

# MIRAVVEREDA



# INDICE

INTRODUCCIÓN .....	3
DISEÑO .....	3
El Diseño General .....	3
La BBDD.....	6
La API.....	11
La Aplicación Móvil .....	14
La Página WEB.....	22
IMPLANTACIÓN .....	29
RECURSOS.....	29
Herramientas Hardware: .....	29
Herramientas Software: .....	29
CONCLUSIONES .....	30
Grado de consecución de objetivos:.....	30
Problemas encontrados: .....	30
Mejoras.....	30

## INTRODUCCIÓN

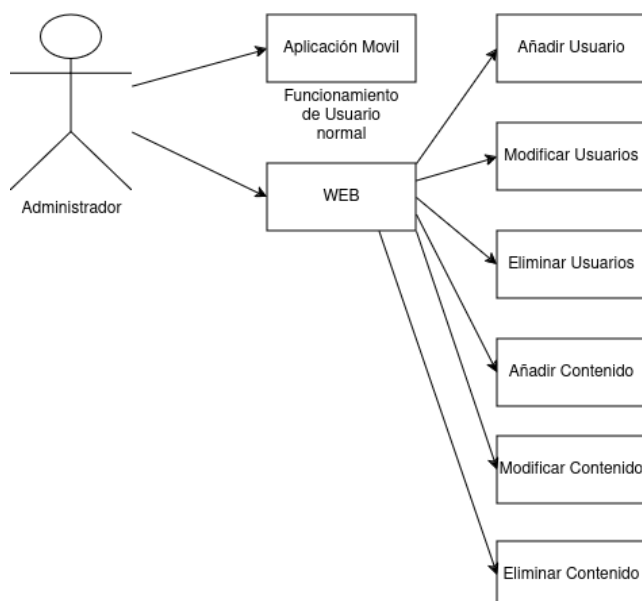
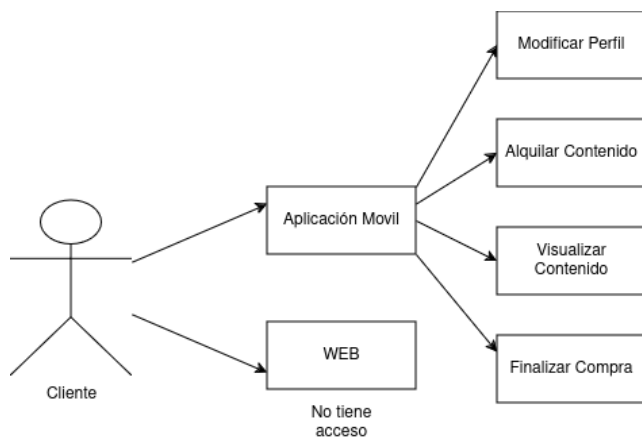
MiraVereda es un proyecto del tercer trimestre de 1ºDAW y 1ºDAM que trata de una plataforma de visualización de películas, series y cortos; cuyo funcionamiento se basa en una API que busca información en una BBDD de Oracle para que un Administrador en un entorno web pueda realizar el CRUD para Usuarios y Contenido y un Cliente en una aplicación Android pueda visualizar, alquilar contenido y modificar su propio perfil.

Este proyecto ha sido desarrollado por: Samuel Civera, Juan Carlos Santamaría, Ruben Flores y Jesus Tarín.

## DISEÑO

El Diseño General es el siguiente:

Dependiendo el Rol del usuario y donde acceda este podrá realizar una serie de acciones:





Los Usuarios podrán modificar todo de su perfil meno su Id de Usuario.

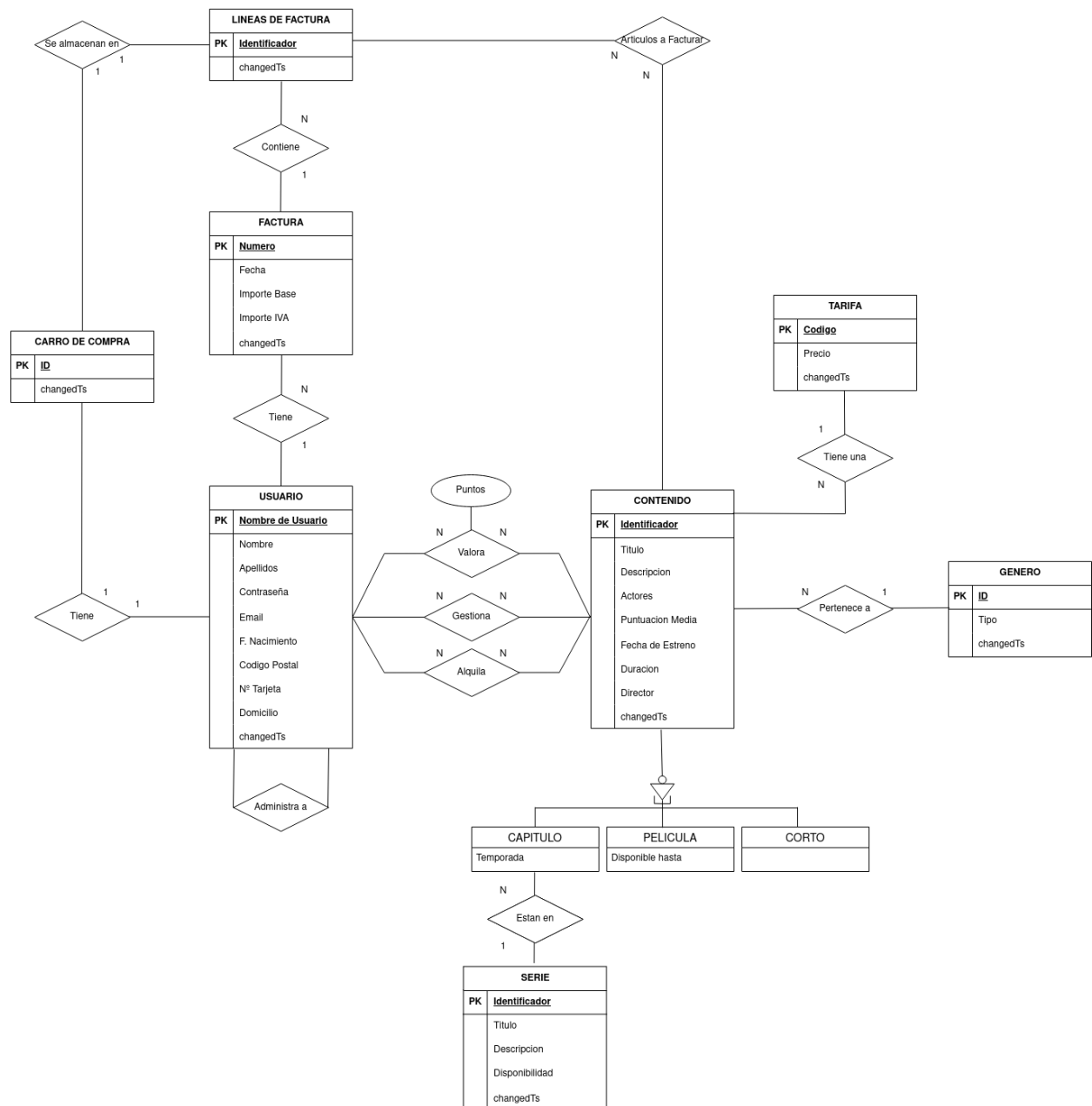
Los Usuarios podrán alquilar cualquier contenido solo una vez por compra de carro, tendrán que esperar a que se caduque de su cuenta para volver a alquilarlo.

Al añadir un capítulo, es importante indicar la id de una serie existente, si no, no te permitirá añadirlo.

Al modificar contenido no se podrá modificar ninguna id de estos y lo mismo pasa al modificar los usuarios.

La BBDD está diseñada de la siguiente manera:

**Diagrama Entidad-Relación:**



**Paso a Tablas:**

**USUARIO** (Identificador, Contraseña, Domicilio, Cod\_Postal, Email, Fech\_Nacimiento, Nombre, Apellidos, Num\_Tarjeta, changedTs).

