UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Aplicación web dinámica para la administración y gestión de talleres automovilísticos

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Fuentes Tornero, José

Tutor/a: Andrés Martínez, David de

CURSO ACADÉMICO: 2022/2023

Resumen

Este trabajo tiene como objetivo desarrollar una aplicación Web que sirva de herramienta a los administradores de talleres automovilísticos. Podrán gestionar reparaciones, clientes, facturas, etc.

Actualmente la mayoría de las aplicaciones dirigidas a la administración y gestión suelen ser de pago. Esto causa que, en pequeñas empresas, no se puedan permitir contratar estos servicios dado que supone un coste adicional.

Este proyecto nace de la necesidad de desarrollar una herramienta para la administración de este tipo de empresas. Además, ofrece otras funcionalidades a los clientes como realizar reservas o pagar facturas.

Resum

Aquest treball té com a objectiu desenvolupar una aplicació web que serveixi d'eina als administradors de tallers automobilístics. Podran gestionar reparacions, clients, factures, etc.

Actualment la majoria de les aplicacions dirigides a l'administració i la gestió solen ser de pagament. Això fa que en petites empreses no els ho puguin permetre perquè suposa un cost addicional.

Aquest projecte neix de la necessitat de desenvolupar una eina específica per a ladministració daquest tipus dempreses. A més ofereix altres funcionalitats als clients com fer reserves o pagar factures.

Abstract

The objective of this work is to develop a Web application that serves as a tool for automobile workshop administrators. They will be able to manage repairs, customers, invoices, etc.

Currently, all applications focused on administration and management are usually paid and for small businesses it may be an additional cost that they cannot afford.

This project arises from the need to create a specific tool for the administration of this type of company, also offering other functionalities to clients such as making reservations or paying invoices.

Índice de contenido

Contenido

1 Introducción

El primer capítulo de la memoria tiene como objetivo mostrar la situación actual de las herramientas para la administración de talleres automovilísticos, la motivación que ha provocado el desarrollo del proyecto, lo objetivos y el impacto esperado que se pretende conseguir una vez haya finalizado el desarrollo del proyecto.

Hoy en día, es importante disponer de una aplicación que ofrezca información de tu empresa o del servicio que vas a proporcionar. Además, suele ser de utilidad proporcionar un sistema de gestión tanto de los clientes como de los productos u objetos que van a ser tratados.

El principal problema que podemos observar en este tipo de herramientas es que, por lo general, se comercializan mediante un modelo de pago por uso. Hoy en día para pequeñas empresas es complicado dar el paso a informatizarse. Esto es debido a que no tienen los recursos necesarios para llevar esta tarea acabo. “Según el informe sobre el Estado de la Digitalización de las Pymes Españolas elaborado por el Banco Europeo de Inversiones (BEI), el 75% de las empresas Españolas apenas están digitalizadas y tan solo el 34% de ellas tiene previsto invertir en tecnología [1]”.

En resumen, esto hecho impide que estas empresas puedan llegar a un público más amplio y tengan menor capacidad para competir en un mercado, cada vez, más digitalizado.

1.1 Motivación

La principal motivación de realizar este trabajo surge de la necesidad de tener un sistema eficiente y completo para la gestión de reparaciones, en los talleres mecánicos.

Actualmente, muchos talleres en todo el país continúan utilizando herramientas antiguas como agendas y facturas en papel que, en muchas ocasiones, conduce a una gestión ineficiente. Hoy en día la digitalización está llegando a todos los sectores, por ello es importante el desarrollo de diferentes herramientas que ayuden en el proceso de informatizar este sector.

Por otro lado, también se quiere desarrollar un aplicación de este tipo dado que se tiene un interés personal, puesto que un familiar cercano dirige un taller y requiere de una aplicación con estas características.

1.2 Objetivos

En este apartado vamos a detallar los objetivos principales de este proyecto.

El objetivo principal es la creación de una aplicación web dinámica. Esta debe facilitar la gestión de las reparaciones que se realicen en talleres mecánicos, comenzado desde una reserva hasta el pago de la factura generada. Para lograr este objetivo se han definido las siguientes tareas:

1. Análisis de requisitos
2. Diseño de la arquitectura que tendrá la aplicación.
3. Desarrollo de la aplicación.
4. Implementación de funcionalidades dedicadas a la parte de administración como la gestión de usuarios, de vehículos , de reservas, etc.
5. Pruebas de validación y corrección de errores en la aplicación.
6. Desarrollo de una documentación completa de la aplicación y de su funcionamiento.

Por tanto, la aplicación estará compuesta por 2 vistas:

Vista del cliente: Mediante menús podrá acceder a los servicios que proporciona la aplicación.

Vista administrador: Podrá gestionar los servicios proporcionados.

Objetivos adicionales: Desarrollar una aplicación que pueda ser utilizada en cualquier dispositivo

1.3 Planificación temporal

Para poder cumplir los objetivos que se expusieron en el apartado anterior, es necesario realizar una planificación temporal para disponer de una idea inicial de tiempo necesario para finalizar el desarrollo del proyecto.

La fase de desarrollo del proyecto durará aproximadamente 3 meses. El desarrollo empieza 27/03/2023 y finaliza el 27/06/2023. En estos tres meses los podemos dividir en las siguientes 3 etapas:

1. Mes01: Análisis de requisitos y diseño de la arquitectura.
2. Mes02: Desarrollo de la aplicación web, tanto el frontend como el backend.
3. Mes03: Pruebas, corrección de errores y desarrollo de la documentación completa de la aplicación y sus funcionalidades.

1.4 Impacto esperado

Se espera que la aplicación final tenga un impacto positivo en el sector de los talleres automovilísticos, facilitando y optimizando la gestión de todos los recursos que se necesitan en este tipo de negocios, además de una mejor atención al cliente puesto que la aplicación también contempla a este tipo de usuarios.

Además, se espera que impulse la digitalización del sector. La aplicación podrá ser utilizada como ejemplo a seguir.

1.5 Metodología

"El mobile first design responde a un ritmo de vida concreto, en el que cada vez se utiliza más el teléfono móvil. Por eso se busca siempre un diseño centrado en el usuario [2]".

Para mejorar el resultado que obtendremos al finalizar el proyecto, hemo utilizado técnicas avanzadas de desarrollo web. Estás consisten en dar prioridad a la creación de una versión para dispositivos móviles. Luego, partiendo de esta versión, se amplía para resoluciones más grandes como computadores o portátiles, garantizando de esta manera, que los usuarios puedan utilizar la aplicación desde cualquier dispositivo.

1.6 Estructura

En este último apartado se realizará una breve descripción de cada uno de los capítulos que componen la memoria que se ha desarrollado. La memoria constará de nueve capítulos que ayudarán a lector a comprender los pasos que se han seguido para el desarrollo de la aplicación.

En el primer apartado de la memoria, se realiza la introducción del proyecto, la motivación, los objetivos, la planificación temporal y el impacto esperado.

El "Estado del Arte", el segundo bloque de la memoria introduce el análisis de la aplicación que se desea desarrollar, haciendo una comparación con aplicaciones similares que haya en el mercado con el objetivo de innovar en el sector donde se planea lanzar la aplicación.

En el tercer capítulo se describe detalladamente las diferentes herramientas para el desarrollo de la aplicación web. Se puede encontrar un resumen de cada una de ellas y con qué objetivo han sido utilizadas.

En el cuarto capitulo se detallará la información más importante del diseño de la aplicación, su arquitectura, sus interfaces y la estructura de la base de datos. Este capítulo estará dividido en 4 apartados previamente nombrados.

En el quinto capítulo se detallará el proceso que se ha seguido en el desarrollo de la aplicación. Es decir, se detallarán los datos más relevantes del desarrollo del servidor (Backend), desde la definición de la base de datos, hasta las peticiones AJAX(Asynchronous JavaScript and XML), que realizará la interfaz de usuario(FrontEnd).

En el sexto capítulo “pruebas”, se detallarán las pruebas realizadas tanto en el servidor de la aplicación como en la interfaz de usuario. Además, se mostrarán los resultados de las pruebas de validación que se han realizado a diferentes usuarios y con diferentes vistas de la aplicación.

En el séptimo capítulo “Resultados de la aplicación”, mostraremos los resultados de desarrollo mediante una serie de capturas de pantalla.

En el octavo capítulo “conclusiones”, se evaluará el cumplimiento de los objetivos propuestos y se discutirán posibles mejoras y actualizaciones del proyecto.

Para terminar, en el último capítulo “Anexo”, se mostrarán los enlaces a la información relevante que se ha utilizado para el desarrollo del proyecto en su totalidad.

2 Estado del Arte

Actualmente se recomienda a las empresas automatizar los procesos de gestión y administración. Ya que hace más sencillo para el administrador realizar estos procesos y, además, se complica la perdida de datos ya que estos suelen estar guardados en una base de datos.

"Automatizar los procesos conlleva, de forma intrínseca, la digitalización y centralización de información, además, la disminución de la participación de los recursos humanos. Por esto incrementa la transparencia y confiabilidad de los datos y ayuda a disminuir o, incluso, eliminar los errores [3]."

Hoy en día, podemos encontrar diferentes herramientas, tanto gratuitas como de pago, centradas únicamente en la parte de gestión, que se encargan de la generación de facturas, etc. "Software de Gestión y contabilidad Online [4]."

Sin embargo, también podemos encontrar aplicaciones similares a la que hemos desarrollado que combina tanto la gestión como el ofrecer diferentes servicios a los clientes que se registren.

En este capítulo de la memoria se realizará una comparación entre aplicaciones similares a la que hemos desarrollado, con el objetivo de identificar las características más importantes de cada una de las aplicaciones, así como las deficiencias de las mimas. Con ello pretendemos mejorar las funcionalidades que la aplicación desarrollada puede ofrecer.

2.1 Descripción de las aplicaciones similares investigadas

A) MechanicAdvisor

Es una aplicación que ofrece la funcionalidad a los usuarios de buscar talleres cercanos a la posición en la que se encuentran. Además, permite solicitar presupuestos y realizar citas en línea.

Imagen01: Logo MechanicAdvisor

De MechanicAdvisor [5], cuyo logo se muestra en la imagen01, podemos destacar aspectos positivos como:

Características interesantes:

1. Ofrece a los usuarios una herramienta de para analizar los problemas que pueden surgir en sus vehículos.
2. Cuenta con una amplia red de talleres registrados en su red con opiniones de otros usuarios sobre estos, dando información adicional al usuario para seleccionar de una manera apropiada el taller en el cual quiere dejar su vehículo para que sea reparado.

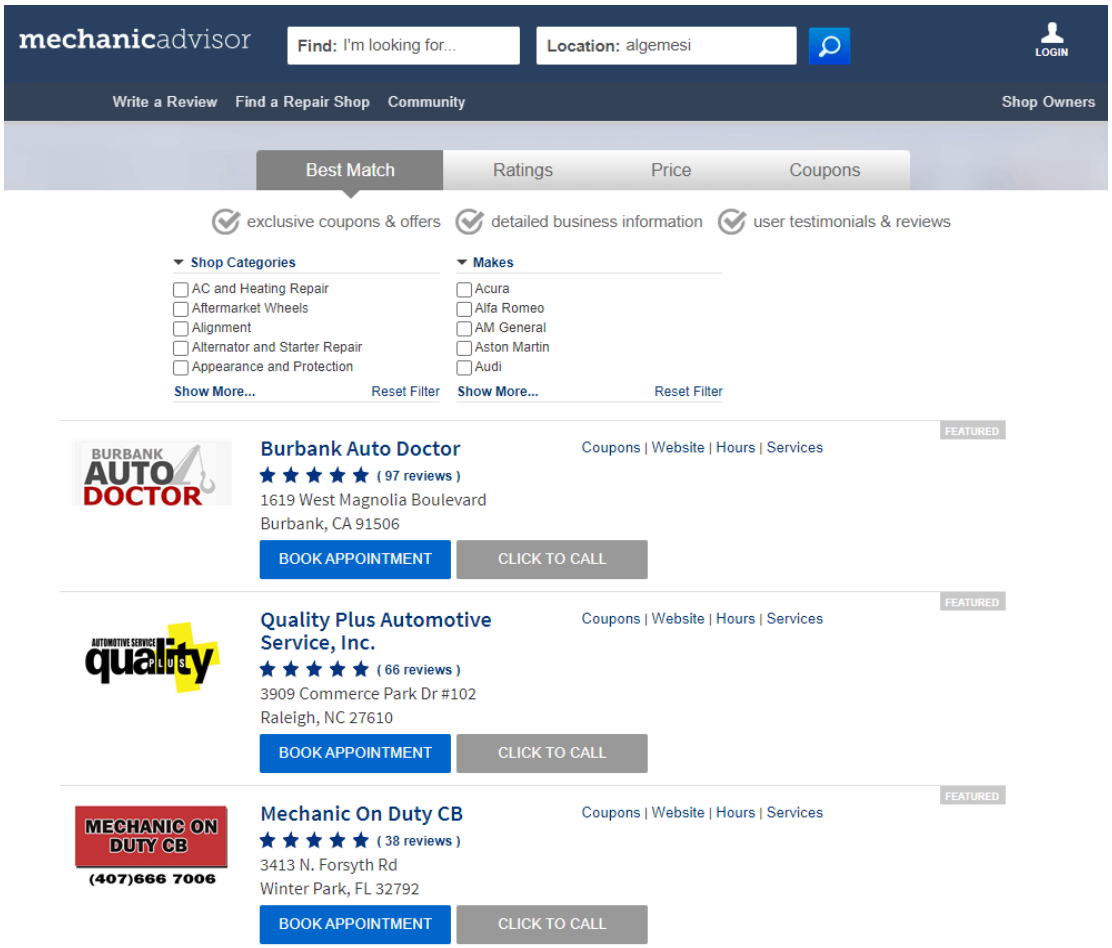


Imagen02: Búsqueda de talleres Mechanic Advisor

En la imagen02 podemos ver la interfaz de búsqueda de talleres en la aplicación web de MechanicAdvisor. Aspectos negativos a destacar:

Aspectos negativos:

1. Interfaz engorrosa y difícil de utilizar.
2. Solo disponible en un idioma.

3. No dispone de una aplicación móvil en Google Play.
4. Poca información sobre las actualizaciones realizadas.

B) YourMechanic

Esta aplicación ofrece reparaciones a domicilio donde los usuarios pueden reservar uno de los mecánicos de los que dispone la aplicación.

Los mecánicos están certificados y ofrecen precios fijos, lo que significa que los usuarios sabrán cuánto costará el trabajo antes de reservarlo. Para ello la aplicación cuenta con un apartado donde introduciendo una serie de datos del vehículo, puede generar un presupuesto.

Imagen03: Logo YourMechanic

De YourMechanic [6], cuyo logo se muestra en la imagen03, podemos destacar aspectos positivos como:

Características interesantes:

1. Ofrece la posibilidad de reparar el vehículo sin desplazarlo a ningún lugar en particular.
2. Los usuarios reciben actualizaciones constantes sobre el progreso de la reparación de su vehículo.
3. Interfaz fácil y sencilla de utilizar.
4. La velocidad de carga de la web para cualquier tipo de servicio proporcionado es muy buena.
5. Dispone de una aplicación para móviles en Google Play con una valoración de 4.5 estrellas.
6.

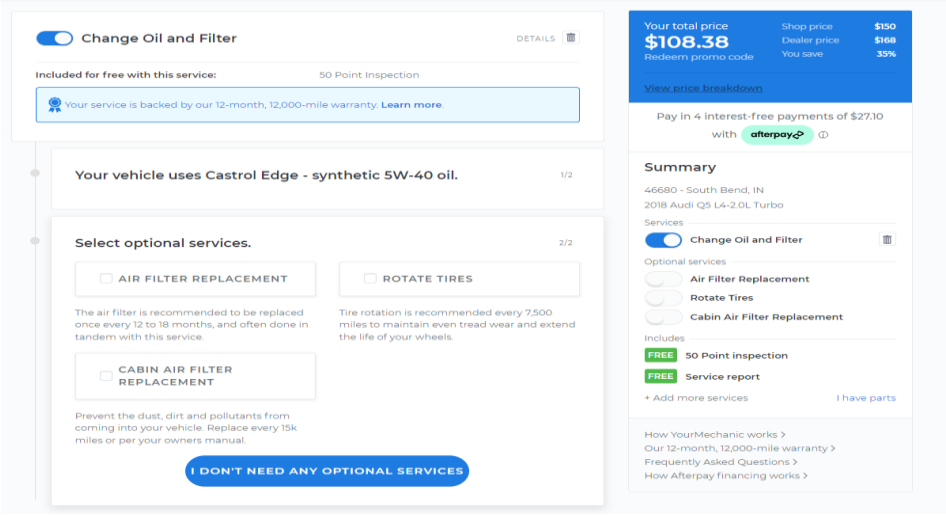


Imagen04: Uso de la aplicación web YourMechanic

Por otro lado, en la imagen04 vemos un ejemplo del uso de la aplicación web yourMechanic. En esta imagen como podemos observar estamos contratando un servicio de cambio de aceite o de filtro (Change oil and filter) y se nos está generando un presupuesto aproximado.

Aspectos negativos:

1. Solo disponible en un idioma.
2. Actualizaciones poco recientes, la última se realizó 15/03/2022.

C) AutoRepairBill

Esta aplicación es una herramienta de gestión donde los administradores pueden realizar facturas, gestionar reparaciones realizadas, pagos realizados, etc.

Imagen05: Logo AutoRepairBill

De AutoRepairBill [7], cuyo logo se muestra en la imagen05, podemos destacar aspectos positivos como:

Características interesantes:

1. Los usuarios reciben actualizaciones constantes sobre el progreso de la reparación de su vehículo.
2. El administrador puede gestionar productos, reparaciones, facturas etc.
3. Interfaz fácil y sencilla de utilizar.
4. La velocidad de carga de la web para cualquier tipo de servicio proporcionado es muy buena.

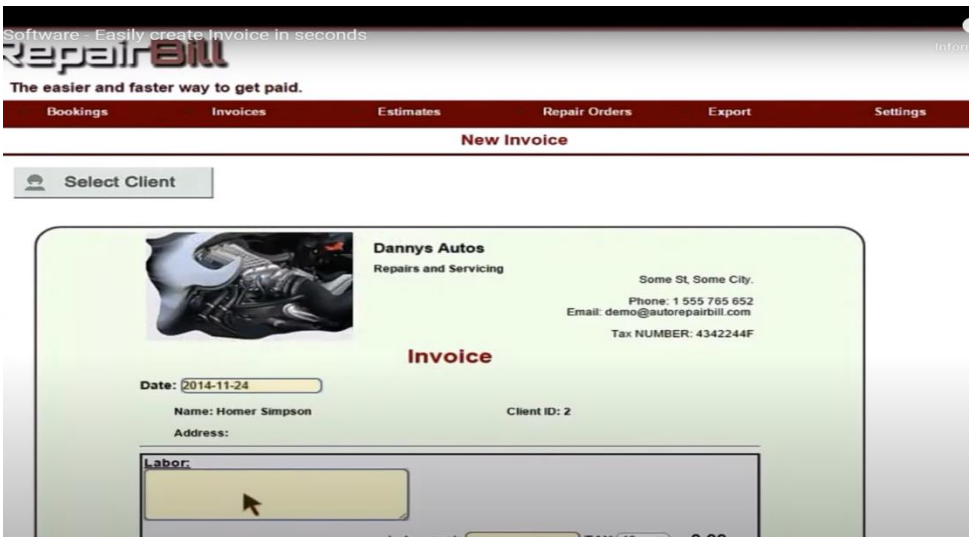


Imagen06: Creación factura

En la imagen06 podemos ver el ejemplo de cómo esta aplicación gestiona las facturas. Concretamente se está creando una factura para la reparación de un auto.

Aspectos negativos:

1. Solo disponible en un idioma.
2. No dispone de una aplicación para dispositivos móviles en Google Play.
3. Para utilizarla se debe disponer de una suscripción (modelo pago por uso).

D) RepairPal

Esta aplicación es bastante similar a la primera que analizamos, ofreciendo herramientas a los clientes para que analicen sus vehículos evitando futuras reparaciones. Además, ofrece estimaciones sobre el costo de la reparación basándose en la marca y el modelo del vehículo que registra el usuario.

Imagen07: Logo Repair Pal

De AutoRepairPal [8], cuyo logo se muestra en la imagen07, podemos destacar aspectos positivos como:

Características interesantes:

1. Ofrece herramientas a los usuarios para que puedan detectar posibles fallos sin la necesidad de acudir a un taller o especialista.
2. Cuenta con una amplia red de talleres registrados en su red con opiniones de otros usuarios sobre estos, dando información adicional al usuario para seleccionar un taller.
3. Proporciona una serie de tutoriales para ayudar al cliente a mantener su vehículo.

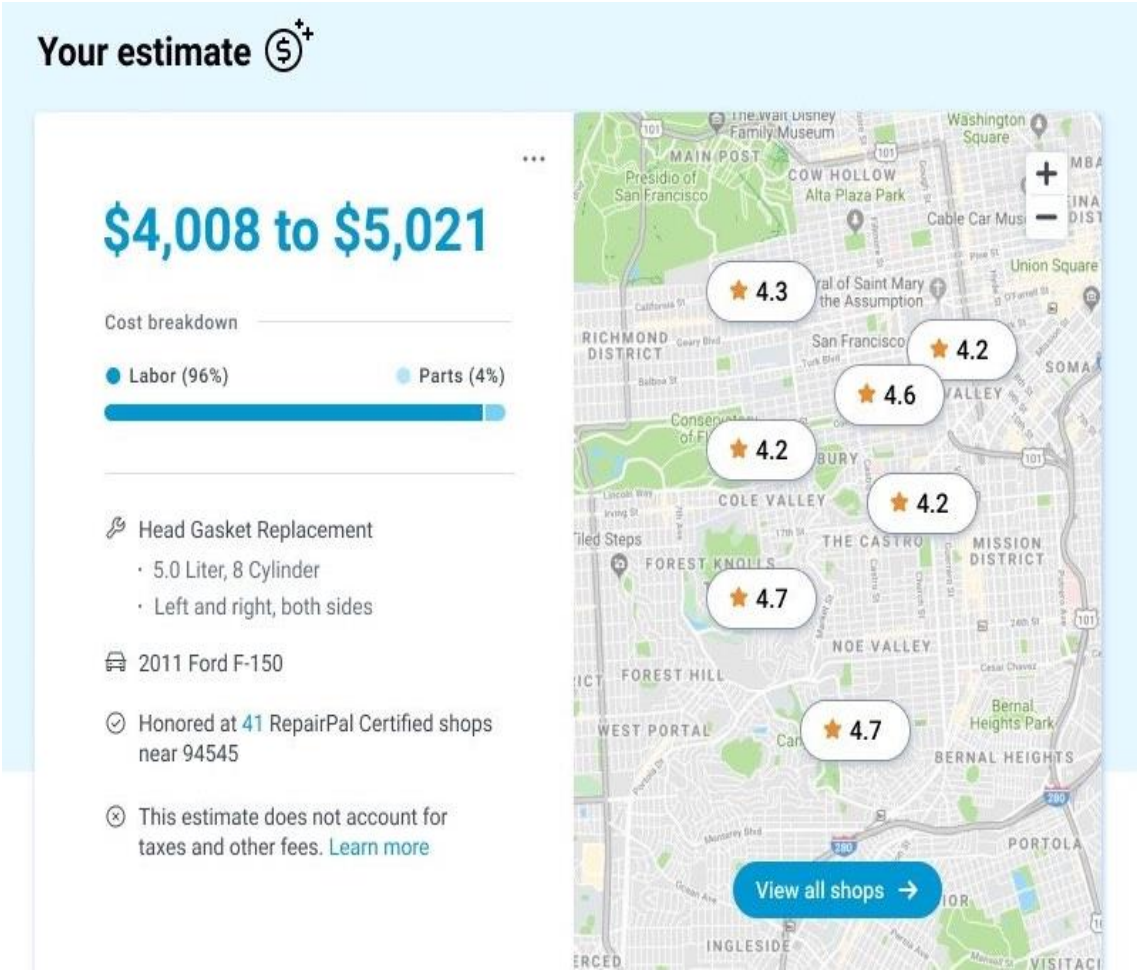


Imagen08: Búsqueda de talleres cercanos

En la imagen 08 podemos ver una serie de talleres con una valoración realizada por los usuarios que están registrados en la aplicación.

