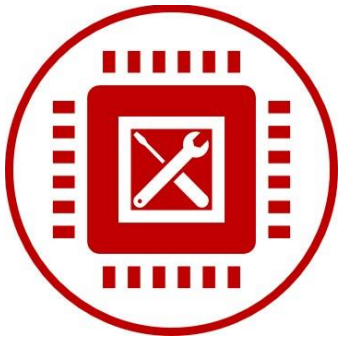

Automatic Workshop

Programa de doble titulación: Grado en Ingeniería Informática y Grado en Administración y Dirección de Empresas

Presentado por: Daniel Pérez Peña

DNI: 54137000Z

Tutorizado por: José Juan Hernández Cabrera y Jorge Marín Rodríguez Díaz

Las Palmas de Gran Canaria, a 10 de junio de 2021

AGRADECIMIENTOS

A Jorge Marín Rodríguez Díaz, por haberme ayudado en todo momento en el desarrollo de este trabajo siempre que lo he necesitado.

A José Juan Hernández Cabrera, por expandir mi idea inicial del proyecto.

A José Daniel Hernández Sosa, por ayudarme con las dudas que he tenido con plazos y documentación durante todo el desarrollo del proyecto.

A mis compañeros de clase, por la ayuda mutua que nos hemos prestado durante todo el período académico.

A mi familia, pareja y amigos, por el apoyo que me han dado.

RESUMEN

En los últimos años el software nos ha facilitado la realización de tareas cotidianas mediante las distintas aplicaciones que se han lanzado al mercado, sin embargo, todavía no se ha generalizado el uso de algún producto que nos ayude a llevar al día el mantenimiento de nuestros vehículos de forma cómoda, una tarea que se caracteriza por ser tan importante como tediosa.

El objetivo de este trabajo es desarrollar una aplicación web que permita alcanzar esta meta, así como crear un modelo de negocio que lo haga viable en el mercado a largo plazo. Dicho modelo será planteado desde la perspectiva de las áreas funcionales que toda empresa posee (Marketing, Finanzas, Producción, Recursos Humanos y Alta dirección).

PALABRAS CLAVE

Desarrollo web, modelo de negocio, automoción, mantenimiento, taller, vehículos, empresa emergente, front end, back end, maquetación.

ABSTRACT

In the past Years, software applications have made easier our everyday tasks. Despite of that, we do not have already a widely used digital product for easily managing a vehicle maintenance, a task which is characterized for being both important and tedious.

The main goal of this project is the development of a web application with which this problem could be solved and creating a business model for transforming the resulting application into a successful digital product, considering the different perspectives of the functional areas in which all companies can be split (Marketing, Finances, Operations, Human Resources and Management).

KEYWORDS

Web development, business plan, automotive, maintenance, workshop, vehicle, start-up, front end, back end, web layout.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. GUÍA DE LECTURA

Apartados comunes:

Motivación (1.2.)

Objetivos (1.3.) - Solo hasta la primera lista de puntos, inclusive.

Aportaciones (1.5.)

Requisitos (1.6.)

Conclusiones y trabajos futuros (5.)

Bibliografía (6.)

Apartados estrictamente relacionados con el grado en Ingeniería Informática:

Objetivos (1.3.) - Todo lo que venga después de la primera lista de puntos dentro de este apartado.

Justificación de las competencias específicas cubiertas (1.4.) - Solo el apartado de las competencias de Ingeniería Informática.

Diseño (2.)

Desarrollo (3.)

Apartados estrictamente relacionados con el grado en Administración y Dirección de Empresas:

Justificación de las competencias específicas cubiertas (1.4.) - Solo el apartado de las competencias de ADE.

Modelo de negocio (4.)

1.2. MOTIVACIÓN

La inspiración para realizar este trabajo surgió cuando me di cuenta de que me había olvidado de realizar el cambio de aceite al coche que conduzco, por lo que tuvo que realizarse con unos cuantos kilómetros de retraso. En ese momento fue cuando observé lo ineficiente y complicado que suele ser la gestión del mantenimiento de un vehículo, siendo el procedimiento típico el siguiente:

1. Revisar periódicamente el libro de mantenimiento que suele encontrarse en la guantera del vehículo cuando el propietario se acuerde de hacerlo.

2. Para todas y cada una de las tareas existentes en el libro de mantenimiento, comprobar el kilometraje y/o la fecha con la que se realizó por última vez con la fecha y el kilometraje actuales del vehículo.

3. Anotar las tareas que estén pasadas de fecha o kilometraje.

4. Contactar con un taller para pedir cita para que lleve a cabo las tareas de mantenimiento que el vehículo necesita.

Como puede observarse, todo este proceso es ineficiente debido a que:

- Hay que estar comprobando manualmente un conjunto de elementos, dependiendo del factor humano, que es el que genera más errores en cualquier tipo de proceso.

- Es potencialmente automatizable, debido a que tareas como la comprobación de fechas y la estimación de un kilometraje son problemas fácilmente resolubles por un algoritmo.

De esta forma fue cómo surgió la idea de crear un producto digital que resolviera estos problemas, es decir, la idea de crear ***Automatic Workshop***.

Una vez planteada esa idea inicial, mi tutor de Ingeniería Informática, José Juan Hernández Cabrera me hizo ver que también se podrían resolver los problemas típicos por los que pasan los propietarios de pequeños talleres de barrio, entre ellos, la escasa capacidad tecnológica

para competir con los talleres asociados a las grandes empresas del sector de la automoción y la dificultad para adquirir nuevos clientes. Para resolver estos problemas, se decidió añadir a ***Automatic Workshop*** la posibilidad de que se crearan cuentas para propietarios de talleres y un sistema de reservas simple mediante el que éstos pudieran negociar la prestación de sus servicios con los propietarios de vehículos cercanos.

1.3. OBJETIVOS

Los objetivos por lograr para asegurar que se resuelven total o parcialmente los problemas previamente planteados se cumplirían con la creación de un producto digital que resolviera las siguientes necesidades:

- Abordar de forma más eficiente el seguimiento de las tareas de mantenimiento del vehículo o los vehículos que una persona posea.

- Reducir la aportación del factor humano en el seguimiento del mantenimiento de un vehículo para evitar fallos.

- Comunicar de forma efectiva, clientes potenciales con propietarios de talleres.

Otro enfoque importante del proyecto a tener en cuenta es el de los objetivos de aprendizaje. Para este proyecto, al estar interesado en expandir mi conocimiento en las tecnologías de desarrollo web más allá de lo aprendido en la universidad, quise aprovechar para aprender a usar las siguientes herramientas:

- Laravel 7 (framework de PHP): me interesé por este framework por su gran popularidad y comunidad. Además, después de haber aprendido a utilizar *NodeJS* como framework de back end en un curso impartido por la Escuela de Organización Industrial, quería aprender a usar un framework sin apoyo docente, leyendo la documentación oficial (Laravel, 2021) y resolviendo los problemas que surgieran mediante búsquedas en Google.

- Grid (display de CSS): Grid y Flexbox son los displays con los que se pueden hacer maquetaciones responsive más fácilmente en CSS. Gracias a este proyecto he podido

aprender a usar el primero y reforzar mis conocimientos del segundo, sabiendo ahora aplicar cada uno en la situación en la que mejor se desenvuelve.

- SCSS (preprocesador de CSS): SCSS es un preprocesador que nos permite escribir un CSS más simple y compacto, proporcionando herramientas como la anidación de selectores, variables, herencia, operadores, etc. En mi caso, solo he usado las dos primeras características.

- JavaScript y Vue.js 2 (framework de JavaScript): a pesar de que antes del desarrollo del TFG tenía algunos conocimientos de Vue.js aprendidos en el curso previamente mencionado, tener que abordar en solitario un proyecto más grande que cualquiera que haya realizado anteriormente con este framework me ha ayudado a aprender a manejar mejor tanto el framework como el JavaScript que éste utiliza, aplicando inversión de control, simplificando código, etc.

- Puppeteer (paquete de Node.js): puppeteer es un paquete de web scraping con capacidad para esperar a que las dependencias de la página consultada terminen de cargarse antes de empezar a recopilar información.

1.4. JUSTIFICACIÓN DE LAS COMPETENCIAS ESPECÍFICAS CUBIERTAS

Grado en Ingeniería Informática:

- CII01 - Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.

- CII08 - Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.

- IS01 - Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y

eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software.

- IS02 - Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones.

Se justifica que el desarrollo de ***Automatic Workshop*** cumple con las competencias listadas, debido a que se ha buscado la seguridad y la calidad en el desarrollo haciendo uso de herramientas avanzadas de Laravel que nos ayudan a lograr este objetivo, siendo las herramientas de seguridad el sistema de tokens "Sanctum" de Laravel para proteger la API y el sistema de "Gates" y "Policies" para proteger el acceso a la base de datos. Por otro lado, las herramientas que nos ayudan a mejorar la calidad del código han sido las "local scopes", que nos ayudan a situar el código que realiza operaciones complejas de acceso a la base de datos en una capa más adecuada, los agrupamientos de "endpoints" por recurso, así como, decisiones de diseño que mejoran la escalabilidad (uso de directorios para las "migrations", "traits", funciones de servicio, optimización de consultas de "Eloquent", etc.). Todo ello se ha realizado teniendo en cuenta los requisitos del proyecto (listados en el apartado 1.6.) que derivan de las necesidades de usuarios potenciales planteadas previamente.

Grado en Administración y Dirección de Empresas:

- CE2 - Habilidad para el diseño y gestión de proyectos.

- CE7 - Poseer y comprender conocimientos acerca de la relación entre la empresa y su entorno.

- CE17 - Redactar proyectos de gestión global o de áreas funcionales de la empresa.

Todas estas competencias pueden verse cumplidas en el desarrollo del modelo de negocio de ***Automatic Workshop,*** que se mostrará de forma levemente resumida en las siguientes páginas de esta memoria.

1.5. APORTACIONES

Las grandes empresas de cualquier sector poseen grandes ventajas competitivas tecnológicas con respecto a las pequeñas y medianas empresas del mismo sector. Es lógico que esto suceda, debido a que su tamaño les permite realizar inversiones tecnológicas que tienen la característica de que, a un coste fijo, pueden proporcionar rentabilidad en todas las localizaciones en las que la empresa tenga presencia, produciendo así economías de escala en el negocio. Esto motiva que este tipo de empresas suela ser más propensa a automatizar sus procesos productivos mediante la adquisición de software hecho a medida, bien sea desarrollado por ellas mismas, o bien desarrollado por una empresa externa.

Este trabajo nos muestra cómo es posible desarrollar productos digitales genéricos enfocados a las pequeñas y medianas empresas de un sector para nivelar las ventajas competitivas tecnológicas que poseen las grandes empresas que utilizan software hecho a medida. Este fenómeno resulta interesante debido a que, como se suele enseñar en las titulaciones pertenecientes a la rama de economía y empresa, mejorar la competitividad en un sector conlleva una mejora en la relación calidad-precio de los productos o servicios que ese sector proporciona a la sociedad, incrementando la satisfacción de los consumidores.

1.6.REQUISITOS

Los requisitos planteados que asegurarían que ***Automatic Workshop*** cumple los objetivos planteados previamente son:

- Deben existir dos tipos de usuarios distintos, que serán, los propietarios de vehículos y los propietarios de talleres.

- Los usuarios que sean propietarios de vehículos deben poder añadir los datos de sus vehículos, verlos, modificarlos y eliminarlos.

- Automatic Workshop usará los datos aportados por los propietarios de vehículos para estimar el kilometraje de los vehículos cada semana.

- Automatic Workshop indicará las tareas de mantenimiento que deban ser realizadas para cada vehículo, ya sea porque estén pasadas de kilometraje o de fecha.

- Los usuarios que sean propietarios de talleres podrán visualizar los vehículos pertenecientes a propietarios de vehículos que se encuentren en su misma región geográfica y que necesiten que se realice alguna tarea de mantenimiento.

- Los usuarios que sean propietarios de talleres podrán enviar ofertas para la realización de tareas de mantenimiento que los vehículos que visualizan necesitan, indicando el precio por tarea de mantenimiento, así como el precio total.

- Los usuarios propietarios de vehículos podrán visualizar las ofertas que les llegan para la realización de tareas de mantenimiento en alguno de sus vehículos, pudiendo descartarlas o aceptar una de ellas. Las ofertas aceptadas se convertirán en citas.

2. DISEÑO

2.1. HERRAMIENTAS DE DESARROLLO

Algunas herramientas de desarrollo que se deciden utilizar condicionan el diseño del software que se programa, debido a que han sido concebidas siguiendo algún tipo de filosofía de desarrollo, como puede ser un patrón de diseño, una arquitectura, etc. Por este motivo, antes de hablar de las decisiones de diseño conviene listar las herramientas de desarrollo utilizadas.

- Editor de código: Visual Studio Code.
- Sistema de control de versiones: Git junto con un repositorio de Github.
- Plataforma de desarrollo: XAMPP.
- Base de datos: MySQL.
- Framework de back end: Laravel 7.
- Framework de front end: Vue.js 2.
- Tecnologías de maquetación: HTML y SCSS.
- Librería de iconos: Font Awesome.
- Librería de alertas: Sweet Alert 2.

2.2. ESTRUCTURA DE LA BBDD

Empezaremos echando un vistazo a la estructura de la base de datos, que se proporciona dividido en las siguientes figuras 1, 2 y 3 para mejor visualización. Observándola, se pueden deducir algunas decisiones de diseño que se comentarán a continuación.

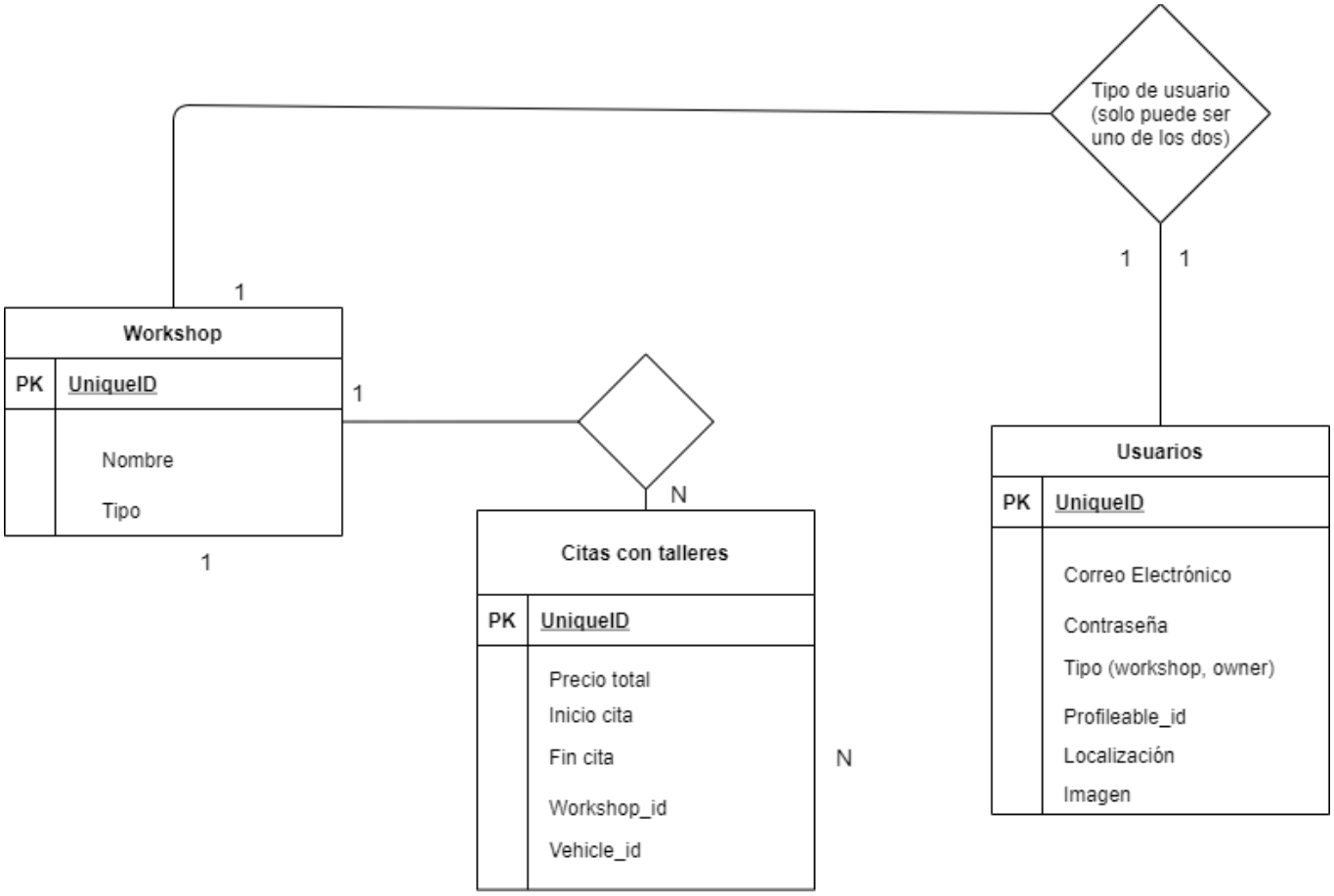

Figura 1. Segmento relativo a los talleres.

Fuente: elaboración propia.