**TARIFA** (Codigo, Precio, changedTs).

**GENERO** (Identificador, Tipo, changedTs).

**CONTENIDO** (Identificador, Titulo, Desc, Url\_Imagen, Actores, Punt\_Media, Fech\_Estreno, Duracion, Director, changedTs, ID\_Genero, ID\_Tarifa).

FK ID\_Genero --> Genero(Identificador)

FK ID\_Tarifa --> Tarifa(Identificador).

CORTO (IdentificadorCont).

PELICULA (IdentificadorCont, Disponible\_hasta).

SERIE (Identificador, Disponibilidad, Titulo, Desc).

CAPITULO (IdentificadorCont, Temporada, ID\_Serie

*FK ID\_Serie --> Serie(Identificador) ).*

CARRO COMPRA (Identificador, changedTs, ID\_Usuario

*FK ID\_Usuario --> Usuario(Identificador) ).*

FACTURA (Numero, Fecha, Imp\_Base, Imp\_IVA, changedTs, ID\_Usuario

*FK ID\_Usuario --> Usuario(Identificador) ).*

LINEA FACTURA (Identificador, changedTs, ID\_Carro, ID\_Factura

*FK ID\_Carro --> Carro\_Compra(Identificador)*

*FK ID\_Factura --> Factura(Numero) ).*

ARTICULOS FACTURAR (ID\_Linea, ID\_Contenido

*FK ID\_Linea --> Linea\_Factura(Identificador)*

*FK ID\_Contenido --> Contenido(Identificador) ).*

VALORA (ID\_Usuario, ID\_Contenido, Puntos

*FK ID\_Usuario --> Usuario(Identificador)*

*FK ID\_Contenido --> Contenido(Identificador) ).*

GESTIONA (ID\_Usuario, ID\_Contenido

*FK ID\_Usuario --> Usuario(Identificador)*

*FK ID\_Contenido --> Contenido(Identificador) ).*

ALQUILA (ID\_Usuario, ID\_Contenido

*FK ID\_Usuario --> Usuario(Identificador)*

*FK ID\_Contenido --> Contenido(Identificador) ).*

Contiene los siguientes procedimientos y funciones:

#### **AGREGAR\_ARTICULO\_CARRITO**

Descripción: El procedimiento AGREGAR\_ARTICULO\_CARRITO agrega un artículo al carrito de compras de un usuario. Inserta una nueva línea de factura y el artículo correspondiente en Articulos\_Facturar.

Parámetros:

- p\_id\_usuario (VARCHAR2): Identificador del usuario.
- p\_id\_contenido (NUMBER): Identificador del contenido que se va a agregar al carrito.

#### **CREAR\_CAPITULO**

Descripción: El procedimiento CREAR\_CAPITULO crea un nuevo capítulo en la base de datos.

Parámetros:

- Parámetros específicos para la creación de un capítulo.

### **CREAR\_CORTO**

Descripción: El procedimiento CREAR\_CORTO crea un nuevo corto en la base de datos.

Parámetros:

- Parámetros específicos para la creación de un corto.

### **CREAR\_PELICULA**

Descripción: El procedimiento CREAR\_PELICULA crea una nueva película en la base de datos.

Parámetros:

- Parámetros específicos para la creación de una película.

### **CREAR\_SERIE**

Descripción: El procedimiento CREAR\_SERIE crea una nueva serie en la base de datos.

Parámetros:

- Parámetros específicos para la creación de una serie.

### **CREAR\_USUARIO**

Descripción: El procedimiento CREAR\_USUARIO crea un nuevo usuario en la base de datos y asigna automáticamente un carrito de compras y una factura preestablecida.

Parámetros:

- Parámetros específicos para la creación de un usuario.

### **FINALIZAR\_PEDIDO**

Descripción: El procedimiento FINALIZAR\_PEDIDO finaliza el pedido de un usuario consolidando las líneas de factura en una única factura, actualizando los importes y eliminando las líneas de factura del carrito.

Parámetros:

- p\_id\_usuario (VARCHAR2): Identificador del usuario.

### **ACTUALIZAR\_CAPITULO**

Descripción: La función ACTUALIZAR\_CAPITULO actualiza la información de un capítulo existente en la base de datos y retorna el número de filas actualizadas.

Parámetros:

- Parámetros específicos para la actualización de contenido.

### **ACTUALIZAR\_CONTENIDO**

Descripción: La función ACTUALIZAR\_CONTENIDO actualiza la información de un contenido existente en la base de datos y retorna el número de filas actualizadas.

Parámetros:

- Parámetros específicos para la actualización de contenido.



## **ACTUALIZAR\_CORTO**

Descripción: La función ACTUALIZAR\_CORTO actualiza la información de un corto existente en la base de datos y retorna el número de filas actualizadas.

Parámetros:

- Parámetros específicos para la actualización de un corto.

## **ACTUALIZAR\_PELICULA**

Descripción: La función ACTUALIZAR\_PELICULA actualiza la información de una película existente en la base de datos y retorna el número de filas actualizadas.

Parámetros:

- Parámetros específicos para la actualización de una película.

## **ACTUALIZAR\_PUNTUACION**

Descripción: La función ACTUALIZAR\_PUNTUACION actualiza la puntuación de un contenido existente en la base de datos y retorna el número de filas actualizadas.

Parámetros:

- p\_id (NUMBER): Identificador del contenido.
- p\_tag\_usuario (VARCHAR2): Identificador del usuario.
- p\_puntuacion (FLOAT): Nueva puntuación del contenido.

## **ACTUALIZAR\_USUARIO**

Descripción: La función ACTUALIZAR\_USUARIO actualiza la información de un usuario existente en la base de datos y retorna el número de filas actualizadas.

Parámetros:

- Parámetros específicos para la actualización de un usuario.

## **ELIMINAR\_CONTENIDO**

Descripción: La función ELIMINAR\_CONTENIDO elimina un contenido existente en la base de datos y retorna el número de filas eliminadas.

Parámetros:

- p\_id (NUMBER): Identificador del contenido a eliminar.

## **ELIMINAR\_USUARIO**

Descripción: La función ELIMINAR\_USUARIO elimina un usuario existente en la base de datos y todas sus dependencias, y retorna el número de filas eliminadas.

Parámetros:

- p\_tag\_usuario (VARCHAR2): Identificador del usuario a eliminar.

Hemos creado 6 secuencias, una para cada tipo de ID para que las claves primarias de las tablas sean únicas:

- SEC\_IDGENERO
- SEC\_IDTARIFA
- SEQ\_FACTURA
- SEQ\_IDCONTENIDO
- SEQ\_LINEAID
- SEQ\_SERIE

Los triggers son los siguientes:

#### **ADMINISTRAR\_USUARIO**

Descripción: El trigger ADMINISTRAR\_USUARIO se ejecuta después de una operación de inserción o eliminación en la tabla Usuario. Su propósito es gestionar automáticamente las operaciones relacionadas con la creación y eliminación de usuarios, incluyendo la asignación de carritos de compra y facturas preestablecidas, así como la eliminación de todas las dependencias asociadas cuando se elimina un usuario.

