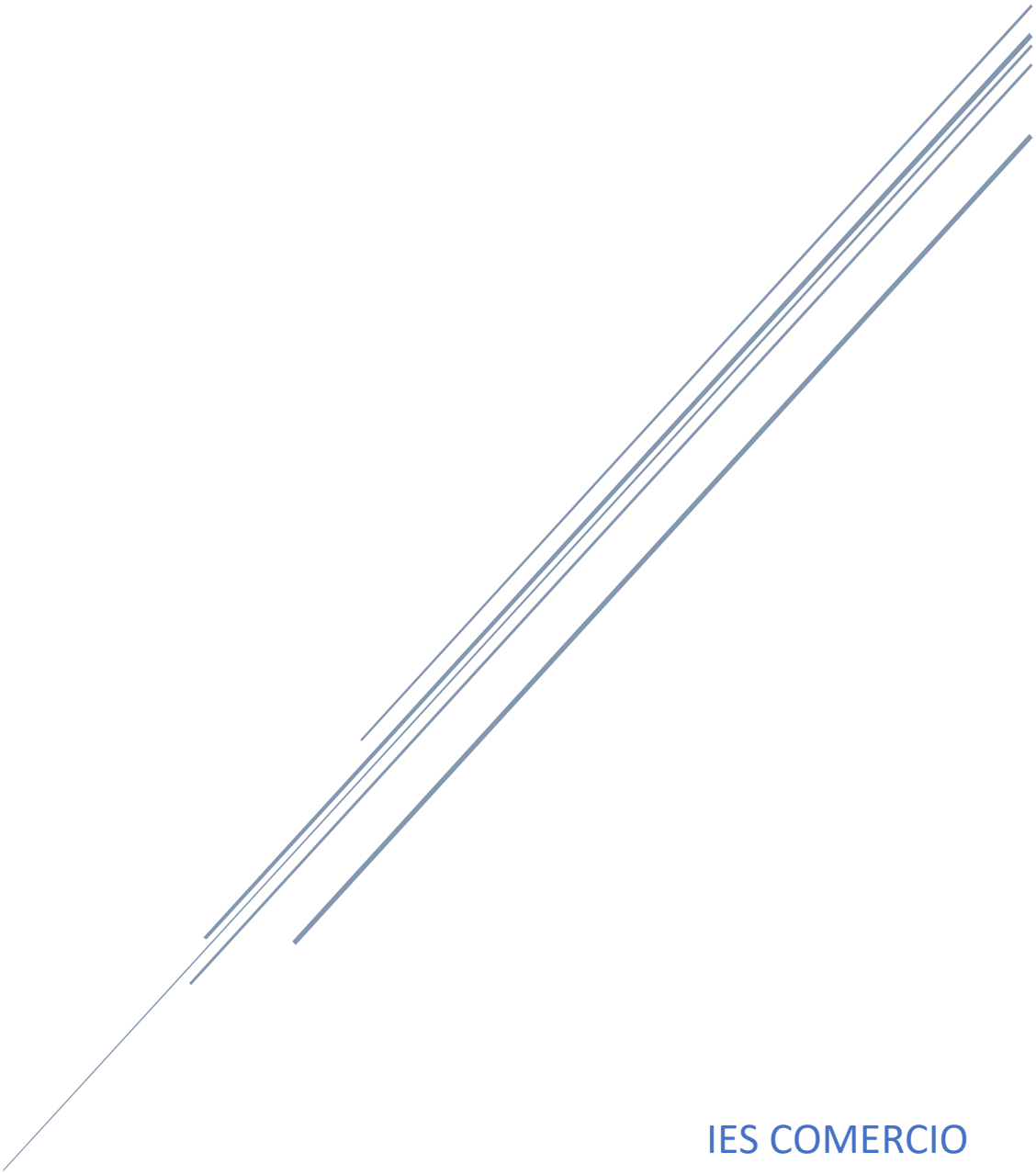
TRABAJO FIN DE GRADO

Circuit Management



IES COMERCIO
MIGUEL LUCENDO ESTEBAN

https://github.com/miguelLucendo/circuitmanagement.test

Índice

Resumen

Este Trabajo de Fin de Grado (TFG) tercer año del Grado Superior DUAL DAM-DAW (el correspondiente a un segundo año del Grado Superior DAW) consta de una aplicación web.

La aplicación trata de permitir a los empleados de un circuito poder gestionar los distintos procesos que se pueden llegar a realizar en el día a día de los mismos, como pueden ser ingresar los incidentes que ocurren en la pista, registrar las reservas que los distintos usuarios realicen, etc.

De forma resumida, estas son las distintas gestiones que se permiten realizar:

- Registrar coches
- Modificar coches
- Eliminar coches
- Ver coches

- Registrar usuarios
- Modificar usuarios
- Eliminar usuarios
- Ver usuarios

- Registrar reservas
- Modificar reservas
- Eliminar reservas
- Ver reservas

- Registrar incidentes
- Modificar incidentes
- Eliminar incidentes
- Ver incidentes

- Registrar tipos de incidentes
- Modificar tipos de incidentes
- Eliminar tipos de incidentes
- Ver tipos de incidentes

Palabras clave

Aplicación web

Symfony

PHP

Twig

TailwindCSS

Doctrine

MySQL

Circuito

Gestión

Introducción

En un mundo global y digitalizado como es el planeta y la sociedad en la que vivimos actualmente, todas o casi todas las empresas deben evolucionar, mirar hacia adelante y, un paso muy importante, si no el que más, pasa por digitalizarse, y, además, de una forma fácilmente accesible por todos, tanto sus clientes como sus trabajadores.

