**Iriel Vargas Pinell**

1. Análisis de requerimientos

Clases:

* 1. **public class Clientes**: Para registrar la información de los clientes en el sistema.
     1. Atributos:
* string Nombre: Identificar al cliente
* string Direccion: Para contactar al cliente
* string Telefono: Comunicarse con el cliente
* string CorreolEctronico: Comunicarse con el cliente
  + 1. Métodos:
* static void RegistrarCliente(): Para solicitar la información del cliente y añadirla a la lista de clientes.
  1. **public class Vehiculos:** Para registrar la información de los vehículos pertenecientes a los clientes.
     1. Atributos:
        + string Marca: Para identificar el vehículo según el fabricante
        + string Modelo: Identificar el vehículo según el modelo
        + int Anno: Identificar el año de fabricación del vehículo
        + string NumeroVIN: Identificar el número único del vehículo
        + Cliente Propietario: Relaciona con la clase Cliente para identificar el dueño del vehículo.
     2. Métodos:
        + static void RegistrarVehiculo() : Para solicitar la información del vehículo y añadirlo a la lista de vehículos, para después poder seleccionar uno de los clientes como propietario llamando de la clase Clientes.
        + Borrar:
  2. **public class Servicios**: Para mostrar los servicios que ofrece el taller.
     1. Atributos:
        + string NombreServicio: Para indicar el servicio que se ofrece
        + string Descripcion: Detallar lo que va incluido en el servicio
        + float Costo: Indicar el precio del servicio que se ofrece
     2. Métodos:
        + static void RegistrarServicio(): Para añadir un nuevo servicio a la lista de servicios.
  3. **public class Mecanico**: Para registrar la información de los mecánicos que trabajan en el taller.
     1. Atributos:
        + string Nombre: Identificar al mecánico
        + string Especialidad: Área en que se desempeña el mecánico
        + bool Disponibilidad: Citas disponibles para ofrecer el servicio
     2. Métodos:
        + static void RegistrarMecanico(): Para añadir un mecánico a la lista de mecánicos.
  4. **public class Cita**: Para gestionar las citas programadas en los servicios
     1. Atributos:
        + Cliente Cliente: Identificar quién solicita el servicio, para gestionar la información de contacto y citas.
        + Vehiculo Vehiculo: Identificar a cuál vehículo se ofrece el servicio, para gestionar marca, modelo, año..
        + Servicio Servicio: Cuál servicio se va a realizar, para conocer el precio y diferentes servicios que se ofrecen.
        + DateTime FechaHora: En qué fecha se va a realizar el servicio, para saber cuándo el cliente llevará el vehículo.
        + Mecanico Mecanico: Cuál mecánico realizará el servicio
     2. Métodos:
        + static void RegistrarCita(): Para crear una cita a partir del cliente, vehículo, servicio y mecánico procediendo a añadirlo a la lista de las citas.

1. Diseño de la pantalla

**Gestión de Clientes:**

* 1. Se declara la clase Clientes: En esta se registra la información de los clientes y es donde se gestionará lo siguiente a realizar.
  2. Menú principal del sistema: Se crea el menú con las 3 opciones mediante un bucle while para ejecutar las 3 opciones. Luego lee la opción y ejecuta un swicth con los dos métodos para las 2 principales opciones de registrar o ver los clientes existentes. De lo contrario salir del sistema.

\*\*\*\*Menú principal\*\*\*\*

* + 1. Registrar cliente
    2. Clientes registrados
    3. Salir

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Los métodos para el switch serían:

Para la opción 1: RegistrarCliente()

\*\*\*\*Registrar cliente\*\*\*\*

Nombre:  
Dirección:  
Teléfono:  
Correo electrónico:

Para la opción 2: ClientesRegistrados()

\*\*\*\*Clientes registrados\*\*\*\*

Nombre:  
Dirección:  
Teléfono:  
Correo electrónico:

Y en la opción de salir del menú sería “break” para salir del sistema y en caso de no presentar una opción válida “default” y break para solicitar intentar nuevamente.