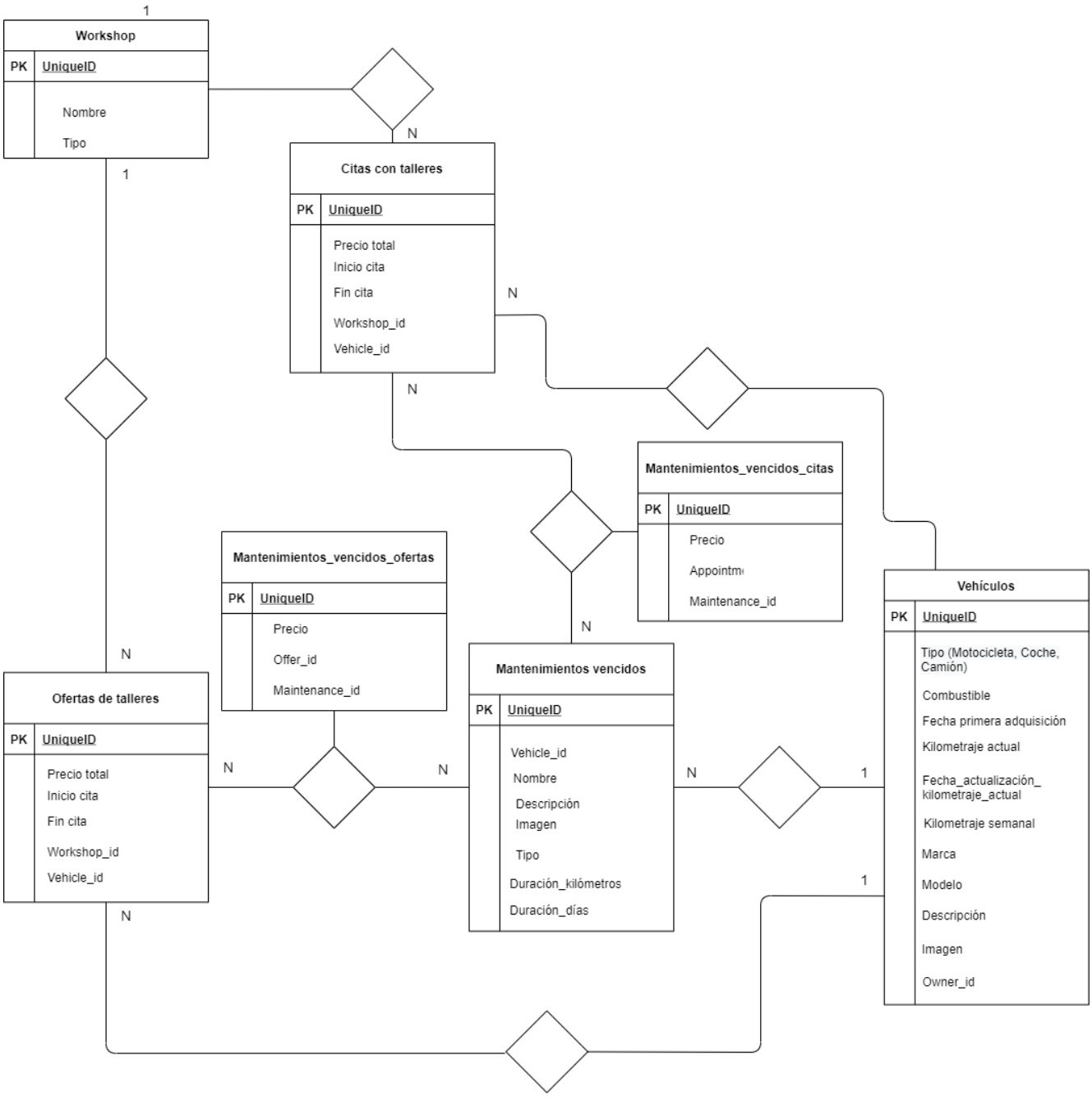

Figura 2. Relaciones entre talleres y vehículos.

Fuente: elaboración propia.

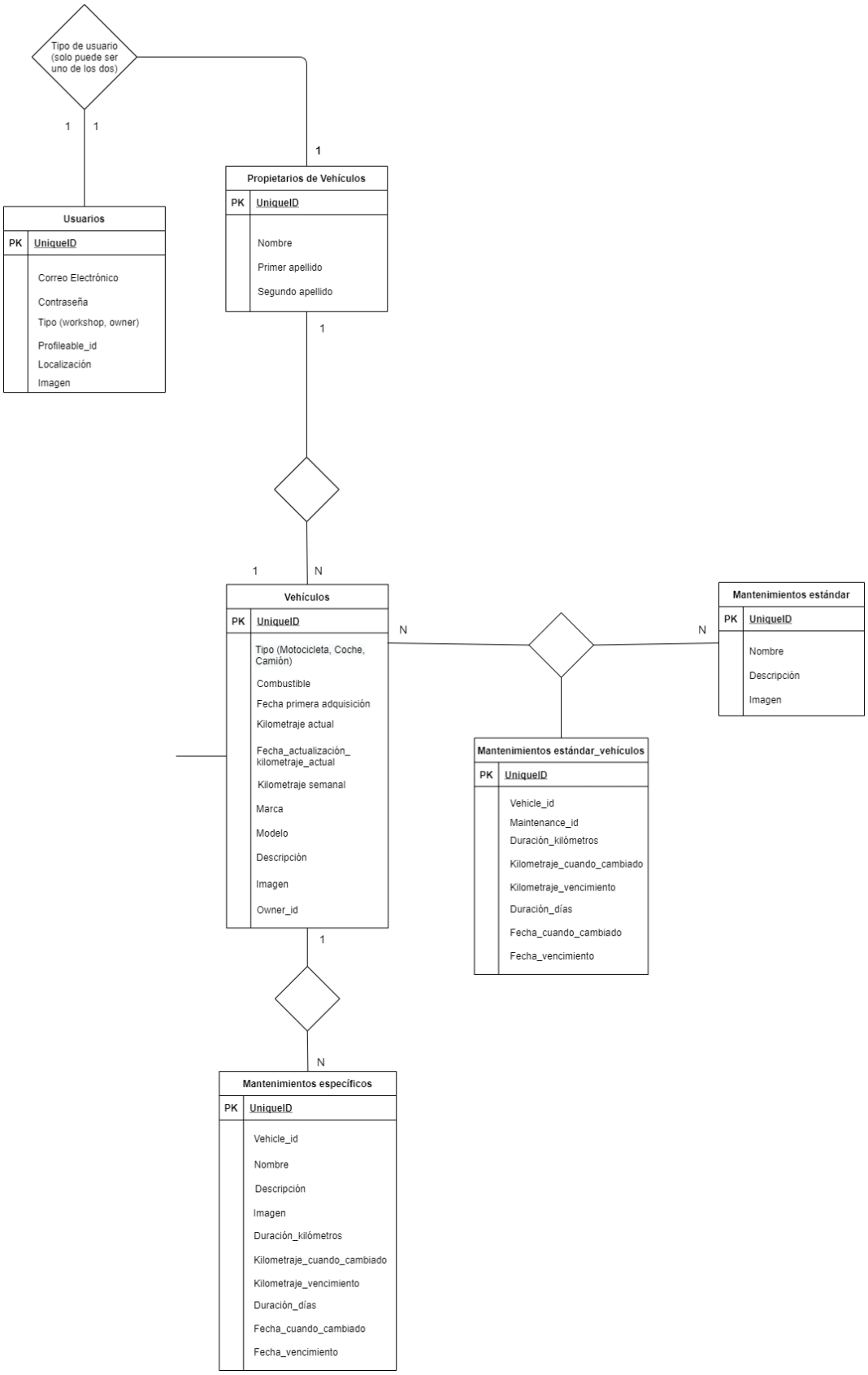

Figura 3. Relaciones entre los vehículos y sus tareas de mantenimiento.

Fuente: elaboración propia.

Tal como se puede apreciar en las figuras 1 y 3, se ha tomado la decisión de agrupar los campos comunes de los distintos tipos de usuarios en una única tabla llamada "usuarios", usando una relación polimórfica de tipo "uno a uno" con la tabla "propietarios de vehículos" o la tabla "talleres", según proceda, para extender la información del usuario.

En la figura 2 se puede observar cómo se ha diseñado la gestión de las tareas de mantenimiento de los vehículos que pertenezcan a un propietario de vehículos. Existe una tabla de "mantenimientos estándar", que recoge las tareas de mantenimiento típicas que suelen tener la mayoría de los vehículos (cambio de aceite, cambio de ruedas, alineación de la dirección…), con el objetivo de evitar al usuario la creación de todas las tareas de mantenimiento desde cero cada vez que quiera asociar tareas de mantenimiento a uno de sus vehículos. Por otro lado, en el caso de que el usuario propietario de vehículos no encuentre en la lista de tareas de mantenimiento comunes alguna tarea de mantenimiento que le gustaría tener controlada, bien sea porque es una tarea muy específica del vehículo que él posee (por ejemplo, si posee un vehículo con motor Wankel, con piezas muy distintas a las de los motores comunes), o porque no ha sido tenida en cuenta desde ***Automatic Workshop***, se puede crear una tarea de mantenimiento específica y asignarla al vehículo que se desee.

De nuevo, en la figura 3, podemos intuir cómo se gestionan las ofertas que los talleres realizan a los propietarios de vehículos y la creación de citas de propietarios de vehículos con talleres. Este procedimiento comienza cuando se añaden automáticamente tareas de mantenimiento a la tabla "mantenimientos vencidos" mediante lo que Laravel denomina como "scheduled task", es decir, una porción de código que se ejecuta con una periodicidad escogida por el programador. En este caso, la labor de nuestra "scheduled task" consiste en revisar las tareas de mantenimiento de los vehículos de la base de datos semanalmente, añadiendo a la tabla "mantenimientos vencidos" aquellas que estén pasadas de kilometraje o fecha. Una vez que la tarea ha sido añadida a esta tabla, ***Automatic Workshop*** permite que los propietarios de talleres que compartan localización geográfica con los propietarios de los vehículos que tengan tareas de mantenimiento vencidas envíen ofertas para realizar dichas tareas de mantenimiento. Los propietarios de vehículos que reciban estas ofertas, pueden verlas en una lista y rechazarlas, por lo que se eliminan de la tabla "ofertas", o aceptarlas, por lo que se convierten en un nuevo registro de la tabla "citas". Adicionalmente, cabe señalar que tanto la tabla "ofertas" como la tabla "citas" hacen uso de una relación "muchos a muchos", para que las citas y las ofertas puedan incluir varias tareas de mantenimiento, cada una con un

precio asociado, así como que cada tarea de mantenimiento pueda estar relacionada con varias ofertas, para que los distintos talleres de una zona puedan enviar una oferta para dicha tarea.

2.3. INTERFAZ GRÁFICA DE USUARIO

A la hora de realizar el trabajo de maquetación web, se ha intentado aprender utilizar el estándar “Material Design” (Material Design, 2021) con el objetivo de mejorar la calidad de las maquetaciones, así como otras prácticas de maquetación modernas, como el uso de iconos semánticos y animaciones. En las siguientes figuras 4, 5, 6 y 7 podemos apreciar los resultados obtenidos.

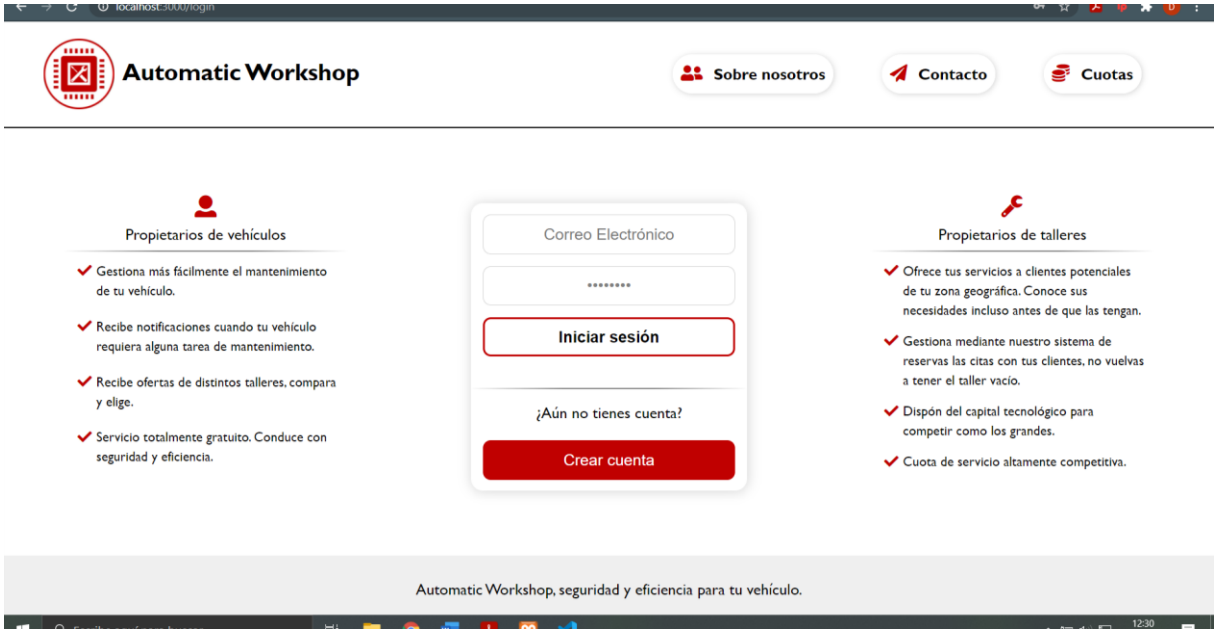

Figura 4. Página de registro e inicio de sesión

Fuente: elaboración propia.

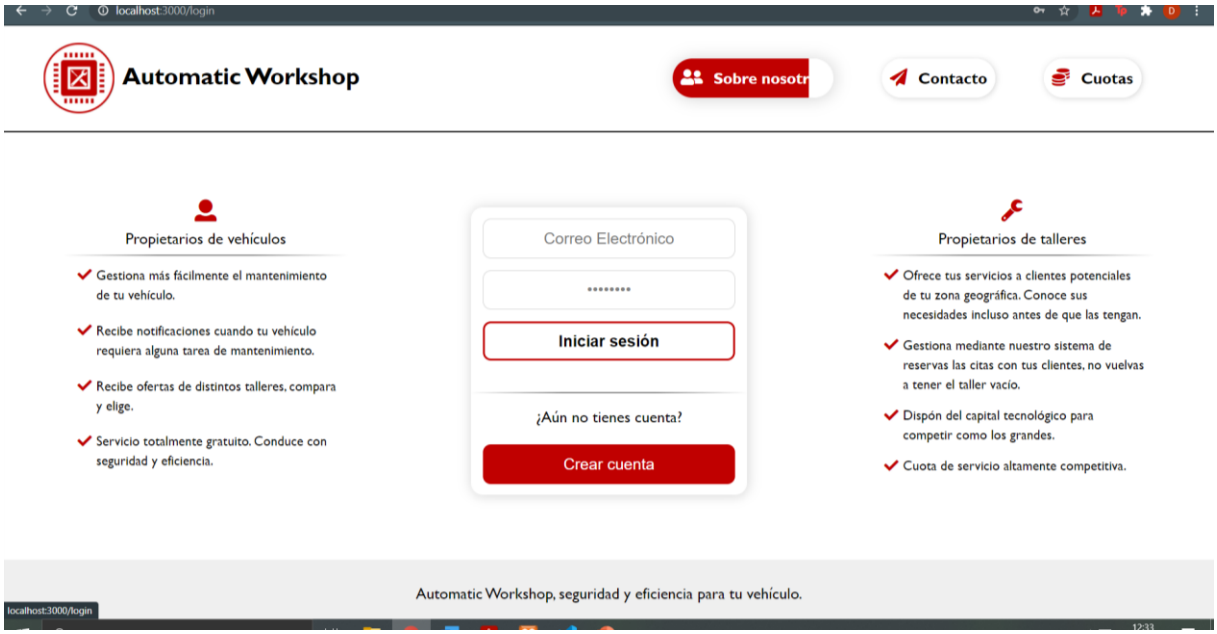

Figura 5. Animación en la página de registro e inicio de sesión.

Fuente: elaboración propia.

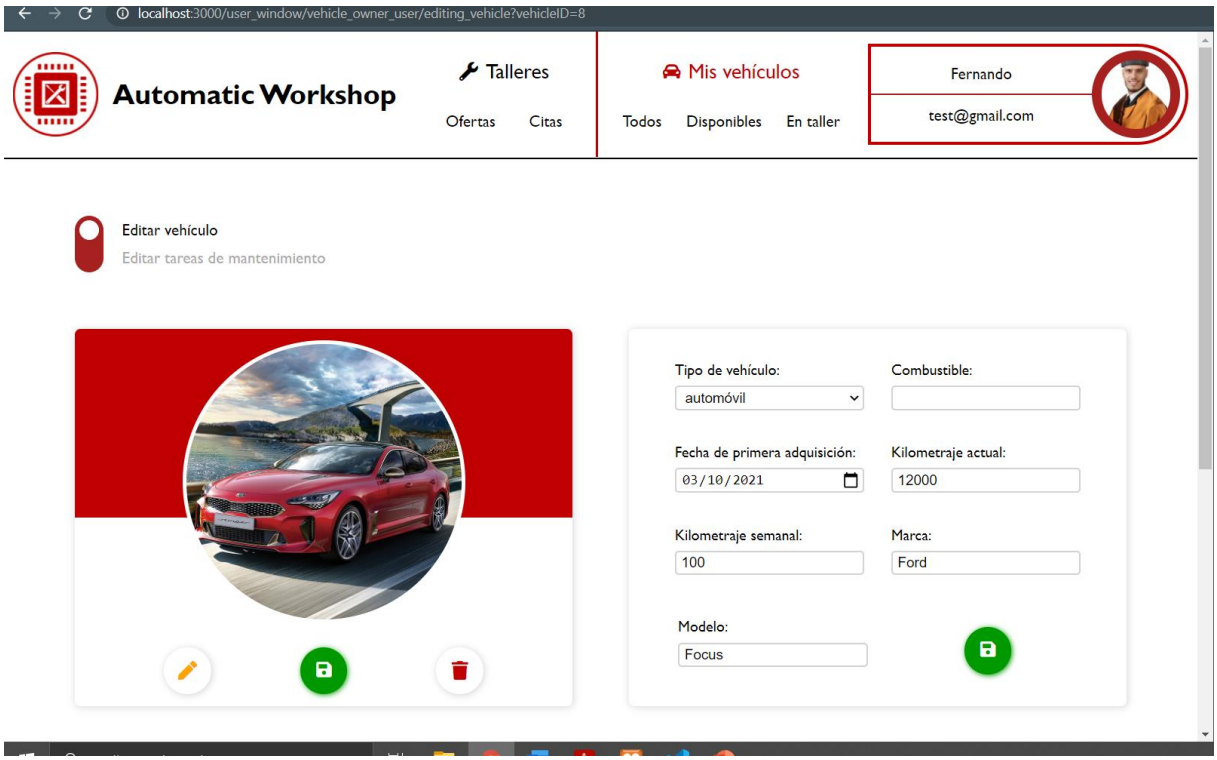

Figura 6. Página de edición de un vehículo.