Es por esto que, he decidido realizar una aplicación web que nos ayude a gestionar de forma sencilla y sin la necesidad de ningún tipo de ordenador con ciertos requisitos de hardware ya que, para acceder a nuestra aplicación, solo necesitaremos dos cosas, acceso a internet y un dispositivo que nos permita conectarnos a la web, como pueden ser un ordenador o un dispositivo móvil.

Lo más probable es que existan muchos más softwares de gestión de circuitos y mejores que el mío que se empleen en los circuitos más conocidos en la actualidad, pero se trata de un proyecto escolar de 30 horas, en el que se mezclan los estudios con los hobbies y los gustos, que nos permiten mantener la motivación por continuar desarrollando nuestra aplicación gracias a que se trata de un tema que nos gusta.

Objetivos

El objetivo de este trabajo de fin de grado es llevar a cabo el desarrollo de una aplicación web especializada en la gestión de un circuito de coches que permita a los empleados del mismo gestionar las “entidades” necesarias, es decir, los clientes que participen, los coches que introduzcan al circuito, las reservas que realicen y los incidentes que ocurren en pista.

Habrá que tomar en consideración que no contamos con recursos ilimitados para realizar el desarrollo de esta aplicación, así que habrá que intentar maximizar el tiempo del que se dispone para realizar el desarrollo e intentar no meterse en desarrollo demasiado complicados que puedan requerir de más tiempo del que disponemos y que por culpa de esto, no se pueda terminar el desarrollo del núcleo de la aplicación.

También tendremos que tener en cuenta con qué tecnología vamos a desarrollarla para poder optimizar el tiempo y los recursos.

Estudio de la situación actual

Estado del arte

Symfony con Twig + TailwindCSS

Una de las tecnologías que me he planteado para utilizar en el desarrollo de esta aplicación es PHP, con el framework Symfony, uno de los frameworks más conocidos, si no el que más, de todos los especializados en desarrollo web.

Para renderizar las páginas en si utiliza plantillas de Twig, que está también basado en PHP y, para maquetarlo, se usaría otro framework, TailwindCSS, que nos permite darle clases directas en el HTML para cada aspecto del CSS que queremos tocar.

Son dos tecnologías que no he visto en clase y que habría que investigar para hacer un uso correcto de las mismas.

Laravel con Blade

En esta opción, igual que la anterior, se trata de un framework de PHP muy popular para el desarrollo de aplicaciones web, que nos facilitaría el desarrollo de la aplicación en sí gracias a su arquitectura MVC (Modelo Vista Controlador).

Además, al ser un framework bastante popular, cuenta con una comunidad de desarrolladores a los que seguro que también les han surgido las mismas dudas y bugs que nos podría surgir también a nosotros si finalmente nos decantamos por esta opción.

Python con Flask

Otra opción interesante para desarrollar la aplicación es Python junto al framework Flask.

Me he planteado esta opción porque Python es un lenguaje moderno, muy popular, que utilicé durante 5-6 meses y que, a pesar de su sintaxis peculiar, es muy potente y ligero.

Además, con Flask tendremos la posibilidad de añadir extensiones que nos permitan realizar distintas acciones de forma más fácil, como la autenticación, la base de datos, etc.

NodeJS con Express + React

La última opción que he estudiado a la hora de realizar la aplicación es programar la parte servidor de la aplicación con NodeJS y Express, que es un framework que nos permite utilizar un lenguaje orientado a cliente como es JavaScript, pero en el servidor y usar React para todo lo relacionado con el frontal de nuestra aplicación.

Aspectos teóricos

PHP

Ilustración 1 - Logotipo de PHP

¿Qué es PHP?

PHP es un lenguaje de programación interpretado del lado del servidor y de uso general que se adapta especialmente al desarrollo web. Fue creado inicialmente por el programador danés-canadiense Rasmus Lerdorf en 1994. En la actualidad, la implementación de referencia de PHP es producida por The PHP Group. PHP originalmente significaba Personal Home Page (Página personal).

Justificación de su uso

En este caso, me he decantado por utilizar PHP como lenguaje de programación del lado del servidor ya que es el lenguaje que más conozco en este aspecto debido a que lo hemos aprendido en uno de los módulos del curso en el instituto este año y, además, lo estoy utilizando en la empresa en la que estoy realizando las prácticas.

Symfony

Ilustración 2 - Logotipo de Symfony

¿Qué es Symfony?

Symfony es un framework diseñado para desarrollar aplicaciones web basado en el patrón Modelo Vista Controlador. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Justificación de su uso

Voy a usar este framework ya que es uno de los más populares y me apetece aprender sobre él porque me parece muy interesante, además, facilita el desarrollo de la aplicación de muchas formas, como, por ejemplo, en el acceso y el tratamiento de la base de datos con el ORM (Object-Relational Mapping) que trae de por sí.

También nos facilita mucho el enrutamiento debido a su sistema de anotaciones, que nos permite indicar en cada método del controlador sobre qué ruta se ejecutará.

TailwindCSS

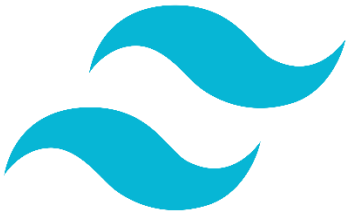


Ilustración 3 - Logotipo de TailwindCSS

¿Qué es TailwindCSS?

Es un framework CSS que nos permite agilizar nuestro desarrollo del frontend de una aplicación web. Nos permite aplicar estilos con unas clases ya predefinidas para cada propiedad de CSS que queramos modificar.

