



Universidad
de Huelva



DESARROLLO DE APLICACIONES WEB

ANTONIO ABAD HERNÁNDEZ GÁLVEZ

4º GRADO DE INGENIERÍA INFORMÁTICA



ÍNDICE

1. Explicación del Proyecto	3
1.1. Tecnologías Utilizadas	3
2. Roles	4
3. Diagrama de Clases	5
4. Paquetes y Carpetas del Proyecto	6
5. Flujo de la Aplicación	7
6. Códigos de Error	12
7. Repositorio GitHub	12



1. Explicación del Proyecto

ToniCar es una aplicación web desarrollada cuyo objetivo es gestionar un concesionario online.

La plataforma permite a los usuarios visualizar vehículos disponibles, registrarse, iniciar sesión, realizar búsquedas y compras, gestionar datos personales y consultar información sobre los coches ofertados.

El sitio web está diseñado siguiendo una arquitectura **MVC** (Modelo–Vista–Controlador) basada en Java EE y Jakarta EE, utilizando JSP, Servlets y JSTL para el procesamiento dinámico.

La aplicación se conecta a una base de datos relacional donde se almacenan *usuarios*, *coches*, *marcas* y *compras*.

El proyecto busca simular un portal real de compra/venta de coches, incorporando funcionalidades típicas de un sistema profesional: autenticación segura, filtrado dinámico, autorización, gestión de sesiones y panel de administrador.

1.1. Tecnologías Utilizadas

Para el **Frontend**:

- HTML5.
- CSS3 para el estilo visual de la web
- JavaScript para la validación de datos.
- Bootstrap v5.3.8 para un diseño responsive.
- Plantillas JSP para reutilizar encabezados, menús y pies de página en las diferentes vistas.
- JSTL para renderizar contenido dinámico en tablas y tarjetas.

Para el **Backend**:

- Java 21 (Jakarta EE 10)
- Servlets para los controladores.
- JSP como tecnología de las vistas.
- JSTL y expresiones EL para la parte dinámica (comunicación entre los controladores y las vistas).
- JavaDB/Java Derby como base de datos.
- JDBC para la comunicación con la base de datos.
- EclipseLink como ORM (mapeo objeto-relacional) que implementa JPA (Java Persistence API) como proveedor de persistencia (para manejar la unidad de persistencia).
- Gestión de sesiones HTTP con HttpSession.

Servidor y Entorno de desarrollo (IDE):

- GlassFish v7.0.14 como servidor de aplicaciones web (servidor local)
- NetBeans 22 como IDE
- Arquitectura MVC (Modelo-Vista-Controlador)



2. Roles

1. Administrador de la Aplicación

Tiene acceso a ver el catálogo de coches y realizar operaciones CRUD sobre los vehículos.

El administrador **NO** puede realizar una compra de un coche, ya que podría bajar el precio de un coche o ponerlo de oferta para poder comprarlo a un precio más bajo (no queremos que eso ocurra en nuestra aplicación).

Si un administrador quiere comprar un coche, deberá iniciar sesión con su cuenta de cliente para poder realizar la compra (autorización controlada en la aplicación).

2. Cliente

Tiene acceso a ver el catálogo de coches, realizar compras de vehículos, consultar los pedidos o compras realizadas en la aplicación y cerrar su sesión.

Al realizar una compra, en una transacción, los clientes solo pueden comprar un coche, es decir, un cliente puede haber realizado varias compras, pero en cada compra, solo puede comprar un coche.

