Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"

Proyecto de desarrollo de aplicaciones multiplataforma
Aplicación Móvil Formula 1

CICLO FORMATIVO DE GRADO SUPERIOR
Desarrollo de Aplicaciones Multiplataforma (IFCS02)

Curso 2021-22

Autor/a/es:
Saúl Mellado Herrera

Tutor/a:
José Luis González Sánchez

Departamento de Informática y Comunicaciones
I.E.S. Luis Vives

Contenido

1 INTRODUCCIÓN

El proyecto se basa en la creación de una aplicación móvil relacionada con uno de mis hobbies que es la Formula 1. Esta aplicación contara con información acerca de las carreras que se van realizando a lo largo del año. Además contará con información sobre los pilotos y los circuitos en los que se corre durante la temporada. También contara con un apartado de noticias que se irán publicando durante el transcurso dela temporada de tal manera que el usuario siempre este informado sobre lo que pasa en el mundo de la Formula 1 y tenga todo al alcance de su mano. Sin necesidad de consultar varias fuentes para enterarse de la actualidad.

1.1 OBJETIVO

Los objetivos que pretendo conseguir con el proyecto son fortalecer conocimientos ya adquiridos durantee el curso, además de obtener nuevos conocimientos mediante el uso de diferentes tecnologías y lenguajesde programación no vistos durante el curso. Además aprender a gestionar el tiempo estableciendo deadlines para la entrega de dichas partes del proyecto.

1.2 ALCANCE

La aplicación va a contar con información de circuitos, pilotos y tabla de puntuación tanto del mundial depilotos como el mundial de constructores (equipos/escuderías). También contara con un apartado de noticias sobre la Formula 1. Tendrá un Login para que puedas iniciar sesión con diferentes cuentas. Opcionalmente me gustaría realizar un sistema de notificaciones para cuando se publique una nueva noticia y si es posible poder filtrar esas notificaciones por los usuarios, es decir, que los usuarios puedan elegir qué tipo de notificaciones desean recibir (cuando empieza una carrera, noticias sobre un cierto piloto / escudería etc). Además se podría realizar una aplicación de escritorio orientado a administradorespara poder gestionar los usuarios.
Opcionalmente también me gustaría que los usuarios pudieran iniciar sesión con sus propias cuentas deGoogle sin necesidad de tener que registrarse mediante un correo y contraseña especifico.
Las noticias serán obtenidas mediante un scrapper que detecte actualizaciones en páginas web oficiales.

1.3 JUSTIFICACIÓN

Este proyecto será realizado para seguir aprendiendo y fortaleciendo tanto conocimientos que ya hemosvisto durante el curso como conocimientos nuevos. Creo que es una aplicación necesaria ya que para la gente que le interesa el mundo de la Formula 1 es muy cómodo tener todo tipo de información que deseas en una aplicación móvil sin tener la necesidad de buscar información en Google. Creo que es mucho mejor tener todo lo necesario en una aplicación móvil en la que puedas encontrar lo que quieras de una forma sencilla y sobre todo amigable.

2 IMPLEMENTACIÓN

2.0.1ANÁLISIS DE LA APLICACIÓN

En el desarrollo de esta aplicación utilizaremos diferentes tipos de tecnologías. Comenzando por elbackend, que estará hecho en Spring Boot.

Spring Boot es una tecnología que nos permite crear aplicaciones autocontenidas, con esto nos podemosolvidar de la arquitectura y enfocarnos únicamente en desarrollo, delegando a Spring Boot labores comoconfiguración de dependencias, desplegar nuestro servicio o aplicación a un servidor de aplicaciones y enfocarnos únicamente en crear nuestro código.
Mediante Spring Boot crearemos nuestra api rest con el contenido necesario para obtener mediante peticiones desde nuestro Cliente Movil. Se almacenará en una base de datos, en este caso la elegida seráMongoDB.

MongoDB es un sistema de base de datos NoSQL orientado a documentos de código abierto y escrito enC++, que en lugar de guardar los datos en tablas lo hace en estructuras de datos BSON (similar a JSON) con un esquema dinámico.
Para obtener las noticias sobre la Formula 1 realizaremos scrapping web mediante la librería de javaJSoup.

El Web Scraping (o Scraping) son un conjunto de técnicas que se utilizan para obtener de forma automática el contenido que hay en páginas web a través de su código HTML. El uso de estas técnicastiene como finalidad recopilar grandes cantidades de datos de diferentes páginas web.

Nuestro FrontEnd será realizado en java mediante el uso de Android Studio.

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Androidy está basado en IntelliJ IDEA.

En nuestro FrontEnd desarrollaremos la interfaz de la aplicación, además obtendrá la información necesaria de nuestro BackEnd y se la mostrará al usuario de una forma bonita y sencilla. Además, será fácil navegar por la aplicación distribuyendo los diferentes apartados de la interfaz de una forma intuitivay cómoda para el usuario.

Para comprobar las consultas a la Api Rest utilizare Postman.

Postman es una aplicación que nos permite realizar pruebas API. Es un cliente HTTP que nos da la posibilidad de testear 'HTTP requests' a través de una interfaz gráfica de usuario, por medio de la cualobtendremos diferentes tipos de respuesta que posteriormente deberán ser validados.