Esto nos permite no tener que crear distintas clases para los distintos estilos que necesitamos para maquetar la página (con lo que eso conlleva, como, por ejemplo, pensar nombres buenos, reutilizables, etc.) y, además, eso nos permite optimizar el rendimiento de nuestra aplicación porque agilizaremos el peso de la misma al no incluir los archivos CSS que necesitaríamos.

Justificación de su uso

Lo he utilizado porque es una tecnología que he aprendido estando en la empresa en la que estoy realizando las prácticas, y es un framework muy cómodo y ágil de utilizar y nos permite customizar de forma más rápida y sencilla cada componente de nuestra aplicación.

MariaDB

Ilustración 4 - Logotipo de MariaDB

¿Qué es MariaDB?

MariaDB es un sistema de gestión de bases de datos que está muy relacionado con MySQL, ya que fue desarrollado por uno de los desarrolladores, Michael "Monty" Widenius. El objetivo de su desarrollo fue el de mantener el software de gestión de base de datos en un modelo de software libre.

El sistema de gestión de bases de datos MariaDB incorpora las distintas funciones características de MySQL añadiendo algunas mejoras, como la posibilidad de ejecutar consultas complejas y almacenarlas directamente en caché, la nueva gestión de conexiones a BD, la posibilidad de acceder a clúster de datos (interesante para el trabajo en la nube) o el soportar la utilización de jerarquías de graphs y estructuras más complejas.

En cuanto a seguridad y rendimiento, MariaDB incorpora mejoras, estando siempre en constante evolución gracias a la aportación de una gran comunidad que se encuentra tras de ella.

Justificación de su uso

He utilizado MariaDB en el desarrollo de esta aplicación ya que es un SGBD (Sistema Gestor de Bases de Datos) muy popular, gratis y potente.

Además, es el SGBD con el que más he trabajado, tanto en los proyectos del curso como durante la estancia en la empresa.

Fases del proyecto

Análisis

Requisitos funcionales

Registro de usuario

El sistema permite a los usuarios registrarse en la aplicación.

Inicio de sesión

Se permite a los usuarios iniciar sesión con su email y contraseña.

Coches

Se permite registrar coches.

No se permite registrar coches con la misma matrícula.

Se permite modificar los campos de un coche.

Se permite eliminar un coche.

Tipos de incidente

Se permite registrar distintos tipos de incidente.

Se permite modificar el nombre del tipo de incidente.

Se permite eliminar el tipo de incidente.

Incidente

Se permite registrar incidentes indicando los coches que han participado en ellos.

Se permite modificar los campos de los incidentes.

Se permite eliminar los incidentes.

Usuario

Se permite añadir nuevos usuarios.

Se permite modificar los usuarios.

Se permite eliminar los usuarios.

Reserva

Se permite registrar reservas.

Se permite modificar los campos de la reserva.

Se permite eliminar las reservas.

Requisitos no funcionales

Necesitamos que la aplicación sea segura, es decir, que no tenga agujeros de seguridad que permita a los atacantes obtener datos confidenciales del sistema, sobre todo los datos relacionados con nuestros usuarios como pueden ser sus nombres y apellidos, pero, por encima de todo, las contraseñas, que, junto con su email, podrían incluso atacar sus cuentas en otros servicios.

También necesitamos que la aplicación se pueda utilizar de forma sencilla, que no cuente con complejos menús e interfaces muy enrevesadas que lo único que hacen es complicar la experiencia del usuario que usa nuestra aplicación.

Además, necesitaremos que sea rápida a la hora de responder a lo que el usuario necesita, es decir, que no necesite mucho tiempo para cargar la información que se solicite.

También necesitaremos que la aplicación sea escalable, es decir, que, por ejemplo, en el futuro necesitemos añadir más entidades a la aplicación y no por esto nuestra aplicación sea más pesada o nos cueste un mundo mantenerla sin romper el resto de la aplicación.

Además, en el mundo tecnológico actual en el que vivimos es totalmente necesario que la aplicación se adapta a distintos tamaños de pantalla, especialmente los más pequeños, porque, a día de hoy, todos contamos con un teléfono móvil en nuestras manos.

Diagrama de caso de uso

En esta aplicación tendremos dos actores, los administradores, o usuarios que manejarán la aplicación en el día a día y que se encargarán de realizar la mayoría de gestiones y los superadministradores, que es el administrador total de la aplicación que también podrá acceder a gestionar todo lo relacionado con el tema de usuarios.

Como podemos ver en el diagrama de abajo, un tipo de usuario puede realizar todas las acciones permitidas en la aplicación mientras que el otro tipo tiene denegado el permiso para gestionar los usuarios.

La palabra gestionar del siguiente diagrama hace referencia a lo siguiente:

- Crear
- Modificar
- Leer
- Eliminar

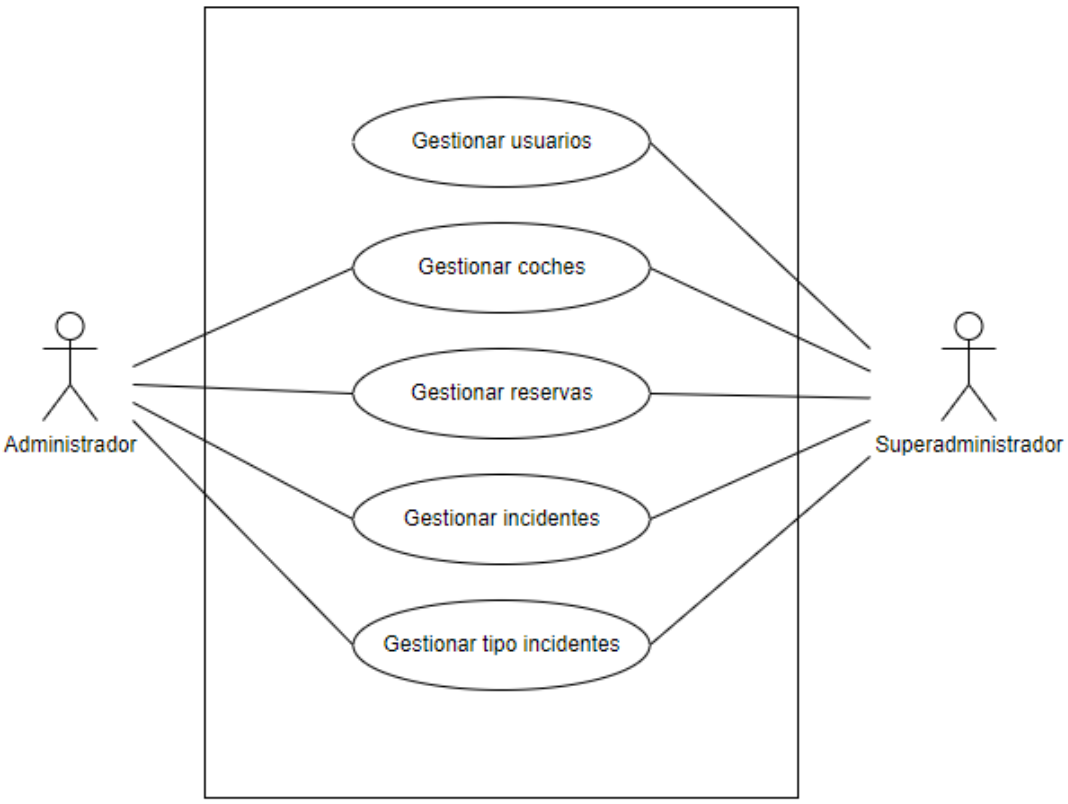


Ilustración 5 - Diagrama de casos de uso

Diagrama de secuencia

A continuación, voy a ilustrar dos posibles diagramas de secuencia que debe seguir nuestra aplicación en su uso normal.

Registro de coche con éxito

Como vemos, primero el usuario administrador intenta registrar un coche en la aplicación.

Esta petición pasa a la base de datos, que nos confirma que se ha registrado exitosamente el coche a nuestra aplicación y, es esta la que informa al usuario que su coche ha sido registrado de forma correcta.

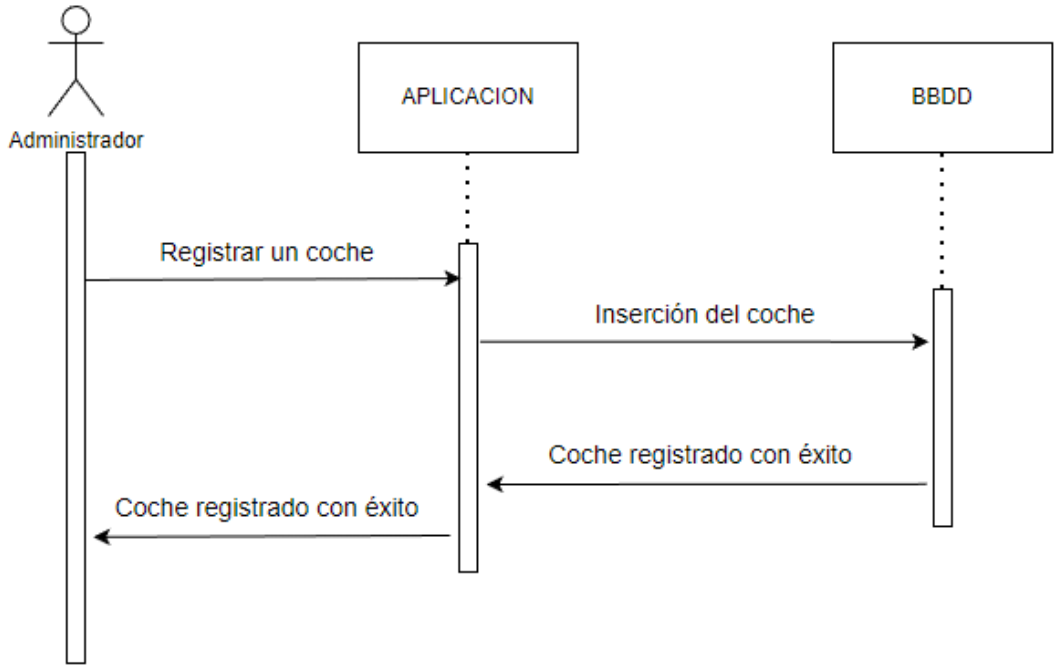


Ilustración 6 - Diagrama de secuencia exitoso

Registro de coche fallido

En este caso, vamos a representar el flujo que debería seguir nuestra aplicación si el registro de un coche es fallido por, por ejemplo, una matrícula repetida.

Como vemos, igual que en el caso exitoso, el usuario intenta registrar un coche en nuestra aplicación y esta es la que intenta insertarlo en la base de datos.

En este caso, la base de datos le devuelve a la aplicación el error en la inserción y es la aplicación la que notifica al usuario que algo ha salido mal.