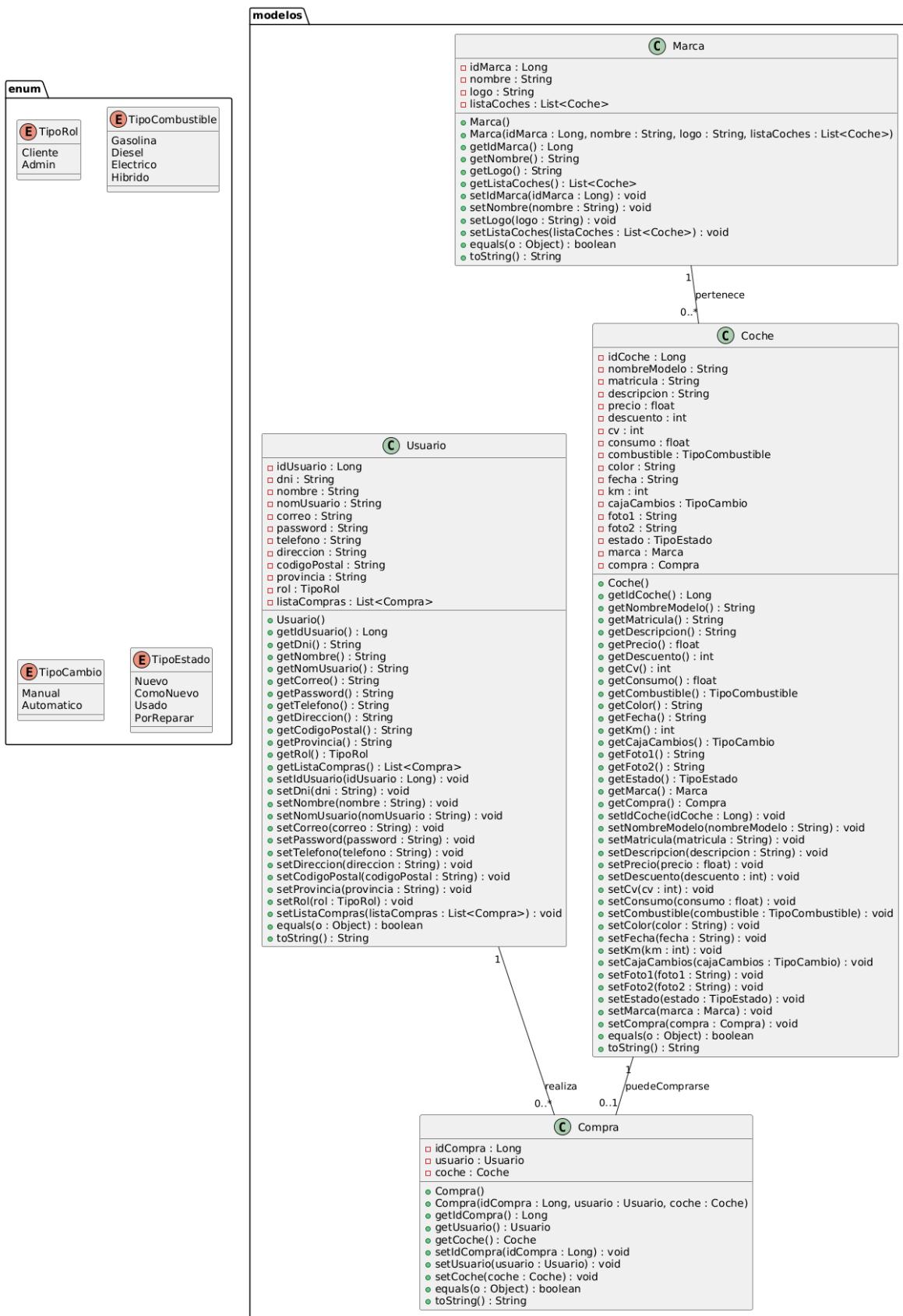
3. Usuario sin sesión iniciada

Tiene acceso a la funcionalidad de iniciar sesión y registrarse como cliente, además de poder ver el catálogo de coches.

Si un usuario que NO tiene la sesión iniciada o no está registrado en el sistema quiere comprar un coche **NO** podrá hacerlo (autorización controlada en la aplicación) y se le redirigirá al formulario de iniciar sesión.