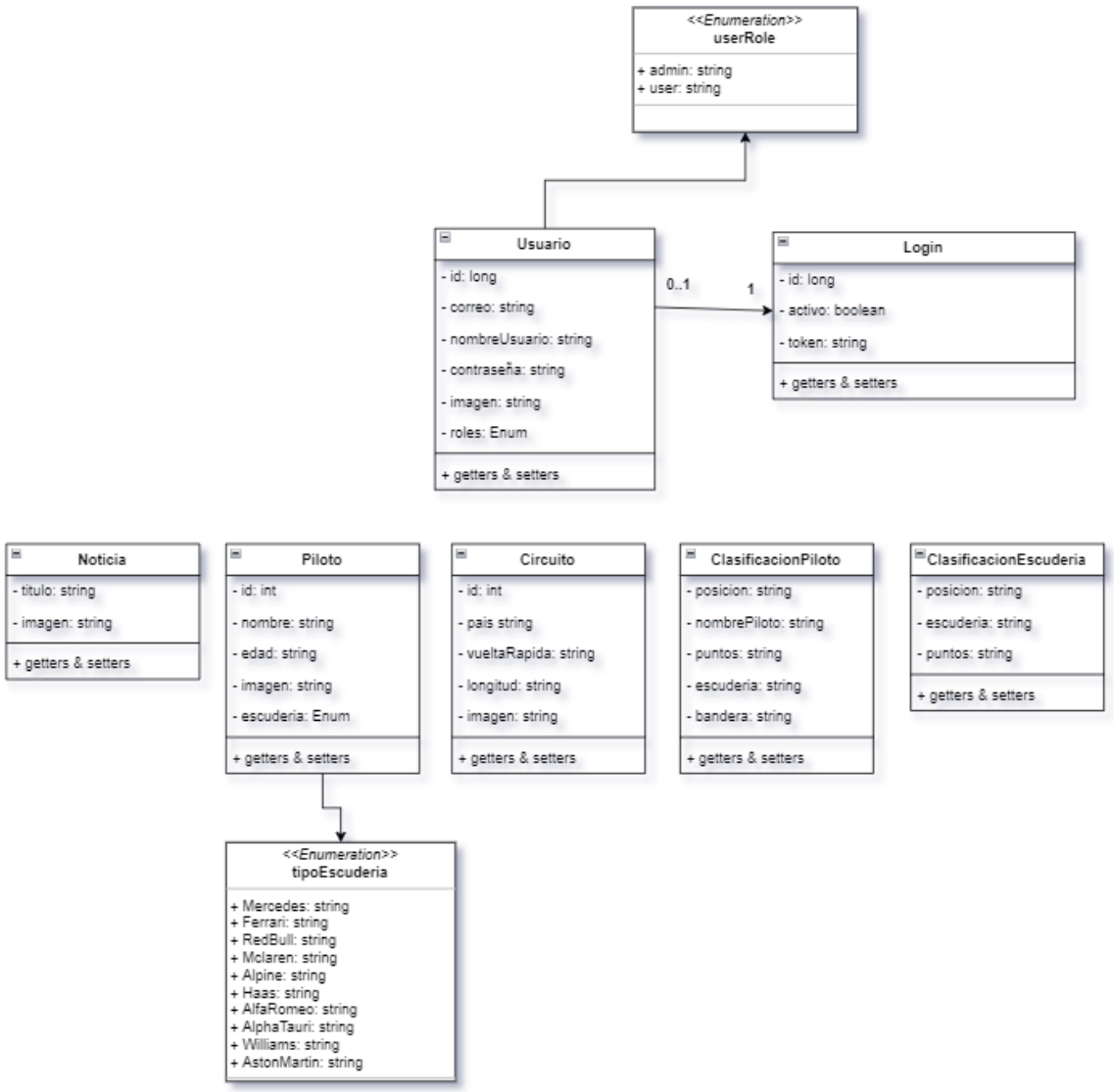
El servidor de la base de datos será lanzado en Docker.

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

Nuestro desarrollo de la api será realizado en el IDE IntelliJ.

IntelliJ IDEA es un entorno de desarrollo integrado para el desarrollo de programas informáticos , inteligente y sensible al contexto para trabajar con Java y otros lenguajes JVM como Kotlin, Scala y Groovyen todo tipo de aplicaciones. Es desarrollado por JetBrains.

2.0.2 DIAGRAMA MODELO DE DATOS



Este sería el diagrama de modelo de datos del backend. Aquí se muestra todo lo necesario para el correcto funcionamiento de las funcionalidades que nos ofrece nuestra aplicación.

Nuestra base de datos contara con las siguientes tablas: Usuario, Login, Noticias, Circuito, Clasificación Piloto, Clasificación Escuderia, Piloto.

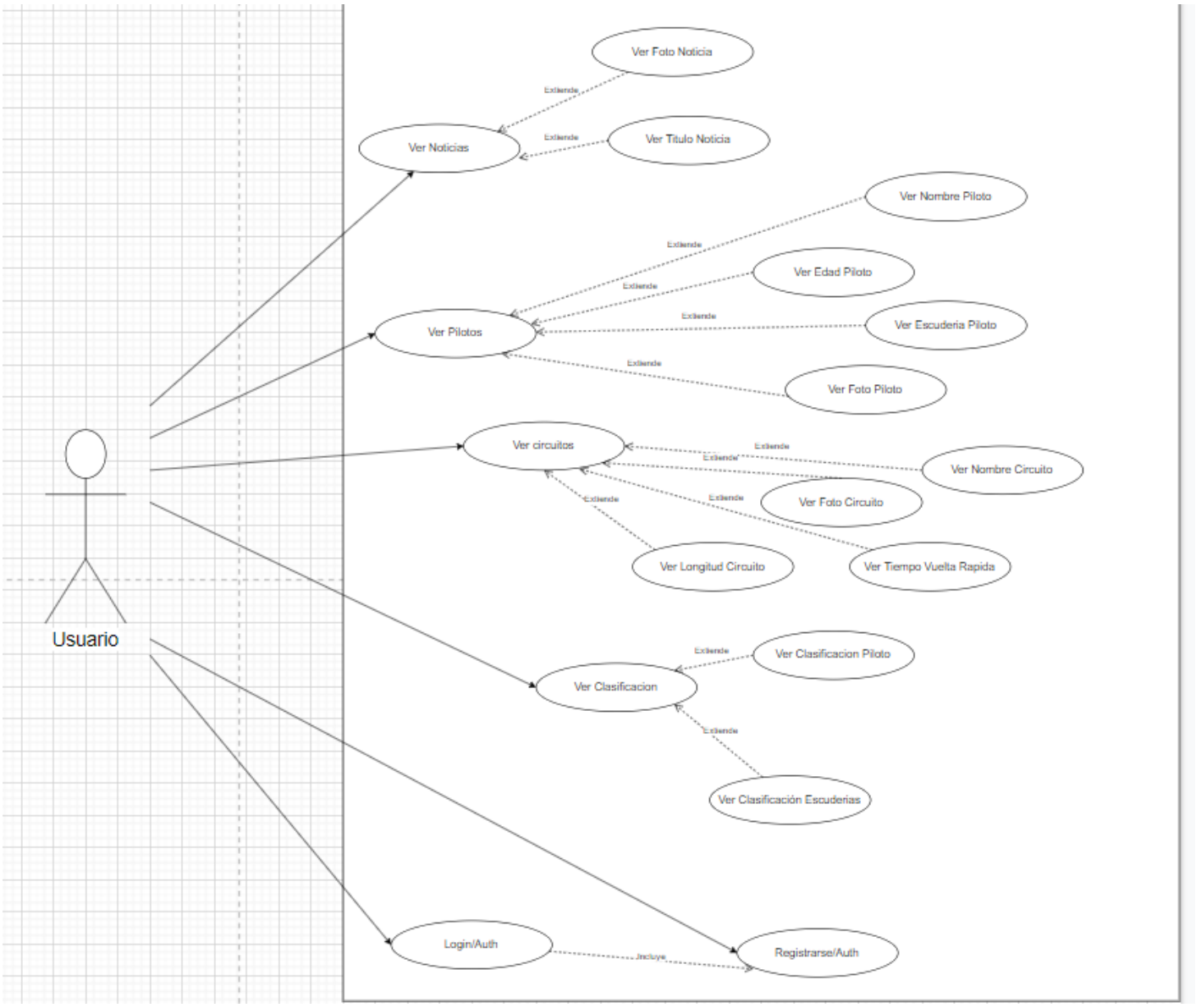
Ademas existen dos tipos Enum para las escuderias a las que pertenecen los pilotos y un UserRole que asigna el rol que tendrá cada usuario dentro de la aplicación.

