Sistema Integral de Matriculación Vehicular en C

Grupo: 9

Integrantes:

Cristhian Albán

Michael Sornoza

Ariana Alomoto

Docente: Ing. Nelson Herrera

Asignatura: Programación

Fecha: 29/07/2025

1. Problema y Objetivo

El proceso de matriculación vehicular presenta ineficiencias como registros manuales, cálculos erróneos de valores a pagar, falta de control sobre revisiones técnicas y dificultades para acceder a la información de los vehículos. Por lo cual existe la necesidad de automatizar y simplificar el complejo proceso de matriculación.

Se busca desarrollar un sistema integral en lenguaje C que permita registrar, gestionar y simular el proceso de matriculación vehicular. El sistema debe aplicar los conceptos aprendidos durante el semestre, tales como: lógica algorítmica, estructuras de control, manejo de archivos, punteros y buenas prácticas de programación, usando los principios de Clean Code

2. Solución

La solución propuesta es crear un Sistema Integral de Matriculación Vehicular en C, el cual permitirá:

- Iniciar sesión: registro de usuario, con sus respectivas validaciones, para luego poder iniciar sesión y acceder al sistema de matriculación.
- Registro de Vehículos: Ingreso y validación de datos como placa, cédula del propietario, año de fabricación, tipo de vehículo y avalúo.
- Cálculo de Matrícula: Aplicación de reglas de negocio que incluyen descuentos por revisiones técnicas cumplidas y la inclusión de multas acumuladas.
- Búsqueda y Listado: Posibilidad de buscar un vehículo por su placa y mostrar sus datos, además de listar todos los vehículos matriculados.
- Generación de comprobante: Visualización en pantalla del resumen del proceso de matrícula y guardado del comprobante en un archivo de texto (.txt).

- Persistencia de datos: Almacenamiento y recuperación de la información mediante archivos para asegurar la disponibilidad de los datos registrados al iniciar el programa.

La solución es adecuada porque aborda directamente la problemática de la gestión de matriculaciones al automatizar los cálculos complejos y la validación de datos. Utiliza el lenguaje C, lo que permite un control preciso sobre los recursos y un rendimiento eficiente, adecuado para sistemas de consola. Además el uso de archivos de texto para la persistencia garantiza que la información no se pierda entre ejecuciones.

3. Metodología y Desarrollo

Se trabajo de acuerdo al cronograma y las sprints, repartiéndonos las funciones del código y usando GitHub para subir los avances.

También se realizó una retroalimentación en cada Sprint, para evaluar el avance, planificar las próximas tareas e identificar posibles problemas o bloqueos.

Sprint	**Objetivo principal**
Sprint 1	Planificación y configuración inicial
Sprint 2	Registro de vehículos, estructura, validación y almacenamiento
Sprint 3	Cálculos de matrícula, funciones de búsqueda/listado y comprobantes
Sprint 4	Validación, manejo de errores, pruebas y creación los entregables

Los recursos claves fueron:

- Lenguaje de Programación: C.
- Control de Versiones: Git y GitHub.
- Principios de Diseño: Clean Code y SOLID (adaptados a C).
- Manejo de Datos: Archivos de texto para persistencia

4. Resultados e Impacto

Logramos crear un sistema funcional que registra usuarios, inicia sesión, registra vehículos, calcula matrículas, busca, lista vehículos y genera comprobantes.

Beneficios:

- Automatización y eficiencia en el proceso de matriculación: elimina la necesidad de cálculos manuales complejos para la matrícula, reduciendo errores y el tiempo de procesamiento, y por otra parte agiliza el registro y la consulta de vehículos.
- Mayor precisión en los cálculos: se asegura que los cálculos de la matrícula sean consistentes y correctos según las reglas de negocio.
- Organización de Datos: centraliza la información vehicular en archivos persistentes.
- Mantenibilidad del código por buenas prácticas: el uso de Clean Code facilita futuras modificaciones y extensiones del sistema.

5. Conclusiones y Recomendaciones

Con este proyecto hemos logrado mejorar el control de las estructuras en C, implementar de mejor manera las validaciones, implementar librerías, generar archivos y aplicar Clean Code.

Sugerencias:

- Añadir funcionalidades de modificación/eliminación de registros, esto podría hacer que el usuario tenga mas facilidad al usar el sistema (por si se equivica)
- Implementar un sistema de manejo de errores más detallado que ofrezca mensajes específicos para cada tipo de error al usuario, para que el usuario entienda de mejor manera lo que ocurre
- Mejorar la portabilidad eliminando dependencias de comandos como: system("pause"); , ya que este tipo de comandos no son compatibles con todos los sistemas