Fuente: elaboración propia.

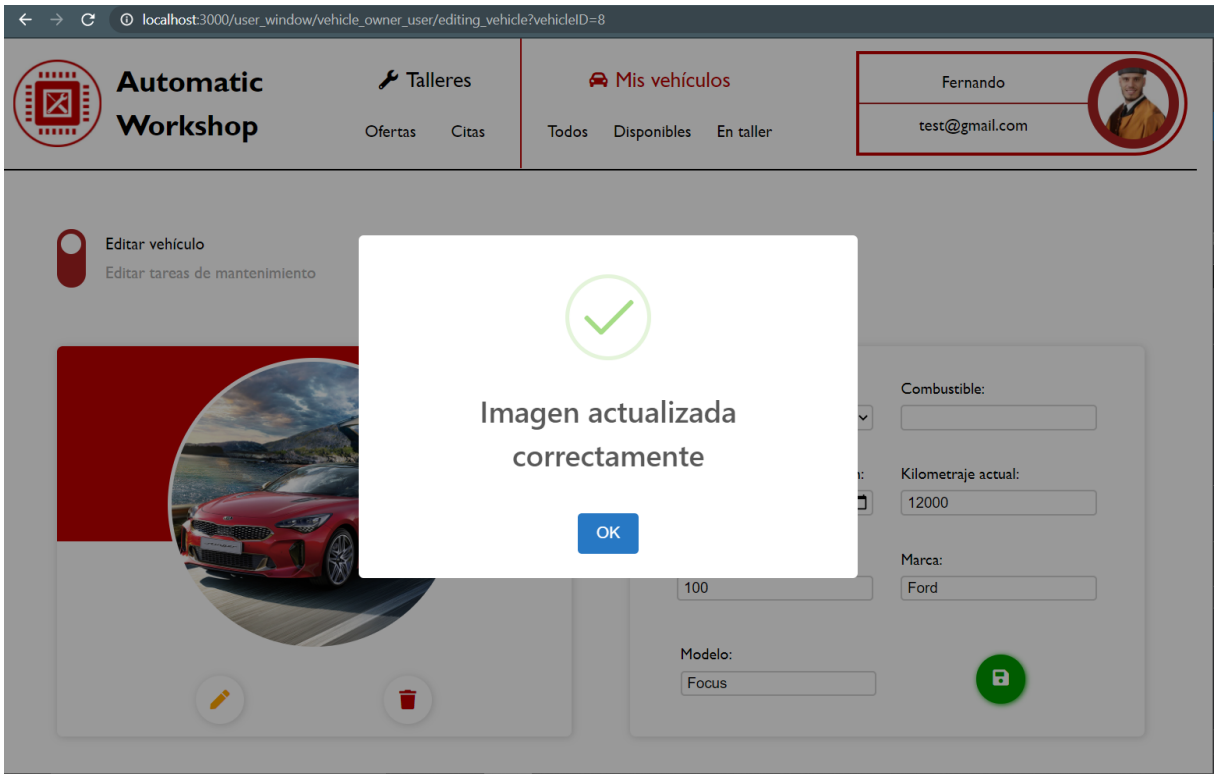

Figura 7. Modal de actualización de imagen.

Fuente: elaboración propia.

Adicionalmente, ***Automatic Workshop*** también se ha diseñado para tener una buena estructura de sus elementos en los distintos tamaños de pantalla (maquetación responsive). Para ello, se han utilizado “media queries” junto con los “displays” “flexbox” y “grid”. Podemos ver un ejemplo en las siguientes figuras 8, 9 y 10.

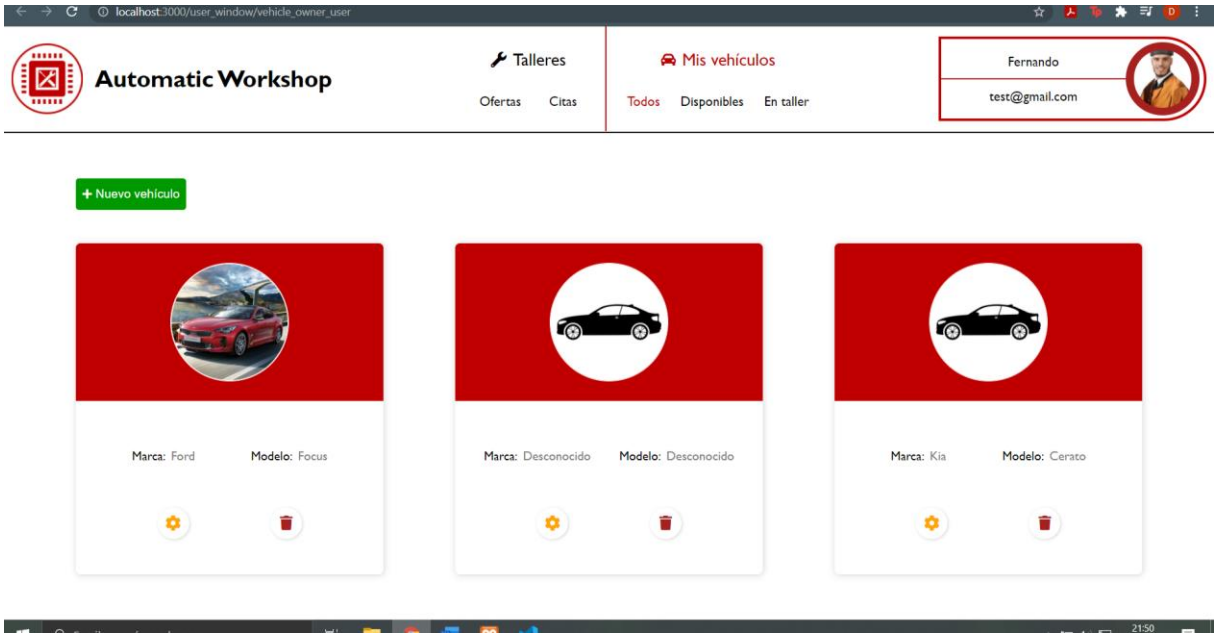

Figura 8. Maquetación para ordenador personal.

Fuente: elaboración propia.

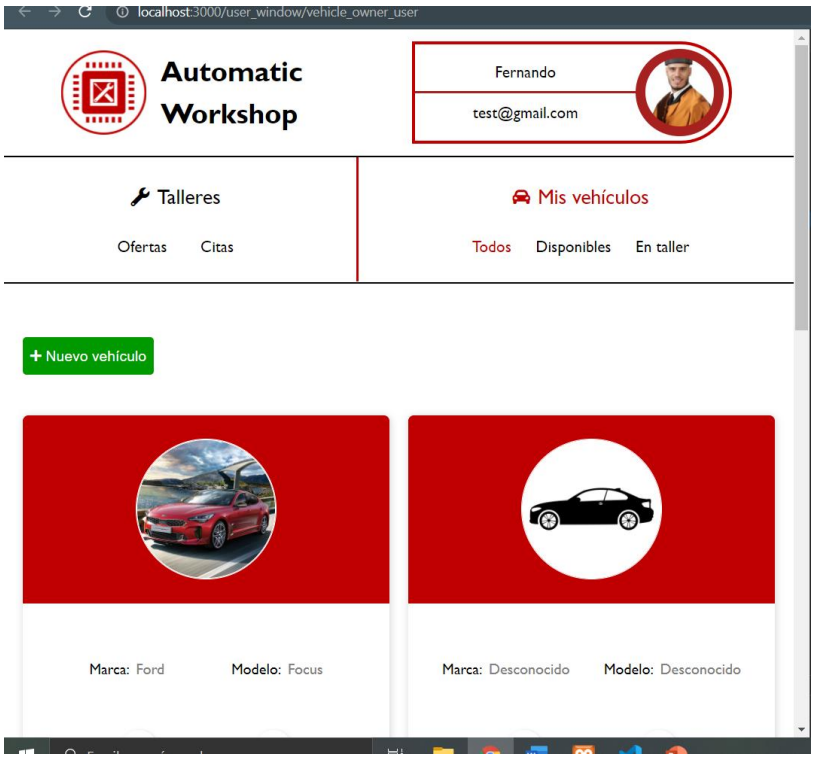

Figura 9. Maquetación para “tablet”.

Fuente: elaboración propia.

Figura 10. Maquetación para “smartphone”.

Fuente: elaboración propia.

2.4. ARQUITECTURA DE BACK END

La arquitectura que Laravel propone por su estructura de ficheros y directorios es la ya conocida “Model View Controller” (MVC). En el caso de Laravel, las vistas se gestionan con las “blades templates” que se encuentran en el directorio “/resources/views”, los controladores se encuentran en “/app/Http/Controllers” y los modelos en “app/Models”. Además, existen otros múltiples directorios con propósitos específicos que son fundamentales para desarrollo con laravel, que realizan tareas de diversa índole, como pueden ser tareas de enrutamiento (“/routes”), control de versiones de las tablas de la BBDD (“/database/migrations”), validación de peticiones (“/app/Http/Requests”), etc.

3. DESARROLLO

3.1. SISTEMA DE AUTENTICACIÓN

Un buen punto de partida en el desarrollo de una aplicación web es la creación del sistema de protección de rutas del servidor para así poder proteger las futuras rutas que desarrollemos. Para este propósito se ha utilizado el paquete "sanctum" de Laravel, que permite crear un sistema de autenticación basado en "tokens" de forma muy sencilla. Este sistema impide que los clientes no autenticados como usuarios de la plataforma puedan acceder a una ruta distinta a la que permite iniciar sesión y crear cuentas nuevas, debido a que las rutas del servidor que realizan estas acciones son las que entregan los "tokens" a los clientes. En la siguiente figura 11 se muestra la implementación de la protección de varias rutas con el middleware "sanctum", un grupo de rutas con la misma protección y las rutas de inicio de sesión y creación de cuentas sin protección.

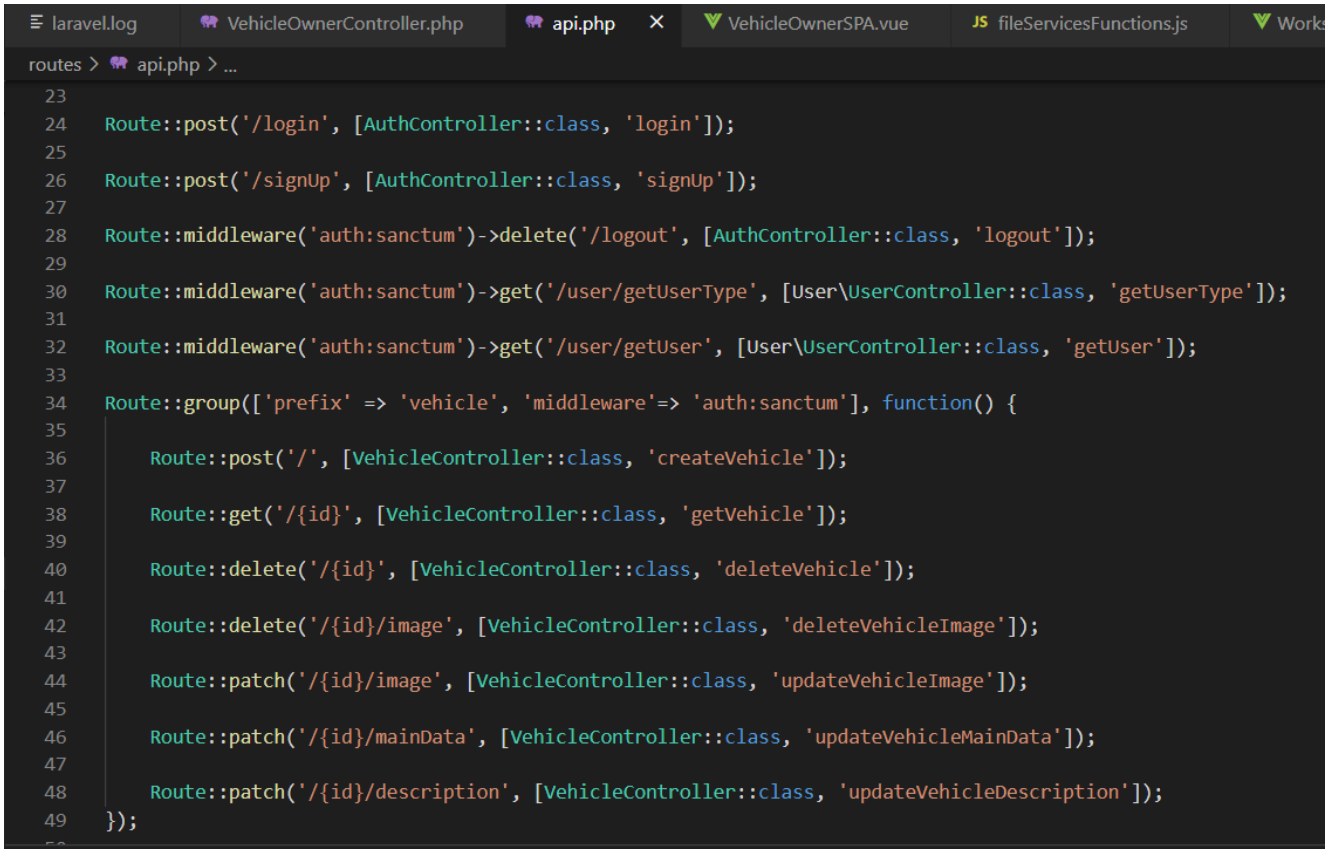

Figura 11. Archivo de rutas "api.php".

Fuente: elaboración propia.

Echando un vistazo a las funciones asociadas a la ruta “/login”, podemos ver cómo se crean y se entregan los tokens en la siguiente figura 12.

```php
public function login(Request $request) {

    $user = User::where('email', $request->email)->get()->first();
    if (empty($user)) abort(404, 'Email '. $request->email .' not found in database.');
    $userType = strtolower($user->type);

    if (!Hash::check($request->password, $user->password)) {
        abort(404, 'Error: wrong password');
    } else if (($userType != 'vehicleowner') && ($userType != 'workshop')) {
        $error = 'Error: user '.$request->email.' is neither a vehicle owner or a workshop owner';
        Log::error($error);
        abort(500, $error);
    } else {
        $user->tokens()->delete();
        $token = $user->createToken($user->email);
        return ['token' => $token->plainTextToken, 'userType' => $userType];
    }
}
```

Figura 12. Función “login”.

Fuente: elaboración propia.

La ruta de “/signUp” es bastante parecida, pero se añade el código que crea un usuario en la base de datos.

Gracias a este sistema de “tokens” se ha podido implementar otro sistema que permita que ***Automatic Workshop*** recuerde las credenciales de los usuarios que han iniciado sesión previamente. Podemos ver su esquema de funcionamiento en la siguiente figura 13.

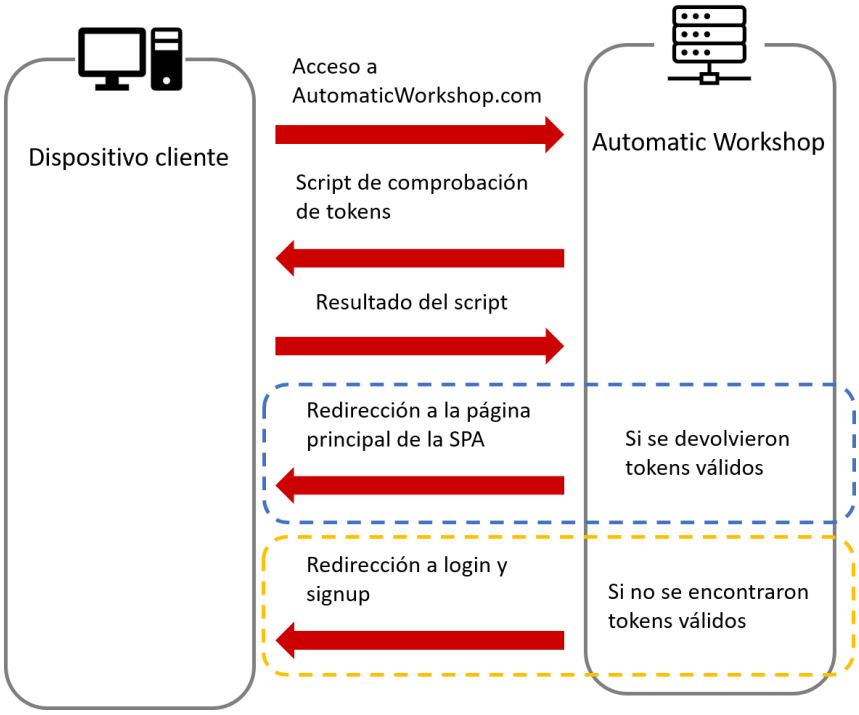

Figura 13. Sistema de recordatorio de credenciales.

Fuente: elaboración propia.

En la siguiente figura 14, se muestra el script que se encarga de validar los tokens de usuario para evitar a los usuarios realizar un nuevo inicio de sesión cada vez que acceden a ***Automatic Workshop***.

```
import APIcallService from '../httpServices/APIcallService';

const userToken = localStorage.getItem('userToken');
if(!userToken) window.location.replace('/login');
const parsedUserToken = JSON.parse(userToken);

APIcallService.performRequestWithoutData(
    'get',
    'user/getUserType',
    {headers: {'Authorization' : 'Bearer ' + parsedUserToken}}
).then(userExists => {
    if (userExists.data) {
        (userExists.data === 'vehicleOwner')
        ? window.location.replace('/user_window/vehicle_owner_user')
        : window.location.replace('/user_window/workshop_user');
    } else {
        window.location.replace('/login');
    }
}).catch((err) => {
    alert('Error: your token does not have any user attached in database');
    window.location.replace('/login');
});

```

Figura 14. Script de recordatorio de credenciales.