Dependiendo del tipo de rol que tenga el usuario podrá hacer unas peticiones u otras a la api. En este caso si el usuario no es admin solo podrá realizar peticiones GET a la api. Es decir, simplemente podrá visualizar la información de los pilotos, circuitos, clasificación y noticias.

En el caso de ser Admin podría insertar nuevos elementos a la api, actualizarlos y eliminarlos en caso de que quiera hacerlo. También podrá visualizar esta información. Todo esto lo realizará mediante peticiones GET, POST, PUT, DELETE.

2.0.3 DIAGRAMA CASOS DE USO

Un caso de uso representa una unidad funcional coherente de un sistema, subsistema o clase. En un caso de uso uno o más actores interaccionan con el sistema que realiza algunas acciones.



En este caso solo tenemos un actor que es nuestro usuario. Cada funcionalidad/ interacción que puede realizar dentro de la aplicación es un caso de uso. El objetivo de este diagrama es representar la comunicación y el comportamiento del sistema mediante la interacción del usuario con las funcionalidades de la aplicación.

2.0.4 REQUISITOS FUNCIONALES

Los requerimientos funcionales de un sistema son aquellos que describen cualquier actividad que este deba realizar, en otras palabras, el comportamiento o función particular de un sistema o software cuando se cumplen ciertas condiciones.

En este caso nuestro usuario tiene la posibilidad de:

1. Logarse en el sistema.
2. Registrarse en el sistema.
3. Consultar clasificación pilotos.
4. Consultar clasificación escuderías.
5. Consultar Información sobre los pilotos.
6. Consultar los circuitos de la temporada.
7. Consultar las noticias más recientes del mundo de la Formula 1.
8. El sistema debe controlar que todas las peticiones son seguras.
9. La aplicación debe mantener unos cánones de diseño similares en cada parte de la aplicación
10. La aplicación debe funcionar en todos los dispositivos móviles Android, independientemente de su tamaño
11. La experiencia del usuario debe ser positiva
12. La interfaz debe ser amigable y sencilla para el usuario.
13. Las peticiones no deben tardar demasiado en realizarse para una experiencia satisfactoria.
14. La aplicación debe de ser escalable para posibles actualizaciones y mejoras.

2.0.5 REQUISITOS NO FUNCIONALES

Los requerimientos no funcionales representan características generales y restricciones de la aplicación o sistema que se esté desarrollando.
Son cualidades las cuales nuestro producto debe tener.

En este caso los requisitos no funcionales son los siguientes:

1. Todos los datos de la aplicación deben de estar almacenados remotamente en un servidor.
2. El cliente móvil esta desarrollado en lenguaje JAVA.
3. Las peticiones a la API REST serán realizadas mediante Retrofit 2.
4. La API REST será realizada en Spring Boot.
5. El almacenamiento de datos será realizado en MongoDB.
6. La API REST debe estar testada mediante JUnit 5 o Mockito.
7. El despliegue de la base de datos será realizado en Docker.
8. La gama de colores de la aplicación deberá contar mínimo con negro y rojo.

2.0.6 REQUISITOS DE INFORMACIÓN

Un requisito de información es una capacidad que el usuario necesita para resolver un problema o conseguir un objetivo específico.
Son las condiciones que se deben cumplir o debe poseer un sistema para satisfacer una norma o una especificación concreta.

En este caso los requisitos de información son los siguientes:

- *Usuario:* Son aquellas personas que utilizan la aplicación. Deben disponer de la siguiente información.
  - Id
  - Correo
  - Nombre Usuario
  - Contraseña
  - Imagen
  - Rol
- *Noticia:* Representa las novedades que suceden en el mundo de la Formula 1. Deben disponer de la siguiente información.
  - Título
  - Imagen

- *Piloto:* Son los competidores que participan en la temporada de Formula 1. Deben disponer de la siguiente información.
    - Id
    - Nombre
    - Edad
    - Imagen
    - Escudería

- *Circuito:* Son aquellos lugares donde serán corridas las carreras de la temporada. Deben disponer de la siguiente información.
    - Id
    - País
    - Vuelta Rápida
    - Longitud
    - Imagen

- *Clasificación Piloto:* Es la tabla de las posiciones que tiene cada piloto en la temporada. Deben disponer de la siguiente información.
    - Posición
    - Nombre Piloto
    - Puntos
    - Escudería
    - Bandera

- *Clasificación Escudería:* Es la tabla de las posiciones que tiene cada escudería en la temporada. Deben disponer de la siguiente información.
    - Posición
    - Escudería
    - Puntos

2.1 DISEÑO

Lo primero que hice fue realizar un prototipo de la aplicación, estudié las necesidades que tenía que cumplir la aplicación.
Imaginé como sería la mejor manera para que el usuario pudiera acceder a todas las funciones de una manera sencilla e intuitiva.
También realicé un estudio sobre la apariencia de la aplicación en cuanto a como quería que se viese la interfaz. Para ello visite las páginas oficiales de la Formula 1 y aplicaciones similares para comprobar si seguían un mismo patrón de diseño y colores.

El objetivo de la interfaz de la aplicación es mantener al usuario el mayor tiempo posible y que disfrute de la navegabilidad y de la apariencia.

