Portada

Índice

[1. Resumen 1](#_Toc102992369)

[2. Introducción. 1](#_Toc102992370)

[3. Objetivos y características del proyecto. 3](#_Toc102992371)

[3.1. FrontEnd. 3](#_Toc102992372)

[3.1.1. Inicio de Sesión. 3](#_Toc102992373)

[3.1.2 Administración de vehículos. 3](#_Toc102992374)

[3.2 Backend. 3](#_Toc102992375)

[3.2.1. Base de datos. 3](#_Toc102992376)

[3.2.2 ApiRest. 4](#_Toc102992377)

[3.2.3. Llamadas a la Api. 5](#_Toc102992378)

[4. Finalidad: Qué queremos conseguir con su implementación. 5](#_Toc102992379)

[5. Medios materiales usados: humanos, hardware, software 5](#_Toc102992380)

[5.1 Humanos. 5](#_Toc102992381)

[5.2 Hardware. 5](#_Toc102992382)

[6. Planificación del proyecto 6](#_Toc102992383)

[7. Fase de pruebas 6](#_Toc102992384)

[8. Conclusiones y trabajos futuros o posibles mejoras 6](#_Toc102992385)

[9. Apéndices. 6](#_Toc102992386)

[10. Referencias bibliográficas. 6](#_Toc102992387)

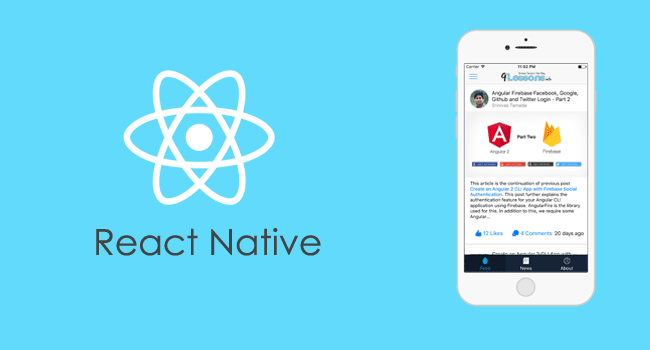
# Resumen

El problema que se ha tratado es la inexistencia de una red social para amantes del motor. Para ello, se ha creado una aplicación móvil, que es tanto para Android como para IOS en la cual se podrá mediante usuarios añadir vehículos y ver los de los demás.

Esto hace que la comunidad del motor esté más unida y se fomente la colección, el mantenimiento, la modificación, y el intercambio de vehículos a motor.

En la aplicación se creará una cuenta para poder acceder a ella. Una vez dentro para poder añadir un vehículo (que podrá ser coche o moto) deberá completar los datos necesarios en el formulario y subir una imagen. Una vez terminado ese proceso se podrá ver el vehículo añadido en la aplicación en la pantalla de visualización.

# Introducción.

A la hora de crear una red social para la comunidad del motor lo quisimos hacer accesible y de fácil uso. Para ello decidimos crear una aplicación para los dos sistemas operativos del mercado, Android e IOS. Es decir, el código de la aplicación es hibrido y se puede desplegar en las dos plataformas de aplicaciones de cada sistema.

La idea de la aplicación es tener una interfaz sencilla, que todo el mundo pueda usar con facilidad, pero que, a la vez, sea útil y completa.

Quisimos abarcar solamente los sectores de las motos y los coches para poder llegar a un público más específico y no intentar llegar a todo el público ya que pensamos que llenar la red social de sectores que estuvieran vacíos o casi vacíos debido a la poca afluencia de usuarios de esas características, daría una imagen vacía o incompleta de nuestra idea original.



Todos los vehículos que se añaden son subidos a una base de datos global, por lo que cada vez que se entra en la aplicación con el usuario se podrá ver todos los vehículos hasta la fecha, ya sean nuevos creados a partir del inicio de sesión, o ya creados anteriormente. Esto hace que se convierta en una galería atemporal de vehículos de interés.



# Objetivos y características del proyecto.

Los objetivos del proyecto de pueden categorizar en dos grandes secciones:

## 3.1. FrontEnd.

En este apartado vamos a ver las partes en las que se compone la sección de fontend, que consiste en la parte visible que tiene el usuario una vez que accede a la aplicación, la interfaz para ser más concretos.