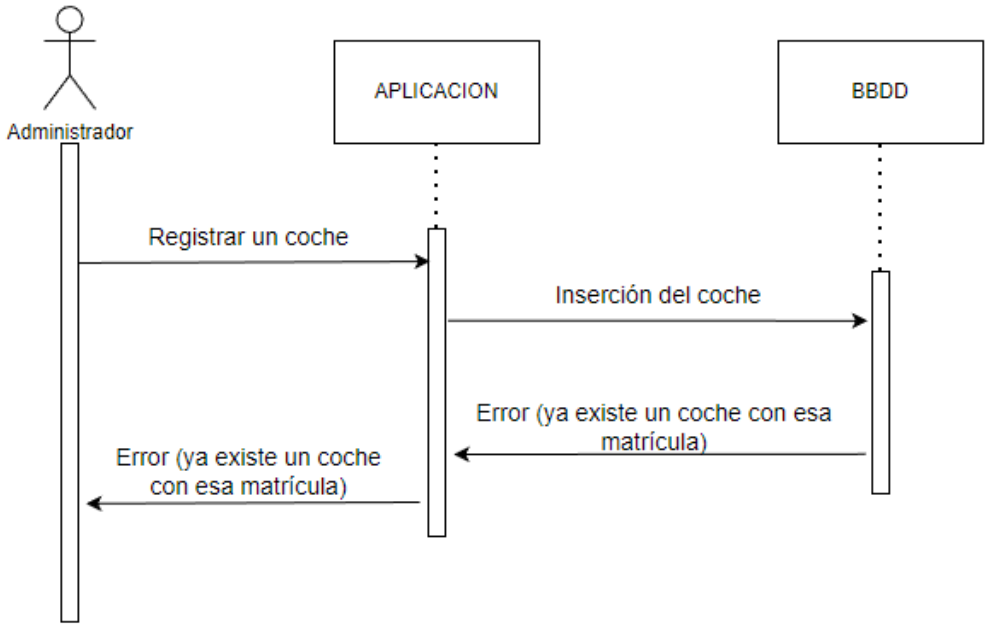


Ilustración 7 - Diagrama de secuencia fallido

Diagrama de navegación

La siguiente ilustración se trata de un diagrama de navegación, en el que podemos ver el flujo que un usuario sigue en el correcto funcionamiento de nuestra aplicación.

Como podemos ver, primero, antes de poder acceder a realizar ninguna acción, el usuario tiene que pasar por una autenticación. Una vez ahí, podrá realizar todas las gestiones necesarias, como, por ejemplo, para añadir un coche, tendrá que pasar por el "menú" de "Coches" y, una vez ahí, añadir el que él quiera añadir.