Para ello seguí los siguientes principios:

1. Diseño siguiendo el patrón de aplicaciones similares.
2. Prevención de errores (Dar feedback al usuario si algo falla)
3. Reconocer antes que recordar (Dando una estructura lógica, ordenada y reconocible)
4. Flexibilidad y eficiencia en el uso (Interfaz fácil de utilizar)

Para la realización de los prototipos se pueden utilizar ciertas herramientas y programas para generar las vistas de nuestra aplicación de una forma sencilla y en la que se comprueba de una forma visual si es realmente lo que queremos que vea el usuario o no. Pero en mi caso decidí realizarlo en papel ya que me era más fácil dibujar exactamente qué era lo que quería mostrar en la aplicación y como tenía que estar estructurada.

Antes de mostrar el resultado de las vistas de la aplicación voy a mostrar los colores principales elegidos en la aplicación. Basándome en los estudios realizados previamente.

Para el fondo de las vistas lo más apropiado eran colores oscuros, por lo que la opción más sencilla era optar por el color negro.

	Black	#000000	rgb(0, 0, 0)

Para iconos y ciertos títulos de la aplicación debía de ser un color llamativo por lo que la opción más fiable y que más se acercaba al mundo de la Formula 1 era el color rojo.

	Red	#FF0000	rgb(255, 0, 0)

Por último los textos de la aplicación se deben leer bien de forma que al usuario no le cueste esfuerzo llegar a leer la información de la aplicación. Ya que el fondo era negro lo mejor era optar por el color blanco.

	White	#FFFFFF	rgb(255, 255, 255)

El Login de la aplicación muestra muy bien la unificación de la paleta de colores (siguiendo la tendencia y patrón de la Formula 1).

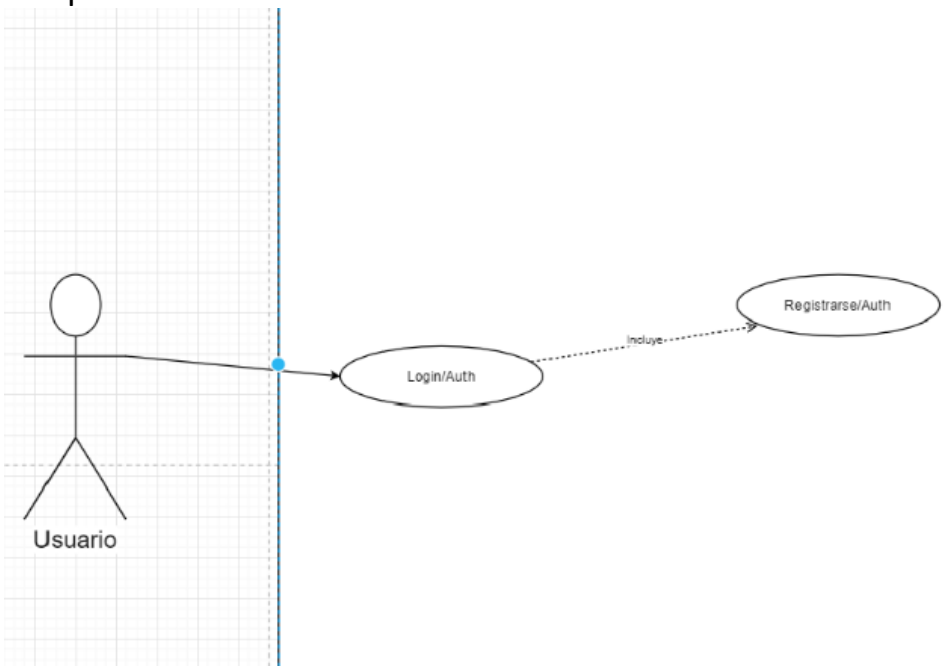
Vemos como el logo de la aplicación juega con la paleta de colores.

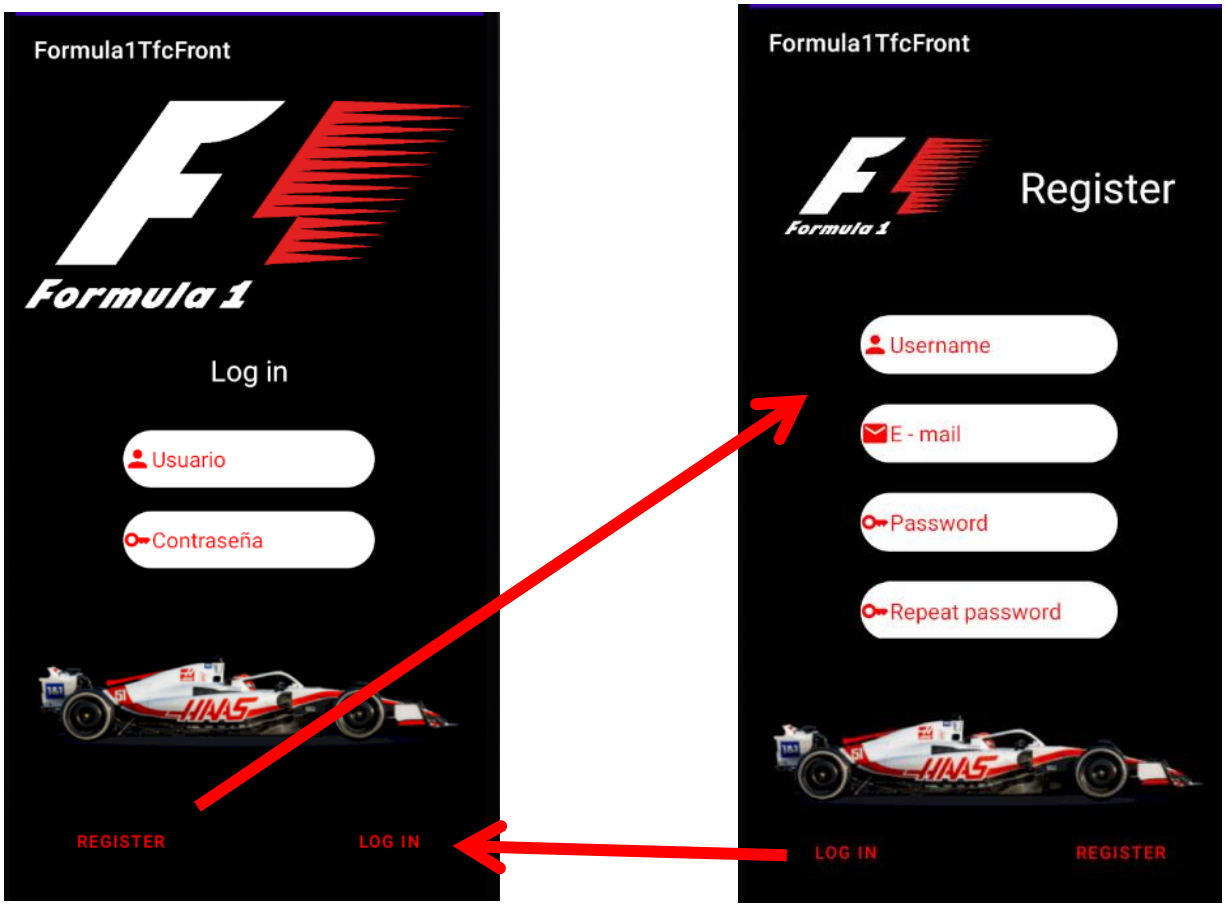
Los campos para iniciar sesión tienen fondo blanco para hacer contraste con el fondo negro.