Aspectos negativos:

1. Solo disponible en un idioma.
2. No dispone de una aplicación para dispositivos móviles en Google Play.
3. La interfaz es poco intuitiva y complicada de usar.
4. 3.8/5 en valoraciones de Facebook.

2.2 Comparación de las aplicaciones

A continuación, la *tabla01* muestra la comparación entre las características de la aplicación que se pretende desarrollar y las descritas en el apartado anterior.

2.2.1 Comparación de características de las aplicaciones analizadas

Tabla01: Comparación de características

Características	La aplicación desarrollada	Mechanic Advisor	RepairPal	YourMechanic	Auto RepairBill
Registro de usuario	✓	✓	✓	✓	✓
Registro de vehículo	✓	✓	✓	✓	✓
Generación de facturas	✓	✓	✓	✓	✓
Generación de presupuestos	✓	✓	✓	✓	✓
Reserva de citas	✓	✓	✓	✓	✓
Notificaciones	✓	✓	✓	✓	✓
Calendario de citas	✓	✓	✓	✓	✓
Anuncios de vehículos de ocasión	✓	✗	✗	✗	✗
Evaluación de talleres	✗	✓	✓	✓	✗
Comentarios de usuarios	✗	✓	✗	✓	✓
Soporte técnico	✓	✓	✓	✓	✓
Precio	Gratis(Todos)	Gratis(Solo usuario)	Gratis(Solo usuario)	Variable	Variable
Múltiples plataformas	✓	✓	✓	✓	✓
Variedad de servicios ofrecidos	Media	Media	Media	Amplia	Amplia
Facilidad de uso	Alta	Media	Alta	Media	Media

Hemos detallado las características más importantes que se pueden comparar en las diferentes aplicaciones que hemos analizado como podemos observar en la tabla01: comparación de características.

2.3 Conclusiones del análisis

Como podemos observar en las descripciones de las aplicaciones anteriores, la aplicación desarrollada combina características de las aplicaciones mencionadas, como, por ejemplo: Permitir a los usuarios solicitar presupuestos y hacer citas en línea, proporcionar información sobre el estado de la reparación del vehículo, generar una factura del coste de la reparación, etc.

Sin embargo, nuestra aplicación no cuenta con alguna de las funcionalidades de las aplicaciones mencionadas como la posibilidad de seleccionar un taller entre los talleres registrados o evaluar la calidad del taller como sí incluyen las demás aplicaciones. Esto es debido a que la aplicación que se pretende desarrollar únicamente contará con un taller, el taller al que va dedicado la aplicación y por tanto no será necesario incluir esta funcionalidad en la aplicación.

Por otro lado, tampoco se incluirá opiniones o valoraciones de los clientes en la versión que se pretende desarrollar, sin embargo, en futuras actualizaciones se planteará al cliente si quiere incluir esta funcionalidad en la aplicación dado que será él quien tome la decisión final.

Por último, es importante destacar de nuestra aplicación frente a las demás, la incorporación de vehículos de ocasión, funcionalidad de la que no disponen las demás aplicaciones analizadas.

En conclusión, la aplicación desarrollada tiene muchas características similares a las aplicaciones que hemos comparado. Mientras que algunas aplicaciones ofrecen características y funcionalidades adicionales como evaluaciones de talleres y precios variable como hemos comentado anteriormente, la aplicación que hemos desarrollado destaca por su interfaz accesible para todo tipo de usuario, su facilidad de uso y por ser gratuita permitiendo a talleres pequeños informatizar su negocio y competir en un mercado cada vez más informatizado.

3 Tecnología utilizada

Para el proceso de desarrollo de la aplicación se ha hecho uso de diferentes herramientas y entornos de desarrollo(Framework) con el objetivo de facilitar dicho proceso. A continuación, se explican detalladamente que herramientas se han utilizado:

3.1 Angular

Angular [9] es un framework de JavaScript de código abierto utilizado comúnmente para la creación de aplicaciones. Fue creado por Google y utiliza (TypeScript" muy parecido a "JavaScript).

Imagen09: Logo Angular

Angular, culo podemos ver en la imagen09, está basado en componentes para manipular el Modelo de Objetos del Documento (del inglés Document Object Model - DOM). Estos componentes son nodos de un árbol que se van utilizando para dar como resultado un documento HTML.

Es una herramienta sólida y bien estructurada para el desarrollo de aplicaciones, que tiene actualizaciones constantes, por ello he decidido utilizarla para el desarrollo de este proyecto.

3.2 Laravel

Laravel [10] es un framework de PHP que es utilizado para el desarrollo de aplicaciones. Utiliza una arquitectura basada en patrones de diseño facilitando el desarrollo de aplicaciones.

Imagen10: Logo Laravel

La característica más importante de este framework es su arquitectura basada en el modelo-vista-controlador.

En resumidas cuentas, Laravel es un framework de PHP que facilita el desarrollo de las aplicaciones web, haciendo la programación fácil y agradable para el programador que lo utiliza.

3.3 Balsamiq Wireframes

Es una herramienta de prototipado rápido y diseño de interfaces de usuario que permite a los diseñadores crear y compartir bocetos de las aplicaciones.

Imagen11: Logo Balsamiq Wireframes

Balsamiq Wireframes [12], cuyo logo se muestra en la imagen11, es conocido por su interfaz simple y fácil de usar. Además, ofrece una alta variedad de elementos de diseño que vienen por defecto en su instalación, como botones, iconos, campos de entrada, etc.

3.4 Visual Studio Code

Visual Studio Code es un editor de código abierto desarrollado por Microsoft [16].

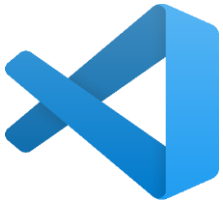


Imagen12: Logo Visual Studio code

Visual Studio Code [13], cuyo logo podemos ver en la imagen 12, es conocido por permitir a sus usuarios adaptar su entorno de trabajo según sus necesidades. Ofrece multitud de características como la finalización de código inteligente, la depuración integrada, el control de versiones, la integración de herramientas externas, etc. Además de ser compatible con diferentes lenguajes de programación y sistemas operativos, convirtiéndolo en una gran herramienta de desarrollo.

3.5 Node Package Manager [NPM]

NPM [20], cuyo logo podemos ver en la imagen 18, es una herramienta que permite instalar, actualizar y administrar las dependencias de los proyectos fácilmente.

Imagen18: NPM

3.6 Lenguaje TypeScript

TypeScript es un lenguaje de programación libre y de código abierto desarrollado por Microsoft [16].

Imagen15: TypeScript

TypeScript [17], cuyo logo podemos ver en la imagen 15, se puede decir que es una extensión de JavaScript añadiendo esencialmente tipos estáticos y objetos basados en clases.

3.7 Git

Git [14] es un *software* de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.

Imagen13: Logo Git

Git, cuto logo podemos ver en la imagen 13, permite a sus usuarios trabajar en sus proyectos de manera independiente y fusionar sus cambios de manera sencilla.

3.8 Gestor de BD MySQL

MySQL [15] es un sistema de gestión de bases de datos relacionales. Trabaja con múltiples tablas que se interconectan entre sí para almacenar la información. Posteriormente mediante consultas SQL se puede acceder a esta información almacenada.

Imagen14: Logo MySQL

La principal característica y uno de los motivos por el que es tan ampliamente utilizado, es que se puede utilizar prácticamente en todas las plataformas.

MySQL, cuyo logo podemos ver en la imagen14, Es conocido por su velocidad y seguridad. Por ello es muy utilizado por multitud de empresas y organizaciones para almacenar y procesar grandes cantidades de datos.

3.9 Lenguaje PHP

PHP [18], cuyo logo podemos ver en la imagen 16, es un lenguaje utilizado para programar el comportamiento del servidor al recibir una petición.

Imagen16: PHP

Cuando recibe una petición por lo general realizará una consulta a una base de datos, y mediante estos datos, generará un documento HTML que enviará como respuesta al cliente que realizó la petición.

3.10 Lenguaje SQL

SQL [19], cuyo logo podemos ver en la imagen 17, es un lenguaje centrado en la gestión de bases de datos relacionales. Se utiliza para crear, modificar y consultar el contenido almacenado en las bases de datos.

Imagen17: SQL

3.11 MAMP

MAMP [21], cuyo logo podemos ver en la imagen 19, es una herramienta que permite desarrollar una web de manera local. De esta manera se puede probar todo el código antes de subir el resultado a un servidor real. Además, se pueden configurar diferentes versiones de PHP, SQL, etc. Característica muy importante en este tipo de herramientas.

Imagen19: MAMP

3.12 POSTMAN

Postman [22], cuyo logo podemos ver en la imagen 20, en sus inicios nace como una extensión de que podría ser utilizada en el navegador para realizar peticiones de una manera sencilla, fácil y sin muchas complicaciones, con el objetivo de testear APIs de tipo REST propias o de terceros.

Imagen20: Postman

Con el tiempo se convirtió en una aplicación de escritorio mucho más cómoda de utilizar que en el propio navegador, pero con la misma funcionalidad, realizar peticiones para testear APIs.

4 Diseño de la aplicación

En este apartado de la memoria y una vez analizado el problema que pretendemos resolver, se procede a diseñar una solución eficiente que posteriormente implementaremos. En esta sección se detallan los diferentes elementos que conforman la aplicación diseñada.

4.1 Arquitectura del sistema

La solución que hemos propuesto se basa en la arquitectura típica en las aplicaciones web, la arquitectura cliente/servidor.

Esta arquitectura se divide en dos componentes principales: el cliente y el servidor.

1. **Cliente:** Realiza peticiones al servidor y espera una respuesta.
2. **Servidor:** Es el encargado de atender las peticiones mencionadas realizando una serie de funciones y devolviendo al cliente el resultado.

En la siguiente imagen se muestra de manera grafica lo que se ha explicado:

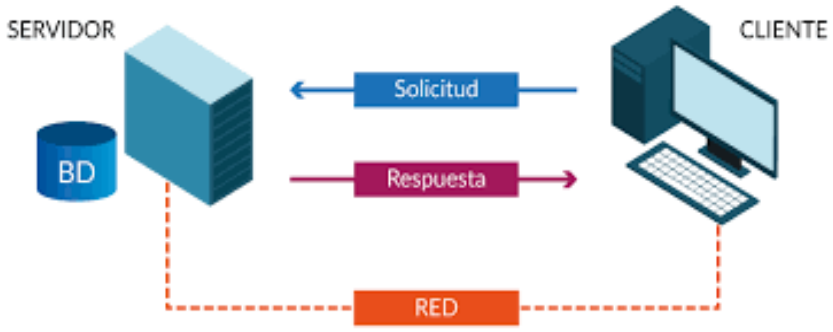


Imagen21: Arquitectura Cliente/Servidor

La característica principal de la arquitectura Cliente/Servidor es su escalabilidad. Esto es debido a que los servidores pueden manejar varias peticiones en paralelo permitiendo que puedan manejar grandes cantidades de tráfico.

Es importante considerar que, en el diagrama anterior, el cliente se refiere al Frontend y el servidor se refiere al Backend.

En este proyecto se ha utilizado Angular que actúa como una capa intermedia entre el cliente final y el servidor. Angular se comunica con el servidor a través de servicios para recuperar los datos necesarios para mostrar la información en la página web.

Por otro lado, Laravel se ha utilizado en el lado del servidor para construir la aplicación web. Se encarga de procesar las solicitudes de los clientes y generar las respuestas.

4.2 Actores

Hemos tenido en cuenta 4 tipos diferentes de usuarios: cliente, empleado, administrador y usuario no autentificado.

Con lo que respecta al usuario sin identificar, los servicios que se le ofrecen son pocos, podrá ver la página de inicio, de contacto, el formulario de registro y el de inicio de sesión. Y en el caso de que intente acceder a otro tipo de servicio, la aplicación le encaminará directamente al formulario de iniciar sesión donde podrá iniciar sesión si dispone de una cuenta o registrarse.

Los demás usuarios los distinguimos mediante el atributo rol de la clase usuarios:

1. **Usuario cliente**: Sera aquel que tenga en el atributo "rol" con valor "client".
2. **Usuario empleado**: este usuario hace referencia a los empleados que el taller tenga contratados. El atributo "rol" tendrá el valor de "employer". Además de acceder a los servicios que los clientes pueden acceder, podrán acceder al portal de nóminas donde obviamente encontrarán sus nóminas.
3. **Usuario administrador**: Este usuario tendrá el atributo "rol" con valor "admin", y podrá administrar todos los servicios que la aplicación ofrece.

4.3 Casos de uso

En este apartado definiremos las acciones esenciales que debe implementar la aplicación. Cada función está compuesta por una entrada y una salida, lo que quiere decir, que para iniciar una acción se debe introducir unos datos de entrada y producirá ciertos cambios en el servidor o en la vista del usuario.

Requisitos funcionales del cliente

A continuación, se detallarán los casos de uso para el cliente:

4.3.1 Iniciar sesión

Descripción: Los usuarios registrados en la aplicación deben de tener la capacidad de acceder a la aplicación mediante las credenciales que han registrado anteriormente.
Referencia: REF01.
Entrada: Usuario y contraseña.
Salida: Estas son las posibles salidas:

a) Si las credenciales son correctas y el rol del usuario es cliente, accede al menú principal.
b) Si las credenciales son correctas y el rol es empleado accede al menú principal, pero además desde ahí podrá acceder al apartado nóminas.
c) Si las credenciales son correctas y el rol es administrador accederá al panel de administración.
d) Si la credenciales son erróneas se notificará un error por pantalla.

4.3.2 Cerrar sesión

Descripción: Los usuarios deben de poder cerrar la sesión para asegurar la privacidad de sus cuentas.
Referencia: REF02.
Entrada: -
Salida: Tras el cierre de sesión, el usuario será redirigido automáticamente a la página de inicio de sesión.

4.3.3 Registrar cuenta

Descripción: La aplicación debe permitir a los nuevos usuarios el registro en la aplicación mediante el formulario pertinente.
Referencia: REF03.
Entrada: Atributos de la clase usuario como son el nombre, apellido, teléfono, DNI, correo electrónico, etc.
Salida: La cuenta de usuario se registra correctamente en la base de datos del sistema.

4.3.4 Gestionar cuenta de usuario

Descripción: Los usuarios registrados en la aplicación deben de tener la posibilidad de actualizar los datos de su cuenta registrada.
Referencia: REF04.
Entrada: Datos que van a ser añadidos o actualizados.
Salida: Los usuarios pueden actualizar la información de su cuenta.

4.3.5 Registrar vehículo

Descripción: La aplicación debe permitir a los usuarios la posibilidad de registrar un vehículo nuevo para posteriormente pedir reservas en la aplicación.
Referencia: REF05.
Entrada: Datos necesarios para la creación de un registro en la tabla vehículos.
Salida: El registro queda registrado correctamente en la base de datos.

4.3.6 Registrar vehículo de ocasión

Descripción: Es importante ofrecer a los usuarios la posibilidad de publicar o no un vehículo como oferta de segunda mano en la aplicación, durante el proceso de registro correspondiente.
Referencia: REF06.
Entrada: Imágenes del vehículo, precio y opinión.
Salida: El vehículo queda registrado correctamente en la base de datos.

4.3.7 Observar vehículos de ocasión

Descripción: Los usuarios registrados en la aplicación deben de tener la capacidad de observar los vehículos de ocasión disponibles.
Referencia: REF07.
Entrada: -.
Salida: Los usuarios reciben una lista de los vehículos de ocasión disponibles junto a información adicional como la marca, el modelo, el año, el precio, etc.

4.3.8 Realizar reserva

Descripción: Para permitir a los usuarios de la aplicación acceder a los servicios de reparación disponibles, se les ofrece la opción de realizar una reserva para llevar su vehículo al taller.
Referencia: REF08.
Entrada: Fecha y hora, identificador del vehículo e identificador de usuario.
Salida: En esta ocasión tenemos dos posibles salidas:

- **a)** La cita es registrada correctamente en la base de datos.
- **b)** La cita no se ha podido registrar.

4.3.9 Registrar método de pago

Descripción: Los usuarios registrados en la aplicación deben de tener la capacidad de registrar un método de pago para poder realizar las transacciones derivadas de las facturas.
Referencia: REF09.
Entrada: Datos de la tarjeta de crédito o débito que se quiera registrar.
Salida: Estas son las posibles salidas:

- **a)** Si los datos de la tarjeta son válidos, se registra la tarjeta en la base de datos y se le notifica al usuario.
- **b)** Si los datos de la tarjeta no son válidos, se notificará un error por pantalla y se le solicitará al usuario que ingrese información valida.

4.3.10 Revisar factura

Descripción: Los usuarios deben de tener la posibilidad de revisar las facturas derivadas de la reparación de su vehículo.
Referencia: REF10.
Entrada: Identificador de factura y de vehículo.
Salida: Estas son las posibles salidas:

- **a)** Si se proporciona los identificadores válidos, se muestra la información de la factura correspondiente.
- **b)** Si no se encuentra ninguna factura con esos identificadores se mostrará un mensaje de error en pantalla.

4.3.11 Pagar factura

Descripción: Los usuarios registrados deben tener la posibilidad de pagar una factura mediante los métodos de pagos registrados previamente.
Referencia: REF11.
Entrada: Identificador de factura y método de pago.
Salida: Estas son las posibles salidas:

- **a)** Si los datos introducidos son válidos, se realiza el pago de la factura correspondiente y se le notifica al usuario.
- **b)** Si los datos nos son válidos, se notificará un error por pantalla y se solicitará al usuario que ingrese información valida.

Requisitos funcionales del empleado

Además, de los requisitos funcionales que hemos visto para el cliente y que también se aplican para los empleados, encontramos:

4.3.12 Consultar nóminas

Descripción: Los usuarios registrados cuyo rol es empleado, deben de poder consultar sus nóminas en la aplicación.
Referencia: REF12.
Entrada: Identificador de empleado.
Salida: Estas son las posibles salidas:

- **a)** Si se proporciona un identificador valido, se muestra la información de la nómina que pertenece a dicho usuario.
- **b)** Si no se encuentra ninguna nómina asociada, se mostrará un mensaje de error por pantalla.

Requisitos funcionales del administrador

A continuación, se mostrarán los requisitos funcionales para el administrador:

4.3.13 CRUD usuarios

Descripción: Los administradores del sistema deben de tener la capacidad de crear, leer, actualizar y eliminar usuarios de la aplicación.
Referencia: REF13.
Entrada: Datos necesarios para la creación, lectura, actualización o eliminación de un usuario dentro de la tabla user de la base de datos.
Salida: Estos son los posibles resultados de la operación:

- **a)** Creación de un usuario con su correspondiente notificación.
- **b)** Lectura de un usuario con su correspondiente notificación.
- **c)** Actualización de un usuario con su correspondiente notificación.
- **d)** Eliminación de un usuario con su correspondiente notificación.

4.3.14 CRUD empleados

Descripción: Los administradores del sistema deben de tener la capacidad de crear, leer, actualizar y eliminar usuarios que son empleados dentro de la aplicación.
Referencia: REF14.
Entrada: Datos necesarios para la creación, lectura, actualización o eliminación de un empleado dentro de la tabla user de la base de datos.
Salida: Estos son los posibles resultados de la operación:

- **a)** Creación de un empleado con su correspondiente notificación.
- **b)** Lectura de un empleado con su correspondiente notificación.
- **c)** Actualización de un empleado con su correspondiente notificación.
- **d)** Eliminación de un empleado con su correspondiente notificación.

4.3.15 CRUD vehículos

Descripción: Los administradores del sistema deben tener permiso para crear, leer, actualizar y eliminar los vehículos registrados en la aplicación.
Referencia: REF15.
Entrada: Datos necesarios para la creación, lectura, actualización o eliminación de un vehículo dentro del sistema.
Salida: Estos son los posibles resultados de la operación:

- **a)** Creación de un vehículo con su correspondiente notificación.
- **b)** Lectura de un vehículo con su correspondiente notificación.
- **c)** Actualización de un vehículo con su correspondiente notificación.

d) Eliminación de un vehículo con su correspondiente notificación.

4.3.16 CRUD vehículos de ocasión

Descripción: Los administradores del sistema deben tener permiso para crear, leer, actualizar y eliminar los vehículos de ocasión que han sido registrados en la aplicación.
Referencia: REF16.
Entrada: Datos necesarios para la creación, lectura, actualización o eliminación de un vehículo de ocasión dentro del sistema.
Salida: Estos son los posibles resultados de la operación:

- **a)** Creación de un vehículo de ocasión.
- **b)** Lectura de un vehículo de ocasión.
- **c)** Actualización de un vehículo de ocasión.
- **d)** Eliminación de un vehículo de ocasión.

4.3.17 CRUD reservas

Descripción: Los administradores del sistema deben de tener la capacidad de crear, leer, actualizar y eliminar reservas creadas a través de la aplicación.
Referencia: REF17.
Entrada: Datos necesarios para la creación, lectura, actualización o eliminación de reservas.
Salida: Estos son los posibles resultados de la operación:

- **a)** Creación de la reserva y posterior notificación del cliente.
- **b)** Lectura de la reserva: se muestra la información de la reserva.
- **c)** Actualización: se actualiza la información de la reserva y se notifica al cliente.
- **d)** Eliminación de la reserva y posterior notificación al cliente.

4.3.18 CRUD reparaciones

Descripción: Los administradores del sistema deben de tener la capacidad de crear, leer, actualizar y eliminar las reparaciones que están registradas en el sistema.
Referencia: REF18.
Entrada: Datos necesarios para la creación, lectura, actualización o eliminación de una reparación.
Salida: Estos son los posibles resultados de la operación:

- **a)** Creación de una reparación.
- **b)** Lectura de la información de una reparación.
- **c)** Actualización de los datos derivados de una reparación.
- **d)** Eliminación de todos los datos provenientes de una reparación.

4.3.19 CRUD facturas

Descripción: Los administradores del sistema deben de tener la capacidad de crear, leer, actualizar y eliminar las facturas registradas en el sistema.
Referencia: REF19.
Entrada: Datos necesarios para la creación, lectura, actualización o eliminación de una factura dentro de la tabla Invoice de la base de datos.
Salida: Estos son los posibles resultados de la operación:

- **a)** Creación de una factura dentro del sistema.
- **b)** Lectura de los datos de una factura dentro del sistema.
- **c)** Actualización de los datos de una factura dentro del sistema.
- **d)** Eliminación de todos los datos derivados de una factura dentro del sistema.

4.3.20 CRUD pagos

Descripción: Los administradores del sistema deben de tener la capacidad de crear, leer, actualizar y eliminarlos pagos que se han realizado en la aplicación.
Referencia: REF20.
Entrada: Datos necesarios para la creación, lectura, actualización o eliminación de un pago dentro de la base de datos.
Salida: Estos son los posibles resultados de la operación:

- **a)** Creación de un nuevo registro dentro de la tabla correspondiente a los pagos realizados en la aplicación.
- **b)** Lectura de la información derivada de un pago.
- **c)** Actualización de los datos que corresponden a un registro de la tabla pagos.
- **d)** Eliminación de un registros dentro de la tabla pagos.