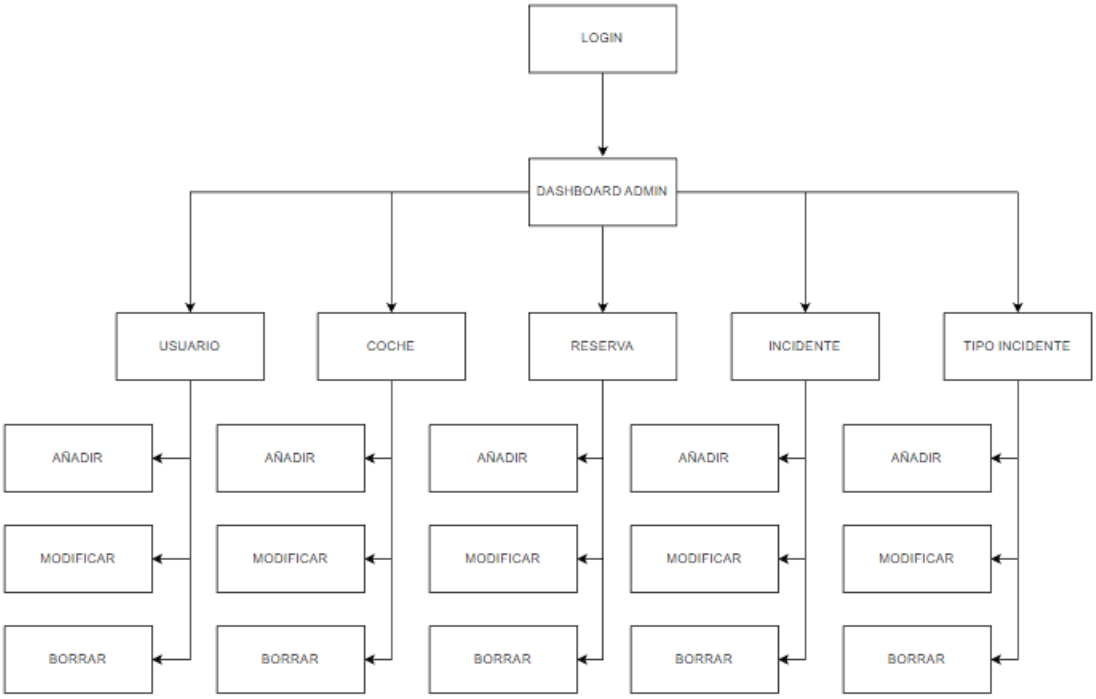


Ilustración 8 - Diagrama de navegación

Diseño

Diagrama Entidad / Relación

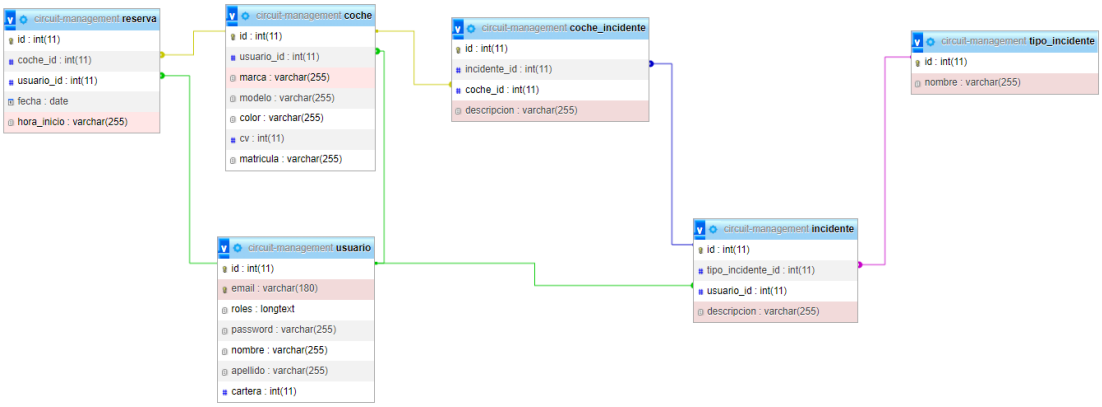


Ilustración 9 - Diagrama entidad - relación

Como podemos ver en este diagrama, nuestra base de datos cuenta con 6 tablas en las que almacenamos los datos, 5 principales y una auxiliar, que nos hace de conexión entre otras dos tablas.

Las rayas de colores indican las relaciones entre tablas y, además, de una forma sutil, también nos marca la cardinalidad de las mismas. Esto lo podemos saber si nos fijamos en la propia relación, la parte de esta que acabe con una forma semicircular mas gruesa que la contraria, es la tabla a la que se le pasa la clave foránea de la otra.

Modelo relacional

USUARIO (**Id**, email, roles, password, nombre, apellido, cartera)

COCHE (**Id**, matricula, marca, modelo, color, cv, matricula, *usuario_id*↑)

RESERVA (**Id**, fecha, hora_inicio, *coche_id*↑, *usuario_id*↑)

INCIDENTE (**Id**, descripción, *tipo_incidente_id*↑, *usuario_id*↑)

TIPO_INCIDENTE (**Id**, nombre)

COCHE_INCIDENTE (**Id**, descripción, *incidente_id*↑, *coche_id*↑)

Implementación

Lenguajes de programación

Como he comentado en puntos anteriores, el lenguaje de programación que se ha utilizado es PHP, ayudándonos del framework Symfony.

Se eligió este lenguaje ya que es un lenguaje que ya conozco debido a su uso durante el transcurso de la asignatura DWES (Desarrollo Web en Entorno Servidor).

Se sigue el patrón MVC con unas entidades que representan las tablas de la base de datos, unos controladores que indican a la aplicación que debe realizar y unas vistas que permiten al usuario realizar y ver las acciones que necesite.

Herramientas y programas usados en el desarrollo

Visual Studio Code

Ilustración 10 - Logotipo de VS Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft. Es software libre y multiplataforma, está disponible para Windows, GNU/Linux y macOS. VS Code tiene una buena integración con Git, cuenta con soporte para depuración de código, y dispone de un sinnúmero de extensiones, que básicamente te da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación.

Github

Ilustración 11 - Logotipo Github

Github es una página web dedicada al alojamiento de repositorios remotos con la tecnología Git, que utilizan la mayoría de desarrolladores de todo el mundo, tanto a nivel personal, como es mi caso actualmente, como a nivel profesional, con cuentas dedicadas a las empresas, las que pueden alojar todos los repositorios de sus proyectos.

Permite a los desarrolladores compartir código entre ellos de forma fácil y colaborar de forma sencilla.

XAMPP

Ilustración 12 - Logotipo de XAMPP

XAMPP es un conjunto de aplicaciones que, como su nombre indica, trae las aplicaciones necesarias para poder desarrollar una aplicación web de forma cómoda.

Estas aplicaciones son las siguientes:

- Apache: Servidor web.
- MySQL: Servidor de base de datos.
- PHP: Lenguaje de programación.
- Perl: Lenguaje de programación.

La 'X' del nombre significa que se puede utilizar en cualquier sistema operativo (al contrario de sus 'rivales', como LAMP o WAMP, que indican Linux y Windows respectivamente)

Ilustración 13 - Logotipo de PHPmyAdmin

PHPmyAdmin es una herramienta web que nos permite consultar los datos de nuestras bases de datos de forma visual, sin complicarnos ni tener que realizar a mano nuestras consultas SQL cuando queramos comprobar que nuestra aplicación está realizando de forma correcta las acciones sobre la base de datos.

Es una herramienta que podemos instalar al realizar la instalación de nuestro servidor XAMPP como un componente extra a parte de los componentes básicos.

Código del programa a comentar

Esta sección constará de las partes que considero más importantes o más 'curiosas' del código de la aplicación.

Código de una clase

Quiero mostrar cómo es el código PHP de una clase utilizando las herramientas que nos trae Symfony.

Como vemos, indicamos que es una Entidad y su repositorio (parte del código donde se realizan las acciones con la base de datos).

Además, también indico que el campo matrícula es única y un mensaje es caso de que se intente insertar una matrícula repetida.

Luego también vemos como indicamos que las propiedades de PHP son columnas de nuestra tabla de la base de datos.

También vemos que hay dos columnas distintas, que indican las relaciones con las otras entidades.

```php
#[ORM\Entity(repositoryClass: CocheRepository::class)]
#[UniqueEntity(fields: ['matricula'], message: 'Ya existe un coche con esta matrícula')]
class Coche
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(length: 255)]
    private ?string $matricula = null;

    #[ORM\Column(length: 255)]
    private ?string $marca = null;

    #[ORM\Column(length: 255)]
    private ?string $modelo = null;

    #[ORM\Column(length: 255)]
    private ?string $color = null;

    #[ORM\Column]
    private ?int $cv = null;

    #[ORM\OneToMany(mappedBy: 'coche', targetEntity: Reserva::class)]
    private Collection $reservas;

    #[ORM\ManyToOne(inversedBy: 'coches')]
    private ?Usuario $usuario = null;
```

Ilustración 14 - Código PHP Entidad

Plantilla admin

Otra parte importante a resaltar de la aplicación es su sistema de plantillas. He elegido la plantilla de administrador que es una plantilla que sigue la misma estructura que el resto de plantillas de nuestra aplicación, esto es, extender de otra plantilla que contiene el 'sidebar' y sobrescribir el bloque 'content', que es todo lo relacionado con el contenido de cada página.