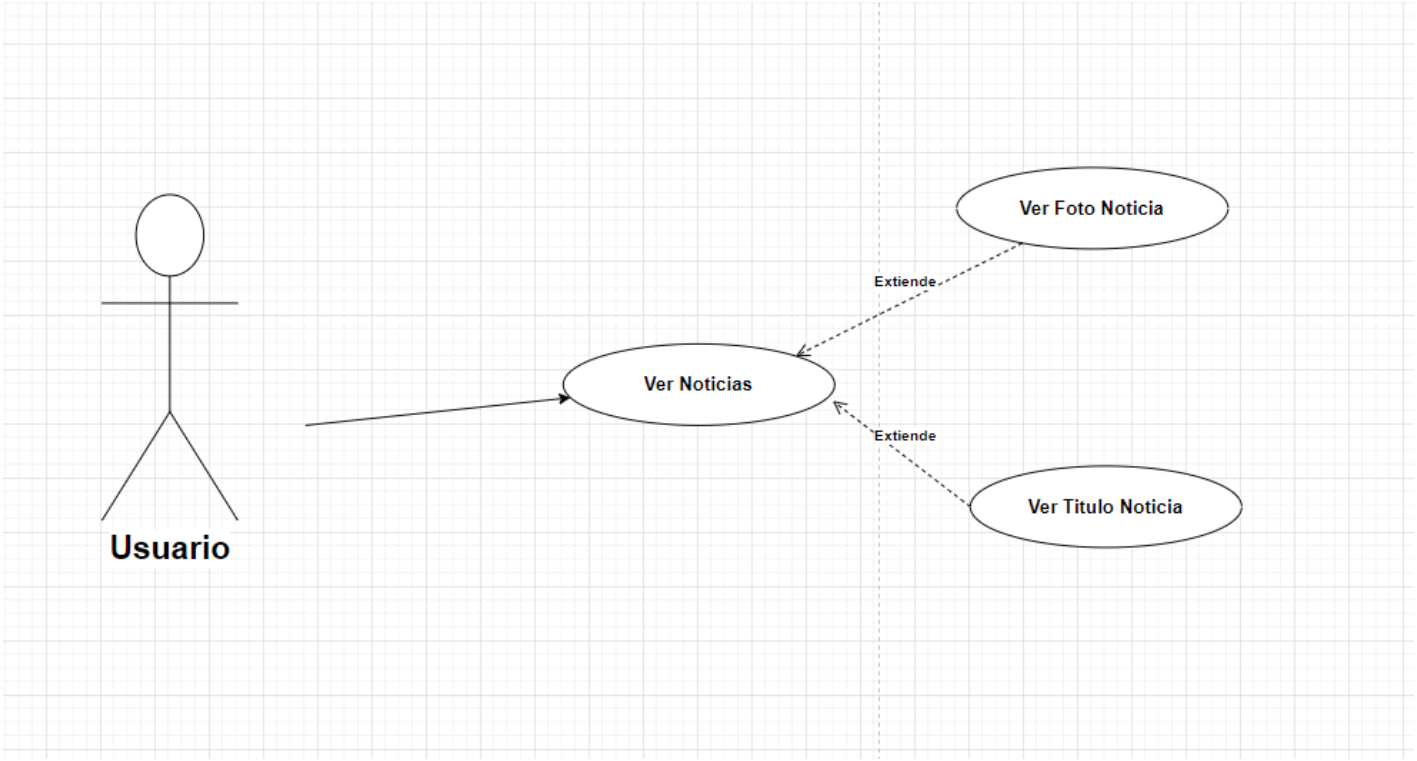
El coche elegido es un Haas, que cuenta con el patrón de colores a la perfección.

El fondo de la aplicación es negro y los botones tienen texto en rojo.

Para mostrar la aplicación de forma ordenada voy a realizarlo siguiendo los casos de uso por orden:



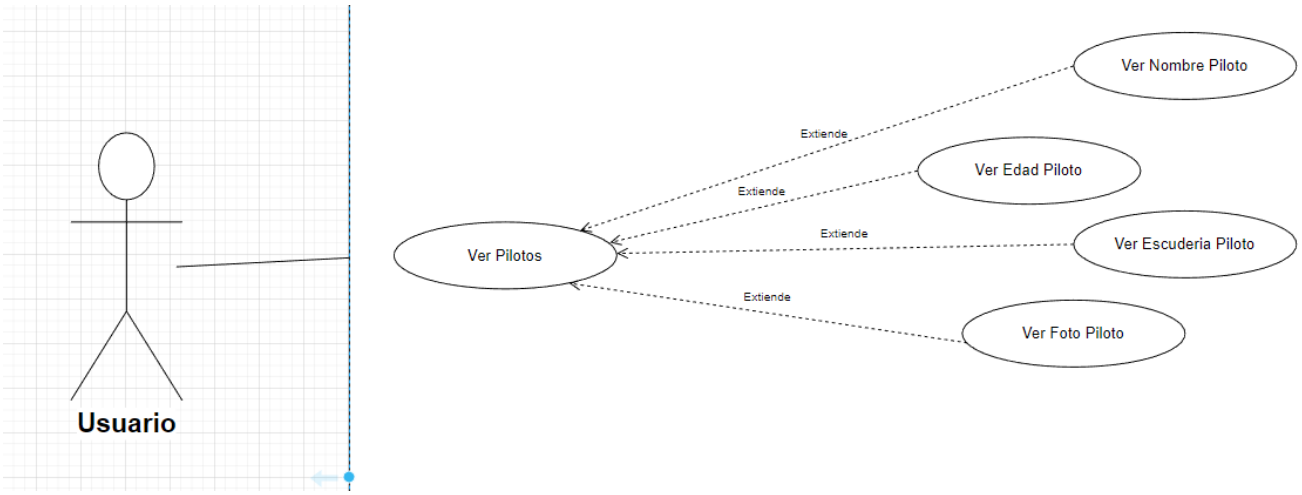


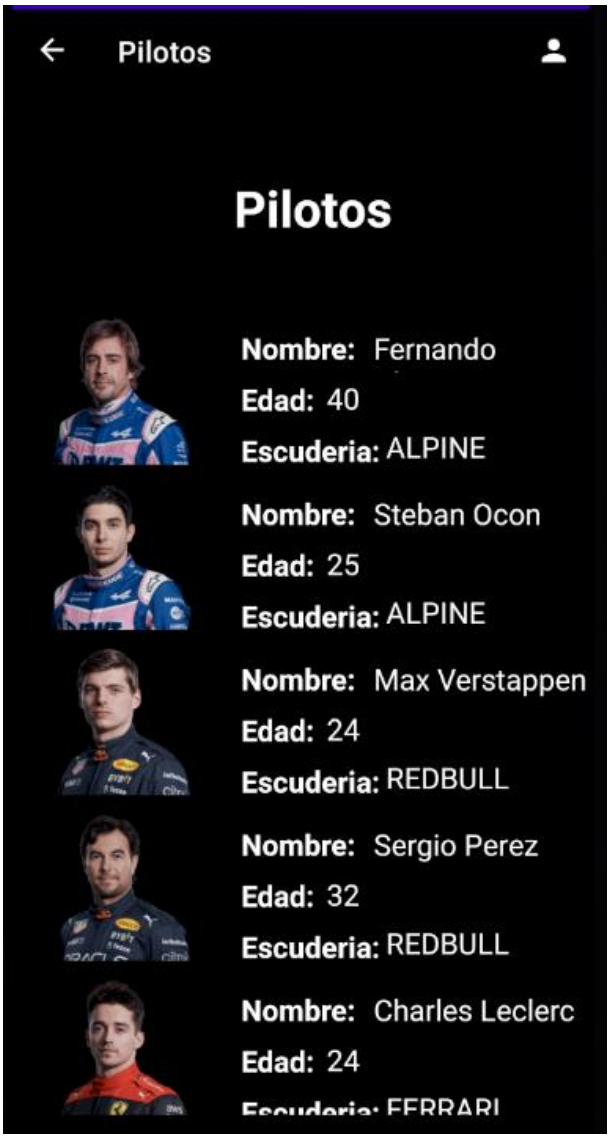


En esta pantalla podremos ver las últimas noticias publicadas en el mundo de la Formula 1.

Podemos ver tanto los títulos de las noticias como las imágenes.

Obteniendo toda la información realizando Scrapping Web de páginas webs de noticias.