Eventos:

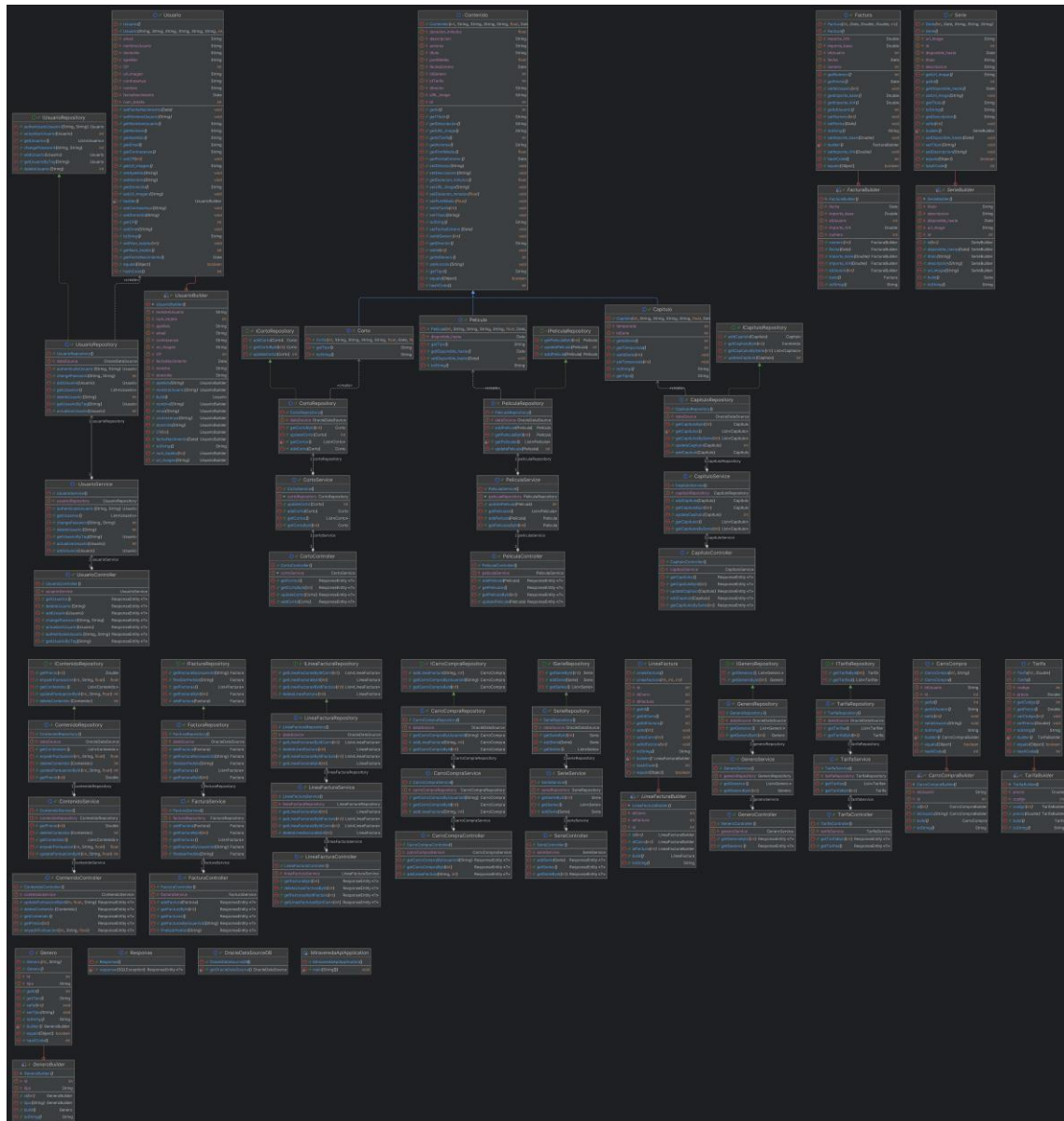
- AFTER INSERT ON Usuario
- AFTER DELETE ON Usuario

Acciones:

- Al Insertar un Usuario:
  - o Crea un carrito de compra para el nuevo usuario.
  - o Crea una factura preestablecida para el nuevo usuario.
- Al Eliminar un Usuario:
  - o Elimina los artículos facturados asociados a las líneas de la factura del usuario.
  - o Elimina las líneas de factura asociadas a las facturas del usuario.
  - o Elimina las facturas asociadas al usuario eliminado.
  - o Elimina el carrito de compra asociado al usuario eliminado.

La API está diseñada de la siguiente manera:

**Diagrama UML:**



Las llamadas son las siguientes:

## Contenido -----

Pedir todo el contenido: <http://IP:8080/contenido/>

Pedir el precio del contenido: <http://IP:8080/contenido/precio/id>

Actualizar puntuación por ID: <http://IP:8080/contenido/update/puntuacion/&id=id&punt=punt>

Borrar contenido: <http://IP:8080/contenido/delete/&id=id&tipo=tipo>

Añadir puntuación: <http://IP:8080/contenido/addPuntuacion/&id=id&user=tag&punt=punt>

Pedir Contenido por ID LineaFactura: <http://IP:8080/contenido/lineaFactura/id>

## Capitulo -----

Pedir todos los capítulos: <http://IP:8080/contenido/capitulo/>

Pedir capitulo por ID: <http://IP:8080/contenido/capitulo/id>

Añadir capitulo: <http://IP:8080/contenido/capitulo/add> (Requiere un objeto tipo Capitulo)

Pedir capitulo por serie: <http://IP:8080/contenido/capitulo/&serie=id>

Actualizar Capitulo: <http://IP:8080/contenido/capitulo/update> (Requiere un objeto tipo Capitulo)

Pedir Capítulos Alquilados por tag de usuario: <http://IP:8080/contenido/capitulo/alquilados/tag>

## Corto -----

Pedir todos los cortos: <http://IP:8080/contenido/corto/>

Pedir corto por ID: <http://IP:8080/contenido/corto/id>

Añadir corto: <http://IP:8080/contenido/corto/add> (Requiere un objeto tipo Corto)

Actualizar Corto: <http://IP:8080/contenido/corto/update> (Requiere un objeto tipo Corto)

Pedir Corto Alquilados por tag de Usuario: <http://IP:8080/contenido/corto/alquilados/tag>

## Película -----

Pedir todas las películas: <http://IP:8080/contenido/pelicula/>

Pedir película por ID: <http://IP:8080/contenido/pelicula/id>

Añadir película: <http://IP:8080/contenido/pelicula/add> (Requiere un objeto tipo Película)

Actualizar película: <http://IP:8080/contenido/pelicula/update> (Requiere un objeto tipo Película)

Pedir Películas alquiladas por tag de Usuario: <http://IP:8080/contenido/pelicula/alquilados/tag>

## Usuario -----

Pedir todos los usuarios: <http://IP:8080/usuario/>

Borrar un usuario: <http://IP:8080/usuario/delete/tag>

Actualizar un usuario: <http://IP:8080/usuario/update> (Requiere objeto Usuario Json da igual que los campos sean nulos, pero tienen que existir los campos)

Pedir usuario por tag: <http://IP:8080/usuario/tag>

## MiraVereda

Verificar login usuario: <http://IP:8080/usuario/login/&user=tag&password=pass>

Añadir un nuevo usuario: <http://IP:8080/usuario/add> (Requiere un objeto tipo Usuario)

Cambiar contraseña: [http://IP:8080/usuario/changePassword/\\$tag=tag&pass=pass](http://IP:8080/usuario/changePassword/$tag=tag&pass=pass)

### **Serie** -----

Pedir todas las series: <http://IP:8080/serie/>

Pedir serie por id: <http://IP:8080/serie/id>

Añadir serie: <http://IP:8080/serie/add> (Requiere un objeto tipo Serie)

Actualizar Serie: <http://IP:8080/serie/update> (Requiere un objeto tipo Serie)

Borrar Serie: <http://IP:8080/serie/delete/id>

### **CarroCompra** -----

Pedir carro por ID: <http://IP:8080/carro/id>

Pedir carro por tag de usuario: <http://IP:8080/carro/&user=tag>

Añadir LineaFactura: <http://IP:8080/carro/addLinea/&user=tag&idCont=id>

### **Factura** -----

Pedir Facturas: <http://IP:8080/factura/>

Pedir Factura por id: <http://IP:8080/factura/id>

Pedir Factura por tag de usuario: <http://IP:8080/factura/&user=tag>

Añadir Factura: <http://IP:8080/factura/add> (Requiere un objeto de tipo Factura)

FinalizarPedido por tag de Usuario: <http://IP:8080/factura/finalizar/tag>

### **LineaFactura** -----

Pedir LineaFactura por ID: <http://IP:8080/lineaFactura/id>

Pedir LineaFacturas por ID de Factura: <http://IP:8080/LineaFactura/&factura=id>

Pedir LineaFactura por ID de Carro: <http://IP:8080/LineaFactura/&carro=id>

Eliminar LineaFactura por ID: <http://IP:8080/LineaFactura/id>

### **Genero** -----

Pedir Generos: <http://IP:8080/genero/>

Pedir Genero por ID: <http://IP:8080/genero/id>

### **Tarifa** -----

Pedir Tarifas: <http://IP:8080/tarifa/>

Pedir Tarifa por ID: <http://IP:8080/tarifa/id>

La Aplicación Móvil está formada de la siguiente manera:

**Diagrama UML:**



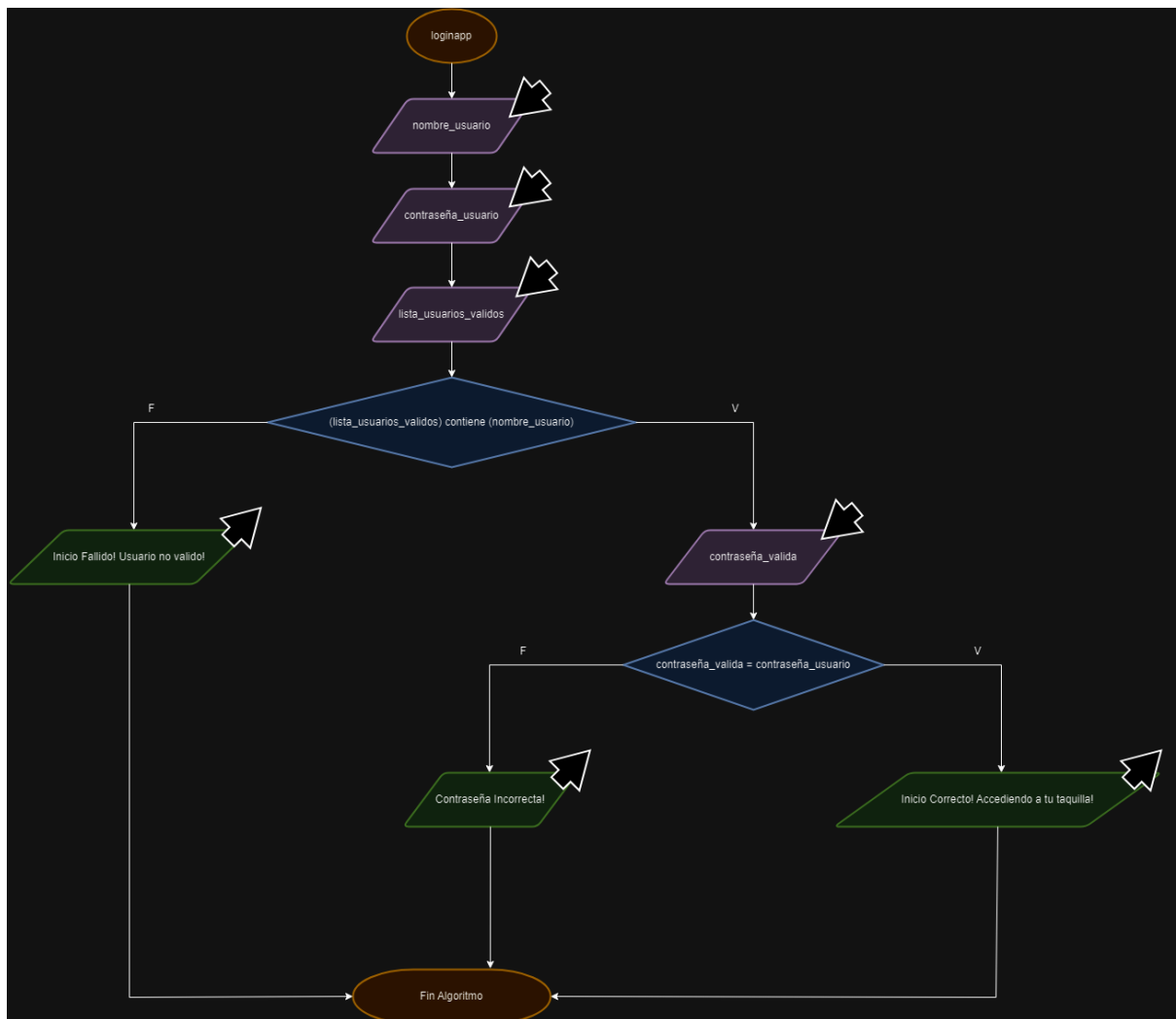
Las Actividades de la APP son las siguientes:

**--Main Activity--**

The image shows a mobile application login screen with a dark blue gradient background. It features two input fields: the first is for the username, labeled 'Nombre de usuario' with a person icon, and the second is for the password, labeled 'Contraseña' with three dots. Below these fields are three buttons with a pink-to-blue gradient: 'INICIAR SESIÓN', 'CREAR CUENTA', and 'HE OLVIDADO MI CONTRASEÑA'.

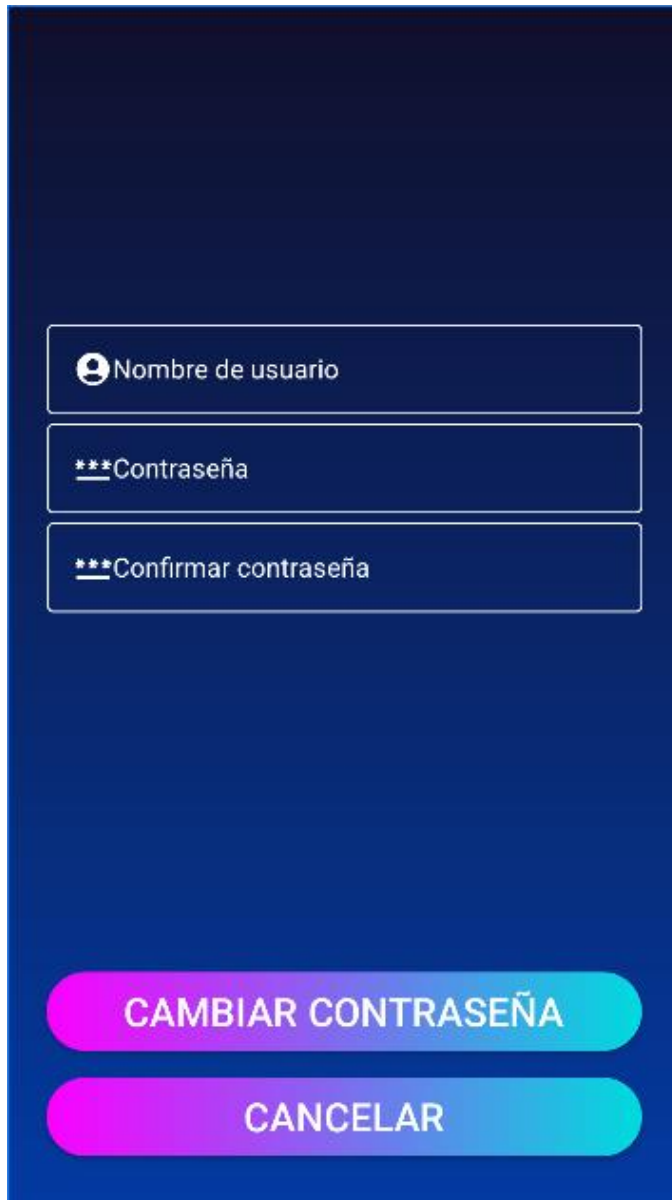
Es la primera actividad que verá el usuario al abrir la aplicación, esta actividad consta de un Inicio de Sesión (al realizar un Inicio de Sesión exitoso, lleva al usuario a ContenidoActivity) con las opciones extra de añadir contraseña (que lleva a la actividad ReiniciarContrasenyaActivity) y crear una nueva cuenta (que lleva a la actividad CrearCuentaActivity).

Ejemplo diagrama de flujo:





--Reiniciar Contraseña Activity--

The image shows a mobile application screen for resetting a password. The background is a dark blue gradient. At the top, there is a header area with a title bar. Below the header, there are three input fields stacked vertically. The first field is labeled 'Nombre de usuario' with a user icon. The second field is labeled '\*\*\*Contraseña' with a password icon. The third field is labeled '\*\*\*Confirmar contraseña' with a password icon. At the bottom of the screen, there are two large, rounded buttons. The top button is labeled 'CAMBIAR CONTRASEÑA' and the bottom button is labeled 'CANCELAR'. Both buttons have a gradient from purple to blue.