Como vemos, todos los elementos del HTML están llenos de clases, esto es, como ya hemos visto con anterioridad, gracias a TailwindCSS, por ejemplo, si nos fijamos en el div principal de esta pantalla, cuenta con las siguientes clases:

- **w-full**: Indica que el ancho de este elemento será todo el disponible.
- **overflow-hidden**: Ocultará todos los elementos que puedan llegar a salirse de él.
- **flex**: Significa que la distribución del elemento será en formato flex.
- **flex-col**: Indica que la orientación será en formato columnas.
- **flex-wrap**: Indica que, si alguno elemento interior se sale de los límites, reajustará el resto.
- **px-4:** Le da un padding de 4 según su escala propia al eje X.
- **py-8**: Le da un padding de 8 según su escala propia al eje Y.
- **gap-12**: Le da una separación entre elementos de 8 según su escala.

```
{% extends 'admin/base.html.twig' %}

{% block content %}
    <div class="w-full overflow-hidden flex flex-col flex-wrap px-4 py-8 gap-12">
        <h1 class="text-3xl font-semibold dark:text-white">Dashboard</h1>
        <p class="text-lg dark:text-white">Bienvenido al panel de control. Aquí podrás administrar la aplicación.</p>

        <div class="grid grid-cols-3 gap-4">
            {% if 'ROLE_SUPER_ADMIN' in app.user.getRoles() %}
                <a href="{{ url('app_usuario_index') }}" class="col-span-3 md:col-span-1 flex overflow-hidden rounded-lg border dark:border-gray-200 shadow-lg p-4 dark:bg-white">
                    <i class="w-1/3 fa-solid fa-user mr-3 fa-2x"></i>
                    <div class="w-2/3 font-semibold">
                        Usuarios
                    </div>
                </a>
            {% endif %}
            <a href="{{ url('app_reserva_index') }}" class="col-span-3 md:col-span-1 flex overflow-hidden rounded-lg border dark:border-gray-200 shadow-lg p-4 dark:bg-white">
                <i class="w-1/3 fa-solid fa-ticket mr-3 fa-2x"></i>
                <div class="w-2/3 font-semibold">
                    Reservas
                </div>
            </a>
            <a href="{{ url('app_coche_index') }}" class="col-span-3 md:col-span-1 flex overflow-hidden rounded-lg border dark:border-gray-200 shadow-lg p-4 dark:bg-white">
                <i class="w-1/3 fa-solid fa-car mr-3 fa-2x"></i>
                <div class="w-2/3 font-semibold">
                    Coches
                </div>
            </a>
            <a href="{{ url('app_incidente_index') }}" class="col-span-3 md:col-span-1 flex overflow-hidden rounded-lg border dark:border-gray-200 shadow-lg p-4 dark:bg-white">
                <i class="w-1/3 fa-solid fa-car-burst mr-3 fa-2x"></i>
                <div class="w-2/3 font-semibold">
                    Incidente
                </div>
            </a>
            <a href="{{ url('app_tipo_incidente_index') }}" class="col-span-3 md:col-span-1 flex overflow-hidden rounded-lg border dark:border-gray-200 shadow-lg p-4 dark:bg-white">
                <i class="w-1/3 fa-solid fa-flag mr-3 fa-2x"></i>
                <div class="w-2/3 font-semibold">
                    Tipos de incidente
                </div>
            </a>
        </div>

    </div>
{% endblock %}
```

Ilustración 15 - Plantilla administrador

Método controlador

En este caso, voy a enseñar como es el método del controlador relacionado con la acción de crear un nuevo registro de una entidad, en este caso, un nuevo coche.

Como vemos, lo primero que tenemos que hacer es indicarle a nuestro programa que este método va a ser ejecutado en cierta ruta.

Si nos fijamos, al declarar la clase del controlador en su totalidad, le indicamos que va a tener una ruta especifica y, en el propio es donde indicamos la siguiente parte de la ruta, es decir, en este ejemplo, para que se ejecutara este método tendríamos que llamar a la ruta '/admin/coche/new'.

Pero si nos fijamos, vemos que la ruta no es lo único a lo que se hace referencia, también al nombre de esta ruta y a los métodos con los que se puede acceder.

El nombre sirve para poder referirnos luego a esta ruta de una forma dinámica para que, en el caso de que queramos cambiar la ruta, no tengamos que cambiarlo uno a uno en todos los demás sitios en los que se le hace referencia.

```php
#[Route('/admin/coche')]
class CocheController extends AbstractController
```

Ilustración 16 - Cabecera controlador

```php
#[Route('/new', name: 'app_coche_new', methods: ['GET', 'POST'])]
public function new(Request $request, CocheRepository $cocheRepository): Response
{
    $coche = new Coche();
    $form = $this->createForm(CocheType::class, $coche);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $cocheRepository->save($coche, true);

        return $this->redirectToRoute('app_coche_index', [], Response::HTTP_SEE_OTHER);
    }

    return $this->render('admin/coche/new.html.twig', [
        'coche' => $coche,
        'form' => $form,
    ]);
}
```

Ilustración 17 -Método new controlador

Pruebas

Durante el desarrollo de la aplicación se han ido probado todos o casi todos los casos posibles que se pueden dar durante el uso normal de la aplicación, pero no se han documentado absolutamente todos, a continuación, se muestran unas pocas pruebas de todas las que se han ido realizando.

