

PROYECTO FINAL

Ciclo Superior de Desarrollo de
Aplicaciones Multiplataforma
(DAM2)

Curso 2022/2023

Autor/a: Carlos Rodrigo Pla



GARAGE GO
EL FUTURO DE TU TALLER

Descripción del Proyecto:	3
Aplicación de Escritorio para el Personal del Taller:	3
Aplicación Web para Clientes:	4
Ventajas de la Aplicación Web Integrada:	4
Análisis y Justificación del Proyecto de Gestión de Talleres de Coches	5
Objetivos y Justificación:	6
Algunas de las características y ventajas de esta integración:	8
Análisis de Mercado de Proyectos Similares en la Gestión de Talleres de Coche	9
Competidores directos:	9
Aspectos que destacan mi proyecto:	10
Análisis del sistema actual y justificación de la solución propuesta:	11
Análisis del sistema	12
Datos de entrada y salida. Diagramas de casos de uso	13
Diseño de la base de datos	15
Justificación	15
Diseño	16
Diccionario de clases	20
Implementación del sistema	27
Estructura del proyecto	27
Interfaces de usuario	43
Seguridad	49
OTP	51
Verificación de la aplicación	52
Abstract	59
Partes del proyecto que han supuesto mayor dificultad	60
Intento fallido de establecer un chat entre android y la aplicación de escritorio	61
Intento fallido de conexión directa a base de datos SQL remota desde Android	62
Conclusión	63
Tecnologías utilizadas:	64
Presupuesto Detallado para el Desarrollo, Implantación y Mantenimiento del Proyecto de Gestión de Talleres de Coche	65
Aspectos a tener en cuenta para realizar el presupuesto:	65
Desarrollo del Proyecto	65
Implantación del Proyecto	65
Mantenimiento del Proyecto	66
Cuantificación	66
Presupuesto	67
Manual	68
Bibliografía	69
Anexo Proyecto SPRING web Application	70

Descripción del Proyecto:

El proyecto consiste en el desarrollo de una aplicación de gestión de talleres de coche que integra una aplicación de escritorio y una aplicación web desarrollada con Spring. La aplicación de escritorio está diseñada para el personal del taller, permitiendo la gestión de clientes, vehículos, partes, control de ingresos, comunicación interna y control horario. Por otro lado, la aplicación web, desarrollada con Spring, se ha creado para ofrecer a los clientes del taller un acceso seguro y conveniente a sus datos.

Aplicación de Escritorio para el Personal del Taller:

La aplicación de escritorio desarrollada para el personal del taller de coche es una herramienta completa y eficiente que permite gestionar todas las actividades relacionadas con la gestión y operación del taller. Con una interfaz intuitiva y funcionalidades avanzadas, la aplicación ofrece las siguientes características principales:

- 1. Registro y Gestión de Clientes y Vehículos:** El personal del taller puede registrar y mantener una base de datos actualizada de los clientes y sus vehículos. Esta funcionalidad permite almacenar información detallada, como datos de contacto, historial de servicios, modelos de vehículos y fechas de mantenimiento.
- 2. Creación y Seguimiento de Partes:** La aplicación permite generar partes de trabajo para cada vehículo que ingresa al taller. Los partes contienen información detallada sobre los servicios solicitados, las reparaciones realizadas, las piezas reemplazadas y los costos asociados. El personal del taller puede dar seguimiento a los partes en tiempo real, actualizar su estado y asignar tareas a los miembros del equipo correspondientes.
- 3. Control de Ingresos y Facturación:** La aplicación facilita el control de los ingresos generados por los servicios prestados en el taller. Permite la generación de facturas y presupuestos personalizados, incluyendo el desglose detallado de los servicios, los costos y los impuestos aplicables. Además, se pueden realizar seguimientos de pagos y gestionar el estado de las facturas pendientes.
- 4. Comunicación Interna:** La aplicación incluye un sistema de chat implementado con sockets que permite una comunicación fluida y en tiempo real entre los miembros del personal del taller. Esto facilita la colaboración, la resolución de problemas y la toma de decisiones rápida y efectiva.

5. Control Horario Integrado: El sistema de fichajes incorporado permite llevar un registro preciso de las horas trabajadas por cada empleado del taller. Esto simplifica la gestión de horarios, la asignación de recursos y la generación de informes de productividad.

La aplicación de escritorio se integra de manera fluida con la aplicación web desarrollada para los clientes, lo que permite una comunicación eficiente y una sincronización de datos en tiempo real entre el personal del taller y los clientes.

En resumen, la aplicación de escritorio desarrollada para el personal del taller de coche proporciona una solución completa y eficiente para gestionar todas las actividades clave de la gestión del taller. Desde el registro de clientes y vehículos hasta la creación de partes, control de ingresos, comunicación interna y control horario, la aplicación ofrece una experiencia intuitiva y potente que mejora la eficiencia y la productividad del taller.

Aplicación Web para Clientes:

La aplicación web desarrollada con Spring proporciona a los clientes del taller una interfaz intuitiva y segura para acceder a sus datos. A través de esta aplicación web, los clientes tienen la posibilidad de ver información relevante sobre sus vehículos, como historial de mantenimiento, reparaciones realizadas y piezas reemplazadas. Además, pueden consultar partes en curso, presupuestos, facturas y realizar pagos de manera segura.

Una característica destacada de la aplicación web es la capacidad de firmar partes directamente desde la plataforma. Esto elimina la necesidad de trámites físicos y agiliza el proceso para los clientes. Además, se garantiza la seguridad y privacidad de los datos mediante medidas de autenticación y cifrado de última generación.

Ventajas de la Aplicación Web Integrada:

La integración de la aplicación web con la aplicación de escritorio presenta varias ventajas significativas:

1. Acceso conveniente: Los clientes del taller pueden acceder a sus datos y gestionar su información desde cualquier lugar y en cualquier momento a través de una interfaz web intuitiva y fácil de usar.

2. Mayor transparencia: Los clientes tienen una visibilidad clara de los servicios realizados en sus vehículos, los costes asociados y los documentos relacionados, lo que promueve la transparencia y la confianza en el taller.

3. Proceso eficiente de firmado y pago: La capacidad de firmar partes y realizar pagos en línea agiliza y simplifica los procesos para los clientes, evitando la necesidad de desplazamientos físicos y trámites adicionales.

En resumen, la integración de la aplicación web desarrollada con Spring en el proyecto de gestión de talleres de coche proporciona a los clientes una experiencia mejorada al permitirles acceder y gestionar sus datos de manera conveniente y segura. La capacidad de firmar partes y realizar pagos en línea agiliza los procesos y promueve la transparencia en las transacciones. Esta integración fortalece aún más la propuesta de valor del proyecto y su competitividad en el mercado.

Análisis y Justificación del Proyecto de Gestión de Talleres de Coches

1. Desarrollar una aplicación de escritorio para la gestión de talleres de coche que permita registrar clientes, vehículos, crear partes y controlar los ingresos (facturas y presupuestos).
2. Implementar diferentes roles de usuario para restringir el acceso a determinadas funciones según las responsabilidades y permisos asignados.
3. Incorporar un sistema de comunicación en tiempo real mediante un chat implementado con sockets para facilitar la colaboración entre los empleados.
4. Integrar un sistema de fichajes para controlar las horas trabajadas de los empleados directamente en la aplicación.
5. Utilizar una base de datos SQL remota para almacenar la información de manera segura y accesible desde diferentes ubicaciones.
6. Utilizar el sistema de almacenamiento distribuido Firebase para guardar documentos y archivos pesados como fotos y PDF.

La justificación de estos objetivos se basa en las siguientes razones:

1. Eficiencia y organización: La aplicación de gestión de talleres de coche permitirá optimizar los procesos y la eficiencia operativa al proporcionar herramientas para registrar clientes, vehículos, generar partes y controlar los ingresos. Esto reducirá la necesidad de gestionar manualmente la información en papel o utilizar sistemas dispersos.

2. Control de acceso y seguridad: La implementación de roles de usuario restringirá el acceso a funciones específicas según las responsabilidades asignadas a cada empleado. Esto garantizará la seguridad de los datos confidenciales y evitará el acceso no autorizado a información sensible.

3. Comunicación eficiente: El sistema de chat implementado con sockets permitirá una comunicación rápida y efectiva entre los empleados del taller, lo que facilitará la colaboración, la resolución de problemas y la toma de decisiones ágiles.

4. Control horario: El sistema de fichajes incorporado brindará un registro preciso de las horas trabajadas por los empleados, lo que permitirá una gestión más eficiente del personal y un seguimiento de los costos laborales.

5. Acceso remoto y seguridad de datos: La utilización de una base de datos SQL remota permitirá acceder a la información desde diferentes ubicaciones de manera segura y confiable. Además, la elección de Firebase como sistema de almacenamiento distribuido garantizará la seguridad y disponibilidad de los documentos y archivos pesados almacenados en la nube.

Objetivos y Justificación:

1- *Desarrollar una aplicación de escritorio para la gestión de talleres de coche que permita registrar clientes, vehículos, crear partes y controlar los ingresos (facturas y presupuestos).*

La gestión de talleres de coche implica lidiar con múltiples aspectos, como el registro de clientes, el seguimiento de vehículos, la creación de partes y el control de ingresos. Un sistema manual o herramientas dispersas pueden generar ineficiencias, errores y pérdida de tiempo. Nuestro objetivo de desarrollar una aplicación de escritorio aborda directamente estos desafíos, proporcionando una solución integral para una gestión eficiente y organizada. Los talleres podrán registrar fácilmente a sus clientes, mantener un seguimiento actualizado de los vehículos, generar partes y controlar los ingresos de manera sistemática y precisa.

2- *Implementar diferentes roles de usuario para restringir el acceso a determinadas funciones según las responsabilidades y permisos asignados.*

La seguridad de los datos y la confidencialidad son aspectos cruciales en cualquier negocio, y los talleres de coche no son una excepción. Al implementar diferentes roles de usuario, aseguramos que solo el personal autorizado pueda acceder a la información y realizar acciones específicas en el sistema. Esto brinda una capa adicional de seguridad y control, evitando el acceso no autorizado y protegiendo la privacidad de los clientes y la integridad de los datos del taller.

3- Incorporar un sistema de comunicación en tiempo real mediante un chat implementado con sockets para facilitar la colaboración entre los empleados.

La comunicación fluida y eficiente entre los empleados es fundamental para el funcionamiento óptimo de un taller de coches. Nuestro objetivo de implementar un sistema de chat en tiempo real con sockets permitirá a los empleados colaborar de manera efectiva, compartir información instantáneamente y resolver problemas rápidamente. Esto no solo mejora la productividad y la eficiencia interna, sino que también tiene un impacto positivo en la satisfacción del cliente al garantizar tiempos de respuesta más rápidos y una mejor coordinación en la prestación de servicios.

4- Integrar un sistema de fichajes para controlar las horas trabajadas de los empleados directamente en la aplicación.

El control preciso de las horas trabajadas es esencial para una gestión efectiva del personal y una planificación adecuada de los recursos. Nuestro objetivo de integrar un sistema de fichajes en la aplicación permitirá un registro automatizado y confiable de las horas trabajadas por los empleados. Esto proporciona una visión clara y precisa de los recursos humanos utilizados, facilita la elaboración de nóminas y mejora la gestión del personal en términos de asignación de tareas y control de costos.

5- Utilizar una base de datos SQL remota para almacenar la información de manera segura y accesible desde diferentes ubicaciones.

En la era digital, la capacidad de acceder y gestionar información de manera segura y en tiempo real desde diferentes ubicaciones es fundamental para la eficiencia y la agilidad empresarial. El objetivo de utilizar una base de datos SQL remota garantiza que la información del taller esté almacenada de forma segura y se pueda acceder a ella desde cualquier ubicación con una conexión adecuada. Esto brinda flexibilidad operativa y facilita la colaboración entre diferentes miembros del equipo, incluso si están físicamente separados.

6- Utilizar el sistema de almacenamiento distribuido Firebase para guardar documentos y archivos pesados como fotos y PDF.

La gestión de talleres de coche implica trabajar con una gran cantidad de documentos y archivos, como fotos de vehículos, facturas y presupuestos. Nuestro objetivo de utilizar el sistema de almacenamiento distribuido Firebase aborda directamente la necesidad de un almacenamiento confiable, seguro y escalable para estos archivos. Al guardar los documentos en la nube, aseguramos su disponibilidad desde cualquier ubicación y protegemos contra posibles pérdidas o daños en el hardware local. Esto también elimina la necesidad de ocupar espacio de almacenamiento local, optimizando los recursos del taller.

En resumen, los objetivos planteados en este proyecto están plenamente justificados para abordar las deficiencias y los desafíos que enfrentan los talleres de coche en su gestión diaria. La aplicación de escritorio propuesta ofrece una solución integral que mejora la eficiencia, garantiza la seguridad de los datos, facilita la comunicación interna, optimiza el control horario y permite un acceso remoto seguro a la información. Estoy convencido de que este proyecto será un éxito y tendrá un impacto significativo en la productividad y el éxito de los talleres de coche que lo implementen.

En adición a los objetivos y justificación previamente mencionados, es importante destacar la integración de una aplicación web desarrollada con Spring en este proyecto. Esta aplicación web proporciona a los clientes del taller una plataforma en línea para acceder y gestionar sus datos, partes, vehículos, facturas y presupuestos.

Algunas de las características y ventajas de esta integración:

Aplicación Web para Clientes: La aplicación web desarrollada con Spring permite a los clientes acceder a sus datos de manera conveniente y segura a través de una interfaz amigable. Los clientes tienen la posibilidad de ver y gestionar información relevante sobre sus vehículos, incluyendo historial de mantenimiento, reparaciones realizadas y piezas reemplazadas. Además, pueden consultar partes en curso, presupuestos, facturas y realizar pagos de forma segura.

La **capacidad de firmar partes** directamente desde la aplicación web agiliza los procesos y elimina la necesidad de trámites físicos. Los clientes pueden revisar, aprobar y firmar partes en línea, lo que mejora la eficiencia y la experiencia del cliente.

Asimismo, la opción de **realizar pagos en línea** brinda comodidad y seguridad, evitando la necesidad de pagos en persona o por otros medios.

La integración de la aplicación web con la aplicación de escritorio garantiza una comunicación eficiente y una sincronización de datos en tiempo real entre el personal del taller y los clientes. Esto facilita la colaboración, reduce los tiempos de respuesta y mejora la satisfacción del cliente al brindar un servicio más ágil y transparente.

En resumen, la integración de la aplicación web desarrollada con Spring en el proyecto de gestión de talleres de coche ofrece una plataforma en línea para que los clientes accedan y gestionen sus datos de manera conveniente y segura. La capacidad de firmar partes y realizar pagos en línea agiliza los procesos y mejora la experiencia del cliente. Esta integración fortalece aún más la propuesta de valor del proyecto y lo distingue de otros proyectos similares en el mercado.

Análisis de Mercado de Proyectos Similares en la Gestión de Talleres de Coche

En el mercado de la gestión de talleres de coche, existen diversos proyectos y soluciones tecnológicas que buscan mejorar la eficiencia y la productividad en este sector. A continuación, realizaré un análisis de mercado destacando los aspectos fundamentales que hacen que tu proyecto se destaque del resto:

Competidores directos:

Identificar los competidores directos es crucial para comprender el panorama competitivo en el mercado de la gestión de talleres de coche. Algunos de los proyectos similares que podrían existir son:

- **Software de gestión de talleres:** Hay varias empresas que ofrecen soluciones de software específicas para la gestión de talleres de coche. Estos sistemas suelen incluir funcionalidades básicas como registro de clientes, vehículos, partes y facturación. Sin embargo, pueden carecer de características adicionales que tu proyecto ofrece, como el sistema de comunicación en tiempo real y el control horario integrado.
- **Aplicaciones móviles de gestión de talleres:** Algunas empresas han desarrollado aplicaciones móviles que permiten a los talleres de coche gestionar sus operaciones desde dispositivos móviles. Estas aplicaciones suelen enfocarse en funciones específicas, como registro de clientes y vehículos, pero pueden carecer de funcionalidades más completas como el control de ingresos y el sistema de fichajes.

Existen varios programas de gestión para talleres mecánicos disponibles en el mercado que podrían ser considerados como competidores de mi proyecto. Algunos de los competidores más populares incluyen *AutoLeap*, *Vyapar*, *Shop Boss*, *RAMP GARAGE MANAGEMENT*, *GDS Workshop*, *AUTOMATE Garage Management Software*, *TechMan Garage Management System*, *Mitchell 1 Automotive Repair*, *AutoPro AutoMotive Software* y *SWS Garage Management System1234*.

Cada uno de estos programas ofrece una variedad de características y funcionalidades para ayudar a los talleres mecánicos a gestionar sus operaciones diarias. Algunas características comunes incluyen la gestión de citas previas, la emisión de presupuestos, albaranes y facturas, el envío de campañas de marketing por SMS o correo electrónico, el control del stock y la gestión documental.

Aspectos que destacan mi proyecto:

A continuación, destacaré los aspectos fundamentales del proyecto que lo diferencian y lo posicionan favorablemente frente a la competencia:

- **Solución integral:** A diferencia de muchos proyectos similares que ofrecen funcionalidades limitadas, el proyecto ofrece una solución integral para la gestión de talleres de coche. Desde el registro de clientes y vehículos hasta la creación de partes, control de ingresos, comunicación interna y control horario, tu aplicación de escritorio abarca todas las áreas clave de gestión en un solo sistema, brindando una experiencia completa y eficiente.
- **Comunicación en tiempo real:** La incorporación de un sistema de chat implementado con sockets destaca el proyecto al facilitar la comunicación en tiempo real entre los empleados del taller. Esta funcionalidad mejora la colaboración, la resolución de problemas y la toma de decisiones ágiles, lo cual es una ventaja significativa en comparación con soluciones que carecen de esta capacidad de comunicación efectiva.
- **Control horario integrado:** El sistema de fichajes que se ha integrado en la aplicación permite un control preciso y automatizado de las horas trabajadas por los empleados. Esto se diferencia de muchos otros proyectos que requieren soluciones separadas o procesos manuales para el registro y seguimiento de las horas laborales. El control horario integrado mejora la gestión del personal, la planificación de recursos y evita errores en la nómina.
- **Acceso remoto y seguridad de datos:** La utilización de una base de datos SQL remota y el sistema de almacenamiento distribuido Firebase destacan el proyecto al garantizar un acceso seguro y confiable a la información desde diferentes ubicaciones. Esto es especialmente valioso en un entorno donde los talleres de coche pueden tener múltiples ubicaciones o personal que necesita acceder a la información de forma remota. La seguridad y disponibilidad de los datos son aspectos esenciales y este proyecto aborda esta necesidad con soluciones sólidas.

En resumen, el proyecto de aplicación de gestión de talleres de coche se destaca de la competencia al ofrecer una solución integral, una comunicación en tiempo real efectiva, un control horario integrado y un acceso remoto seguro a la información. Estos aspectos fundamentales le brindan una ventaja competitiva al proporcionar una experiencia completa y eficiente para los talleres de coche, mejorando su productividad y facilitando su gestión operativa.

Análisis del sistema actual y justificación de la solución propuesta:

El sistema actual de gestión de talleres de coche puede estar basado en métodos manuales y sistemas dispersos, lo que genera ineficiencias y limita la capacidad de análisis y control. Algunas de las deficiencias identificadas son:

1. **Gestión fragmentada:** El uso de diferentes herramientas y sistemas para registrar clientes, vehículos, generar partes y controlar los ingresos dificulta la integración de la información y la generación de informes consolidados.
2. **Comunicación limitada:** La falta de un sistema de comunicación interno eficiente puede dar lugar a retrasos en la colaboración y la toma de decisiones, lo que afecta la productividad y la satisfacción del cliente.
3. **Control horario deficiente:** La ausencia de un sistema de fichajes automatizado dificulta el seguimiento preciso de las horas trabajadas por los empleados y puede generar errores en la nómina y la planificación del personal.
4. **Acceso y seguridad de datos:** La falta de un sistema centralizado de almacenamiento de datos y la dependencia de métodos físicos de archivo pueden comprometer la seguridad y accesibilidad de la información.

La solución propuesta, a través de la aplicación de escritorio desarrollada, aborda estas deficiencias al ofrecer una plataforma integral que permite:

1. Centralizar la gestión de talleres de coche en una sola aplicación, eliminando la necesidad de utilizar múltiples sistemas y herramientas.
2. Facilitar la comunicación interna a través de un sistema de chat implementado con sockets, lo que mejora la colaboración, la resolución de problemas y la toma de decisiones.
3. Automatizar el control horario mediante un sistema de fichajes incorporado, lo que garantiza un registro preciso de las horas trabajadas y facilita la gestión del personal.
4. Mejorar la seguridad y accesibilidad de los datos mediante el uso de una base de datos SQL remota y el sistema de almacenamiento distribuido Firebase, lo que permite un acceso seguro desde diferentes ubicaciones y asegura la integridad de los archivos pesados.

En resumen, el proyecto de aplicación de gestión de talleres de coche ofrece una solución integral, eficiente y segura para optimizar las operaciones y mejorar la productividad en los talleres, superando las limitaciones del sistema actual.

Análisis del sistema

Se ha optado para el desarrollo de este proyecto por la realización de una aplicación de escritorio en Java Swing, modular y modulable en función de los distintos tipos de roles establecidos, personal de oficina y mecánico.

Además se ha desarrollado en Spring Tool una aplicación web para el acceso de los clientes a sus datos personales más relevantes del sistema.

Se ha elegido como base de datos una base de datos SQL remota a la que acceden las diferentes instancias de la aplicación de escritorio así como la aplicación web. Se ha establecido un sistema de almacenamiento de documentos y archivos en la nube llamado Firebase que garantiza la integridad del sistema. La elección de trabajar con datos y documentos en red supone que los requisitos de funcionamiento de los equipos que integren el sistema serán mínimos.

Se ha implementado en la aplicación de escritorio además de las funcionalidades típicas de cualquier aplicación de este ámbito un chat para la comunicación interna de los empleados de la empresa que lo integre y un sistema de control de horario.

Se ha implementado también una API que se integra en el sistema para cubrir diferentes funcionalidades, la API de Sendgrid, un gestor de emails utilizada para implementar un OTP(one time password) y para el envío de facturas y presupuestos a los clientes.

Además se ha preparado la aplicación para su internacionalización, traduciéndola a tres idiomas mediante la extracción de los Strings que visualiza el usuario, a varios archivos de propiedades, gracias a los cuales la aplicación cambia de idioma con tan solo pulsar en un menú, inspirado en las aplicaciones de android.

Java Swing es una biblioteca de interfaz de usuario (UI) para la creación de aplicaciones de escritorio en el lenguaje de programación Java. Proporciona un conjunto completo de componentes gráficos y herramientas para el desarrollo de interfaces de usuario interactivas y visualmente atractivas.

Spring Tools es un conjunto de herramientas integradas de desarrollo (Integrated Development Environment, IDE) basado en Eclipse para el desarrollo de aplicaciones Java con el framework Spring. Proporciona un entorno de desarrollo robusto y eficiente que facilita la creación, pruebas y depuración de aplicaciones basadas en Spring.

Filebase es un sistema de almacenamiento distribuido en la nube que proporciona una solución eficiente y segura para el almacenamiento y manejo de archivos y documentos pesados. Se basa en la tecnología de almacenamiento en la nube y está diseñado para ofrecer una plataforma escalable y confiable para el almacenamiento y la gestión de datos.

SendGrid es una plataforma de correo electrónico en la nube que facilita el envío y la entrega confiable de correos electrónicos transaccionales y de marketing. Es ampliamente utilizado por empresas y desarrolladores para gestionar y automatizar el envío de correos electrónicos a gran escala.

db4free es un servicio de base de datos en línea que ofrece a los desarrolladores y usuarios la posibilidad de utilizar una base de datos MySQL de forma gratuita. Es un entorno de desarrollo y pruebas conveniente para aquellos que desean experimentar, probar y aprender sobre el funcionamiento de una base de datos MySQL sin incurrir en costos adicionales.

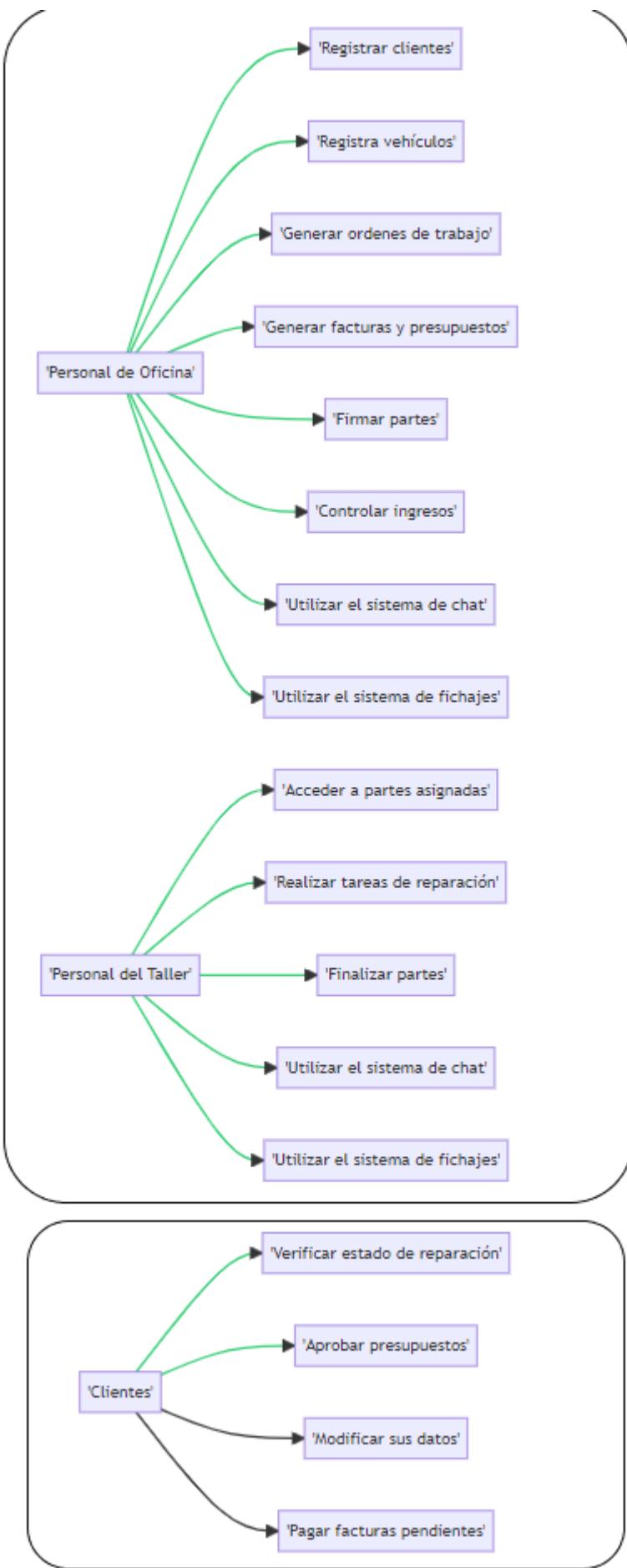
Para el desarrollo del proyecto se ha decidido seguir el patrón **MVC** modelo vista controlador. La principal ventaja del patrón MVC es la separación clara de preocupaciones y la modularidad del código. Permite un desarrollo más estructurado y facilita la colaboración en equipo, ya que cada componente tiene una responsabilidad bien definida. Además, el patrón MVC promueve la reutilización del código, ya que los componentes pueden ser intercambiables y modificados de forma independiente sin afectar a los demás. El patrón MVC proporciona una arquitectura clara y organizada para el desarrollo de aplicaciones. Al separar la lógica de negocio, la presentación y el control de las acciones del usuario, facilita el mantenimiento, la escalabilidad y la flexibilidad de la aplicación. Es un enfoque ampliamente adoptado en el desarrollo de software y ha demostrado ser efectivo para construir aplicaciones robustas y bien estructuradas.