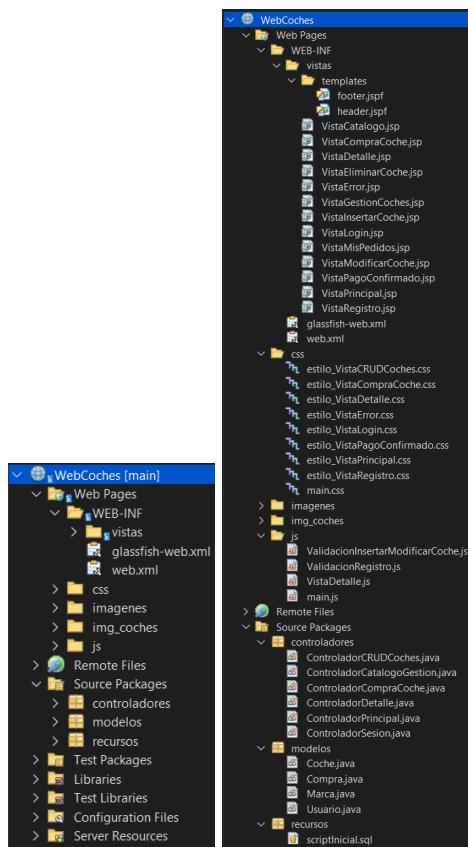


3. Diagrama de Clases





4. Paquetes y Carpetas del Proyecto





5. Flujo de la Aplicación

Si NO tenemos la sesión iniciada:

- Por defecto, al iniciar el programa (URL ‘WebCoches/inicio’):
ControladorPrincipal.java → VistaPrincipal.jsp
- Al pulsar en el botón de Iniciar Sesión (URL ‘WebCoches/sesion/login’):
ControladorSesion.java → VistaLogin.jsp
- Al pulsar en el botón de Registrarse (URL ‘WebCoches/sesion/registrar’):
ControladorSesion.java → VistaRegistro.jsp
- Al pulsar la opción Catálogo de Coches (URL ‘WebCoches/catalogo’):
ControladorCatalogoGestion.java → VistaCatalogo.jsp

Si tenemos la sesión iniciada con rol de cliente:

- Al pulsar sobre la imagen de perfil y seleccionar la opción Mis Pedidos
(URL ‘WebCoches/sesion/misPedidos’): ControladorSesion.java → VistaMisPedidos.jsp
- Al pulsar sobre la imagen de perfil y seleccionar la opción Cerrar Sesión
(URL ‘WebCoches/sesion/cerrarSesion’): ControladorSesion.java
- Dentro del Catálogo de Coches, al pulsar el botón de Ver Detalle
(URL ‘WebCoches/detalle/idCoche’): ControladorDetalle.java → VistaDetalle.jsp
- Dentro del Detalle del Coche, al pulsar el botón Comprar Ahora
(URL ‘WebCoches/comprar/idCoche’): ControladorCompraCoche.java → VistaCompraCoche.jsp
- Dentro del resumen de la Compra del Coche, al pulsar el botón Confirmar Compra
(URL ‘WebCoches/confirmarCompra/idCoche/idUsuario’):
ControladorCompraCoche.java → VistaPagoConfirmado.jsp

Si tenemos la sesión iniciada con rol de administrador:

- Al pulsar la opción Gestión de Coches (URL ‘WebCoches/gestion’):
ControladorCatalogoGestion.java → VistaGestionCoches.jsp
- Dentro de la Gestión de Coches, al pulsar el botón Insertar Coche
(URL ‘WebCoches/gestion/insertar’): ControladorCRUDCoches.java → VistaInsertar.jsp
- Dentro de la Gestión de Coches, al pulsar el botón Modificar Coche
(URL ‘WebCoches/gestion/modificar/idCoche’): ControladorCRUDCoches.java → VistaModificar.jsp
- Dentro de la Gestión de Coches, al pulsar el botón Eliminar Coche
(URL ‘WebCoches/gestion/eliminar/idCoche’): ControladorCRUDCoches.java → VistaEliminar.jsp