Fuente: elaboración propia.

3.2. FRONT END

Pasando al desarrollo del lado del cliente, se ha decidido construir una “Single Page Application” (SPA), debido a que este tipo de aplicación web proporciona una mejor experiencia de usuario, al entregar la funcionalidad de navegación sin que se experimenten recargas en el navegador, además de las previamente existentes peticiones AJAX (Asynchronous JavaScript and XML), que permiten pedir información a la base de datos sin que se experimenten las mismas recargas en el navegador. Para este propósito se ha utilizado el “framework” de desarrollo “Vue.js”, que se integra perfectamente con Laravel cuando instalamos el paquete “laravel/ui”. Podemos ver un ejemplo de integración de un componente de “Vue.js” con una plantilla “blade” de Laravel en la siguiente figura 15.

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>AutomaticWorkshop</title>
        <link href="https://use.fontawesome.com/releases/v5.0.1/css/all.css" rel="stylesheet">
        <link rel="stylesheet" href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
            integrity="sha384-AYmEC3Yw5cVb3ZcuHtOA93w35dYTsvhLPVnYs9eStHfGJvOvKxVfELGroGkvsg+p"
            crossorigin="anonymous"
        />
    </head>
    <body class="antialiased">
        <div id="app"> {{-- id necessary for vue to work properly --}}
            <login-component></login-component>
        </div>
        <style>
            /*! normalize.css v8.0.1 | MIT License | github.com/necolas/normalize.css */html{line-height:1.15;-webkit-text-size-adjust:100%}
        </style>
    </body>
    {{--Adding Javascript to the document. Important to import it after </body> so it charges the scripting when HTML structure is ready--}}
    <script src="{{ asset('js/app.js') }}"></script>
</html>
```

Figura 15. Componente “login” en “blade template”.
Fuente: elaboración propia.

En la figura 16, encontramos cuál es el código del archivo “app.js” que nos permite que estos componentes de Vue puedan ser insertados en las plantillas “blade”.

```
import router from './router';

require('./bootstrap');

window.Vue = require('vue');

/**
 * The following block of code may be used to automatically register your
 * Vue components. It will recursively scan this directory for the Vue
 * components and automatically register them with their "basename".
 *
 * Eg. ./components/ExampleComponent.vue -> <example-component></example-component>
 */

// const files = require.context('./', true, /\.vue$/i)
// files.keys().map(key => Vue.component(key.split('/').pop().split('.')[0], files(key).default))

Vue.component('login-component', require('./authentication/Login.vue').default);

Vue.component('sign-up', require('./authentication/SignUp.vue').default);

Vue.component('user-window', require('./userWindow/UserWindow.vue').default);

/**
 * Next, we will create a fresh Vue application instance and attach it to
 * the page. Then, you may begin adding components to this application
 * or customize the JavaScript scaffolding to fit your unique needs.
 */

const app = new Vue({
    el: '#app',
    router
});
```

Figura 16. Archivo “app.js”.

Fuente: elaboración propia.

Quien haya examinado la figura 16 con detenimiento se habrá percatado de que la instancia de Vue utiliza la dependencia “router”. Esta dependencia o paquete es la que nos facilita el trabajo de creación de la antes mencionada SPA con Vue. En las siguientes figuras 17, 18 y 19 podemos analizar su funcionamiento.

```
import AllVehicles from './vehicleRelatedViews/AllVehicles/AllVehicles';
import AvailableVehicles from './vehicleRelatedViews/AvailableVehicles/AvailableVehicles';
import InWorkshop from './vehicleRelatedViews/InWorkshop/InWorkshop';
import EditingVehicle from './vehicleRelatedViews/EditingVehicle/EditingVehicle';

import WorkshopOffers from './workshopRelatedViews/WorkshopOffers';
import workshopAppointments from './workshopRelatedViews/workshopAppointments';

const vehicleOwnerBasePath = '/user_window/vehicle_owner_user';

const vehicleOwnerRoutes = [
    {
        path: vehicleOwnerBasePath,
        name: 'AllVehicles',
        component: AllVehicles
    },
    {
        path: vehicleOwnerBasePath + '/avaiable',
        name: 'AvaiableVehicles',
        component: AvailableVehicles
    },
    {
        path: vehicleOwnerBasePath + '/in_workshop',
        name: 'InWorkshop',
        component: InWorkshop
    },
```

Figura 17. Archivo “vehicleOwnerRoutes.js”.

Fuente: elaboración propia.

```
    {
        path: vehicleOwnerBasePath + '/editing_vehicle',
        name: 'EditingVehicle',
        component: EditingVehicle
    }
];

export default vehicleOwnerRoutes;
```

Figura 18. Archivo “vehicleOwnerRoutes.js”.

Fuente: elaboración propia.

Como se ve en las figurar previas, la definición de rutas consiste en la creación y exportación de un vector de objetos que contienen el nombre, dirección y componente asociado de cada ruta.

```
import Vue from "vue";
import VueRouter from 'vue-router'

import vehicleOwnerRoutes from './userWindow/vehicleOwnerViews/vehicleOwnerRoutes';
import workshopRoutes from './userWindow/workshopViews/workshopRoutes';
import commonViews from './userWindow/commonViews/commonViews';

const routes = vehicleOwnerRoutes.concat(workshopRoutes, commonViews);

Vue.use(VueRouter);

const router = new VueRouter({
    mode: 'history',
    routes: routes
});

export default router;
```

Figura 19. Archivo “routes.js”.

Fuente: elaboración propia.

Esta figura es la que nos muestra la creación de un “router” como instancia de “VueRouter”, a partir de la concatenación de todos los vectores de objetos con información sobre las rutas creados en otros archivos. Posteriormente, se exporta el “router” para usarlo en “app.js”.

Por último, los servicios de validación de datos son otro aspecto del front end al que es interesante echar un vistazo. Para trabajar más cómodamente con los errores de validación, se han estandarizado creando una clase para definirlos, de esta forma los errores son objetos con una propiedad “success” que siempre tiene el valor “false” y otra propiedad “errors” que especifica los tipos de errores que hayan podido ocurrir. El código que implementa esta funcionalidad se encuentra en la siguiente figura 20.

```
export default class ValidationError {
    constructor() {
        this.success = false;
        this.errors = [];
    }

    addError(errorType, msg) {
        this.errors[errorType] = msg;
    }
}
```

Figura 20. Clase “ValidationError”.

Fuente: elaboración propia.

Siguiendo el principio de desarrollo de software conocido como “DRY” (Don’t Repeat Yourself), se ha creado un servicio que agrupa las tareas típicas de validación, extrayendo así esta responsabilidad de los servicios de validación acoplados al recurso sobre el que se quiere actuar. En las siguientes figuras 21 y 22 encontramos el archivo que realiza esta tarea.

```
import ValidationError from '../../types/validationError';

const validationTopLevelFunctions = {

    validateEmptyField(field) {
        if(!field) return {success: false};
        return {success: true};
    },
    validateLength(name, characters) {
        if(name.length > characters) return {success: false};
        return {success: true};
    },
    validateName(name) {
        const regex = new RegExp(/^([A-zÀ-ú]+['|\-|\s]?)*[A-zÀ-ú]+$/);
        const validationResult = regex.test(name);
        if(!validationResult) return {success: false};
        return {success: true};
    },
    validateNumber(number) {
        if(isNaN(number)) return {success: false};
        return {success: true};
    },
    validateNumerInterval(number, min, max) {
        if(number > max || number < min) return {success: false};
        return {success: true};
    },
```

Figura 21. Archivo “validationFinalFunctions.js”.

Fuente: elaboración propia.

```
    validateEmailStructure(email) {
        const regex = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{
        if(!regex.test(email)) {
            let validationErrors = new ValidationError();
            validationErrors.addError('email', 'Formato de correo electrónico inválido');
            return validationErrors;
        } else {
            return {success: true};
        }
    },
    validateDateTime(dateTime) {
        const regex = /^([1-2]\d{3})-([0]?[1-9]|1[0-2])-([0-2]?[0-9]|3[0-1]) (20|21|22|23|[0-1]?\d{1}):([0-5]?\d{1})$/;
        if(!regex.test(dateTime)) {
            let validationErrors = new ValidationError();
            validationErrors.addError('dateTime', 'El formato del momento de entrada o salida es incorrecto');
            return validationErrors;
        } else {
            return {success: true};
        }
    }
}

export default validationTopLevelFunctions;
```

Figura 22. Archivo “validationFinalFunctions.js”.

Fuente: elaboración propia.

El código cliente de las funciones que se muestran en las dos figuras anteriores será aquel cuyo objetivo sea validar datos de entrada relacionados con un recurso concreto de la base de datos. Veamos un ejemplo en las siguientes figuras 23 y 24.

```
import ValidationError from '../../../../types/validationError';
import validationFinalFunctions from '../../../../commonServices/validationServices/validationFinalFunctions';

const validationTopLevelFunctions = {
    validateVehicle(
        fuelType,
        firstAcquisitionDate,
        currentKilometers,
        weeklyKilometers,
        brandName,
        modelName
    ) {
        let validationResult = validateVehicleEmptyFields(firstAcquisitionDate, currentKilometers, weeklyKilometers);
        if (!validationResult.success) return validationResult;

        validationResult = validateNamesLength(brandName, modelName, fuelType);
        if (!validationResult.success) return validationResult;

        validationResult = validateNames(brandName, modelName, fuelType);
        if (!validationResult.success) return validationResult;

        validationResult = validateKilometers(currentKilometers, weeklyKilometers);
        if (!validationResult.success) return validationResult;

        return {success: true};
    }
}

export default validationTopLevelFunctions;
```

Figura 23. Función “validateVehicle”.

Fuente: elaboración propia.

```
function validateVehicleEmptyFields(firstAcquisitionDate, currentKilometers, weeklyKilometers) {
    let validationErrors = new ValidationError();
    let validationResult;

    validationResult = validationFinalFunctions.validateEmptyField(firstAcquisitionDate);
    if (!validationResult.success) {
        validationErrors.addError(
            'firstAcquisitionDate', 'Debe proporcionar la fecha de primera adquisición'
        );
        return validationErrors;
    }

    validationResult = validationFinalFunctions.validateEmptyField(currentKilometers);
    if (!validationResult.success) {
        validationErrors.addError(
            'currentKilometers', 'Debe proporcionar el kilometraje actual'
        );
        return validationErrors;
    }

    validationResult = validationFinalFunctions.validateEmptyField(weeklyKilometers);
    if (!validationResult.success) {
        validationErrors.addError(
            'weeklyKilometers', 'Debe proporcionar el kilometraje semanal'
        );
        return validationErrors;
    }

    return {success: true};
}
```

Figura 24. Función “validateVehicleEmptyFields”.

Fuente: elaboración propia.

Como puede observarse, se ha seguido la técnica de composición de funciones para dividir las tareas en distintas capas de responsabilidad, facilitando así la mantenibilidad y el desarrollo del código.

3.3. BACK END

En cuanto al desarrollo del back end, como se ha mencionado anteriormente, se ha utilizado la arquitectura “MVC” que Laravel propone en su estructura. Podemos decir que las funcionalidades relacionadas con la vista se encuentran en las rutas del archivo “web.php” y en las “blade templates” (vistas en la figura 15 del apartado anterior) del directorio “views”. En la siguiente figura 25 veremos cómo funciona el archivo “web.php” y en la figura 26, cómo se organizan en los directorios las “blade templates” y los archivos de rutas.

```php
Route::view('/', 'check_local_storage');

Route::view('/login', [AuthController::class, 'login']);

Route::view('/sign_up', [AuthController::class, 'sign_up']);

Route::get('/serverError', function (Request $request) {
    return view('server_error', [
        'originURL' => $request->get('originURL'),
        'errors' => json_decode($request->get('errors'), true)
    ]);
});

//using vue_capture var for staying in the same page while SPA is changing subroutes
Route::view('/user_window/{vue_capture?}', 'user_window')->where('vue_capture', '[\/\w\.-]*')->name('user_window');
```

Figura 25. Archivo “web.php”.

Fuente: elaboración propia.

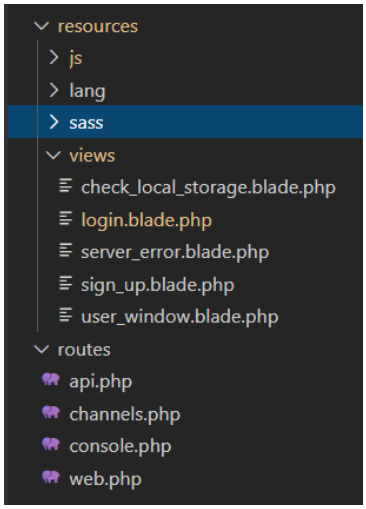

Figura 26. Directorio “views”.

Fuente: elaboración propia.

Las funcionalidades asociadas al componente controlador se desarrollan en los archivos que se encuentran dentro del directorio “/app/Http/Controllers”. Estos archivos contienen una clase controladora con funciones que se asocian a rutas específicas mediante los archivos de rutas. En la siguiente figura 27 vemos una clase controladora de ejemplo.

```php
class VehicleOwnerController extends Controller {

    public function getVehicles(Request $request) {
        return VehicleOwner::find(
            $request->user()->profileable_id
        )->vehicles()->select(
            'id', 'image', 'brand_name', 'model'
        )->simplePaginate(9);
    }
```

Figura 27. Clase controladora.

Fuente: elaboración propia.

Vemos que la clase controladora “VehicleOwnerController” de la anterior figura 27 cuenta con una función “getVehicles”. Esta función será invocada cuando en un archivo de rutas se utilize la “fachada” (Laravel, 2021) “Route”, llamando a cualquier función estática que represente alguno de los verbos básicos del protocolo HTTP (Mozilla, 2021) y pasándole como segundo argumento un vector que en su primera posición tenga la ruta a la clase controladora y una cadena de texto con el nombre de la función a ejecutar de dicha clase en su segunda posición (encontraremos un ejemplo de esto en la anterior figura 11).

Otro fenómeno que es interesante señalar sobre la figura anterior es la forma que tiene la función “getVehicles()” de llevar a cabo su cometido, que podemos intuir leyendo el nombre de la clase a la que pertenece y su propio nombre (obtener los vehículos de un propietario de vehículos). Apreciamos que esta función ejecuta instrucciones que permiten realizar varias operaciones de acceso a base de datos (selección de datos y paginación) en unas pocas líneas. Esto sucede gracias a Eloquent, el ORM (“Object-relational mapping”) de Laravel y al “query builder”, que permiten la abstracción del SQL puro para generar un código más legible.

Una tarea común que se debe realizar cuando las funciones controladoras trabajan con datos que reciben desde el cliente en la petición es la validación de dichos datos. En Laravel existen dos maneras de realizar esta tarea, usando el método “validate” de la clase “Request”, cuando la validación no es muy compleja y puede tener lugar dentro de la función del controlador, o creando una “FormRequest” cuando la validación es suficientemente compleja o es susceptible de ser reutilizada en otras funciones controladoras. En las siguientes figuras 28, 29 y 30 encontramos un ejemplo de cada uno.

```php
    public function updateVehicleDescription(Request $request, $id) {
        $vehicle = Vehicle::find($id);
        if ($request->user()->cannot('update', $vehicle)) abort(401);

        $request->validate([
            'vehicleDescription' => 'nullable | max:1000'
        ]);

        $vehicle->description = $request->vehicleDescription;
        $vehicle->save();
        return;
    }
}
```

Figura 28. Uso del método “validate”.

Fuente: elaboración propia.

```php
    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules() {
        $result = array_merge(
            [
                'mainData.maintenanceTitle' => ['required', 'max:40', 'regex:/^([A-zÀ-ú0-9]+[\s]?)*[A-zÀ-ú0-9]+$/'],
                'mainData.maintenanceDescription' => ['nullable', 'max:1000'],
                'maintenanceImage' => ['nullable', 'image', 'mimes:jpg,png,jpeg', 'max:500000']
            ],
            $this->traitTaskTrackingData($this->route('id'))
        );
        return $result;
    }

    protected function prepareForValidation() {
        $this->merge([
            'taskTrackingData' => json_decode($this->taskTrackingData, true),
            'mainData' => json_decode($this->mainData, true),
            'maintenanceImage' => json_decode($this->maintenanceImage, true)
        ]);
    }
}
```

Figura 29. Validación mediante la “FormRequest” “SpecificTaskRequest”.

Fuente: elaboración propia.

```php
class SpecificMaintenancesController extends Controller {

    public function getMaintenances(Request $request, $id) {
        $vehicle = Vehicle::find($id);
        return $vehicle->specificMaintenances()->simplePaginate(9);
    }

    public function addMaintenance(SpecificTaskRequest $request, $id) {
```

Figura 30. Uso de la “FormRequest” “SpecificTaskRequest”.

Fuente: elaboración propia.

Ahora que se ha introducido el componente controlador, junto con el ORM “Eloquent” y las “FormRequest”, resulta apropiado introducir el concepto de “middleware”. Un middleware es un fragmento de código que se ejecuta antes o después de la función controladora cuando se atiende una petición. De esta manera, las “FormRequest” y el paquete “Sanctum” son middlewares que se ejecutan antes de entrar al componente controlador (por tanto, se ejecutan entre la vista y el controlador), mientras que “Eloquent” es un middleware que se ejecuta después de la entrada en una función del componente controlador (por tanto, se ejecuta entre el controlador y el modelo).

Vistos los componentes “controlador” y “vistas” del patrón MVC en Laravel, procedemos a explicar cómo se desarrolla el componente “modelo” en el framework. Para este cometido, Laravel cuenta con un directorio “/app/Model” dentro del cual se deben situar los archivos que contienen las clases que representan a las entidades que participan en el funcionamiento de la aplicación. En la siguiente figura 31 se muestra un archivo de este directorio como ejemplo.

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Workshop extends Model {
    use HasFactory;

    protected $table = 'workshops';

    protected $fillable = ['name', 'workshop_type', 'capacity'];

    public function user() {
        return $this->morphOne(User::class, 'profileable');
    }

    public function offers() {
        return $this->hasMany(WorkshopOffer::class);
    }

    public function appointments() {
        return $this->hasMany(WorkshopAppointment::class);
    }

}
```

Figura 31. Archivo de modelo de la entidad “taller”.