A la hora de diferenciar las partes de este podemos encontrar el inicio de sesión y la administración de vehículos una vez pasado esta primera parte.

### 3.1.1. Inicio de Sesión.

Aquí podemos encontrar un conjunto de pantallas en las que principalmente podemos entrar con una cuenta ya creada que este en la base de datos o crear una nueva y registrarla.

A la hora de iniciar sesión se nos dará la opción de recordar la contraseña y de registrar una nueva cuenta dado que cabe la posibilidad que sea la primera vez que se entra en la aplicación o de que no tengamos ninguna cuenta registrada.

A la hora de recordar la contraseña tendremos que introducir nuestro correo electrónico de la cuenta en cuestión y se enviará un email con la contraseña de nuestra cuenta.

Si optamos por la opción de crear una nueva cuenta, se nos enviará a la pantalla para realizarlo.

### 3.1.2 Administración de vehículos.

Aquí podemos encontrar principalmente dos pantallas.

En la primera pantalla podemos ver como todos los vehículos existentes en la base de datos serán visualizados de forma descendente mostrando todos los datos de interés.

En la segunda podremos observar una especie de formulario en el que tendremos que aportar los datos necesarios, así como una imagen para poder subir nuestro vehículo.

## 3.2 Backend.

En este apartado veremos las partes en las que se dividen este proyecto en cuanto a Backend, que consiste en la parte de código y de funcionalidades que el usuario no ve desde la interfaz, es decir el usuario solo ve los elementos interactivos, pero no lo que estos hacen en segundo plano

### 3.2.1. Base de datos.

La base de datos que vamos a necesitar es una base de datos relacional para guardar los datos en tablas. Estos se harán desde phpMyAdmin.

En primer lugar, vamos a necesitar una tabla para poder administrar los usuarios. Esta será su estructura:

Tabla Usuarios:

-Nombre.

-Apellidos.

-Correo Electrónico.

-Nick.

-Contraseña.

Luego para poder administrar los vehículos que se suban desde la aplicación crearemos una tabla.

Tabla Vehículos:

-Marca.

-Modelo.

-Apodo.

-Matricula.

-Descripción.

-Fecha.

-Imagen.

-Tipo.

### 3.2.2 ApiRest.

Una ApiRest es la parte que comunica la aplicación con la base de datos, de forma que, el usuario simplemente tiene que rellenar formularios y clicar en elementos interactivos y el programador solamente tiene que escribir una sola vez la parte del código que se encarga de trabajar con la base de datos.

Todo esto funciona mediante *llamadas* al api, la cuales se envían con un cuerpo o *body* donde se encuentran diferentes parámetros necesarios, y se envían a una dirección con su ruta y parámetros correspondientes, que hace que, el api lo reciba y procese la información.

La Api necesita un mapeo de todas las entidades que tenga la base de datos para así poder trabajar con esta. Además de esto necesita unos controladores que mediante unas rutas se pueda decidir con cual entidad interactuar, así como, métodos para poder diferenciar el tipo de llamada que se realiza.

La Api para poder funcionar correctamente en el cuerpo se le tiene que entregar los datos en forma de Json, y a su vez esta te devolverá un cuerpo con el mismo formato o un cuerpo con un parámetro booleano.

### 3.2.3. Llamadas a la Api.

Esta parte es concreta, se trata de crear métodos que son llamados por los elementos colocados en la parte de FrontEnd que se encarguen que llamar a la Api y enviarles los datos necesarios en el formato necesario.

Así como llamarla para obtener los vehículos existentes y poder hacer que se muestren en la pantalla de visualización.

# Finalidad: Qué queremos conseguir con su implementación.

A la hora de implementar nuestra aplicación nuestra intención es conseguir una visión del mundo del motor mas amigable y cercana. Desde siempre ha sido un hobby un poco marginado en ese sentido y la comunidad ha tenido que resguardarse en foros cutres de internet que tenían poca o nula visibilidad.

Mediante nuestra app, uno mismo desde el store de su dispositivo móvil puede descargarla y usarla. Es un sistema bastante sencillo pero muy vistoso, simplemente con un sistema de cuentas de usuario convencional y un par de pantallas fáciles de usar conseguimos que todo esté más unido.