MVC	ELEMENTO	DESCRIPCIÓN
VISTAS (.jsp)	VistaPrincipal	Contiene un carrusel dividido en 2 (en el lado izquierdo van rotando varias imágenes y en el lado derecho van rotando diversos textos)
	VistaLogin	Contiene un formulario con los datos de inicio de sesión del usuario. Si NO tenemos una cuenta, tiene un acceso al botón para Registrarse.
	VistaRegistro	Contiene un formulario con los datos de registro para un nuevo usuario. Si los datos introducidos NO son correctos (Validación JS y en el Controlador), muestra los campos que estaban introducidos antes del error, NO aparece el formulario vacío de nuevo. Cada campo que escribamos irá acompañado de un símbolo de éxito o error. En caso de error, para saber detalladamente qué es lo que está mal en ese campo debemos dejar el ratón encima del símbolo del error.
	VistaCatalogo	Contiene una tarjeta con algunos datos informativos (marca, nomModelo, precio, fecha, km, combustible, cajaCambios, cv) de todos los coches que hay en la BD disponibles para la venta y un botón para ver más detalles sobre el coche que queramos del catálogo. Si un coche del catálogo es comprado, dicho coche ya NO aparecerá en el catálogo. Si el coche NO tiene imagen, mostramos una por defecto ('silueta_coche.jpg') Si el coche tiene imagen, mostramos la primera ('foto1')
	VistaDetalle	Contiene todos los datos informativos e imágenes del coche que hemos seleccionado previamente del catálogo. Contiene un botón para realizar la compra de dicho coche.
	VistaCompraCoche	Contiene un resumen de la factura de compra del coche para revisar o confirmar que todos los datos sean correctos antes de realizar el pago.
	VistaPagoConfirmado	Muestra que el pago se ha realizado con éxito, muestra el código de compra (como si fuera el número de pedido) y muestra la matrícula y la dirección de entrega del coche (dirección del usuario). Tiene un botón para volver al catálogo de coches y otro para ver los pedidos realizados por el usuario.
	VistaGestionCoche	Contiene una tabla con todos los coches que hay en la BD que NO han sido comprados. En la esquina superior derecha de la tabla, encontramos el botón Insertar Coche para añadir un nuevo coche a la BD. Cada coche tiene algunos datos informativos para poder identificarlo de forma más sencilla (idCoche, matricula, color, precio, descuento, estado, foto1, foto2). Cada coche de la tabla va acompañado de un botón para eliminarlo del sistema o modificar sus datos. Entre el header y la tabla, se muestran los mensajes de éxito o error tras realizar una operación de inserción, modificación o eliminación de un coche.
	VistaInsertarCoche	Contiene un formulario con los datos del coche que queremos insertar. Si los datos introducidos NO son correctos (Validación JS y en el Controlador), muestra los campos que estaban introducidos antes del error, NO aparece el formulario vacío de nuevo. Cada campo que escribamos irá acompañado de un símbolo de éxito o error. En caso de error, para saber detalladamente qué es lo que está mal en ese campo debemos dejar el ratón encima del símbolo del error.



