

FASE CIERRE CONCESIONARIO

PROGRAMACIÓN:

El programa es un gestor de la base de datos de concesionario, en la que se logra gestionar la base de datos o realizar compras como cliente. Lo que significa que dependiendo de quien inicie sesión la información que aparece es distinta.

```
private boolean comprobar_credenciales_admin(String usuario, String pass){
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        cn = DriverManager.getConnection("jdbc:mysql://localhost:3306/concesionario_grupo5?serverTimezone=Europe/Madrid", user, "root", password "1213");
        System.out.println("Se conectó correctamente");

        sql = cn.createStatement();

        String SQL = "SELECT * FROM cuenta_empleado WHERE usuario = '" + usuario + "' AND contrasena = '" + pass + "'";

        registro = sql.executeQuery(SQL);

        return registro.next();
    } catch (ClassNotFoundException | SQLException ex) {
        JOptionPane.showMessageDialog(null, "Error al conectar o consultar:\n" + ex.getMessage());
        return false;
    }
}

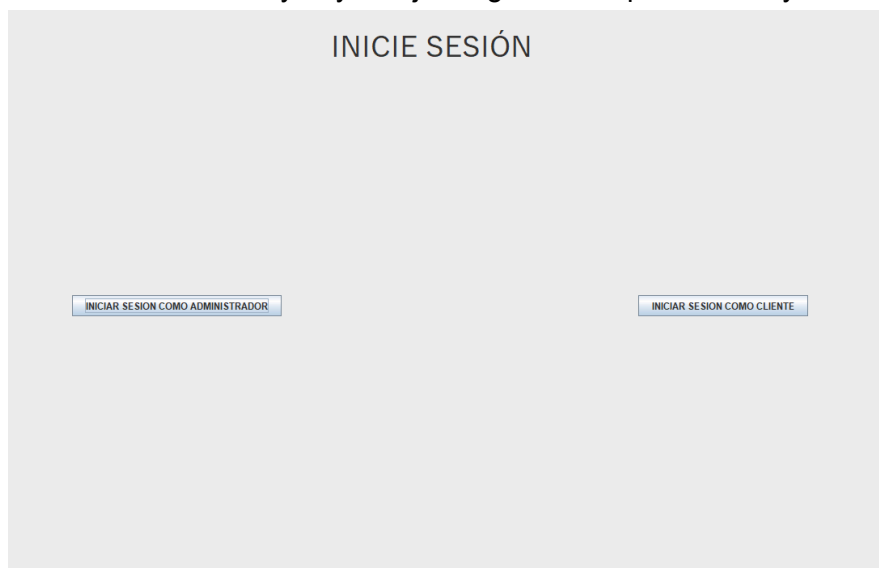
private boolean comprobar_credenciales_cliente(String usuario, String pass) {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        cn = DriverManager.getConnection("jdbc:mysql://localhost:3306/concesionario_grupo5?serverTimezone=Europe/Madrid", user, "root", password "1213");
        System.out.println("Se conectó correctamente");

        String SQL = "SELECT ID_cliente FROM cuenta_cliente WHERE usuario = ? AND contrasena = ?";
        ps = cn.prepareStatement(SQL);
        ps.setString(1, usuario);
        ps.setString(2, pass);

        registro = ps.executeQuery();

        if (registro.next()) {
            Session.idCliente = registro.getInt(1, "ID_cliente"); // Guarda el ID del cliente logueado
            return true;
        } else {
            return false;
        }
    } catch (ClassNotFoundException | SQLException ex) {
        JOptionPane.showMessageDialog(null, "Error al conectar o consultar:\n" + ex.getMessage());
        return false;
    }
}
```

Se puede ver que hay un metodo para comprobar las credenciales de administrador y para comprobar las credenciales del cliente, además hay una interfaz de inicio de sesión que hace elegir entre admin o cliente y hay dos jDialog distintos para admin y cliente.



Estas son las dos pantallas que dependiendo del inicio elegido se muestra, una gestiona la base de datos y la otra permite comprar los coches y que se muestren en el garaje.

```

public void insertar_coches_bd(Coche obj_coches) {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        cn = DriverManager.getConnection("jdbc:mysql://localhost:3306/concesionario_grupo5?serverTimezone=Europe/Madrid", user: "root", password: "1213");
        System.out.println("Se conectó correctamente");

        String SQL = "INSERT INTO coches (marca, modelo, matricula, kilometraje, año_fabricación, precio, cv, color) VALUES ("
            + obj_coches.getMarca() + ", "
            + obj_coches.getModelo() + ", "
            + obj_coches.getMatricula() + ", "
            + obj_coches.getKilometraje() + ", "
            + obj_coches.getAño_fabricación() + ", "
            + obj_coches.getPrecio() + ", "
            + obj_coches.getCv() + ", "
            + obj_coches.getColor() + ")";

        sql = cn.createStatement();
        sql.executeUpdate(sql);
        System.out.println("Coches añadido correctamente.");
    } catch (SQLException | ClassNotFoundException ex) {
        Logger.getLogger("Conexion_BD.class.getName()).log(Level.SEVERE, null, ex);
    }
}

public ArrayList<Coche> Consultar_coches() {
    ArrayList<Coche> catalogo = new ArrayList<>();

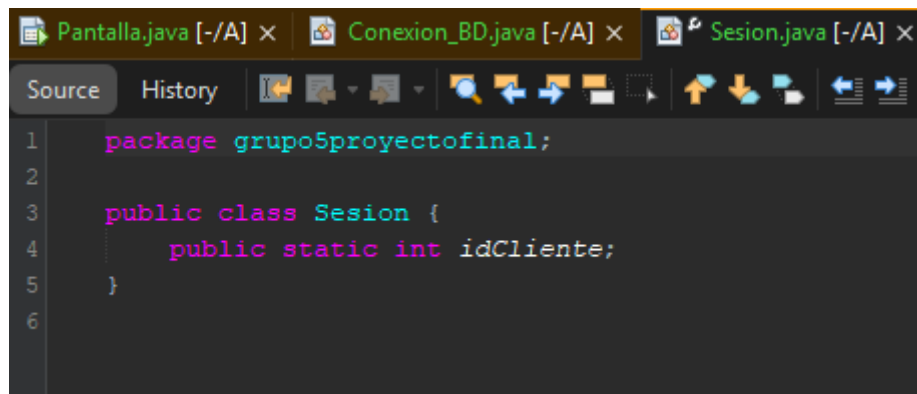
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        cn = DriverManager.getConnection("jdbc:mysql://localhost:3306/concesionario_grupo5?serverTimezone=Europe/Madrid", user: "root", password: "1213");
        System.out.println("Se conectó correctamente");
        String SQL = "Select * from coches";
        sql = cn.prepareStatement(sql);
        registro = sql.executeQuery(sql);

        while (registro.next()) {
            Coche obj_coches = new Coche();
            obj_coches.setMarca(marca: registro.getString(columnLabel: "marca"));
            obj_coches.setModelo(modelo: registro.getString(columnLabel: "modelo"));
            obj_coches.setMatricula(matricula: registro.getString(columnLabel: "matricula"));
            obj_coches.setKilometraje(kilometraje: registro.getFloat(columnLabel: "kilometraje"));
            obj_coches.setAño_fabricación(año_fabricación: registro.getString(columnLabel: "año_fabricación"));
        }
    }
}

```

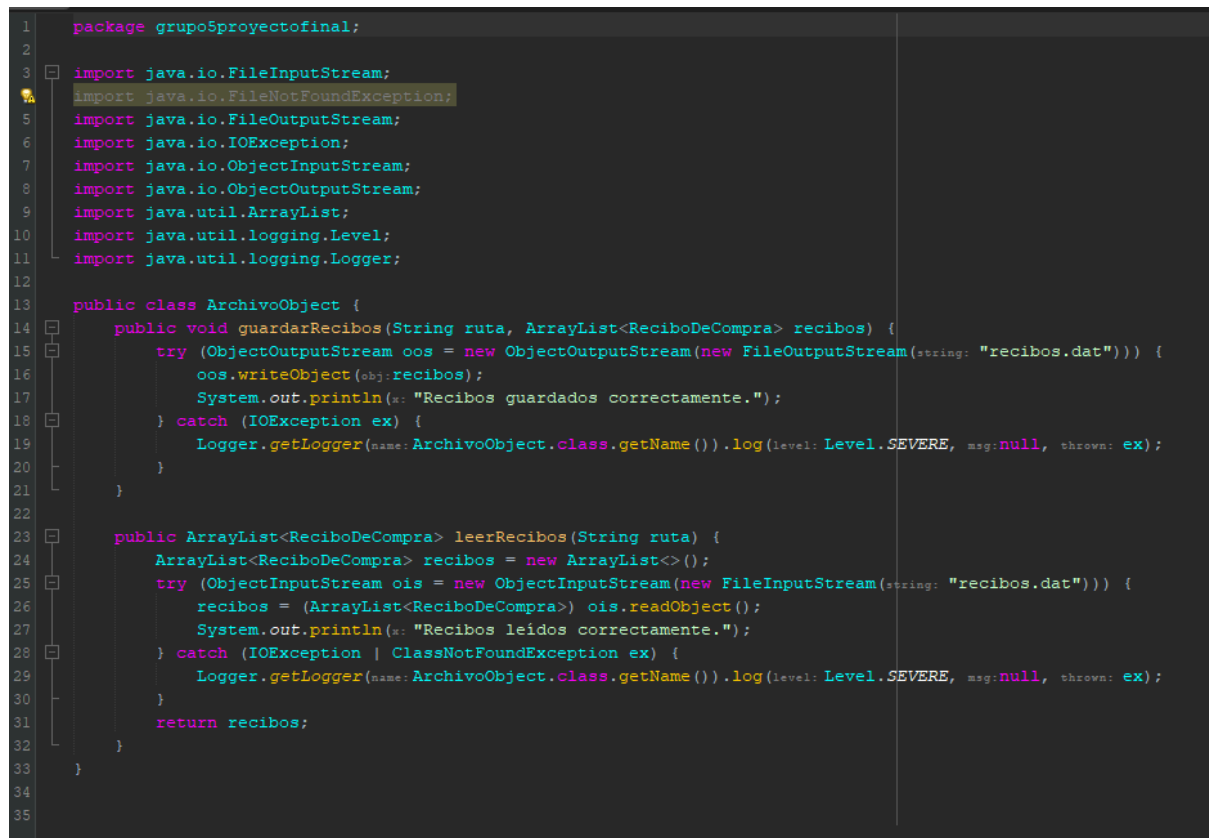
Los metodos que gestionan la base de datos están metidos en la misma clase para recurrir al mismo objeto de conexión a la base de datos.