4.3.21 CRUD métodos de pago

Descripción: Los administradores del sistema deben de tener la capacidad de crear, leer, actualizar y eliminar los métodos de pago que lo usuarios registran en la aplicación.
Referencia: REF21.
Entrada: Datos necesarios para la creación, lectura, actualización o eliminación de un método de pago dentro del sistema.
Salida: Estos son los posibles resultados de la operación:

- **a)** Creación de un nuevo método de pago.
- **b)** Lectura de la información correspondiente a un método de pago.
- **c)** Actualización de la información derivada de un método de pago.

d) Eliminación de un registro dentro de la tabla métodos de pago.

4.3.22 CRUD nóminas

Descripción: Los administradores del sistema deben de tener la capacidad de crear, leer, actualizar y eliminar la nóminas derivadas del trabajo de los empleados.
Referencia: REF22.
Entrada: Datos necesarios para la creación, lectura, actualización o eliminación de una nómina dentro del sistema.
Salida: Estos son los posibles resultados de la operación:

- **a)** Creación de una nueva nómina.
- **b)** Lectura de la información derivada de las nóminas registradas.
- **c)** Actualización de la información de las nóminas.
- **d)** Eliminación de un registro dentro de la clase nóminas.

4.3.23 CRUD servicios

Descripción: Los administradores del sistema deben de tener la capacidad de crear, leer, actualizar y eliminar de los servicios que ofrece el taller en sus reparaciones.
Referencia: REF23.
Entrada: Datos necesarios para la creación, lectura, actualización o eliminación de un servicio dentro de la base de datos.
Salida: Estos son los posibles resultados de la operación:

- **a)** Creación de un nuevo servicio.
- **b)** Lectura de la información que ofrece un servicio.
- **c)** Actualización de los datos de un servicio.
- **d)** Eliminación de un servicio.

4.3.24 CRUD productos

Descripción: Los administradores del sistema deben de tener la capacidad de crear, leer, actualizar y eliminar los productos que ofrecen en la aplicación.
Referencia: REF24.
Entrada: Datos necesarios para la creación, lectura, actualización o eliminación de un producto registrado dentro del sistema.
Salida: Estos son los posibles resultados de la operación:

- **a)** Creación de un nuevo producto.
- **b)** Lectura de la información derivada de un producto.
- **c)** Actualización de la información de un producto.
- **d)** Eliminación de un producto.

4.3.25 CRUD proveedores

Descripción: Los administradores del sistema deben de tener la capacidad de crear, leer, actualizar y eliminar proveedores registrados en el sistema.
Referencia: REF25.
Entrada: Datos necesarios para la creación, lectura, actualización o eliminación de un proveedor dentro del sistema.
Salida: Estos son los posibles resultados de la operación:

- **a)** Creación de un nuevo proveedor.
- **b)** Lectura de la información derivada de un proveedor.
- **c)** Actualización de los datos de un proveedor.
- **d)** Eliminación de un proveedor.

4.3.26 CRUD categorías

Descripción: Los administradores del sistema deben de tener la capacidad de crear, leer, actualizar y eliminar las categorías dentro de la aplicación.
Referencia: REF26.
Entrada: Datos necesarios para la creación, lectura, actualización o eliminación de una categoría dentro de la aplicación.
Salida: Estos son los posibles resultados de la operación:

- **a)** Creación de una nueva categoría.
- **b)** Lectura de la información derivada de una categoría.
- **c)** Actualización de los datos de una categoría.
- **d)** Eliminación de una categoría dentro de la base de datos.

4.4 Diseño de la capa de persistencia

En este apartado se describirán las diferentes entidades que forman parte del proyecto. En primer lugar, se mostrará el diagrama de clases que se ha diseñado y posteriormente se describirá cada una de las clases junto a sus atributos y relaciones.

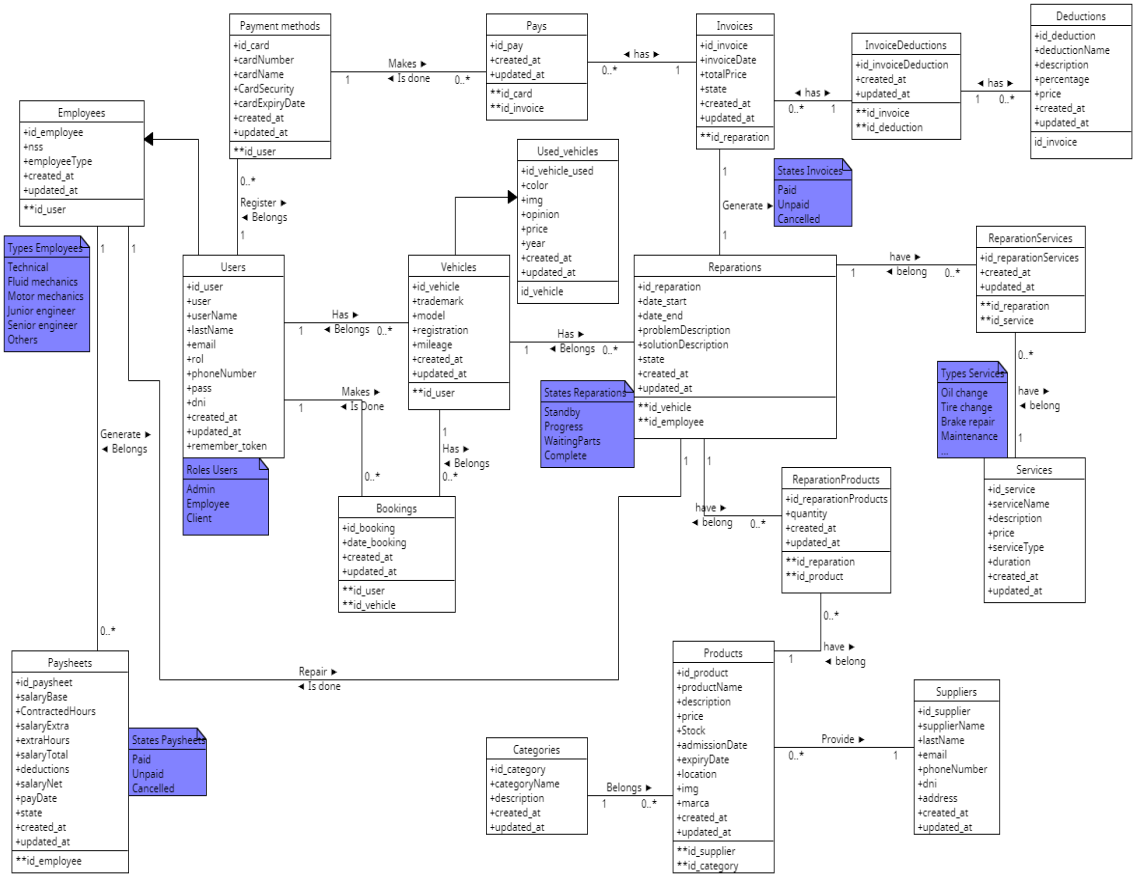


Imagen22: Diagrama de clases UML

Con el objetivo de proporcionar una mejor explicación del diagrama de clases, a continuación, serán descritas las entidades que podemos observar en el diagrama.

4.4.1 Clase User

Esta clase representa a un usuario que tiene la capacidad de realizar diversas acciones en el sistema. Entre los atributos más comunes de esta entidad encontramos el nombre, apellidos, dirección de correo electrónico, contraseña, entre otros.

Además, el atributo 'type' define la relación que el usuario mantiene con el sistema y puede tener 3 valores distintos: 'admin', 'employee' y 'client'. Dependiendo del valor de este atributo, el usuario podrá realizar diferentes acciones en el sistema.

4.4.2 Subclase Employees

Esta clase representa a los empleados del taller. Es una clase derivada de la clase 'user' que añade el atributo 'nss', que representa al número de la seguridad social y el atributo 'speciality' que representa el tipo de trabajo que realiza y para el que han sido contratados.

4.4.3 Clase Paysheet

Esta clase representa las nóminas correspondientes a los empleados de las empresa, y cuenta con una variedad de atributos para ello, tales como el salario, las horas contratadas, las horas extra realizadas, entre otros.

Además, incluye el atributo 'State' el cual puede tomar uno de tres valores posibles: 'paid', 'unpaid' y 'cancelled'. Cada uno de estos valores proporciona información sobre el estado de la nómina, indicando si el pago ha sido realizado o no.

4.4.4 Clase Booking

Esta clase representa la reserva de un servicio específico. Incluye información como la fecha y la hora, así como el identificador de usuario y el identificador del vehículo correspondiente.

4.4.5 Clase Vehicle

Esta clase representa un vehículo registrado en el sistema por un usuario. Tenemos atributos típicos que describen al vehículo como la marca, el modelo, el kilometraje, entre otros.

4.4.6 Subclase Used Vehicle

Esta clase representa los vehículos de ocasión y que son anunciados en la aplicación web. El cliente si lo desea, puede marcar cuando registra su vehículo, si quiere que este se anuncie en vehículos de ocasión. Para esta función deberá rellenar 3 campos adicionales: incorporar una o varias imágenes del vehículo, una opinión o descripción y un precio del cual el taller se llevará una comisión en caso de ser vendido.

4.4.7 Clase Reparation

Esta clase representa la reparación o la aplicación de un servicio a un vehículo. Tenemos en ella atributos que representan la fecha de inicio y la de fin, el problema encontrado y la solución propuesta.

Además, incluimos el atributo 'State' que puede tener 4 posibles valores: 'Standby', 'Progress', 'WaitingParts' y 'Complete'. Con estos atributos el cliente puede informase del estado en el que se encuentra la reparación.

4.4.8 Clase Service

Esta clase representa los servicios que se pueden aplicar a los vehículos como podría ser el cambio de neumáticos o de aceite. Para ello tenemos atributos que representan el nombre, descripción y precio.

4.4.9 Clase Product

Esta clase hace referencia a los materiales que son necesarios para la aplicación de un servicio en específico. Para ello tenemos atributos que describen estos materiales como el nombre, descripción, precio, stock, entre otros.

4.4.10 Clase Supplier

Hace referencia el proveedor de uno o varios materiales. Necesitamos atributos como el nombre, dirección, email, número de contacto, entre otros.

4.4.11 Clase Category

Hace referencia a la categoría de un producto en concreto. Para ello tenemos atributos que describen la categoría junto a un nombre que la representa.

4.4.12 Clase Payment Methods

En esta clase se guardan los métodos de pago que cada usuario quiera registrar en la aplicación. Para ello tenemos los atributos número de tarjeta, nombre del titular y la fecha de caducidad. Atributos necesarios para que el usuario pueda pagar a través de la aplicación.

4.4.13 Clase Pay

Esta clase guardará todos los pagos que se hagan en la aplicación. Para ello enlaza las clases ‘Payment methods’ e ‘Invoice’

4.4.14 Clase ReparationProducts

Es una clase relación que hace referencia a los productos que contiene la clase reparación. Por ejemplo, tenemos el producto neumático que puede pertenecer a varias reparaciones y que varias reparación pueden hacer uso de este producto. Es por ello por lo que se crea esta clase relación donde además guardamos la cantidad de este producto que se utiliza en la reparación de un vehículo concreto.

4.4.15 Clase ReparationService

Hace referencia a los servicios que son utilizados en la reparación de un vehículo. Como podría ser por ejemplo el cambio de aceite o de neumáticos.

4.4.16 Clase Deduction

Hace referencia a los descuentos que el administrador podrá utilizar. En esta clase se crearán registros tanto de impuestos como de descuentos que luego podrán ser utilizado en las facturas para calcular el precio final.

4.4.17 Clase DeductionInvoices

Hace referencia a los descuentos que son aplicados en una factura, y se creará un registro que relacione la factura con el descuento o impuesto que se le aplique.

4.4.18 Clase Invoice

Hace referencia a la factura que se genera al acabar una reparación y que debe ser pagada por el usuario del vehículo. Tenemos atributos como la fecha y el estado que representa si se ha pagado, si está por pagar o si se ha cancelado.
Además, tenemos el atributo precio que representa lo que debe abonar el usuario y que se calcula automáticamente juntando los precios del servicio y de los productos que se han necesitado para llevarlo a cabo.

4.5 Diseño de la capa de presentación

A continuación, se detallará uno de los aspectos más importantes en cualquier aplicación web. Comúnmente denominada interfaz de usuario permite al cliente navegar entre los diferentes servicios que ofrece la aplicación, es por ello por lo que es una de las partes más importantes en el desarrollo y que de estar mal diseñada puede ocasionar problemas como que el cliente no encuentre el servicio al que quiere acceder, etc.

Con el objetivo de facilitar el posterior desarrollo de la aplicación se han diseñado una serie de prototipos que a continuación serán mostrados y descritos.

Es importante aclarar que estos prototipos muestran la información fundamental que será mostrada en las diferentes partes de la página web diseñada. Esto quiere decir que cosas repetitivas como son el encabezado o el pie de página serán eliminadas en todas las páginas menos en el primer prototipo que será el encargado de mostrar esta información.

4.5.1 Página de inicio

Esta página es muy importante dado que será la primera aproximación que tendrá el cliente con nuestra aplicación. Por lo tanto, en ella encontraremos un menú con los diferentes servicios que proporcionamos y una descripción del taller. Lo importante aquí es que el taller junto con sus imágenes sea lo más importante.

Prototipo diseñado:

Imagen23: Página de inicio

Descripción detallada del prototipo

Como podemos observar en el prototipo, tenemos en primer lugar el título y un botón para iniciar sesión.

A continuación, un menú con todos los servicios proporcionados.

En tercer lugar, un slider donde se irán pasando imágenes del taller del cual se está diseñando la aplicación.

En cuarto lugar, tenemos un poco de información del taller y el eslogan de este.

Por último, tenemos un contenedor que contendrá enlaces a redes externas como Instagram o Facebook, la dirección y el número de contacto.

Este prototipo será mostrado para todos los usuarios(cliente registrado, no registrado, mecánico y administrador).

4.5.2 Inicio de Sesión

Página que de nuevo será compartida para los diferentes tipos de usuarios que hemos mencionado anteriormente. El prototipo diseñado:

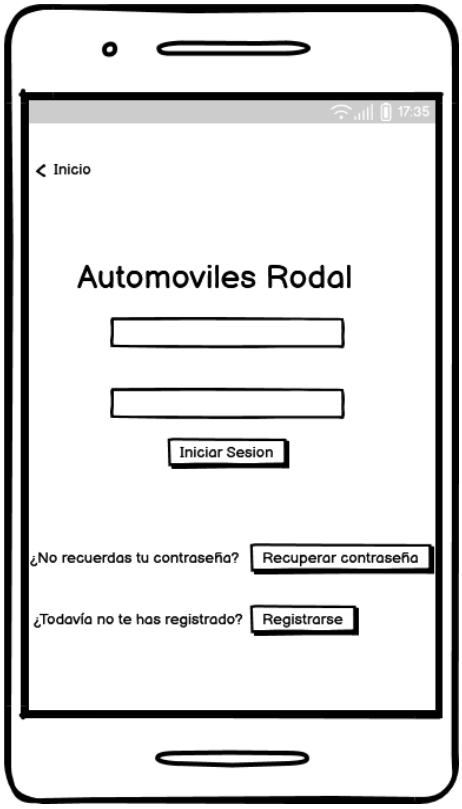


Imagen24: Apartado de inicio de Sesión

Descripción detallada del prototipo

En este prototipo diferenciamos diferentes elementos:

1. Botón Inicio: Volveremos a la pestaña de inicio en caso de no iniciar sesión
2. El título de nuestra página web
3. Usuario y contraseña junto al botón de inicio de sesión para poder iniciar sesión en la aplicación

Además, tenemos dos botones muy importantes para los usuarios

4. Botón de registrarse en la aplicación que nos llevará al formulario de registro
5. Botón de recuperar contraseña que simplemente nos llevará a una página donde nos pedirá el correo electrónico con el que estamos registrados

Este prototipo será mostrado para todos los usuarios(cliente registrado, no registrado, mecánico y administrador).

4.5.3 Formulario de registro

Esta página es de gran importancia, ya que el usuario debe comprende de manera sencilla y práctica los datos debe introducir. Si esto no ocurre, el proceso de registro puede ser tedioso y el negocio podría perder un posible cliente.

El prototipo que hemos diseñado para este cometido y que posteriormente implementaremos:

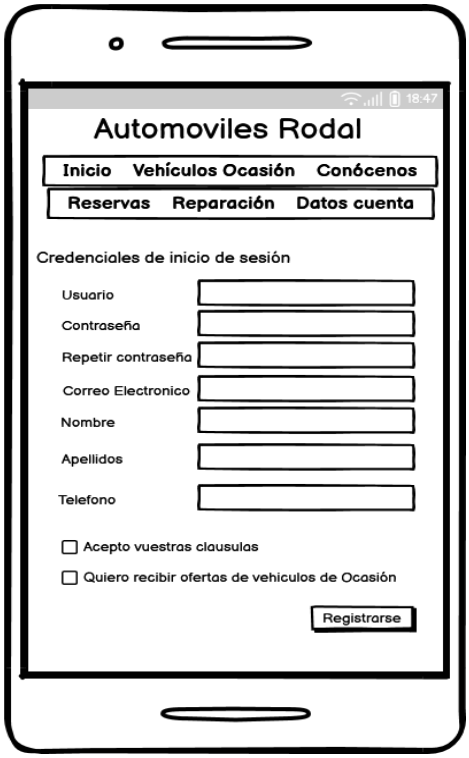


Imagen25: Apartado de registro usuario

Descripción detallada del prototipo

En este prototipo el cliente deberá de introducir una serie de datos para completar el registro. Estos datos necesarios para el registro son:

1. Nombre de usuario
2. Contraseña
3. Correo electrónico
4. Nombre
5. Apellidos
6. Teléfono de contacto

Luego deberá aceptar las políticas de la aplicación y opcionalmente aceptar si quiere recibir en su email las ofertas que el administrador pondrá en el apartado de vehículos de ocasión.

Este prototipo será mostrado para todos los usuarios menos el administrador dado que no será necesario que se registre pues tendrá unas credenciales por defecto que podrá cambiar en el panel de administrador.

4.5.4 Detalles de la cuenta

Este prototipo será mostrado para todos los usuarios(cliente registrado, no registrado, mecánico y administrador).

El objetivo de este apartado de la aplicación es ofrecer al cliente una forma de modificar los datos que introdujo a la hora del registro. Además de administrar sus vehículos, vehículos de ocasión, métodos de pago… como iremos viendo más adelante.

El prototipo que hemos diseñado para este cometido y que posteriormente implementaremos:

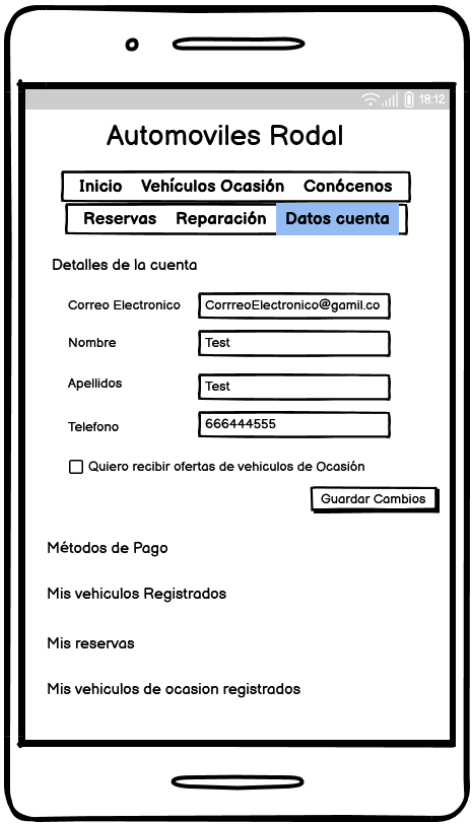


Imagen26: Detalles cuenta personal

Descripción detallada del prototipo

Como podemos ver en el prototipo tenemos diferentes apartados. En esta ocasión presentamos el primero de ellos donde el usuario podrá cambiar el nombre y apellidos en caso de haberse equivocado, además de cambiar su correo electrónico y su número de teléfono de contacto.

También ofrecemos la posibilidad de que el cliente desmarque la casilla de recibir la ofertas de vehículos de ocasión en el caso de que no quiera recibir más información de este tipo.

Este prototipo será mostrado para todos los usuarios menos el administrador, dado que no será necesario que se registre pues tendrá unas credenciales por defecto que podrá cambiar en el panel de administrador.

4.5.5 Detalles de la cuenta - Pagos

En este prototipo se presenta cómo el cliente podrá añadir un método de pago para poder pagar directamente desde la aplicación cuando se haya finalizado una de las reparaciones.

El prototipo que hemos diseñado para este cometido y que posteriormente implementaremos:

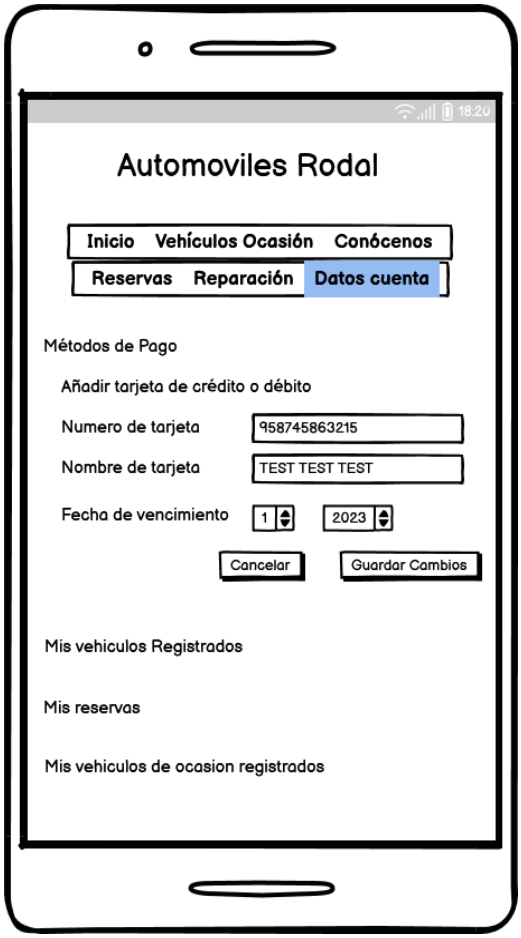


Imagen27: Detalles cuenta personal – Métodos de Pago

Descripción detallada del prototipo

Como podemos observar en el prototipo, el menú para añadir un método de pago se despliega al pulsar en el enlace “Métodos de pago”. En él podremos registrar una tarjeta de crédito o débito y guardarla para futuros pagos.

Cabe destacar que no es necesario completar estos campos, pudiendo rellenarlos cuando se vaya a hacer el pago y evitar de esta manera guardar los datos de la tarjeta del usuario en la aplicación.

4.5.6 Detalles de la cuenta – Vehículos Registrados

Este prototipo muestra cómo el cliente de la aplicación podrá acceder y revisar los vehículos que tiene registrados, así como modificarlos o añadir uno nuevo en caso de que fuese necesario.