Fuente: elaboración propia.

En la figura se observan las tareas principales que se lleva a cabo una clase modelo, que son, la relación del modelo con una tabla de base de datos (Laravel las relaciona automáticamente por nombre, pero declarar la tabla nos protege de que no lo haga por si no escribimos todo siguiente la notación que Laravel utiliza para relacionar tablas y modelos), la declaración de los campos que pueden ser editados por Eloquent (fillable), la declaración de campos que no se permite que sean editados (guarded), los campos que deben ser ocultados cuando se mapeen a objetos desde la base de datos (hidden), las relaciones (los objetos del ORM utilizan métodos para representar las relaciones que existen en la base de datos con claves extranjeras. En este caso observamos que existen, por orden de aparición, una relación "uno a uno" polimórfica con la clase "usuario" y dos relaciones "uno a muchos" con las clases "oferta" y "cita"), etc.

La definición de la lógica de acceso a la base de datos también se encuentra entre las tareas de una clase de modelo. En Laravel se podría decir que esta tarea la desempeñan, al menos en parte, las "local scopes", unas funciones que nos permiten añadir restricciones a las consultas que se le hagan a un objeto de la clase modelo. Vemos un ejemplo de ello en la siguiente figura 32.

```php
public function scopeCurrentLocation($query, $location) {
    return $query->select('vehicle_id')->whereHas('vehicle', function($query) use ($location) {
        $query->select('id', 'owner_id')->whereHas('owner', function($query) use ($location) {
            $query->select('id')->whereHas('user', function($query) use ($location) {
                $query->select('profileable_id', 'location')->where('location', $location);
            });
        });
    });
}
```

Figura 32. "Local scope".

Fuente: elaboración propia.

Además de las clases de modelo, existe otra herramienta que Laravel proporciona para gestionar el intercambio de información con la base de datos: las "policies" y "gates". Estos dos conceptos forman parte del mismo sistema que se encarga de restringir el acceso a la base de datos para evitar problemas de seguridad. En las siguientes figuras 33 y 34 se mostrará el funcionamiento de este sistema.

```php
class VehiclePolicy {
    use HandlesAuthorization;

    /**
     * Determine whether the user can view the model.
     *
     * @param  \App\Models\User  $user
     * @param  \App\Models\Vehicle  $vehicle
     * @return mixed
     */
    public function view(User $user, Vehicle $vehicle) {
        return ($user->type === 'vehicleOwner') && ($user->profileable_id === $vehicle->owner_id);
    }

    /**
     * Determine whether the user can create models.
     *
     * @param  \App\Models\User  $user
     * @return mixed
     */
    public function create(User $user) {
        return $user->type === 'vehicleOwner';
    }

    /**
     * Determine whether the user can update the model.
     *
     * @param  \App\Models\User  $user
     * @param  \App\Models\Vehicle  $vehicle
     * @return mixed
     */
    public function update(User $user, Vehicle $vehicle) {
        return ($user->type === 'vehicleOwner') && ($user->profileable_id === $vehicle->owner_id);
    }
```

Figura 33. Política de acceso al modelo “vehículo”.

Fuente: elaboración propia.

Como se puede intuir al analizar esta figura, una política es una clase que contiene una serie de funciones que devuelven un valor booleano, si el resultado es verdadero, se puede realizar la tarea que muestra el nombre de la función, en caso contrario se devolverá un error en la petición.

```php
public function getVehicle(Request $request, $id) {
    $vehicle = Vehicle::find($id);
    if ($request->user()->cannot('view', $vehicle)) abort(401);
    return $vehicle;
}
```

Figura 34. Uso de la política “VehiclePolicy”.

Fuente: elaboración propia.

Para que una política sea invocada, hay que escribir una “gate” que compruebe si el usuario actual tiene permiso para realizar la acción que se va a ejecutar en las siguientes líneas de código, tal como se muestra en la anterior figura 34.

Acabando con el apartado de back end, fuera de la arquitectura MVC, es interesante mostrar el funcionamiento de la “Scheduled Task” (tarea programada), que permite predecir el estado de las tareas de mantenimiento de un vehículo. En este caso, se ha implementado una tarea programada que ejecuta un “job”, es decir, una clase que contiene métodos cuya ejecución es encolada en una cola de trabajos para que los clientes de la aplicación no se vean afectados por una bajada de rendimiento cuando se esté ejecutando la tarea programada. En las siguientes figuras 35 y 36 se muestran la tarea programada y el “job”.

```php
<?php

namespace App\Console;

use App\Jobs\UpdateCurrentKilometers;

use Illuminate\Console\Scheduling\Schedule;
use Illuminate\Foundation\Console\Kernel as ConsoleKernel;

class Kernel extends ConsoleKernel
{
    /**
     * The Artisan commands provided by your application.
     *
     * @var array
     */
    protected $commands = [
        //
    ];

    /**
     * Define the application's command schedule.
     *
     * @param  \Illuminate\Console\Scheduling\Schedule  $schedule
     * @return void
     */
    protected function schedule(Schedule $schedule)
    {
        $schedule->job(new UpdateCurrentKilometers)->weekly();
    }
```

Figura 35. “Scheduled Task”.

Fuente: elaboración propia.

```php
class UpdateCurrentKilometers implements ShouldQueue {
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

    /**
     * Create a new job instance.
     *
     * @return void
     */
    public function __construct() {
        //
    }

    /**
     * Execute the job.
     *
     * @return void
     */
    public function handle() {
        $today = Carbon::now();
        foreach(Vehicle::all() as $vehicle) {
            $this->updateVehicleTrackingData($vehicle, $today);
            $this->updateStandardMaintenanceTaskTrackingData($vehicle);
            $this->updateSpecificMaintenanceTaskTrackingData($vehicle);
        }
        return;
    }
```

Figura 36. “Sckeduled Task Job”.

Fuente: elaboración propia.

Una vez llamado por la tarea programada, el “job” ejecutará su función “handle”, que en nuestro caso ejecuta varias funciones privadas para actualizar kilometrajes y fechas, así como valorar si una tarea está pasada de fecha o kilometraje para añadirla a la tabla “expiredMaintenanceTasks”.

3.4. BASE DE DATOS

Laravel gestiona las tablas de las bases de datos utilizando lo que denomina “migrations”, que son las clases que encontraremos en los archivos situados dentro de “/database/migrations”, que funcionan como una especie de sistema de control de versiones de cada tabla de base de datos. Debido a que la cantidad de archivos en este directorio tiende a crecer bastante según se va desarrollando el back end, es recomendable agrupar por carpetas las “migrations” pertenecientes a la misma tabla, para poder trabajar mejor con ellas. En la siguiente figura 37 se mostrará la estructura de una “migration”.

```php
class CreateVehiclesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('vehicles', function (Blueprint $table) {
            $table->id();
            $table->enum('type', ['motorcycle', 'car', 'truck']);
            $table->string('fuel_type')->nullable();
            $table->dateTimeTz('first_acquisition_date');
            $table->integer('current_kilometers');
            $table->dateTimeTz('current_kilometers_update_date');
            $table->integer('weekly_kilometers');
            $table->string('brand_name')->nullable();
            $table->string('model')->nullable();
            $table->string('description')->nullable();
            $table->longText('image')->nullable();
            $table->foreignId('owner_id')
                ->references('id')
                ->on('vehicle_owners')
                ->constrained()
                ->onUpdate('cascade')
                ->onDelete('cascade');
            $table->timestamps();
        });
    }
```

Figura 37. “migration” de la tabla “vehículos”.

Fuente: elaboración propia.

Se intuye cómo la “migration” define la estructura de una tabla SQL, incluyendo las claves extranjeras para las relaciones entre tablas.

4. MODELO DE NEGOCIO

En este capítulo presentamos un modelo de negocio basado en la estrategia de un producto mínimo viable (MVP), a partir de la cual obtener información ("feedback") de su funcionamiento en el mercado real. Una vez probado y mejorada la propuesta de valor y el modelo de negocio en su conjunto, éste podría ser redimensionado a una escala mucho mayor. En definitiva, de lo que se trata es seguir el método "Lean Startup", propuesto por Eric Ries (Ries, 2011), e ir evolucionando el negocio mediante la innovación continua. Por tanto, inicialmente nos centraremos en la puesta en marcha del negocio en Gran Canaria, dejando para el futuro la expansión a otros mercados nacionales y, en su caso, internacionales en función del éxito alcanzado y las mejoras incorporadas.

La estructura de la propuesta que realizamos se basa en la herramienta "lienzo del modelo de negocio" ("business model canvas", BMC) desarrollada por Yves Pigneur y Alexander Osterwalder (Osterwalder, 2010).

4.1. ANÁLISIS DE LA COMPETENCIA

Una de las primeras tareas que se debe realizar cuando se pretende lanzar un producto al mercado es asegurar que todavía no existe otro producto que ofrezca los mismos beneficios, pudiendo así entrar en un nicho de mercado con necesidades insatisfechas. Para este propósito se han analizado distintas aplicaciones web y móviles recomendadas por páginas especializadas y por la "app store" de Android.

Las páginas web especializadas que se han consultado han sido "Motor Mapfre" (Mapfre, 2020) y "Auto10" (auto10, 2020), encontrando en ellas los siguientes competidores potenciales. A continuación, los mostramos junto con los motivos por los que ninguno de ellos ha sido una razón para no desarrollar ***Automatic Workshop*** (ver tablas 1-6).

aCar	Motivos por los que se puede competir con esta aplicación
(Play Store, 2020)	Está mal traducida al español
	Diseño anticuado
	Complicada de manejar (UX/UI)
	Pregunta demasiadas cosas innecesarias a la hora de añadir un vehículo que, además, el usuario no tiene porqué saber (tipo de motor, transmisión, tracción...)
	Usa terminología no amigable, como abreviaciones y siglas de palabras en inglés
	No sugiere los mantenimientos típicos que hay que hacerle a un vehículo, el usuario tiene que configurar todos los recordatorios manualmente
	No integrada con talleres

Tabla 1. aCar.

Fuente: elaboración propia.

Torque	Motivos por los que se puede competir con esta aplicación
(Play Store, 2020)	No es para gestionar el mantenimiento de un vehículo, es para conectar un OBD al bluetooth de un smartphone y recibir métricas sobre el estado general del vehículo
	No integrada con talleres

Tabla 2. Torque.

Fuente: elaboración propia.

MyGarage	Motivos por los que se puede competir con esta aplicación
(Play Store, 2020)	El display de la app funciona mal, los bloques de texto aparecen cortados
	No tiene alertas, solo permite añadir notas sobre el mantenimiento que se le ha realizado a un vehículo
	No integrada con talleres

Tabla 3. My Garage.

Fuente: elaboración propia.

Vehic	Motivos por los que se puede competir con esta aplicación
(Play Store, 2020)	Es una especie de red social o foro sobre el mundo de los coches, no es una app para gestionar el mantenimiento de un vehículo
	La mayoría de los posts están en árabe, no se encuentra contenido en inglés o español
	No integrada con talleres

Tabla 4. Vehic.

Fuente: elaboración propia.

Car Controller	Motivos por los que se puede competir con esta aplicación
(Apple, 2020)	Solo disponible para iOS
	Parece más orientada a la navegación que a la gestión del mantenimiento
	No integrada con talleres

Tabla 5. Car Controller.

Fuente: elaboración propia.

Drivvo	Motivos por los que se puede competir con esta aplicación
(Play Store, 2020)	No tiene recordatorios para gestionar el mantenimiento
	No integrada con talleres

Tabla 6. Drivvo.

Fuente: elaboración propia.

Por otro lado, en las recomendaciones de Google Play, encontramos los dos competidores más interesantes, cuya información se muestra en las siguientes tablas 7 y 8.

My Car - Administración del vehículo	Motivos por los que se puede competir con esta aplicación
(Play Store, 2020)	No permite personalizar los recordatorios y solo tiene 5 opciones sin el título "otro"
	No integrada con talleres

Tabla 7. My Car.

Fuente: elaboración propia.

Simply Auto - Mantenimiento y Seguimiento del Auto	Motivos por los que se puede competir con esta aplicación
(Play Store, 2020)	Mal traducida al español
	Si se quiere utilizar más de 9 recordatorios hay que pagar
	Diseño algo anticuado
	No integrada con talleres

Tabla 8. Simply Auto.

Fuente: elaboración propia.

Dados estos competidores, podemos ver en las siguientes figuras 38 y 39 un diagrama de pétalos de competidores y una matriz 2x2 que los clasifica por funcionalidades y facilidad de manejo.

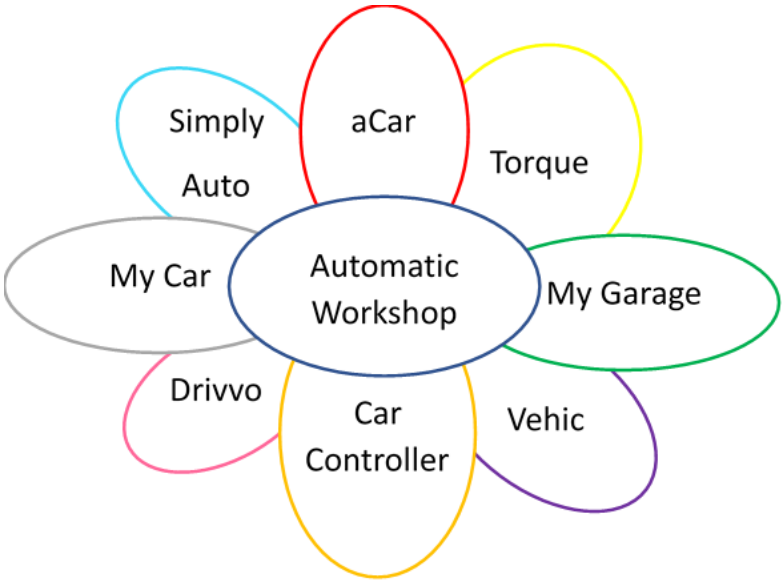

Figura 38. Diagrama de pétalos.

Fuente: elaboración propia.

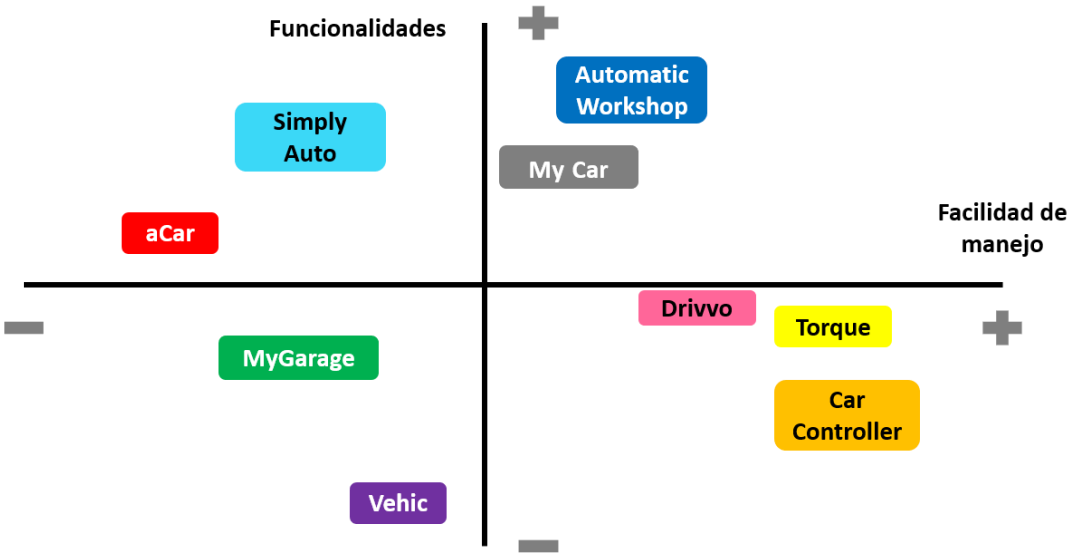

Figura 39. Matriz 2x2.

Fuente: elaboración propia.

4.2. SEGMENTOS DE CLIENTES

Una vez estudiada la competencia, es importante hacernos una idea sobre quiénes son los clientes a los que pretendemos satisfacer. Para ello hay que responder a las siguientes preguntas que se plantean:

- **¿Quiénes son los clientes?**

    - **Cliente industrial (propietarios de talleres con tamaño empresarial de PYME):** de este cliente se pretende obtener la mayoría de los ingresos del negocio, debido a que se le cobraría una cuota por el uso del software a cambio de aportarles una ventaja competitiva para que puedan medirse con las cadenas de talleres oficiales de las distintas marcas de vehículos.

    - **Cliente personal (propietarios de vehículos):** no se espera sacar muchos ingresos de ellos, como mucho una pequeña parte de los ingresos totales de la app, mediante publicidad no invasiva. El mayor beneficio que esperamos de ellos es tener una amplia red de usuarios que incentive a los talleres a adquirir el software.

- **¿Qué quieren los clientes?**

    - **Cliente industrial:** volverse más competitivos e incrementar los beneficios.

    - **Cliente personal:** ahorrar tiempo y dinero en cualquier actividad rutinaria.