Cada uno de los métodos realiza una función dentro de la BD, como por ejemplo realizar una consulta, o agregar un valor a una tabla etc.



```
1 package grupo5proyectofinal;
2
3 public class Sesion {
4     public static int idCliente;
5 }
6
```

Una función muy interesante es la de guardar el ID del cliente que inicia sesión, de tal manera que solo se muestren en la tabla de garaje los coches que el mismo ha comprado, pero no los del resto mediante un método que selecciona los coches filtrando por ID.



```
1 package grupo5proyectofinal;
2
3 import java.io.FileInputStream;
4 import java.io.FileNotFoundException;
5 import java.io.FileOutputStream;
6 import java.io.IOException;
7 import java.io.ObjectInputStream;
8 import java.io.ObjectOutputStream;
9 import java.util.ArrayList;
10 import java.util.logging.Level;
11 import java.util.logging.Logger;
12
13 public class ArchivoObject {
14     public void guardarRecibos(String ruta, ArrayList<ReciboDeCompra> recibos) {
15         try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("recibos.dat"))) {
16             oos.writeObject(recibos);
17             System.out.println("Recibos guardados correctamente.");
18         } catch (IOException ex) {
19             Logger.getLogger(ArchivoObject.class.getName()).log(Level.SEVERE, null, ex);
20         }
21     }
22
23     public ArrayList<ReciboDeCompra> leerRecibos(String ruta) {
24         ArrayList<ReciboDeCompra> recibos = new ArrayList<>();
25         try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream("recibos.dat"))) {
26             recibos = (ArrayList<ReciboDeCompra>) ois.readObject();
27             System.out.println("Recibos leídos correctamente.");
28         } catch (IOException | ClassNotFoundException ex) {
29             Logger.getLogger(ArchivoObject.class.getName()).log(Level.SEVERE, null, ex);
30         }
31         return recibos;
32     }
33 }
34
35
```

Se implementaron ficheros para que fuera posible guardar la información de la tabla de recibos antes y que guarde la información filtrada para posteriormente hacer el filtro de coches por ID.

```
Pantalla.java [-/A] x Conexion_BD.java [-/A] x Sesion.java [-/A] x ArchivoObject.java [-/A] x
Source Design History
670 } else {
671     JOptionPane.showMessageDialog(parentComponent:null, message: "Error al actualizar el coche.");
672 }
673 }
674
675 private void Table_ClienteMouseClicked(java.awt.event.MouseEvent evt) {
676     // TODO add your handling code here:
677 }
678
679 private void btn_actualizar_tabla_ClienteActionPerformed(java.awt.event.ActionEvent evt) {
680     Conexion_BD obj_con = new Conexion_BD();
681     ArrayList<Coche> catalogo = obj_con.Consultar_coche();
682
683     DefaultTableModel vaciar_tabla = (DefaultTableModel) Tabla_Cliente.getModel();
684     vaciar_tabla.setNumRows(0);
685     Tabla_Cliente.setModel(dataModel: vaciar_tabla);
686
687     DefaultTableModel modelo = (DefaultTableModel) Tabla_Cliente.getModel();
688     for (int i = 0; i < catalogo.size(); i++) {
689         Object[] coche = new Object[8];
690         coche[0] = catalogo.get(index: i).getMarca();
691         coche[1] = catalogo.get(index: i).getModelo();
692         coche[2] = catalogo.get(index: i).getMatricula();
693         coche[3] = catalogo.get(index: i).getKilometraje();
694         coche[4] = catalogo.get(index: i).getAño_fabricacion();
695         coche[5] = catalogo.get(index: i).getPrecio();
696         coche[6] = catalogo.get(index: i).getCv();
697         coche[7] = catalogo.get(index: i).getColor();
698         modelo.addRow(data: coche);
699     }
700     Tabla_Cliente.setModel(dataModel: modelo);
701 }
702
703 private void btn_Comprar_CocheActionPerformed(java.awt.event.ActionEvent evt) {
704     int filaSeleccionada = Tabla_Cliente.getSelectedRow();
705     if (filaSeleccionada != -1) {
706         try {
707             String matricula = Tabla_Cliente.getValueAt(filaSeleccionada, columna: 2).toString();
708
709             String sqlBuscarIdCoche = "SELECT ID_coche FROM coche WHERE matricula = ?";
710             ps = cn.prepareStatement(sqlBuscarIdCoche);
711             ps.setString(parameterIndex: 1, s: matricula);
712             registro = ps.executeQuery();
713
714             if (registro.next()) {
715                 int idCoche = registro.getInt(columnLabel: "ID_coche");
716
717                 // Insertar la compra en la tabla recibo_de_compra
718                 String sqlInsertarCompra = "INSERT INTO recibo_de_compra (ID_cliente, ID_coche) VALUES (2, 3)";
719             }
720         } catch (SQLException ex) {
721             JOptionPane.showMessageDialog(this, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
722         }
723     }
724 }
```

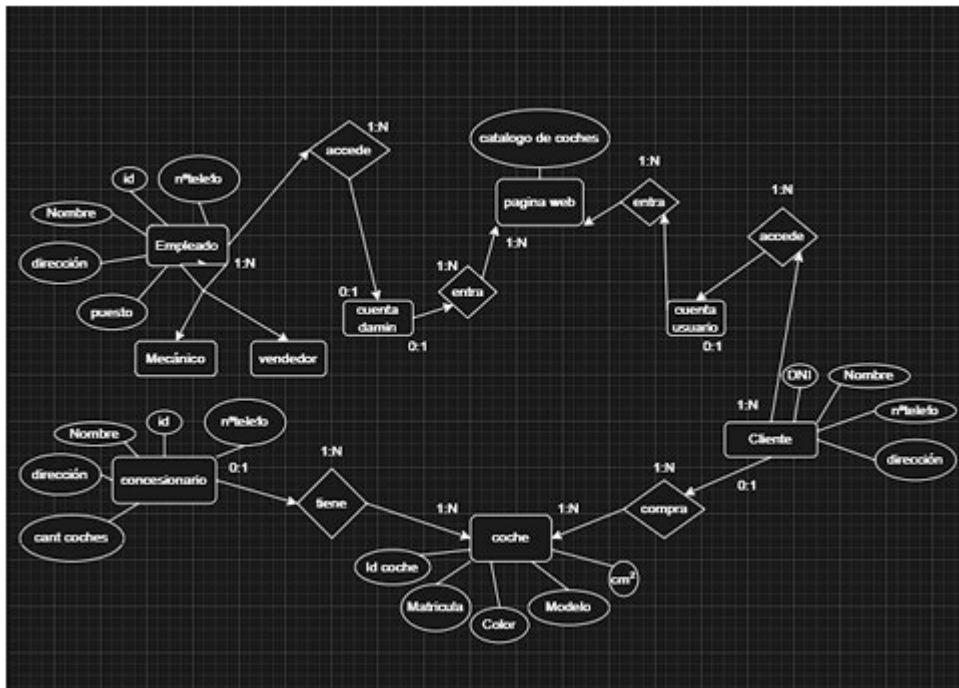
Lo cierto es que el resto de métodos no tiene ninguna complicación, simplemente con las queries de los métodos de la clase Conexion_BD y los valores que se reciben se guardan en un objeto y se muestran en una tabla.

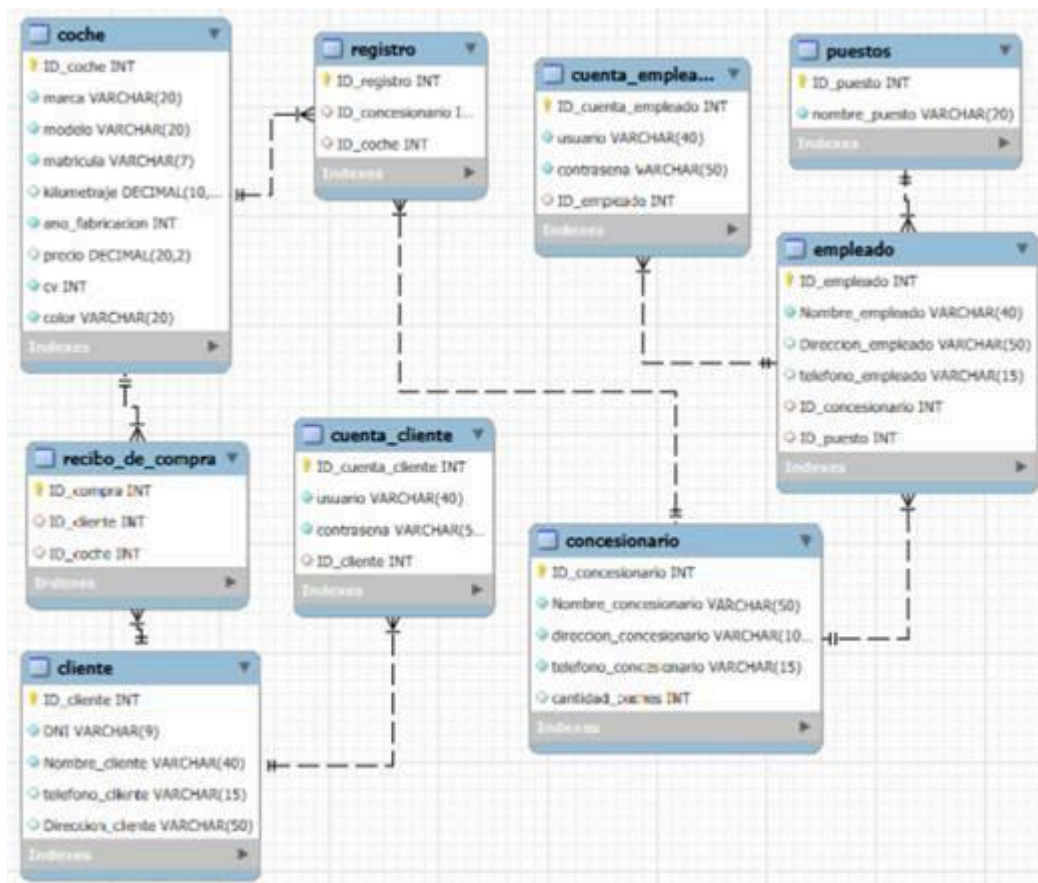
El resto de funcionalidades son simplemente abrir pantallas y guardar los cambios en la BD.