Datos de entrada y salida. Diagramas de casos de uso

Un diagrama de casos de uso es una representación visual que describe las interacciones entre los actores y el sistema en un escenario específico. Este tipo de diagrama se utiliza comúnmente en el análisis y diseño de sistemas de software para comprender los requisitos funcionales de un sistema desde la perspectiva del usuario.

En el siguiente diagrama se muestran los diferentes actores que interactúan con el sistema y las acciones que pueden realizar en él.

a
p
p
d
e
s
c
r
i
t
o
a
p
p
w
e
b



No lo especifica el diagrama, pero lo indico ahora, aunque sea obvio, cualquier interacción con el sistema requiere estar previamente registrado en el y estar logueado.

Diseño de la base de datos

Justificación

La elección de una base de datos SQL para el proyecto de gestión de talleres de coche se justifica por varias razones:

Estructura y relaciones de datos: Una base de datos SQL es adecuada cuando se necesita mantener una estructura relacional y establecer relaciones entre diferentes conjuntos de datos. En un sistema de gestión de talleres de coche, es probable que existan relaciones entre entidades como clientes, vehículos, partes, facturas, presupuestos, etc. Una base de datos SQL permite establecer estas relaciones de manera eficiente y asegurar la integridad de los datos.

Flexibilidad y consultas complejas: Las bases de datos SQL ofrecen un lenguaje de consulta poderoso (SQL) que permite realizar consultas complejas y extraer información de manera eficiente. En un proyecto de gestión de talleres, es probable que necesites realizar consultas y análisis de datos para generar informes, realizar búsquedas específicas o realizar cálculos complejos. Una base de datos SQL te brinda la flexibilidad necesaria para realizar estas operaciones de manera eficiente y rápida.

Escalabilidad y rendimiento: Las bases de datos SQL están diseñadas para ser escalables y manejar grandes volúmenes de datos. A medida que tu proyecto crezca y maneje más clientes, vehículos, partes y otros datos, una base de datos SQL te permitirá escalar verticalmente (aumentar los recursos del servidor) o horizontalmente (distribuir la carga en múltiples servidores) para garantizar un rendimiento óptimo y una respuesta rápida.

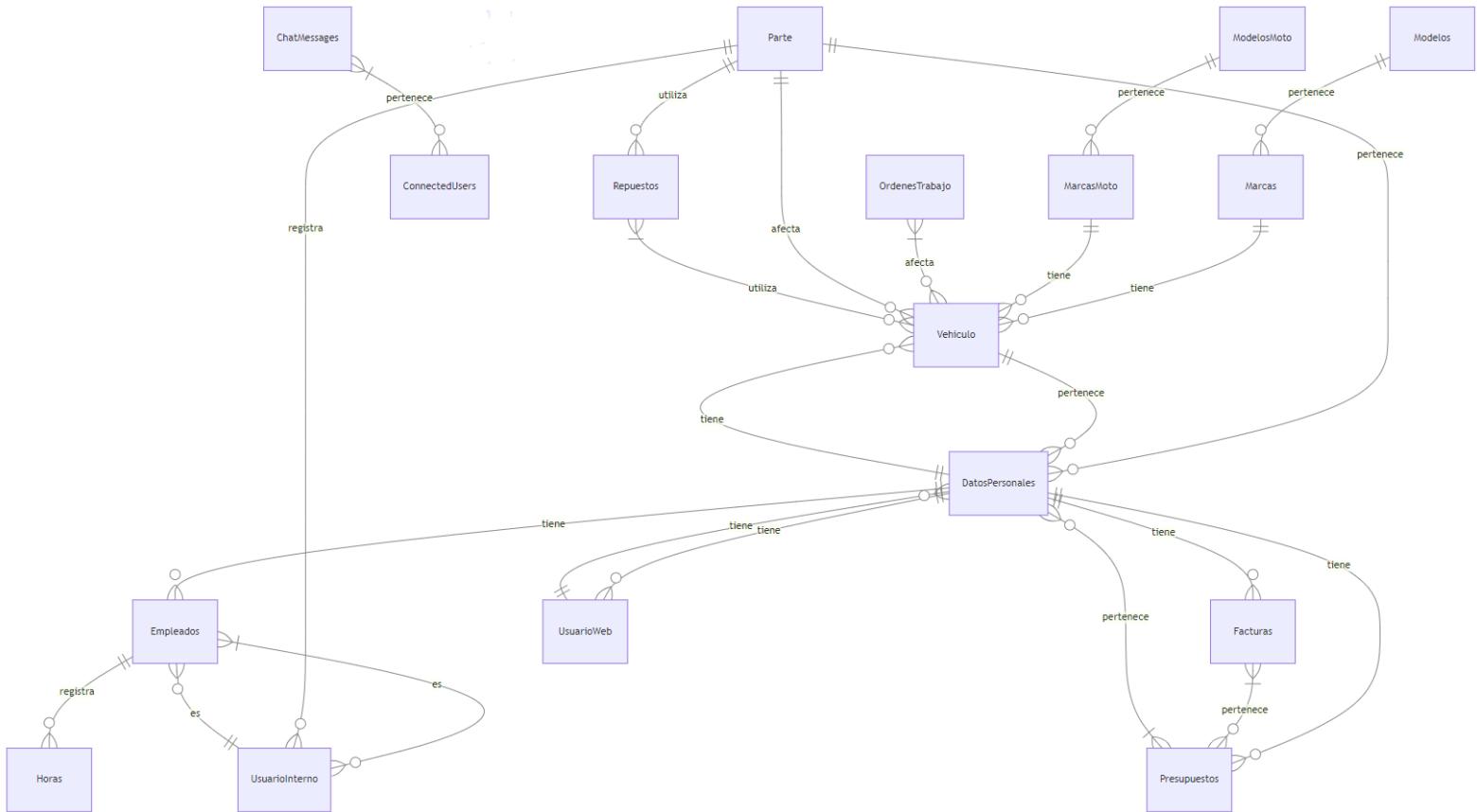
Seguridad y control de acceso: Las bases de datos SQL proporcionan mecanismos de seguridad robustos para proteger los datos confidenciales. Puedes establecer privilegios de acceso a nivel de usuario y controlar quién tiene acceso y qué operaciones pueden realizar en la base de datos. Esto te permite garantizar la seguridad y confidencialidad de los datos de tus clientes y cumplir con las regulaciones de protección de datos.

Amplia compatibilidad y soporte: Las bases de datos SQL son ampliamente utilizadas y cuentan con una gran cantidad de herramientas, bibliotecas y frameworks compatibles. Esto facilita el desarrollo y la integración de tu aplicación con otros sistemas y tecnologías, lo que te brinda más opciones y flexibilidad para hacer crecer tu proyecto.

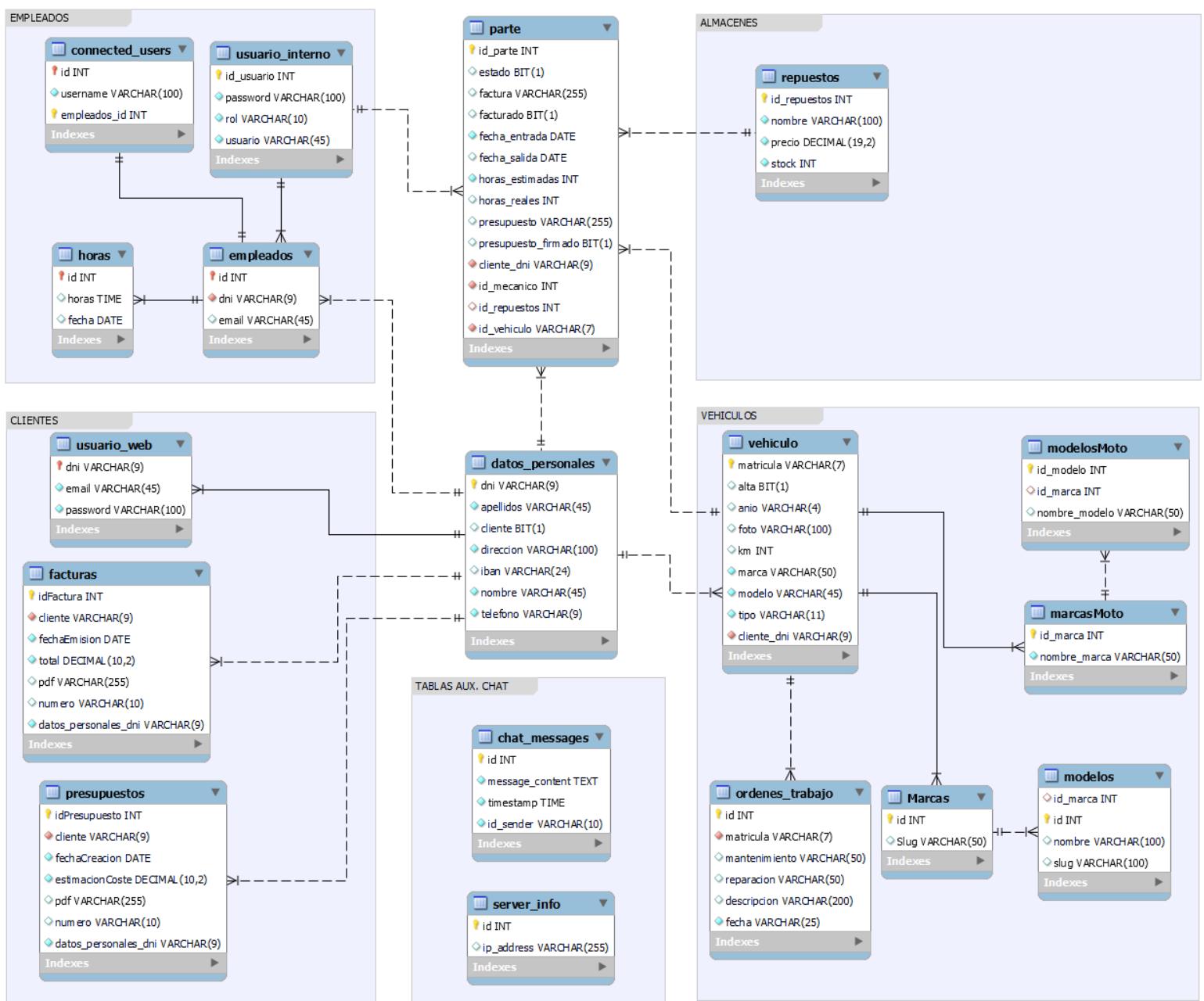
En resumen, la elección de una base de datos SQL para el proyecto de gestión de talleres de coche se justifica por su capacidad para manejar estructuras de datos relacionales, realizar consultas complejas, escalar y mantener un alto rendimiento, brindar seguridad y control de acceso, así como por su amplia compatibilidad y soporte en la industria.

Diseño

El siguiente diagrama EER muestra cómo se relacionan las diferentes entidades para su posterior traducción a las diferentes tablas que formarán la base de datos

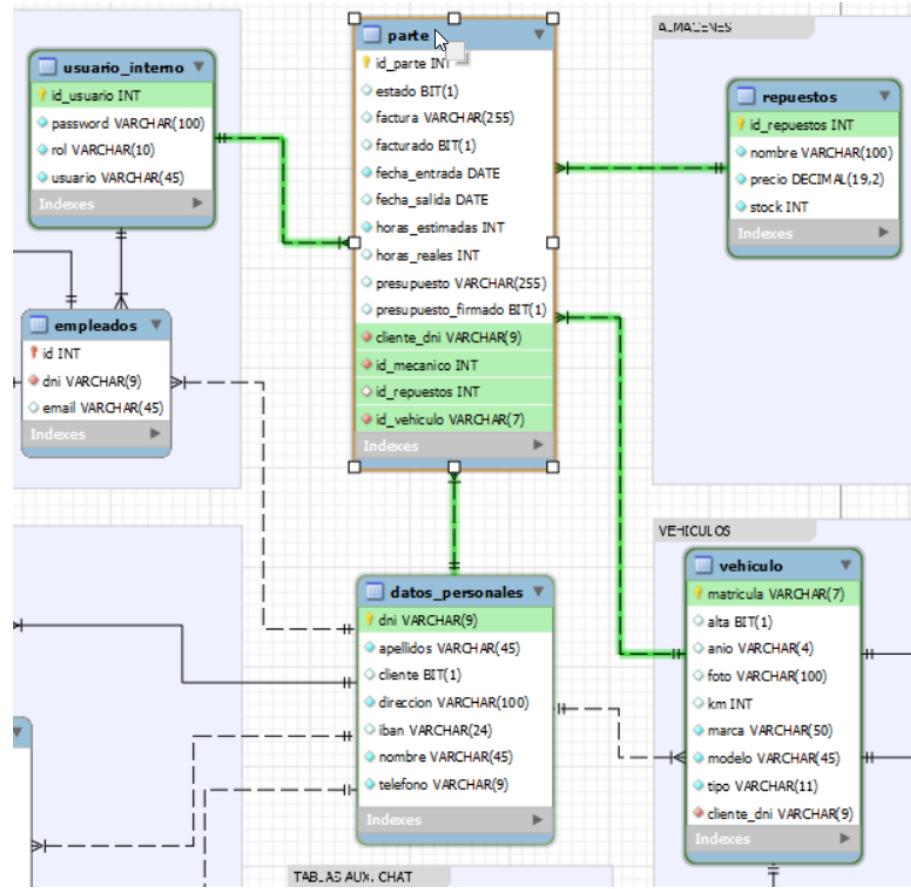


A continuación se muestra el esquema de las tablas de la base de datos y sus relaciones:

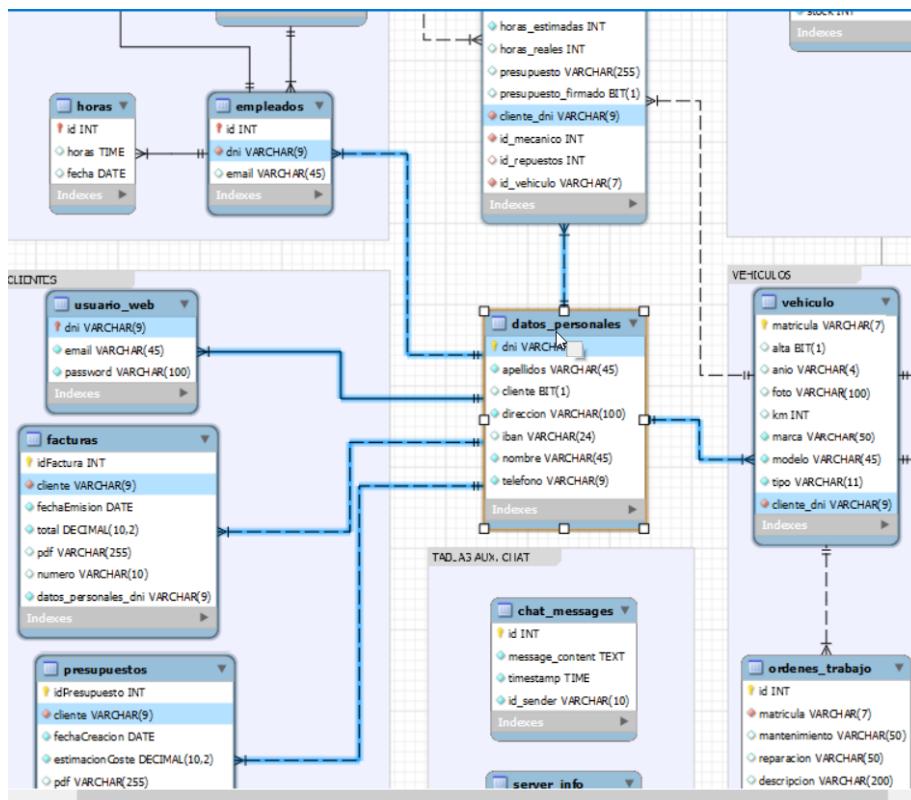


Como se puede apreciar en el esquema las dos tablas más importantes alrededor de las cuales se estructura toda la base de datos son “parte” y “datos_personales” que son las que permiten que toda la estructura esté relacionada.

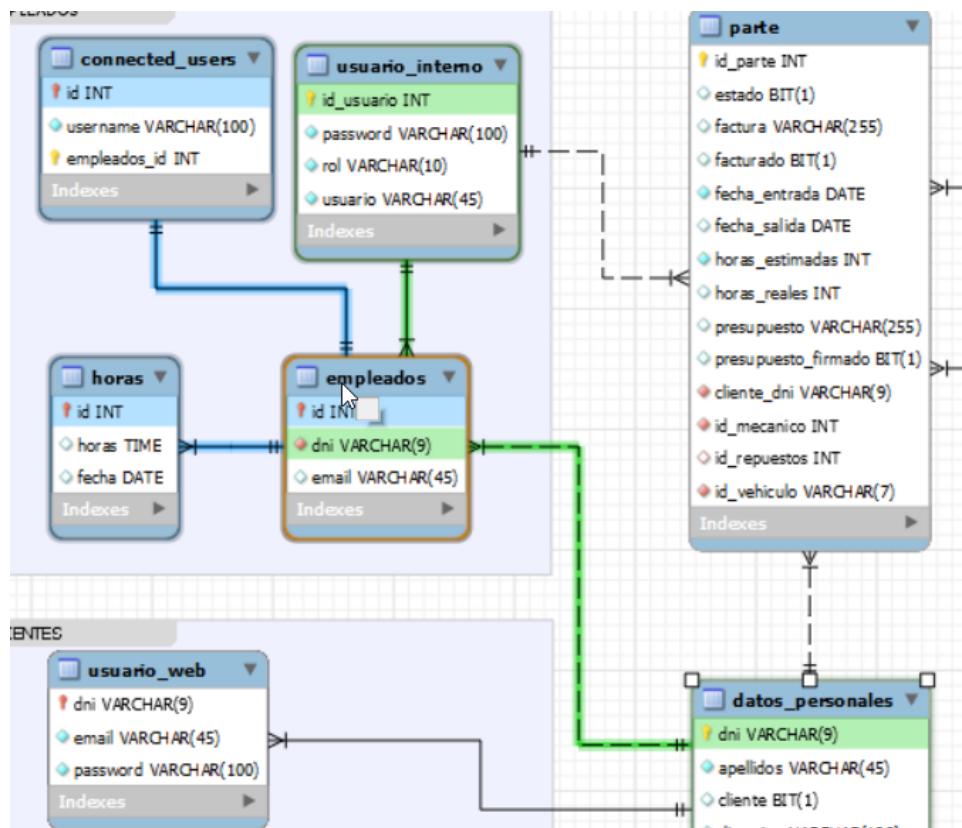
A continuación muestro una vista detallada de la tabla “parte” y sus relaciones:



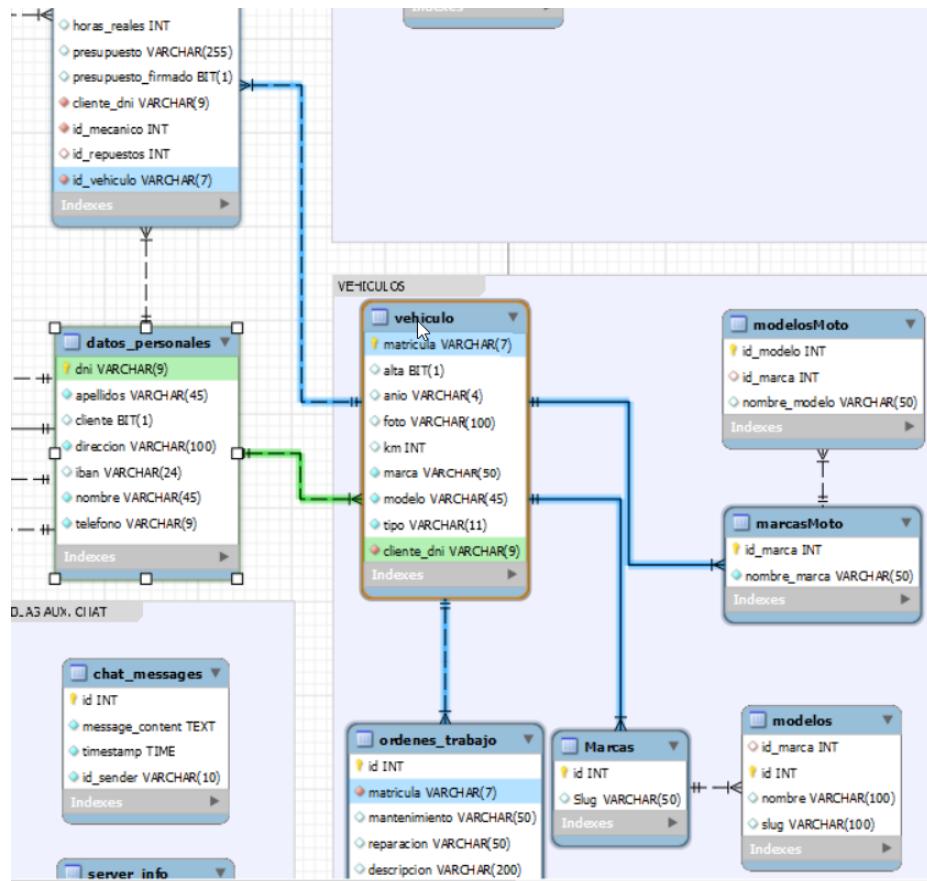
La tabla “datos_personales” y sus relaciones:



La tabla “empleados” y sus relaciones:



La tabla “vehiculos” y sus relaciones:

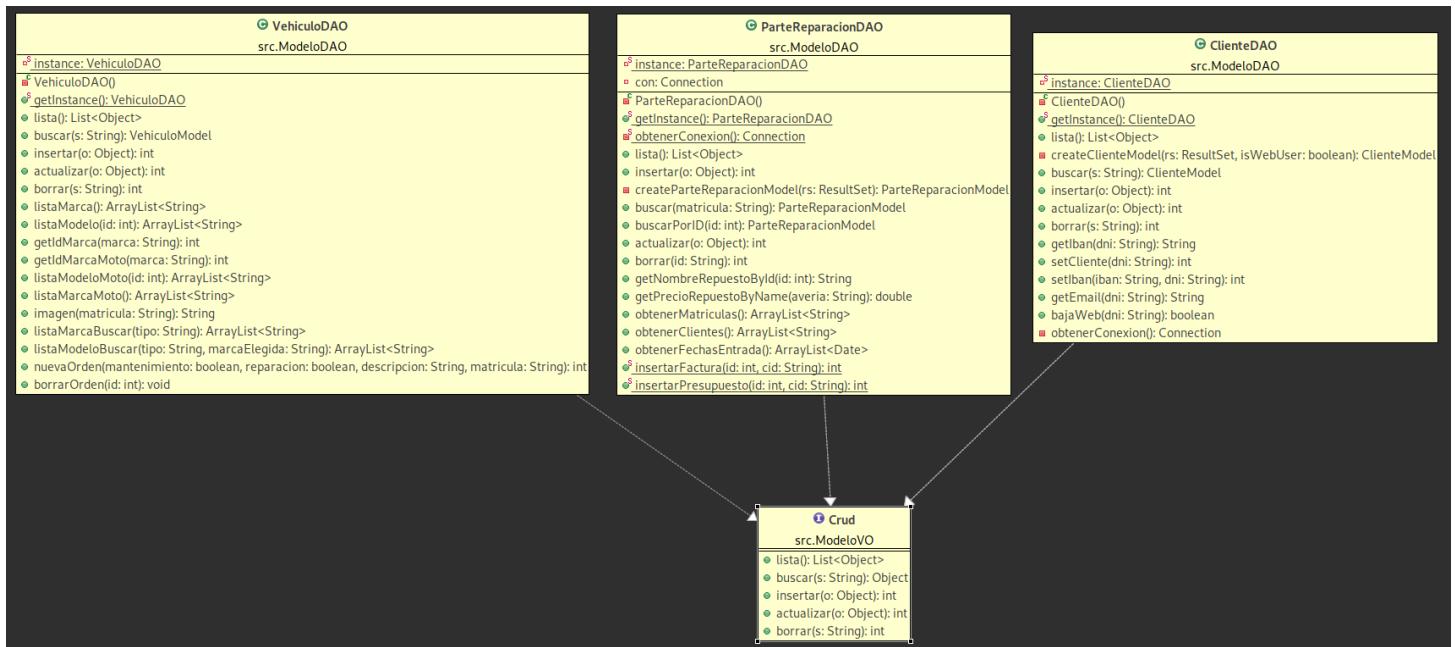


Como se puede observar no existe una tabla “clientes” esto se debe a que tanto clientes como empleados comparten la mayoría de los datos que aparecen en la tabla “datos_personales”, existiendo un campo booleano “cliente” que es el que determina si los datos pertenecen a un cliente o a un empleado de esta manera no se hace necesaria la implementación de otra tabla específica para los clientes.