MVC	ELEMENTO	DESCRIPCIÓN
CONTROLADORES (.java)	VistaModificarCoche	Es el mismo formulario que el de la 'VistaInsertarCoche.jsp', pero con la diferencia de que los datos de cada campo están ya cargados en el formulario y que la matrícula del coche NO podemos modificarla.
	VistaEliminarCoche	Contiene una tabla con algunos datos informativos del coche que queremos eliminar antes de darle al botón de eliminar. Si presionamos el botón de eliminar, se abrirá una ventana modal para confirmar la eliminación de dicho coche por seguridad.
	VistaMisPedidos	Solo es accesible si el usuario tiene la sesión iniciada y tiene rol de cliente. Muestra una tabla resumen de todas las compras realizadas por el usuario. La dirección de entrega es la misma que la dirección del Usuario.
	VistaError	Contiene una tarjeta con los datos del error que se ha producido (código de error y mensaje de error). Si el error NO tiene ningún código, es decir, si el error tiene un código diferente a los que están controlados (400, 401, 403, 404, 500) el código de error es el 0.
CONTROLADORES (.java)	ControladorPrincipal	urlPatterns: /inicio Peticiones doGet: /inicio Peticiones doPost: - Muestra la VistaPrincipal.jsp en el arranque de la Aplicación, ya que he modificado las propiedades Run del proyecto y he hecho que el Context Path sea '/WebCoches' y que la Relative URL sea '/inicio'
	ControladorSesion	urlPatterns: /sesion/* Peticiones doGet: /login, /registrar, /cerrarSesion, /misPedidos Peticiones doPost: /login, /registrar, /comprobarNombreUsuario, /comprobarCorreo Tiene un método 'encriptarPassword()' para encriptar las contraseñas de cada usuario mediante el código hash MD5. Tiene varios métodos para realizar diferentes consultas en la BD necesarias para la lógica de la Aplicación y realiza la comprobación de los datos que le llegan del formulario de Registro antes de registrar el nuevo usuario en la BD (doble verificación de los datos).
	ControladorCatalogoGestion	urlPatterns: /catalogo, /gestion Peticiones doGet: /catalogo, /gestion Peticiones doPost: - Tiene el método 'obtenerCoches()' para realizar una consulta en la BD necesaria para la lógica de la Aplicación.
	ControladorDetalle	urlPatterns: /detalle, /detalle/* Peticiones doGet: /detalle Peticiones doPost: - Si el usuario intenta comprar el coche sin la sesión iniciada, se le redirigirá a la 'VistaLogin.jsp'.



MVC	ELEMENTO	DESCRIPCIÓN								
		<p>Si el usuario intenta comprar el coche con la sesión iniciada, se le redirigirá a la 'VistaCompraCoche.jsp'</p> <p>Tiene el método 'buscarCochePorId()' para realizar una consulta en la BD necesaria para la lógica de la Aplicación.</p>								
	ControladorCRUDCoches	<p>urlPatterns: /gestion/* Peticiones doGet: /insertar, /modificar, /eliminar Peticiones doPost: /insertar, /modificar, /eliminar, /comprobarMatricula</p> <p>Tiene varios métodos auxiliares y de consulta en la BD necesarios para la lógica de la Aplicación y realiza la comprobación de los datos que le llegan del formulario de Insertar o Modificar un Coche antes de actualizar el nuevo coche en la BD (doble verificación de los datos).</p>								
	ControladorCompraCoche	<p>urlPatterns: /comprar, /comprar/*, /confirmarCompra/* Peticiones doGet: /comprar, /confirmarCompra Peticiones doPost: -</p> <p>Tiene varios métodos auxiliares y de consulta en la BD necesarios para la lógica de la Aplicación.</p>								
MODELOS (.java)	Usuario	Mapea la entidad Usuario de la BD.								
	Marca	Mapea la entidad Marca de la BD.								
	Coche	Mapea la entidad Coche de la BD.								
	Compra	Mapea la entidad Compra de la BD.								
RECURSOS	scriptInicial.sql	<p>Carga un conjunto de datos de prueba en las diferentes entidades de la BD para testear la aplicación. Muy utilizado en etapas iniciales del desarrollo del proyecto.</p> <p>Contiene los siguientes usuarios creados para realizar test:</p> <table border="1"> <tr> <th>usuario</th><th>contraseña</th></tr> <tr> <td>cliente</td><td>1234</td></tr> <tr> <td>cliente2</td><td>1234</td></tr> <tr> <td>admin</td><td>1234</td></tr> </table> <p>cliente → NO tiene ningún coche comprado cliente2 → Tiene 2 coches comprados admin → Es el único administrador del sistema</p>	usuario	contraseña	cliente	1234	cliente2	1234	admin	1234
usuario	contraseña									
cliente	1234									
cliente2	1234									
admin	1234									
TEMPLATES (.jspx)	header	Contiene la barra de navegación, los botones para Iniciar Sesión y Registrarse y un ícono para las acciones del perfil del usuario una vez que hemos iniciado sesión.								
	footer	Es el pie de la página web que contiene información sobre el proyecto, sobre el autor, enlaces de interés y un mapa de donde se encuentra la ETSI.								