El prototipo que hemos diseñado para este cometido y que posteriormente implementaremos:

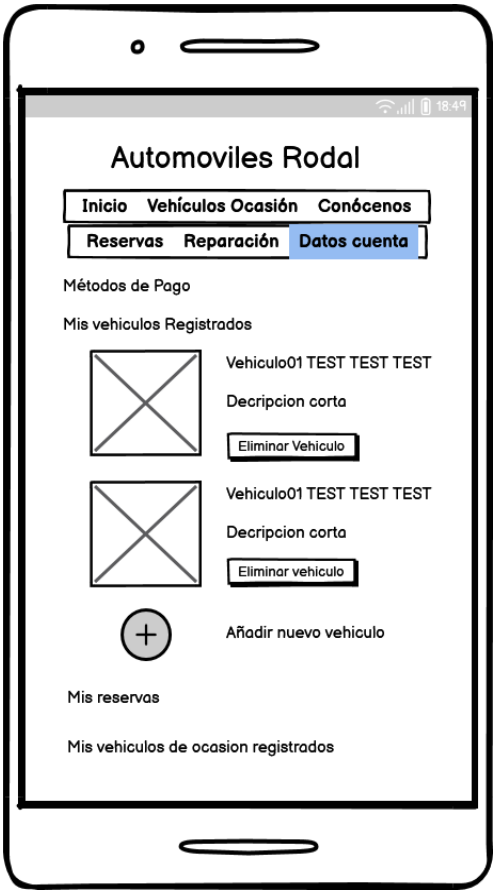


Imagen28: Detalles cuenta personal – Vehículos Registrados

Descripción detallada del prototipo

Como podemos apreciar en el prototipo, el cliente al desplegar el menú de "Mis vehículos Registrados", podrá ver los vehículos que ha registrado en la aplicación.

Además, podrá añadir, eliminar o modificar vehículos.

El apartado "vehículos de ocasión" funcionará igual que este apartado, pero como vimos en el diagrama de clases, tendrá 2 atributos más, las imágenes que serán mostradas en la aplicación con respecto al vehículo, y el precio que el propietario ha elegido para su vehículo.

4.5.7 Detalles de la cuenta – Añadir nuevo vehículo

Este prototipo muestra cómo el cliente de la aplicación, a través del formulario que se muestra en el prototipo podrá registrar un nuevo vehículo en la aplicación.

El prototipo que hemos diseñado para este cometido y que posteriormente implementaremos:

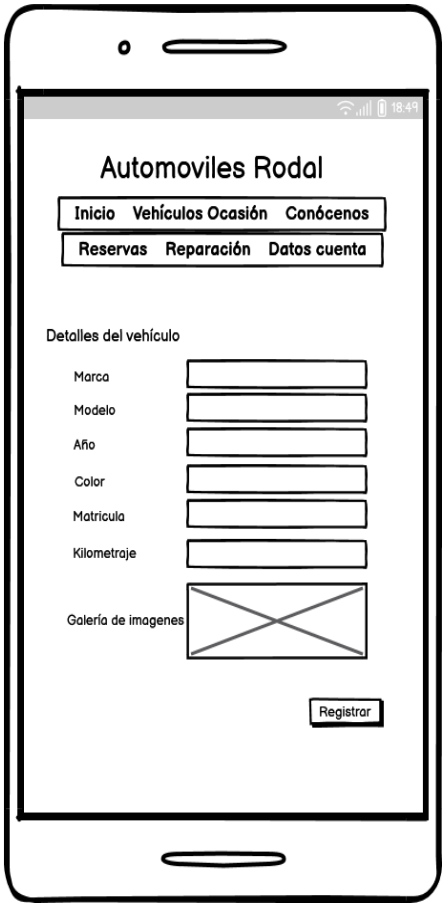


Imagen29: Detalles cuenta personal – Añadir nuevo vehículos

Descripción detallada del prototipo

Como podemos observar, para registrar un vehículo en la aplicación únicamente será necesario rellenar un pequeño formulario con los datos esenciales del vehículo.

El campo de imágenes será obligatorio en caso de querer registrar el vehículo también en la sección de vehículos de ocasión, sin embargo, si ese no es el propósito, podría dejar vacío este campo.

4.5.8 Detalles de la cuenta - Reservas

Este prototipo muestra cómo el cliente de la aplicación puede consultar la reservas que tiene registradas en la aplicación. Es un apartado bastante importante y el cliente debe entender de una manera fácil y sencilla cómo registrar la reserva y qué datos debe introducir como veremos en el siguiente prototipo.

El prototipo que hemos diseñado para este cometido y que posteriormente implementaremos:

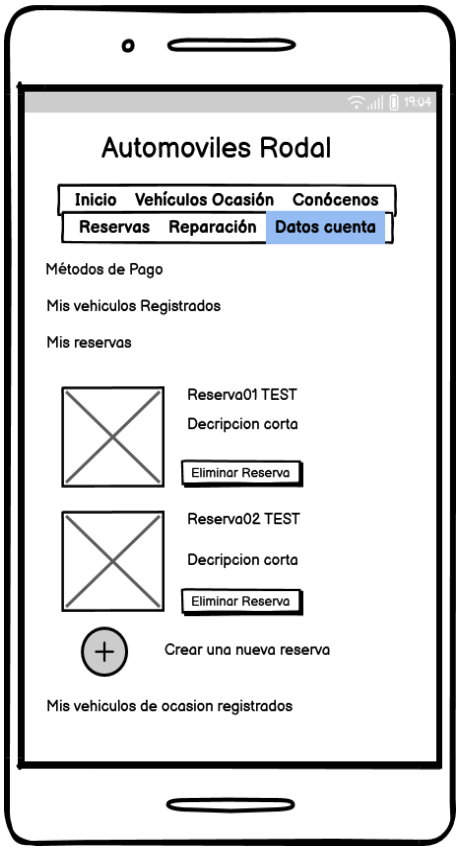


Imagen30: Detalles cuenta personal – Reservas

Descripción detallada del prototipo

Como podemos ver en el prototipo, el cliente podrá listar de una manera sencilla las reservas que tiene registradas y en el caso que lo desee eliminarlas de la aplicación.

Por otro lado, también podrá añadir una nueva reserva dándole en el botón.

Cabe destacar que en el menú también tenemos un apartado único para las reservas, mostrará la misma información que podemos ver en el apartado datos de cuenta, por ello no se ha diseñado un prototipo nuevo, dado que la información a mostrar es la misma que la de este prototipo.

4.5.9 Detalles de la cuenta -Añadir nueva reserva

Este prototipo muestra cómo el cliente de la aplicación, mediante un formulario, puede crear una nueva reserva dentro del sistema de la aplicación.

El prototipo que hemos diseñado para este cometido y que posteriormente implementaremos:

Imagen31: Detalles cuenta personal – Añadir nueva reserva

Descripción detallada del prototipo

Como podemos ver en el prototipo, mediante un pequeño formulario el cliente registrará una nueva reserva. Estos campos obviamente tendrán que ser el vehículo que quiere registrar, la descripción del problema que el cliente puede observar y que servirá a los mecánicos para hacerse una idea, la fecha y la hora de la reserva.

Cabe destacar que en el menú también tenemos un apartado único para las reservas, mostrará la misma información que podemos ver en el apartado datos de cuenta, por ello no se ha diseñado un prototipo nuevo, dado que la información a mostrar es la misma que la de este prototipo.

4.5.10 Detalles de la cuenta - Nóminas

Este prototipo muestra cómo el empleado puede consultar las nóminas calculadas en base al salario y las horas trabajadas.

El prototipo que hemos diseñado para este cometido y que posteriormente implementaremos:

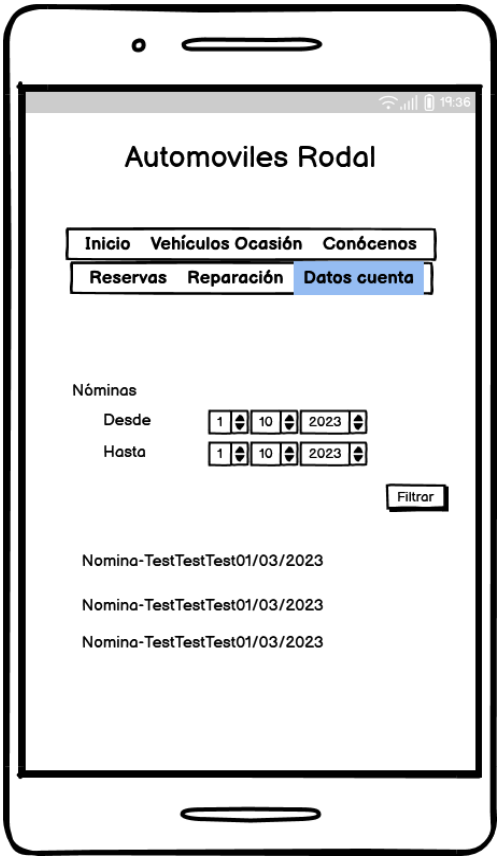


Imagen32: Detalles cuenta personal – Nóminas

Descripción detallada del prototipo

Como podemos ver en el prototipo, el empleado podrá consultar sus nóminas y aplicar un filtro para que solo se listen aquellas nóminas en las que el empleado está interesado.

Cuando el empleado clique en una de las nóminas esta se mostrará junto a todos su datos y podrá descargarla pulsado en el botón imprimir que le aparecerá.

4.5.11 Reparación

Este prototipo muestra cómo el cliente de la aplicación puede consultar las reparaciones de sus vehículos que se están llevando a cabo o que ya han finalizado.

El prototipo que hemos diseñado para este cometido y que posteriormente implementaremos

Imagen33: Reparaciones

Descripción detallada del prototipo

Como podemos ver en el prototipo, el cliente podrá listar, de una manera sencilla, las reparaciones que se han realizado o que están en proceso ahora mismo.

Aparte de dar información sobre el estado de la reparación, se informa también de si la factura se ha pagado o esta impagada. Como describimos en los objetivos, el cliente tendrá la elección de pagar en tarjeta mediante la aplicación o pagar en efectivo en el taller, en cuyo caso, será el administrador del taller el encargado de cambiar el estado de la factura derivada de la reparación a pagado.

4.5.12 Pago reparación/Facturas

Este prototipo muestra cómo el cliente de la aplicación puede descargase la factura derivada de la aplicación, además de pagar el coste derivado de ella.

El prototipo que hemos diseñado para este cometido y que posteriormente implementaremos

Imagen34: Pagos

Descripción detallada del prototipo

Como podemos ver en el prototipo, el cliente podrá efectuar el pago del costo calculado en base al precio del servicio, el precio de los productos y el precio de mano de obra que el administrador indicará.

Esta factura podrá ser descargada en formato PDF y describirá los servicios que se han aplicado, y los productos que se han utilizado para la reparación o cualquier servicio que se haya aplicado.

Por último, podemos seleccionar el método de pago que queramos utilizar y pagar a través de la aplicación.

4.5.13 Vehículos de ocasión

Este prototipo muestra cómo el cliente de la aplicación puede consultar la reservas que tiene registradas en la aplicación. Es un apartado bastante importante y que dado que el a partir de la reserva se construye casi todos los servicios que ofrece la aplicación, es por ello por lo que el cliente debe entender de una manera fácil y sencilla como registrar la reserva y que datos debe introducir como veremos en el siguiente prototipo.

El prototipo que hemos diseñado para este cometido y que posteriormente implementaremos:

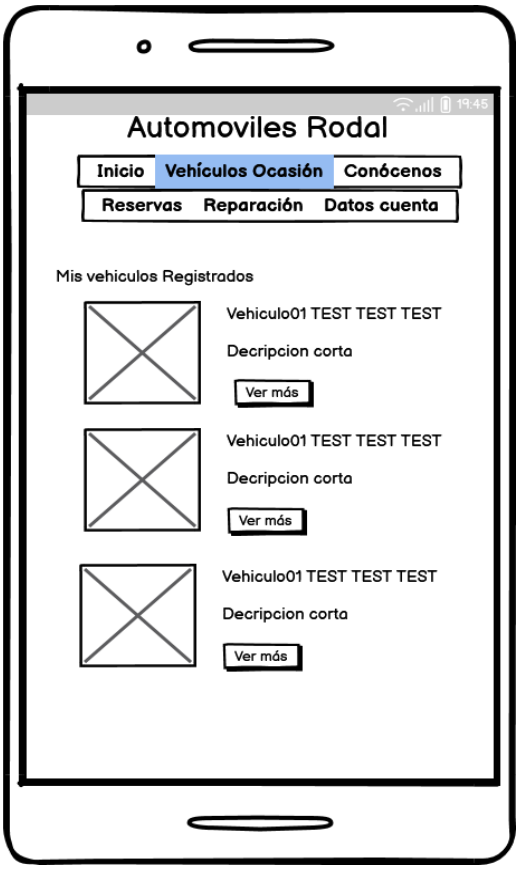


Imagen35: Vehículos de ocasión

Descripción detallada del prototipo

Como podemos ver en el prototipo, el cliente podrá listar de una manera sencilla los diferentes vehículos que han sido anunciado a través de la aplicación.

En primera instancia podremos ver el nombre del vehículo y una breve descripción de este. En el caso que le interese saber más al cliente, podrá pulsar en ver más y observar el precio, imágenes y número de contacto del dueño del vehículo.

La aplicación solo hará de intermediaria, no se hará cargo de ninguna venta y tampoco se llevará ninguna comisión por anunciar los vehículos a través de la aplicación

Con este último prototipo terminamos las vistas de usuarios y empleados, continuaremos mostrando las vistas para el administrador.

4.5.14 Panel de administración

Este prototipo muestra la página de inicio del usuario que entre mediante una cuenta de administrador.

El prototipo que hemos diseñado para este cometido y que posteriormente implementaremos:

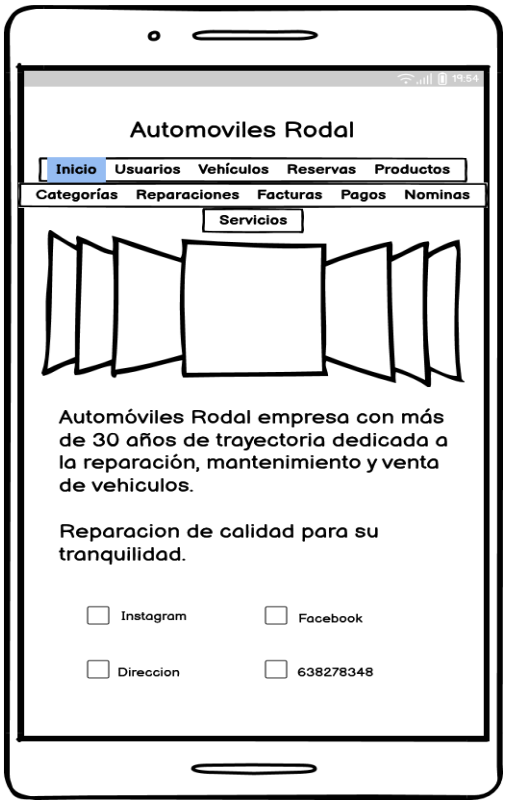


Imagen36: Panel de administrador

Descripción detallada del prototipo

Como podemos ver en el prototipo, cambiará totalmente el menú del administrador. Mediante este menú podremos navegar entre los diferentes CRUD que el administrador debe ser capaz de hacer como expusimos.

Estos apartados son usuarios, vehículos, reservas, productos, categorías, reparación, facturas, pagos, nóminas y servicios. Y en cada uno de ellos el administrador podrá leer, crear, actualizar y eliminar registros pertenecientes a esa tabla

4.5.15 Prototipo CRUD

Para evitar alargar más de la cuenta la memoria hemos decidido agrupar en un solo prototipo todas las acciones que el administrador puede realizar.

El prototipo que hemos diseñado para este cometido y que posteriormente implementaremos:

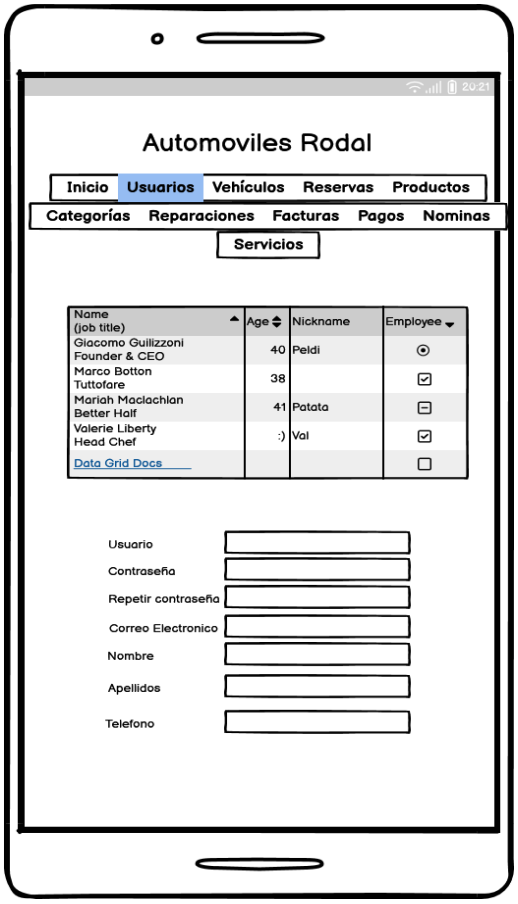


Imagen37: Prototipo CRUD

Descripción detallada del prototipo

Como podemos ver en el prototipo, el administrador listará todos los registros pertenecientes a esa tabla y será capaz de eliminar o actualizar.

En el caso de que quisiera crear un nuevo registro en la tabla, como podría ser un nuevo usuario, vehículo, etc. Se abrirá una tabla bajo para que el administrador pueda rellenar los campos necesarios para crear un nuevo registro en la tabla.

Para acabar este apartado quiero comentar que los prototipos mostrados son una idea inicial pero que a lo largo del desarrollo estos pueden cambiar y mejorarse por lo tanto lo mostrado en este apartado no es definitivo.

5 Desarrollo de la aplicación

Este capítulo detallará el proceso que se ha seguido para realizar el desarrollo de la aplicación. Para ello dividiremos el capítulo en dos apartados generales:

Por un lado, detallaremos el desarrollo interno de la aplicación (backend), donde veremos la creación de la base de datos, la creación de los modelos y controladores, etc.

En segundo lugar y para concluir este apartado, veremos el desarrollo de la parte exterior de la aplicación, es decir, como realizamos las peticiones al servidor, los documentos HTML que generaremos, los estilos CSS que aplicaremos, etc.

5.1 Desarrollo del Backend

En primer lugar, es necesario tener las herramientas necesarias instaladas y bien configuradas Node, NPM, Composer, etc.

Con todas las herramientas instaladas correctamente, en los siguiente subapartados se detallarán el proceso que se ha seguido para el desarrollo de la aplicación.

5.1.1 Estructura del proyecto Larabel.

La siguiente imagen muestra las estructura de directorios y archivos del proyecto:

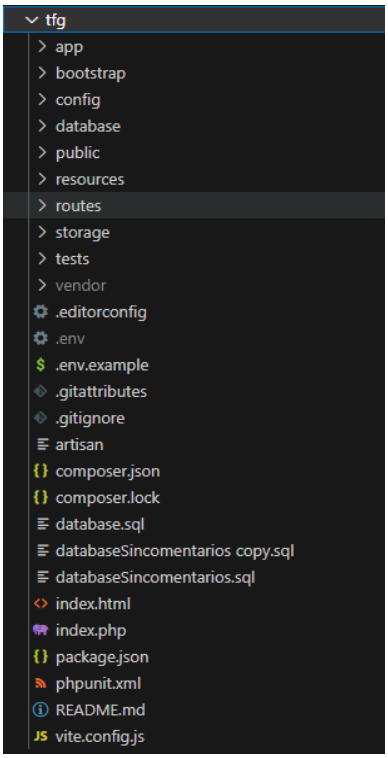


Imagen38: Prototipo CRUD

De todos estos archivos los más importantes son:

1. **/app:** Contiene la mayoría de los archivos de la aplicación Laravel, incluyendo modelos, controladores, middleware y otros componentes principales.
2. **/bootstrap:** Contiene archivos relacionados con el arranque y carga inicial de la aplicación.
3. **/config:** Contiene los archivos de configuración del proyecto.
4. **/routes**: Contiene los archivos de rutas que definen las URL y los controladores asociados.
5. **/storage:** Contiene archivos generados por la aplicación, como archivos de registro, sesiones, cachés y archivos cargados por los usuarios.
6. **/tests:** Contiene archivos de pruebas.
7. **/vendor:** Contiene las dependencias del proyecto.
8. **.env:** Contiene la configuración específica del entorno, como la conexión a la base de datos y las credenciales.
9. **composer.json:** Especifica las dependencias del proyecto.

5.1.2 Desarrollo de la base de datos

Un fragmento del código generado para este proceso es el siguiente:

databaseSincomentarios copy.sql

```sql
CREATE DATABASE IF NOT EXISTS tfg_automovilesRodalPruebas;
USE tfg_automovilesRodalPruebas;

CREATE TABLE Users(

    id_user             int auto_increment NOT NULL,
    user                varchar(50)  NOT NULL,
    userName            varchar(50)  NOT NULL,
    lastName            varchar(100) NOT NULL,
    email               varchar(255) NOT NULL,
    rol                 varchar(20)  NOT NULL,
    phoneNumber         int NOT NULL,
    pass                varchar(255) NOT NULL,
    dni                 varchar(20)  NOT NULL,

    created_at          datetime DEFAULT NULL,
    updated_at          datetime DEFAULT NULL,
    remember_token      varchar(255),

    CONSTRAINT pk_user PRIMARY KEY(id_user)

)ENGINE=InnoDb;

INSERT INTO `users` (`id_user`, `user`, `userName`, `lastName`, `email`, `rol`, `phoneNumber`, `pass`, `dni`, `created_at`, `updated_at`, `remember_token`)
            VALUES (NULL, 'admin', 'admin', 'admin', 'admin@test.com', 'admin', '000000000', 'admin', '00000000Z', NULL, NULL, NULL);


CREATE TABLE Employees(

    id_employee         int auto_increment NOT NULL,
    id_user             int NOT NULL,
    nss                 varchar(255) NOT NULL,
    employeeType        varchar(20)  NOT NULL,

    created_at          datetime DEFAULT NULL,
    updated_at          datetime DEFAULT NULL,

    CONSTRAINT pk_employee PRIMARY KEY(id_employee),
    CONSTRAINT pk_employee_user FOREIGN KEY(id_user) REFERENCES Users(id_user)

)ENGINE=InnoDb;
```

Imagen39: Fragmento código SQL

Como podemos apreciar en la imagen desarrollamos una tabla llamada “Users” que almacenará los usuarios registrados en la aplicación. Tendrá atributos como el nombre, el DNI la contraseña, etc.

Asignamos una clave primaria, como mecanismo de almacenamiento MySQL utilizamos “InnoDB” e insertamos un usuario de prueba.

Seguidamente creamos la tabla empleados(Employers) que tendrá atributos como el numero de la seguridad social, etc. Además, definimos una clave foránea de empleados a usuarios dado que un empleado será un usuario. Y, por último, insertamos un empleado de pruebas.