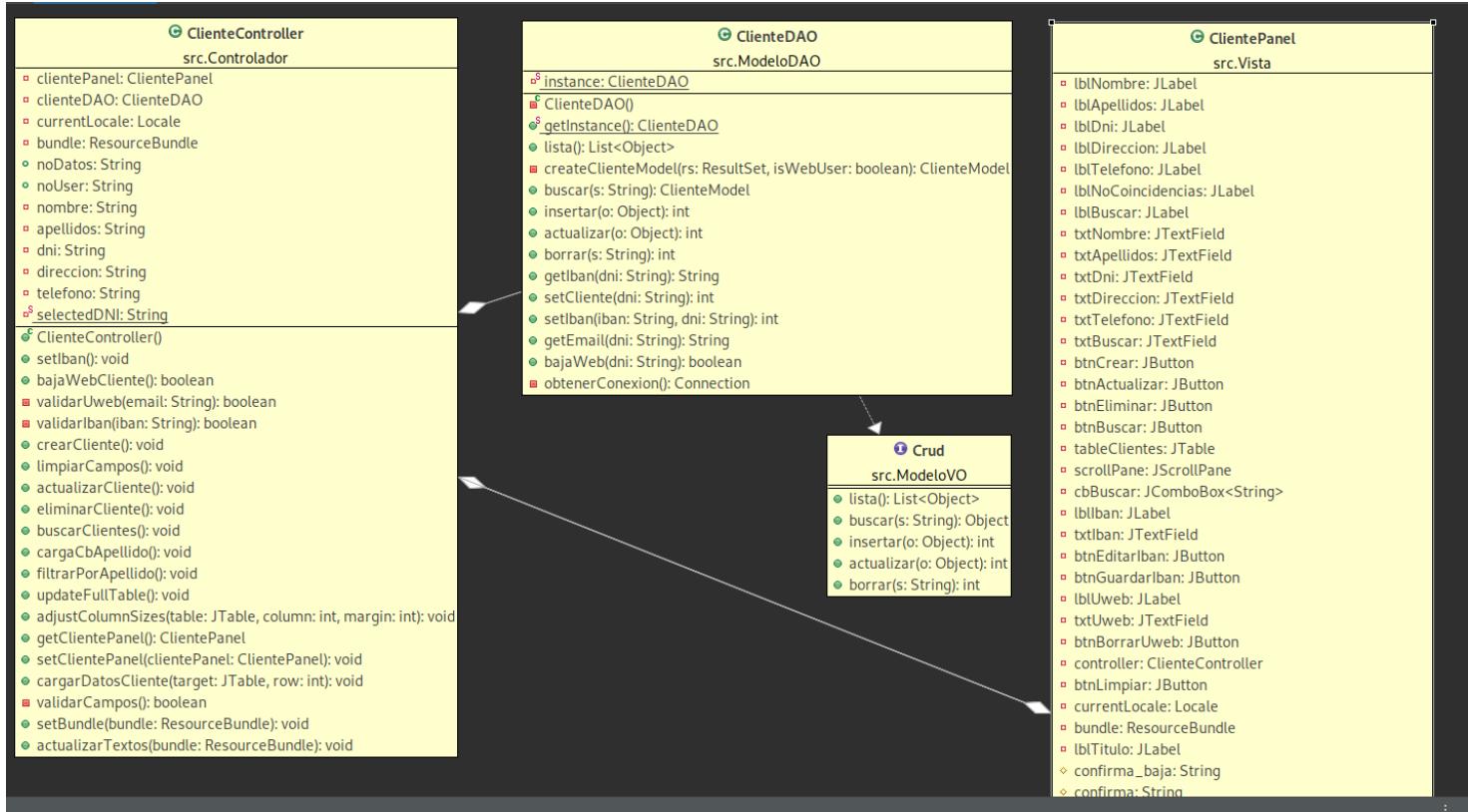
Diccionario de clases

A continuación se detallan las clases más relevantes del proyecto mediante sus correspondientes diagramas de clases.

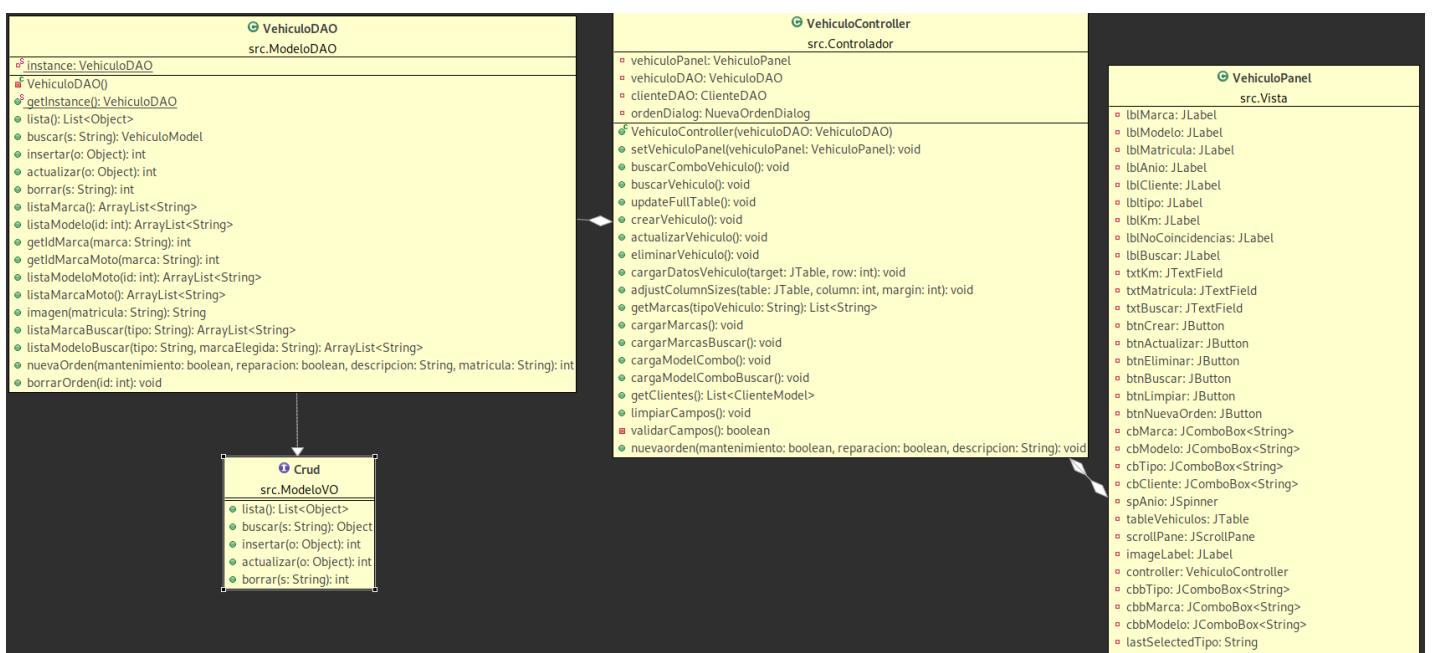
Interfaz crud: La interfaz crud que implementan las clases DAO, ‘ClienteDAO’, ‘VehiculoDAO’ y ‘ParteReparacionDAO’ se utiliza para homogeneizar la implementación de las principales clases DAO y de esta manera facilitar su implementación.



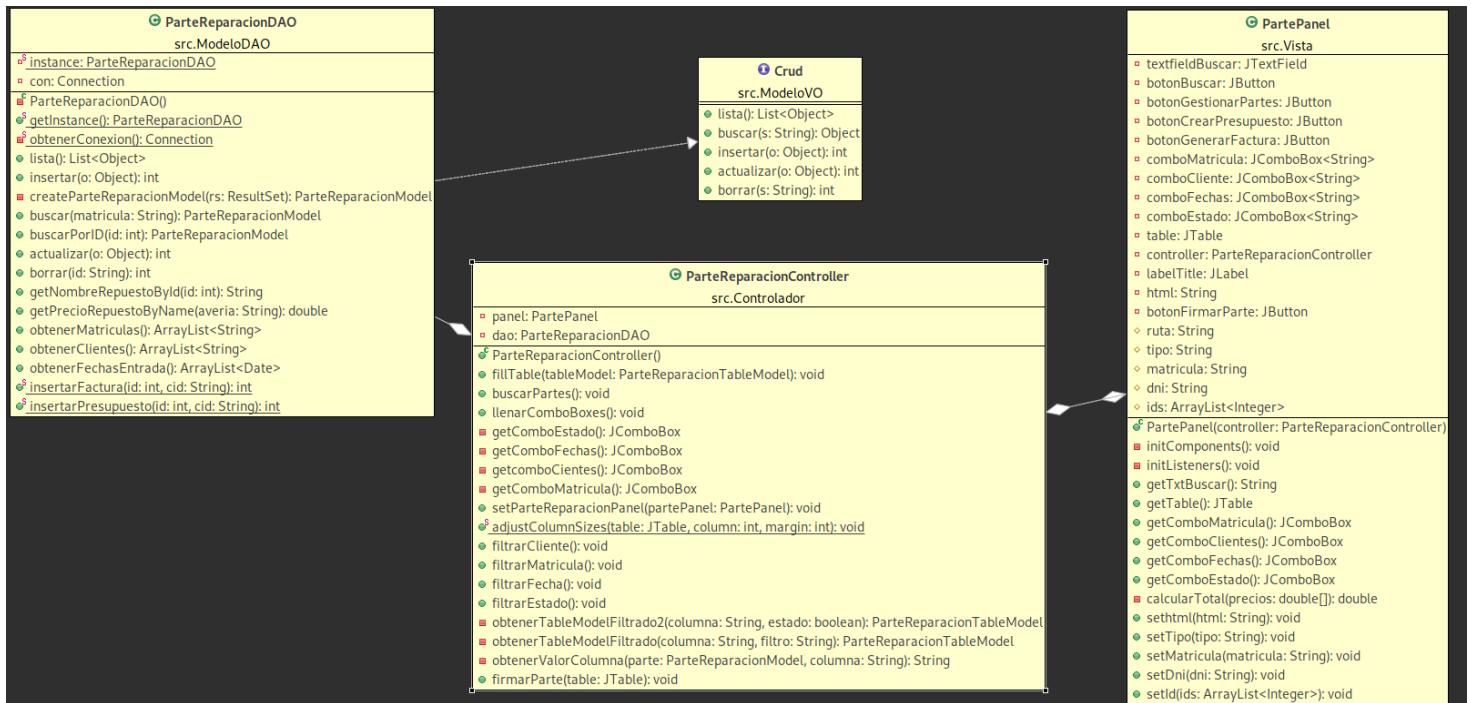
La clases relacionadas con los clientes:



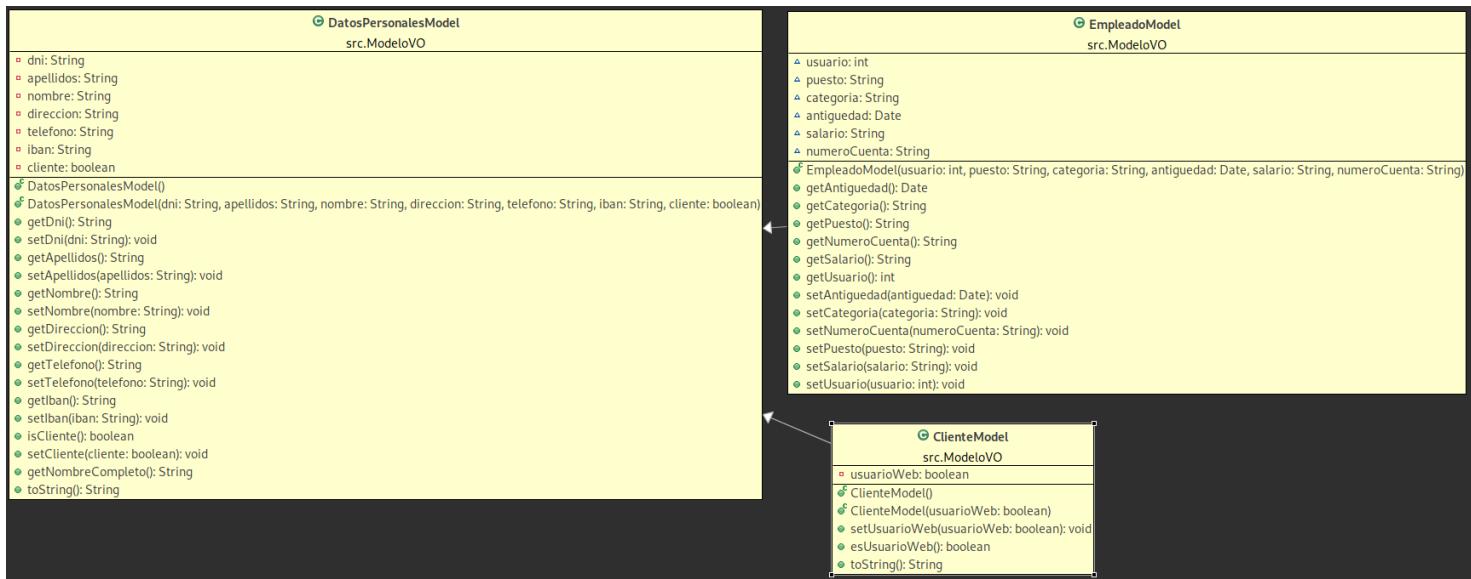
Las clases relacionadas con los vehiculos:



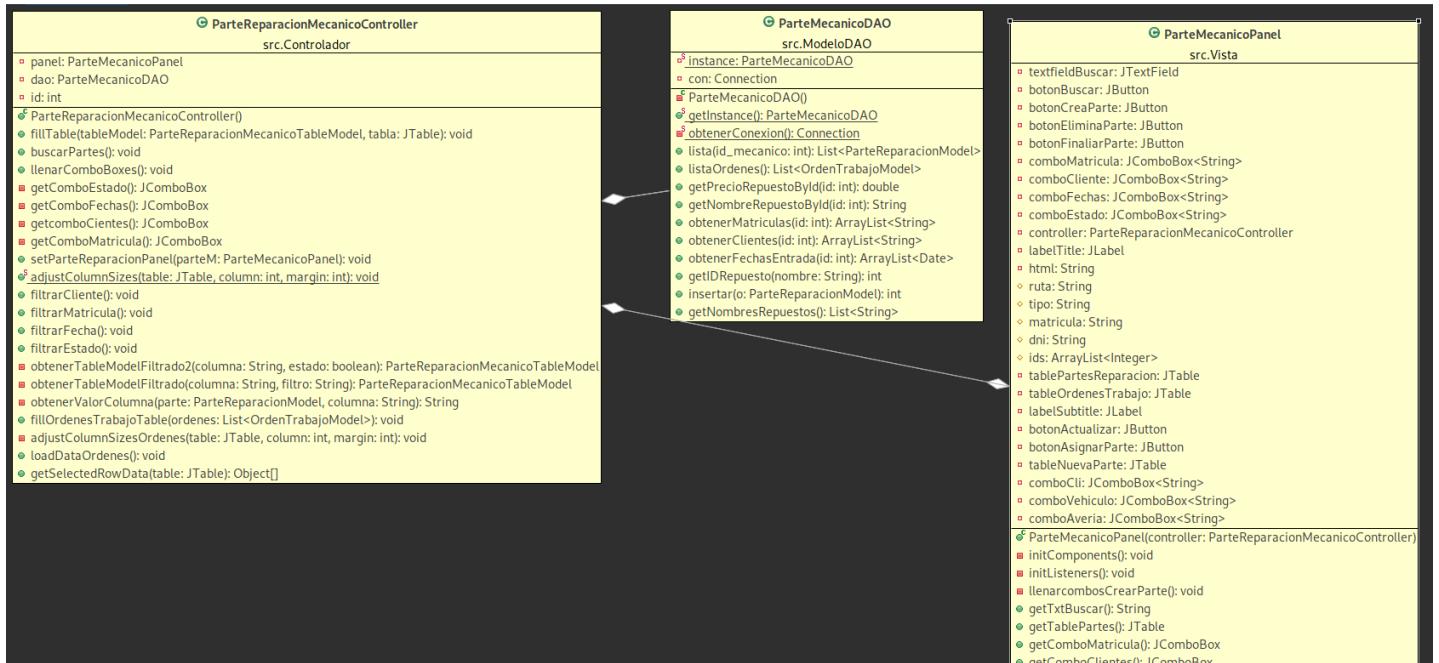
Las clases relacionadas con los partes de reparación:



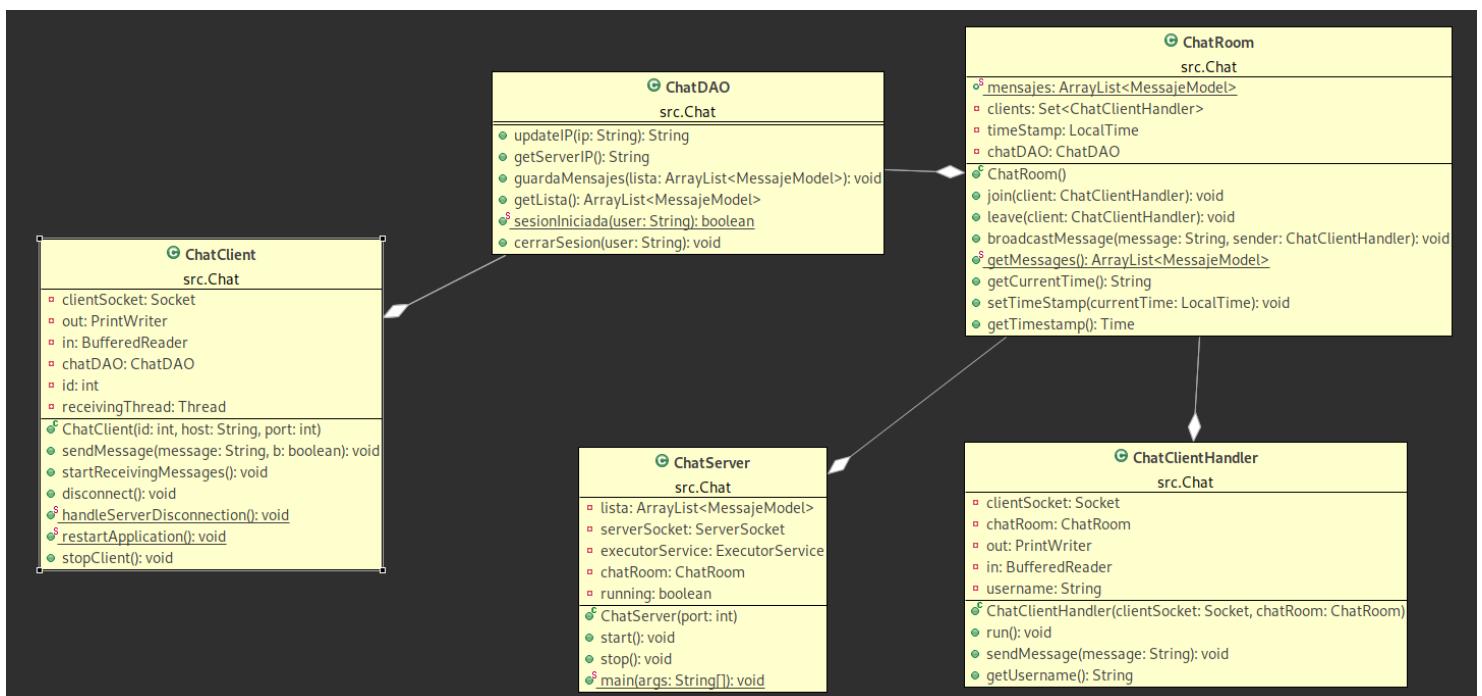
La clase 'DatosPersonaleModel' de la que heredan 'ClienteModel' y 'EmpleadoModel':



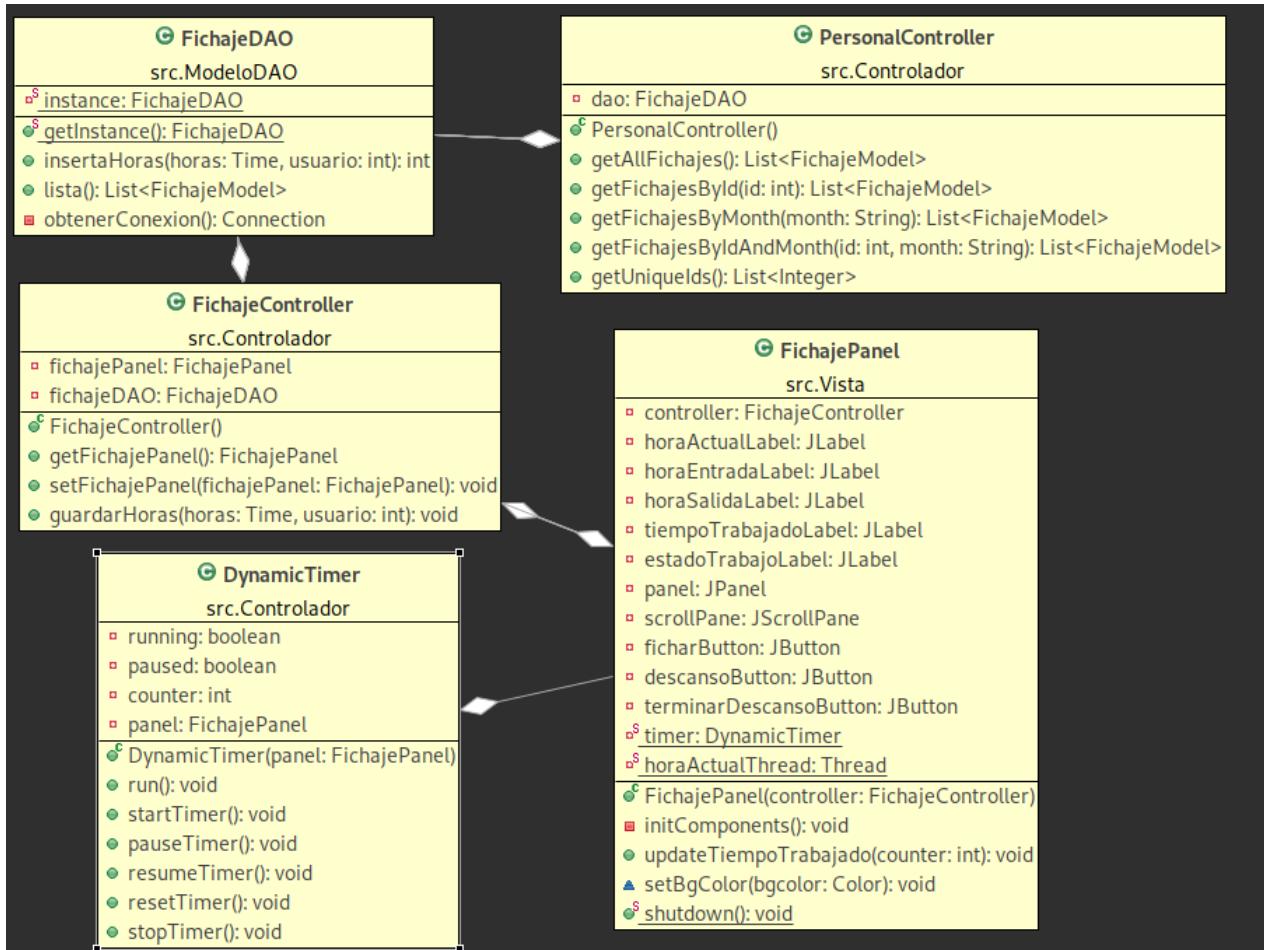
La clases relacionadas con los partes de los mecánicos:



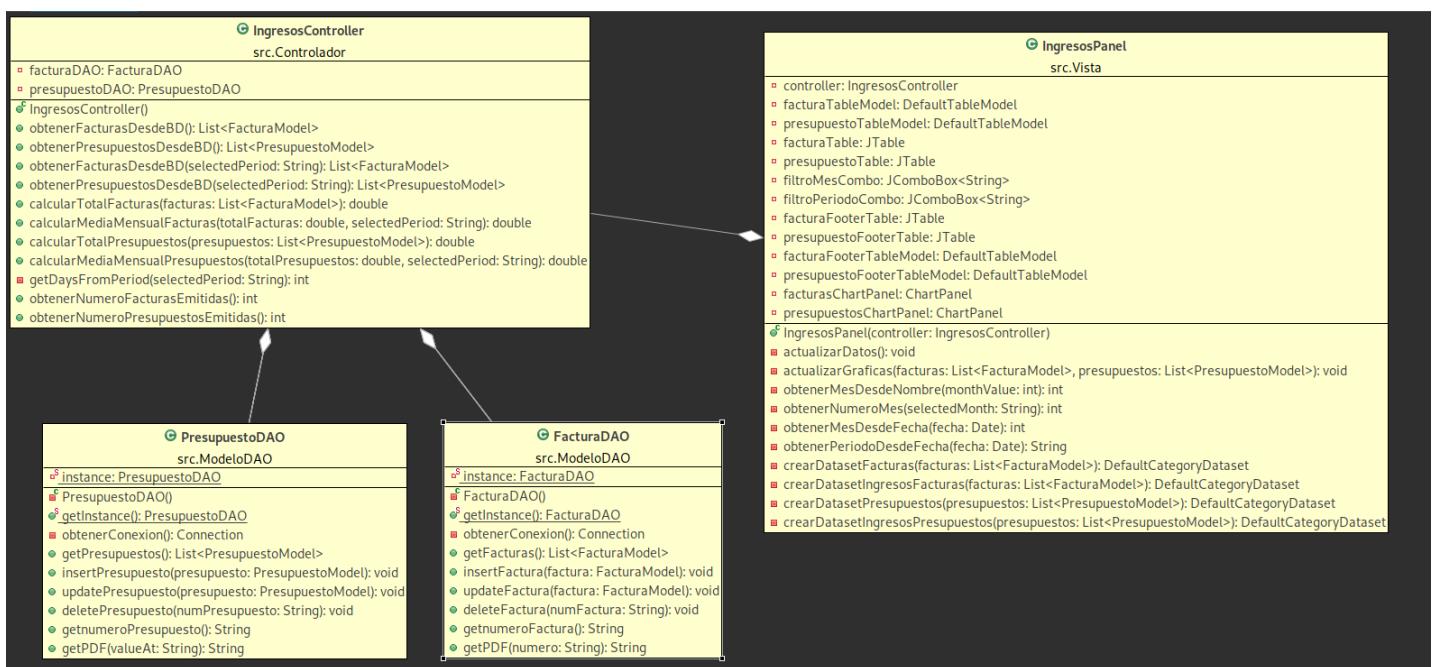
Las clases que implementan el chat:



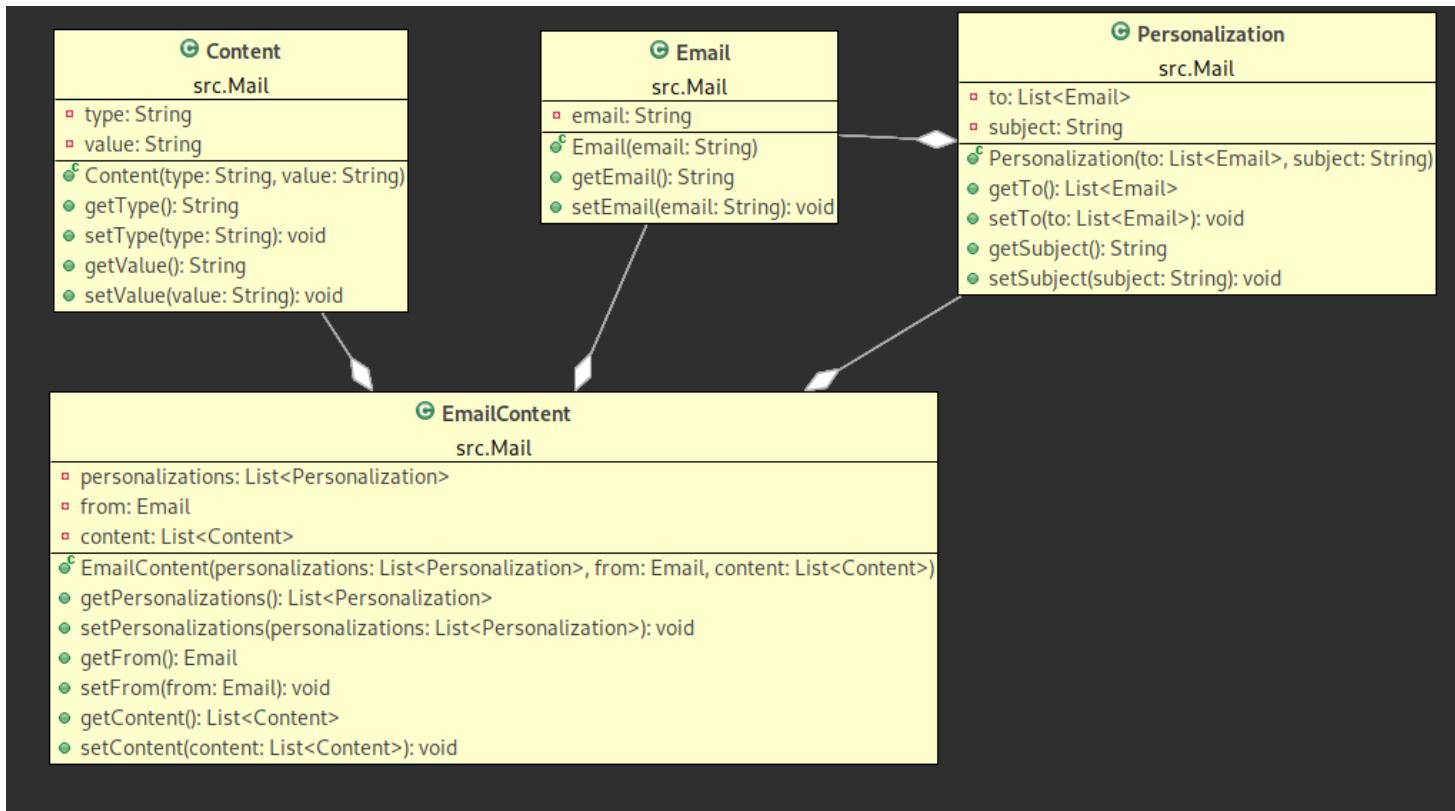
Las clases que implementan el sistema de fichajes:



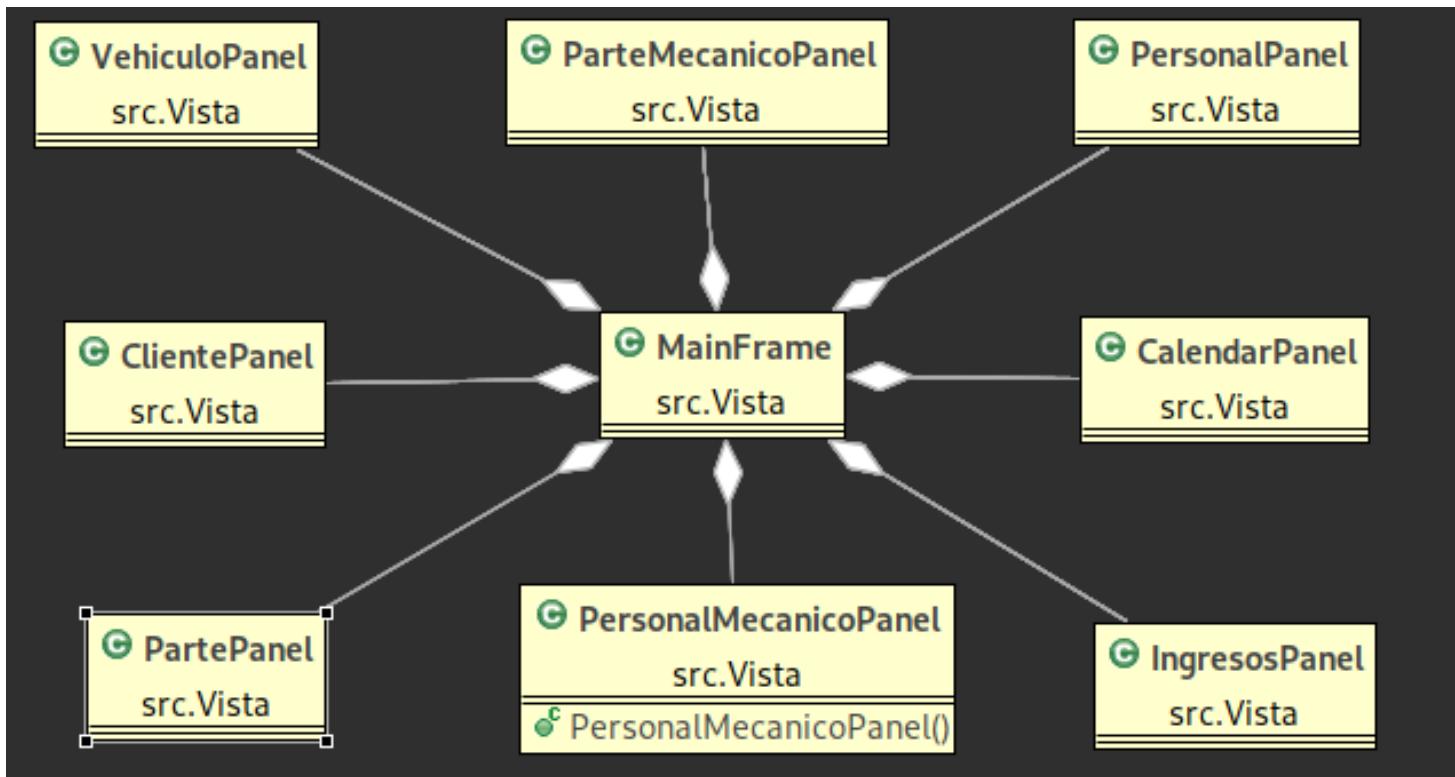
Las clases relacionadas con los ingresos:



Las clases que gestionan los emails:

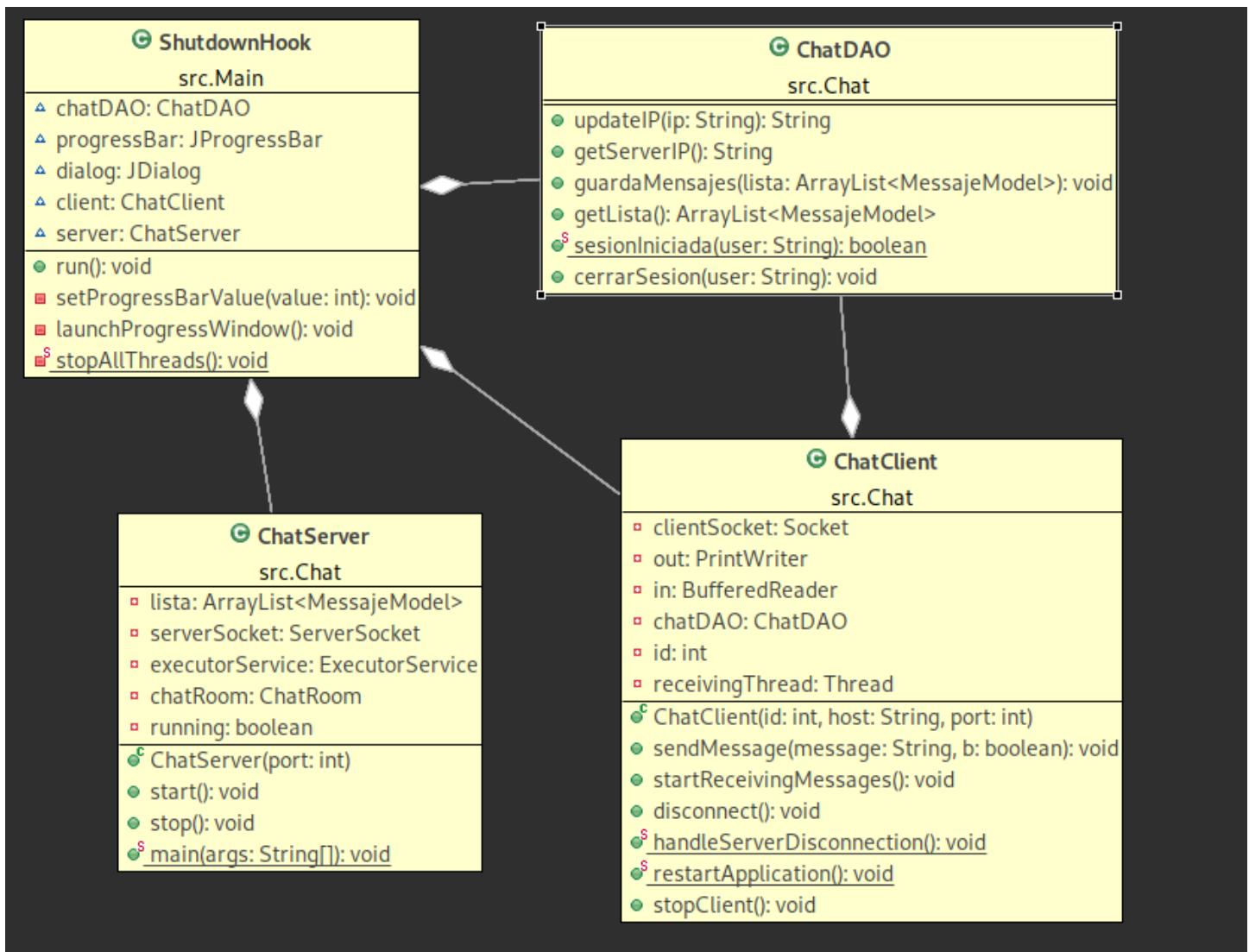


La clase Mainframe y los diferentes paneles que la integran:



La clase ShutdownHook encargada de gestionar el cierre de todos los recursos al salir de la aplicación, esta clase es particularmente importante en mi sistema debido a la gran cantidad de hilos que se manejan en el y que es necesario cerrar para que se liberen los recursos al salir de la aplicación, esta clase facilita este trabajo ya que se ejecuta cuando el programa recibe la orden de apagarse.

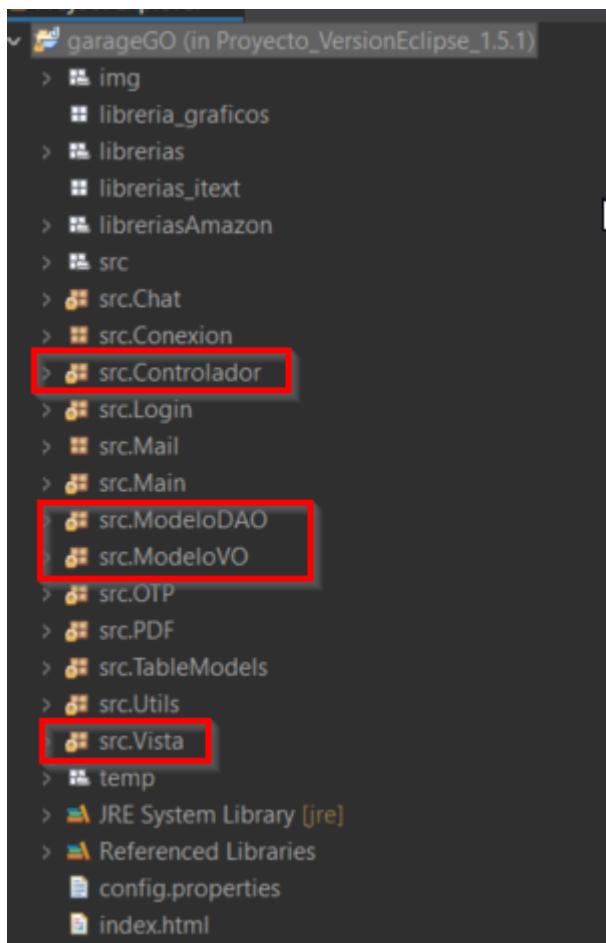
Esta es una visión general de cómo está diseñado el sistema



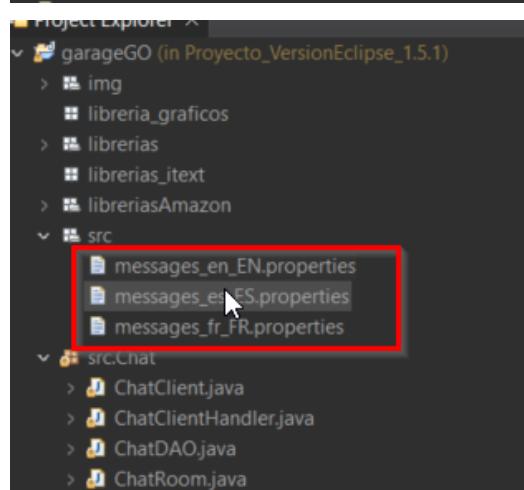
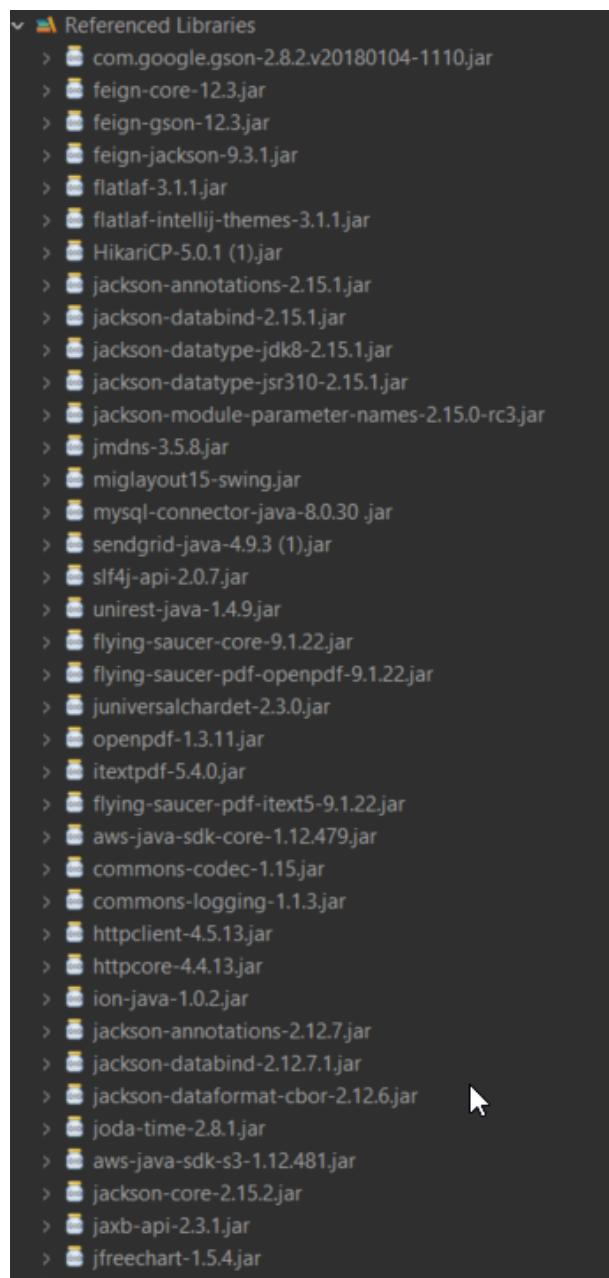
Implementación del sistema

Estructura del proyecto

Como se ha mencionado anteriormente el proyecto está estructurado siguiendo la arquitectura MVC, separando ademá en paquetes adicionales partes del programa más independientes como pueden ser el chat, la gestión de los emails, de los PDFs, la implementación del OTP, el Login, los diferentes Tablemodels personalizados y un paquete Utils donde hay algunas clases auxiliares.



Se han utilizado una serie de librerías y dependencias necesarias para el envío de emails, creación de PDFs, conexión con Firebase, además de la biblioteca *Hikari* para implementar un pool de conexiones, el conector SQL, *miglayout* para la gestión de layouts en la interfaz gráfica, *flatlaf* para la personalización del aspecto visual de la interfaz o *jfreechart* para la creación de gráficos. Además de una serie de dependencias necesarias para complementar algunas de las bibliotecas anteriormente descritas como por ejemplo *jackson* para la creación de documentos json necesarios para usar la API de envío de emails.



Archivos de propiedades utilizados para la traducción del proyecto.

Proyecto completo:

The screenshot displays the file structure of a Java project named "garageGO". The project is organized into several packages:

- garageGO (in Proyecto_VersionEclipse_1.5.1)**
 - img
 - libreria_graficos
 - librerias
 - librerias_iText
 - libreriasAmazon
 - src
 - Chat
 - ChatClient.java
 - ChatClientHandler.java
 - ChatDAO.java
 - ChatRoom.java
 - ChatServer.java
 - Conexion
 - Conexion.java
 - Controlador
 - ClienteController.java
 - DynamicTimer.java
 - FichajeController.java
 - IngresosController.java
 - ParteReparacionController.java
 - ParteReparacionMecanicoController.java
 - PersonalController.java
 - VehiculoController.java
 - Login
 - LoginDialog.java
 - PasswordUtils.java
 - Mail
 - Content.java
 - Email.java
 - EmailContent.java
 - Personalization.java
 - Main
 - MainApp.java
 - ShutdownHook.java

src

- PDF
 - FacturaEditor.java
 - FacturaPdf.java
- TableModels
 - CustomTableModel.java
 - NuevoParteTableModel.java
 - OrdenTrabajoTableModel.java
 - ParteReparacionMecanicoTableModel.java
 - ParteReparacionTableModel.java
- Utils
 - ButtonEditor.java
 - ButtonRenderer.java
 - CheckBoxRenderer.java
 - CheckBoxRendererOrdenes.java
 - Filebase.java
 - IngresosChart.java
- Vista
 - CalendarPanel.java
 - ClientePanel.java
 - FichajePanel.java
 - IngresosPanel.java
 - MainFrame.java
 - NuevaOrdenDialog.java
 - ParteMecanicoPanel.java
 - PartePanel.java
 - PersonalMecanicoPanel.java
 - PersonalPanel.java
 - VehiculoPanel.java
- temp
 - ShutDownHook.java

- ModeloDAO
- ClienteDAO.java
- FacturaDAO.java
- FichajeDAO.java
- ParteMecanicoDAO.java
- ParteReparacionDAO.java
- PresupuestoDAO.java
- UserDAO.java
- VehiculoDAO.java
- ModeloVO
- ClienteModel.java
- Crud.java
- DatosPersonalesModel.java
- EmpleadoModel.java
- FacturaModel.java
- FichajeModel.java
- MensajeModel.java
- OrdenTrabajoModel.java
- ParteReparacionModel.java
- PresupuestoModel.java
- usuario_internoModel.java
- VehiculoModel.java
- src-OTR

A continuación mostraré una serie de clases a modo de ejemplo de la implementación del MVC:

Clase POJO:

```
1 public class VehiculoModel {
2     private String matricula;
3     private String tipo;
4     private String marca;
5     private String modelo;
6     private String anio;
7     private String ClienteDni;
8     private Integer km;
9     private boolean alta = true;
10    private String foto;
11
12    public static final String TIPO_TURISMO = "TURISMO";
13    public static final String TIPO_FURGONETA = "FURGONETA";
14    public static final String TIPO_MOTOCICLETA = "MOTOCICLETA";
15
16    public VehiculoModel(){}
17    public VehiculoModel(String matricula, String ClienteDni, String tipo, String marca, String modelo) {
18
19        this.matricula = matricula;
20        this.marca = marca;
21        this.modelo = modelo;
22        this.tipo = tipo;
23        this.anio=null;
24
25        this.ClienteDni=ClienteDni;
26    }
27
28    public VehiculoModel(String matricula, String tipo, String marca, String modelo, String anio, String ClienteDni, Integer km, String foto) {
29        this.matricula = matricula;
30        this.tipo = tipo;
31        this.marca = marca;
32        this.modelo = modelo;
33        this.anio = anio;
34        this.ClienteDni = ClienteDni;
35        this.km = km;
36        this.foto = foto;
37    }
38
39    public VehiculoModel(String matricula, String ClienteDni, String tipo, String marca, String modelo, String año) {
40        this.matricula = matricula;
41        this.marca = marca;
42        this.modelo = modelo;
43        this.tipo = tipo;
44        this.ClienteDni = ClienteDni;
45        this.anio = año;
46    }
47
48    public Integer getKm() {
49        return km;
50    }
51
52    public void setKm(Integer km) {
53        this.km = km;
54    }
55 }
```

Clase de acceso a datos (DAO):

```
public class VehiculoDAO implements Crud {

    private static VehiculoDAO instance;

    private VehiculoDAO() {
        // Constructor privado para evitar instancias directas
    }

    public static VehiculoDAO getInstance() {
        if (instance == null) {
            instance = new VehiculoDAO();
        }
        return instance;
    }

    @Override
    public List<Object> lista() {
        List<Object> lista = new ArrayList<>();
        String cadenaSQL = "SELECT * FROM vehiculo";

        try (Connection con = Conexion.getConnection();
             PreparedStatement consulta = con.prepareStatement(cadenaSQL);
             ResultSet rs = consulta.executeQuery()) {

            while (rs.next()) {
                VehiculoModel v = new VehiculoModel();
                v.setMatricula(rs.getString("matricula"));
                v.setClienteDni(rs.getString("cliente_dni"));
                v.setTipo(rs.getString("tipo"));
                v.setMarca(rs.getString("marca"));
                v.setModelo(rs.getString("modelo"));
                v.setAnio(rs.getString("anio"));
                v.setKm(rs.getInt("km"));
                lista.add(v);
            }
        } catch (SQLException ex) {
            Logger.getLogger("Error en modelo.dao.VehiculoDAO.lista()");
        }
        return lista;
    }

    @Override
    public VehiculoModel buscar(String s) {
        VehiculoModel v = null;
        String cadenaSQL = "SELECT * FROM vehiculo WHERE matricula=?";

        try (Connection con = Conexion.getConnection();
             PreparedStatement consulta = con.prepareStatement(cadenaSQL)) {

            consulta.setString(1, s);
            try (ResultSet rs = consulta.executeQuery()) {
```

Clase controladora:

```
public class VehiculoController {
    private VehiculoPanel vehiculoPanel;
    private VehiculoDAO vehiculoDAO;
    private ClienteDAO clienteDAO;
    private NuevaOrdenDialog ordenDialog;

    public VehiculoController(VehiculoDAO vehiculoDAO) {
        this.vehiculoDAO = vehiculoDAO;
        this.vehiculoPanel = new VehiculoPanel(this);
        this.clienteDAO = ClienteDAO.getInstance();

        updateFullTable();
    }

    public void setVehiculoPanel(VehiculoPanel vehiculoPanel) {
        this.vehiculoPanel = vehiculoPanel;
    }

    public void buscarComboVehiculo() {
        String tipobuscado = (String) vehiculoPanel.getTxtBuscar().toLowerCase(); // Convertir la cadena de búsqueda a minúsculas
        List<Object> vehiculos = vehiculoDAO.lista();
        DefaultTableModel tableModel = new DefaultTableModel(
            new String[] { "Matrícula", "Propietario", "Tipo", "Marca", "Modelo", "Año", "Kilómetros" }, 0);
        for (Object o : vehiculos) {
            VehiculoModel vehiculo = (VehiculoModel) o;
            String tipo = vehiculo.getTipo();
            String marca = vehiculo.getMarca();
            String modelo = vehiculo.getModelo();

            // Comprobar si los valores seleccionados son null o están vacíos
            boolean tipoCoincide = (tipobuscado == null || tipobuscado.isEmpty() || tipobuscado.equals(tipo));
            boolean marcaCoincide = (marcabuscada == null || marcabuscada.isEmpty() || marcabuscada.equals(marca));
            boolean modeloCoincide = (modelobuscado == null || modelobuscado.isEmpty() || modelobuscado.equals(modelo));

            // Si todas las condiciones son verdaderas, entonces añade una fila a la tabla
            if (tipoCoincide && marcaCoincide && modeloCoincide) {
                tableModel.addRow(new Object[] { vehiculo.getMatricula(), vehiculo.getClienteDni(), vehiculo.getTipo(),
                    vehiculo.getMarca(), vehiculo.getModelo(), vehiculo.getAnio(), vehiculo.getKm() });
            }
        }

        vehiculoPanel.setTableModel(tableModel);
        vehiculoPanel.mostrarNoCoincidencias(tableModel.getRowCount() == 0);

        JTable table = vehiculoPanel.getTable();
        DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
        centerRenderer.setHorizontalAlignment(JLabel.CENTER);

        // Ajuste de las columnas al contenido
        for (int i = 0; i < table.getColumnCount(); i++) {
            table.getColumnModel().getColumn(i).setCellRenderer(centerRenderer);
        }
    }
}
```

Vista:

```
public class VehiculoPanel extends JPanel {

    private JLabel lblMarca;
    private JLabel lblModelo;
    private JLabel lblMatricula;
    private JLabel lblAnio;
    private JLabel lblCliente;
    private JLabel lblTipo;
    private JLabel lblKm;
    private JLabel lblNoCoincidencias;
    private JLabel lblBuscar;

    private JTextField txtKm;
    private JTextField txtMatricula;
    private JTextField txtBuscar;
    private JButton btnCrear;

    private JButton btnActualizar;
    private JButton btnEliminar;
    private JButton btnBuscar;
    private JButton btnLimpiar;
    private JButton btnNuevaOrden;

    private JComboBox<String> cbMarca;
    private JComboBox<String> cbModelo;
    private JComboBox<String> cbTipo;
    private JComboBox<String> cbCliente;

    private JSpinner spAnio;

    private JTable tableVehiculos;
    private JScrollPane scrollPane;

    private JLabel imageLabel;

    private VehiculoController controller;
    private JComboBox<String> cbbTipo;
    private JComboBox<String> cbbMarca;
    private JComboBox<String> cbbModelo;

    private String lastSelectedTipo = "";
    private String lastSelectedMarca = "";

    /**
     * Constructor de la clase VehiculoPanel.
     *
     * @param controller el controlador que maneja las acciones del panel
     */
    public VehiculoPanel(VehiculoController controller) {
        this.controller = controller;

        setLayout(new MigLayout("", "[grow]", "[grow][grow][grow][grow][grow][grow][grow][grow]"));
        initComponents();
        initListeners();
    }
}
```

A continuación mostraré una serie de clases que tienen cierta importancia en el funcionamiento de la aplicación:

MainApp, la clase de acceso al programa , encargada también al inicio de ir lanzando el cliente o el servidor del chat en función de las necesidades, el chat implementado es automático, lo que quiere decir que no necesita configuración alguna para ponerse en funcionamiento, aquí en el main lo primero que intenta hacer es conectarse si hay algún servidor en marcha, para ello accede a la base de datos y comprueba en la tabla donde se guarda la ip del servidor, recoge la ip almacenada e intenta crear un socket con esa ip, si falla quiere decir que no hay ningún servidor corriendo y entonces se pone en 'modo' servidor, lanza el servidor y almacena su propia ip en la tabla de la base de datos para que la próxima instancia del programa que arranque pueda conectarse a él comprobando la ip en la BD.