Nombre de usuario

\*\*\*Contraseña

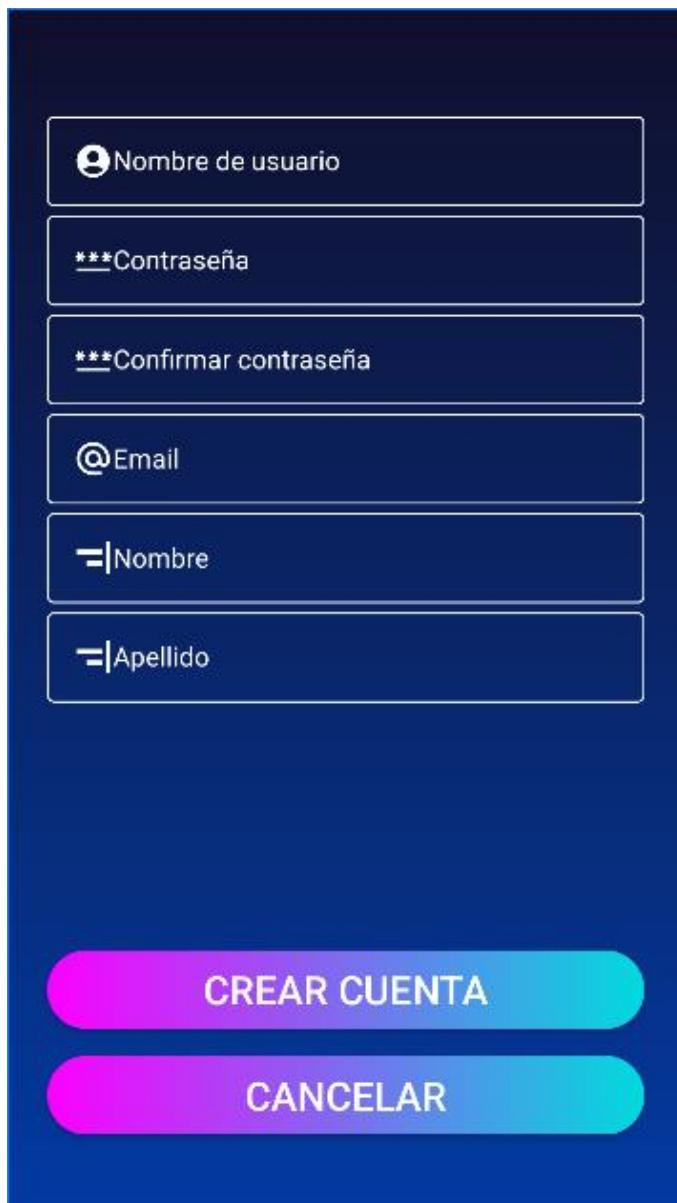
\*\*\*Confirmar contraseña

CAMBIAR CONTRASEÑA

CANCELAR

Cuando el usuario quiera reiniciar su contraseña llamará a esta actividad y a partir de un nombre de usuario podrá cambiarse la contraseña.

--Crear Cuenta Activity--



Nombre de usuario

\*\*\*Contraseña

\*\*\*Confirmar contraseña

@Email

|Nombre

|Apellido

CREAR CUENTA

CANCELAR

Quando el usuario quiera registrarse en la base de datos, llamará a esta actividad en la que introducirá los datos necesarios para crear la nueva cuenta.

**--Contenido Activity--**

Una vez iniciada la sesión correctamente, el usuario entrará en esta actividad la cual le mostrará todo el contenido disponible, para ver diferente tipo de contenido (películas, cortos o series), utilizará los filtros de la parte superior de la actividad ya que solo puede visualizar un listado de un tipo de contenido al mismo tiempo.

Cuando el usuario quiera visualizar un contenido, tendrá que pulsar encima de este (esto le direccionará a la actividad ContenidoAmpliadoActivity).

--Contenido Ampliado Activity--



## Harry Potter y la piedra filosofal

El día en que cumple once años, Harry Potter descubre que es hijo de dos conocidos hechiceros, de los que ha heredado poderes mágicos. Deberá acudir entonces a una famosa escuela de magia y hechicería: Howards.



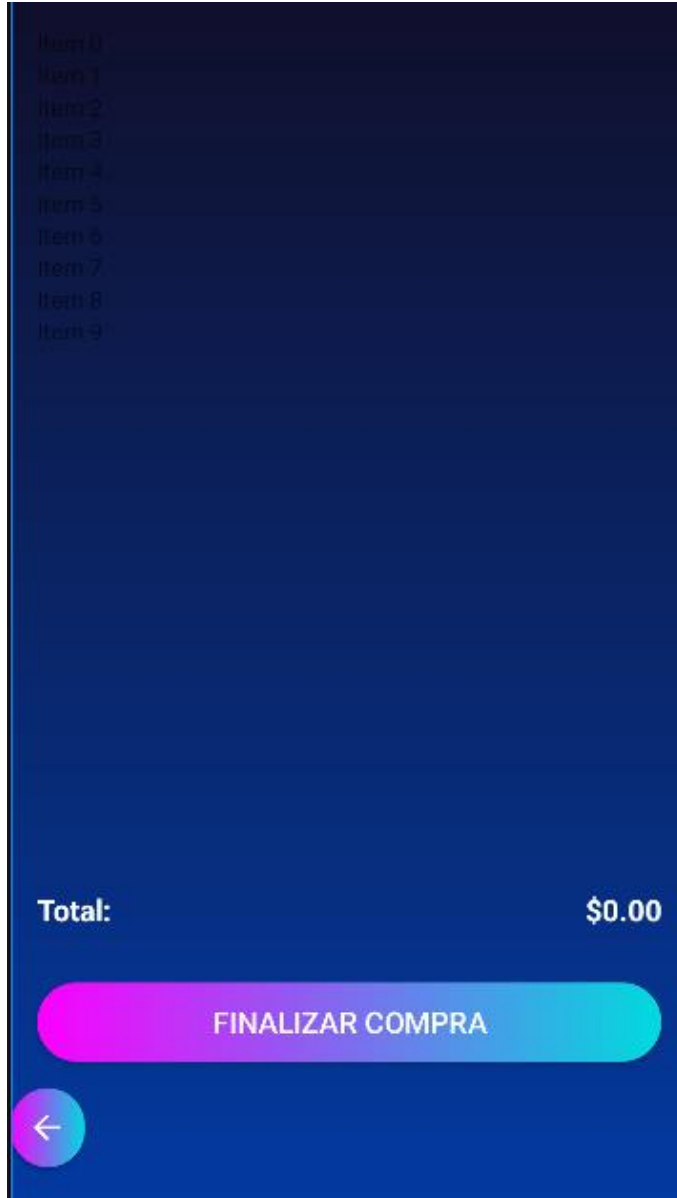
Después de seleccionar un contenido específico, se llegará a esta actividad que le mostrará en detalle toda la información de ese contenido (título, descripción, puntuación, etc.). Además, contiene un botón de retorno que le devuelve al listado general de contenido, y un botón de carro de compra, que solo aparece en caso de que no tenga este contenido y al presionarlo añadirá una línea de factura con ese contenido. Si lo tiene alquilado, se le mostrará un botón de reproducción.

Además, en caso de que sea una serie, en la parte inferior se le mostrará un listado con los capítulos de esta divididos por temporadas, pudiendo pulsar cada uno de estos para verlos en detalle (esto dirigirá al usuario a CapituloActivity).

### --Capitulo Activity--

Igual que la actividad anterior, esta muestra la información detallada del capítulo, además de la posibilidad de comprar o reproducir dependiendo de tener o no el contenido alquilado.