Este pequeño fragmento que se acaba de detallar se ha realizado para todo el diagrama de clases que diseñamos anteriormente. Una vez ejecutado la estructura es la siguiente:

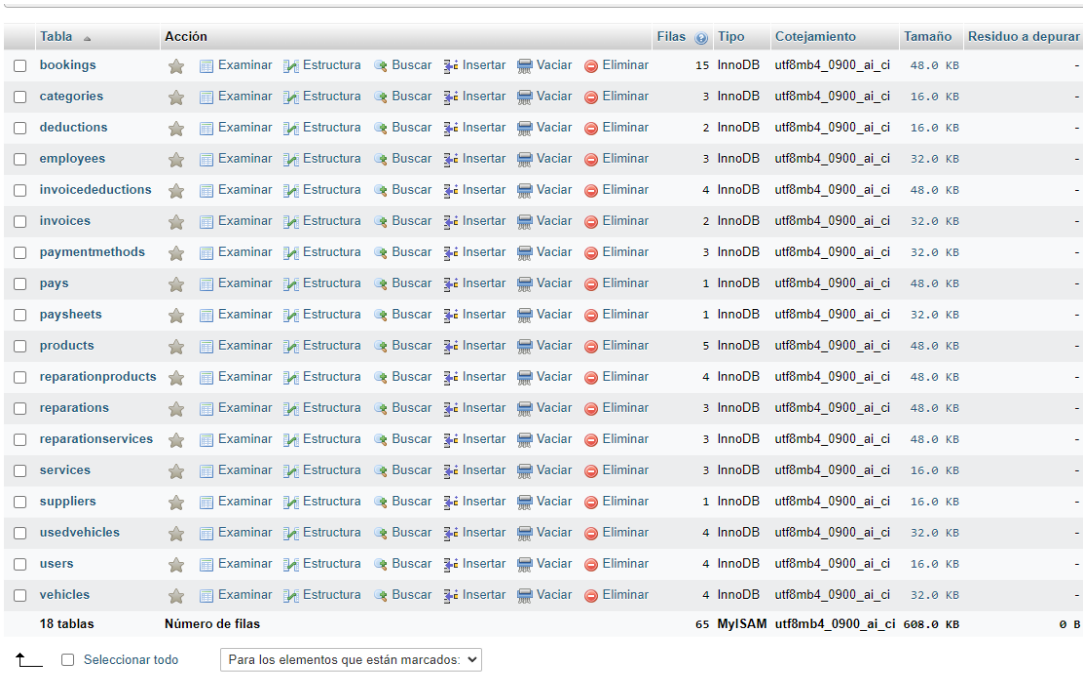


Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
bookings	Examinar Estructura Buscar Insertar Vaciar Eliminar	15	InnoDB	utf8mb4_0900_ai_ci	48.0 KB	-
categories	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_0900_ai_ci	16.0 KB	-
deductions	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_0900_ai_ci	16.0 KB	-
employees	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_0900_ai_ci	32.0 KB	-
invoicedeductions	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_0900_ai_ci	48.0 KB	-
invoices	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	InnoDB	utf8mb4_0900_ai_ci	32.0 KB	-
paymentmethods	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_0900_ai_ci	32.0 KB	-
pays	Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	utf8mb4_0900_ai_ci	48.0 KB	-
paysheets	Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	utf8mb4_0900_ai_ci	32.0 KB	-
products	Examinar Estructura Buscar Insertar Vaciar Eliminar	5	InnoDB	utf8mb4_0900_ai_ci	48.0 KB	-
reparationproducts	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_0900_ai_ci	48.0 KB	-
reparations	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_0900_ai_ci	48.0 KB	-
reparationservices	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_0900_ai_ci	48.0 KB	-
services	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_0900_ai_ci	16.0 KB	-
suppliers	Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	utf8mb4_0900_ai_ci	16.0 KB	-
usedvehicles	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_0900_ai_ci	32.0 KB	-
users	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_0900_ai_ci	16.0 KB	-
vehicles	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_0900_ai_ci	32.0 KB	-
18 tablas	Número de filas	65	MyISAM	utf8mb4_0900_ai_ci	608.0 KB	0 B

Seleccionar todo Para los elementos que están marcados:

Imagen40: Fragmento código SQL

5.1.3 Creación de los modelos

Primero de todo, es necesario definir los modelos en Laravel. Un modelo es una representación de una tabla en la base de datos, es decir, está asociado a una tabla y se utiliza para interactuar con los registros que hay almacenados en ella.

Por tanto, tendremos que crear un modelo por cada tabla que hemos definido en la base de datos. Lo haremos de la siguiente manera para todas las tablas que creamos en el apartado anterior.

```
C:\wamp64\www\ServidorPruebasTFG\tfg>php artisan make:model vehicles

  INFO  Model [C:\wamp64\www\ServidorPruebasTFG\tfg\app/Models/vehicles.php] created successfully.
```

Imagen41: Creación de los modelos

Ejecutando el comando que vemos en la imagen41, creamos los modelos necesarios para la aplicación. Una vez creados, la estructura resultante se puede observar en la imagen42.

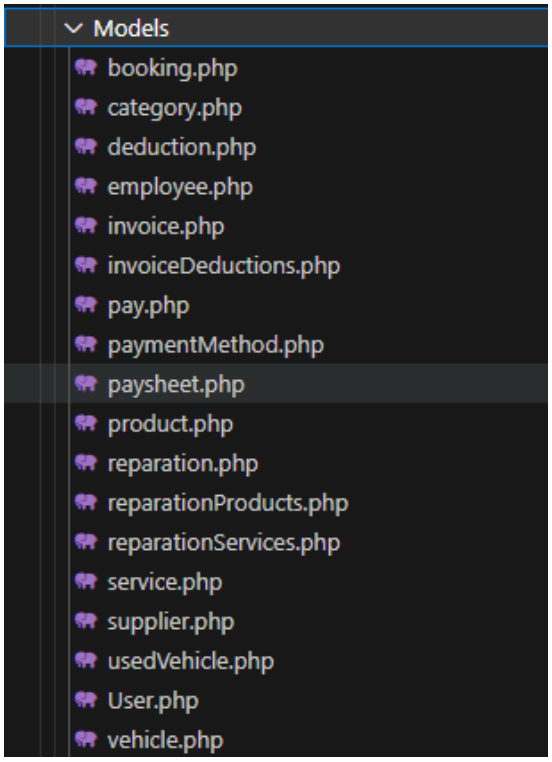


Imagen42: Estructura de los modelos

Ahora tendremos que configurar estos modelos y definir las relaciones con los demás, es decir la claves foráneas que hemos definido al crear la base de datos.

Por ejemplo, en el modelo “product”, el código quedaría de la siguiente manera:

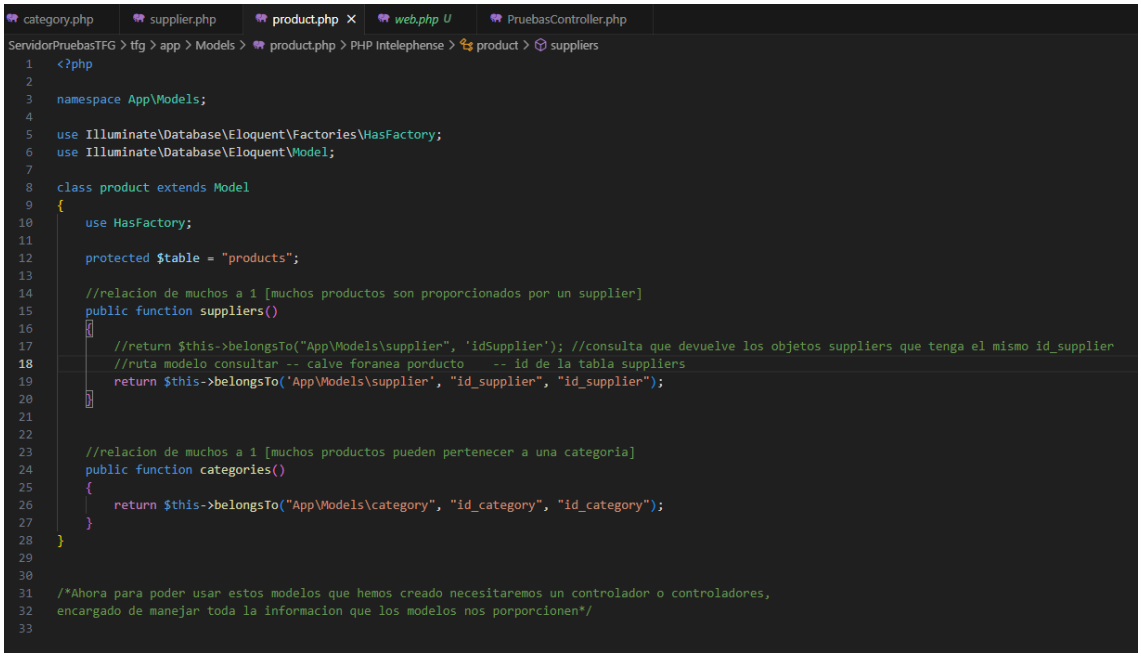


Imagen43: código del modelo product

Como podemos observar en la imagen, definimos una variable “protected” con el nombre de la tabla a la que hace referencia. Posteriormente definimos las relaciones de las demás clases que en este caso son dos:

1. Relación con “suppliers”: Dado que un producto se relaciona con un proveedor(Supplier), y que un proveedor puede proveer uno o muchos productos, la relación la definimos de tipo “belongsTo” pasando como parámetros la ruta del modelo Supplier junto con el atributo que los relaciona en este caso id_supplier.
2. Relación con “categories”: La relación se define de igual manera que en el caso de suppliers.

Por otro lado, tenemos que definir la relación inversa, por ejemplo, en el modelo Supplier:

Imagen44: código del modelo Supplier

De igual manera que hicimos en el modelo product, definimos una variable $table que hace referencia al nombre de la tabla. En este caso además definimos una variable adicional $primaryKey diciendo que la clave primaria se corresponde con el atributo id_supplier. Esto es necesario dado que las consultas en Larabel se hacen de manera automática y Larabel espera que la clave primaria se llame id, por ello para solucionar este problema debemos definir que la clave primaria se llama id_supplier en este caso.

Posteriormente definimos la relación que este modelo tiene con product, que en este caso es la relación inversa que definimos en la clase product “hasMany” dado que un proveedor puede proveer muchos productos.

Lo que acabamos de detallar lo realizaremos para todos los modelos que hemos definido y para probar que todo funciona correctamente hemos definido un test que se detallará en el capítulo de pruebas. *(Ver en capítulo 6, apartado 6.1 Test ORM).*

5.1.4 Creación de los controladores

Primero de todo, es necesario definir los controladores. Mediante los controladores agrupamos y organizamos las solicitudes HTTP de la aplicación.

Por tanto, tendremos un controlador por cada modelo que creamos anteriormente.

Para crear los controladores utilizamos Artisan de la siguiente manera:

```
C:\wamp64\www\ServidorPruebasTFG\tfg>php artisan make:controller UserController

  INFO  Controller [C:\wamp64\www\ServidorPruebasTFG\tfg\app/Http/Controllers/UserController.php] created successfully.
```

Imagen45: creación de un controlador

Una vez tenemos los controladores para cada uno de los modelos creados, podemos empezar a desarrollar la lógica de cada uno. Dado que todos los controladores deben de ser capaces de recibir peticiones CRUD (create, read, update, delete), hemos creado un archivo que implementa estos métodos. De esta manera, creando una instancia de esta clase y llamando a los métodos definidos reducimos el código de cada uno de los controladores.

```php
160     * Funcion para listar todos los objetos registrados
161     */
162    public function index($object, $model)
163    {
164        //con load añadimos un parametro mas al json de la consulta, en este caso el objeto categoria asociado
165        if (!empty($object)) {
166            $response = array(
167                'status' => 'success',
168                'code'   => 200,
169                'message' => "Se ha accedido a todo los $model registrados correctamente.",
170                '$model' => $object
171            );
172        } else {
173            $response = array(
174                'status' => 'error',
175                'code'   => 404,
176                'message' => "No se ha podido acceder a los $model registrados.",
177            );
178        }
179
180        return response()->json($response, $response['code']);
181    }
182
183    /**
184     * Funcion para encontrar un registro
185     */
186    public function show($object, $model, $id)
187    {
188        //podriamos hacer tambien el load aqui para mostrar mas cosas asociada
189        if (!empty($object)) {
190            $response = array(
191                'status' => 'success',
192                'code'   => 200,
193                'message' => "El $model con id=$id ha sido consultado correctamente.",
194                '$model' => $object
195            );
196        } else {
197            $response = array(
198                'status' => 'error',
199                'code'   => 404,
200                'message' => "No se ha encontrado ningun $model asociado al identificador $id.",
201            );
202        }
203
204        return response()->json($response, $response['code']);
205    }
```

Imagen46: creación de un controlador

En la imagen podemos ver dos de los método más comunes:

1. El de mostrar un registro de una tabla filtrado por el identificador
2. El de listar todos los registros de una tabla.

Ahora en el controlador simplemente tendríamos que crear una instancia y llamar a estos métodos. Veamos un ejemplo con el controlador de productos:

```php
/**
 * Funcion para listar todos los productos registrador
 *
 * RUTA: http://tfg.com.devel/product [GET]
 */
public function index()
{
    $crud = new \App\Helpers\CRUD();
    return $crud->index(product::all(), "productos");
}

/**
 * Funcion para mostrar el producto con el $id que se le pase
 *
 * RUTA: http://tfg.com.devel/product/$id [GET]
 */
public function show($id)
{
    //podriamos hacer tambien el load aqui para mostrar mas cosas asociadas
    $crud = new \App\Helpers\CRUD();
    return $crud->show(product::find($id), "producto", $id);
}
```

Imagen47: Métodos index/show de un controlador

El funcionamiento de una petición “get” a la ruta http://tfg.com.devel/product es el siguiente:

Se recibe la petición y se crea un objeto llamado $crud que instancia un objeto de la clase CRUD que mencionábamos anteriormente. Se llama al método index, y se le pasa como parámetros, un nombre llamado productos y el resultado de “product:all()” que básicamente es una consulta SQL, pero utilizando el ORM que nos proporciona Larabel.

El método index de la clase CRUD recibirá todos los registros que se encuentran en la tabla producto y construirá una respuesta en formato JSON que será lo que devolveremos, en este caso, como repuesta a la petición realizada.

Esto lo tendremos que hacer para todos los controladores. Y una vez terminados los controladores, la manera que hemos utilizado para comprobar el correcto funcionamiento de estos, es mediante la herramienta Postman que se detallará en el capítulo de pruebas. *(Ver capítulo 6 pruebas apartado 6.2 Peticiones mediante Postman)*

5.1.5 Creación de las rutas del proyecto

Obviamente es necesario crear una serie de rutas para gestionar las acciones qué deben ejecutar dependiendo de la ruta a la que llegue la petición.

Para definir las rutas en Larabel lo hacemos en el directorio /routes en el archivo web.php.

```php
// (9) Rutas del controlador de Productos
Route::resource('/product', 'App\Http\Controllers\ProductController');
Route::post('/product/storeImage', 'App\Http\Controllers\productController@storeImage');
Route::get('/product/getImage/{filename}', 'App\Http\Controllers\productController@getImage');
Route::get('/product/getProductsByCategory/{id}', 'App\Http\Controllers\productController@getProductsByCategory');
Route::get('/product/getProductsBySupplier/{id}', 'App\Http\Controllers\productController@getProductsBySupplier');

// (10) Rutas del controlador de Suppliers
Route::resource('/supplier', 'App\Http\Controllers\SupplierController');

// (11) Rutas del controlador de Service
Route::resource('/service', 'App\Http\Controllers\ServiceController');

// (12) Rutas del controlador de ReparationProduct
Route::resource('/reparationProducts', 'App\Http\Controllers\reparationProductsController');
Route::get('/reparationProducts/findByCamp/{camp}', 'App\Http\Controllers\reparationProductsController@findByCamp');
Route::get('/reparationProducts/findByCampProduct/{camp}', 'App\Http\Controllers\reparationProductsController@findByCampProduct');

// (13) Rutas del controlador de reparation
Route::resource('/reparation', 'App\Http\Controllers\reparationController');
Route::get('/reparation/findByCamp/{camp}', 'App\Http\Controllers\reparationController@findByCamp');
```

Imagen48: Creación de rutas para el proyecto

En la imagen anterior generamos mediante “Route::resource” todos las rutas necesarias para los métodos CRUD que mencionamos anteriormente.

En el caso del controlador de productos necesitamos rutas adicionales dado que tenemos que tratar con imágenes o debemos filtrar los registros por determinados campos. Para ello básicamente creamos más rutas y asociamos esas rutas a los métodos que definamos en el controlador de productos.

Cuando se avanza en el proyecto, por lo general se tienen muchas rutas, otra cosa que nos proporciona Larabel es el comando “route::list” mediante Artisan.

```
:\wamp64\www\ServidorPruebasTFG\tfg>php artisan route:list

GET|HEAD   /
POST       _ignition/execute-solution ignition.executeSolution › Spatie\LaravelIgnition › ExecuteSolutionController
GET|HEAD   _ignition/health-check ..... ignition.healthCheck › Spatie\LaravelIgnition › HealthCheckController
POST       _ignition/update-config ..... ignition.updateConfig › Spatie\LaravelIgnition › UpdateConfigController
GET|HEAD   api/user
GET|HEAD   category/prueba ..... categoryController@prueba
GET|HEAD   product/prueba ..... productController@prueba
GET|HEAD   pruebas/{nombre?}
GET|HEAD   pruebasController ..... PruebasController@index
GET|HEAD   pruebasControllerConsulta ..... PruebasController@testORM
GET|HEAD   pruebasVista/{nombre?}
GET|HEAD   reparation/prueba ..... ReparationController@prueba
GET|HEAD   sanctum/csrf-cookie ..... sanctum.csrf-cookie › Laravel\Sanctum › CsrfCookieController@show
GET|HEAD   service/prueba ..... ServiceController@prueba
GET|HEAD   serviceProduct/prueba ..... ServiceProductController@prueba
GET|HEAD   serviceReparation/prueba ..... ServiceReparationController@prueba
GET|HEAD   supplier/prueba ..... SupplierController@prueba
GET|HEAD   testORM ..... testORM@testORM
POST       user/login ..... UserController@login
GET|HEAD   user/prueba ..... UserController@prueba
POST       user/register ..... UserController@register
GET|HEAD   vehicle/prueba ..... VehicleController@prueba

                                                    Showing [22] routes
```

Imagen49: Rutas creadas para el proyecto

5.1.6 Instalación de la librería JWT

Para terminar, detallaremos qué es la librería JWT, porque la hemos utilizado en el proyecto y cómo la hemos instalado.

En primer lugar, JWT(JSON Web Token), es una herramienta utilizada para la verificación de tokens generados en aplicaciones web. La hemos utilizado para añadir una capa de seguridad a la aplicación, de esta manera, verificando el token que se mandará siempre en la una de las cabeceras de la petición HTTP, verificamos si el usuario propietario de dicho token puede realizar la acción o no.

A continuación, veremos resumidamente como hemos utilizado esta herramienta: (se debe tener en cuenta que previamente se debe hacer una instalación de las dependencias necesarias para el uso de esta herramienta).

Creación del directorio /helpers y la clase JWT que implementará los métodos tanto para generar el token como para obtener la identidad del usuario a partir del token.

```php
public function signup($usuario, $pass, $getToken = false)
{
    //buscar si existe el usuario con sus credenciales
    $user = User::where([
        'user' => $usuario,
        'pass' => $pass
    ])->first(); //consulta que comprueba si estos dos parametros existen
    //validar
    $signup = false;
    if (is_object($user)) {
        $signup = true;
    }
    //generar token con los datos del usuario identificado
    if ($signup) {
        $token = array( //generamos el token si el usuario existe, es decir si tenemos un objeto en $user
            'sub'    => $user->id_user,
            'user'   => $user->user,
            'userName'   => $user->userName,
            'lastName'   => $user->lastName,
            'email' => $user->email,
            'phoneNumber'   => $user->phoneNumber,
            'dni' => $user->dni,
            'rol'    => $user->rol,
            'iat'    => time(), //fecha en la que se crea el token
            'exp'    => time() + (7 * 24 * 60 * 60) //tiempo que dura el token = 7dias
        );
        $jwt = JWT::encode($token, $this->key, 'HS256'); //generamos el jwt con el token, la clave y el algoritmo de cifrado
        $decode = JWT::decode($jwt, $this->key, array('HS256'));
        //devolver los datos decodificados o el token
        if ($getToken) { //devolvemos el token decodificado
            $data = $decode;
        } else { //delvolvemos el token codificado
            $data = $jwt;
        }
    } else {
        $data = array( //en caso de no ser logeado generamos un mensaje de erro
            'status' => 'error',
            'message' => 'login incorrecto',
            'user' => $user
        );
    }
    return $data;
}
```

Imagen50: Función signup

La función “signup” realiza las acciones:

1. Comprueba que el usuario y la contraseña sean correctos.
2. Si este es el caso genera un token que devuelve en la respuesta.

Por otro lado, el método checkToken:

```php
//comporbacion de tokens
public function checkToken($jwt, $getIdentity = false)
{

    $auth = false;

    try {
        $jwt = str_replace('"', '', $jwt); //remplazamos la comillas en caso de que el token nos llegue entrecomillado
        $decode = JWT::decode($jwt, $this->key, array("HS256"));
    } catch (\UnexpectedValueException $e) {
        $auth = false;
    } catch (\DomainException $e) {
        $auth = false;
    }

    if (!empty($decode) && is_object($decode) && isset($decode->sub)) { //si ha llegado el token y lo hemos podido desencriptar, tenemos dientificador
        $auth = true;
    }

    if ($getIdentity) {
        return $decode;
    } else {
        return $auth;
    }
}
```

Imagen37: Función signup

Esta función resumidamente comprueba si el token es correcto y si, además le pasamos un segundo parámetro a true nos devolverá la identidad de ese token (nombre de usuario, rol, etc).

Creamos un "middleware" que utilizaremos en el método "contruct" de los controladores haciendo más fácil el uso de la librería JWT y reduciendo el código de los controladores.

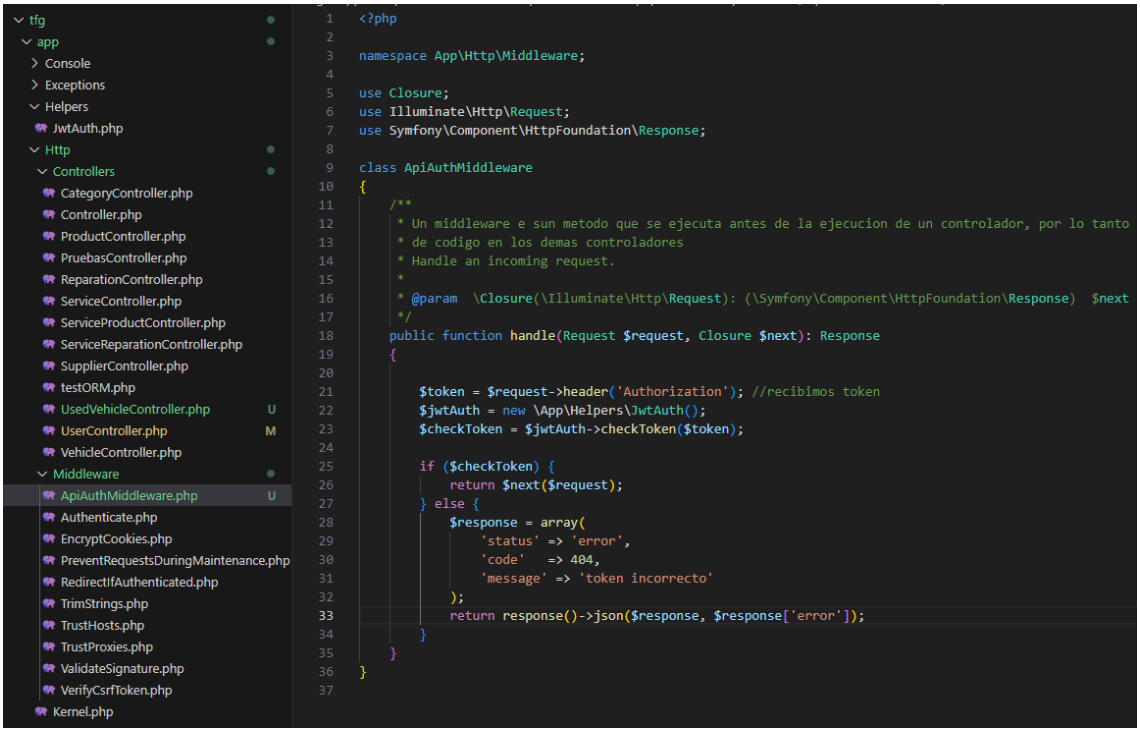


Imagen51: Middleware JWT

El middleware llamará a la función "checkToken" que detallamos anteriormente, para comprobar que el token es correcto.