Prueba realizada	*Resultado esperado*	*Resultado obtenido*
Inicio de sesión incorrecto	Mensaje avisando que no se ha podido iniciar sesión	Prueba correcta
Inicio de sesión correcto	Redirección a pantalla dashboard	Prueba correcta
Registro coche	El coche aparece en la tabla de coches disponible	Prueba correcta
Eliminación coche	El coche desaparece de la tabla de coches disponibles	Prueba correcta
Modificación campos coche	Los campos actualizados se ven reflejados en la tabla de coches disponibles y en la ficha del propio coche	Prueba correcta
…	…	…

Además, también se adjuntan algunas capturas de pruebas realizadas.

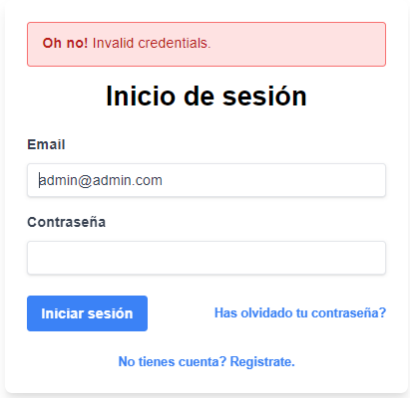


Ilustración 18 - Inicio de sesión inválido

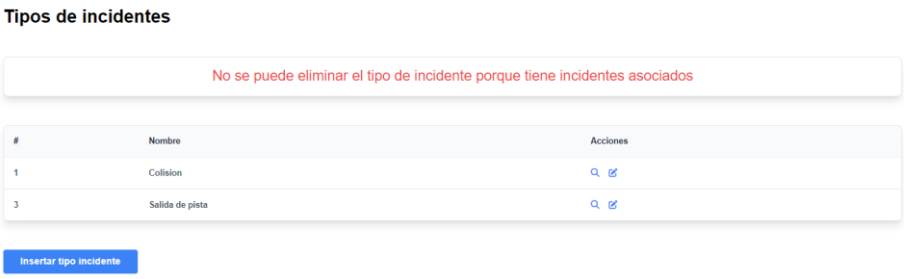


Ilustración 19 - Intento de eliminación registro inválido

Ampliación y posibles mejoras

En esta sección se tendrán en cuenta las posibles ampliaciones a realizar en este proyecto si se diera el caso de contar con más tiempo disponible para desarrollar de forma más exhaustiva la aplicación:

- **Permitir a los usuarios finales interactuar:** Actualmente, esta aplicación solo la pueden utilizar los trabajadores del circuito. Estaría bien que los propios clientes finales de nuestro circuito puedan realizar ellos mismos las reservas, registrarse en la aplicación, registrar los coches con los que van a introducirse al circuito, etc.

- **Permitir seguimiento:** Se podría integrar un dispositivo en los coches que se introducen a rodar al circuito para poder realizar un seguimiento de los mismos a través de algún tipo de mapa integrado en la página web. Quizás incluso permitir el acceso a este mapa a los usuarios comunes para que puedan saber en cualquier momento la "acción" que se está llevando a cabo sobre el asfalto del circuito.

- **Aplicación móvil:** También se podría desarrollar esta aplicación como una aplicación nativa de móvil disponible en App Store y Play Store, para permitir que los usuarios puedan realizar sus reservas desde el propio teléfono móvil (ya pueden gracias al diseño responsive, pero sería de forma más nativa) o los trabajadores del circuito pudieran gestionar los incidentes desde el propio lugar del incidente.

- **Externalizar base de datos:** Otra de las posibilidades que se plantean a la hora de ampliar la aplicación sería externalizar la base de datos, es decir, que no fuera esta aplicación la aplicación principal que la gestione, sino que existiera un servicio de API con unos endpoints estandarizados para que el resto de aplicaciones (app móvil, por ejemplo) pudieran comunicarse con esta sin tener en cuenta nada más que la dirección a la que realizar la petición.

- **Mejorar aplicación actual:** Como última posible mejora que se podría realizar, es mejorar la aplicación ya existente en sí, es decir, continuar puliendo la funcionalidad ya existente en la aplicación, corrigiendo distintos bugs que puedan ir siendo descubiertos debido a su uso.

Conclusión

Ya terminado el desarrollo de nuestra aplicación, tenemos una aplicación web que nos permite realizar las operaciones más básicas en un circuito de coches, como pueden ser el control de los usuarios, los coches que estos tienen, las reservas que realicen, los incidentes en pista que se puedan dar o los tipos de incidentes que pueden llegar a ocurrir.

Al realizar esta aplicación he aprendido de cero y he mejorado con los siguientes lenguajes/frameworks/herramientas:

- Git
- Github
- PHP
- Symfony
- TailwindCSS

Bibliografía

Wikipedia. (2016, 1 de mayo). *PHP. En Wikipedia, la enciclopedia libre.*

https://es.wikipedia.org/wiki/PHP

Wikipedia. (s.f.) *Symfony. En Wikipedia, la enciclopedia libre*

https://es.wikipedia.org/wiki/Symfony

Hostingplus. (2020, 14 de diciembre). *Qué es MariaDB y cuáles son sus características*

https://www.hostingplus.com.es/blog/que-es-mariadb-y-cuales-son-sus-caracteristicas/

OpenWebinars.net (2022, 22 de Julio). Qué es Visual Studio Code y qué ventajas ofrece.

https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/