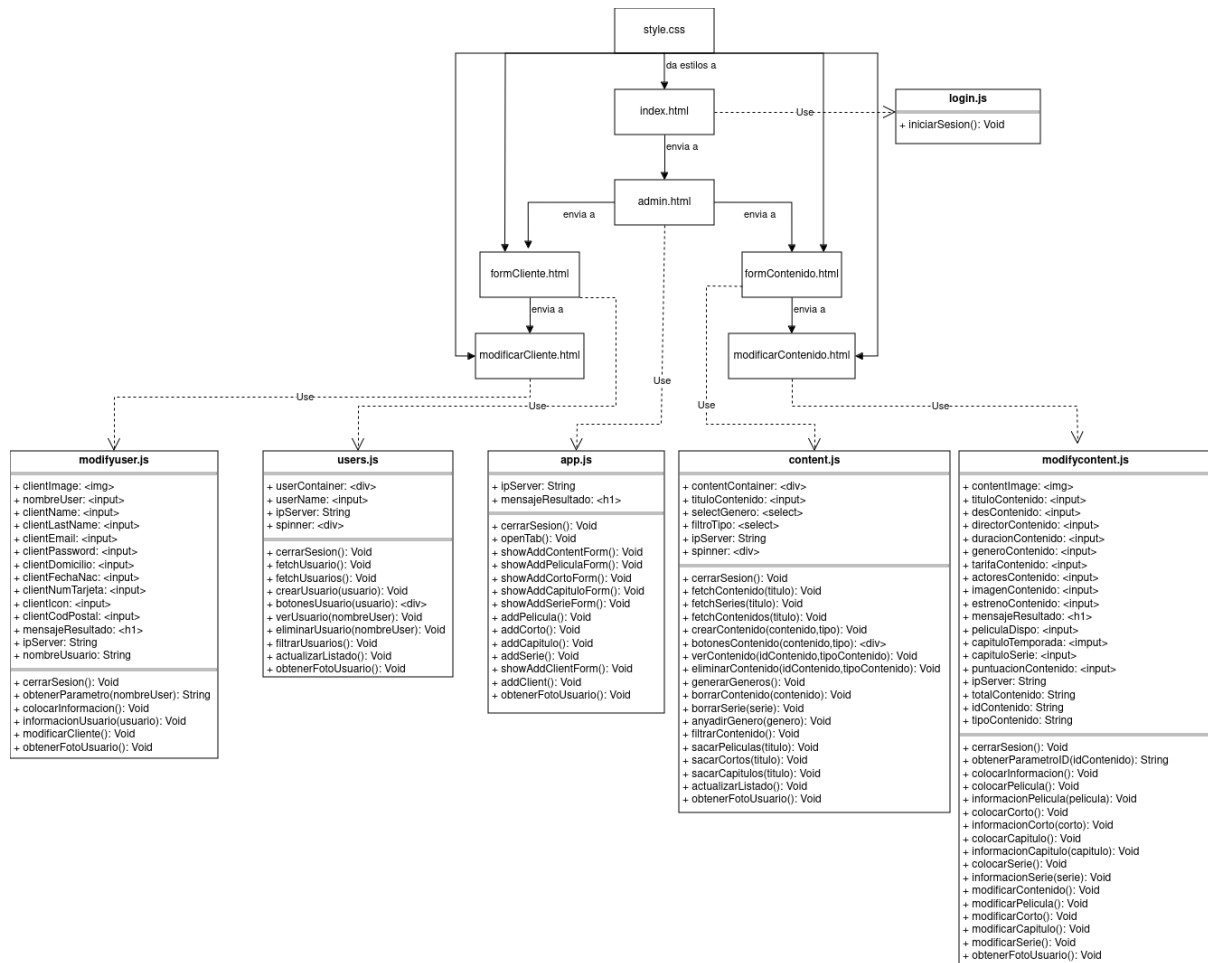
### --Carrito Activity--



En caso de que el usuario quiera finalizar su compra, se llamará a esta actividad, esta mostrara todo el contenido que va a alquilar el usuario (imagen, titulo y precio) y un botón de finalizar pedido, que creará la factura y permitirá al usuario alquilar el contenido que ha seleccionado.


La Página WEB está diseñada de la siguiente manera:

Diagrama UML:



Las páginas son las siguientes:

-- Index.html --

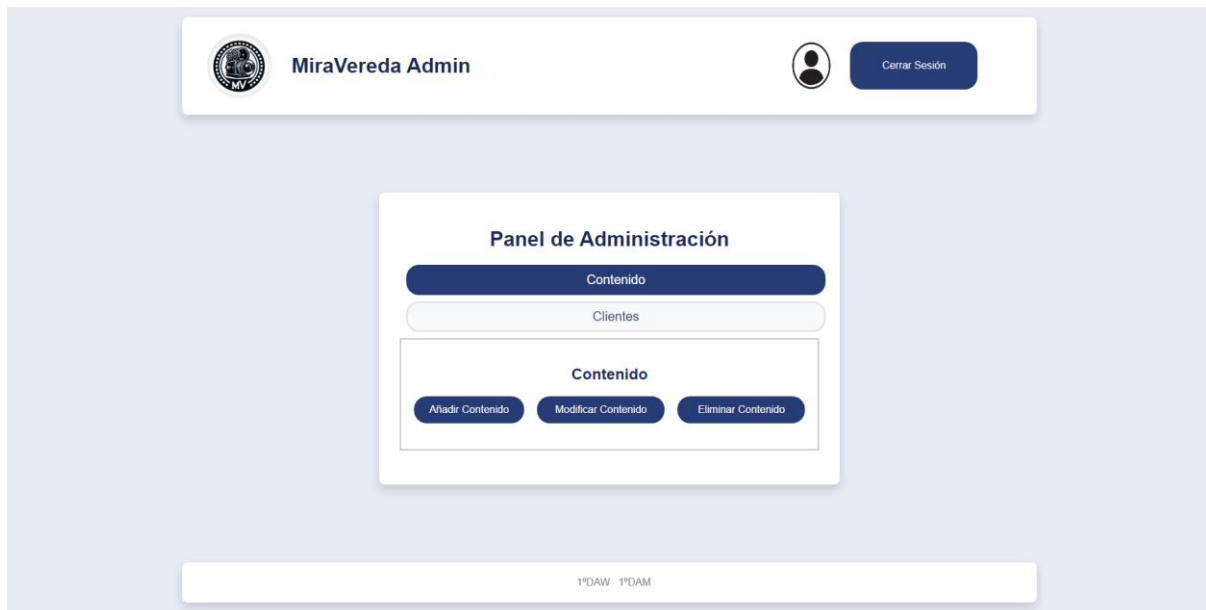


Es el login de la WEB, solo acepta usuarios administradores, si el inicio de sesión es correcto, envía al usuario a admin.html y le crea una variable de sesión en Local.

Ejemplo de pseudocódigo:

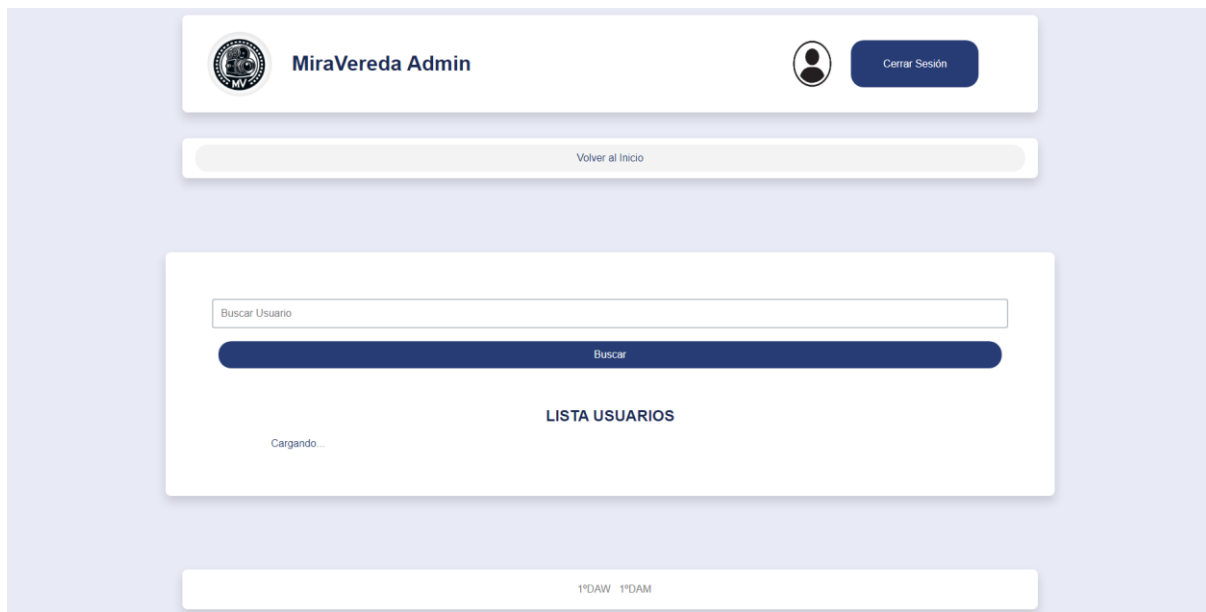
```
1  Algoritmo inicio_sesion_web
2  Leer nombre_usuario
3  Leer contraseña_usuario
4
5  Si nombre_usuario = "root" Entonces
6      Si contraseña_usuario = "root" Entonces
7          Escribir "Inicio Correcto! (dirigiendo a admin.html)"
8      SiNo
9          Escribir "Contraseña erronea!"
10     Fin Si
11 SiNo
12     Si nombre_usuario = "pepanav" Entonces
13         Si contraseña_usuario = "1234" Entonces
14             Escribir "Inicio Correcto! (dirigiendo a admin.html)"
15         SiNo
16             Escribir "Contraseña erronea!"
17         Fin Si
18     SiNo
19         Si nombre_usuario = "pepagarc" Entonces
20             Si contraseña_usuario = "1234" Entonces
21                 Escribir "Inicio Correcto! (dirigiendo a admin.html)"
22             SiNo
23                 Escribir "Contraseña erronea!"
24             Fin Si
25         SiNo
26             Escribir "Inicio Incorrecto! Este usuario no es administrador!"
27         Fin Si
28     Fin Si
29 Fin Si
30 FinAlgoritmo
31
```

-- Admin.html --



La página central de la WEB, en esta, el administrador puede crear Usuarios o Contenido (dividiéndose respectivamente en Película, Corto, Capitulo y Serie) y acceder a las páginas de modificación y eliminación (formCliente.html y formContenido.html).

## -- FormCliente.html --



La página del listado de Usuarios, en esta se obtienen todos los Usuarios y se le da al administrador la opción de modificarlos o eliminarlos, además, si el administrador así lo desea, puede filtrar por nombre de Usuario.

Si quiere modificar un Usuario, será enviado a modificarCliente.html, si quiere eliminarlo, simplemente debe pulsar el botón y este será eliminado de la BBDD.



**-- ModificarCliente.html --**

En esta página, se recibirá un id de Usuario, y a partir de este se mostrará la información del usuario respectivo para su modificación.

**-- FormContenido.html --**

La página de listado de Contenido, en esta se obtiene todo el Contenido y se le da al administrador la opción de modificarlos o eliminarlos, además, si el administrador así lo desea, puede filtrar por título del Contenido, tipo de Contenido (Película, Corto, Capítulo o Serie) y por Género.


Si quiere modificar un Contenido, será enviado a modificarContenido.html, si quiere eliminarlo, simplemente debe pulsar el botón y este será eliminado de la BBDD.


Ejemplo con tabla de decisión del funcionamiento del filtrado:


Situaciones
-------------

Filtra																	
Genero																	
Mismo																	
Genero																	
Contiene																	
Titulo																	
Mismo																	
Tipo																	
Acciones																	
Mostrar																	

## -- ModificarContenido.html --


MiraVereda Admin


Cerrar Sesión



Titulo

Descripción

Director

Duración en Minutos

Género

Tráiler

Actores

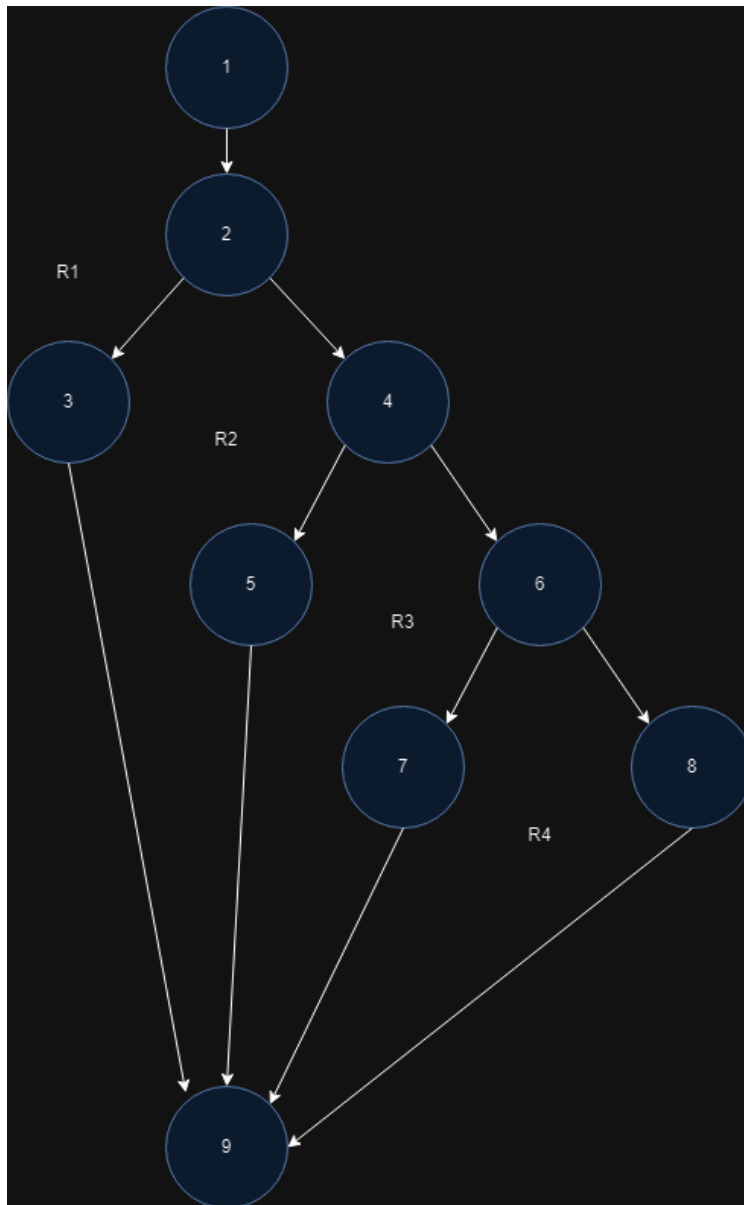
URL Imagen

URL del video

Actualizar Contenido

En esta página, se recibirá un id de Contenido y un tipo de Contenido, y a partir de estos, se colocará la información del Contenido para su modificación, dependiendo del tipo de Contenido se mostrarán más campos a modificar o menos equivalente a los atributos del tipo obtenido.