BASES DE DATOS:

Para realizar la Base de Datos hemos usado el programa MySQL WorkBench, En esta hemos creado una base de datos llamada “concesionario_grupo5”.

Inicialmente se creó un esquema básico de cómo debería ser la base de datos, este evolucionando gracias a correcciones de diseño y ayuda de tanto como compañeros como profesores en el siguiente.





Con este esquema final, los scripts de las tablas fueron creados para dar información y unas pautas a seguir para los valores en la base de datos.

Tablas:

Concesionario

```

CREATE TABLE Concesionario (
    ID_concesionario INT PRIMARY KEY AUTO_INCREMENT,
    Nombre_concesionario VARCHAR(50) NOT NULL,
    direccion_concesionario VARCHAR(100) NOT NULL,
    telefono_concesionario VARCHAR(15) NOT NULL,
    cantidad coches INT
);
  
```

Puestos

```
CREATE TABLE Puestos (  
    ID_puesto INT PRIMARY KEY AUTO_INCREMENT,  
    nombre_puesto VARCHAR(20) NOT NULL  
);
```

Clientes

```
CREATE TABLE Cliente (  
    ID_cliente INT PRIMARY KEY AUTO_INCREMENT,  
    DNI VARCHAR(9) UNIQUE NOT NULL,  
    Nombre_cliente VARCHAR(40) NOT NULL,  
    telefono_cliente VARCHAR(15),  
    Direccion_cliente VARCHAR(50)  
);
```

Coche

```
CREATE TABLE Coche (  
    ID_coche INT PRIMARY KEY AUTO_INCREMENT,  
    marca VARCHAR(20) NOT NULL,  
    modelo VARCHAR(20) NOT NULL,  
    matricula VARCHAR(7) NOT NULL UNIQUE,  
    kilometraje DECIMAL(10,1),  
    ano_fabricacion INT NOT NULL,  
    precio DECIMAL(20,2),  
    cv INT NOT NULL,  
    color VARCHAR(20) NOT NULL  
);
```

Empleado

```
CREATE TABLE Empleado (  
    ID_empleado INT PRIMARY KEY AUTO_INCREMENT,  
    Nombre_empleado VARCHAR(40) NOT NULL,  
    Direccion_empleado VARCHAR(50),  
    telefono_empleado VARCHAR(15),  
    ID_concesionario INT,  
    ID_puesto INT,  
    FOREIGN KEY (ID_concesionario) REFERENCES Concesionario(ID_concesionario),  
    FOREIGN KEY (ID_puesto) REFERENCES Puestos(ID_puesto)  
);
```

Registro

```
• CREATE TABLE Registro (  
    ID_registro INT PRIMARY KEY AUTO_INCREMENT,  
    ID_concesionario INT,  
    ID_coche INT,  
    FOREIGN KEY (ID_concesionario) REFERENCES Concesionario(ID_concesionario),  
    FOREIGN KEY (ID_coche) REFERENCES Coche(ID_coche)  
);
```

Recibo de Compra

```
• CREATE TABLE Recibo_de_compra (  
    ID_compra INT PRIMARY KEY AUTO_INCREMENT,  
    ID_cliente INT,  
    ID_coche INT,  
    FOREIGN KEY (ID_cliente) REFERENCES Cliente(ID_cliente),  
    FOREIGN KEY (ID_coche) REFERENCES Coche(ID_coche)  
);
```

Cuenta de empleado

```
• CREATE TABLE cuenta_empleado (  
    ID_cuenta_empleado INT PRIMARY KEY AUTO_INCREMENT,  
    usuario VARCHAR(40) NOT NULL,  
    contrasena VARCHAR(50) NOT NULL,  
    ID_empleado INT,  
    FOREIGN KEY (ID_empleado) REFERENCES Empleado(ID_empleado)  
);
```

Cuenta de Cliente

```
• CREATE TABLE cuenta_cliente (  
    ID_cuenta_cliente INT PRIMARY KEY AUTO_INCREMENT,  
    usuario VARCHAR(40) NOT NULL,  
    contrasena VARCHAR(50) NOT NULL,  
    ID_cliente INT,  
    FOREIGN KEY (ID_cliente) REFERENCES Cliente(ID_cliente)  
);
```

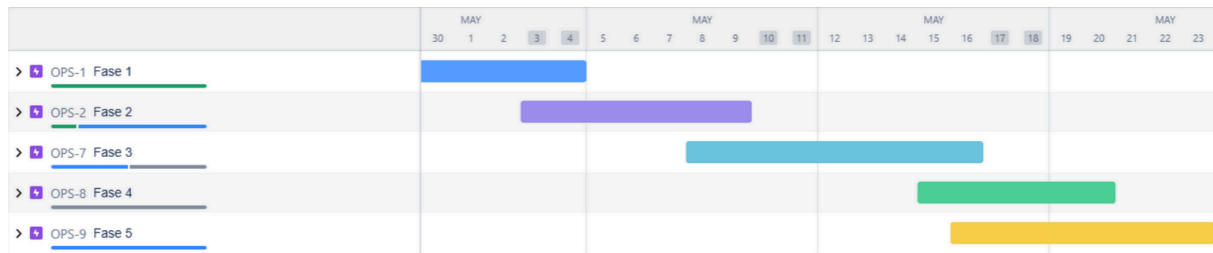

Tras todo esto, en MySQL se crearon una serie de valores para no tener tan vacía la base de datos

- ```
INSERT INTO Empleado (Nombre_empleado, Direccion_empleado, telefono_empleado, ID_concesionario, ID_puesto)
VALUES
('Lucía Martínez', 'Calle Real 12, Madrid', '611223344', 1, 1), -- Vendedor
('Carlos Pérez', 'Av. Norte 77, Madrid', '622334455', 1, 2), -- Mecánico
('María Gómez', 'Calle Sur 33, Madrid', '633445566', 1, 3); -- Gerente
```
- ```
INSERT INTO Cliente (DNI, Nombre_cliente, telefono_cliente, Direccion_cliente)
VALUES
('12345678A', 'Juan Torres', '644556677', 'Calle Luna 5, Madrid'),
('23456789B', 'Elena Ruiz', '655667788', 'Av. Sol 22, Madrid'),
('34567890C', 'Pedro Sánchez', '666778899', 'Paseo del Prado 18, Madrid');
```
- ```
INSERT INTO cuenta_empleado (usuario, contrasena, ID_empleado)
VALUES
('lucia.m', 'lucial23', 1),
('carlos.p', 'carlos123', 2),
('maria.g', 'marial23', 3);
```
- ```
INSERT INTO cuenta_cliente (usuario, contrasena, ID_cliente)
VALUES
('juan.t', 'juan123', 1),
('elena.r', 'elena123', 2),
('pedro.s', 'pedrol23', 3);
```
- ```
INSERT INTO Registro (ID_concesionario, ID_coche)
VALUES
(1, 1),
(1, 2),
(1, 3),
(1, 4),
(1, 5),
(1, 6);
```

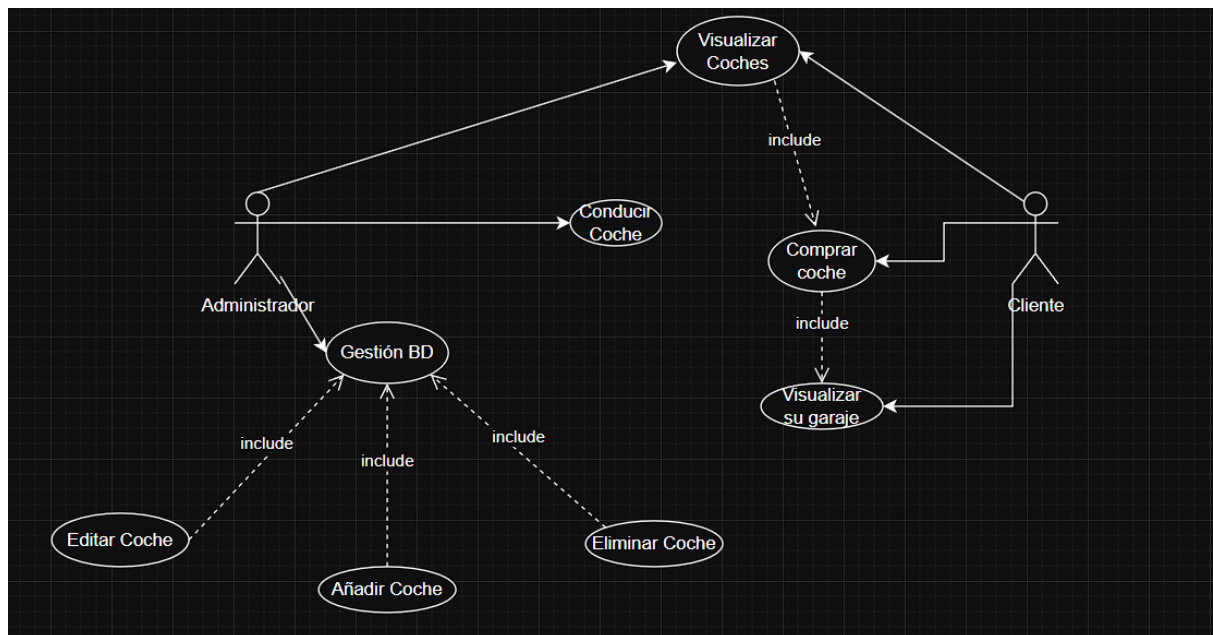
Para finalizar con lo realizable en la base de datos, se decidió crear una serie de vistas para buscar y catalogar por ciertos requisitos, como que el precio de un coche supere los 1000€.