Por último, el uso de este middleware sería el siguiente:

```php
/**
 * Funcion para utlizar el middleware antes del qualquier metodo de este controlador excepto index y show
 */
public function __construct()
{
    $this->middleware('\App\Http\Middleware\ApiAuthMiddleware::class', ['except' => ['index', 'show']]);
}
```

Imagen52: Uso del middleware

Resumidamente, antes de ejecutar cualquier acción del controlador se ejecutará el middleware comprobando el token que se reciba y devolviendo un error en caso de que el token no sea correcto.

Además, en el caso de los métodos “index” y “show”, métodos que listan todos los registros de una tabla o buscan un registro en concreto, no queremos que se compruebe el token, por ello añadimos en “except” estos dos métodos como podemos observar en la imagen, evitando que se ejecute el middleware si se realizan peticiones a estos dos métodos.

Con esto terminamos la explicación detallada del desarrollo del backend.

5.2 Desarrollo del frontend

En este segundo punto detallaremos como se ha desarrollado el frontend. Hay que tener en cuenta que previamente se debe haber instalado Angular.

5.2.1 Estructura del proyecto Angular

La siguiente imagen muestra las estructura de directorios y archivos del proyecto:

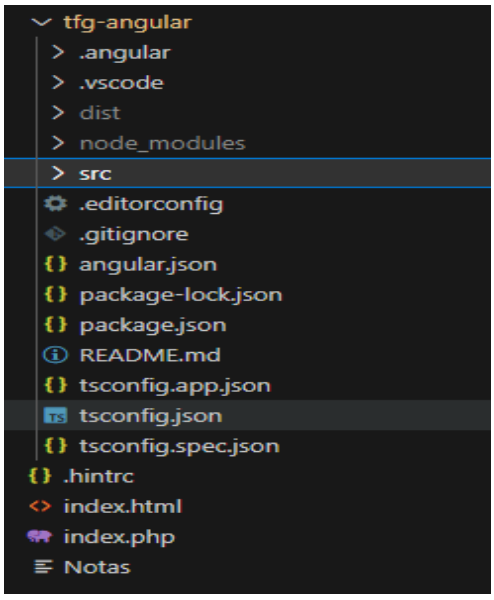


Imagen53: Estructura proyecto Angular

De todos estos archivos los más importantes son:

1. **/src:** Directorio principal donde se encuentran todo el código fuente de la aplicación.
2. **/src/app:** Contiene la mayoría del código fuente, es decir, los componentes , servicios, modelos entre otros elementos.
3. **/src/assets/:** Este directorio se utiliza para los archivos estáticos como los archivos CSS
4. **/src/index.html:** El archivo principal de la aplicación.
5. **/src/main.ts:** Archivo TypeScript que se encarga de iniciar la aplicación, cargando los módulos principales.
6. **/src/styles.css:** Contiene los estilos globales de la aplicación.
7. **/src/environments/:** contiene todos los archivos de configuración del entorno.
8. **Angular.json**: Archivo de configuración principal que define la estructura de directorios del proyecto, las rutas de compilación, etc.
9. **Pakage.json:** Es el archivo de configuración NPM que contiene las dependencias del proyecto.

5.2.2 creación de los modelos

Los modelos en Angular son clases en TypeScript que se utilizan para estructurar los datos de la aplicación. Es decir, definen las propiedades y métodos necesarios para trabajar con los datos que obtengamos de la aplicación. Por tanto, necesitaremos un modelos por cada tabla que tenemos definidos en la base de datos de igual manera que ocurría en Laravel.

```
export class user{

    constructor(
        public id_user: number,
        public user: string,
        public userName: string,
        public lastName:string,
        public email: string,
        public rol: string,
        public phoneNumber: number,
        public pass: string,
        public dni: number,
        public created_at: any,
        public updated_at:any,
        public getToken:any
    ){
    }
}
```

Imagen54: Modelo Angular

Los modelos deberán tener todos los campos que tengamos definidos en la base de datos, de esta manera al recibir la respuesta desde el backend podemos transformar el JSON que recibimos en un objeto y será más fácil trabajar con los datos.

Definiremos los modelos para todas las tablas de igual manera que hemos hecho con los usuarios. *Ruta: \tfg-angular\src\app\models\user.ts*

5.2.3 Creación de los componentes

Un componente en Angular está compuesto por 3 archivos importantes:

1. Clase del componente(TS): define la lógica del componente en TypeScript, cargando un modelo, haciendo peticiones y cargando los datos que necesita, etc.
2. Plantilla HTML: Define la estructura visual que se mostrará al usuario.
3. Archivo CSS: son los estilos privados para el componente.

Generamos el componente de la siguiente manera:

```
C:\wamp64\www\ServidorPruebasTFG\tfg-angular>ng g component components/login
CREATE src/app/components/login/login.component.html (20 bytes)
CREATE src/app/components/login/login.component.spec.ts (552 bytes)
CREATE src/app/components/login/login.component.ts (198 bytes)
CREATE src/app/components/login/login.component.css (0 bytes)
UPDATE src/app/app.module.ts (403 bytes)
```

Imagen55: Generación componente

Y una vez generado ya podemos desarrollar su lógica mediante los archivos que hemos detallado anteriormente.

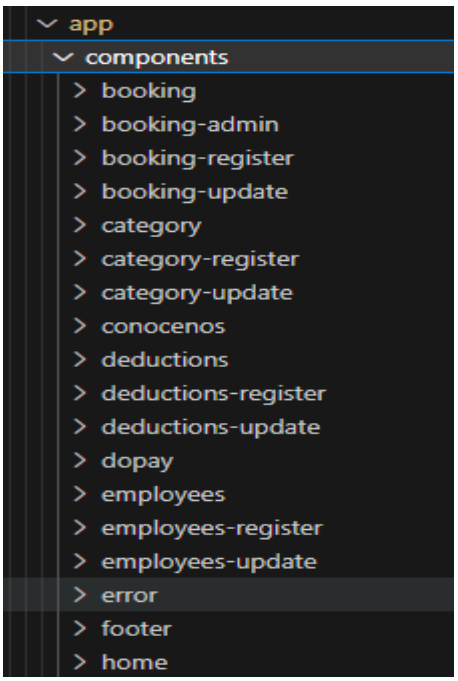


Imagen56: Listado de componentes generados

En la imagen anterior podemos ver un listado de los componentes que hemos generado para el desarrollo del proyecto. Veamos en detalle uno de ellos:

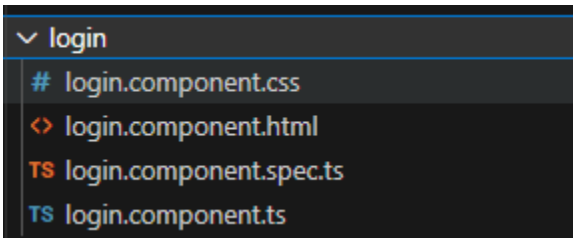


Imagen57: Componente Login

Este componente es el encargado de realizar una petición al backend para identificar a un usuario y devolver el token que lo identifica.

```
<div class="loginContent col-md-12 mt-3">
    <h1>{{page_title}}</h1>
    <p>Identificate en nuestra plataforma</p>

    <form class="col-md-5 ml-0 pl-0" #loginForm='ngForm' (ngSubmit)="onSubmit(loginForm)">

        <div class="form-group">
            <label class="titleWeight" for="user">Usuario: </label>
            <input type="text" name="user" id="user" placeholder="¿Cual quieres que sea tu nombre de usuario?" class="form-control"
                    #user="ngModel" [(ngModel)]="userData.user" required pattern="[a-zA-Z0-9]+" >  <!-- enlazamos el valor de name diciendo que p
                    <small *ngIf="!user.valid && user.touched" class="invalid-feedback d-block"><!-- Si el campo que introducimo no es correcto s
                        El nombre de usuario no es valido
                    </small>
        </div>

        <div class="form-group">
            <label class="titleWeight" for="pass">Contraseña: </label>
            <input type="text" name="pass" id="pass" placeholder="¿Cual quieres que sea tu contraseña?" class="form-control"
                #pass="ngModel" [(ngModel)]="userData.pass" required>
                <small *ngIf="!pass.valid && pass.touched" class="invalid-feedback d-block">
                    Se debe indicar una contraseña por su seguridad
                </small>
        </div>

        <div class=" butonLogin form-group">
            <input type="submit" value="Identificarme" class="btn btn-success" [disabled]="loginForm.invalid"><!--no dejamos enviar-->
        </div>
    </form>
```

Imagen58: Estructura HTML del componente

Como podemos apreciar estamos utilizando el “ngModel” de Angular para almacenar los datos que se reciben a través del formulario en una variable llamada “userData”.

Veamos a continuación qué acciones realizamos con los datos que recibimos, qué acciones tomamos y cómo realizamos la peticiones HTTP al servidor.

```
onSubmit(form: any){
  console.log(this.userData);
  this._userService.signup(this.userData).subscribe(
    response=>{
        //TOKEN
        this.status = 'success';
        this.token = response;
        //indetity
        console.log(this.userData);
        this._userService.signup(this.userData,'true').subscribe(
          response =>{
            this.identity = response;
            console.log(this.identity.token.user);//pq viene dentro de token
            console.log(this.token.token);

            if(this.identity.token.status == "error"){
              console.log("login incorrecto");
              this.identity = null; //vaciamos tambien los parametros
              this.token = null;
            }else{
              //guardamos en el navegador para persitir los datos de sesion
              localStorage.setItem('token', this.token.token);
              localStorage.setItem('identity', JSON.stringify(this.identity.token));
            }
            //redireccion a la pagina principal O administracion dependiendo del logeo
            this._router.navigate(['inicio']);
          },
          error =>{
            this.status = 'error';
            console.log(error);
          }
        );
    },
    error=>{
      this.status = 'error';
      console.log(error);
    }
  );
}
```

Imagen59: Método OnSubmit del componente Login

Cuando el usuario completa el formulario de registro, los datos que este introdujo los tendremos almacenados en la variable UserData que utilizaremos para validar si el usuario existe en la base de datos.

El método signup realiza una petición HTTP a la base de datos y la respuesta será el token qué se genera.

Una vez tenemos el token enviamos otra petición para obtener la identidad del usuario, el nombre, rol, etc. Y cuando tenemos toda la información que necesitamos, guardamos esta información en el navegador para persistir la sesión hasta que el usuario cierre sesión que eliminaremos estas variables del navegador.

"_userService", es un servicio que hemos definido para hacer las peticiones al servidor y abstraer la lógica que conlleva de la lógica del componente. A continuación, se explica la lógica de este servicio.

```
/**
 *  METODO REGISTRO DE USUARIOS
 *  METODO EN EL QUE RECIBIREMOS UN USARIO DEL MODELO USUARIO OVIAMENTE
 *  QUE TRANFORMAREMOS EL JSON PARA ENVIARLO AL BACKEND
 *
 * register(user: user): Observable<any> recibimos un objeto de tipo user, y devolvemos algo,
 */

register(user: user): Observable<any>{
    let json = JSON.stringify(user);
    let params = "json=" + json;

    let headers = new HttpHeaders().set('Content-type', 'application/x-www-form-urlencoded');
    return this._http.post(this.url+'user/register', params, {headers: headers});
}


signup(user:user, gettoken:any = null){
    if(gettoken != null){
        user.getToken = 'true';
    }

    let json = JSON.stringify(user);
    let params = 'json='+json;
    let headers = new HttpHeaders().set('Content-type', 'application/x-www-form-urlencoded');
    return this._http.post(this.url+'user/login', params, {headers: headers});
}
```

Imagen60: UserService

Como vemos en la imagen tenemos dos métodos, el de registrar un usuario y el de iniciar sesión. Ambos métodos necesitan realizar una petición post al backend. Para ello transforman el objeto que reciben a JSON. Crean las cabeceras necesarias y realizan la petición a la URL que definimos anteriormente en el backend.

El resultado de estos métodos en concreto es el token. Sin embargo, hemos definido muchos más métodos que no se detallarán para no extender la memoria más de lo necesario.

5.2.4 creación de rutas Angular

Para la creación de las rutas en Angular utilizaremos el archivo tfg-angular\src\app\app.routing.ts donde cargaremos el componente que vayamos a utilizar en la ruta y crearemos la ruta:

```
import { PayComponent } from './components/pay/pay.component';
import { DopayComponent } from './components/dopay/dopay.component';


// Definir las rutas
const appRoutes: Routes = [
  { path: '', component: HomeComponent }, // localhost:4200/
  { path: 'inicio', component: HomeComponent }, // inicio
  { path: 'login', component: LoginComponent },
  { path: 'logout/:sure', component: LoginComponent}, //lo que haremos
  { path: 'register', component: RegisterComponent },
  { path: 'listUsers', component: ListUsersComponent },
  { path: 'listClients', component: ListClientsComponent },
```

Imagen61: Rutas Angular

5.2.5 Componente principal

Por último, se detallarán cómo se cargan todos los componentes y cómo podemos navegar a través de la aplicación por cada uno de ellos.

Para ello tendremos que situarnos en el archivo principal donde hemos definido y el contenido general de cada una de las páginas que corresponde a los componentes que hemos creado.

```
      <li class="submenu5" *ngIf="identity">
        <a>Finanzas</a>
        <ul>
          <li (click)="disapear()"><a [routerLink]="['/pay']">Pagos</a></li>
          <li (click)="disapear()"><a [routerLink]="['/paymentMethod']">Nóminas</a></li>
          <li (click)="disapear()"><a [routerLink]="['/paymentMethod']">Métodos de pago</a></li>
          <li (click)="disapear()"><a [routerLink]="['/deductions']">Descuentos</a></li>
          <li (click)="disapear()"><a [routerLink]="['/invoices']">Facturas</a></li>
        </ul>
      </li>
      <li (click)="disapear()" *ngIf="identity"><a [routerLink]="['/logout/1']">Salir</a></li>
    </ul>
  </nav>

  <div class="contentMain">

    <router-outlet></router-outlet><!--Muestra el componente asociado a la ruta que estamos mostrando-->


  </div>
  <app-footer></app-footer>


</body>
</html>
```

Imagen62: Componente principal

En esta imagen podemos observar la manera que tenemos en el menú de cambiar la URL, mediante [routerLink]. El div “contentMain” mostrará el contenido del componente que esté asociado a la ruta gracias a la etiqueta <router-outlet>. Es decir, mostrará el HTML aplicando los estilos y lógica que definamos en los archivo correspondientes dentro del componente.

Por último, también cargamos un componente, el componente pie de página o Footer, que mostrará los enlaces a redes sociales, etc.

Con esto terminamos el capítulo 5 de la memoria, es importante destacar que se ha resumido enormemente, no se muestra mucha de la lógica que se ha desarrollado para la aplicación, como cargar la imágenes, formularios de registro, etc.

5.3 Política de privacidad de los datos

Dado que nuestra aplicación recopila datos personales de los usuarios que se registran, debemos redactar unas políticas de privacidad que el usuario deberá aceptar.

Debemos cumplir con la regulaciones en el ámbito de la privacidad de los datos y la protección de la información personal:

1. La GDRP: legislación de la unión europea que entró en vigor en 2008. Establece una serie de normas de como las empresas y organizaciones deben recopilar y proteger los datos de los ciudadanos europeos.
2. La LOPD: legislación española que establece una serie de obligaciones y derechos relacionados con la protección de los datos personales en España.

Para ello en el formulario de registro hemos puesto un pequeño apartado:

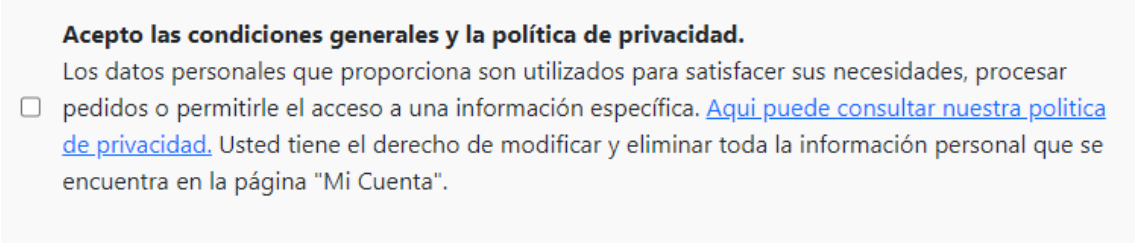


Imagen63: Política de privacidad

Como vemos en la imagen63, en el propio formulario ya nos da información de las políticas de privacidad que el usuario debe aceptar para poder registrarse en la aplicación. Además, si accede a “Aquí puede consultar nuestras políticas de privacidad” podrá encontrar la información mas detallada como vemos en la imagen 64.

Imagen64: Política de privacidad

6 Pruebas

Una de las etapas más importantes en el desarrollo de un proyecto es la parte de pruebas que garantiza que todo funcione correctamente. En este capítulo se detallarán las pruebas que hemos realizado para comprobar el correcto funcionamiento de la aplicación.

6.1 Test ORM

Object-Relation-Mapping (ORM) es una técnica utilizada para mapear objetos desde una base de datos relacional proporcionando una capa de abstracción entre el código de la aplicación y la base de datos, haciendo más fácil la manipulación de los datos.

Para comprobar que el ORM funciona correctamente se ha diseñado un test que, mediante los modelos que definimos en el proyecto, accede a los datos registrados en las tablas de la base de datos definida.

Para ello se ha implementado un controlador que crea instancia de cada modelo realiza las peticiones pertinentes y muestra los datos obtenidos.

```
?>
        <html lang='es'>

        <head>
            <meta charset='UTF-8'>
            <meta http-equiv='X-UA-Compatible' content='IE=edge'>
            <meta name='viewport' content='width=device-width, initial-scale=1.0'>
            <title>TEST ORM MODELS</title>
        </head>

        <body style="margin: 5%; background-color: burlywood; ">
            <h1 style=" text-align: center;">Pruebas modelos con base de datos</h1>

    <?php

        echo "<div style='margin: 5% 0;'><h2> Modelo Producto </h2>";
        $products = product::all();
        foreach ($products as $product) {
            echo "<h4>" . $product->productName . "</h4>";
            echo "<p> Categoria: " . $product->categories->categoryName . "</p>";
            echo "<p> Proveedor: " . $product->suppliers->supplierName . "</p>";
        }
        echo "</div>";


        echo "<div style='margin: 5% 0;'><h2> Modelo Categoria </h2>";
        $categories = category::all();
        foreach ($categories as $category) {
            echo "<h4> categoria: $category->categoryName </h4>";
            foreach ($category->products as $product) {
                echo "<h5> En esta categoria tenemos el producto" . $product->productName . "</h5>";
                echo "<p> cuya categoria como deciamos es: " . $product->categories->categoryName . "</p>";
                echo "<p> cuyo proveedor es: " . $product->suppliers->supplierName . "</p>";
            }
        }
        echo "</div>";


        echo "<div style='margin: 5% 0;'><h2> Modelo Supplier </h2>";
        $suppliers = supplier::all();
        foreach ($suppliers as $supplier) {
            echo "<h4> Nombre: $supplier->supplierName </h4>";
            foreach ($supplier->products as $product) {
                echo "<h5> Producto que proporciona:" . $product->productName . "</h5>";
                echo "<p> cuya categoria es: " . $product->categories->categoryName . "</p>";
                foreach ($product->reparationProducts as $sp) {
                    echo "<p> cuyo id serviceProduct es: " . $sp->id_reparationProducts . ", relacion con prod
                }
            }
```

Imagen65: Fragmento código Test ORM

En la imagen anterior podemos observar un pequeño fragmento del código del Test ORM. Podemos ver que se crea una instancia del modelo product. Realiza una consulta para obtener todos los registros de la tabla product y muestra el nombre del producto, la categoría y el proveedor al que pertenece haciendo uso de la claves foráneas que definimos.

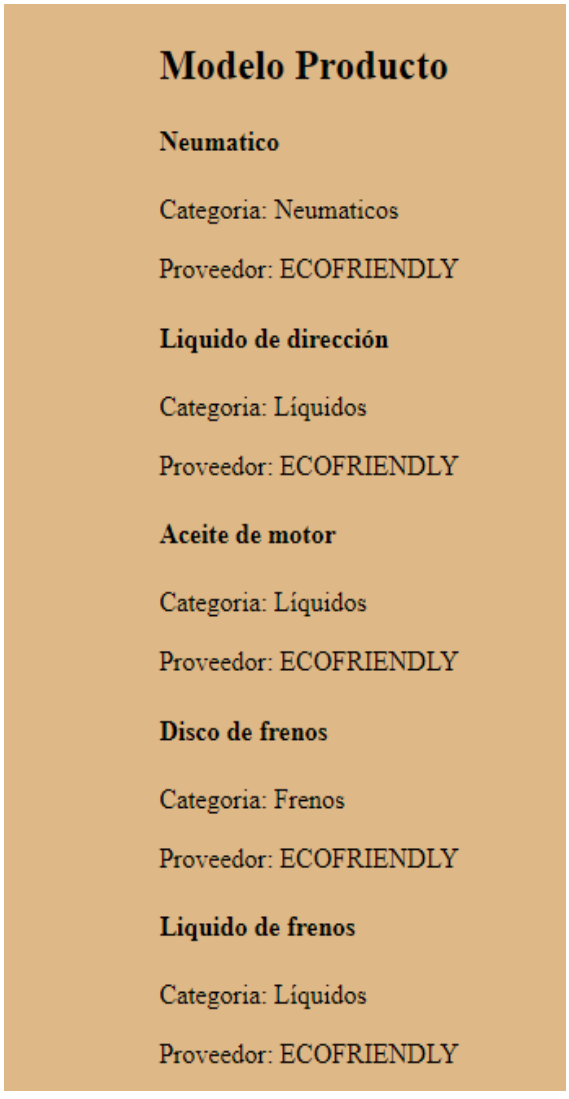


Imagen66: Resultado modelo product Test ORM

En la imagen anterior vemos el resultado del test para el modelo Product. Se muestran los productos que hay registrados en la base de datos, el proveedor y la categoría a la que pertenecen.

Esto se ha realizado para todos los modelos que hemos definido comprobando que todos acceden a la base de datos correctamente.

6.2 Peticiones mediante Postman

Mediante Postman hemos realizado peticiones a todas las URL definidas en el proyecto Laravel para comprobar que los datos que devuelven los controladores son los esperados.