MVC	ELEMENTO	DESCRIPCIÓN
ESTILOS (.css)	estilo_VistaPrincipal	Define la hoja de estilos utilizada en VistaPrincipal.jsp
	estilo_VistaLogin	Define la hoja de estilos utilizada en VistaLogin.jsp
	estilo_VistaRegistro	Define la hoja de estilos utilizada en VistaRegistro.jsp
	estilo_VistaDetalle	Define la hoja de estilos utilizada en VistaDetalle.jsp
	estilo_VistaComprarCoche	Define la hoja de estilos utilizada en VistaComprarCoche.jsp
	estilo_VistaPagoConfirmado	Define la hoja de estilos utilizada en VistaPagoConfirmado.jsp
	estilo_VistaError	Define la hoja de estilos utilizada en VistaError.jsp
	estilo_VistaCRUDCoches	Define la hoja de estilos de las Vistas de las operaciones CRUD del Admin de la aplicación ('VistaInsertarCoche.jsp', 'VistaEliminarCoche.jsp' y 'VistaModificarCoche.jsp')
	main	Define la hoja de estilos genérica de la Aplicación ToniCar, es decir, es utilizada por elementos comunes en diferentes vistas (.jsp)
IMAGENES (.jpg, .png, .webp)		Contiene las imágenes genéricas de la aplicación (carrusel de la 'VistaPrincipal.jsp' e iconos de los perfiles).
IMG_COCHES (.jpg, .png, .webp)		Contiene las imágenes de los distintos vehículos del catálogo. (En esta carpeta se insertan las imágenes de los nuevos vehículos que el administrador inserta en la Aplicación).
SCRIPTS (.js)	main	Contiene las funciones necesarias para que las imágenes del carrusel de la 'VistaPrincipal.jsp' vayan cambiando con el tiempo.
	ValidacionRegistro	Funciones para validar los campos del formulario de Registro. Contiene un fetch para los campos 'correo' y 'nomUsuario'. Además, selecciona la provincia automáticamente en base al Código Postal introducido.
	Validacion InsertarModificarCoche	Funciones para validar los campos del formulario de Insertar o Modificar Coche (Solo accesible por el Administrador) Contiene un fetch para el campo 'matrícula'.
	VistaDetalle	Contiene las funciones necesarias para que las imágenes del carrusel de la 'VistaDetalle.jsp' puedan cambiar cada 4s. Se utiliza para mostrar las 2 imágenes de cada coche pudiendo pulsar los botones para ver la siguiente foto o la anterior de forma cíclica. Si el coche solo tiene 1 foto, el carrusel NO permitirá pasar, ni mostrar la foto 2, ya que NO existe.



6. Códigos de Error

1. Error 401 (Unauthorized)

El Usuario sin sesión iniciada, intenta acceder a una opción que requiere de un inicio de sesión previo.

2. Error 403 (Forbidden)

El Usuario intenta acceder a una funcionalidad que requiere de un rol distinto del suyo a través de la aplicación o modificando la URL (inyección de parámetros).

3. Error 404 (Not Found)

El Usuario intenta acceder a una URL que NO existe.

4. Error 500 (Internal Server Error)

El desarrollador de la Aplicación comete un error de programación y el servidor de aplicaciones lanza alguna excepción no atrapada.

7. Repositorio GitHub

Contiene un [readme.md](#) con información breve del proyecto y el código fuente disponible para descargar.

Para descargar el código: Code → Download ZIP

Link directo al repositorio → https://github.com/antonioabadpro/Proyecto_DAW