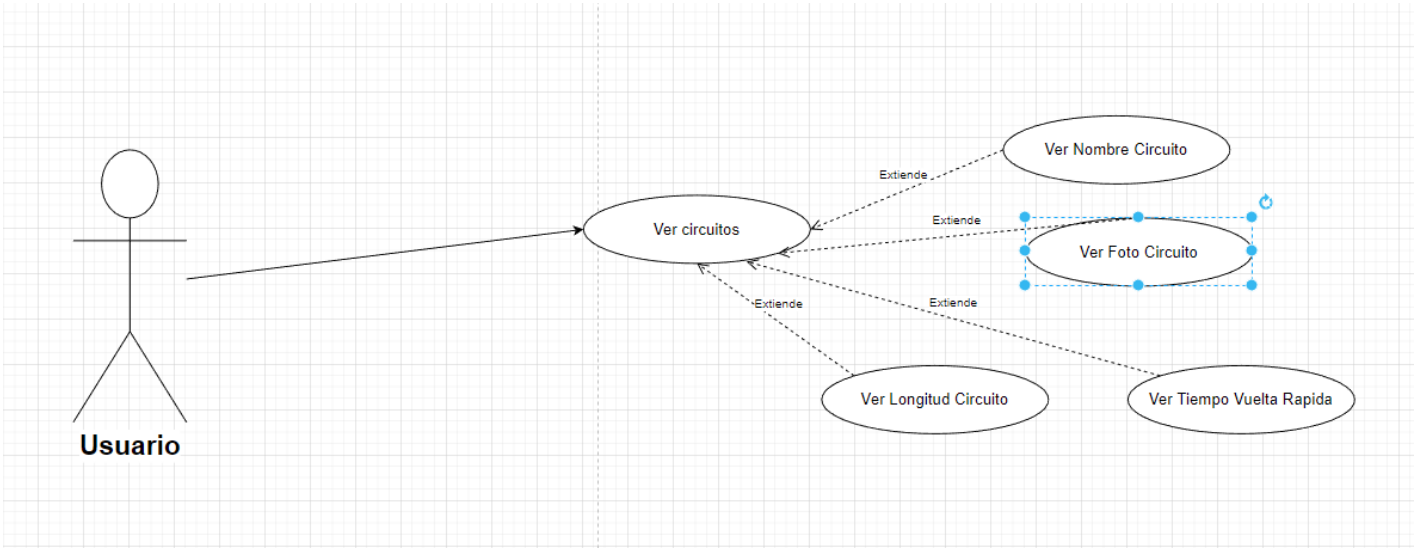


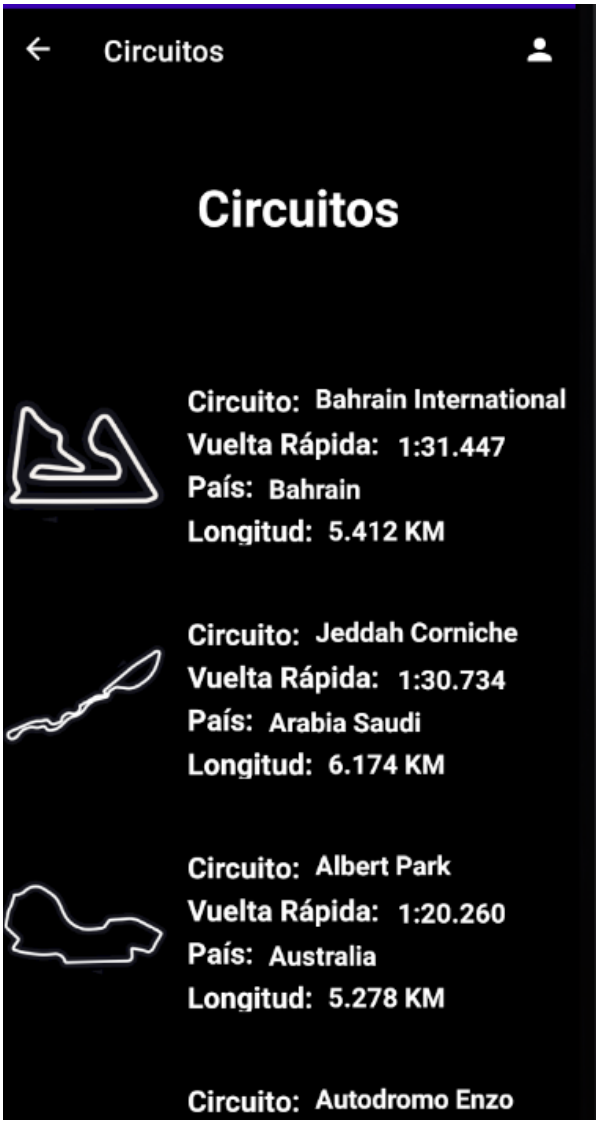


En esta pantalla podremos todos los pilotos que participan en el mundial de Fórmula 1.

Podemos ver tanto los nombres de cada uno de ellos como su edad, escudería e imagen.

Estos pilotos están insertados dentro de nuestro servidor, asique para obtenerlos simplemente realizamos una petición GET.

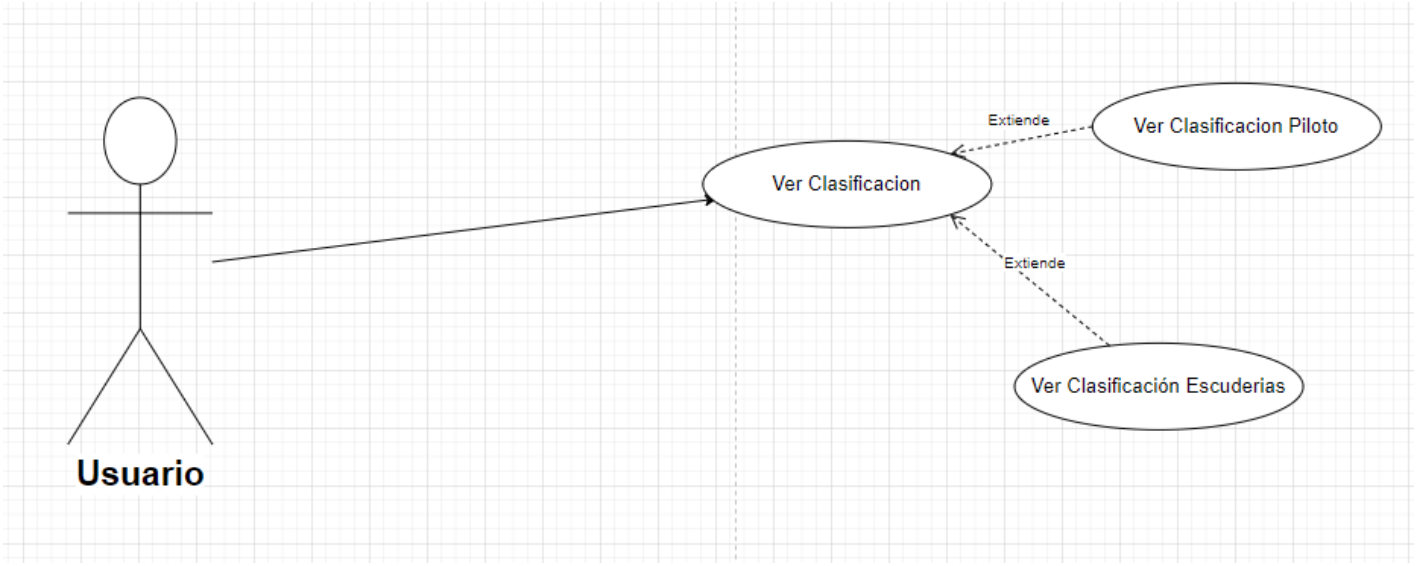




En esta pantalla podremos todos los circuitos en los que se van a realizar las carreras durante el mundial de Fórmula 1.

Podemos ver tanto los nombres de cada uno de ellos como su longitud, vuelta más rápida realizada en ese circuito y país en el que se encuentra.

Estos circuitos están insertados dentro de nuestro servidor, asique para obtenerlos simplemente realizamos una petición GET.



En esta pantalla podremos ver la clasificación actual del mundial de pilotos de la Fórmula 1.

Podemos tanto el orden de la clasificación como el nombre de cada piloto, los puntos que tiene, su nacionalidad y la escudería a la que pertenece.

Obteniendo toda la información realizando Web de la página oficial de la Fórmula 1.

En esta pantalla podremos ver la clasificación actual del mundial de escuderías de la Fórmula 1.

Podemos tanto el orden de la clasificación como el nombre de cada escudería y los puntos que tiene.

Obteniendo toda la información realizando Web de la página oficial de la Fórmula 1.

2.2 CÓDIGO

Una vez finalizada la parte de diseño voy a comentar los aspectos más relevantes del código. Comenzando por el backend en SpringBoot.

2.2.1 Aspectos Interesantes Spring Boot

Comenzamos por la configuración inicial del proyecto. Este es el aplication.properties, donde se configura el servidor y se establece la conexión con nuestra base de datos.

```
spring.data.mongodb.authentication-database=admin
#spring.data.mongodb.auto-index-creation=true
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=formula1
server.port=8080
spring.web.locale=es_ES
spring.web.locale-resolver=fixed
upload.root-location=upload-dir
#extras
#spring.data.mongodb.field-naming-strategy= # Fully qualified name of the FieldNamingStrategy to use.
#spring.data.mongodb.grid-fs-database= # GridFS database name.
spring.data.mongodb.repositories.enabled=true
spring.main.allow-bean-definition-overriding= true
#Enable Mongo repositories.
#spring.data.mongodb.uri=mongodb://mongoadmin:mongopass@localhost:27017/formula1
#spring.data.mongodb.uri=mongodb://localhost:8081 # Mongo database URI. When set, host and port are ignored.
spring.data.mongodb.username=mongoadmin
spring.data.mongodb.password=mongopass
#SWAGGER
spring.mvc.pathmatch.matching-strategy=ANT_PATH_MATCHER
#para visualizar la documentación -> http://localhost:8080/swagger-ui.html
```