Por ejemplo, comprobemos que el método registrar un nuevo usuario funciona correctamente. Para ello abrimos la aplicación Postman y pasamos los atributos necesarios para completar el registro:

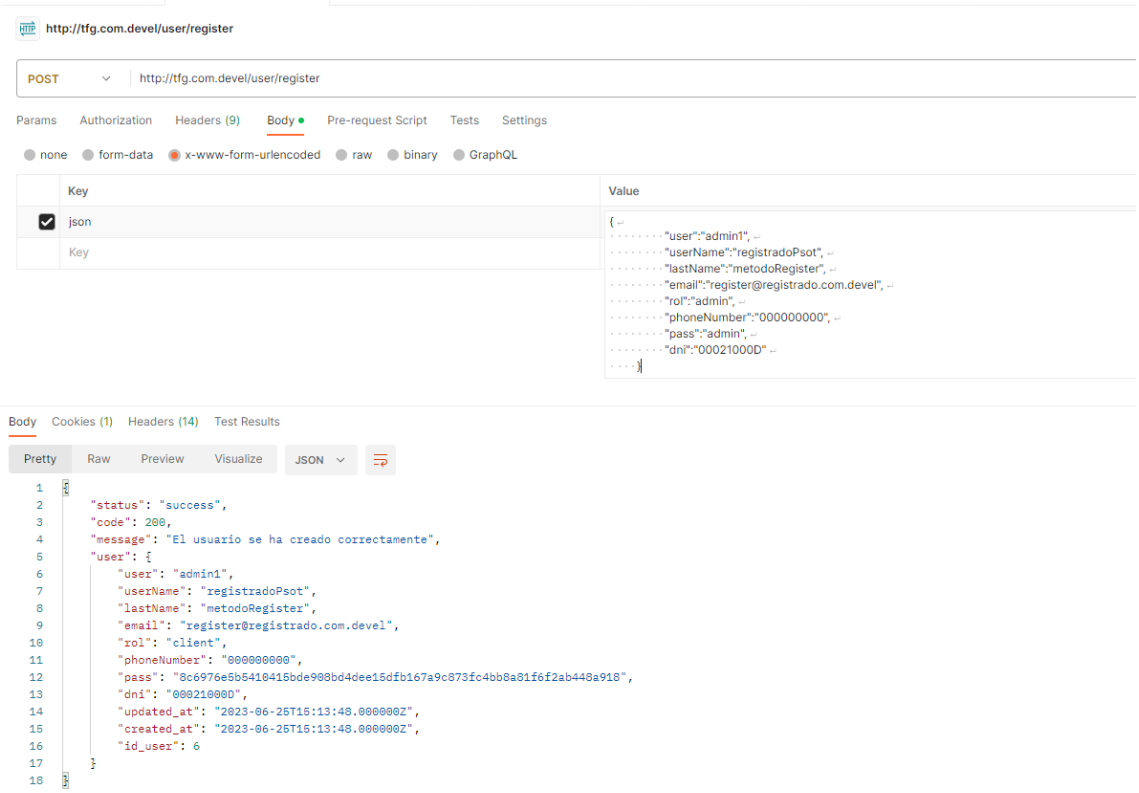


Imagen67: Resultado modelo product Test ORM

Como vemos en la imagen realizamos una petición post a la ruta que definimos en el proyecto Larabel para el controlador de usuarios. Se realizan una petición http pasando en el cuerpo de mensaje un objeto de tipo JSON con los datos necesarios para completar el registro de un nuevo usuario.

La respuesta que nos proporciona el controlador de usuario la podemos ver un la parte inferior de la imagen donde nos devuelve un código 200, un mensaje “el usuario se ha creado correctamente” y el objeto usuario que se ha creado y almacenado en la base de datos.

Este proceso que acabamos de detallar para el método registro del controlador usuarios, se ha realizado para cada uno de los métodos de todos los controladores comprobando que todos funcionan correctamente.

Con estas dos pruebas(6.1 y 6.2) se ha comprobado que el backend tiene el comportamiento esperado.

6.3 Pruebas frontend

Para realizar el diseño de las pruebas del frontend se ha elaborado un plan en el que se realiza una detallada descripción para cada una de las pruebas. Estas pruebas están pensadas para evitar los errores más comunes que pueden suceder en estos casos.

Las pruebas diseñadas tendrán 4 partes importantes:

1. Objetivo: Detalla el objetivo que se debe cumplir cuando se ejecute la prueba.
2. Requisitos: Condiciones que se deben cumplir para realizar la prueba.
3. Pasos: Pasos a seguir.
4. Resultados: Se detalla el resultado esperado tras la ejecución de la prueba.

6.3.1 Ejecución de las pruebas

Tras haber realizado el diseño de las pruebas se ejecutarán. Si se obtiene un resultado diferente al esperado tendremos que realizar una búsqueda del error que lo está causando y posteriormente corregirlo.

La realización de estas pruebas se ha hecho con personas que no conocen el funcionamiento interno de la aplicación y utilizando dispositivos diferentes (tabletas, dispositivos móviles, computadores). De esta manera aseguramos que la vista sea la correcta para los diferentes dispositivos y que además la interfaz sea sencilla y fácil de usar para las personas que prueban la aplicación.

Al finalizar las pruebas se ha pedido a los usuarios que realicen un pequeño cuestionario. Mediante los resultados de este formulario, podemos encontrar fallos que han surgido, puntos de mejora, etc. De esta manera podemos corregir en próximas actualizaciones fallos y problemas que hemos detectado.

Primero se detallará el plan de pruebas que se realizará y en el punto 6.3.4 se mostrarán los resultados obtenidos del cuestionario y se analizarán para ver posibles mejoras.

6.3.2 Plan de pruebas diseñado para el cliente

Tabla02 Prueba registro

Prueba 01	
Objetivo	Registro.
Requisitos	Ninguno.
Pasos	1. Acceder al formulario de registro .
	2. Completar los campos necesarios.
Resultados	El usuario se registra en la base de datos.

Tabla03: Prueba inicio de sesión

Prueba 02	
Objetivo	Inicio de Sesión.
Requisitos	Usuario registrado.
Pasos	1. Acceder al formulario de inicio de sesión.
	2. Completar los campos necesarios.
Resultados	El usuario inicia sesión en la aplicación.

Tabla04: Editar usuario

Prueba 03	
Objetivo	Editar usuario.
Requisitos	Usuario registrado.
Pasos	1. Acceder al formulario de edición de usuario .
	2. Completar los campos necesarios.
Resultados	Los datos del usuario se actualizan en la base de datos.

Tabla05: Añadir vehículo

Prueba 04	
Objetivo	Añadir vehículo.
Requisitos	Usuario registrado.
Pasos	1. Acceder al formulario de registro .
	2. Completar los campos necesarios.
Resultados	El vehículo queda registrado en la base de datos.

Tabla06: Editar vehículo

Prueba 05	
Objetivo	Editar vehículo.
Requisitos	Usuario y vehículo registrado.
Pasos	1. Acceder al formulario de registro .
	2. Completar los campos necesarios.
Resultados	Los datos del vehículo se actualizan en la base de datos.

Tabla07: Añadir reserva

Prueba 06	
Objetivo	Anadir Reserva.
Requisitos	Usuario y vehículo registrado.
Pasos	1. Acceder al calendario de reservas .
	2. Completar los campos necesarios.
Resultados	Se guarda la reserva en la base de datos.

Tabla08: Añadir vehículo de ocasión

Prueba 07	
Objetivo	Añadir vehículo de ocasión.
Requisitos	Usuario y vehículo registrados.
Pasos	1. Acceder al formulario de registro .
	2. Completar los campos necesarios.
Resultados	El vehículo de ocasión es registrado y publicado.

Tabla09: Consultar reparaciones.

Prueba 08	
Objetivo	Consultar reparaciones
Requisitos	Usuario, vehículo y reparación registrados.
Pasos	1. Acceder a las reparaciones.
Resultados	Se listan las reparaciones que el vehículo o los vehículos del usuario tienen o han tenido en curso.

Tabla10: Añadir método de pago

Prueba 09	
Objetivo	Añadir método de pago
Requisitos	Usuario registrado.
Pasos	1. Acceder al formulario de registro .
	2. Completar los campos necesarios.
Resultados	El método de pago se registra en la base de datos.

Tabla11: Editar método de pago

Prueba 10	
Objetivo	Editar método de pago.
Requisitos	Usuario y método de pago registrado.
Pasos	1. Acceder al formulario de registro .
	2. Completar los campos necesarios.
Resultados	El método de pago es actualizado en la base de datos.

Tabla12: Consultar facturas

Prueba 11	
Objetivo	Consultar facturas.
Requisitos	Usuario, vehículo, reparación y factura registrada.
Pasos	1. Acceder al listado de facturas.
Resultados	El usuario se registra en la base de datos

Tabla12: Pagar factura

Prueba 12	
Objetivo	Pagar factura
Requisitos	Usuario, vehículo, reparación y factura registrada.
Pasos	1. Acceder al formulario de registro
	2. Completar los campos necesarios
Resultados	El estado de la factura cambia a “pagado”

6.3.3 Plan de pruebas diseñado para el administrador

Tabla013: CRUD usuarios

Prueba 01	
Objetivo	CRUD usuarios
Requisitos	Usuarios registrados (para lectura, actualización y eliminación).
Pasos	1. Acceder al panel de administración de usuarios.
	2. Completar los campos necesarios.
Resultados	Se realiza la creación, lectura, actualización y eliminación de un usuario.

Tabla014: CRUD vehículos.

Prueba 02	
Objetivo	CRUD vehículos.
Requisitos	vehículos registrados (para lectura, actualización y eliminación).
Pasos	1. Acceder al panel de administración de vehículos.
	2. Completar los campos necesarios.
Resultados	Se realiza la creación, lectura, actualización y eliminación de un vehículo.

Tabla015: CRUD vehículos de ocasión.

Prueba 03	
Objetivo	CRUD vehículos de ocasión.
Requisitos	Vehículos de ocasión registrados (para lectura, actualización y eliminación).
Pasos	1. Acceder al panel de administración de vehículos de ocasión.
	2. Completar los campos necesarios.
Resultados	Se realiza la creación, lectura, actualización y eliminación de un vehículo de ocasión.

Tabla013: CRUD reservas.

Prueba 04	
Objetivo	CRUD Reservas.
Requisitos	Reservas registrados (para lectura, actualización y eliminación).
Pasos	1. Acceder al panel de administración de reservas.
	2. Completar los campos necesarios.
Resultados	Se realiza la creación, lectura, actualización y eliminación de una reserva.

Tabla014: CRUD reservas.

Prueba 05	
Objetivo	CRUD Reparaciones.
Requisitos	Reparaciones registrados (para lectura, actualización y eliminación).
Pasos	1. Acceder al panel de administración de reparaciones.
	2. Completar los campos necesarios.
Resultados	Se realiza la creación, lectura, actualización y eliminación de una reparaciones.

Tabla015: CRUD servicios.

Prueba 06	
Objetivo	CRUD Servicios.
Requisitos	Servicios registrados (para lectura, actualización y eliminación).
Pasos	1. Acceder al panel de administración de servicios.
	2. Completar los campos necesarios.
Resultados	Se realiza la creación, lectura, actualización y eliminación de un servicio.

Tabla16: CRUD productos.

Prueba 07	
Objetivo	CRUD productos.
Requisitos	Productos registrados (para lectura, actualización y eliminación).
Pasos	1. Acceder al panel de administración de productos.
	2. Completar los campos necesarios.
Resultados	Se realiza la creación, lectura, actualización y eliminación de un producto.

Tabla16: CRUD categorías.

Prueba 08	
Objetivo	CRUD categorías.
Requisitos	Categorías registrados (para lectura, actualización y eliminación).
Pasos	1. Acceder al panel de administración de categorías.
	2. Completar los campos necesarios.
Resultados	Se realiza la creación, lectura, actualización y eliminación de una categoría.

Tabla17: CRUD Proveedores.

Prueba 09	
Objetivo	CRUD proveedores.
Requisitos	Proveedores registrados (para lectura, actualización y eliminación).
Pasos	1. Acceder al panel de administración de proveedores.
	2. Completar los campos necesarios.
Resultados	Se realiza la creación, lectura, actualización y eliminación de un proveedor.

Tabla17: CRUD métodos de pago.

Prueba 10	
Objetivo	CRUD métodos de pago.
Requisitos	Proveedores registrados (para lectura, actualización y eliminación).
Pasos	1. Acceder al panel de administración de métodos de pago.
	2. Completar los campos necesarios.
Resultados	Se realiza la creación, lectura, actualización y eliminación de un método de pago.

Tabla18: CRUD facturas.

Prueba 11	
Objetivo	CRUD facturas.
Requisitos	Facturas registrados (para lectura, actualización y eliminación).
Pasos	1. Acceder al panel de administración de facturas.
	2. Completar los campos necesarios.
Resultados	Se realiza la creación, lectura, actualización y eliminación de una factura.

Tabla19: CRUD descuentos.

Prueba 12	
Objetivo	CRUD descuentos.
Requisitos	Descuentos registrados (para lectura, actualización y eliminación).
Pasos	1. Acceder al panel de administración de descuentos.
	2. Completar los campos necesarios.
Resultados	Se realiza la creación, lectura, actualización y eliminación de un descuento.

Tabla20: CRUD pagos.

Prueba 13	
Objetivo	CRUD pagos.
Requisitos	Pagos registrados (para lectura, actualización y eliminación).
Pasos	1. Acceder al panel de administración de pagos.
	2. Completar los campos necesarios.
Resultados	Se realiza la creación, lectura, actualización y eliminación de un pago.

Tabla21: CRUD Nóminas.

Prueba 14	
Objetivo	CRUD nóminas.
Requisitos	Nóminas registrados (para lectura, actualización y eliminación).
Pasos	1. Acceder al panel de administración de nóminas.
	2. Completar los campos necesarios.
Resultados	Se realiza la creación, lectura, actualización y eliminación de una nómina.

6.3.4 Resultados de las pruebas

Los resultados obtenidos generalmente son buenos. Veamos los resultados del cuestionario para ver qué opinión tienen los usuarios y que aspecto podemos mejorar.

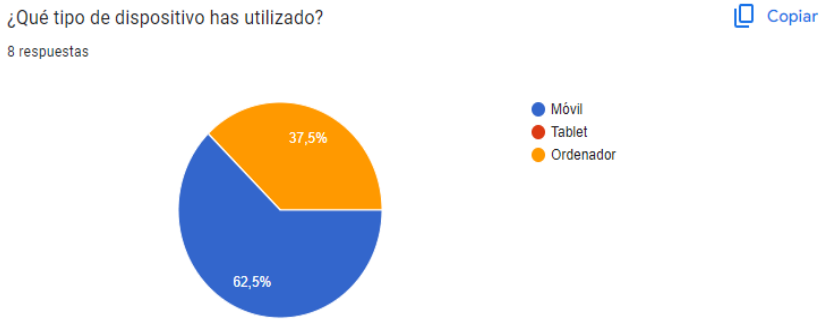


Imagen68: Dispositivos utilizados para el cuestionario

Una de las preguntas que se realizaban en el cuestionario tiene la finalidad de saber que dispositivos se ha usado para realizar las pruebas como vemos en la imagen66. Esto nos ayudará a determinar qué cambios tenemos que realizar y para que resoluciones aplicarlo.

En el cuestionario contiene las siguientes preguntas:

1. ¿Pudiste acceder al formulario de registro correctamente?
2. ¿El registro se completó correctamente?
3. ¿Pudiste iniciar sesión con la cuenta que creaste?
4. ¿Pudiste añadir un vehículo?
5. ¿Pudiste realizar una reserva?
6. Aunque la reparación era ficticia ¿Pudiste ver los datos y el estado de la reparación correctamente?
7. Aunque la reparación era ficticia ¿Pudiste ver e imprimir los datos de la factura asociada?

Preguntas que hacen referencia a la mayoría de los servicios que hemos implementado y a la que todas las respuestas han sido un rotundo sí.

Sin embargo, obviamente la aplicación no es perfecta y los usuarios han dejado algunos comentarios con respecto a modificaciones que deberíamos realizar:

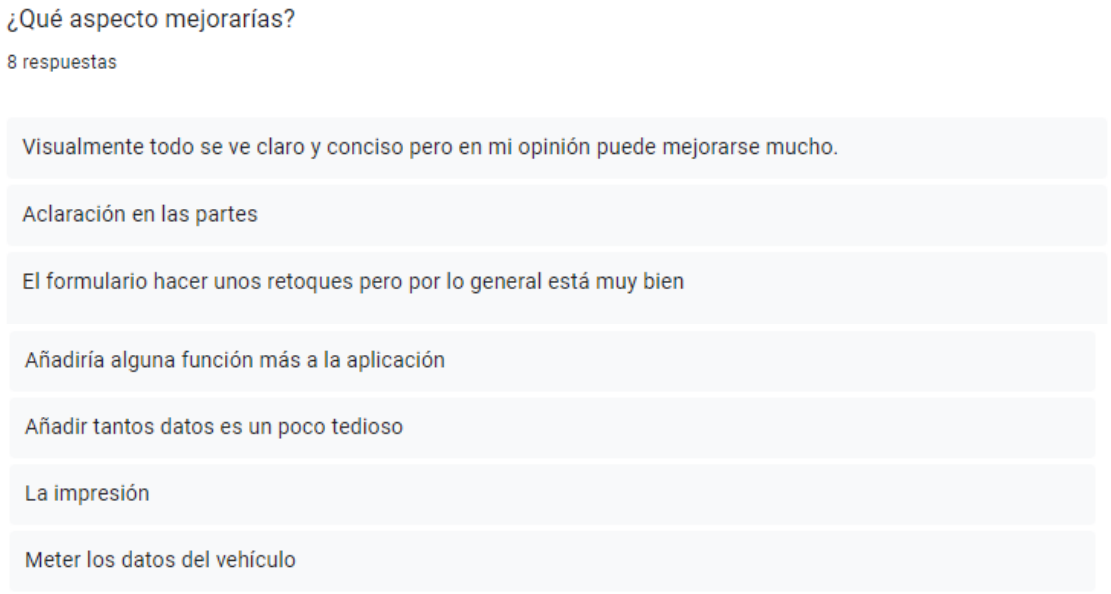


Imagen69: Aspectos a mejorar según los usuarios

Analizando estas respuesta podemos extraer varios cambios que deberíamos realizar:

1. Mejoras generales en los estilos de la aplicación.
2. Detallar mejor los datos que se están pidiendo dado que varios usuarios se quejan o de que hay muchos datos a introducir o que no queda claro que datos se está pidiendo.

3. Mejorar la impresión de la factura, los datos se muestran en simples tablas, deberíamos mejorar la visualización.
4. Añadir alguna funcionalidad más para los clientes.

Por último, también se ha pedido al usuario que califique la aplicación como podemos ver en la imagen 68.

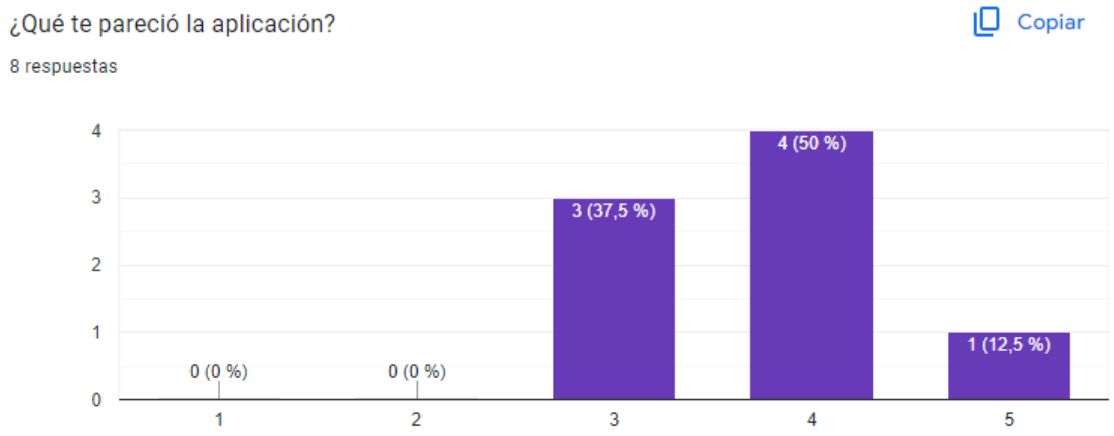


Imagen70: Dispositivos utilizados para el cuestionario

Los resultado son buenos, pero no son los mejores, deberemos seguir mejorando la aplicación para ofrecer un mejor servicio.

7 Resultados

En este capítulo de la memoria se mostrará una serie de capturas de las vistas más importantes, donde se podrá observar cual es el resultado visual de la aplicación desarrollada.

7.1 menú vista computador

Imagen71: Menú computador.

En la imagen 71, podemos ver una captura de pantalla del menú que se muestra para resoluciones de pantalla de 1200 píxeles o superior. En este caso es el menú que se le mostraría al cliente, tiene un apartado de inicio se muestra información sobre el taller y en la parte de conócenos se concreta más indicando incluso la ubicación y el horario del taller.

Luego tendríamos el apartado de operación donde el usuario puede realizar una reserva, añadir un vehículo, un vehículo de ocasión o consultar las reparaciones que se están realizando en alguno de sus vehículos.

La parte de gestión podría consultar las facturas los pagos y los métodos de pago que tiene registrados.

El apartado “client”, es el nombre de usuario, en este caso Client, y es donde el usuario podrá modificar datos de su cuenta como el correo, contraseña, nombres, etc.

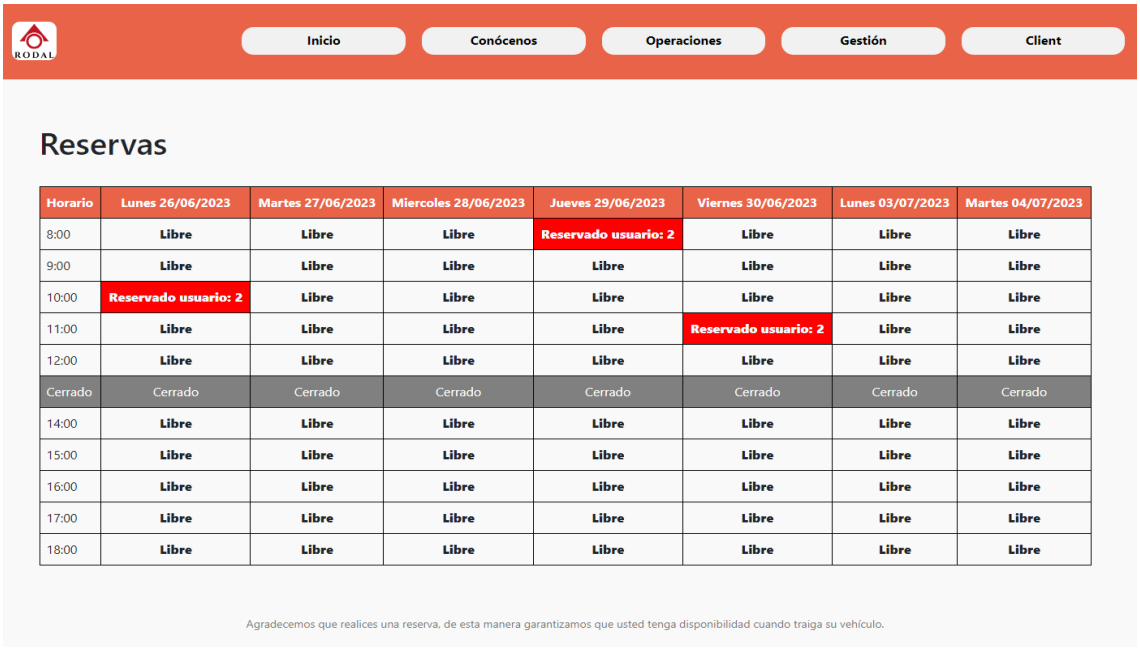
Por último, el apartado de salir, cierra la sesión en la aplicación.

7.2 menú vista móvil

Imagen72: Menú móvil.

La imagen 72, muestra el menú que hemos explicado en el apartado anterior, pero esta vez para dispositivos con una resolución menor a 1200 píxeles.

7.3 Reservas vista computador



Horario	Lunes 26/06/2023	Martes 27/06/2023	Miercoles 28/06/2023	Jueves 29/06/2023	Viernes 30/06/2023	Lunes 03/07/2023	Martes 04/07/2023
8:00	Libre	Libre	Libre	Reservado usuario: 2	Libre	Libre	Libre
9:00	Libre	Libre	Libre	Libre	Libre	Libre	Libre
10:00	Reservado usuario: 2	Libre	Libre	Libre	Libre	Libre	Libre
11:00	Libre	Libre	Libre	Libre	Reservado usuario: 2	Libre	Libre
12:00	Libre	Libre	Libre	Libre	Libre	Libre	Libre
Cerrado	Cerrado	Cerrado	Cerrado	Cerrado	Cerrado	Cerrado	Cerrado
14:00	Libre	Libre	Libre	Libre	Libre	Libre	Libre
15:00	Libre	Libre	Libre	Libre	Libre	Libre	Libre
16:00	Libre	Libre	Libre	Libre	Libre	Libre	Libre
17:00	Libre	Libre	Libre	Libre	Libre	Libre	Libre
18:00	Libre	Libre	Libre	Libre	Libre	Libre	Libre

Imagen73: Reservas computador.