- `create view vista_coches as  
select * from Coche ;`
- `create view vista_coches_caros as  
select * from Coche where precio >1000;`
- `create view vista_coches_baratos as  
select * from Coche where precio <1000;`
- `create view analisis_clientes as  
select * from Cliente;`
- `create view analisis_empleados as  
select * from Empleado;`

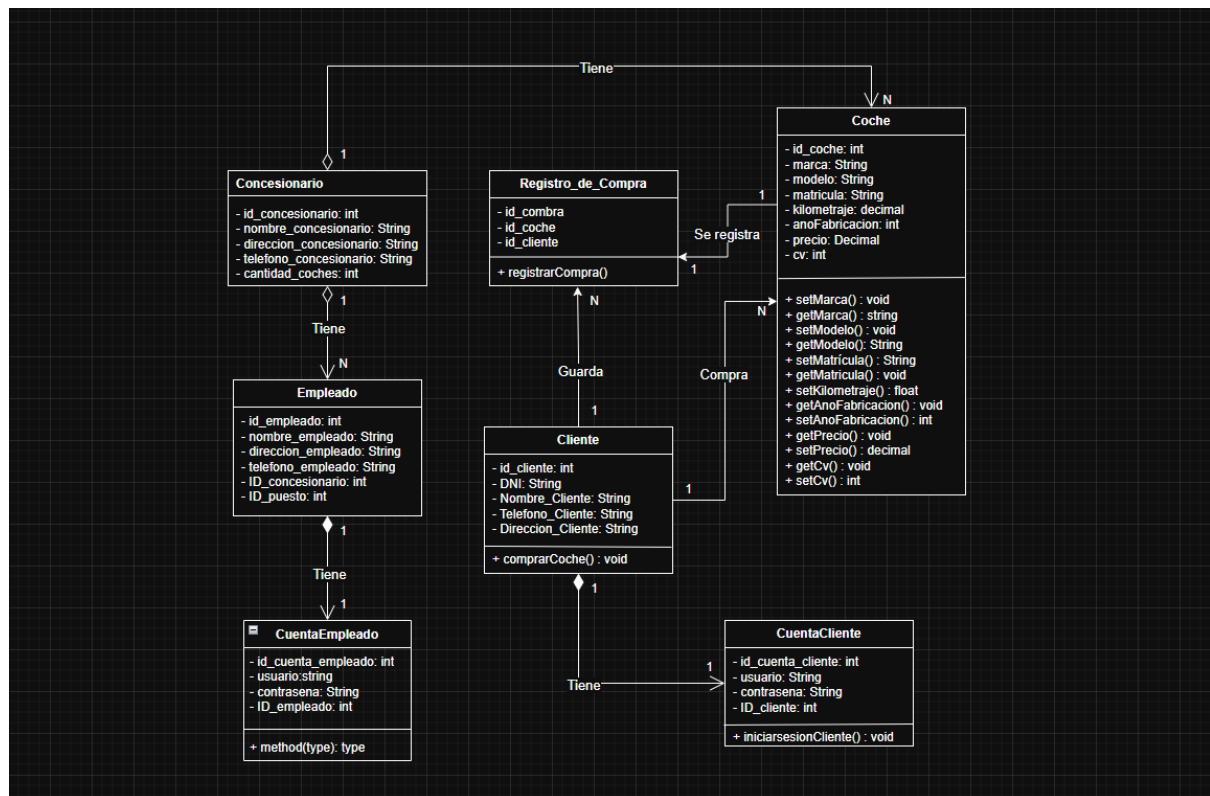
## ENTORNOS DE DESARROLLO:



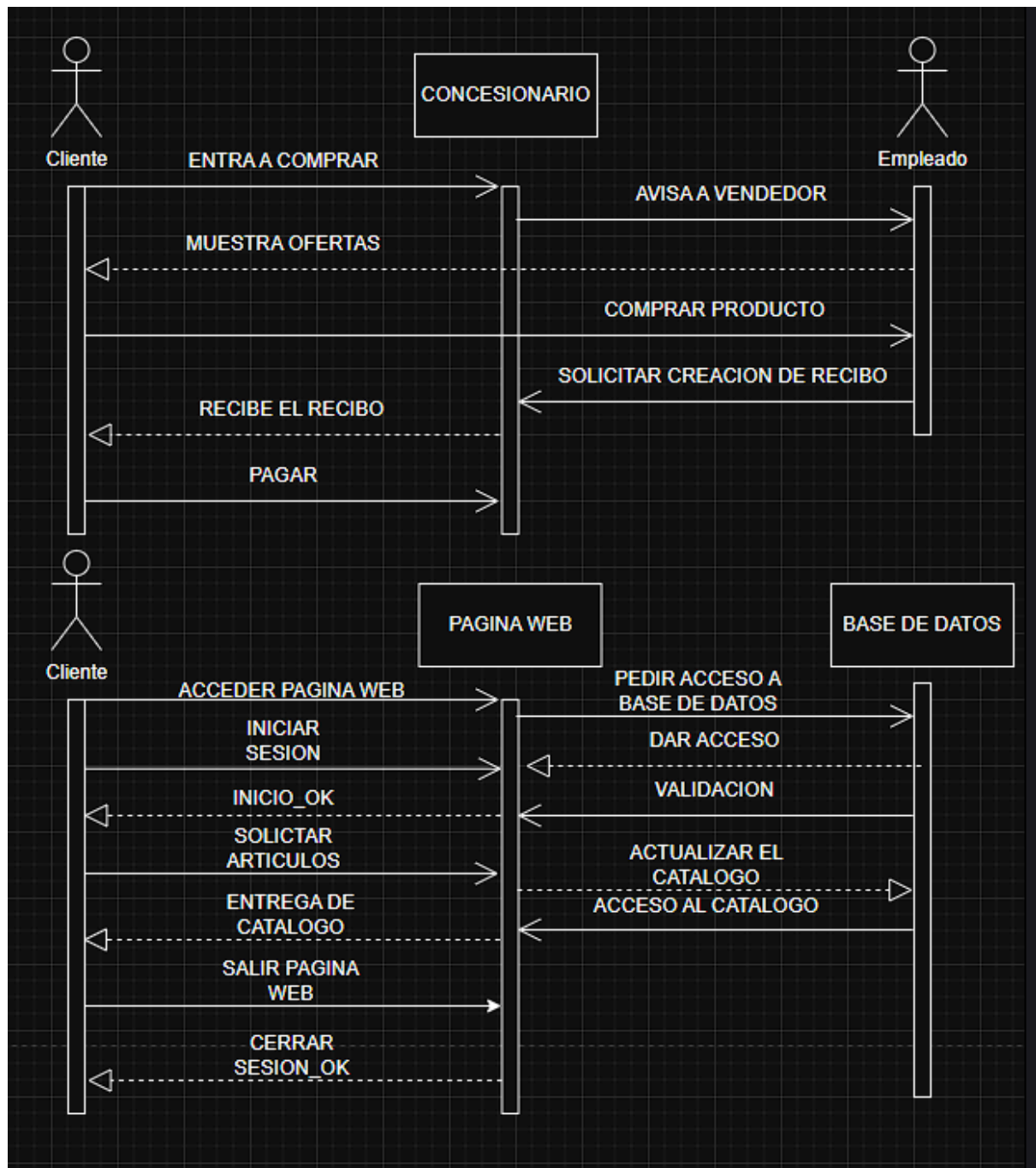
Implementamos JIRA para realizar el diagrama de Gantt



El diagrama de casos de uso del programa.



El diagrama de clases



El diagrama de secuencia

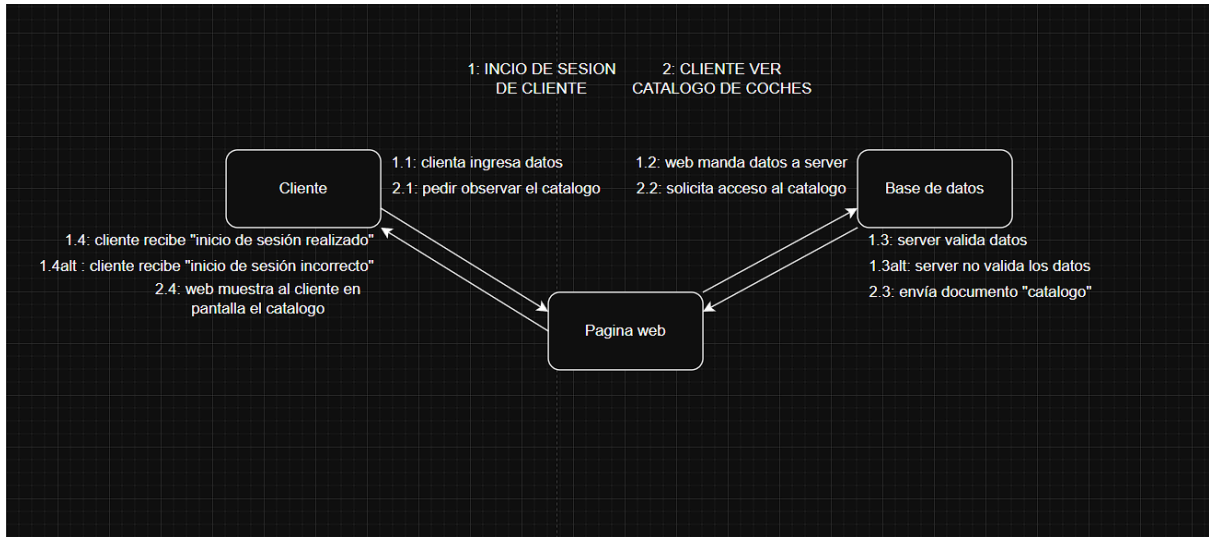
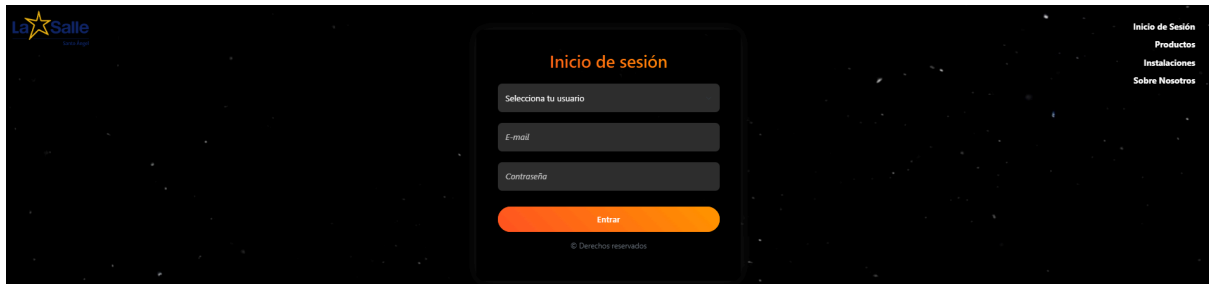


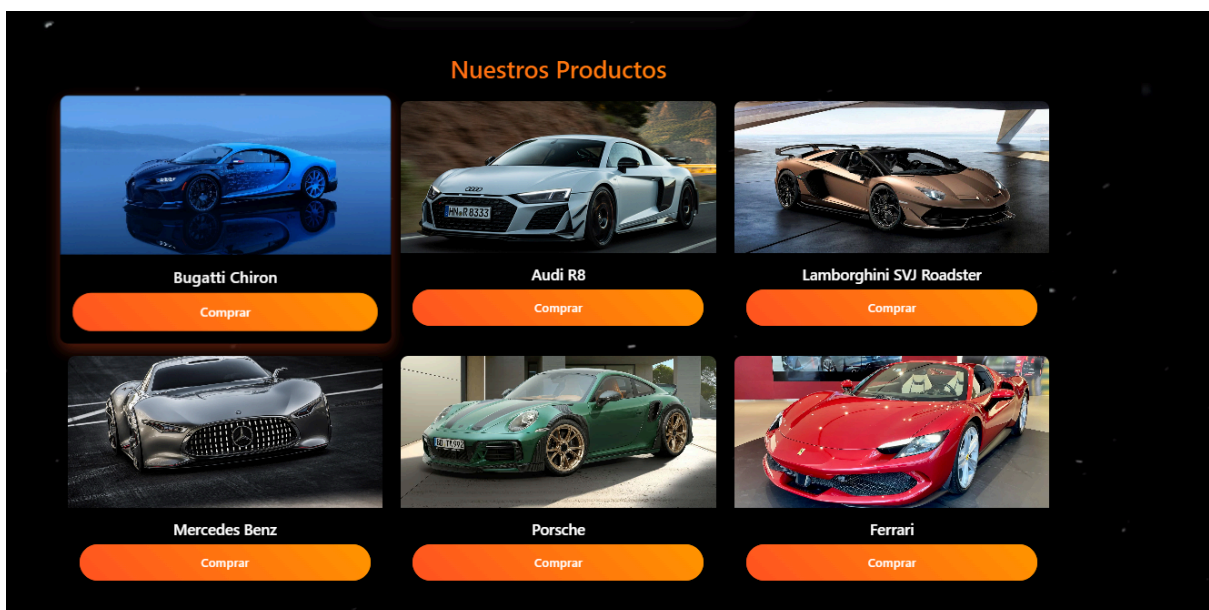
Diagrama de colaboración del programa

# LENGUAJE DE MARCAS:

## Estructura y cabecera



La página Web Consiste en un cuadro de inicio de sesión donde se puede seleccionar administrador, cliente y empleado.



Una selección de Nuestros productos, como se puede apreciar en la imagen son los 6 productos que tenemos con un botón para comprar los coches



En la última sección tenemos unas imágenes de nuestras instalaciones, tenemos 3 tarjetas Con el almacén, la zona de exposición y el taller de mantenimiento.

Las 4 últimas tarjetas son:

- Sobre nosotros
- Tarjetas de contacto
- Acuerdos legales
- Condiciones de Uso

## FUNCIONAMIENTO DEL CÓDIGO INTERNO DE LA PÁGINA

### HTML:

```
<!DOCTYPE html>
<html lang="es">

<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Prestigio</title>
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
 <link rel="stylesheet" href="styles.css">
</head>

<body>

 <video autoplay loop muted id="video_background">
 <source src="videoplayback.mp4" type="video/mp4">
 </video>

 <div id="side-menu">

 Inicio de Sesión
 Productos
 Instalaciones
 Sobre Nosotros

 </div>

 <div class="container mt-5" id="login-section">
 <h2 class="text-center mb-4">Inicio de sesión</h2>
 <form>
 <div class="mb-3">
 <label for="user-type" class="form-label">Tipo de usuario</label>
 <select class="form-select" id="user-type" required>
 <option selected disabled value="">Selecciona tu usuario</option>
 <option value="admin">Empleado</option>
 <option value="client">Cliente</option>
 </select>
 </div>
 <div class="mb-3">
 <label for="email" class="form-label">Email</label>
 <input type="email" class="form-control" id="email" placeholder="E-mail" required>
 </div>
 <div class="mb-3">
 <label for="password" class="form-label">Contraseña</label>
 <input type="password" class="form-control" id="password" placeholder="Contraseña" required>
 </div>
 </form>
 </div>
```



## 1. Declaración y <head>

- <!DOCTYPE html>

Indica que el documento es HTML5 para que el navegador lo renderice con las reglas más actuales.

- <html lang="es">

Define el idioma principal de la página ("es" = español), útil para accesibilidad y SEO.

- <meta charset="UTF-8">

Especifica la codificación de caracteres UTF-8, permitiendo acentos y caracteres especiales.

- <meta name="viewport" ...>

Hace que el diseño sea responsive, adaptándose al ancho del dispositivo (móvil, tableta, desktop).

- <title>

Texto que aparece en la pestaña del navegador.

Bootstrap y estilos propios

Se carga el CSS de Bootstrap 5.3 desde CDN para usar su sistema de rejilla, componentes y utilidades.

Luego se enlaza styles.css para estilos personalizados (animaciones, colores, posicionamiento, etc.).

## 2. Fondo y logo

- <img id="logo">

Muestra un logotipo (probablemente en la esquina superior). El atributo alt es texto alternativo para accesibilidad.

<video id="video\_background" autoplay loop muted>

Se reproduce automáticamente, en bucle y sin sonido, ocupando seguramente toda la pantalla (dependiendo del CSS). Su uso es crear un fondo animado.

## 3. Menú lateral fijo

Un menú de navegación vertical (posiblemente posicionado fijo vía CSS).

Cada <a href="#..."> enlaza a una sección interna de la misma página mediante "anclas" (id="login-section", etc.).

## 4. Sección Inicio de Sesión

.container mt-5: contenedor de Bootstrap con margen superior.

<form> sin action ni method, por lo que no envía datos a ningún servidor hasta que se añada JS o atributos.

Clases de Bootstrap (form-control, form-select, mb-3, etc.) para darle estilo coherente.

Botón de ancho completo (w-100) y con la clase btn-animated (definida en CSS para algún efecto hover).

```
<div class="container mt-5" id="products-section">
 <h2 class="text-center mb-4 text-white">Nuestros Productos</h2>
 <div class="row g-4">
 <div class="col-sm-6 col-md-4">
 <div class="card">

 <div class="card-body">
 <h5 class="card-title">Bugatti Chiron</h5>
 Comprar
 </div>
 </div>
 </div>
 <div class="col-sm-6 col-md-4">
 <div class="card">

 <div class="card-body">
 <h5 class="card-title">Audi R8</h5>
 Comprar
 </div>
 </div>
 </div>
 </div>
</div>
```

## 5. Sección de Productos

Sistema de rejilla: row g-4: fila con “gap” de 4 unidades. Cada col-sm-6 col-md-4 ocupa media pantalla en móviles y un tercio en tablet/desktop. Card de Bootstrap: imagen, título y botón de compra con estilo.

Destacar que está es la única parte en la que hemos implementado bootstrap en nuestra página web y que esta captura del código donde solo se muestran dos de nuestros productos, en realidad son 6 pero es la misma metodología para cada uno de ellos.

```

<div class="container mt-5" id="facilities-section">
 <h3 class="text-center text-white mb-4">Nuestras Instalaciones</h3>
 <div class="row g-4">
 <div class="col-md-4">
 <div class="card">

 <div class="card-body">
 <p class="card-text text-center">Almacén principal con capacidad para 500 vehículos.</p>
 </div>
 </div>
 </div>
 <div class="col-md-4">
 <div class="card">

 <div class="card-body">
 <p class="card-text text-center">Zona de exposición de vehículos de lujo.</p>
 </div>
 </div>
 </div>
 <div class="col-md-4">
 <div class="card">

 <div class="card-body">
 <p class="card-text text-center">Área de mantenimiento y preparación de vehículos.</p>
 </div>
 </div>
 </div>
 </div>
</div>

```

## 6. Sección de Instalaciones

Muy similar a la sección de productos, pero con párrafos de texto en lugar de títulos y botones.

```

<div class="container mt-5" id="about-section">
 <div class="row">
 <div class="col-md-6">
 <h3 class="text-white">Sobre Nosotros</h3>
 <p class="text-white">Somos una empresa dedicada a ofrecer los mejores vehículos de lujo del mercado. Nuestro compromiso es brindar una experiencia única</p>
 </div>
 <div class="col-md-6">
 <h3 class="text-white">Tarjetas de Contacto</h3>
 <ul class="list-unstyled text-white">
 Email: contacto@luxurycars.com
 Teléfono: +34 123 456 789
 Dirección: Calle de la Estrella, 123, Madrid, España

 </div>
 </div>
 <hr class="text-white">
 <div class="row">
 <div class="col-md-6">
 <h4 class="text-white">Acuerdos Legales</h4>
 <p class="text-white">Al utilizar este sitio web, aceptas nuestros términos y condiciones. Todos los derechos reservados.</p>
 </div>
 <div class="col-md-6">
 <h4 class="text-white">Condiciones de Uso</h4>
 <p class="text-white">Este sitio está diseñado para proporcionar información y servicios relacionados con vehículos de lujo. Por favor, utiliza este sitio</p>
 </div>
 </div>
</div>

```

## 7. Sección “Sobre Nosotros” y datos de contacto

Dos filas (row) de dos columnas cada una (col-md-6), para distribuir texto y lista de contacto. <hr> con color blanco para separar visualmente. Todo el texto tiene clase text-white para mostrarse en blanco (sobresaliendo sobre el fondo oscuro o el video).

## CSS

### 1. Estilos globales del body

```
body {
 user-select: none; /* Deshabilita la selección de texto */
 -webkit-user-select: none; /* Compatibilidad con navegadores basados en WebKit */
 -ms-user-select: none; /* Compatibilidad con Internet Explorer */
 font-size: 16px;
}
```

user-select: none evita que el usuario pueda arrastrar y seleccionar texto, imágenes o elementos en la página.

*—Se incluyen los prefijos -webkit- y -ms- para asegurar compatibilidad con navegadores basados en WebKit y versiones antiguas de Internet Explorer/Edge.—*

El font-size: 16px establece un tamaño de referencia para toda la tipografía.

### 2. Vídeo de fondo

```
#video_background {
 position: fixed;
 right: 0;
 bottom: 0;
 min-width: 100%;
 min-height: 100%;
 z-index: -1;
}
```

position: fixed lo coloca fijo respecto a la ventana, no se mueve al hacer scroll.

min-width y min-height al 100% garantizan que cubra siempre todo el ancho y alto del viewport.

z-index: -1 sitúa el vídeo por detrás de todos los demás elementos.

### 3. Formulario de login

```

#login-section {
 background-color: rgba(0, 0, 0, 0.9); /* Fondo negro más oscuro */
 border-radius: 15px;
 padding: 40px;
 max-width: 500px;
 box-shadow: 0 0 30px rgba(255, 255, 255, 0.1);
}

```

Un fondo semitransparente negro (rgba(0,0,0,0.9)) para destacar el formulario sobre el vídeo.

Bordes redondeados y sombra blanca muy suave para dar efecto de “flotante”.

#### 4. Estilo de encabezados con gradiente

```

h2, #products-section h2, #facilities-section h3 {
 font-weight: 600;
 margin-bottom: 30px;
 background: linear-gradient(45deg, #ff5722, #ff9800); /* Gradiente naranja */
 -webkit-background-clip: text;
 -webkit-text-fill-color: transparent;
 background-clip: text;
 text-align: center;
}

```

Se aplica a todos los <h2> y al <h3> de instalaciones:

Degradado lineal entre dos tonos naranjas.

Con background-clip: text y text-fill-color: transparent el texto adopta el degradado.

Texto centrado y peso seminegrita.

#### 5. Campos de formulario (input, select)

```

.form-control, .form-select {
 background-color: rgba(50, 50, 50, 0.9); /* Fondo gris oscuro */
 color: #fff; /* Texto blanco */
 border: none;
 border-radius: 8px;
 padding: 15px;
 margin-bottom: 20px;
 box-shadow: 0 0 5px rgba(0, 0, 0, 0.3); /* Sombra suave */
 appearance: none; /* Elimina el estilo predeterminado del navegador */
}

.form-control:focus, .form-select:focus {
 outline: none;
 box-shadow: 0 0 8px rgba(255, 87, 34, 0.5); /* Sombra naranja al enfocar */
}

.form-control::placeholder,
.form-select::placeholder {
 color: rgba(255, 255, 255, 0.7); /* Color blanco semitransparente */
 font-style: italic; /* Opcional: estilo cursivo */
}

.form-label {
 display: none; /* Oculta las etiquetas como en la imagen */
}

```

.form-control y .form-select se estilizan con fondo oscuro, cantos redondeados y sombra interior para simular un “campo flotante”.

appearance: none elimina flechas o estilos por defecto de los <select>.

Al enfocar (:focus), resalta con sombra naranja.

Los placeholders van en cursiva y semitransparentes.

Las etiquetas <label> se ocultan por completo.

## 6. Botones

```
.btn-primary {
 background-color: #0275d8;
 border: none;
 border-radius: 30px;
 padding: 12px;
 font-weight: 500;
 text-transform: capitalize;
 margin-top: 10px;
}

.btn-primary:hover {
 background-color: #0267be;
}
```

El botón base .btn-primary (para el login) tiene esquinas muy redondeadas y un azul uniforme.

En hover oscurece ligeramente el fondo.

## 7. Botones con animación de gradiente

```
.btn-animated {
 position: relative;
 overflow: hidden;
 color: #fff;
 background: linear-gradient(45deg, #ff5722, #ff9800); /* Gradiente naranja */
 border: none;
 border-radius: 30px;
 padding: 12px;
 font-weight: 500;
 text-transform: capitalize;
 transition: transform 0.3s ease, box-shadow 0.3s ease;
}

.btn-animated::before {
 content: '';
 position: absolute;
 top: 0;
 left: -100%;
 width: 200%;
 height: 100%;
 background: linear-gradient(45deg, #ff9800, #ff5722, #ff9800);
 transition: left 0.5s ease;
 z-index: 0;
}

.btn-animated:hover::before {
 left: 0;
}

.btn-animated:hover {
 transform: translateY(-3px);
 box-shadow: 0 8px 20px #ff9800; /* Sombra naranja */
}

.btn-animated span {
 position: relative;
 z-index: 1;
}
```

Se crea un pseudo-elemento (::before) más ancho que el botón con otro degradado.

Al hacer hover, dicho degradado se desliza desde la izquierda, y el botón “salta” y proyecta sombra.

## 8. Tarjetas (.card)

```
.card img {
 height: 200px;
 object-fit: cover;
 border-radius: 10px 10px 0 0;
 /* Elimina la animación */
 /* transform: scale(1.1); */
 /* transition: transform 0.3s ease, box-shadow 0.3s ease; */
 /* box-shadow: 0 8px 15px #ff9800; */
}

.card img:hover {
 /* Elimina el efecto hover */
 transform: none;
 box-shadow: none;
}

.card {
 background-color: #000; /* Fondo negro */
 border: none; /* Elimina el borde */
 border-radius: 10px;
 transition: transform 0.3s ease, box-shadow 0.3s ease;
}

.card:hover {
 transform: scale(1.05); /* Agrandar ligeramente la tarjeta */
 box-shadow: 0 8px 20px #ff9800; /* Sombra naranja */
}

.card-title, .card-text {
 color: #fff; /* Texto blanco */
 text-align: center;
}
```

Las imágenes de la tarjeta mantienen proporción y se recortan (object-fit: cover).

La .card tiene fondo negro semitransparente y, al pasar el ratón, se amplía un 5% con sombra naranja.

Los títulos y textos de la tarjeta aparecen en blanco y centrados.

## 9. Logo e interfaz del menú lateral

```
#logo {
 position: absolute;
 top: 10px;
 left: 10px;
 width: 150px; /* Ajusta el tamaño de la imagen */
 z-index: 10; /* Asegura que esté por encima del fondo */
}

#side-menu {
 position: fixed;
 top: 10px; /* Mueve el menú a la parte superior */
 right: 10px; /* Alinea el menú a la derecha */
 background-color: rgba(0, 0, 0, 0.9); /* Fondo negro más oscuro */
 padding: 20px;
 border-radius: 10px; /* Bordes redondeados */
 z-index: 100;
}

#side-menu ul {
 list-style: none;
 padding: 0;
 margin: 0;
}

#side-menu ul li {
 margin-bottom: 10px; /* Reduce el espacio entre los elementos */
}

#side-menu ul li a {
 text-decoration: none;
 color: #fff; /* Texto blanco */
 font-size: 16px;
 font-weight: bold;
 display: block;
 transition: color 0.3s ease;
 text-align: right; /* Alinea el texto a la derecha */
}

#side-menu ul li a:hover {
 color: #ff9800; /* Color naranja al pasar el cursor */
}
```

El logo se posiciona en la esquina superior izquierda y siempre encima.

El menú lateral flota en la esquina superior derecha sobre fondo oscuro redondeado.

Cada enlace cambia a naranja en hover y se alinea a la derecha.

## 11. Media Queries (diseño responsivo)



```

@media (max-width: 576px) {
 #logo {
 width: 100px; /* Reduce el tamaño del logo */
 top: 5px;
 left: 5px;
 }

 #side-menu {
 top: 5px;
 right: 5px;
 padding: 10px;
 }

 #side-menu ul li {
 margin-bottom: 5px;
 }

 #login-section {
 padding: 20px;
 max-width: 100%; /* Ocupa todo el ancho */
 }

 .card img {
 height: 150px; /* Reduce la altura de las imágenes */
 }

 .card-title {
 font-size: 14px; /* Reduce el tamaño del texto */
 }

 .btn-animated {
 font-size: 14px; /* Reduce el tamaño de los botones */
 padding: 10px;
 }
}

/* Ajustes para pantallas medianas (tabletas) */
@media (max-width: 768px) {
 #logo {
 width: 120px; /* Ajusta el tamaño del logo */

```

- Máximo 576px: ajustes para móviles muy pequeños (reduce tamaños y márgenes).
- Máximo 768 px: adapta tamaños para tablets.
- Mínimo 992px: en desktops grandes se restauran las dimensiones originales.

## **XML**

### **Contexto y Objetivo**

-Se dispone de una tabla SQL llamada 'Coche' con campos como ID\_coche, marca, modelo, matrícula, kilometraje, año de fabricación, precio, cv y color.

- El objetivo es transformar esta estructura relacional a un documento XML para facilitar intercambio de datos y validación mediante DTD.

### **2. Estructura del Documento XML**

#### *a) Prolog:*

Declaración XML: DOCTYPE: Referencia al archivo 'coches.dtd' para validación.

#### *b) Elemento Raíz*

Envoltorio que agrupa múltiples elementos .

#### *c) Elemento :*

Cada instancia representa un registro de la tabla original. Sub-elementos en orden secuencial según definición DTD.

### **3. Mapeo de Columnas a Elementos:**

Identificador único del coche (tipo entero). Datos alfanuméricos (cadena).  
Valores numéricos (flotantes).

### **Números enteros.**

### **4. Definición del DTD**

Puede contener cero o más .

- Secuencia obligatoria y ordenada de elementos.
- Cada sub-elemento definido como (#PCDATA) para textos simples.

### **5. Limitaciones y Buenas Prácticas**

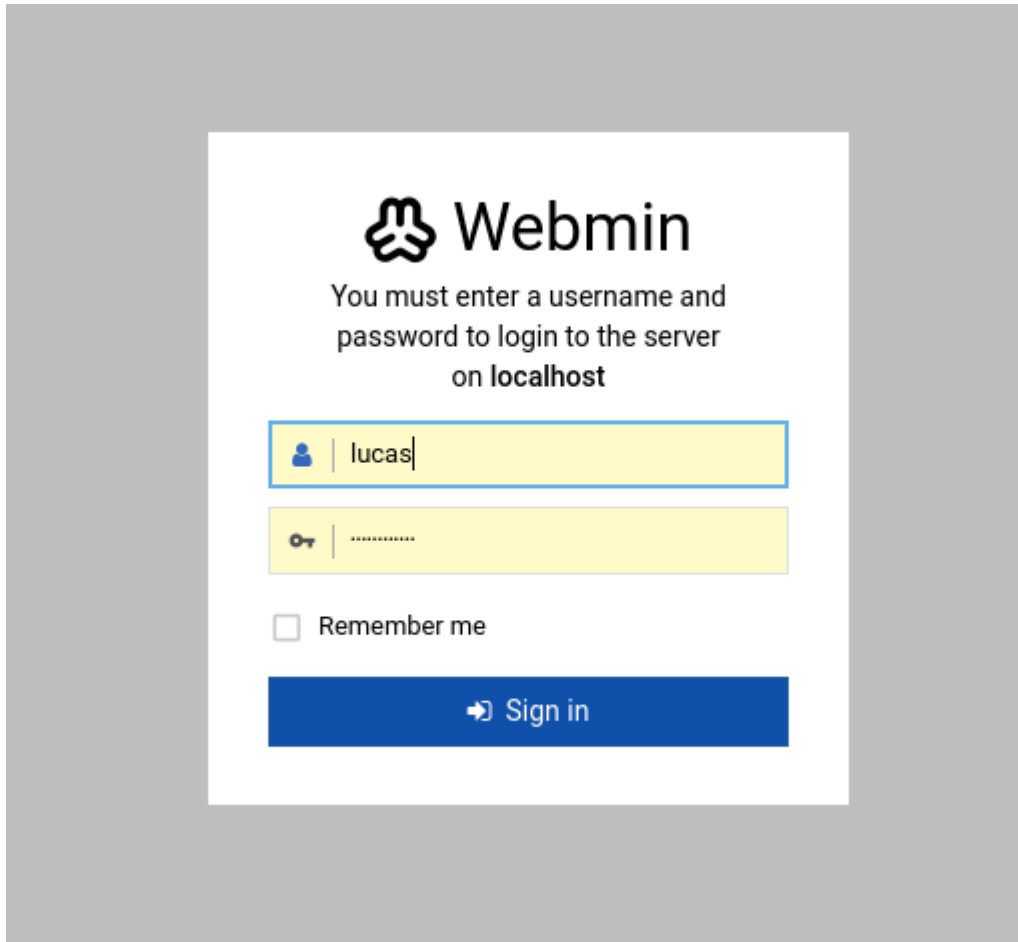
DTD válida estructura y orden, pero no valida tipos de datos (se usan #PCDATA). - Para validación de tipos más estricta, se recomienda usar XML Schema (XSD). - Utilizar herramientas como 'xmllint --noout --dtd valid coches.dtd coches.xml' para validar.

### **6. Ejemplo de Validación y Uso**

- Compartir el XML con sistemas externos que acepten DTD. - Transformar con XSLT para generar HTML o integrar con otros XML. Con esta documentación se asegura comprensión completa de la estructura, mapeo de datos y proceso de validación de los archivos XML y DTD generados.

## SISTEMAS INFORMÁTICOS:

Se ha preparado una máquina virtual (kubuntu) con webmin, apache, java y netbeans instalados



El usuario Webmin y su contraseña son los mismos que el de la máquina virtual

Creamos un servidor virtual con “servidor apache”, que se encuentra en módulos sin usar, así que lo instalaremos

Opciones de Por-Directorio

Camino	Tipo
Directory /var/www/proyectofinal	Directorio

Crear Opciones de Por-Directorio, Archivos o Localización

Tipo:

¿Expresión: ☒ Coincidencia Exacta ☐ Coincidir con expresión

Camino:

---

Opciones de Servidor Virtual

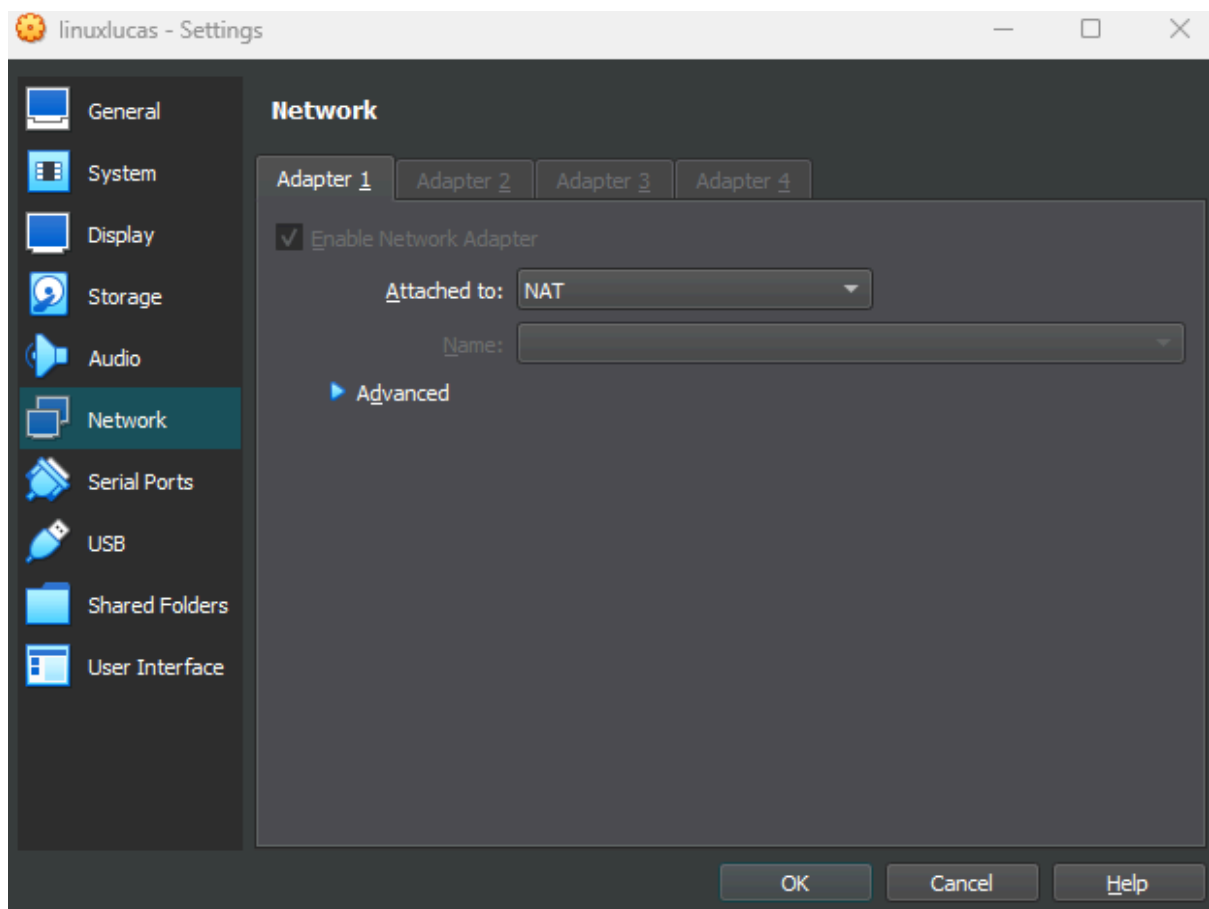
Dirección: ☐ Servidor por defecto ☐ Cualquiera  
☒ clase.es

Puerto: ☐ Por defecto ☐ Cualquiera ☒ 81

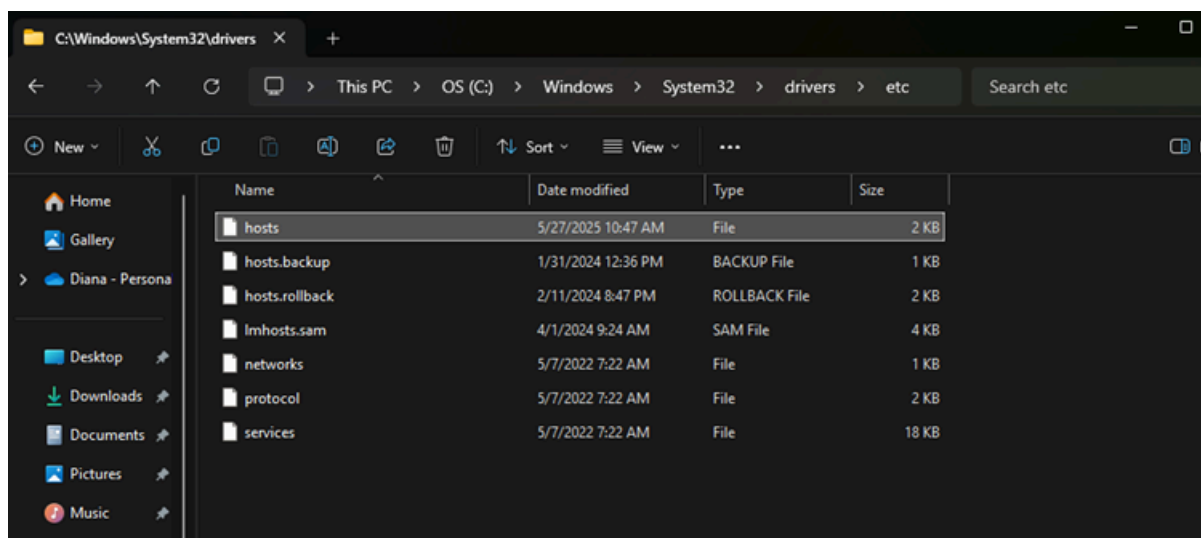
Raíz de Documento: ☐ Por defecto ☒ /var/www/proyectofinal

Nombre de Servidor: ☒ Por defecto ☐

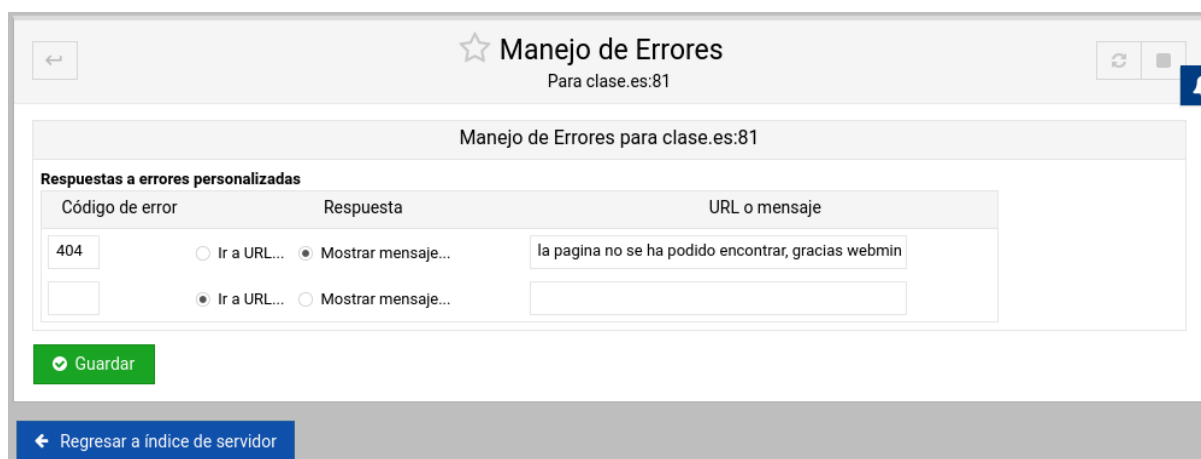
Tenemos un servidor apache con la información que se ve en la imagen, antes de crearlo se utilizaron comandos para crear el directorio /va/www/proyectofinal y el archivo index.html



El adaptador de red se cambia ya que con NAT tiene acceso a internet pero NO a la página, se tiene que utilizar Adaptador puente



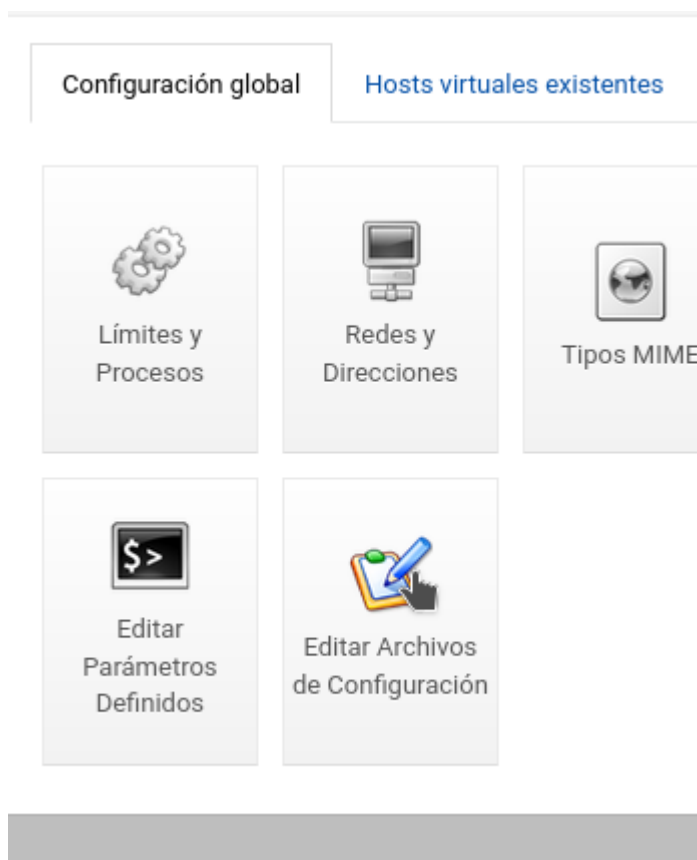
En el archivo host debemos poner la ip (la cual cambia constantemente) de la máquina virtual y el dominio de la web ([clase.com](http://clase.com) en este caso), esto es para que nuestra máquina física pueda acceder a la página que se aloja en la máquina virtual



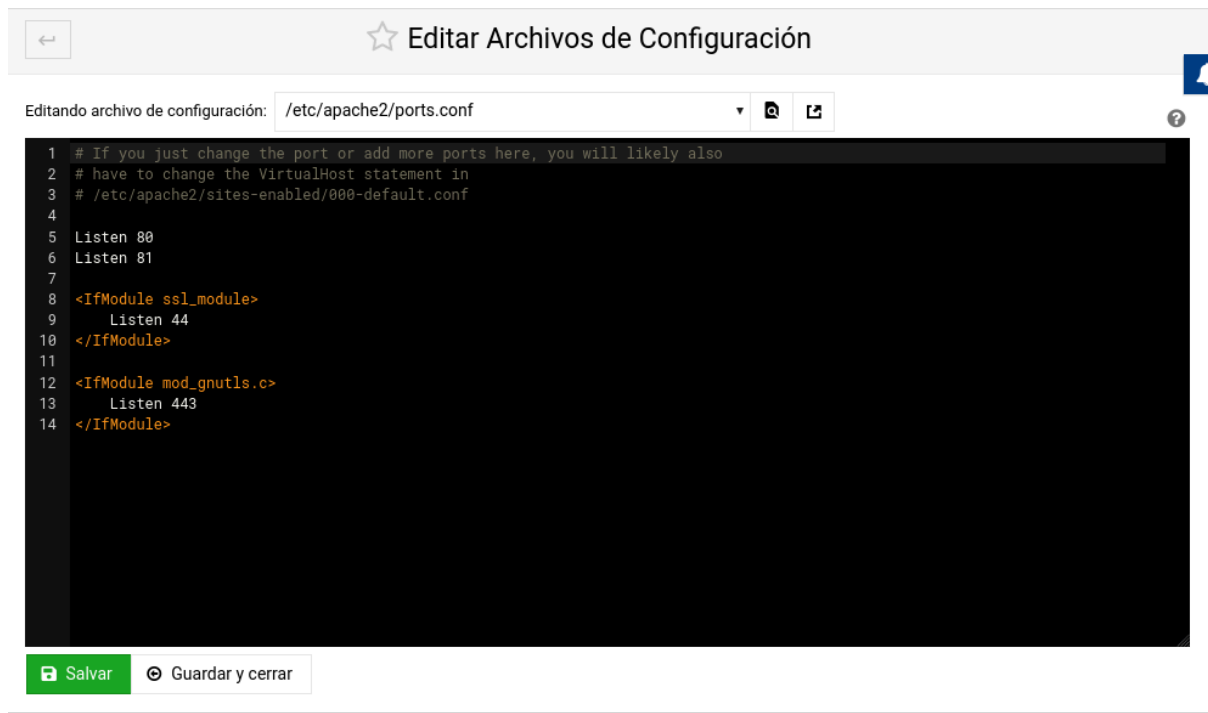
ponemos en manejo de errores un mensaje personalizado para el error 404

```
> -: sudo — Konsole
copia.txt Documentos Imágenes Plantillas Público Vídeos
Descargas Escritorio Música practica_linux snap webmin-setup-repo.sh
lucas@LucasLinux:~$ su
Contraseña:
su: Fallo de autenticación
lucas@LucasLinux:~$ sudo su
[sudo] contraseña para lucas:
root@LucasLinux:/home/lucas# cd /var
root@LucasLinux:/var# cd www
root@LucasLinux:/var/www# ls
html projectofinal users
root@LucasLinux:/var/www# cd projectofinal/
root@LucasLinux:/var/www/projectofinal# ls
index.html
root@LucasLinux:/var/www/projectofinal# nano index.html
root@LucasLinux:/var/www/projectofinal# cd /etc
root@LucasLinux:/etc# nano hosts
root@LucasLinux:/etc# ping clase.es
PING clase.es (192.168.85.77) 56(84) bytes of data.
64 bytes from clase.es (192.168.85.77): icmp_seq=1 ttl=64 time=0.190 ms
64 bytes from clase.es (192.168.85.77): icmp_seq=2 ttl=64 time=0.049 ms
64 bytes from clase.es (192.168.85.77): icmp_seq=3 ttl=64 time=0.051 ms
64 bytes from clase.es (192.168.85.77): icmp_seq=4 ttl=64 time=0.042 ms
^C
--- clase.es ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3063ms
rtt min/avg/max/mdev = 0.042/0.083/0.190/0.061 ms
root@LucasLinux:/etc#
```

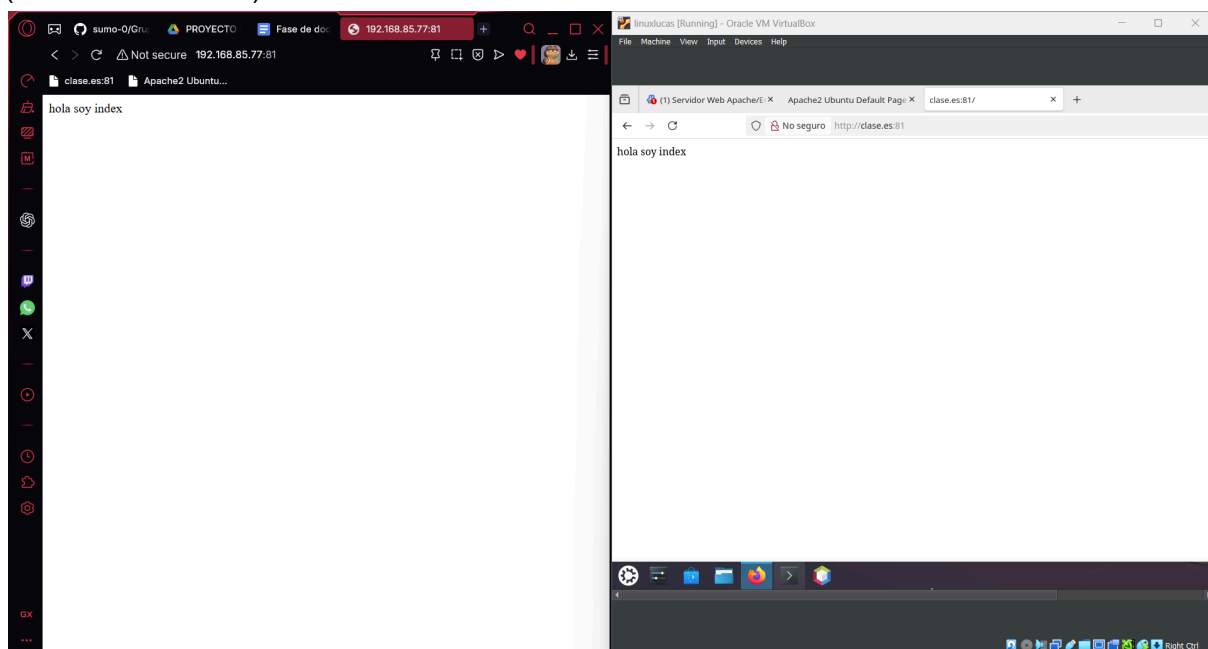
para que la VM pueda acceder al server se tendrá que agregar la ip en el documento hosts (igual que hicimos en la máquina física)



en configuración global y editar archivos de configuración



y en /etc/apache2/ports.conf hacemos que escuche al puerto que le asignamos al server (en este caso el 81)



y como se puede ver ahora se conectan al servidor