Después de esto lanza el Login y a continuación se conecta como cliente al chat.

```
public class MainApp {
    private static ChatDAO chatDAO;
    private static MainFrame ventanaPrincipal;
    private static LoginDialog loginDialog;
    public static usuario_internoModel user;
    private static ChatServer server;
    private static ChatClient chatClient;
    private static Socket socket;

    public static void main(String[] args) {
        user = new usuario_internoModel();
        Runtime.getRuntime().addShutdownHook(new ShutdownHook());

        Font labels = new Font("Segoe", Font.BOLD, 16);
        Font botones = new Font("Segoe", Font.BOLD, 14);
        UIManager.put("Button.arc", 999);
        UIManager.put("Component.arc", 999);
        UIManager.put("ProgressBar.arc", 999);
        UIManager.put("TextComponent.arc", 999);
        UIManager.put("Component.focusWidth", 3);
        UIManager.put("Label.font", labels);
        UIManager.put("Button.font", botones);

        SwingUtilities.invokeLater(() -> {
            ClienteDAO clienteDAO = ClienteDAO.getInstance();
            VehiculoDAO vehiculoDAO = VehiculoDAO.getInstance();
            ParteReparacionDAO parteReparacionDAO = ParteReparacionDAO.getInstance();

            ClienteController clienteController = new ClienteController();
            VehiculoController vehiculoController = new VehiculoController(vehiculoDAO);
            ParteReparacionController parteReparacionController = new ParteReparacionController();
            FichajeController fichajeController = new FichajeController();
            IngresosController ingresosController = new IngresosController();

            chatDAO = new ChatDAO();
            String ip = chatDAO.getServerIP();
            int port = 9090;

            try {
                new Socket(ip, port);
                System.out.println("Un servidor ya está en marcha.");
            } catch (IOException e) {
                System.out.println("No hay un servidor en marcha, iniciando uno nuevo...");
            }

            Thread serverThread = new Thread(() -> {
                try {
                    server = new ChatServer(port);
                    server.start();
                } catch (IOException e1) {
                    System.out.println("Error al iniciar el servidor");
                    e1.printStackTrace();
                } catch (InterruptedException e1) {
                    e1.printStackTrace();
                }
            });
        });
    }
}
```

```

        serverThread.start();
    }

    try {
        Thread.sleep(2000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    lanzarLogin();

    chatClient = null;

    try {
        chatClient = new ChatClient(user.getId_usuario(), ip, port);
        System.out.println("cliente iniciado");
    } catch (IOException e) {
        e.printStackTrace();
        System.err.println("no se pudo iniciar el cliente");
        return;
    }

    chatClient.startReceivingMessages();

    if (loginDialog.isLoggedIn()) {
        ventanaPrincipal = new MainFrame(clienteController, vehiculoController, parteReparacionController,
            fichajeController, chatClient, ingresosController);
        ventanaPrincipal.setVisible(true);
    } else {
        System.exit(0);
    }
});

}

public static void lanzarLogin() {
    loginDialog = new LoginDialog(ventanaPrincipal);
    loginDialog.setVisible(true);
}

public static void stopServerAndClient() {
    if (socket != null) {
        try {
            socket.close();
        } catch (IOException e) {
            System.err.println("no se pudo cerrar el socket");
        }
    }
    if (server != null) {
        server.stop();
        System.out.println("Servidor parado");
    }
    if (chatClient != null) {

        try {
            chatClient.stopClient();
        } catch (Exception e) {
            System.err.println("no se pudo parar el cliente");
        }
    }
}

```

Clase ShutdownHook: Como he comentado anteriormente esta clase se encarga de gestionar el cierre y liberación de los recursos al cerrarse la aplicación, la aprovecho tambien para realizar una serie de consultas a la base de datos, como la inserción de los mensajes de chat para que se carguen al inicio de la siguiente sesión o un delete de la tabla de usuarios conectados que controla que el mismo usuario no pueda estar conectado en dos instancias diferentes al mismo tiempo.

```
public class ShutdownHook extends Thread {  
  
    ChatDAO chatDAO;  
    JProgressBar progressBar;  
    JDialog dialog;  
    ChatClient client;  
    ChatServer server;  
  
    @Override  
    public void run() {  
        SwingUtilities.invokeLater(this::launchProgressWindow);  
  
        try {  
            System.out.println("Cerrando aplicación...");  
            setProgressBarValue(10);  
  
            chatDAO = new ChatDAO();  
            chatDAO.guardaMensajes(ChatRoom.mensajes);  
            System.out.println("Mensajes de chat guardados.");  
            setProgressBarValue(30);  
  
            chatDAO.cerrarSesion(MainApp.user.getUsuario());  
            System.out.println("Registro de sesión borrado.");  
            setProgressBarValue(50);  
  
            try {  
                stopAllThreads();  
            } catch (Exception e) {  
                MainApp.stopServerAndClient();  
                System.out.println("...Recursos liberados.");  
                setProgressBarValue(100);  
                e.printStackTrace();  
                System.exit(0);  
            }  
        }  
    }  
}
```

```

        MainApp.stopServerAndClient();
        System.out.println("...Recursos liberados.");
        setProgressBarValue(70);

        Conexion.closeConnection();
        setProgressBarValue(100);
    } catch (Exception e) {
        System.exit(0);
    }

    try {
        Thread.sleep(2000); // Tiempo adicional para permitir que la barra de progreso llegue al 100%
    } catch (InterruptedException e) {
        // Hilo interrumpido
        System.exit(0);
    }
}

// Cerrar el diálogo de progreso cuando se haya terminado de liberar los recursos
System.exit(0);
}

private void setProgressBarValue(int value) {
    try {
        SwingUtilities.invokeLater(() -> progressBar.setValue(value));
    } catch (Exception e) {
        setProgressBarValue(100);
    }
}

private void launchProgressWindow() {
    FlatProgressBarUI ui = new FlatProgressBarUI();
    progressBar = new JProgressBar();
    progressBar.setUI(ui);
    progressBar.setMinimum(0);
    progressBar.setMaximum(100);
    progressBar.setStringPainted(true);

    // Adjust the height of the progress bar
    progressBar.setPreferredSize(new Dimension(progressBar.getPreferredSize().width, 10)); // Set the preferred height to 20

    // Change the font size of the text inside the progress bar
    Font font = progressBar.getFont();
    float newSize = font.getSize() * 1.5f; // Increase the font size by 50%
    progressBar.setFont(font.deriveFont(newSize));

    dialog = new JDialog((Frame) null, "Cerrando aplicación", true);
    dialog.getContentPane().add(progressBar, BorderLayout.CENTER);
    dialog.setSize(350, 70);
    dialog.setLocationRelativeTo(null);
    SwingUtilities.invokeLater(() -> dialog.setVisible(true));
}
}

```

```

private static void stopAllThreads() {
    ThreadGroup rootGroup = Thread.currentThread().getThreadGroup();
    ThreadGroup parentGroup;
    while ((parentGroup = rootGroup.getParent()) != null) {
        rootGroup = parentGroup;
    }

    Thread[] threads = new Thread[rootGroup.activeCount()];
    while (rootGroup.enumerate(threads, true) == threads.length) {
        threads = new Thread[threads.length * 2];
    }

    for (Thread thread : threads) {
        if (thread != null) {
            thread.interrupt();
        }
    }
}
}
<
```

Clase Conexion: Esta clase implementa un pool de conexiones de Hikari para gestionar el uso racional de las conexiones a la base de datos. Está limitado a un máximo de 16 conexiones concurrentes, esto es debido a que la base de datos admite un máximo de 50 conexiones y limitándolo a 16 me aseguro de que pueden correr tres instancias de la aplicación al mismo tiempo

```
public class Conexion {  
    private static final String DRIVER = "com.mysql.cj.jdbc.Driver";  
    private static final String DATABASE = "workbenchmanager";  
    private static final String HOSTNAME = "db4free.net";  
    private static final String PORT = "3306";  
    private static final String URL = "jdbc:mysql://" + HOSTNAME + ":" + PORT + "/" + DATABASE + "?serverTimezone=Europe/Madrid";  
    private static final String USERNAME = "wbmanager";  
    private static final String PASSWORD = "wbmanager";  
  
    private static HikariConfig config = new HikariConfig();  
    private static HikariDataSource ds;  
  
    static {  
        config.setJdbcUrl(URL);  
        config.setUsername(USERNAME);  
        config.setPassword(PASSWORD);  
        config.setDriverClassName(DRIVER);  
        config.setMaximumPoolSize(16); //Se puede ajustar este valor dependiendo de las necesidades En mi caso concreto la base de dato  
        config.addDataSourceProperty("cachePrepStmts", "true");  
        config.addDataSourceProperty("prepStmtCacheSize", "250");  
        config.addDataSourceProperty("prepStmtCacheSqlLimit", "2048");  
        ds = new HikariDataSource(config);  
    }  
  
    private Conexion() {}  
  
    public static Connection getConnection() throws SQLException {  
        return ds.getConnection();  
    }  
  
    public static void closeConnection() {  
        ds.close();  
    }  
}
```

La clase Login: Esta clase se encarga de dar acceso a la aplicación, en caso de introducir incorrectamente el usuario o contraseña de forma errónea, lanza una ventana y manda un email al correo del usuario que está intentando acceder con una contraseña de un solo uso que deberá introducir para poder acceder, esta contraseña se calcula cogiendo los seis últimos dígitos del timestamp() del momento en que se han cumplido los tres intentos.

```
public class LoginDialog extends JDialog {  
    private JTextField usernameField;  
    private JPasswordField passwordField;  
    private JButton loginButton;  
    private JButton cancelButton;  
    private int attempts = 0;  
    private boolean loggedIn = false;  
    private String otp;  
    private JDialog dialog;  
    private JProgressBar progressBar;  
  
    public LoginDialog(JFrame parent) {  
        super(parent, "Login", true);  
  
        // Configurar la ventana para pantalla completa  
        setUndecorated(true);  
        setBounds(getGraphicsConfiguration().getBounds());  
  
        // Asegúrate de que la ruta a tu imagen es correcta  
        ImageIcon originalIcon = new ImageIcon("img/LOGOTIPO.png");  
  
        // Cambia el tamaño de la imagen  
        Image originalImage = originalIcon.getImage();  
        Image resizedImage = originalImage.getScaledInstance(700, 700, Image.SCALE_SMOOTH);  
        ImageIcon resizedIcon = new ImageIcon(resizedImage);  
  
        // Agregar el logo al panel  
        JLabel logoLabel = new JLabel(resizedIcon);  
        logoLabel.setVerticalAlignment(JLabel.NORTH);  
        logoLabel.setHorizontalAlignment(JLabel.CENTER);  
  
        // Establecer el logo como el contenido del panel  
        setContentPane(logoLabel);  
  
        // Pintar el fondo de blanco  
        logoLabel.setOpaque(true);  
        logoLabel.setBackground(Color.WHITE);  
  
        setLayout(new BorderLayout());  
  
        setLayout(new MigLayout("align center 95%", "[20[]", "[20[]"));  
        ///////////////////////////////  
        usernameField = new JTextField(20);  
        passwordField = new JPasswordField(20);  
        loginButton = new JButton("Login");  
        cancelButton = new JButton("Cancel");  
  
        loginButton.setPreferredSize(  
            new Dimension(cancelButton.getPreferredSize().width * 2, cancelButton.getPreferredSize().height));  
        loginButton.setBackground(Color.ORANGE);  
        cancelButton.setBackground(Color.LightGray);  
  
        add(new JLabel("Username:"), "cell 0 0,align right");  
        add(usernameField, "cell 1 0,align left,wrap");  
        add(new JLabel("Password:"), "cell 0 1,align right");  
    }  
}
```

```

        add(passwordField, "cell 1 1,align left,wrap");
        add(loginButton, "cell 0 2 2 1, align center"); // span two cells horizontally
        add(cancelButton, "cell 0 2, align center");

        passwordField.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                loginButton.doClick();
            }
        });

        loginButton.addActionListener(e -> {
            usuario_internoModel user = UserDAO.getUser(getUsername());

            // Se permiten tres intentos de login
            if (user == null) {
                JOptionPane.showMessageDialog(this, "Login incorrecto. Intenta de nuevo.", "Error",
                    JOptionPane.ERROR_MESSAGE);
            } else {

                if (PasswordUtils.checkPassword(getPassword(), user.getPassword())) {

                    if (ChatDAO.sesionIniciada(getUsername())) {
                        JOptionPane.showMessageDialog(this, "El usuario ya tiene una sesión iniciada en otro lugar.", "Error",
                            JOptionPane.ERROR_MESSAGE);
                    } else {
                        loggedIn = true;
                        MainApp.user = user;
                        String rol = user.getRol();

                        SwingWorker<Void, Void> worker = new SwingWorker<Void, Void>() {
                            @Override
                            protected Void doInBackground() {
                                // Simulate some work
                                for (int i = 0; i <= 100; i++) {
                                    final int progress = i;
                                    SwingUtilities.invokeLater(new Runnable() {
                                        @Override
                                        public void run() {
                                            progressBar.setValue(progress);
                                        }
                                    });
                                }
                                try {
                                    Thread.sleep(70); // Pause for a bit
                                } catch (InterruptedException e) {
                                    e.printStackTrace();
                                }
                            }
                            return null;
                        }

                        @Override
                        protected void done() {
                            dispose();
                            dialog.dispose();
                        }
                    }
                }
            }
        });
    }
}

```

```

        };
        worker.execute();
        launchProgressWindow();
    }

} else {
    attempts++;
    if (attempts < 3) {
        JOptionPane.showMessageDialog(this, "Login incorrecto. Intenta de nuevo.", "Error",
            JOptionPane.ERROR_MESSAGE);
    } else {
        otp = SendOTP.generateOTP();

        String email = UserDAO.getUserEmail(user.getId_usuario());
        SendOTP.sendEmail(otp, email);

        UIManager.put("OptionPane.background", Color.ORANGE);
        /*
         * * UIManager.put("Panel.background", Color.white);
         */ UIManager.put("OptionPane.messageFont", new Font("Segoe", Font.BOLD, 14));

        String otpInput = JOptionPane.showInputDialog(null,
            "<html><div style='text-align: center;'>" +
            + "<img src='./img/candado.png' width='200' height='200'></div>" +
            + "<h1>Has alcanzado el número máximo de intentos</h1><p>" +
            + "<Se ha enviado un OTP (One Time Password) a tu correo electrónico.<p>" +
            + "<b>Por favor, introduce el OTP:</b><br>" + "</html>",
            "OTP", JOptionPane.PLAIN_MESSAGE);

        if (otp.equals(otpInput)) {
            loggedIn = true;
            MainApp.user = user;
            setVisible(false);
        } else {
            JOptionPane.showMessageDialog(this, "OTP incorrecto. No puedes iniciar sesión.", "Error",
                JOptionPane.ERROR_MESSAGE);
            new Thread(new ShutdownHook()).start();
        }
    }
}
cancelButton.addActionListener(e -> setVisible(false));
}

public String getUsername() {
    return usernameField.getText();
}

public String getPassword() {
    return new String(passwordField.getPassword());
}

public boolean isLoggedIn() {
    return loggedIn;
}
<
```

```

    >     private void launchProgressDialog() {
    >         FlatProgressBarUI ui = new FlatProgressBarUI();
    >         progressBar = new JProgressBar();
    >         progressBar.setUI(ui);
    >         progressBar.setMinimum(0);
    >         progressBar.setMaximum(100);
    >         progressBar.setStringPainted(true);
    >         progressBar.setPreferredSize(new Dimension(progressBar.getPreferredSize().width, 20));
    >         Font font = progressBar.getFont();
    >         float newSize = font.getSize() * 1.5f;
    >         progressBar.setFont(font.deriveFont(newSize));
    >
    >         dialog = new JDialog((Frame) null, "Iniciando aplicación", true);
    >         dialog.getContentPane().add(progressBar, BorderLayout.CENTER);
    >         dialog.setSize(350, 70);
    >         dialog.setLocationRelativeTo(null);
    >         SwingUtilities.invokeLater(() -> dialog.setVisible(true));
    >
    >     }
    >
    >     public void updateProgress(int progress) {
    >         SwingUtilities.invokeLater(() -> progressBar.setValue(progress));
    >     }
    >
    > }

```

Interfaces de usuario

Lo primero que quiero comentar es que para el diseño de la interfaz de la aplicación en Java Swing me he apoyado en dos bibliotecas que han resultado ser muy útiles, Miglayout y FlatLaf.

MigLayout es una biblioteca de diseño de interfaz de usuario (UI) en Java que permite crear diseños flexibles y dinámicos para aplicaciones de escritorio. Proporciona un enfoque sencillo y potente para organizar los componentes de la interfaz de usuario en contenedores utilizando una sintaxis intuitiva y expresiva.

FlatLaf es una biblioteca de apariencia y estilo (look and feel) para aplicaciones de Java Swing que proporciona un aspecto moderno y plano inspirado en el diseño de interfaces de usuario de Material Design de Google. Permite aplicar un conjunto de estilos y colores consistentes a los componentes de la interfaz de usuario, lo que resulta en una apariencia más actualizada y atractiva.

La combinación de estas dos librerías me han permitido conseguir un diseño moderno y profesional sin necesidad de utilizar las extensiones gráficas para el diseño de la interfaz.

La interfaz gráfica se estructura de la siguiente manera, hay una pantalla principal dividida en dos partes donde la parte de la izquierda es donde se van a ir cargando los diferentes paneles o vistas con los que interactuemos y en la parte derecha, ocupando una franja del 25% del tamaño de la pantalla existen dos partes la superior donde está ubicado el chat y la inferior donde está el sistema de fichajes.

A continuación muestro una serie de pantallas de la interfaz grafica.

Pantalla de Login:



GARAGE GO

EL FUTURO DE TU TALLER

Username:

Password:

Pantalla principal:

MENÚ CONFIGURACIÓN AYUDA Gestión de taller

GARAGE GO

Bienvenido Miab S



Sala de Chat

[20:10] Miab S :
[19:36] Jonatan L : buenas tardes
[20:10] Miab S : iold
[20:10] Miab S : iold
[20:10] Miab S :
[05:19] YO : Hola

Send

Area personal

Hora actual: 05:19:44
Hora de entrada:
Hora de salida:
Tiempo trabajado:
NO TRABAJANDO

Iniciar jornada

Pantalla de clientes:

GARAGE GO

Gestión comercial

Clientes

Sala de Chat

[20:10] Miab S :
[19:36] Jonatan L : buenas tardes
[20:10] Miab S :
[20:10] Miab S :
[20:10] Miab S :
[20:10] Miab S :
[05:19] YO : Hola

Area personal

Hora actual: 05:20:19
Hora de entrada:
Hora de salida:
Tiempo trabajado:
NO TRABAJANDO
Iniciar jornada

Pantalla de vehículos:

GARAGE GO

Gestión comercial

Vehículos

Sala de Chat

[20:10] Miab S :
[19:36] Jonatan L : buenas tardes
[20:10] Miab S :
[20:10] Miab S :
[20:10] Miab S :
[20:10] Miab S :
[05:19] YO : Hola

Area personal

Hora actual: 05:22:02
Hora de entrada:
Hora de salida:
Tiempo trabajado:
NO TRABAJANDO
Iniciar jornada

Dialogo de creación de ordenes de trabajo:

Nueva Orden de Trabajo

TURISMO

Matrícula: 7548FTT

Marca: Audi **Modelo: 80**

Mantenimiento Reparación

Descripción:

Confirmar **Cancelar**

Pantalla de partes de reparación:

Gestión comercial

Partes de Reparación

Selección...	Parte	Mecánico	Matrícula	Cliente	Avería	Entrada	Salida	h Estimad..	h Reales	Presupues...	En reparación	Pres. Firm...
<input type="checkbox"/>	26	3	8668JYR	1234567...	Filtro antipartículas	2023-06...		3	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	33	2	7548FTT	8765432...	Embrague	2023-06...	2023-06...	2	2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	34	2	8668JYR	1234567...	Cadena de distrib...	2023-06...		4	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	36	2	1548M...	1234565...	Filtro antipartículas	2023-06...		4	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	37	2	8668JYR	1234567...	Mantenimiento 40k	2023-06...		5	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	40	2	7784TGH	1234567...	Filtro antipartículas	2023-06...		2	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	41	2	7784TGH	1234567...	Turbo	2023-06...		4	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Sala de Chat

- [20:10] Miab S :
- [19:36] Jonatan L : buenas tardes
- [20:10] Miab S :
- [20:10] Miab S : iold
- [20:10] Miab S :
- [20:10] Miab S : iold
- [20:10] Miab S :
- [05:19] YO : Hola

Área personal

Hora actual: 05:22:20

Hora de entrada:

Hora de salida:

Tiempo trabajado:

NO TRABAJANDO

Send

Crear Presupuesto **Generar Factura** **Firmar Parte**

Modo oscuro:

Gestión comercial

Partes de Reparación

Selección...	Parte	Mecánico	Matrícula	Cliente	Avería	Entrada	Salida	h Estimad..	h Reales	Presupues...	En reparación	Pres. Firm...
<input type="checkbox"/>	26	3	8668JYR	1234567...	Filtro antipartículas	2023-06...		3	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	33	2	7548FTT	8765432...	Embrague	2023-06...	2023-06...	2	2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	34	2	8668JYR	1234567...	Cadena de distrib...	2023-06...		4	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	36	2	1548M...	1234565...	Filtro antipartículas	2023-06...		4	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	37	2	8668JYR	1234567...	Mantenimiento 40k	2023-06...		5	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	40	2	7784TGH	1234567...	Filtro antipartículas	2023-06...		2	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	41	2	7784TGH	1234567...	Turbo	2023-06...		4	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Sala de Chat

- [20:10] Miab S :
- [19:36] Jonatan L : buenas tardes
- [20:10] Miab S :
- [20:10] Miab S : iold
- [20:10] Miab S :
- [20:10] Miab S : iold
- [20:10] Miab S :
- [05:19] YO : Hola

Área personal

Hora actual: 05:24:14

Hora de entrada:

Hora de salida:

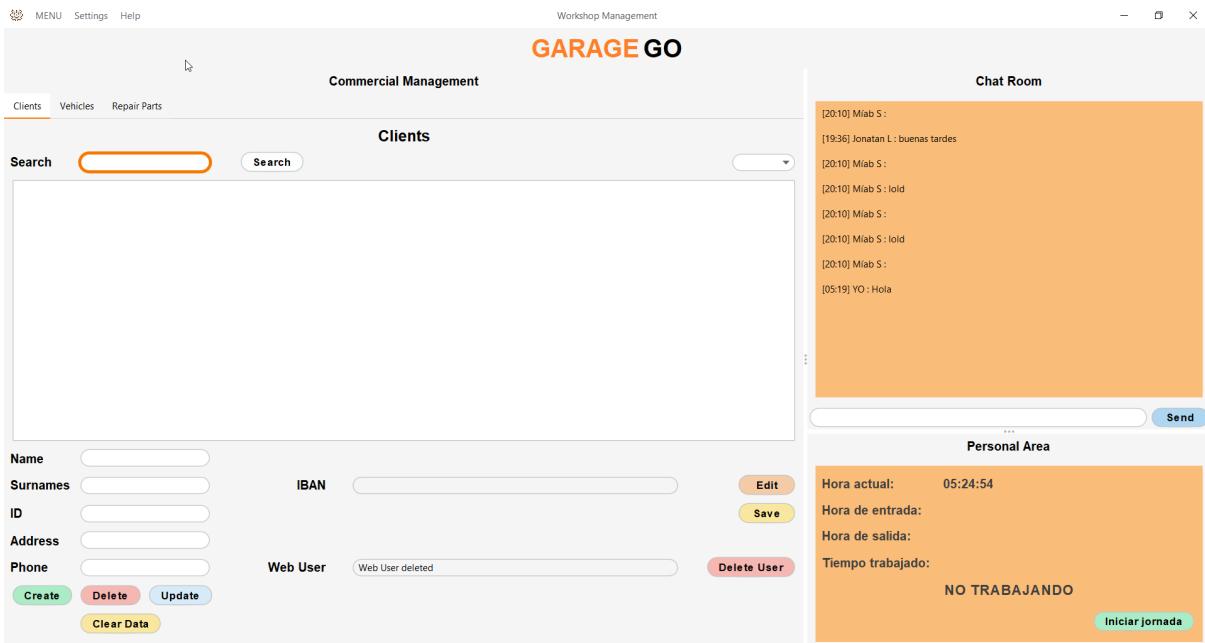
Tiempo trabajado:

NO TRABAJANDO

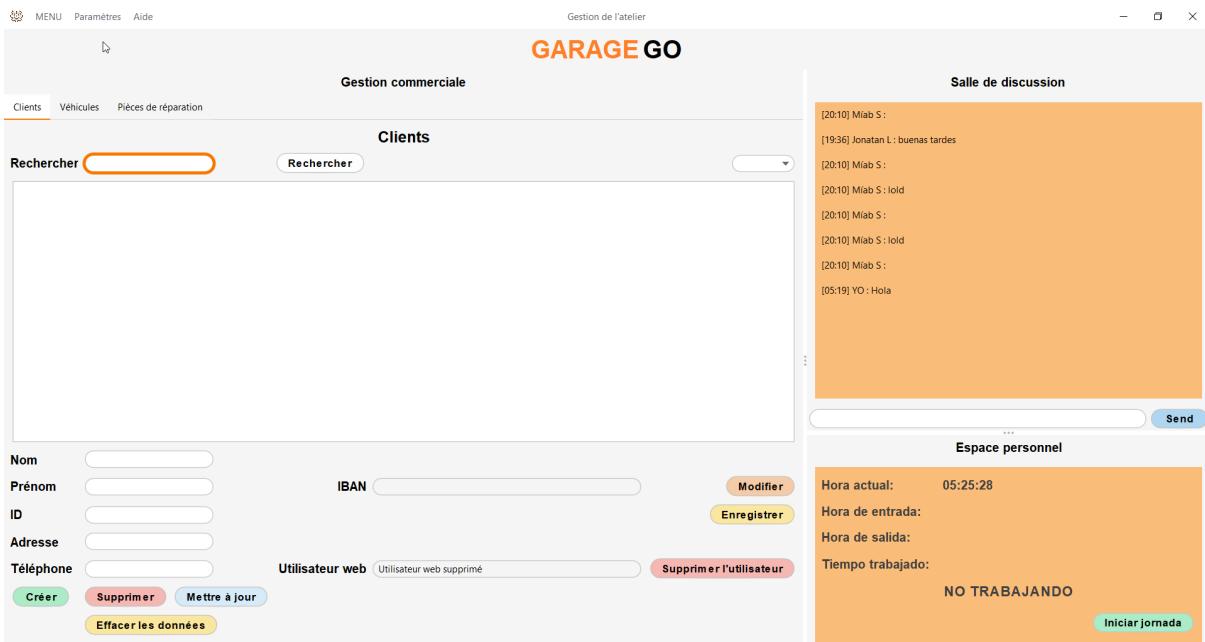
Send

Crear Presupuesto **Generar Factura** **Firmar Parte**

Aplicación traducida al inglés:



Aplicacion traducida al frances:



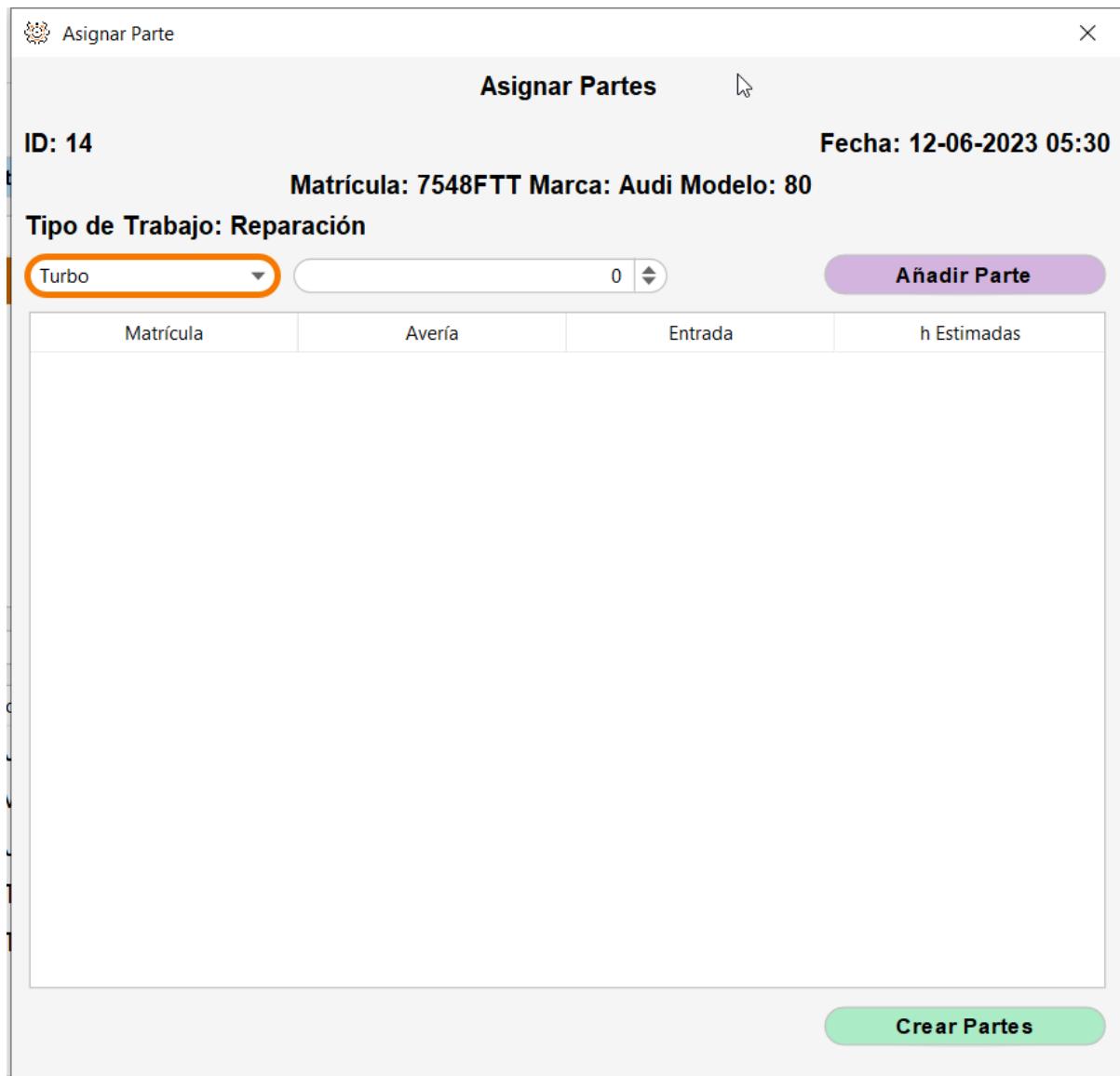
Pantalla mis partes (rol mecánico):

The screenshot shows the 'GARAGE GO' software interface. At the top, there's a menu bar with 'MENÚ', 'CONFIGURACIÓN', and 'AYUDA'. The main area is titled 'Gestión de taller' and 'Gestión comercial'. A central section is labeled 'Partes de Reparación' with a sub-section 'Órdenes de trabajo' and a 'Actualizar' button. Below this is a table titled 'Partes' with columns: Parte, Mecánico, Matrícula, Cliente, Reparación, Avería, Entrada, Salida, h Est., h Real, and Firmado. The table contains several rows of data. At the bottom of this section are buttons for 'Crear Parte', 'Eliminar parte', and 'Finalizar parte'. To the right, there's a 'Sala de Chat' window showing a log of messages from users like 'Miab S', 'Jonatan L', and 'Miab L'. Below it is an 'Area personal' section with fields for 'Hora actual', 'Hora de entrada', 'Hora de salida', and a status indicator 'NO TRABAJANDO' with a 'Iniciar jornada' button.

Dialogo nuevo parte:

This is a modal dialog box titled 'Nuevo Parte'. The main title is 'Crear nuevo parte'. Inside, there is a large input field with a dropdown arrow icon. To its right is a numeric input field with a spin button. At the bottom of the dialog are two buttons: a green 'Crear' button and a red 'Cancelar' button. The background of the dialog is white, and it has a standard window frame with a close button in the top right corner.

Diálogo de asignación de partes:



Seguridad

La seguridad es un aspecto crítico en nuestra aplicación de gestión de talleres de coche, ya que maneja datos confidenciales de los clientes y realiza transacciones financieras. En este informe, se detalla cómo hemos implementado diversas medidas de seguridad para garantizar la protección de los datos y prevenir accesos no autorizados.

Autenticación y Control de Acceso:

Se ha implementado un sólido sistema de autenticación que requiere que los usuarios ingresen sus credenciales (nombre de usuario y contraseña) para acceder a

la aplicación. Se utilizan algoritmos de hash seguros y políticas de contraseñas robustas para almacenar y proteger las contraseñas de forma adecuada.

Adicionalmente, hemos implementado un mecanismo de OTP (One-Time Password) para aumentar la seguridad en caso de intentos fallidos de inicio de sesión. Después de tres intentos de inicio de sesión fallidos, se genera y envía un código de un solo uso al correo electrónico o al número de teléfono registrado del usuario. Este código OTP debe ingresarse correctamente para permitir el acceso nuevamente, lo que agrega una capa adicional de seguridad y protección contra ataques de fuerza bruta.

Protección de Datos Sensibles:

Hemos implementado medidas de seguridad para proteger los datos sensibles de los clientes, como nombres, direcciones y números de teléfono. Utilizamos técnicas de encriptación y cifrado para proteger la confidencialidad de la información almacenada en la base de datos SQL remota.

Además, se establecen restricciones de acceso a nivel de base de datos y se realizan copias de seguridad periódicas para garantizar la integridad y disponibilidad de los datos en caso de cualquier evento inesperado.

Comunicación Segura:

Para garantizar la privacidad de la comunicación entre la aplicación y los servidores remotos, utilizamos una capa de sockets seguros (SSL/TLS) para cifrar los datos en tránsito. Esto evita que los datos confidenciales se vean comprometidos durante la transmisión, protegiendo la información sensible de posibles ataques de interceptación.

Actualizaciones y Parches de Seguridad:

Implementamos un proceso regular de actualización y aplicación de parches de seguridad para mantener la aplicación protegida contra vulnerabilidades conocidas. Monitoreamos activamente las actualizaciones de seguridad de los componentes utilizados en la aplicación, como el sistema operativo, frameworks y bibliotecas de terceros. De esta manera, mitigar cualquier riesgo potencial y nos aseguramos de mantener la seguridad actualizada.

Conclusiones:

La implementación de medidas sólidas de seguridad en nuestra aplicación de gestión de talleres de coche nos permite proteger los datos confidenciales de los clientes y garantizar la confianza de los usuarios. La autenticación robusta, el control de acceso, la protección de datos sensibles, el uso de OTP y la comunicación segura son elementos fundamentales de nuestro enfoque de seguridad.

Continuaremos evaluando y mejorando nuestras medidas de seguridad para adaptarnos a las nuevas amenazas y vulnerabilidades emergentes, asegurando que

nuestra aplicación mantenga un alto nivel de protección para los datos y la privacidad de nuestros usuarios.

OTP

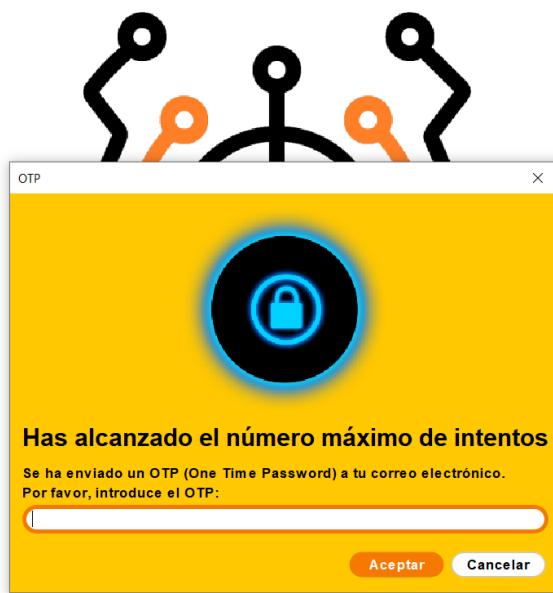


GARAGE GO

EL FUTURO DE TU TALLER

Username:

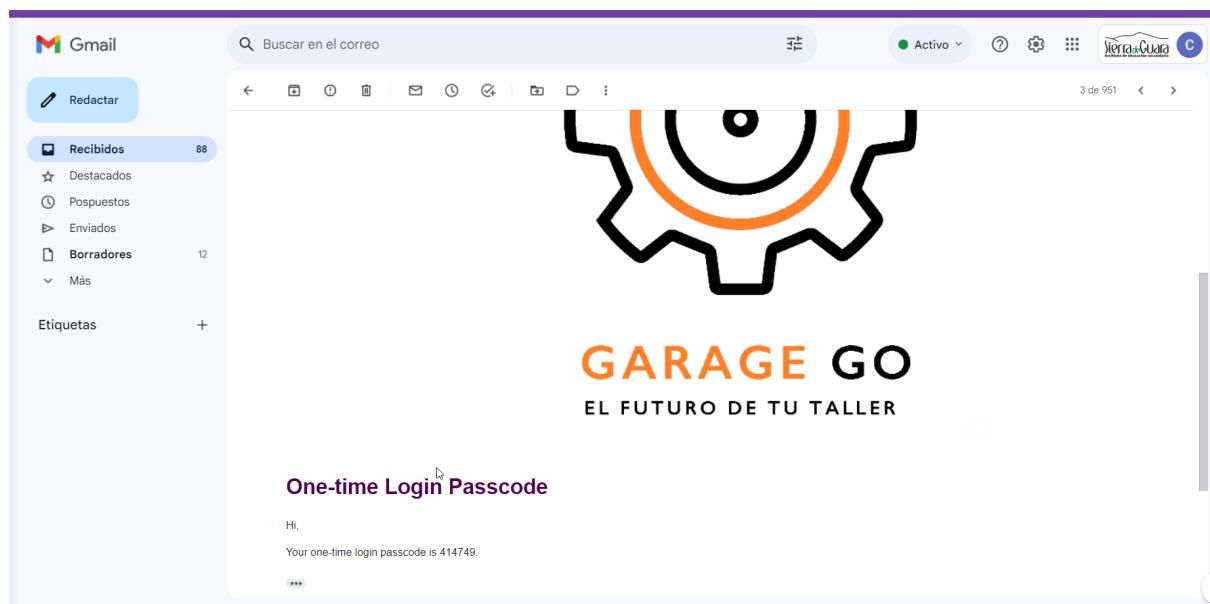
Password:



EL FUTURO DE TU TALLER

Username:

Password:



Verificación de la aplicación

Acceso:



GARAGE GO
EL FUTURO DE TU TALLER

Username:

Password:

 **Login**

Cancel



GARAGE GO

EL FUTURO DE TU TALLER

Username:

Password:

Login

Cancel

Fichaje:



Clientes:

Gestion comercial

Clients Vehículos Partes de reparación

Clients

Buscar Buscar ▾

Nombre
Apellidos
DNI
Dirección
Teléfono

iban Editar
Guarda

Usuario web Dar de baja

Crear Eliminar Actualizar Limpiar datos

Error
! Error al eliminar el cliente.
Aceptar

Nombre pepe
Apellidos popo
DNI rrrrrrrrrrrr
Dirección calella
Teléfono 987554455

iban
Usuario web

Crear Eliminar Actualizar Limpiar datos

Error de validación
! El DNI debe tener un formato válido
Aceptar

Error de validación

! El teléfono debe tener un formato válido

Aceptar

Nombre: pepe

Apellidos: popo

DNI: 12345678p

Dirección: calella

Teléfono: 9875544ttt

iban:

Usuario web:

Crear **Eliminar** **Actualizar**

Limpiar datos

Windows Taskbar: Buscar, Icons, Start button, File Explorer, Mail, Edge, Photos, OneDrive, Power User

Vehículos:

Error

! No se pudo eliminar el vehículo

Aceptar

Tipo: ▼

Popietario: ▼

Marca: ▼

Matricula:

Modelo: ▼

Año: 2.023 ▲▼

Kilometros:

Crear **Eliminar** **Actualizar**

Limpiar datos

Windows Taskbar: Buscar, Icons, Start button, File Explorer, Mail, Edge, Photos, OneDrive, Power User

1234LKJ	12345678A	MOTOCICLETA	Ducati	1198 S	2008
1478PPJ	12345678A	TURISMO	Hyundai	Santa Fe	2017
1548MMB	12345654E	TURISMO	Bmw	Compact	2020
7548FTT	87654321B	TURISMO	Audi	80	2022
7784TGH	12345678A	TURISMO	Daewoo	Nexia	2022
8668JYR	12345678A	TURISMO	Volkswagen	Touren	2018
9986PPM	87654321B			e	1990

Error de validación
Todos los campos deben estar completos.

Aceptar

Tipo	TURISMO	Popietario	Rodrigo Plá, Carlos - 12345678A
Marca	Bmw	Matricula	7784tgh
Modelo	Serie 3		
Año	2.022		
Kilometros	██████████		
<input type="button" value="Crear"/> <input type="button" value="Eliminar"/> <input type="button" value="Actualizar"/> <input type="button" value="Limpiar datos"/>			

Partes:

Partes de Reparación													
Buscar	Buscar	Parte	Mecánico	Matrícula	Cliente	Avería	Entrada	Salida	h Estimadas	h Reales	Presupuest...	En reparaci...	Pres. Firma...
<input type="checkbox"/>		33	2	7548FTT	87654321B	Embrague	2023-06-06	2023-06-11	2	2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Error ! Error, ningún parte seleccionado para firmar. Aceptar													

Crear Presupuesto **Generar Factura** **Firmar Parte**

Buscar Buscar

Selección...	Parte	Mecánico	Matrícula	Cliente	Avería	Entrada	Salida	h Estimad...	h Reales	Presupues...	En repar...	Pres. Firm...
<input type="checkbox"/>	26	3	8668JYR	1234567...	Filtro antipartículas	2023-06...		3	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	33	2	7548FTT	8765432...	Embrague	2023-06...	2023-06...	2	2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	34	2	8668JYR	1234567...	Cadena de distrib...	2023-06...		4	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	36	2	1548M...	1234565...	Filtro antipartículas	2023-06...		4	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	37	2	8668JYR	1234567...	Mantenimiento 40k	2023-06...		5	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	40	2	7784TGH	123456...				2	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	41	2	7784TGH	123456...				4	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Error
! No se ha seleccionado ningún parte
Aceptar

Crear Presupuesto Generar Factura Firmar Parte

Mis partes(rol mecanico):

MENÚ CONFIGURACIÓN AYUDA Gestión de taller GARAGE GO Gestión comercial

Mis partes

Órdenes de trabajo Actualizar Partes de Reparación Asignar Partes

Error ! Selecciona una fila para eliminar Aceptar

Partes

Parte	Mecánico	Matrícula	Cliente	Reparación	Avería	Entrada	Salida	h Est.	h Real	Firmado
34	2	8668JYR	12345678A	<input checked="" type="checkbox"/>	Cadena de distribución	2023-06-08		4	0	<input checked="" type="checkbox"/>
36	2	1548MMB	12345654E	<input checked="" type="checkbox"/>	Filtro antipartículas	2023-06-08		4	0	<input checked="" type="checkbox"/>
37	2	8668JYR	12345678A	<input checked="" type="checkbox"/>	Mantenimiento 40k	2023-06-08		5	0	<input checked="" type="checkbox"/>
40	2	7784TGH	12345678F	<input checked="" type="checkbox"/>	Filtro antipartículas	2023-06-11		2	0	<input type="checkbox"/>
41	2	7784TGH	12345678F	<input checked="" type="checkbox"/>	Turbo	2023-06-11		4	0	<input type="checkbox"/>

Crear Parte Eliminar parte Finalizar parte

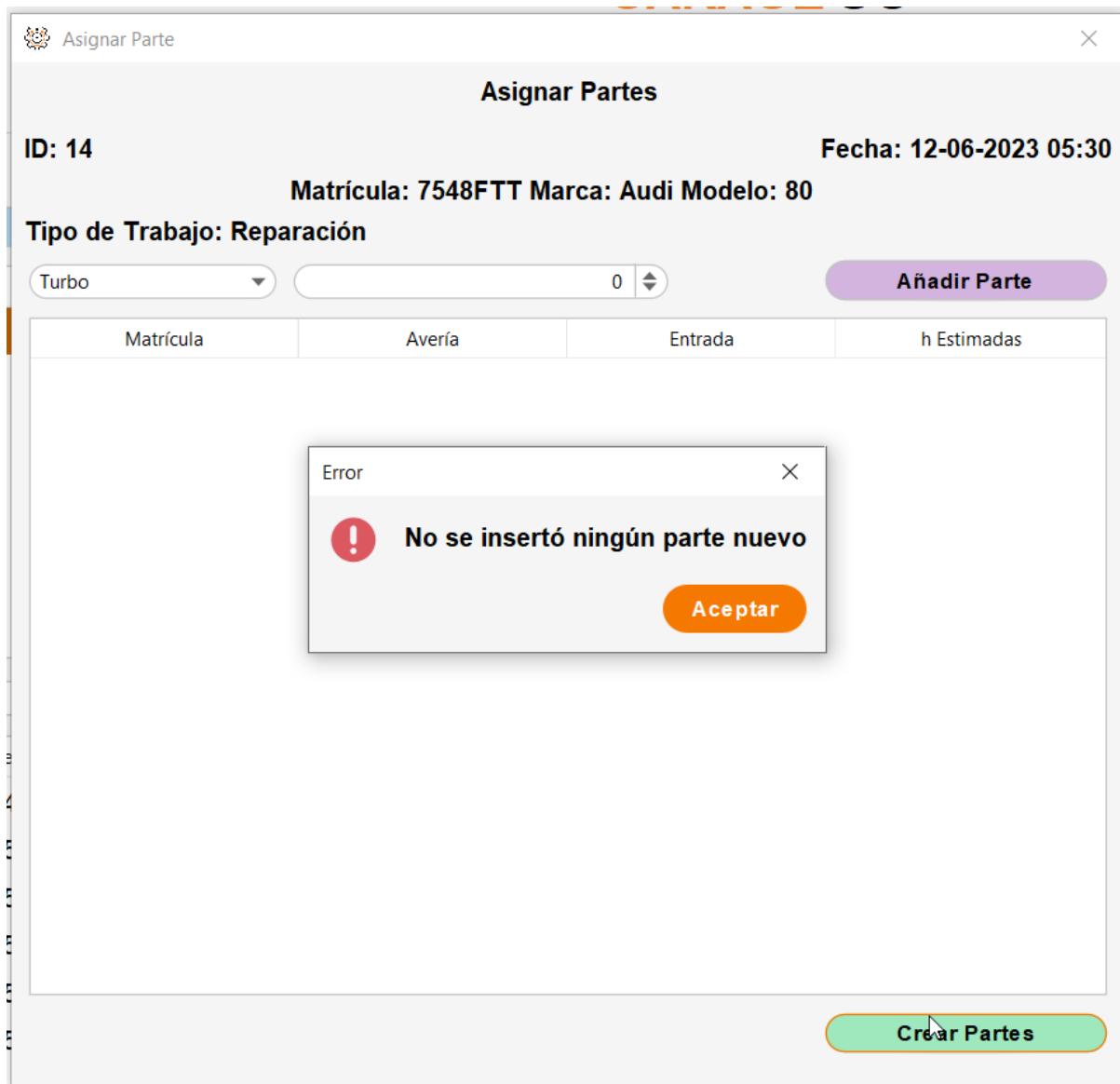
Sala de Chat

- [20:10] Miab S:
- [20:10] Miab S:
- [19:36] Jonatan L: buenas tardes
- [20:10] Miab S:
- [20:10] Miab S : iold
- [20:10] Miab S :
- [20:10] Miab S : iold
- [20:10] Miab S :
- [05:19] Miab S : Hola

Area personal

Hora actual: 06:03:18
Hora de entrada:
Hora de salida:
Tiempo trabajado:
NO TRABAJANDO Iniciar jornada

Diálogo Asignar partes:



Abstract

The project aims to develop a comprehensive desktop and web application for managing car workshops. The desktop application allows workshop staff to register clients, vehicles, create work orders, and control income through invoices and estimates. Different user roles are implemented to restrict access to specific functionalities based on assigned responsibilities. Additionally, a real-time communication system is integrated, enabling employees to collaborate efficiently through a chat implemented with sockets. The application also features a time-tracking system to monitor employees' working hours directly within the system. All data is securely stored in a remote SQL database, ensuring accessibility from various locations. Firebase, a distributed storage system, is utilized for storing heavy files such as photos and PDFs in the cloud.

The web application, developed with Spring, provides clients with access to their data, work orders, vehicles, invoices, and estimates, enabling them to sign work orders and even make payments. By offering a user-friendly interface and seamless navigation, the application enhances customer engagement and satisfaction. The implementation of the Model-View-Controller (MVC) software design pattern in the web application ensures separation of concerns, making the application more maintainable, reusable, and scalable.

The project provides numerous advantages compared to other similar solutions in the market. It offers a comprehensive set of features, including client and vehicle management, work order creation, income tracking, real-time communication, and integrated time-tracking. The integration of both desktop and web applications provides a seamless user experience and enables efficient data management across different platforms.

By adopting a SQL database, the project ensures data integrity, consistency, and reliability. SQL databases provide robust query capabilities, transactional support, and data integrity constraints, making them well-suited for managing complex data relationships and ensuring data accuracy.

In conclusion, the project offers a powerful and user-friendly solution for managing car workshops through its desktop and web applications. By leveraging the capabilities of SQL databases and advanced technologies such as Spring and Firebase, the project provides an efficient and secure platform for workshop management, enhancing productivity, collaboration, and customer satisfaction.