En la imagen 73, la vista para realizar las reservas. Como vemos el horario comprendido es desde el día actual hasta una semana después.

El usuario con un simple clic puede registrar un vehículo para un día y una hora en concreto.

7.4 Reservas vista móvil

Horario	Lunes 26/06/2023	Martes 27/06/2023	Miercoles 28/06/2023
8:00	Libre	Libre	Libre
9:00	Libre	Libre	Libre
10:00	Reservado usuario: 2	Libre	Libre
11:00	Libre	Libre	Libre
12:00	Libre	Libre	Libre
Cerrado	Cerrado	Cerrado	Cerrado
14:00	Libre	Libre	Libre
15:00	Libre	Libre	Libre
16:00	Libre	Libre	Libre
17:00	Libre	Libre	Libre
18:00	Libre	Libre	Libre

Imagen74: Reservas computador.

En la imagen74, es la vista para la reservas, pero para dispositivos con una resolución menor a 1200 píxeles.

7.5 Vehículos de ocasión vista computador

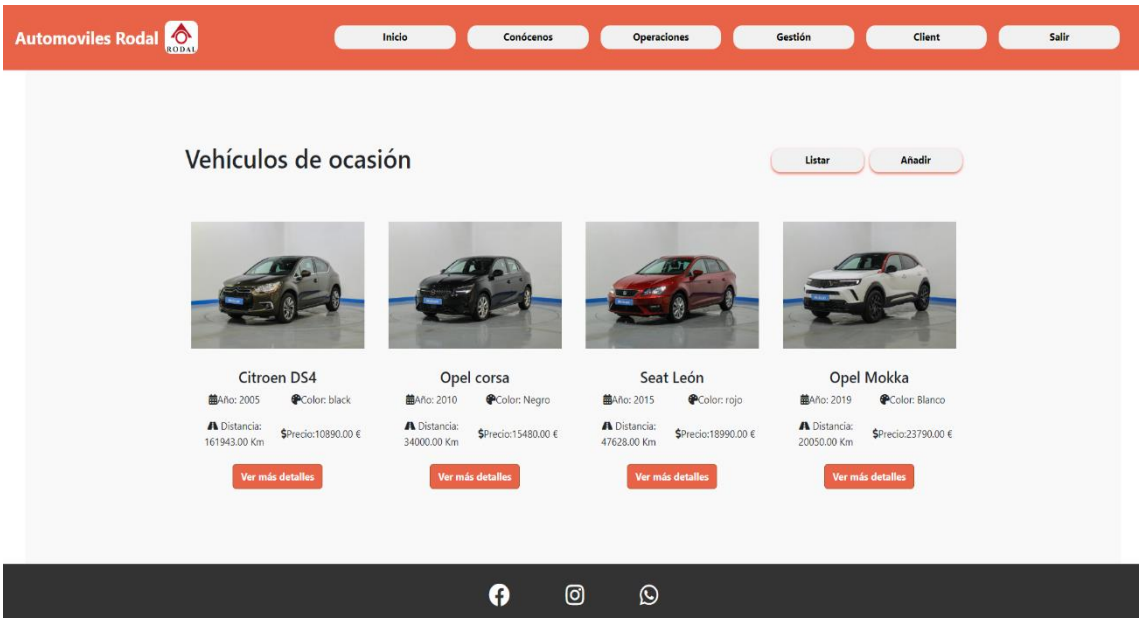


Imagen75: Vehículos de ocasión.

En la imagen75, se muestra la vista de los vehículos de ocasión que usuarios han creado. Vemos una lista con todos los vehículos que están en venta, y dos botones:

1. Listar: lista los vehículos de ocasión que el usuario tiene registrado, en caso de no tener ninguno no aparece el botón.
2. Añadir: se redirige al usuario aun formulario donde se le pedirá que añada los datos necesarios para anunciar el vehículo de ocasión.

7.6 Vehículos de ocasión vista móvil

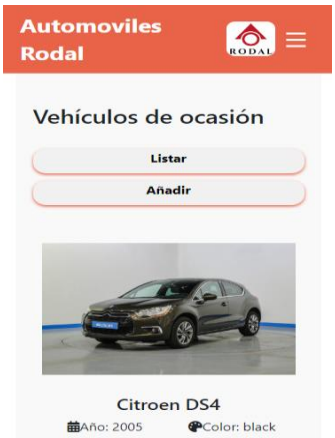


Imagen76: Vehículos de ocasión móvil.

La imagen 76 muestra cómo se vería este apartado de la web en el caso de visitarlo a traves de un dispositivo con una resolución menor a 1200 píxeles.

7.7 Formulario de registro vista computador

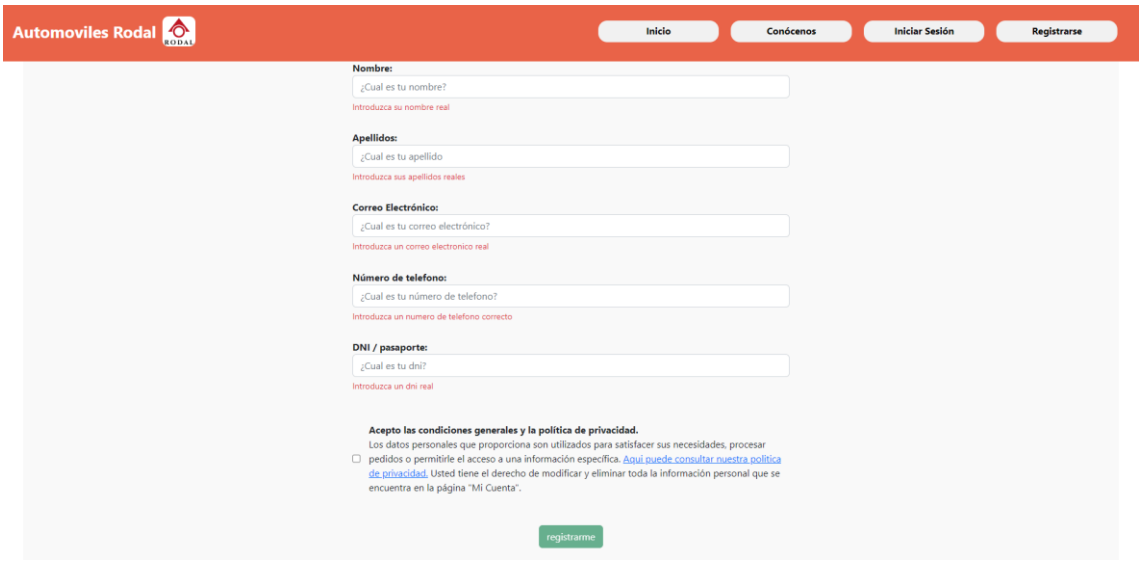


Imagen77: Registro vista computador.

En la imagen77 se puede observar los datos necesarios para completar el registro de una nueva cuenta. Se pide el correo electrónico, el nombre de usuario, etc.

También se pide al usuario si acepta las políticas de privacidad de la aplicación.

7.8 Formulario de registro vista móvil

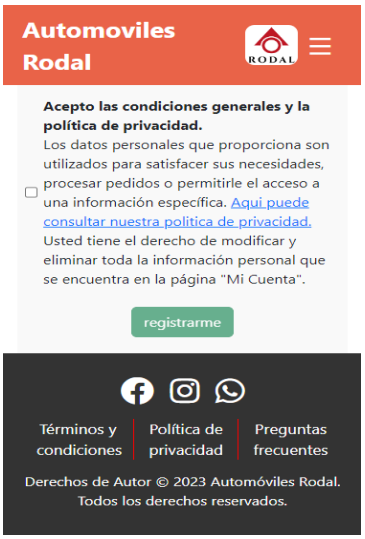


Imagen78: Registro vista móvil.

La imagen 78 muestra el mismo formulario que el apartado anterior, pero en esta ocasión para dispositivos con una resolución menor a 1200 píxeles.

7.9 Facturas de las reparaciones

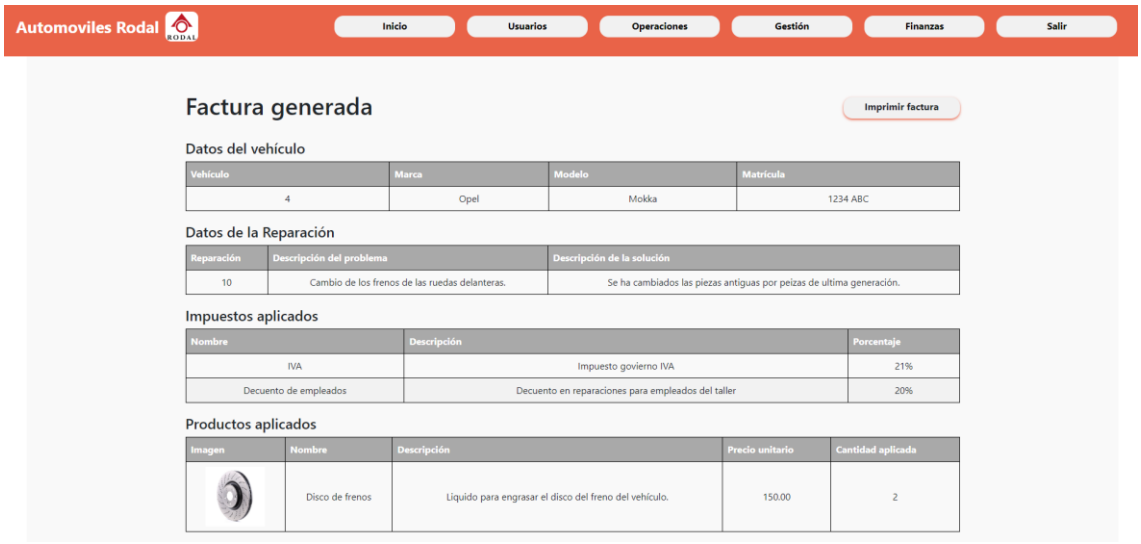


Imagen79: Factura generada a partir de una reparación.

En la imagen 79 podemos ver una factura generada a partir de una reparación. En la captura no se muestran todos los datos que se generan, pero podemos apreciar que da información sobre el vehículo reparado, los impuestos aplicados, los productos usados, etc.

7.10 Facturas de las reparaciones impresión

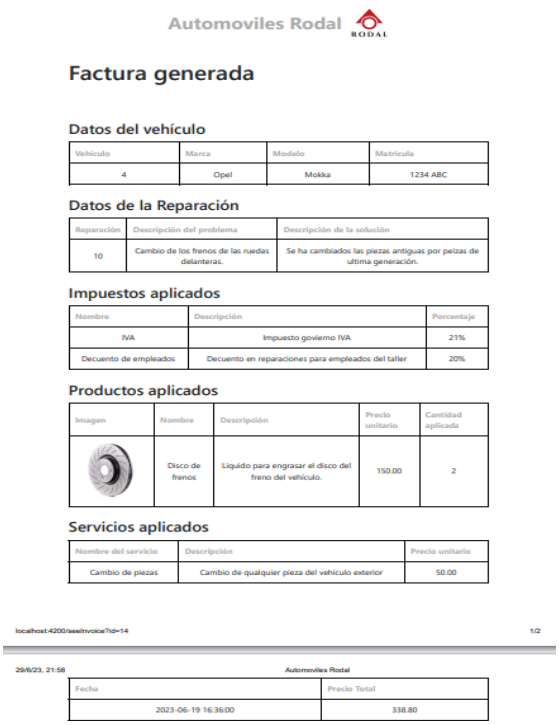


Imagen80: Factura generada a partir de una reparación.

En la imagen 80, podemos ver la factura que se genera completamente. Esta captura muestra la vista de impresión que se genera cuando pulsamos en el botón imprimir.

Se muestran todos los datos, productos usados, servicios aplicados, etc.

Y finalmente se muestra el precio a abonar, que se ha calculado automáticamente en base a los precios, impuestos y descuentos que se han aplicado a la reparación.

Para terminar con este apartado, es importante destacar que faltan muchas capturas para mostrar completamente la aplicación que hemos desarrollado. Sin embargo, con las capturas que hemos puesto en este capítulo de la memoria el lector puede hacer una idea general de cómo ha quedado la aplicación, además se muestran los servicios más importantes que se han implementado.

8 Conclusiones

En este capítulo se detallan la conclusiones que hemos obtenido después de terminar el desarrollo de la aplicación.

Primero de todo, hay que destacar que hemos conseguido el objetivo principal del proyecto, el desarrollo de una aplicación web dinámica para la administración y gestión de talleres automovilísticos, que permite administrar todo el proceso de reparación y generación de facturas de un taller.

Objetivos cumplidos:

1. Creación de una interfaz para registro e inicio de sesión.
2. Creación de una interfaz para añadir vehículos y vehículos de ocasión.
3. Creación de una interfaz para realizar reservas.
4. Creación de una interfaz administrar su cuenta, así como sus métodos de pago.
5. Creación de una interfaz para consultar las reparaciones.
6. Creación de una interfaz para consultar las facturas e imprimirlas si lo desean.
7. Creación de una interfaz para pagar el importe de las facturas.

Con lo que respecta al administrador, hemos creado una serie de paneles de administración para que pueda gestionar todos los métodos CRUD de cada uno de los objetivos que expusimos en el capítulo primero de la memoria.

Para cumplir con ello se ha conseguido el objetivo del desarrollo de un backend totalmente funcional, que proporcione los servicios para administrar todo. Crear la API ha sido un proceso complejo con muchas horas de trabajo. Pero el resultado obtenido es un código que ofrece una buena lógica, rápido y eficiente.

El mayor de los retos ha sido corregir todos los errores que nos han surgido durante el proceso de desarrollo, pero que finalmente hemos podido solventar.

Por último, para cumplir con el objetivo de desarrollar una aplicación que pudiese ser utilizada desde cualquier dispositivo, se ha seguido la metodología de trabajo y diseñar la aplicación en primer lugar para dispositivos móviles y ampliar este diseño para resoluciones más grandes.

8.1 Relación del trabajo desarrollado con los estudios cursados

El desarrollo de este proyecto no ha permitido adquirir conocimiento que sin duda nos serán útiles en el mercado laboral. Se han seguido las bases de programación adquiridas al cursar el Grado de Ingeniería Informática en la Universidad Politécnica de Valencia.

Las asignaturas que más han influido en el desarrollo de este proyecto:

1. Ingeniería del software
2. Gestión de proyectos
3. Redes de computadores

9 Anexo

9.1 Objetivos de desarrollo sostenible

Tabla22: Objetivos de desarrollo sostenible (ODS)

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**				**X**
ODS 3. **Salud y bienestar.**				**X**
ODS 4. **Educación de calidad.**				**X**
ODS 5. **Igualdad de género.**				**X**
ODS 6. **Agua limpia y saneamiento.**				**X**
ODS 7. **Energía asequible y no contaminante.**				**X**
ODS 8. **Trabajo decente y crecimiento económico.**		**X**		
ODS 9. **Industria, innovación e infraestructuras.**		**X**		
ODS 10. **Reducción de las desigualdades**.				**X**
ODS 11. **Ciudades y comunidades sostenibles**.				**X**
ODS 12. **Producción y consumo responsables.**	**X**			
ODS 13. **Acción por el clima.**				**X**
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**				**X**
ODS 17. **Alianzas para lograr objetivos.**				**X**

La aplicación que hemos desarrollado se centra en la gestión de talleres automovilísticos, buscando mejorar la eficiencia de estos negocios. En la tabla22 (Objetivos de desarrollo sostenible), se puede observar el grado de relación de los objetivo con nuestro proyecto. Sin embargo, a continuación, se realizará una análisis más en profundidad con los objetivos que hemos marcado anteriormente en la tabla.

En primer lugar, el objetivo de producción y consumo responsable (ODS12) está directamente relacionado con la aplicación que hemos desarrollado.

Como bien explica el artículo[23], las activades que se llevan a cabo en los talleres, tienen un impacto negativo en el medio ambiente. En estas actividades utilizan recursos naturales(agua, energía, combustible) y no de una manera muy eficiente. Además, muchos de materiales que tienen almacenados tienen un mala gestión, se caducan y se desperdician.

Creemos que nuestra aplicación tendrá un impacto positivo en el medio ambiente ya que mediante su uso mejoraremos la gestión de los materiales que suelen almacenarse en los talleres, sabremos en todo momento la ubicación de estos y su fecha de caducidad permitiendo que no se desperdicien. En resumen, ayudaremos a utilizar estos recursos de una manera óptima y eficiente.

En segundo lugar, debido a que nuestra aplicación busca mejorar la gestión de talleres mediante el uso de la aplicación desarrollada, consideramos que el objetivo de Desarrollo Sostenible innovación e infraestructura(OD9) se relaciona con nuestro proyecto. Ayudaremos a la gestión de los recursos, planificación de citas y comunicación con los clientes. En este aspecto, buscamos impulsar la competitividad y productividad en el sector, ofreciendo así una contribución al crecimiento económico y al desarrollo sostenible en el sector.

En tercer lugar, al mejorar la administración de los talleres automovilísticos, consideramos que el objetivo de trabajo decente y crecimiento económico(ODS8) se relaciona en menor medida con nuestro proyecto. Nuestra aplicación permite una gestión más eficiente de los talleres, lo que provoca una reducción de la carga de trabajo. Al mejorar la productividad en los talleres, fomentamos el crecimiento económico del negocio y creamos oportunidades de empleo de mayor calidad.

Para concluir con este apartado, la aplicación que hemos desarrollado se centra en mejorar la gestión de talleres automovilísticos, y su impacto en los objetivos de desarrollo sostenible creemos que es significativo. Como hemos detallado anteriormente, nuestra aplicación se relaciona con objetivo de producción y consumo responsable (ODS12), con el objetivo de Innovación e Infraestructura (ODS9) y con el objetivo de Trabajo decente y crecimiento económico (ODS8). Estos dos últimos en menor medida.

Sin embargo, es importante resaltar que, aunque solo nos hemos centrado en detallar estos tres objetivos de desarrollo sostenible debido a que creemos que son en los que más influye nuestra aplicación, perfectamente podría tener impactos positivos en Otros objetivos, como en el objetivo reducción de las desigualdades (ODS10) ya que mejoráramos las oportunidades de trabajo en el sector o en la igualdad de género (ODS5).

9.2 Agradecimientos

Para terminar con la memoria, me gustaría agradecer a mi tutor de prácticas y a la Universidad Politécnica de Valencia, por la ayuda que me han ofrecido.

En primer lugar, a mi tutor de prácticas que ha estado presente durante todo el desarrollo de la aplicación y que me ha ofrecido su ayuda siempre que lo he necesitado. Su compromiso y dedicación han sido inspiradores y me han motivado a dar lo mejor de mí en el desarrollo del proyecto y a seguir formándome en el futuro.

Por otro lado, también quiero expresar mis agradecimiento a la Universidad Politécnica de Valencia por darme la oportunidad de realizar el trabajo de final de grado.

10 Referencias

[1] Laura Fernández Durán, "La digitalización, un reto para las pymes", 3 de mayo de 2023. [En línea] Disponible en: https://economia3.com/2023/05/03/573417-la-digitalizacion-un-reto-para-las-pymes/ [Último acceso: 21 de junio de 2023].

[2] Immune Technology Institute, "Mobile first design, ¿Cómo repercute en la experiencia del usuario?", 9 de septiembre de 2022. [En línea] Disponible en: https://immune.institute/blog/mobile-first-design/ [Último acceso: 20 de junio de 2023].

[3] Vicente Serrano, "¿Por qué automatizar los procesos de gestión?", 16 de junio de 2022. [En línea] Disponible en: https://www.datadec.es/blog/por-que-automatizar-los-procesos-de-gestion/ [Último acceso: 15 de junio de 2023].

[4] My gestión, "*Software de Gestión y contabilidad Online*", [En línea] Disponible en: https://www.mygestion.com/landing.jsp/ [Último acceso: 28 de junio de 2023].

[5] Mechanic Advisor, "aplicación web", fundada en 2006. [En línea] Disponible en: https://www.mechanicadvisor.com/ [Último acceso: 28 de junio de 2023].

[6] YourMechanic, "aplicación web", fundada en 2012. [En línea] Disponible en: https://www.yourmechanic.com/ [Último acceso: 23 de junio de 2023].

[7] Auto Repair Bill, "Software de facturación, reserva y presupuestos para mecánicos y talleres de reparación de automóviles". [En línea] Disponible en: https://www.autorepairbill.com/ [Último acceso: 10 de junio de 2023].

[8] Repair Pal, "Aplicación web con base en San Francisco que ofrece servicios en el sector automovilístico". [En línea] Disponible en: https://repairpal.com/ [Último acceso: 4 de junio de 2023].

[9] Laravel, "Framework de desarrollo en php". [En línea] Disponible en: https://laravel.com/ [Último acceso: 3 de marzo de 2023].

[10] Angular, "Framework de desarrollo web en typeScript". [En línea] Disponible en: https://laravel.com/ [Último acceso: 3 de marzo de 2023].

[11] Laravel, "Framework de desarrollo en php". [En línea] Disponible en: https://angular.io/ [Último acceso: 20 de marzo de 2023].

[12] Balsamiq Wireframes, "herramienta de diseño de interfaces". [En línea] Disponible en: https://balsamiq.com/ [Último acceso: 25 de marzo de 2023].

[13] Visual Studio Code, "Entorno de desarrollo". [En línea] Disponible en: https://code.visualstudio.com/ [Último acceso: 25 de marzo de 2023].

[14] Git, "Sistema de control de versiones". [En línea] Disponible en: https://git-scm.com/ [Último acceso: 10 de marzo de 2023].

[15] Gestor de Base de datos MySQL, “Sistema de gestión de base de datos”. [En línea] Disponible en: https://www.mysql.com/ [Último acceso: 21 de marzo de 2023].

[16] Microsoft, “empresa Microsoft fundada el 4 de abril de 1975”. [En línea] Disponible en: https://www.microsoft.com/ [Último acceso: 5 de marzo de 2023].

[17] Lenguaje TypeScript, “Lenguaje de programación JavaScript tipado”. [En línea] Disponible en: https://es.wikipedia.org/wiki/TypeScript [Último acceso: 22 de marzo de 2023].

[18] PHP, “Lenguaje de programación se servidores php”. [En línea] Disponible en: https://www.php.net/manual/es/intro-whatis.php [Último acceso: 22 de marzo de 2023].

[19] SQL, “Lenguaje de consultas SQL”. [En línea] Disponible en: https://es.wikipedia.org/wiki/SQL [Último acceso: 10 de marzo de 2023].

[20] NPM, “administrados de paquetes”. [En línea] Disponible en: https://www.npmjs.com/ [Último acceso: 12 de marzo de 2023].

[21] MAMP, “servidor de pruebas local”. [En línea] Disponible en: https://www.mamp.info/en/windows/ [Último acceso: 10 de marzo de 2023].

[22] POSTMAN, “Aplicación para realizar pruebas y peticiones”. [En línea] Disponible en https://www.postman.com/ [Último acceso: 15 de marzo de 2023].

[23] Artículo de Auto Crash, “La deuda ambiental de los talleres con el planeta”. [En línea] Disponible en *https://www.revistaautocrash.com/la-deuda-ambiental-de-los-talleres-con-el-planeta/* [Último acceso: 20 de marzo de 2023].