- **¿Cuál es la propuesta de valor del producto?:** una aplicación web con la que se pretende facilitar la gestión del mantenimiento de el/los vehículo/s que una persona tenga en propiedad, así como aportar una ventaja competitiva a los propietarios de talleres mecánicos de tamaño empresarial pequeño o mediano, conectando a esos clientes con necesidades de realizar tareas de mantenimiento con los propietarios de talleres que posean el software.

- **¿Por qué estarían dispuestos a comprar/instalar? ¿Qué beneficios aportamos? ¿Qué dolores resolvemos?**

    - **Cliente industrial**

        - Visibilidad: cuando un usuario tiene vencido, o cercano al vencimiento el kilometraje o la fecha de alguna de las piezas de alguno de sus vehículos, los usuarios propietarios de talleres cercanos al propietario del vehículo recibirán una notificación al respecto. En caso de tener disponibilidad para realizar la tarea de mantenimiento que requiere el propietario del vehículo, podrán ofrecerle una cita en su taller. Esta información le llegaría al propietario del vehículo en un apartado de ofertas, pudiendo elegir cualquiera de los talleres que se ofrecen en la fecha que más le convenga. Esta sería una acción de marketing directo sobre un cliente potencial que probablemente estaría interesado en el servicio.

        - Incremento en la facturación: el propietario del taller podría fijarse en la cola de citas para realizar compras preventivas de piezas y recambios para no tener que esperar a que lleguen desde el almacén del proveedor. De esta forma, se permite completar antes cualquier reparación o tarea

de mantenimiento. Esto aumentaría la facturación del taller, debido a que la capacidad para almacenar vehículos de clientes es limitada, por lo que a veces se ven en la situación de no poder ofrecer servicios a algún cliente porque el taller está lleno y el cliente tiene que irse a un taller de la competencia. También aumentaría la facturación debido a que se atendería a más clientes en el mismo período de tiempo.

- **Cliente personal:** la aplicación les ahorraría tiempo al avisar mediante notificaciones de las tareas de mantenimiento pasadas de kilometraje o fecha de caducidad, en vez de tener que revisar periódicamente el libro de mantenimiento que típicamente se guarda en algún compartimento de almacenaje del vehículo (por ejemplo, la guantera) y evitaría despistes con las tareas de mantenimiento, que hacen que se tengan que realizar cuando ya ha pasado el kilometraje recomendado por el fabricante, lo que podría ocasionar desgastes prematuros de piezas y/o averías.

- **¿A qué tipo de mercado va dirigido el producto?:** a un mercado resegmentado, debido a que es un subconjunto o nicho de un mercado existente. El mercado sería el del software de gestión empresarial y el nicho, los talleres mecánicos constituidos como pequeña o mediana empresa.

- **¿Quiénes tienen los distintos roles comerciales en este mercado?**

  - **Decisor:** cliente industrial y cliente personal.

  - **Usuarios:** cliente industrial (usuario de la "subweb" para propietarios de talleres) y cliente personal (usuario de la "subweb" para propietarios de vehículos).

  - **Comprador económico**: cliente industrial.

  - **Influencer y recomendador:** cliente personal (puede recomendar tanto a otros clientes personales como a los propietarios de sus talleres habituales).

- **Saboteador:** apps de la competencia que quieran diversificarse para copiar nuestra propuesta de valor.

4.3. FUENTES DE INGRESOS

Una vez conocidos nuestros clientes, toca responder a una de las preguntas más importantes de cualquier modelo de negocio: ¿cómo generará ingresos nuestra empresa? Los siguientes subapartados serán los que nos respondan detalladamente a esta pregunta.

- **Precios de las distintas fuentes de ingresos**

  - **Publicidad:** se pretende conseguir una pequeña rentabilidad de los usuarios propietarios de vehículos mediante la visualización de anuncios contratados con AdSense. En la página oficial AdSense nos ofrece una estimación de cuánto se ganaría mediante la publicidad colocada en nuestras páginas, en base a la región de la audiencia y al contenido de la web (AdSense, 2021). La estimación que obtenemos con nuestros datos la podemos observar en la siguiente figura 40.

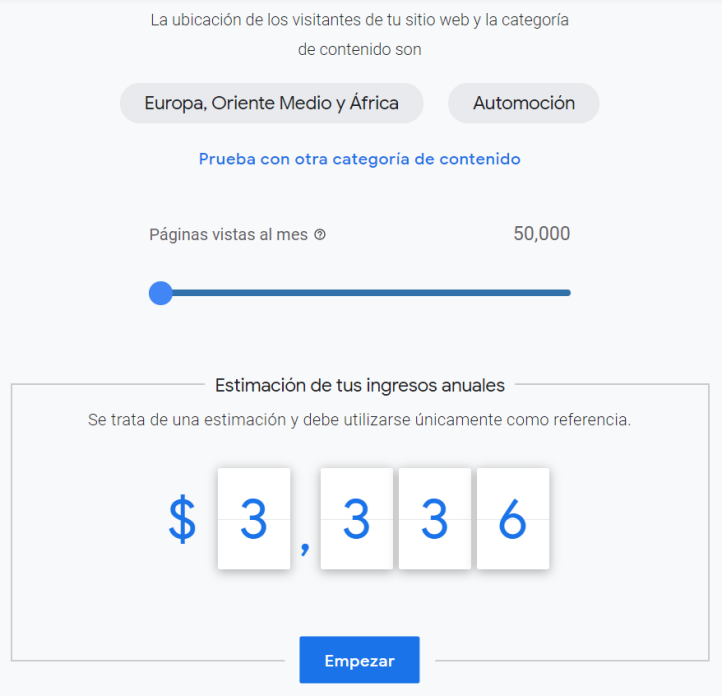

Figura 40. Estimación de ingresos publicitarios.
Fuente: AdSense (Google).

A partir de los datos obtenidos en esta estimación, podemos ver que los ingresos por visita serían 3.336 / (50.000*12) = 0,00556 dólares, hay que tener en cuenta que se cuenta la visita por cada vez que se carga un anuncio, por lo que cada vez que un usuario acceda a la web ocurrirán varias visitas dependiendo de lo que haga. Estimando que un usuario típico podría acceder a la web 2 veces al mes para comprobar el estado de sus vehículos y que, por cada vistita a la web se cargan 8 anuncios de media (por el flujo de secciones por las que debe pasar para encontrar la información que le interesa), obtenemos que de un usuario típico se obtienen 0,00556*8*2 = 0,08896 $ mensuales, que equivalen a 1,06752 $ anuales.

- **Cuotas de propietarios de vehículos:** se plantea la posibilidad de que los propietarios de vehículos que no deseen visualizar publicidad en la web puedan abonar una cuota que permita suplir los ingresos generados por dicha publicidad, más un cierto margen de beneficio. Teniendo en cuenta el cálculo anterior, se ha decidido que esta cuota sea de 1,5 € anuales, una cuota bastante asumible que supera casi en un 50% a los ingresos anuales estimados por publicidad.

- **Cuotas de propietarios de talleres:** de esta fuente es de la que se pretende obtener la mayor parte de los ingresos, debido a que a los talleres se les proporciona un servicio que les permitiría incrementar considerablemente su facturación. Sin embargo, estos ingresos llegarían más adelante en el tiempo, en una segunda fase de despliegue de la web, con todas las funcionalidades relativas a este tipo de usuario refinadas, de forma que se pueda satisfacer sus necesidades aportando la calidad de servicio esperada. Por estos motivos, esta cuota sería considerablemente más alta que la cuota para propietarios de vehículos que no desean visualizar publicidad, probablemente en torno a unos 25 € mensuales, aunque dependerá de varios factores, entre ellos, el comportamiento de las variables del macroentorno y el microentorno, la disposición a pagar de los propietarios de talleres, etc.

- **Estrategia de ingresos:** la estrategia de ingresos consiste en conseguir una buena base de propietarios de vehículos en la primera fase de lanzamiento del producto debido a que éstos, no solo proporcionarán ingresos, si no que serán los principales influenciadores que incentivarían más adelante a los propietarios de talleres a adquirir el producto.

- **Tácticas de precios:** lo más relevante sobre las tácticas de precios que se puede contar es que, por el lado de los propietarios de vehículos, parece lógico que se pueda usar la web de forma gratuita, a cambio de visualizar contenido publicitario, debido a que la sociedad está acostumbrada a usar infinidad de productos digitales de esta forma (YouTube, Gmail, Twitch, Google…) y el simple hecho de tener que abonar una cuota periódica para poder usar un producto digital, probablemente reducirá desmesuradamente la cantidad de usuarios que accederán al mismo, provocando que la web tenga menos tráfico y por tanto, menos capacidad para aparecer en mejor posición en los motores de búsqueda, menos atractivo para los propietarios de talleres y peor capacidad para negociar con aliados clave (esto último, debido a que algunas puertas solo se abren cuando existe cierto número de usuarios en una plataforma).

- **Cómo llega el dinero desde el cliente/usuario a la empresa:** en el caso de los anuncios que aparecen en ***AutomaticWorkshop***, el cobro por los anuncios contratados con AdSense se gestiona automáticamente a través del sistema informático de Google. Por otro lado, los cobros de las cuotas de los dos tipos de usuarios habría que gestionarlo integrando una plataforma intermediaria entre la web y una cuenta bancaria, como podría ser “Stripe”, que es la más usada para este tipo de operaciones.

4.4. RECURSOS FÍSICOS Y HUMANOS

Este apartado del modelo de negocio nos ilustra qué recursos físicos y humanos serán necesarios para el correcto funcionamiento de ***Automatic Workshop*** una vez se haya lanzado al mercado.

- **Recursos necesarios para realizar los procesos y subprocesos productivos y serviductivos:** entendemos como proceso productivo a toda actividad destinada al

desarrollo, mantenimiento, despliegue y actualización del producto. En la siguiente figura 41 se muestra un esquema de las principales actividades del proceso productivo.

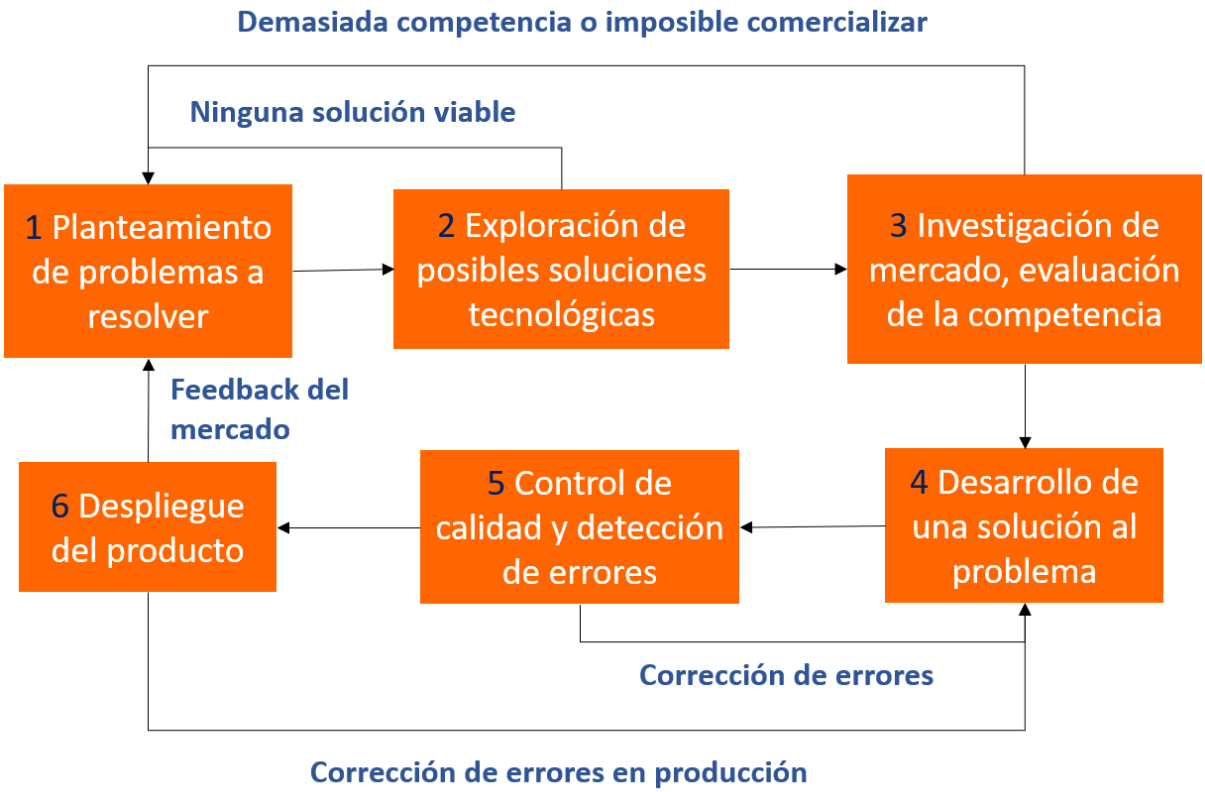

Figura 41. Actividades del proceso productivo.

Fuente: elaboración propia.

Las tareas 4 y 6 estarían parcialmente subcontratadas debido a que se necesita un conocimiento experto (al nivel de lo que en la industria del desarrollo de software se denomina “Programador Senior”) para asegurar que el software cumple con los criterios mínimos de seguridad, calidad y eficiencia, así como para desplegarlo correctamente mediante un proveedor de tecnología “serverless” (es decir, una empresa que ofrece en alquiler servidores y bases de datos gestionados por ella misma).

La organización que probablemente sería ideal para realizar estas tareas de forma subcontratada sería la empresa “Calima Solutions”, debido a que:

1. Las tecnologías principales que dominan sus empleados son las mismas que las del producto mínimo viable previamente desarrollado (Laravel + Vue.js).

2. Al ser una empresa de nueva creación, sus tarifas son muy inferiores a la media del sector.

3. Está liderada por un desarrollador con más de 5 años de experiencia laboral, que ha desarrollado proyectos para varias empresas, entre ellas, Caixa Bank, Hyundai y BP y ha lanzado varios productos propios. Además, es experto en SEO técnico, una disciplina del desarrollo web que consiste en conocer y aplicar las mejores prácticas requeridas por los navegadores y buscadores para que en el momento de realizar una búsqueda en cualquier buscador, las páginas con mejores prácticas de SEO sean las que se muestren primero.

4. Sus oficinas están situadas en el campus de Tafira. La cercanía en el mundo del desarrollo es algo que siempre se agradece, debido a que ayuda a minimizar los malentendidos, que son la principal causa de los retrasos en las entregas.

Los procesos serviductivos en este caso, serían las actividades destinadas a promocionar el producto y los servicios que permiten que los clientes puedan consumirlo, descritos en las siguientes figuras 42 y 43.

Figura 42. Servicio de alojamiento web.
Fuente: elaboración propia

Como proveedor de los servicios de administración de servidores, sistemas y redes (comúnmente conocidos como “servicios de alojamiento” o “hosting”), elegiríamos Amazon Web Services por ser extremadamente fiable, uno de los proveedores “serverless” que ofrecen la mejor relación calidad precio, lo que le ha llevado a que *el*

5,8% de todo el tráfico de internet (w3techs, 2021), así como *el 34% del tráfico "serverless"* (builtwith, 2021) estén alojados en este proveedor de servicios.

Figura 43. Actividades de promoción.

Fuente: elaboración propia.

- **Recursos necesarios para las actividades de promoción del producto:** en cuanto a los servicios de promoción que necesitaría el producto para darse a conocer, dividimos estos en tres categorías, debido a la finalidad que se pretende conseguir con cada forma de promocionar el producto.

    - Venta personal: la finalidad que persigue esta forma de promoción es la de mostrar el producto a los usuarios de los que se pretende obtener la mayor parte de los ingresos, los propietarios de talleres, una vez se hayan desarrollado las funcionalidades que permiten resolver sus problemas. Para ello es importante que un vendedor acuda personalmente al lugar en donde se encuentran, teniendo así la posibilidad de explicarles detalladamente el producto, resolver dudas y escuchar las sugerencias para transmitirlas a los desarrolladores. Se recalca que los vendedores deberían estar subcontratados o bajo contrato mercantil, debido a que, para la primera fase de lanzamiento del producto, después de haber hecho contacto con los talleres más importantes de la isla, el nivel de actividad de las tareas de ventas caería considerablemente.

- Publicidad: el objetivo de la publicidad es el de atraer usuarios masivamente en España. Los usuarios que podrían interesarse por el producto deberían ser aquellos a los que les llama la atención el mundo del motor y tener sus vehículos en buenas condiciones y a lo que no se les haga extraño gestionar tareas mediante aplicaciones. Por este motivo, se ha elegido como medio de transmisión el canal de YouTube de la empresa "coches.net", ya que al ser un medio que se consume de manera digital, se reduce la probabilidad de colisionar con personas no adaptadas al manejo de la tecnología y, debido a la temática del canal, es razonable que encontremos personas interesadas pro el mundo del motor.

- Relaciones públicas: acudir a ferias industriales y eventos relacionados con el sector, no es solo interesante por conseguir nuevos usuarios para la plataforma, si no por hacer contactos con potenciales aliados, socios, proveedores y demás "stakeholders" que podrían facilitar el rumbo del negocio de formas no planteadas previamente, así como ofrecernos algún trato interesante.

- **Localización de la empresa**

  - **Macrolocalización:** la opción más razonable parece que la empresa tenga su domicilio social en Las Palmas de Gran Canaria, debido a que es donde el fundador reside, dónde se encuentra "Calima Solutions" (la empresa que podría ser un aliado clave) y debido a que esta decisión de macrolocalización podría permitir en un futuro acceder a las ventajas fiscales de la Zona Especial Canaria (ZEC).

  - **Microlocalización:** las características de este negocio no plantean la necesidad de la posesión de instalaciones físicas en las que desempeñar el trabajo. Por tanto, la opción elegida es no contar con instalaciones físicas específicas, considerando que:

    - No alquilar una oficina o espacio de coworking supone un importante ahorro de costes.

        - Los únicos empleados que se plantea contratar serían los vendedores, que trabajarían mayoritariamente de manera temporal y en la calle. Para los momentos en los que sea necesario realizar una reunión, la reunión tendría lugar por videoconferencia y cualquier otro tipo de comunicación con los vendedores puede tener lugar de forma telemática.

        - El contexto actual (pandemia provocada por la COVID-19) incita a adoptar una forma de trabajo más orientada al "home office", para minimizar los riesgos de contagio.

- **Recursos materiales:** no se prevé la necesidad de que la empresa posea recursos materiales debido a que para desarrollar software y mantenerlo, es suficiente con que quienes realizan estas tareas dispongan de un ordenador con conexión a internet y a que, como los servicios de mantenimiento de servidores y redes se van a subcontratar, la posesión de estos elementos no es necesaria.

- **Recursos humanos**
    - **Fundador de la empresa**
        - **Tareas a realizar:** desarrollo, mantenimiento y despliegue del producto con apoyo de la empresa subcontratada, RRHH, marketing, toma de decisiones estratégicas, demás tareas relativas a la gestión del negocio.

        - **Retribución:** ninguna. Trabajaría como empresario autónomo y su retribución sería los beneficios que genere la empresa.

        - **Tipo de jornada:** jornada completa.

    - **Vendedores (trabajarían como autónomos a comisión por ventas):**
        - **Tareas a realizar:** dar a conocer e intentar vender el producto a propietarios de talleres de la isla de Gran Canaria.

        - **Personas necesarias:** dependiendo del presupuesto con el que se cuente, pero probablemente, en torno a unos 4 vendedores que permitan cubrir las zonas más rentables de la isla.

        - **Tipo de jornada:** jornada completa.

        - **Perfil necesario:** experiencia y formación en ventas, carné de conducir y vehículo propio y disponibilidad de ordenador con conexión a internet.

    - **Desarrolladores de software (subcontratados a Calima Solutions):**
        - **Tareas a realizar:** desarrollar, mantener, actualizar y desplegar el producto.

        - **Personas necesarias:** entre 1 y 2, dependiendo del presupuesto y el volumen de trabajo.

        - **Tipo de jornada:** jornada completa o media jornada, las empresas de desarrollo de software suelen trabajar en varios proyectos a la vez, por lo que no es extraño que no trabajen a jornada completa en este. Esto no supone un gran problema debido a que su trabajo sería apoyar al fundador con las tareas de desarrollo con las que menos experiencia tiene y aportar sugerencias técnicas para mejorar el producto.

        - **Perfil necesario:** conocimientos generales de desarrollo web, conocimientos de Laravel, Vue, MySQL y AWS y experiencia desplegando servicios web.

- **Organigrama:** teniendo en cuenta la información anterior, podemos formar el organigrama del negocio que se muestra a continuación en la figura 44.

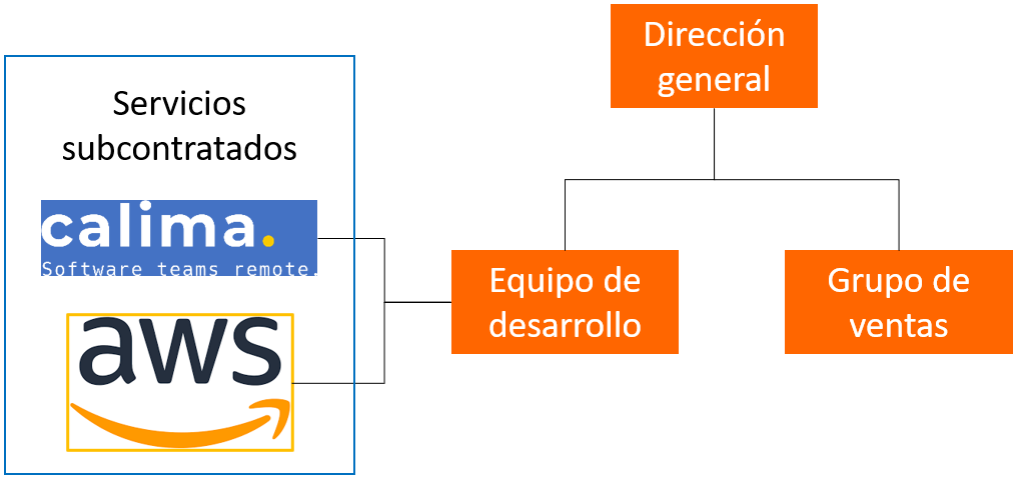

Figura 44. Organigrama de Automatic Workshop.

Fuente: elaboración propia.

- **Capacidad productiva**

  - **Potenciales cuellos de botella:** los cuellos de botella pueden ocurrir en cualquier fase del proceso de desarrollo, desde el diseño de las nuevas funcionalidades que se vayan a añadir, pasando por el desarrollo, el control de calidad y hasta la fase de despliegue.

    No se prevé que ocurran cuellos de botella en las tareas de ventas, debido a que es lógico que el personal de ventas tenga que esperar a que el producto esté desplegado para poder ofrecerlo, pero el equipo de producción no tenga que esperar nada para mejorar el producto o arreglar posibles defectos. Un software nunca está en perfecto estado y siempre se pueden realizar tareas menos críticas en los tiempos muertos (escribir documentación, investigar mejores formas de programar alguna funcionalidad, mejorar la eficiencia…).

  - **Capacidad de producción máxima por unidad de tiempo:** al encontrarnos ante un software que no necesita ser reprogramado para adaptarlo a cada cliente al que se entrega (como sucede con las aplicaciones de ofimática, las aplicaciones de diseño asistido por computador, los videojuegos, los sistemas operativos…), la capacidad de producción se puede definir como el conjunto de nuevas funcionalidades añadidas al producto en cada “release”. Como en la empresa se aplicaría la metodología de trabajo “scrum”, suponiendo que una

"release" consta de cuatro "sprints" y un "sprint" dura dos semanas, podríamos decir que se tiene la capacidad de entregar una "release" cada dos meses.

4.5. RECURSOS FINANCIEROS Y COSTES

Después de mencionar los recursos físicos y humanos necesarios para el funcionamiento de la empresa, se hace necesario indicar cuáles serán los costes asociados a dichos recursos y cómo se pretenden financiar.

- **Costes**
  - **Vendedores:** trabajarían a comisión, las ventas a los talleres se producirían en un momento en el que el producto ya tuviera una acogida considerable entre los propietarios de vehículos (de otro modo no habría incentivos para los talleres para adquirir el producto). Existen múltiples fenómenos que hacen que no tenga mucho sentido intentar averiguar el coste en estos momentos, entre ellas:

    - No se contratarían en un corto plazo y las circunstancias actuales (crisis del coronavirus, inflación, etc.) hacen que la realidad económica sufra cambios drásticos incluso en pequeños períodos de tiempo, por lo tanto, es bastante seguro que la situación del entorno en el momento de contratar a los vendedores sea muy distinta a la actual.

    - Las comisiones dependerían en gran parte de las negociaciones que se lleven a cabo en el momento de la contratación y de si se decide optar por externalizar al equipo de ventas o contratarlos mediante contrato mercantil o laboral.

    - Las decisiones sobre la contratación de vendedores se verán influenciadas por la acogida de la web entre los propietarios de vehículos, es probable que cuando la web tenga un número de usuarios considerable, se les anime a realizar una encuesta sobre lo necesario que ven que puedan recibir ofertas de sus talleres más cercanos, en caso

de que la encuesta no arroje resultados positivos, lo normal será que no se prepare esa funcionalidad para lanzarla a producción y ni siquiera se llegue a contratar a un equipo de ventas.

- **Anuncios en YouTube (coches.net):** debido a lo variable que es la relación entre el coste y el número de impactos (Google, 2021), antes que fijar un objetivo de impactos y luego averiguar su precio es preferible fijar un presupuesto, averiguar los impactos que se consiguen con el mismo y luego tomar decisiones en base a los resultados. Se puede proponer un presupuesto inicial de 120 € (recordamos que al principio solo queremos que el anuncio aparezca en Gran Canaria) para los primeros 3 meses después de tener la web en producción.

- **Amazon Web Services:** en las primeras etapas de la vida de la aplicación se intentará utilizar la infraestructura más económica disponible, gracias a que "cloudcraft.co" permite calcular los costes de los servicios de AWS por adelantado, podemos ver que el coste del proveedor "serverless" será de unos 25,71 € al mes, tal como se puede apreciar en las siguientes figuras 45 y 46.

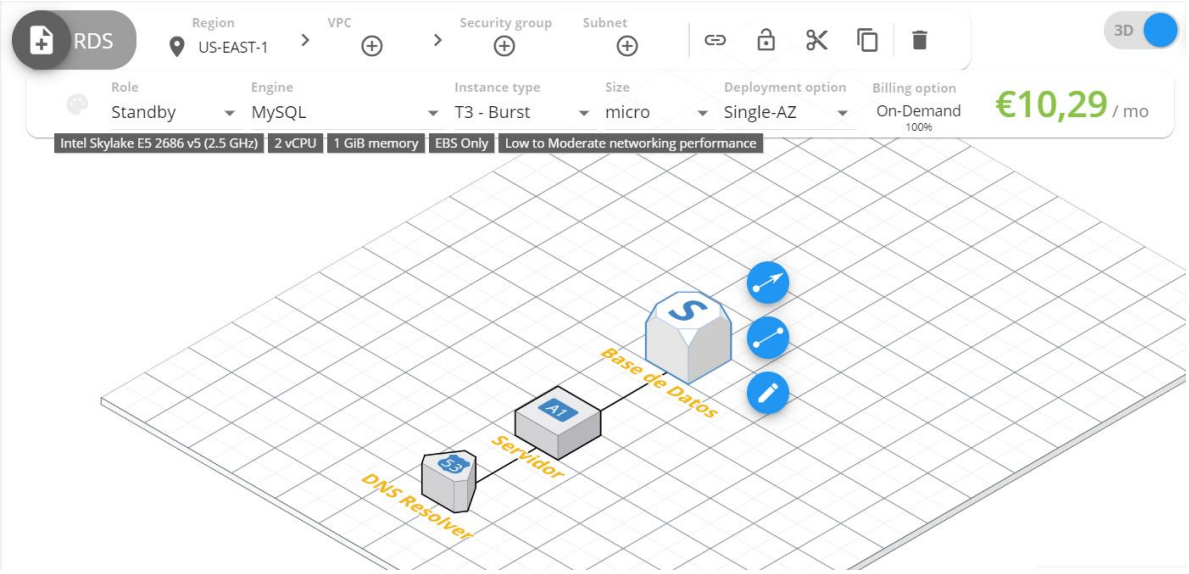

Figura 45. Arquitectura "serverless" de "Automatic Workshop".

Fuente: elaboración propia en "cloudcraft.io".

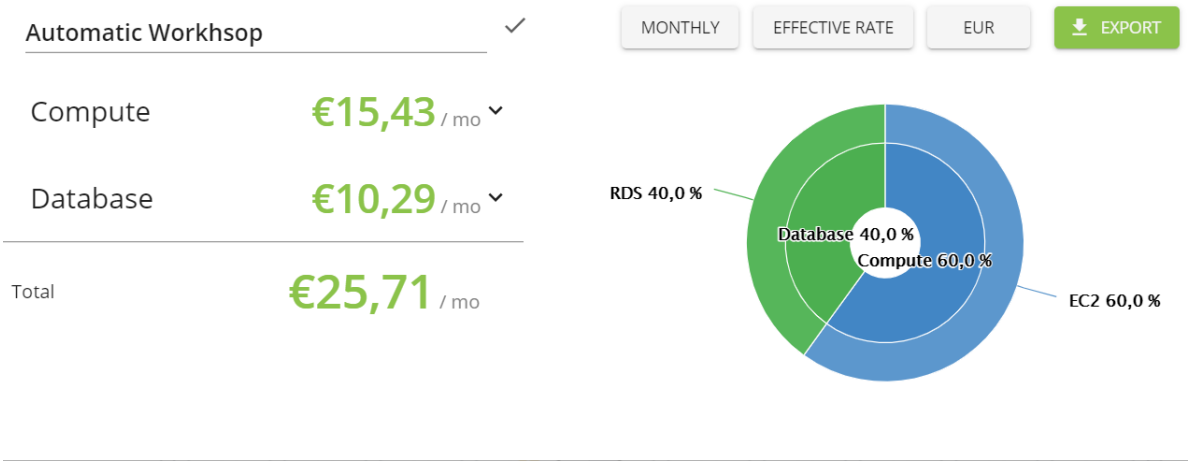

Figura 46. Detalle del presupuesto "serverless".

Fuente: elaboración propia en "cloudcraft.io".

- **Calima Solutions:** la tarifa actual de la empresa para clientes de Canarias es de 35 € por hora de trabajo, para tener la web funcionando para los propietarios de vehículos en un entorno real, harían falta unas 60h de trabajo por parte de esta empresa, por lo que se deduce un coste de 2.100 € en concepto de desarrollo.

- **Cuota de autónomos:** como el fundador no ha estado dado de alta como autónomo previamente, podría acogerse a la tarifa plana de 60 € anuales durante un período comprendido entre 2 y 3 años (BOE, 2021).

- **Ingresos:** como se ha explicado en el apartado "fuentes de ingresos", esperamos facturar 0,08896 $ mensuales por cada usuario propietarios de vehículos que visualice publicidad en la plataforma, 1,5 € anuales por cada usuario propietario de vehículos que pague la cuota que le permite usar la plataforma sin visualizar publicidad y 25 € mensuales por cada usuario propietario de taller.

- **Financiación:** teniendo en cuenta toda la información anterior, las necesidades de financiación que se pueden deducir para los dos primeros años (24 meses) serían las siguientes:

- Amazon Web Services: 617,04 € (25,71*24).
- Calima Solutions: 2.100 €.
- Anuncios en YouTube: 120 €.
- Cuota de autónomos: 120 € (60*2).

Lo que nos indica que debería realizarse una inversión inicial de 2.957,04 €. Para realizar este desembolso, se evitaría pedir préstamos o créditos para no tener que abonar los gastos producidos por los intereses, y se intentaría obtener todas las subvenciones posibles para minimizar el riesgo, como son las subvenciones nacionales para "startups" (Bankia, 2021), que incluyen las del Ministerio de Ciencia, Innovación y Universidades, las de la Empresa Nacional de Innovación del Ministerio de Industria, Energía y Turismo, las del Centro para el Desarrollo Tecnológico e Industrial y las del Instituto de Crédito Oficial. Por otro lado, también se intentaría acceder a las subvenciones que ofrecen distintas instituciones en la web del Gobierno de Canarias (Gobierno de Canarias, 2021).

Adicionalmente, si resulta que no es posible obtener ninguna subvención, se buscarían otras formas de financiación con las que minimizar el riesgo, como los fondos de capital riesgo, Business Angels, inversores institucionales o particulares, etc.

4.6. ALIANZAS

En los apartados anteriores se ha mencionado a varias empresas y personas que serían aliados clave en el desarrollo de la actividad de ***Automatic Workshop***. En este apartado especificaremos quiénes son estos aliados y que características tienen sus alianzas con nosotros.

- **¿Qué aliados necesitamos y por qué?:** entre los aliados clave que tendremos, se encuentran los dos proveedores más importantes, Calima Solutions, empresa proveedora de servicios de desarrollo de software y coches.net, la empresa que posee

el canal de YouTube en el que pretendemos anunciarnos en las primeras etapas de vida de Automatic Workshop.
También estarían dentro de esta categoría los inversores o "business angels", si los hubiera, debido a que, si una persona u organización decide invertir en nuestro negocio, es evidente que hará lo que pueda para lograr que dicho negocio sea rentable, mientras que Automatic Workshop hará lo propio por sacar el mayor rendimiento posible del capital recibido.

- **¿Qué riesgos tienen estas alianzas?:** el riesgo principal sería que, a la hora de negociar con los proveedores o socios principales Automatic Workshop les parezca una buena idea y, a la vez, con poca complejidad a la hora de implementarla, lo que podría suponer un incentivo para copiarla y desarrollarla por su cuenta.
  Otro riesgo en el que cabe pensar es que desde coches.net nos pidan precios demasiado altos por publicitarnos, haciendo inviable el lanzamiento del producto.

- **¿Por qué estarían dispuestos a aliarse con nosotros?:** a los socios potenciales les podría interesar una alianza con Automatic Workshop porque supone acceder a un producto mínimo viable que evita los costes de unos seis meses de desarrollo, además de todo el trabajo hecho en el modelo de negocio y del capital intelectual que el creador ha adquirido durante todo el proceso, que sería de ayuda a la hora de plantear modificaciones o actualizaciones en el proyecto.
  Los proveedores clave, por otro lado, no necesitarían incentivos especiales para querernos como clientes, más allá de la contraprestación que recibirían por proporcionarnos sus servicios.

- **¿Qué coste puede suponer la alianza?**

    - **Proveedores:** el coste por la prestación de sus servicios que se encuentra detallado en el apartado "Recursos financieros y costes".

    - **Socios:** dependerá del capital a aportar negociado (C) y del valor que se le dé en la negociación (como activo intangible) al producto mínimo viable que se usaría como base sobre la que se desarrollaría Automatic Workshop (V).

Dependiendo de estos dos factores podríamos calcular la porción del negocio que les pertenecería como “(C/(C+V))*100” (resultado en tanto por ciento).

- **Diagrama de flujo de dinero en la relación con los aliados (ver figura 47)**

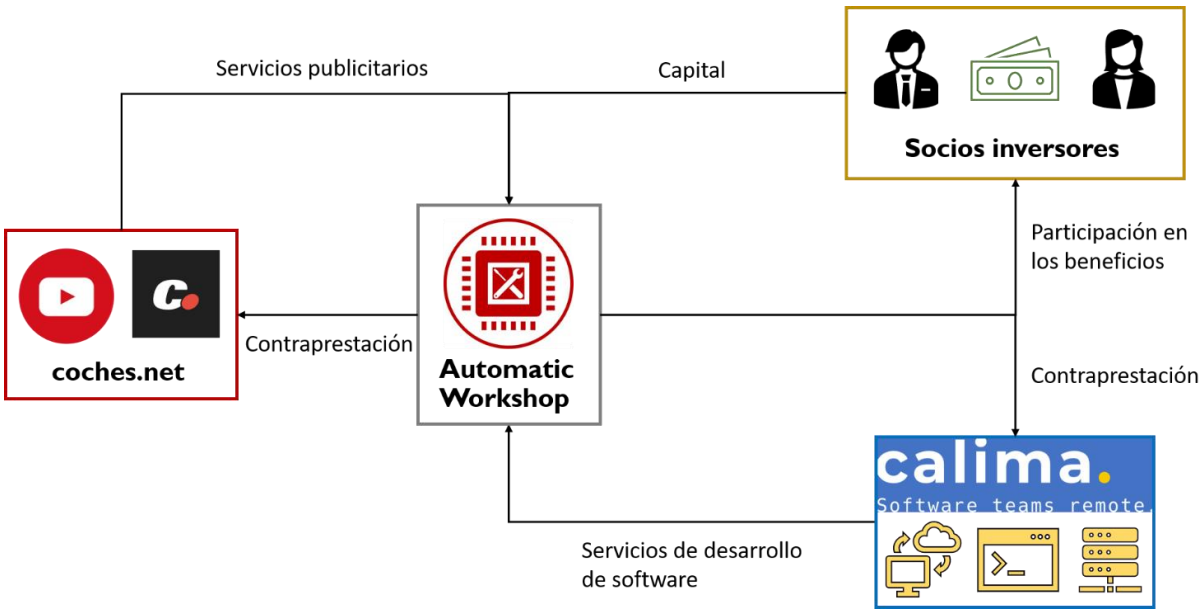

Figura 47. Flujo de dinero y servicios.

Fuente: elaboración propia.

4.7. RELACIÓN CON EL CLIENTE

Habiendo terminado con los apartados relacionados con el funcionamiento interno del negocio, se procede a detallar la relación que se pretende tener con el cliente con el objetivo de maximizar las ventas.

- **Definición del doble embudo de ventas:**

1. **Adquirir interés:**

1.1. Propietarios de vehículos: visualización de nuestros anuncios en coches.net

1.2. Propietarios de talleres: contacto con algún vendedor que hayamos contratado o subcontratado.

2. **Activar ventas:**

    2.1. Propietarios de vehículos: decidir crear una cuenta después de haber sido informados por los anuncios de YouTube de las ventajas de usar la web.

    2.2. Propietarios de talleres: considerar que contratar el producto puede ser beneficioso para su negocio después de que un vendedor se lo dé a conocer.

3. **Mantener clientes:** actualizar el producto periódicamente, proporcionando nuevas funcionalidades que aporten valor y escuchando siempre las quejas y sugerencias de los usuarios.

4. **Desagregar**: no se prevé alguna forma de proporcionar otros servicios distintos del uso de la propia web que puedan separarse en distintas categorías.

5. ***Up-Selling*:**

    5.1. Propietarios de vehículos: cuenta "premium" sin anuncios.

    5.2. Propietarios de talleres: se podría ofrecer una cuota anual, con una pequeña rebaja respecto de la mensual como primera opción.

6. **Ventas cruzadas:** al contar únicamente con Automatic Workshop en la cartera de productos, no existe posibilidad de hacer ventas cruzadas, al menos en los primeros momentos de vida de la empresa.

7. **Recomendaciones:** se añadirá un descuento a las cuentas para propietarios de talleres si invitan a otro taller a crear una cuenta.

- **Estimaciones sobre el CAC y VCV**

    - **CAC (Coste de Adquisición del Cliente:** de entre las opciones que Google nos proporciona para ubicar anuncios en sus plataformas, elegiremos la de coste por click, dado que nos interesa saber cuántas personas acceden a la web a crearse una cuenta. Debido al funcionamiento del coste por click en Google (Google, 2021), es prácticamente imposible estimar cuánto costará cada adquisición de cada cliente porque el coste se calcula especificando cuánto es lo máximo que se quiere pagar por el click para que luego, cuando un usuario de Google esté visualizando contenido en una de sus páginas, ocurra una subasta entre quienes se quieren anunciar dentro de ese contenido y quién finalmente acabe apareciendo como anuncio, será quién haya especificado una puja más alta. Debido a esta situación, se ha decidido realizar una inversión inicial de 120 € para luego observar sus efectos en la captación de clientes. Con esta inversión se espera conseguir al menos la creación de al menos, 60 cuentas en la web. Esto nos deja con un CAC de 120/60 = 2 €.

    - **VCV (Valor del Ciclo de Vida):** para los propietarios de vehículos, que serán los únicos usuarios que existirán en las primeras fases del proyecto, se ha estimado en apartados anteriores (Análisis de los Recursos Financieros) que aportarán por la publicidad visualizada 0.08896 $ mensuales. Siendo cierto que para amortizar el CAC, cada usuario de este tipo debería usar la web durante 2/0.08896 = 22.48 meses = 1.87 años (asumiendo que 1 USD = 1 EUR para simplificar los cálculos), los usuarios captados no aportan solo ingresos a la empresa, sino que también pueden incentivar a otras personas a crearse una cuenta y en el futuro, a los propietarios de los talleres a los que llevan sus vehículos, por lo que, se podría decir que a partir de un año y medio de uso **VCV>CAC.**

- **Acciones que se realizarán para obtener clientes y mantenerlos**

    - **Acciones para obtener nuevos clientes:** inversión publicitaria en YouTube y más adelante, contratación o subcontratación de vendedores para las cuentas de propietarios de talleres.

    - **Acciones para mantener los clientes:** como se ha dicho anteriormente, actualizar el producto periódicamente, proporcionando nuevas funcionalidades que aporten valor y escuchando siempre las quejas y sugerencias de los usuarios.

4.8. ANÁLISIS DE LA DISTRIBUCIÓN

Una vez señalada las acciones que se quieren emprender para atraer el mayor número de clientes posible y formar una relación de mutuo beneficio a largo plazo con ellos, procedemos a hablar de la distribución del producto.

- **Características del producto:** es un producto digital (aplicación web) que se ejecuta en los navegadores de los ordenadores personales, las “tablets” y “smartphones”.
  No se descarta que, en un futuro, si el producto funciona bien, se pueda desarrollar una aplicación móvil híbrida para que exista cierta usabilidad cuando los “smartphones” y las “tablets” no tengan conexión. Si la aplicación híbrida funciona bien, se podría explorar la posibilidad de desarrollar aplicaciones nativas para los sistemas Android e iOS.

- **Características de los canales:** Como se ha explicado en el apartado de “recursos físicos y humanos”, al enfocarnos en dos tipos distintos de clientes, habrá un canal de distribución para cada uno:

    - **Propietarios de talleres:** este tipo de cliente será el que se prevé que nos proporcionará el grueso de la facturación, por lo que se optará por utilizar la

venta personal, acudiendo al lugar de trabajo, para poder dar todo tipo de información, resolver dudas y captar sugerencias.

A pesar del coste que conlleva el uso de la venta personal, es recomendable que el esfuerzo de ventas sea especialmente intensivo con este tipo de clientes, no solo porque nos pueda proporcionar más ingresos, sino porque puede incentivar a los clientes de su taller a utilizar la web una vez él mismo ha adquirido una cuenta.

- **Propietarios de vehículos:** en las primeras etapas de vida del producto, buscaremos realizar una estrategia de penetración de mercado en el segmento en el que creemos que el producto puede tener una mayor aceptación y una adopción más rápida, es decir, contactar con los llamados "early adopters" de la teoría de difusión de la innovación (Boston University, 2021). Para lograrlo, atacaremos al público con una edad comprendida entre los 20 y 50 años (un público suficientemente maduro como para tener que preocuparse por el mantenimiento de su vehículo, pero suficientemente joven como para estar habituado al uso de medios informáticos), interesados por el mundo del motor y residentes España, que creemos que será el sector en el que se encontrarán la mayoría de los ya mencionados "early adopters". El medio que creemos que será adecuado para contactar a este segmento, es el canal de YouTube de "coches.net", en el que se suben regularmente vídeos mostrando los nuevos modelos de coches que llegan a España y Europa.

  Posteriormente, buscando ampliar la cuota de mercado, se publicarán anuncios para un público más general que también podría estar interesado en el producto debido a que al ser menos expertos en el mundo del motor, pueden tener más necesidad de usarlo para que se les haga menos complicado gestionar el mantenimiento de su vehículo. Para este público no cambiaría el rango de edad, debido a que los motivos para escoger el intervalo entre 20 y 50 años no cambian. Por otro lado, sí cambiaría el medio en el que se alojarían nuestros anuncios, que seguiría siendo un medio digital para garantizar que el receptor está familiarizado con el uso de smartphones y ordenadores personales, pero utilizando las páginas recomendadas por Google para anunciarnos debido a que es difícil que para este tipo de público la dirección de Automatic Workshop

tome mejores decisiones respecto a dónde alojar los anuncios que las inteligencias artificiales que utiliza Google para distribuir los anuncios en función de la información que tiene del receptor.

4.9. LIENZO DEL MODELO DE NEGOCIO

En este apartado resumimos, en la figura 48, el modelo de negocio propuesto utilizando la representación gráfica del lienzo (canvas) de Yves Pigneur y Alexander Osterwalder (Osterwalder, 2010).

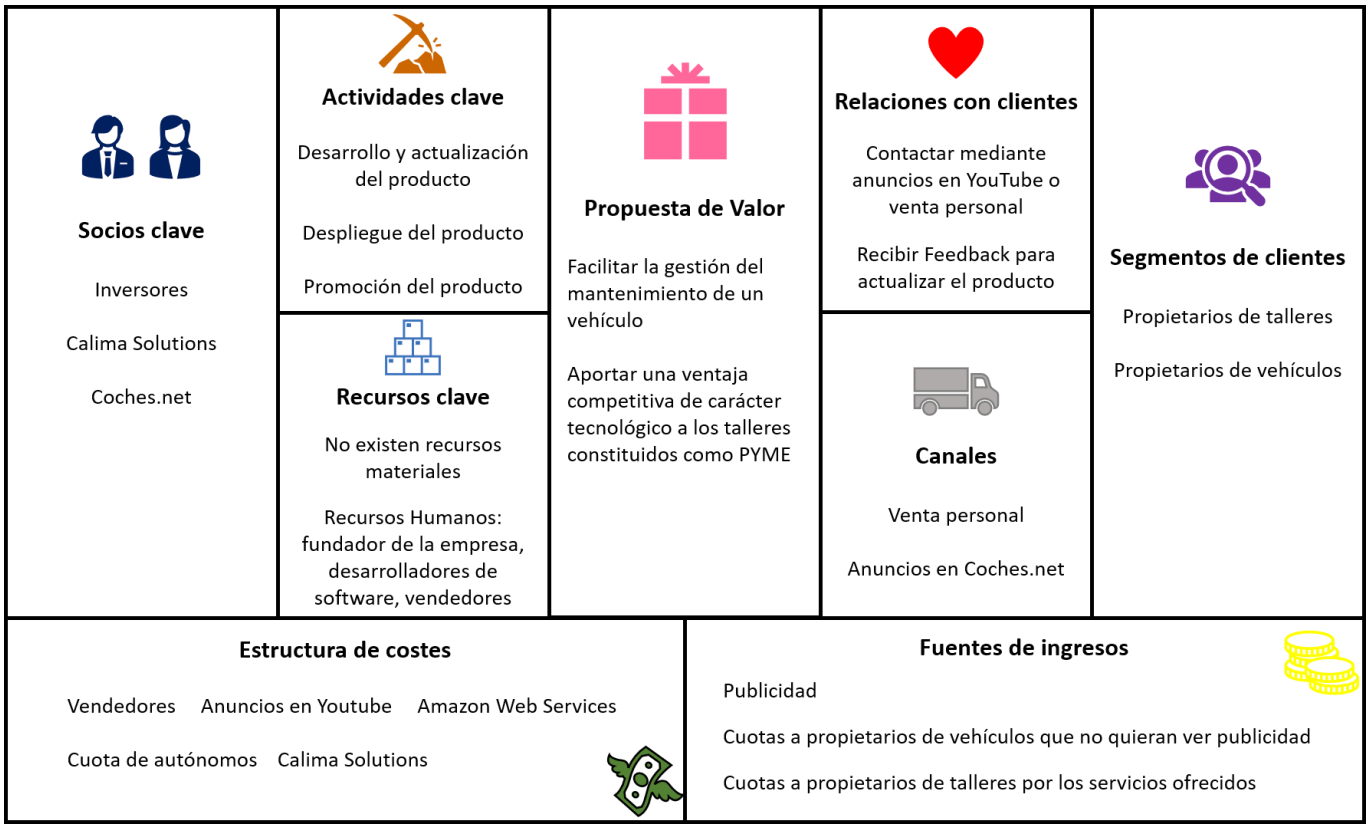

Figura 48. Lienzo del modelo de negocio.

Fuente: elaboración propia.

5. CONCLUSIONES Y TRABAJOS FUTUROS

La principal conclusión que se extrae de este trabajo es que no es necesaria una gran inversión inicial para crear productos digitales que mejoren las capacidades tecnológicas de las pequeñas y medianas empresas, ayudándolas así a minimizar el efecto que tienen las ventajas competitivas tecnológicas de las empresas de mayor tamaño, que pueden permitirse la adquisición de software hecho a medida con el objetivo de mejorar la eficiencia de sus procesos internos.

El trabajo futuro que se propone es, efectivamente, poner en marcha este modelo de negocio una vez se haya adaptado el producto para funcionar con los requisitos mínimos de calidad y seguridad y, partiendo de la propuesta mínima viable que contiene, mejorarlo y expandirlo. Probablemente, en esta segunda fase sí sea necesario disponer de una mayor inversión para lo que sería deseable conseguir inversores (business angels o capital riesgo) que apuesten por el proyecto.

6. BIBLIOGRAFÍA

AdSense. (10 de Mayo de 2021). *AdSense.* Obtenido de AdSense: https://www.google.com/adsense/start/

Apple. (18 de Julio de 2020). *Carcontroller.* Obtenido de App Store: https://apps.apple.com/es/app/carcontroller/id1466934442

auto10. (18 de Julio de 2020). *TOP 5 DE APLICACIONES PARA EL MANTENIMIENTO DEL COCHE.* Obtenido de auto10: https://www.auto10.com/reportajes/top-5-de-aplicaciones-para-el-mantenimiento-del-coche/6041

Bankia. (10 de Mayo de 2021). Obtenido de https://www.bankia.es/es/bankademia/mis-finanzas/ayudas-y-subvenciones/startups

BOE. (07 de 05 de 2021). *Boletín Oficial del Estado.* Obtenido de https://www.boe.es/buscar/doc.php?id=BOE-A-2018-17992

Boston University. (17 de 05 de 2021). *MPH Online Learning Modules.* Obtenido de https://sphweb.bumc.bu.edu/otlt/MPH-Modules/SB/BehavioralChangeTheories/BehavioralChangeTheories4.html#:~:text=Diffusion%20of%20Innovation%20(DOI)%20Theory,specific%20population%20or%20social%20system.

builtwith. (3 de Mayo de 2021). Obtenido de https://trends.builtwith.com/hosting/cloud-hosting/traffic/Top-10k

Gobierno de Canarias. (10 de Mayo de 2021). Obtenido de https://www.emprenderencanarias.es/servicios-de-apoyo/creacion-de-empresas-3/financiacion/

Google. (07 de 05 de 2021). *Ayuda de Google Ads.* Obtenido de https://support.google.com/google-ads/answer/2472735?hl=es

Laravel. (08 de Junio de 2021). *Laravel.* Obtenido de Laravel: https://laravel.com/docs/7.x

Laravel. (08 de Junio de 2021). *Laravel.* Obtenido de Laravel: https://laravel.com/docs/7.x/facades

Mapfre. (18 de Julio de 2020). *5 Apps para el Mantenimiento del Coche.* Obtenido de Motor Mapfre: https://www.motor.mapfre.es/consejos-practicos/consejos-de-mantenimiento/5-apps-para-el-mantenimiento-del-coche/

Material Design. (04 de Junio de 2021). *Material Design.* Obtenido de https://material.io/design/introduction

Mozilla. (08 de Junio de 2021). *HTTP request methods: MDN web docs.* Obtenido de developer.mozilla.org: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

Osterwalder, Y. P. (2010). *Business Model Generation.* Wiley.

Play Store. (18 de Julio de 2020). *aCar.* Obtenido de Google Play: https://play.google.com/store/apps/details?id=com.zonewalker.acar&hl=es_419&showAllReviews=true

Play Store. (18 de Julio de 2020). *Drivvo.* Obtenido de Google Play: https://play.google.com/store/apps/details?id=br.com.ctncardoso.ctncar&hl=es_419

Play Store. (18 de Julio de 2020). *My Car.* Obtenido de Google Play: https://play.google.com/store/apps/details?id=com.kineapps.mycar&hl=es

Play Store. (18 de Julio de 2020). *My Garage.* Obtenido de Google Play: https://play.google.com/store/apps/details?id=com.moremu.mygarage&hl=es_419&showAllReviews=true

Play Store. (18 de Julio de 2020). *Simply Auto.* Obtenido de Google Play: https://play.google.com/store/apps/details?id=mrigapps.andriod.fuelcons&hl=es

Play Store. (18 de Julio de 2020). *Torque Lite.* Obtenido de Google Play: https://play.google.com/store/apps/details?id=org.prowl.torquefree&hl=es_419

Play Store. (18 de Julio de 2020). *Vehic.* Obtenido de Google Play: https://play.google.com/store/apps/details?id=com.vehic.app&hl=es_419

Ries, E. (2011). *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses.* The Crown Publishing Group.

w3techs. (3 de Mayo de 2021). Obtenido de https://w3techs.com/technologies/history_overview/web_hosting