La aplicación tendrá unas funcionalidades de lanzamiento que serán ampliadas a medida que los usuarios nos entreguen un feedback y unas necesidades adicionales que no estén implementadas, haciendo que se convierta en una aplicación variable a gusto de los consumidores.

# Medios materiales usados: humanos, hardware, software

## 5.1 Humanos.

Los medios humanos utilizados en este proyecto ha sido una única persona, el autor de este documento:

Raúl Gutiérrez España, estudiante de Cesur en el Grado Superior de Desarrollo de Aplicaciones Multiplataforma.

## 5.2 Hardware.

Para realizar la aplicación se ha necesitado únicamente mi ordenador personal, que está pensado para gaming por lo que no hemos tenido problemas de recursos a la hora de utilizar emuladores y multitud de programas en multitarea.

-Procesador: Intel i7 9700f 3GHZ

-Memoria RAM: 16GB

-Tarjeta Gráfica: NVIDIA RTX 3070 8GB

-Disco Duro 1: 500GB SSD

-Disco Duro 2: 1TB HDD

-Sistema Operativo: Windows 10 pro

## 5.2 Software

Para poder desarrollar correctamente hemos necesitado una serie de programas:

### 5.2.1 Visual Studio Code.

Visual Studio Code es un software de edición de texto con consola de comandos implementados. Mediante un solo programa es posible programar múltiples lenguajes debido a su gran capacidad de albergar modificaciones y plug ins que permiten desde cambiar la apariencia hasta crear cualquier tipo de proyecto con cualquier tipo de consola de comandos.

Dentro de este hemos instalado los plug ins básicos para poder editar código de javascript que es el código base del framework que hemos utilizado en la aplicación.

También hemos instalado los plug ins necesarios para poder editar código php ya que, la Api esta creada mediante symfony que es un framework de php, así como la base de datos que esta administrada mediante phpMyAdmin.

### 5.2.2 NODE JS.

Node Js es una plataforma en la que podemos tener alojado nuestro código basado en javascript, lo que nos ahorrará mucho tiempo y muchos errores, de forma que no simplificará mucho la creación del proyecto que necesitamos para desarrollar la aplicación.

### 5.2.3 React Native.

React Native es una variante de React, se trata de un framework de JavasCript que nos permitirá crear aplicaciones móviles tanto para Android como para ios.

Utilizar elementos que pueden ser compilado en los dos sistemas operativos sin obtener resultados disonantes entre ellos, es decir, para cada elemento, existe un homólogo en cada sistema.

### 5.2.4 Expo CLI.

Para poder desarrollar en local y poder probar los avances que hagamos en la aplicación en React Native se ha decidido utilizar Expo CLI en local, debido a que React Native CLI debido a su sencillez de instalación y uso, ya que este último suele arrojar muchos errores en su mayoría laberínticos y con soluciones poco intuitivas y a priori no tiene ninguna desventaja en cuanto a nuestros objetivos en el desarrollo de la App.

Cabe destacar que se ha realizado en local para poder hace pruebas con la Api que hemos creado sin tener que desplegarla en un servidor en la nube.

### 5.2.5 Symfony CLI.

Symfony Cli es una consola de comandos que nos permite generar proyectos en php que nos servirán para crear Apis personalizadas.

Nos permite crear un servidor en local donde alojar la Api para poder hacerle llamadas y probar que funciona.

### 5.2.6 PhpMyAdmin.

Es un gestor de bases de datos que nos permite crear bases de datos y administrar su contenido. Es algo básico pero muy útil. De esta forma aprovechamos que usamos todos los servidores de prueba en php y no hay que instalar muchas mas aplicaciones.

### 5.2.7 XAMPP.

Es un administrador de servicios php el cual nos será de ayuda para poder utilizar servidores en local basados en php y poder probar las funcionalidades de nuestra aplicación como si estuviera desplegada en servidores online.

# Planificación del proyecto

# Fase de pruebas

# Conclusiones y trabajos futuros o posibles mejoras

# Apéndices.

# Referencias bibliográficas.