Aquí presentamos un grafo de flujo, su complejidad ciclomática y las pruebas a realizar a la hora de mostrar el contenido a modificar:

**Grafo de flujo:****Complejidad Ciclomática (CC):**

$$V(G) = NR = 4$$

**Conjuntos de caminos básicos y realización de pruebas:**

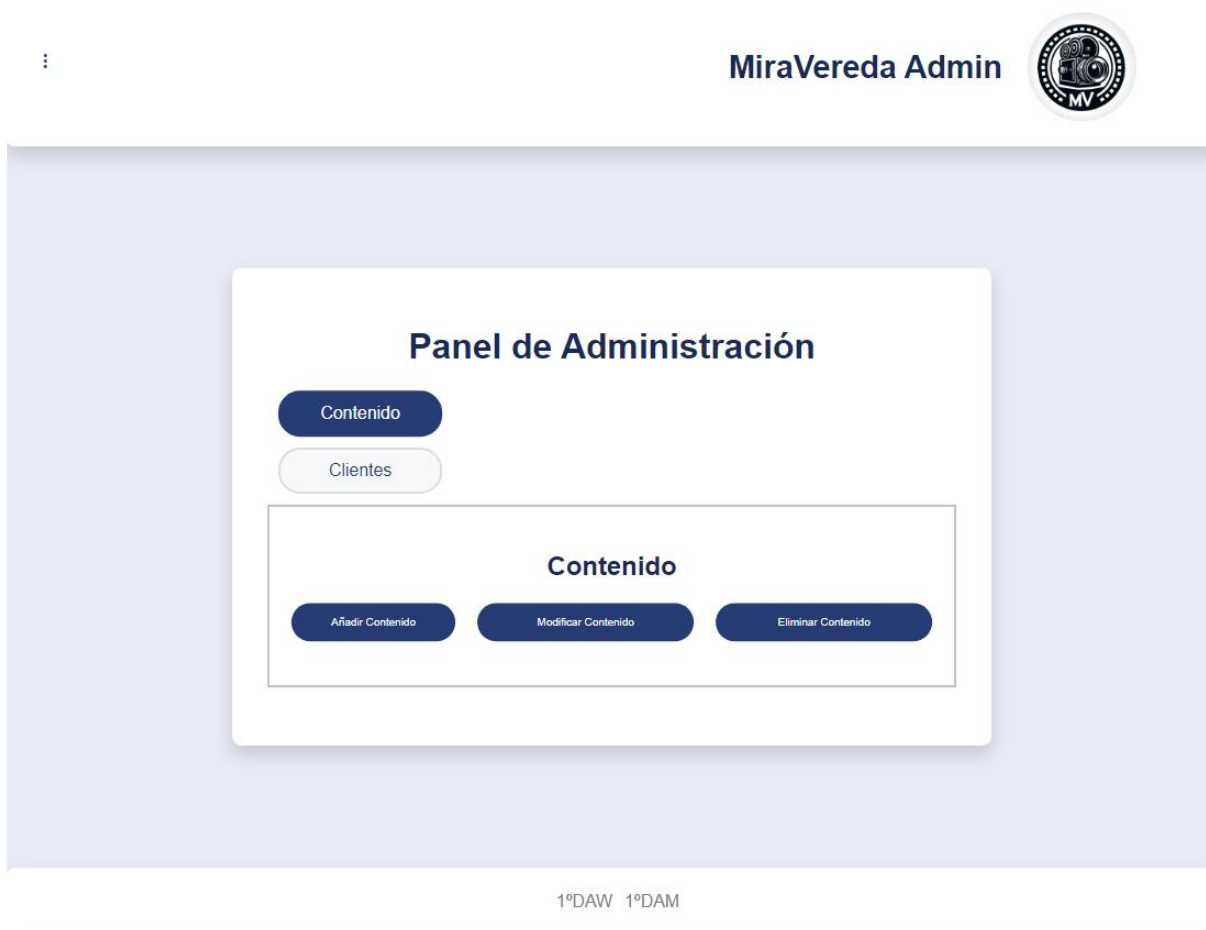
<b>C1</b>	1-2-3-9	Tipo = Película	Se insertan los datos de la película en el formulario añadiendo los campos necesarios.
<b>C2</b>	1-2-4-5-9	Tipo = Corto	Se insertan los datos del corto en el formulario añadiendo los campos necesarios.
<b>C3</b>	1-2-4-6-7-9	Tipo = Capítulo	Se insertan los datos del capítulo en el formulario añadiendo los campos necesarios.
<b>C4</b>	1-2-4-6-8-9	Tipo = Serie	Se insertan los datos de la serie en el formulario eliminando y añadiendo los campos necesarios

## -- Información Extra --

En cada página de la WEB (menos el login), el usuario podrá volver a la página anterior con un botón y podrá Cerrar Sesión en cualquier momento, retornándolo a index.html y eliminando su variable de sesión Local.



La página es responsive y adaptable, cambiando el tamaño del contenido dependiendo la resolución y cambiando el menú y el aspecto a partir de un tamaño de pantalla.



## IMPLANTACIÓN

Nuestro primer objetivo fue diseñar, estructurar y poner en marcha la BBDD de Oracle, la desarrollamos en SQL Developer, primero realizamos en Entidad-Relación y su paso tablas y finalmente su DDL y DML. Tomamos esta decisión, para empezar a trabajar en una BBDD funcional lo antes posible.

Una vez finalizados los pasos anteriores, centramos nuestro objetivo en el PL/SQL y en la API (desarrollada en IntelliJ Idea), ya que las llamadas y los procedimientos y funciones de los que dependen eran cruciales para el entorno cliente y su funcionamiento.

A la par que realizábamos la parte funcional del servidor, realizamos el diseño de la parte cliente (los layouts de la Aplicación Móvil (desarrollada en Android Studio), y la estructura del HTML y CSS de la página WEB de Administración (desarrollada en Visual Studio Code)).

Una vez realizado el CRUD fundamental del servidor, nos centramos en realizar la funcionalidad básica de la parte de cliente a la par que añadir las funcionalidades más complejas al apartado de servidor.

Con este proceso de desarrollo, logramos el desarrollo de la parte servidor y cliente a la par la mayor parte del tiempo, permitiéndonos avanzar en la funcionalidad más compleja de la parte de cliente rápidamente.

## RECURSOS

### Herramientas Hardware:

- Uso de 4 ordenadores simultáneamente para el desarrollo del proyecto
- 1 ordenador levantando la API
- 1 ordenador con el servidor WEB
- 1 ordenador/teléfono con la Aplicación Móvil.

### Herramientas Software:

- IntelliJ Idea (desarrollo de la API)
- Android Studio (desarrollo de la Aplicación Móvil)
- SQL Developer (desarrollo de la BBDD Oracle)
- Visual Studio Code (desarrollo de la WEB de Administración)
- Draw IO (usado para la creación de los Diagramas)

## CONCLUSIONES

Grado de consecución de objetivos:

1. Funcionamiento de la BBDD (Entidad-Relación, Paso a tablas, DDL, DML)
2. PL/SQL de la BBDD
3. Funcionamiento básico de la API y Diseño de la parte cliente (Aplicación Móvil y WEB de Administración)
4. Funcionamiento avanzado de la API y funcionamiento básico de la parte cliente.
5. Funcionamiento avanzado de la parte cliente.
6. Documentación

Problemas encontrados:

- **BBDD:** Dificultades con la integridad referencial, alargando el código necesario por procedimiento significativamente. Dificultades a la hora de diseñar la estructura de los pedidos (factura y carro de compra) y la de contenido. Han sido necesarios cambios a lo largo del desarrollo para facilitar las llamadas y el funcionamiento de la parte cliente.
- **WEB:** Dificultades con la política CORS a la hora de realizar las llamadas a la API, siendo necesario añadir cabeceras en la API para permitir llamadas desde orígenes específicos. Dificultades para modificar y añadir contenido, ya que se divide en subcategorías con atributos añadidos, siendo necesario una personalización compleja de cada apartado en la WEB.
- **API:** Dificultades con la cantidad de llamadas necesarias y la complejidad de cada una.
- **APP MOVIL:** Dificultades a la hora de crear los RecyclerView debido a la complejidad de los elementos que recibía. Dificultades a la hora de realizar las llamadas y recibir correctamente lo necesario para realizar la actividad. Dificultades a la hora de implementar el Carro de Compra y sus funcionalidades ya que para mejorar la experiencia visual del usuario era necesario realizar un código mucho más complejo del previsto.

Mejoras

Se mejorarían el aspecto WEB de la página de Administración, así como facilitar al administrador su uso, cambiando el uso de ID al añadir y modificar contenido por selects de lo necesario (Tarifa, Serie, Genero).

Se mejoraría la interfaz y las funciones de navegación de la APP Móvil, para facilitar la experiencia del usuario al utilizarla, además de implementar las funciones planeadas restantes que no se han podido añadir por falta de tiempo.