Partes del proyecto que han supuesto mayor dificultad

Sin duda alguna la mayor dificultad que he tenido que superar ha sido el intento frustrado de desarrollar un chat en Java que pudiese integrarse en un sistema Android. De hecho tuve que descartar mi idea inicial de implementar una aplicación Android para que fuese utilizada por los mecánicos en el taller debido a este y otros motivos que surgieron en el intento de integración. Teniendo incluso que abandonar varias versiones del chat que funcionaban en la aplicación de escritorio. Finalmente opté por abandonar la idea de integrar la apk de android en mi proyecto. A continuación describiré las diferentes opciones que he intentado y los motivos por los que me ha sido imposible conseguir este objetivo.

Intento fallido de establecer un chat entre android y la aplicación de escritorio

En la primera versión del chat, se utilizó una solución basada en multicast para permitir la comunicación entre múltiples clientes. Si bien en la aplicación de escritorio funcionaba perfectamente, fue imposible conseguir que hubiera comunicación con una aplicación en Android. Durante el proceso de desarrollo y pruebas, se descubrió que esta solución no era compatible con las aplicaciones Android a las que se deseaba conectar. A continuación, se explicarán en detalle las razones detrás de este cambio y la necesidad de adoptar una nueva solución.

Incompatibilidad de redes: El multicast se basa en la transmisión de paquetes a un grupo de direcciones IP específicas. Si bien el multicast es ampliamente compatible en redes cableadas y en algunos entornos de red locales, no todas las redes, especialmente las redes móviles y algunas configuraciones de redes corporativas, admiten el multicast. Esto significa que los dispositivos Android, que a menudo se conectan a través de redes móviles o redes empresariales, no podrían participar en el chat utilizando esta solución.

Limitaciones de las políticas de firewall: En muchos entornos de red, los firewalls y las políticas de seguridad pueden bloquear el tráfico multicast. Esto puede deberse a restricciones de seguridad o a configuraciones específicas del entorno. Como resultado, los dispositivos Android que se encuentren detrás de un firewall o que estén conectados a una red con políticas restrictivas no podrán recibir los mensajes multicast, lo que los excluye del chat.

Soporte limitado en Android: Si bien el multicast puede ser compatible con ciertos sistemas operativos y plataformas, su soporte en dispositivos Android es limitado. Android no proporciona una interfaz de nivel de API estándar para el manejo de multicast, lo que dificulta su implementación y uso en aplicaciones Android. Esto significa que desarrollar una solución de chat basada en multicast requeriría un esfuerzo adicional y personalizado para garantizar la compatibilidad con dispositivos Android.

Debido a estas limitaciones y desafíos técnicos, se decidió abandonar la solución inicial basada en multicast y buscar una alternativa que fuera compatible con dispositivos Android. La solución final adoptada involucró el uso de sockets TCP/IP para establecer conexiones punto a punto entre los clientes y el servidor, lo que permitió una comunicación fiable y compatible en todas las plataformas, incluidas las aplicaciones Android.

La elección de los sockets TCP/IP ofrecía varias ventajas:

Amplia compatibilidad: Los sockets TCP/IP son ampliamente compatibles en todas las plataformas, incluidos los dispositivos Android. Esto asegura que los clientes puedan conectarse y participar en el chat sin limitaciones de compatibilidad de red.

Comunicación fiable: Los sockets TCP/IP proporcionan una comunicación fiable y garantizan que los mensajes se entreguen en el orden correcto y sin pérdidas. Esto es especialmente importante en un chat donde la integridad y la secuencia de los mensajes son fundamentales.

Flexibilidad en la configuración de red: Al utilizar sockets TCP/IP, los clientes no están restringidos por las limitaciones del multicast o las políticas de firewall. Pueden conectarse directamente al servidor, lo que facilita la configuración en diferentes entornos de red y evita posibles bloqueos.

En resumen, la incompatibilidad del multicast con las aplicaciones Android y las limitaciones en la compatibilidad de redes y el soporte en dispositivos Android fueron las razones principales detrás del abandono de la primera solución basada en multicast. La necesidad de garantizar la compatibilidad con dispositivos Android, así como la comunicación confiable y flexible en entornos de red diversos, impulsaron la búsqueda de una alternativa más adecuada.

Implementando esta solución conseguí que hubiera comunicación entre android y la aplicación de escritorio pero no se por que, solo había comunicación en un sentido, es decir los mensajes enviados desde Android se recibían en java pero en android no se recibía ninguna comunicación enviada desde la aplicación de escritorio. Finalmente decidí aparcar el tema temporalmente e implementar otros aspectos de mi proyecto, como el acceso a mi base de datos SQL remota que estaba utilizando desde el inicio de mi proyecto, al intentar conectarme desde android a esta, surgieron nuevos problemas que describo en el siguiente apartado.

Intento fallido de conexión directa a base de datos SQL remota desde Android

A continuación, describiré cada paso que he seguido y justificaré las decisiones tomadas, así como las razones por las que no he logrado que funcione correctamente. Mi objetivo es presentar una memoria completa y detallada que demuestre mi esfuerzo y conocimiento en el desarrollo de esta funcionalidad.

En primer lugar, opté por utilizar el controlador JDBC de MySQL, ya que es una solución comúnmente utilizada para establecer conexiones con bases de datos MySQL desde aplicaciones Java. Investigando en la documentación oficial de Android, descubrí que el controlador JDBC no es compatible directamente con

Android debido a diferencias en la forma en que se manejan las conexiones y los recursos en la plataforma. A pesar de esto, decidí intentarlo inicialmente, creyendo que podría encontrar una solución o una alternativa que me permitiera hacerlo funcionar.

Después de implementar el controlador JDBC de MySQL en mi aplicación y configurar la conexión utilizando la URL, el nombre de usuario y la contraseña adecuados, me encontré con problemas. Durante la ejecución, se produjo una excepción que indicaba que el controlador JDBC no se podía encontrar. Investigando más a fondo, descubrí que la incompatibilidad entre el controlador JDBC y Android es un problema conocido y ampliamente documentado. Aunque esto fue frustrante, decidí cambiar de enfoque y buscar alternativas que fueran compatibles con Android.

Mi siguiente intento fue utilizar el controlador de MariaDB en lugar del controlador JDBC de MySQL. Investigando en la documentación oficial de MariaDB, encontré que proporcionan un controlador específico para Android que se puede utilizar para establecer conexiones con bases de datos MariaDB desde aplicaciones móviles. Siguiendo los pasos recomendados, adapté mi clase de Conexión para utilizar el controlador de MariaDB y la URL de conexión adecuada. Sin embargo, cuando ejecuté la aplicación, me encontré con otro error que indicaba que no se podía establecer la conexión. Este error fue desconcertante y me llevó a creer que puede haber problemas adicionales en la configuración o en la compatibilidad del controlador de MariaDB con la versión de Android que estaba utilizando.

En un último intento, investigué la posibilidad de utilizar el controlador JTDS, que es compatible con SQL Server. Decidí probarlo como una opción alternativa, pensando que podría funcionar en mi caso, ya que el controlador JTDS se ha utilizado con éxito en aplicaciones Android en el pasado. Siguiendo las instrucciones, modifiqué mi clase Conexion para utilizar el controlador JTDS y la URL de conexión correspondiente. Sin embargo, al ejecutar la aplicación, me encontré con un error relacionado con el tipo de paquete desconocido, lo que indicaba que no se podía establecer la conexión correctamente.

A pesar de mis esfuerzos y los diferentes enfoques que he intentado, no he logrado establecer una conexión exitosa desde la aplicación de Android a la base de datos remota hasta el momento. Las limitaciones de las bibliotecas y controladores disponibles para Android, junto con las posibles incompatibilidades y problemas de configuración, han sido los principales obstáculos.

Conclusión

Debido a los problemas encontrados en la aplicación de android para implementar dos aspectos que eran fundamentales en el desarrollo de mi proyecto; la implementación de una vía de comunicación entre el taller y la oficina a través de un chat y el acceso remoto a una base de datos SQL, en la que se basaba el resto de mi

aplicación, tuve que tomar la difícil decisión de abandonar el desarrollo de la aplicación android y no integrarla en mi proyecto. Sin duda esta ha sido una de las mayores frustraciones a las que me he enfrentado desarrollando el proyecto, tanto por no poder integrar una parte que en un principio se me antojaba fundamental en el mismo, como por la cantidad de tiempo invertido que al final no ha tenido ningún resultado.

Tecnologías utilizadas:



mikaelgrev/
miglayout

Official MIG Layout for Swing, SWT and JavaFX

JFormDesigner/
FlatLaf

FlatLaf - Swing Look and Feel (with Darcula/IntelliJ themes support)



Presupuesto Detallado para el Desarrollo, Implementación y Mantenimiento del Proyecto de Gestión de Talleres de Coche

Aspectos a tener en cuenta para realizar el presupuesto:

Desarrollo del Proyecto

Análisis y Diseño del Sistema: Esta etapa implica el análisis de requisitos, la definición de casos de uso y la elaboración de los diseños arquitectónicos y de interfaz de usuario.

Desarrollo de la Aplicación de Escritorio:

Aquí se llevará a cabo la programación y la implementación de todas las funcionalidades del sistema, como el registro de clientes y vehículos, creación de partes, control de ingresos, sistema de chat, sistema de fichajes, integración de la base de datos SQL remota y sistema de almacenamiento distribuido Firebase. .

Pruebas y Control de Calidad:

Se realizarán pruebas exhaustivas para garantizar que la aplicación funcione correctamente, detectar y corregir errores y asegurar la calidad del software.

Implantación del Proyecto

Instalación y Configuración del Sistema: Esta etapa abarca la instalación de la aplicación de escritorio en los equipos del taller, así como la configuración de los parámetros y las integraciones necesarias.

Migración de Datos:

Si es necesario, se migrarán los datos existentes del taller al nuevo sistema. Esto implica la extracción, transformación y carga de los datos en la base de datos SQL remota.

Capacitación y Formación del Personal:

Es crucial brindar capacitación y formación adecuadas al personal del taller para garantizar una adopción exitosa de la nueva aplicación. Esto puede incluir la creación de materiales de capacitación, la realización de sesiones de formación y el soporte inicial.

Mantenimiento del Proyecto

Actualizaciones y Mejoras: Con el tiempo, pueden surgir nuevas funcionalidades o mejoras en el sistema.

Soporte Técnico: Es importante contar con un equipo de soporte técnico para brindar asistencia a los usuarios, resolver problemas y asegurar un funcionamiento sin problemas del sistema.

Además de estos costes laborales, también habría que considerar los costes de licencias de software, infraestructura (servidores, equipos, redes), posibles costes de consultoría adicional y servicios de terceros, como servicios de nube y seguridad.

Cuantificación

Desarrollo del Proyecto:

- Análisis y Diseño del Sistema: Entre 50 y 100 horas de trabajo de analistas y consultores especializados a un costo aproximado de € 1.346 - € 2.692 (50-100 horas * € 26,92/hora).
- Desarrollo de la Aplicación de Escritorio: Entre 200 y 500 horas de trabajo de desarrolladores y programadores a un costo aproximado de € 10.769 - € 26.923 (200-500 horas * € 53,85/hora).
- Pruebas y Control de Calidad: Entre 50 y 100 horas de trabajo de testers y especialistas en control de calidad a un costo aproximado de € 1.327 - € 2.654 (50-100 horas * € 26,54/hora).

Implantación del Proyecto:

- Instalación y Configuración del Sistema: Entre 20 y 40 horas de trabajo de técnicos de instalación y configuración a un costo aproximado de € 423 - € 846 (20-40 horas * € 21,15/hora).
- Migración de Datos: Entre 40 y 80 horas de trabajo de consultores especializados en migración de datos. No pude encontrar información específica sobre el salario medio de consultores especializados en migración de datos en España por lo tanto para los cálculos tomaré de referencia en anterior. Costo aproximado € 846 - € 1692 (40-80 horas* € 21,15/hora)
- Capacitación y Formación del Personal: Entre 30 y 60 horas de trabajo de formadores y consultores de capacitación a un costo aproximado de € 1.038 - € 2.077 (30-60 horas * € 34,62/hora).

Mantenimiento del Proyecto:

- Actualizaciones y Mejoras: Dependiendo de la frecuencia y la complejidad de las actualizaciones, se puede estimar entre 50 y 100 horas de trabajo de

desarrolladores y programadores por año a un costo aproximado de € 2.692 - € 5.385 (50-100 horas * € 53,85/hora) por año.

- Soporte Técnico: Esto puede variar según las necesidades, pero una estimación inicial sería de 10 a 20 horas de trabajo de técnicos de soporte por mes a un costo aproximado de € 188 - € 375 (10-20 horas * € 18,75/hora) por mes.

Presupuesto

Desarrollo del Proyecto:

● Análisis y Diseño del Sistema	€ 1.346	€ 2.692
● Desarrollo de la Aplicación de Escritorio	€ 10.769	€ 26.923
● Pruebas y Control de Calidad	€ 1.327	€ 2.654

Implantación del Proyecto:

● Instalación y Configuración del Sistema:	€ 423	€ 846
● Migración de Datos	€ 1.038	€ 2.077
● Pruebas y Control de Calidad	€ 1.327	€ 2.654
● Capacitación y Formación del Personal	€ 1.038	€ 2.077

Mantenimiento del Proyecto:

● Actualizaciones y Mejoras:	€ 2.692	€ 5.385
● Soporte Técnico	€ 188	€ 375

TOTAL ESTIMADO (MIN/MAX): € 20148 € 45683

Esto son costos estimados basados en los salarios medios en España para los roles requeridos. Estos costos no incluyen costos adicionales como impuestos, seguros sociales, beneficios para empleados, etc. Además, estos costos podrían variar según la ubicación geográfica, la experiencia y otros factores.

Manual

El manual ha sido realizado con flipsnack, como es una cuenta gratuita no se puede descargar pero se puede acceder a través del siguiente enlace o escaneando el código QR:

<https://www.flipsnack.com/9D9DCF99E8C/pruebagaragego.html>



Bibliografía

1. Java Documentation - Oracle: Documentación oficial de Java. Disponible en: <https://docs.oracle.com/en/java/>
2. Spring Framework Documentation - Spring: Documentación oficial de Spring Framework. Disponible en: <https://spring.io/docs>
3. MigLayout Documentation - MiG InfoCom AB: Documentación oficial de MigLayout. Disponible en: <https://www.miglayout.com/docs/>
4. FlatLaf Documentation - FormDev Software: Documentación oficial de FlatLaf. Disponible en: <https://www.formdev.com/flatlaf/>
5. Firebase Documentation - Firebase, Inc.: Documentación oficial de Firebase. Disponible en: <https://firebase.com/docs/>
6. SendGrid Documentation - SendGrid: Documentación oficial de SendGrid. Disponible en: <https://sendgrid.com/docs/>
7. DB4free Documentation - db4free.net: Documentación oficial de db4free. Disponible en: <https://www.db4free.net/documentation.php>
8. Stack Overflow: Plataforma en línea de preguntas y respuestas relacionadas con programación y desarrollo de software. Disponible en: <https://stackoverflow.com/>
9. ChatGPT - OpenAI: ChatGPT es un modelo de lenguaje desarrollado por OpenAI que ofrece respuestas y asistencia en diversas áreas, incluyendo programación. Disponible en: <https://www.openai.com/>
10. Bing - Microsoft: Motor de búsqueda en línea que proporciona acceso a una amplia variedad de recursos, tutoriales y preguntas frecuentes relacionadas con la programación en Java. Disponible en: <https://www.bing.com/>
11. Google: Motor de búsqueda en línea que ofrece una amplia gama de recursos, tutoriales, libros y foros de programación en Java. Disponible en: <https://www.google.com/>
12. "Effective Java" de Joshua Bloch
13. "Spring in Action" de Craig Walls
14. "Clean Code: A Handbook of Agile Software Craftsmanship" by Robert C. Martin

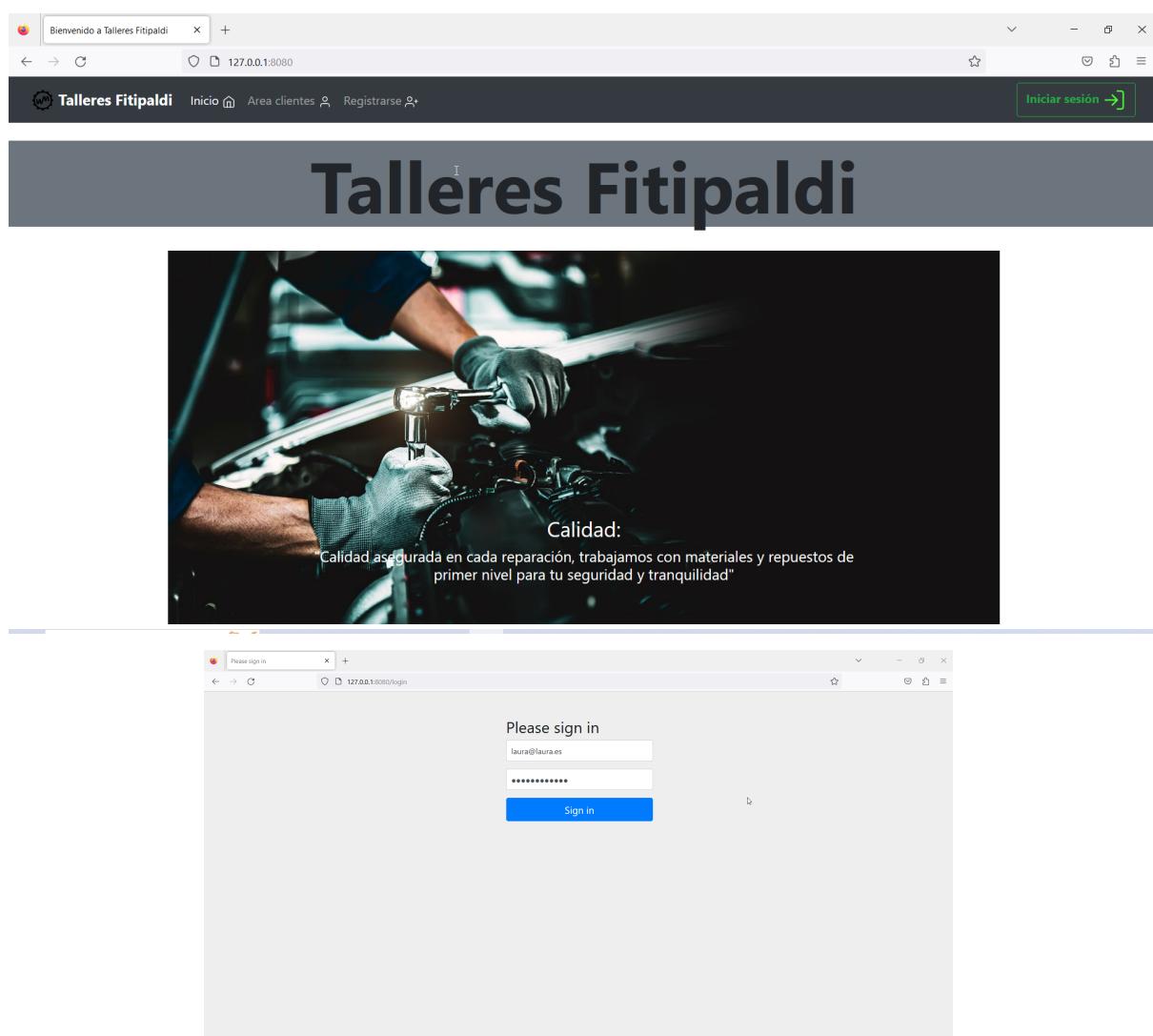
Anexo Proyecto SPRING web Application

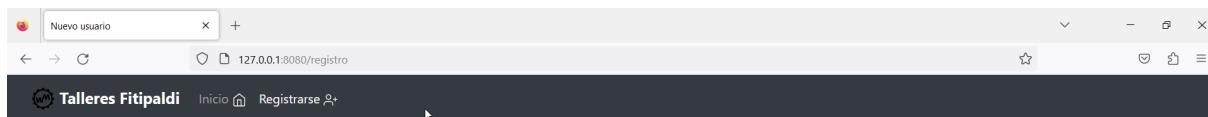
Este es un pequeño resumen de la aplicación web para los clientes del taller de coches. Adjunto capturas de pantalla de las diferentes páginas de la interfaz gráfica y también una muestra del código de la aplicación. La clase controlador que es la que realiza todas las peticiones GET/POST y la página Vehiculos.html como muestra del código de la aplicación.

Consideraciones: La aplicación desarrollada con spring utiliza thymeleaf como gestor de plantillas y usa algo de javascript y css también. La aplicación se conecta a la misma base de datos que la aplicación de escritorio y además de consultas realiza algunas operaciones CRUD.

Como he explicado en la memoria, sirve como complemento del programa principal y para que los clientes reciban feedback del taller sobre sus vehículos, reparaciones ect.

Capturas de la interfaz:





Registro de usuario - Registrarse

Email
Contraseña
DNI
Registrarse

The screenshot shows a browser window with the following details:

- Title bar: Área de clientes
- Address bar: 127.0.0.1:8080/cliente
- Page header: Talleres Fitipaldi, Inicio, Reparaciones, Vehículos, Datos personales
- User info: Bienvenido Laura Martinez Plá, Cerrar sesión
- Main content:
 - Datos de acceso**

E-mail	Contraseña
laura@laura.es	\$2a\$10\$R4AMKSOwTMueGclzOpLQeu6Kdae2vE99lmDPjAgzld3TYlzDtRhK
 - Datos Personales**

Nombre	Laura
Apellidos	Martinez Plá
DNI	87654321B
Teléfono	987498989
Dirección	Calle Tenerías, 19, 3º3ª, 22004 Huesca

Vehículos								
eliminar	ver	foto	matrícula	marca	modelo	tipo	año	kilómetros
Eliminar	Ver		7548FTT	Audi	80	TURISMO	2022	100000
Eliminar	Ver		9986PPM	Cadillac	Seville	TURISMO	1990	153000
Nuevo vehículo								

Vehículos								
eliminar	ver	foto	matrícula	marca	modelo	tipo	año	kilómetros
Eliminar	Ver		7548FTT	Audi	80	TURISMO	2022	100000
Eliminar	Ver		9986PPM	Cadillac	Seville	TURISMO	1990	153000
Nuevo vehículo								

Datos del vehículo:

Matrícula	7548FTT
Marca	Audi
Modelo	80
Tipo	TURISMO
Año	2022



Registrar vehículo - Nuevo

MATRÍCULA

MARCA

MODELO

AÑO

KILÓMETROS

TURISMO

Nuevo

Registrar vehículo - Nuevo

MATRÍCULA
Rellene este campo.

MODELO
Rellene este campo.

AÑO

KILÓMETROS

TURISMO

Nuevo

Pendientes de pago

ID	Mecánico	Marca	Matrícula	Tipo	Reparación	Fecha Entrada	Fecha Salida	Horas Estimadas	Horas Reales	Importe	Pagar
33	2	Audi	7548FTT	TURISMO	Embrague	2023-06-06	2023-06-11	2	2	665.5000€	

Historico de reparaciones

ID	Mecánico	Marca	Matrícula	Tipo	Reparación	Fecha Entrada	Fecha Salida	Horas Estimadas	Horas Reales	Importe	Factura
31	2	Audi	7548FTT	TURISMO	Mantenimiento 40k	2023-06-06	2023-06-09	4	4	508.2000€	
27	2	Audi	7548FTT	TURISMO	Mantenimiento 80k	2023-06-05	2023-06-08	1	1	363.0000€	
17	2	Audi	7548FTT	TURISMO	Embrague	2023-03-06	2023-06-06	4	3	726.0000€	
13	2	Audi	7548FTT	TURISMO	Embrague	2022-12-07	2022-12-07	4	6	907.5000€	

Reparaciones

Talleres Fitipaldi Inicio Reparaciones Vehículos Datos personales Bienvenido **Laura Martinez Plá** Cerrar sesión [→]

Elija forma de pago

forma de pago	número de cuenta	pagar
Pago domiciliado	ES1234567899876543216544	Editar Pagar
PayPal		

[Volver](#)

Reparaciones

Talleres Fitipaldi Inicio Reparaciones Vehículos Datos personales Bienvenido **Laura Martinez Plá** Cerrar sesión [→]

Elija forma de pago

forma de pago	número de cuenta	pagar
Pago domiciliado	ES1234567899876543216544	Editar Pagar
PayPal		

Introduzca su Iban:

Iban [Actualizar](#)

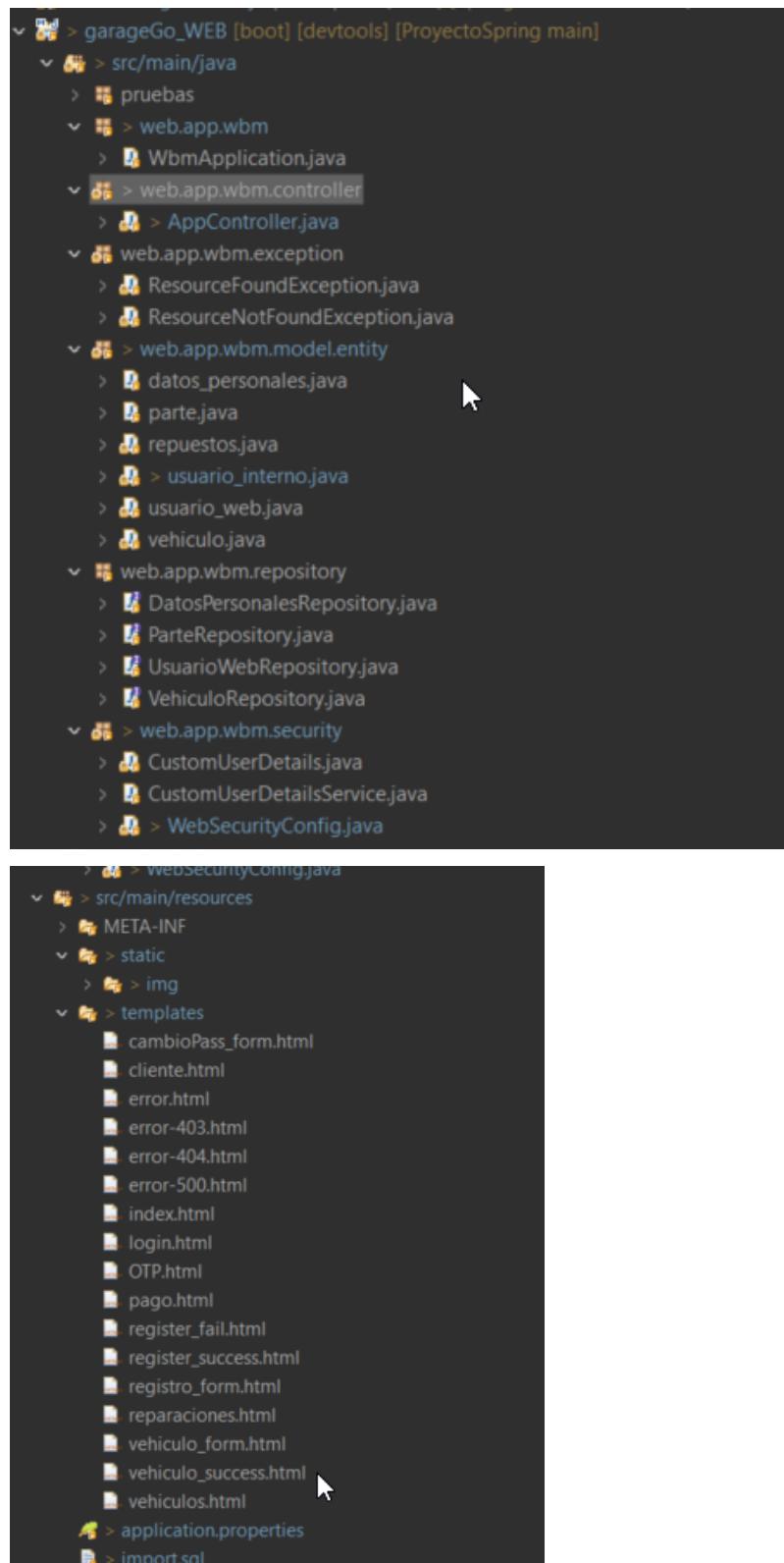
[Volver](#)

Confirm Log Out?

Are you sure you want to log out?

[Log Out](#)

Estructura del proyecto:



clase Controller:

```
1 package web.app.wbm.controller;
2
3 import java.io.ByteArrayInputStream;[]
47
48 @Controller
49 public class AppController implements ErrorController {
50
51     @Autowired
52     private UsuarioWebRepository userRepo;
53
54     @Autowired
55     private DatosPersonalesRepository datosRepo;
56
57     @Autowired
58     private VehiculoRepository vehiculoRepo;
59
60     @Autowired
61     private ParteRepository parteRepo;
62
63     public usuario_web getUsuario() {
64         Object principal = SecurityContextHolder.getContext().getAuthentication().getPrincipal();
65         System.out.println(principal.toString());
66         UserDetails userDetails = null;
67         if (principal instanceof UserDetails) {
68             userDetails = (UserDetails) principal;
69         }
70         String userName = userDetails.getUsername();
71         return userRepo.findByUsuario(userName);
72     }
73
74     public datos_personales getDatos() {
75         return datosRepo.findByDni(getUsuario().getDni());
76     }
77
78     @GetMapping(value = "/index")
79     public String viewHomePage(Model model) {
80         if (getUsuario() != null)
81             model.addAttribute("nombre", getDatos().getNombreCompleto());
82         return "index";
83     }
84
85     @GetMapping(value = "/registro")
86     public String showRegistrationForm(Model model) {
87         model.addAttribute("user", new usuario_web());
88         return "registro_form";
89     }
90 }
```

```

@PostMapping(value = "/process_register")
public String processRegister(usuario_web user) {

    if (datosRepo.existsById(user.getDni())) {
        BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
        String encodedPassword = passwordEncoder.encode(user.getPassword());
        user.setPassword(encodedPassword);
        datos_personales datos = datosRepo.findByDni(user.getDni());
        if (userRepository.existsById(user.getDni().toUpperCase()) || userRepository.findByUsuario(user.getEmail()) != null
            || datos == null || !datos.isCliente()) {
            return "register_fail";
        }
        userRepository.save(user);
        return "register_success";
    } else {
        return "register_fail";
    }
}

@GetMapping(value = "/cliente")
public String listUsers(Model model) {

    model.addAttribute("listUsers", getUsuario());
    model.addAttribute("listDatos", getDatos());
    model.addAttribute("nombre", getDatos().getNombreCompleto());

    return "cliente";
}

@PostMapping(value = "/cliente")
public String updateDatos(@RequestParam("dni") String dni, @RequestParam("telefono") String telefono,
    @RequestParam("direccion") String direccion) {
    datos_personales datos = datosRepo.findByDni(dni);
    datos.setTelefono(telefono);
    datos.setDireccion(direccion);
    datosRepo.save(datos);
    return "redirect:/cliente";
}

@GetMapping(value = "/cliente/passwd")
public String updatePass(Model model) {

    model.addAttribute("nombre", getDatos().getNombreCompleto());
    return "cambioPass_form";
}

@PostMapping(value = "/cliente/passwd")
public String updateDatos(@RequestParam("currentPasswd") String currentPasswd,
    @RequestParam("newPasswd") String newPasswd, @RequestParam("newPasswd2") String newPasswd2, Model model) {
    BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();

    if (!passwordEncoder.matches(currentPasswd, getUsuario().getPassword())) {
        model.addAttribute("error", "Contraseña incorrecta");
        return "cambioPass_form";
    }

    if (!newPasswd.equals(newPasswd2)) {
        model.addAttribute("error2", "Las contraseñas no coinciden");
        return "cambioPass_form";
    }

    usuario_web user = getUsuario();
    user.setPassword(passwordEncoder.encode(newPasswd));
    userRepository.save(user);

    return "redirect:/cliente";
}

@GetMapping(value = "/vehiculos")
public String listVehiculos(Model model) {
    List<vehiculo> listacoches = vehiculoRepo.findAllByDni(getUsuario().getDni());
    model.addAttribute("listacoches", listacoches);
    model.addAttribute("nombre", getDatos().getNombreCompleto());

    return "vehiculos";
}

```

```

@PostMapping(value = "/vehiculos")
public String updateDatos(@RequestParam("matricula") String matricula, @RequestParam("km") String km, Model model) {
    vehiculo vehiculo = vehiculoRepo.findByMatricula(matricula);
    if (vehiculo.getKm() == null) {
        vehiculo.setKm(0);
    }
    if (vehiculo.getKm() > Integer.parseInt(km)) {
        List<vehiculo> listacoches = vehiculoRepo.findAllByDni(getUsuario().getDni());
        model.addAttribute("listacoches", listacoches);
        model.addAttribute("nombre", getDatos().getNombreCompleto());
        model.addAttribute("error",
                "Error al actualizar los datos: Los kilómetros deben de ser superiores a los que ya tiene el vehículo");
        return "vehiculos";
    } else {
        vehiculo.setKm(Integer.parseInt(km));
        vehiculoRepo.save(vehiculo);
    }
    return "redirect:/vehiculos";
}

@GetMapping(value = "/vehiculos/eliminar/{matricula}")
public String eliminarViculo(@PathVariable(value = "matricula") String matricula) {
    vehiculo vehiculo = vehiculoRepo.findByMatricula(matricula);
    vehiculo.setAlta(false);
    vehiculoRepo.save(vehiculo);
    return "redirect:/vehiculos";
}

@GetMapping(value = "/vehiculos/editar/{matricula}")
public String editVehiculo(@PathVariable(value = "matricula") String matricula, Model model) {
    List<vehiculo> listacoches = vehiculoRepo.findAllByDni(getUsuario().getDni());
    model.addAttribute("listacoches", listacoches);
    model.addAttribute("nombre", getDatos().getNombreCompleto());

    vehiculo vehiculo = vehiculoRepo.findByMatricula(matricula);
    model.addAttribute("coche", vehiculo);
    return "/vehiculos";
}

@GetMapping(value = "/vehiculos/nuevo")
public String nuevoVehiculo(Model model) {
    model.addAttribute("vehiculo", new vehiculo());
    return "vehiculo_form";
}

@GetMapping(value = "/vehiculos/nuevo")
public String nuevoVehiculo(Model model) {
    model.addAttribute("vehiculo", new vehiculo());
    return "vehiculo_form";
}

@PostMapping("/vehiculos/nuevo")
public String updateDatos(vehiculo vehiculo, Model model) {
    if (vehiculoRepo.findByMatricula(vehiculo.getMatricula()) != null
            && vehiculoRepo.findByMatricula(vehiculo.getMatricula()).isAlta()) {
        model.addAttribute("error", "Error, La matrícula coincide con un vehículo que ya está dado de alta.");
        return "vehiculo_form";
    } else if (vehiculoRepo.findByMatriculaBaja(vehiculo.getMatricula()) != null) {
        if (vehiculo.getMarca()
                .equalsIgnoreCase(vehiculoRepo.findByMatriculaBaja(vehiculo.getMatricula()).getMarca())
                && vehiculo.getTipo()
                .equalsIgnoreCase(vehiculoRepo.findByMatriculaBaja(vehiculo.getMatricula()).getTipo())
                && vehiculo.getModelo()
                .equalsIgnoreCase(vehiculoRepo.findByMatriculaBaja(vehiculo.getMatricula()).getModelo())) {
            model.addAttribute("error2", "Vehículo ya existente. ¿Desea darse de alta como su nuevo propietario?");
        } else {
            model.addAttribute("error3",
                    "Datos erroneos. La matrícula proporcionada coincide con otro vehículo que no coincide con los datos introducidos.");
            return "vehiculo_form";
        }
    }
    return "vehiculo_form";
}
vehiculo.setMatricula(vehiculo.getMatricula().toUpperCase());
vehiculo.setMarca(vehiculo.getMarca().toUpperCase());
vehiculo.setTipo(vehiculo.getTipo());
vehiculo.setAlta(true);
vehiculo.setPropietario(getDatos());
vehiculoRepo.save(vehiculo);

return "vehiculo_success";
}

```

```

    @GetMapping(value = "/reparaciones")
    public String listaReparaciones(Model model, @RequestParam(defaultValue = "0") int pagina) {
        List<parte> listaparte = parteRepo.findByDniPendientes(getUsuario().getDni());
        if (listaparte.size() != 0)
            model.addAttribute("listaparte", listaparte);

        List<parte> listaparte2 = parteRepo.findByDniReparando(getUsuario().getDni());
        if (listaparte2.size() != 0)
            model.addAttribute("listaparte2", listaparte2);

        List<parte> listaparte3 = parteRepo.findByDniSinPagar(getUsuario().getDni());
        if (listaparte3.size() != 0)
            model.addAttribute("listaparte3", listaparte3);

        Page<parte> listaparte4 = parteRepo.findByDniFacturado(getUsuario().getDni(), PageRequest.of(pagina, 4));
        model.addAttribute("listaparte4", listaparte4.getContent());
        model.addAttribute("totalPaginas", listaparte4.getTotalPages());
        model.addAttribute("paginaActual", pagina);

        if (pagina == listaparte4.getTotalPages() - 1)
            model.addAttribute("fin", "Última página");
        if (pagina == 0)
            model.addAttribute("inicio", "Primera página");
        model.addAttribute("nombre", getDatos().getNombreCompleto());

        return "reparaciones";
    }

    @GetMapping(value = "/reparaciones/{id}")
    public String firmarPresupuesto(@PathVariable(value = "id") Integer id) {
        parte parte = parteRepo.findByIdParte(id);
        parte.setPresupuesto_firmado(true);
        parteRepo.save(parte);
        return "redirect:/reparaciones";
    }

    @GetMapping(value = "/reparaciones/pago/{id}")
    public String formapago(@PathVariable(value = "id") Integer id, Model model) {
        parte parte = parteRepo.findByIdParte(id);

        model.addAttribute("parte", parte);
        model.addAttribute("nombre", getDatos().getNombreCompleto());
        return "pago";
    }

```

```

    @PostMapping("/reparaciones/pago/confirma/{id}")
    public String pagar(@PathVariable(value = "id") Integer id, Model model) {
        datos_personales cliente = datosRepo.findByDni(getDatos().getDni());
        parte parte = parteRepo.findByIdParte(id);
        if(getDatos().getIban()==null) {
            model.addAttribute("cliente", cliente);
            model.addAttribute("nombre", getDatos().getNombreCompleto());
            model.addAttribute("parte", parteRepo.findByIdParte(id));
            model.addAttribute("error", "ERROR: Introduzca el iban antes de confirmar el pago domiciliado");
            return "pago";
        }
        parte.setFacturado(true);
        parteRepo.save(parte);
        return "redirect:/reparaciones";
    }

    @GetMapping(value = "/reparaciones/pago/editar/{id}")
    public String editiban(@PathVariable(value = "id") Integer id, Model model) {
        datos_personales cliente = datosRepo.findByDni(getDatos().getDni());
        model.addAttribute("cliente", cliente);
        model.addAttribute("nombre", getDatos().getNombreCompleto());
        model.addAttribute("parte", parteRepo.findByIdParte(id));
        model.addAttribute("editar", "editar");
        return "pago";
    }

    @PostMapping(value = "/reparaciones/pago/{id}")
    public String updateIban(@PathVariable(value = "id") Integer id, @RequestParam("iban") String iban, Model model) {
        datos_personales cliente = datosRepo.findByDni(getUsuario().getDni());
        iban=iban.toUpperCase();
        if (iban.length()==0)
            iban=null;
        cliente.setiban(iban);
        datosRepo.save(cliente);
        model.addAttribute("nombre", getDatos().getNombreCompleto());
        model.addAttribute("parte", parteRepo.findByIdParte(id));
        return "pago";
    }
}

```

```

@PostMapping("/upload/{matricula}")
public String uploadfile(@RequestPart("file") MultipartFile file, Model model,
    @PathVariable(value = "matricula") String matricula) {

    if (!file.isEmpty()) {
        Path directorioImagenes = Paths.get("src/main/resources/static/img/coches");
        String rutaAbsoluta = directorioImagenes.toFile().getAbsolutePath();

        try {
            byte[] bytesImg = file.getBytes();
            Path rutaCompleta = Paths.get(rutaAbsoluta + "/" + file.getOriginalFilename());
            Files.write(rutaCompleta, bytesImg);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    vehiculo vehiculo = vehiculoRepo.findByMatricula(matricula);
    vehiculo.setFoto(file.getOriginalFilename());
    vehiculoRepo.save(vehiculo);

    return "redirect:/vehiculos";
}

```

```

@PostMapping("/uploadFB/{matricula}")
public String uploadFileFB(@RequestPart("file") MultipartFile file, Model model,
    @PathVariable(value = "matricula") String matricula) throws IOException {

    long contentLength = file.getSize();
    ObjectMetadata metadata = new ObjectMetadata();
    metadata.setContentLength(contentLength);

    // Obtener vehículo asociado a la matrícula
    vehiculo vehiculo = vehiculoRepo.findByMatricula(matricula);

    // Obtener cliente asociado al vehículo
    datos_personales cliente = vehiculo.getPropietario();

    // Identificarse en firebase a través de la api de aws
    AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
        .withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration("s3.firebaseio.com", "us-east-1"))
        .build();

    // Crear carpeta en Firebase con el DNI del cliente si no existe
    String bucketName = "taller";
    String folderName = "clientes/" + cliente.getDni() + "/fotos/" + vehiculo.getMatricula();
    if (!s3Client.doesObjectExist(bucketName, folderName)) {
        s3Client.putObject(bucketName, folderName, new ByteArrayInputStream(new byte[0]), null);
    }

    // Guardar imagen en Firebase usando el DNI del cliente como prefijo del nombre
    // de archivo
    String fileName = vehiculo.getMatricula() + "_" + file.getOriginalFilename();
    PutObjectRequest putRequest = new PutObjectRequest(bucketName, folderName + "/" + fileName,
        file.getInputStream(), null).withMetadata(metadata);

    PutObjectResult result = s3Client.putObject(putRequest);

    // Actualizar nombre de archivo en el vehículo y guardar en la base de
    // datos(Guardo el cid en la base de datos para poder
    // acceder a ella directamente con la ruta)
    vehiculo.setFoto(result.getMetadata().getUserMetadata().get("cid"));
    vehiculoRepo.save(vehiculo);

    return "redirect:/vehiculos";
}

```

```

    @RequestMapping("/error")
    public ModelAndView handleError(HttpServletRequest response) {
        ModelAndView modelAndView = new ModelAndView();

        if (response.getStatus() == HttpStatus.NOT_FOUND.value()) {
            modelAndView.setViewName("error-404");
        } else if (response.getStatus() == HttpStatus.FORBIDDEN.value()) {
            modelAndView.setViewName("error-403");
        } else if (response.getStatus() == HttpStatus.INTERNAL_SERVER_ERROR.value()) {
            modelAndView.setViewName("error-500");
        } else {
            modelAndView.setViewName("error");
        }

        return modelAndView;
    }

    @Override
    public String getErrorPath() {
        return "/error";
    }
}

```

Vehiculos.html:

```

1  <!DOCTYPE html>
2  <html xmlns:th="http://www.thymeleaf.org">
3
4  <head>
5      <meta charset="UTF-8" />
6      <title th:text="Vehiculos">Vehiculos</title>
7      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
8      <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/sweetalert2@11.1.4/dist/sweetalert2.min.css">
9      <style>
10         .error {
11             color: #ff0000;
12             font-style: italic;
13             font-weight: bold;
14         }
15     </style>
16 </head>
17
18 <body>
19
20     <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
21         <div class="container-fluid">
22             <a class="navbar-brand" href="">
23                 
25                 <b>Talleres Fitipaldi</b>
26             </a>
27
28             <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"
29                 aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
30                 <span class="navbar-toggler-icon"></span>
31             </button>
32             <div class="collapse navbar-collapse" id="navbarNav">
33                 <ul class="navbar-nav">
34                     <li class="nav-item">
35                         <a class="nav-link " aria-current="page" th:href="@{/index}">Inicio </a>
37                     </li>
38                     <li class="nav-item">
39                         <a class="nav-link" th:href="@{/reparaciones}">Reparaciones </a>
41                     </li>
42                     <li class="nav-item">
43                         <a class="nav-link active" th:href="@{/vehiculos}">Vehículos </a>
45                     </li>
46                     <li class="nav-item">
47                         <a class="nav-link " th:href="@{/cliente}">Datos personales </a>
49                 </ul>
50             </div>
51         </div>
52     </nav>
53
54     <div class="container">
55         <div class="row justify-content-center">
56             <div class="col-12 col-md-8 col-lg-6">
57                 <div class="card border-0 shadow p-4 mb-4" style="background-color: #f9f9f9">
58                     <div class="card-body">
59                         <div class="mb-3">
60                             <label for="exampleText" class="form-label">Nombre del VehículoNúmero de ChasisNúmero de MotorAño de FabricaciónNúmero de PlacaNúmero de DocumentosNúmero de ReparacionesNúmero de MantenimientosNúmero de RevisionesNúmero de ServiciosNúmero de ReparacionesNúmero de Mantenimientos</label>
457                             <input type="
```

```

51         </li>
52     </ul>
53   </div>
54  <!-- IF(NO ESTAMOS LOGUEADOS) -->
55  <a sec:authorize="!isAuthenticated()" class="btn btn-outline-success" role="button" aria-disabled="true"
56    href="/Login"><b>Iniciar sesión</b>
57    <!--#49F3B3--> 
58  </a>
59
60  <a sec:authorize="isAuthenticated()" class="navbar-brand" style="color:#FFFFFF0">
61    Bienvenido <b>${[ ${nombre} ]}</b>
62  </a>
63  <a sec:authorize="isAuthenticated()" class="btn btn-outline-danger" role="button" aria-disabled="true" href="/logout"><b>Cerrar sesión</b>
64    <!--#F54D4D--> 
65  </a>
66
67
68
69  </div>
70 </nav>
71
72
73  <div class="container text-center alert alert-warning" role="warning">
74    <br />
75    <h1 class=" border border-success border-top-0 border-left-0 border-right-0"><b>Vehículos</b></h1>
76    <br />
77  <table class="table table-striped table-responsive-xl">
78    <thead class="bg-dark">
79      <tr style="color:#FFFF0">
80        <th>eliminar</th>
81        <th>ver</th>
82        <th>foto</th>
83        <th>matrícula</th>
84        <th>marca</th>
85        <th>modelo</th>
86        <th>tipo</th>
87        <th>anو</th>
88        <th>kilómetros</th>
89      </tr>
90    </thead>
91  <tbody>
92    <tr th:each="vehiculo: ${listacoches}">
93      <td>
94
95
96      <button class="btn btn-danger" th:href="@{/vehiculos/eliminar/} + ${vehiculo.matricula}">
97          th:text="''Eliminar'' onclick="confirmDelete(event)">
98      </button>
99      </td>
100     <td><a class="btn btn-primary btn-xs" th:href="@{/vehiculos/editar/} + ${vehiculo.matricula}">
101       th:text="!Ver'"></a></td>
102     <td></td>
103
104     <td th:text="${vehiculo.matricula}"></td>
105     <td th:text="${vehiculo.marca}"></td>
106     <td th:text="${vehiculo.modelo}"></td>
107     <td th:text="${vehiculo.tipo}"></td>
108     <td th:text="${vehiculo.anio}"></td>
109     <td th:text="${vehiculo.km}">0</td>
110
111   </tr>
112 </tbody>
113
114 </table>
115
116 <p><a th:href="@{/vehiculos/nuevo}" class="btn btn-success btn-xs">Nuevo vehículo</a></p>
117 <div th:if="${error}" th:text="${error}" class="container center error"></div>
118 <div th:if="${megas}" th:text="${megas}" class="container center error"></div>
119 </div>
120 <div th:if="${coche}" class="container text-center alert alert-info" role="info">
121
122
123
124  <h1 class=" border border-success border-top-0 border-left-0 border-right-0">
125    <br />
126    <b>Datos del vehículo:</b>
127    <br />
128  </h1>
129
130
131
132
133  <div class="container">
134    <div class="row">
135      <div class="col-md-6">
136        <form th:action="@{/vehiculos}" th:object="${coche}" method="post">
137          <br />
138
139          <div class="form-group row">
140            <label class="col-sm-3 col-form-label"><b>Matrícula</b></label>
141            <div class="col-sm-8">
142              <input type="text" th:field="*{matricula}" class="form-control-plaintext" readonly />
143
144

```

```

145          </div>
146      </div>
147      <div class="form-group row">
148          <label class="col-sm-3 col-form-label"><b>Marca</b></label>
149          <div class="col-sm-8">
150              <input type="text" th:field="*{marca}" class="form-control-plaintext" readonly />
151          </div>
152      </div>
153      <div class="form-group row">
154          <label class="col-sm-3 col-form-label"><b>Modelo</b></label>
155          <div class="col-sm-8">
156              <input type="text" th:field="*{modelo}" class="form-control-plaintext" readonly />
157          </div>
158      </div>
159      <div class="form-group row">
160          <label class="col-sm-3 col-form-label"><b>Tipo</b></label>
161          <div class="col-sm-8">
162              <input type="text" th:field="*{tipo}" class="form-control-plaintext" readonly />
163          </div>
164      </div>
165      <div class="form-group row">
166          <label class="col-sm-3 col-form-label"><b>Año</b></label>
167          <div class="col-sm-8">
168              <input type="text" th:field="*{anio}" class="form-control-plaintext" readonly />
169          </div>
170      </div>
171      <div class="form-group row">
172          <label class="col-sm-3 col-form-label"><b>Kilometros</b></label>
173          <div class="col-sm-8">
174              <input type="text" class="form-control" th:field="*{km}" minlength="0" maxlength="6"
175                  title="Introduzca los kilómetros actuales de su vehículo"
176                  pattern="^([0-9]{1-9}|[1-9]{1}[0-9]{1,5}|[1-9]{1}[0-9]{0,4})$" />
177          </div>
178      </div>
179
180      <div class="form-group row">
181          <div class="col text-right">
182              <input type="submit" value="Actualizar kilometros" class="btn btn-primary" />
183          </div>
184          <div class="col text-right">
185
186
187
188
189          </div>
190      </div>
191  </form>
192
193
194
195
196
197      <div class="col-md-6">
198          
200
201      <form method="post" th:action="@{/uploadFB} + *{matricula}" enctype="multipart/form-data"
202          id="fileUploadForm" class="container">
203          <div class="form-group">
204              <label for="formFile" class="form-label">
205                  Si lo desea cambiar la foto de su vehículo aqui:
206                  <input type="file" name="file" class="form-control" id="imagenUpload" />
207              </label>
208          </div>
209          <div style="margin-bottom: 10px">
210              <button type="submit" class="btn btn-success">Cambiar</button>
211          </div>
212      </form>
213
214  </div>
215 </div>
216
217 </div>
218 </div>
219 <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
220 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>
221 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
222 <script>
223     function confirmDelete(event) {
224         event.preventDefault();
225         Swal.fire({
226             title: '¿Estás seguro que quieres eliminar?',
227             icon: 'warning',
228             showCancelButton: true,
229             confirmButtonColor: 'mediumvioletred',
230             cancelButtonColor: 'grey',
231             confirmButtonText: 'Sí, eliminar',
232             cancelButtonText: 'Cancelar'
233         }).then(result) => {
234             if (result.isConfirmed) {
235                 window.location.href = event.target.getAttribute('href');
236             }
237         })
238     }
239 </script>
240 <script>
```

```
241     const fileInput = document.getElementById('imagenUpload');
242     const fileUploadForm = document.getElementById('fileUploadForm');
243
244     fileUploadForm.addEventListener('submit', (event) => {
245         event.preventDefault();
246
247         const file = fileInput.files[0];
248
249         if (file && file.size <= 1048576) {
250             fileUploadForm.submit();
251         } else {
252             Swal.fire({
253                 title: 'La imagen seleccionada supera el tamaño máximo permitido (1MB). Por favor seleccione otra imagen',
254                 icon: 'error',
255                 confirmButtonColor: 'mediumvioletred',
256                 confirmButtonText: 'Aceptar'
257             }).then((result) => {
258                 if (result.isConfirmed) {
259                     // El usuario hizo clic en el botón "Aceptar"
260                 }
261             })
262         }
263     });
264
265 </script>
266 <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11.1.4/dist/sweetalert2.min.js"></script>
267     <script src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.0.0/js/bootstrap.bundle.min.js"></script>
268
269
270 </body>
271
272 </html>
<
```