credenciales
2. Normalizar datos
3. Buscar en archivo ``admins.txt``
4. Comparar credenciales

Retorno: 1 si exitoso, 0 si falla

- **Registro de Vehículos**

registro()

Características especiales:

- Implementa recursividad para múltiples registros
- Validación robusta con manejo de errores
- Interfaz amigable con mensajes claros

- **Cálculos Financieros**

calcular_matricula()

Calcula el valor total a pagar por la matriculación de un vehículo según sus características.



- Extrae los datos relevantes del vehículo (año, cilindraje, tipo, avalúo, revisiones).
- Aplica las fórmulas de tarifas por cilindraje y edad, y calcula el impuesto verde si corresponde.
- Calcula multas por revisiones técnicas faltantes.
- Suma todos los valores y muestra el resultado detallado al usuario.

Características:

- Algoritmo decisional con múltiples condiciones.
- Precisión en cálculos y presentación de resultados.
- Permite generar comprobante oficial.

• Funciones de Búsqueda

Propósito: Permite al usuario buscar un vehículo por su placa y calcular automáticamente el valor de matrícula correspondiente.

- Solicita al usuario la placa a buscar y valida el formato usando ``esPlacaValida()``.
- Recorre la lista de registros en memoria y compara cada placa con la ingresada.
- Si encuentra el vehículo, llama a ``calcular_matricula()`` para mostrar el cálculo detallado y ofrece la opción de generar el comprobante.
- Si no encuentra la placa, informa al usuario que no existe el registro.

Características:

- Búsqueda lineal eficiente para conjuntos pequeños.
- Integración directa con el cálculo y generación de comprobantes.
- Mensajes claros y validación robusta de entrada.

buscarplaca()

-Propósito: Realiza una búsqueda directa de un vehículo en la lista de registros utilizando la placa como clave.

- Flujo principal:

- Recibe la placa a buscar y la lista de registros.
- Itera sobre todos los registros y compara la placa de cada uno con la buscada.
- Si encuentra coincidencia, muestra todos los datos del vehículo y propietario.
- Si no existe, informa al usuario que no se encontró el registro.

-Características:

- Algoritmo de búsqueda lineal ($O(n)$).
- Presentación completa de los datos encontrados.
- Utilidad para consultas rápidas y verificación de información.

BIBLIOGRAFÍA

Agencia Nacional de Tránsito del Ecuador (2024). *Normativa de matriculación vehicular*. Recuperado de: <https://www.ant.gob.ec/>

Kernighan, B. W., & Ritchie, D. M. (1988). *The C Programming Language* (2nd ed.). Prentice Hall.

Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall.