Análisis y Desarrollo del Sistema para Taller Mecánico

Análisis de Requerimientos

Clases:

- **Cliente:**
    - Atributos: nombre, dirección, teléfono, correo electrónico.
    - Métodos: registrarCliente(), modificarCliente(), eliminarCliente(), buscarCliente().
- **Vehículo:**
    - Atributos: marca, modelo, año, VIN, propietario (Cliente).
    - Métodos: registrarVehiculo(), modificarVehiculo(), eliminarVehiculo(), buscarVehiculo().
- **Servicio:**
    - Atributos: nombre, descripción, costo.
    - Métodos: registrarServicio(), modificarServicio(), eliminarServicio(), buscarServicio().
- **Cita:**
    - Atributos: cliente (Cliente), vehículo (Vehículo), servicio (Servicio), fecha, hora, mecánico (Mecánico).
    - Métodos: registrarCita(), modificarCita(), eliminarCita(), buscarCita().
- **Mecánico:**
    - Atributos: nombre, especialidad, disponibilidad.
    - Métodos: registrarMecánico(), modificarMecánico(), eliminarMecánico(), buscarMecánico().

Interfaz de usuario básica:

El código incluye una interfaz de usuario de consola simple con un menú que permite al usuario:

- Agregar clientes
- Borrar clientes
- Consultar clientes
- Editar clientes
- Mostrar todos los clientes
- Salir del sistema

El usuario interactúa con el menú escribiendo la opción que desea realizar y presionando Enter.

Justificación:

Estas clases se definen para representar diferentes elementos del sistema y permitir una gestión estructurada de la información. Cada clase se centra en una entidad específica, lo que facilita la organización y reutilización de su código. Estas clases se han definido para representar los diferentes componentes del sistema, permitiendo gestionar la información de forma estructurada. Cada clase se centra en una entidad específica, lo que facilita la organización y reutilización del código.

Atributos:

Los atributos de cada clase se han definido con el fin de capturar la información relevante para cada entidad. Por ejemplo, la clase **Cliente** requiere almacenar su nombre, dirección, teléfono y correo electrónico, mientras que la clase **Servicio** necesita el nombre, la descripción y el costo del servicio.

Métodos:

Los métodos definidos para cada clase permiten realizar las operaciones básicas de CRUD (Create, Read, Update, Delete) sobre los datos, permitiendo la gestión del sistema. Por ejemplo, el método **registrarCliente()** de la clase **Cliente** crea un nuevo registro de cliente, mientras que el método **buscarVehiculo()** de la clase **Vehículo** permite consultar la información de un vehículo específico.

Diseño de Pantalla

Para este ejemplo, se ha elegido la **gestión de clientes** como la funcionalidad que se desarrollará en la pantalla.

Pantalla de Gestión de Clientes:

- **Título:** Gestión de Clientes
- **Botones:** Agregar, Editar, Eliminar, Buscar
- **Campos de entrada:** Nombre, Dirección, Teléfono, Correo electrónico
- **Tabla:** Muestra la lista de clientes con la información de cada uno.

Funcionalidad:

- **Agregar:** Permite añadir un nuevo cliente al sistema.
- **Editar:** Permite modificar la información de un cliente existente.
- **Eliminar:** Permite eliminar un cliente del sistema.
- **Buscar:** Permite filtrar la lista de clientes por nombre, teléfono o correo electrónico.

Descripción del Código:

- AgregarCliente(): Este método permite agregar un nuevo cliente a la lista de clientes. Solicita al usuario que ingrese el nombre, dirección, teléfono

y correo electrónico del cliente, y luego crea un nuevo objeto Cliente con esa información y lo agrega a la lista clientes.

- MostrarClientes(): Este método muestra la lista de clientes registrados. Si no hay clientes registrados, muestra un mensaje indicando que no hay clientes registrados. De lo contrario, itera sobre la lista de clientes y muestra la información de cada cliente utilizando el método MostrarInformacion() de la clase Cliente.
- ConsultarCliente(): Este método permite consultar la información de un cliente específico. Solicita al usuario que ingrese el nombre del cliente a consultar, y luego busca el cliente en la lista clientes utilizando el método Find(). Si el cliente es encontrado, muestra su información utilizando el método MostrarInformacion() de la clase Cliente. De lo contrario, muestra un mensaje indicando que el cliente no fue encontrado.
- EditarCliente(): Este método permite editar la información de un cliente específico. Solicita al usuario que ingrese el nombre del cliente a editar, y luego busca el cliente en la lista clientes utilizando el método Find(). Si el cliente es encontrado, solicita al usuario que ingrese la nueva información del cliente y actualiza los campos correspondientes del objeto Cliente. Luego, muestra un mensaje indicando que el cliente fue actualizado correctamente.
- EliminarCliente(): Este método permite eliminar un cliente específico de la lista de clientes. Solicita al usuario que ingrese el nombre del cliente a eliminar, y luego busca el cliente en la lista clientes utilizando el método Find(). Si el cliente es encontrado, lo elimina de la lista clientes utilizando el método Remove(). Luego, muestra un mensaje indicando que el cliente fue eliminado correctamente.