Este es mi fichero .yml, este fichero lo que hace es levantar el servicio de mongo y mongoExpress en un contenedor de Docker.

```
version: '3.7'
services:
  mongodb-server:
    image: mongo
    container_name: mongodb-server
    ports:
      - 27017:27017
    expose:
      - 27017
    environment:
      MONGO_INITDB_ROOT_USERNAME: mongoadmin
      MONGO_INITDB_ROOT_PASSWORD: mongopass
      MONGO_INITDB_DATABASE: formula1
      command: --auth

    volumes:
      - mongodb-volume:/data/db
      - ./init:/docker-entrypoint-initdb.d

    networks:
      - mongo-network
  mongo-express:
    image: mongo-express
    container_name: mongo-express
    ports:
      - 8081:8081
    networks:
      - mongo-network
    depends_on:
      - mongodb-server

    environment:
      ME_CONFIG_MONGODB_ADMINUSERNAME: mongoadmin
      ME_CONFIG_MONGODB_ADMINPASSWORD: mongopass
      ME_CONFIG_MONGODB_SERVER: mongodb-server
    restart: unless-stopped
volumes:
   mongodb-volume:
networks:
```

POM, Listado de las dependencias utilizadas en el proyecto:

- SpringBoot Starter Data MongoDB
- Flapdoodle Embed Mongo
- SpringBoot Starter Security
- Spring Security Code
- JJWT Impl
- JJWT Api
- JJWT Jackson
- SpringBoot Starter Web
- SpringBoot Devtools
- Lombok
- SpringBoot Starter Test
- Spring Security Test
- SpringFox Swagger
- SpringFox Swagger UI
- ModelMapper
- Validation Api
- SpringBoot Starter Data JPA
- SpringBoot Starter
- Jsoup
- Commons Lang 3

Clase que configura Swagger

```
@Configuration
@EnableSwagger2
public class Swagger {
    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
                .select()
                .apis(
                        RequestHandlerSelectors
                                .basePackage("application")
                )
                .paths(PathSelectors.any())
                .build()
                .apiInfo(apiInfo());
    }

    @Bean
    public ApiInfo apiInfo() {
        return new ApiInfoBuilder()
                .title("Api Formula 1")
                .description("Api de Formula 1 con toda la informacion acerca de Pilotos, Escuderias, Circuitos, Carreras y Noticias.")
                .version("1.0")
                .contact(new Contact("Saúl Mellado Herrera", "https://github.com/saulmella12", "saulmella12@gmail.com"))
                .build();
    }
}
```

Configuración CORS:

```
@Configuration
public class CorsConfig {
    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurer() {

            @Override
            public void addCorsMappings(CorsRegistry registry) {
                registry.addMapping("/**")
                        .allowedMethods("GET", "POST", "PUT", "DELETE")
                        .maxAge(3600);
            }

        };
    }
```

Security Config

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
            .csrf().disable()
            .exceptionHandling().authenticationEntryPoint(jwtAuthenticationEntryPoint)
            .and()
            .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
            .and()
            .authorizeRequests()
            .antMatchers(HttpMethod.POST, "/auth/login").permitAll()
            .antMatchers(HttpMethod.GET, "/usuario/**").permitAll()
            .antMatchers(HttpMethod.GET, "/piloto/**").permitAll()
            .antMatchers(HttpMethod.POST, "/piloto/**").hasRole("ADMIN")
            .antMatchers(HttpMethod.PUT, "/piloto/**").hasRole("ADMIN")
            .antMatchers(HttpMethod.DELETE, "/piloto/**").hasRole("ADMIN")

            .antMatchers(HttpMethod.GET, "/circuito/**").permitAll()
            .antMatchers(HttpMethod.POST, "/circuito/**").hasRole("ADMIN")
            .antMatchers(HttpMethod.PUT, "/circuito/**").hasRole("ADMIN")
            .antMatchers(HttpMethod.DELETE, "/circuito/**").hasRole("ADMIN")

            .antMatchers(HttpMethod.GET, "/noticia/**").permitAll()
            .antMatchers(HttpMethod.POST, "/noticia/**").hasRole("ADMIN")
            .antMatchers(HttpMethod.PUT, "/noticia/**").hasRole("ADMIN")
            .antMatchers(HttpMethod.DELETE, "/noticia/**").hasRole("ADMIN")

            .antMatchers(HttpMethod.GET, "/clasificacion/**").permitAll()
            .antMatchers(HttpMethod.POST, "/clasificacion/**").hasRole("ADMIN")
            .antMatchers(HttpMethod.PUT, "/clasificacion/**").hasRole("ADMIN")
            .antMatchers(HttpMethod.DELETE, "/clasificacion/**").hasRole("ADMIN")

            .anyRequest().authenticated();

    http.addFilterBefore(jwtAuthorizationFilter, UsernamePasswordAuthenticationFilter.class);
}
```

Clasificación Service

```
@Service
public class ClasificacionService {

    private String urlReal = "https://www.formula1.com/en/results.html/2022/drivers.html";
    private String urlReal2 = "https://www.formula1.com/en/results.html/2022/team.html";

    Set<ClasificacionPiloto> responseDTOS = new HashSet<>();
    Set<ClasificacionEscuderia> responseDTOS2 = new HashSet<>();

    public Set<ClasificacionPiloto> extraerClasificacionPilotos(){
        try{
            Document document = Jsoup.connect(urlReal).get();

            Elements element = document.getElementsByTag("tr");

            Elements prueba =  new Elements();

            prueba.addAll(element);


            for (Element ads: prueba){
                ClasificacionPiloto responseDTO = new ClasificacionPiloto();
                responseDTO.setNombrePiloto(ads.getElementsByClass("hide-for-tablet").html()+" "+ads.getElementsByClass("hide-for-m
                responseDTO.setBandera(ads.getElementsByClass("dark semi-bold uppercase").text());
                responseDTO.setPuntos(ads.getElementsByClass("dark bold").html());
                responseDTO.setEscuderia(ads.getElementsByClass("grey semi-bold uppercase ArchiveLink").html());
                responseDTO.setPosicion(ads.getElementsByClass("limiter").next().html());

                if (!Objects.equals(responseDTO.getNombrePiloto(), "") && !Objects.equals(responseDTO.getPuntos(), "")){
                    responseDTOS.add(responseDTO);
                }
            }

        } catch (IOException e) {
```

Noticia Service

```
@Service
public class NoticiaService {
    //@Value("${paginaWeb}")
    private String urlReal = "https://www.formulaf1.es/";

    Set<Noticia> responseDTOS = new HashSet<>();

    public Set<Noticia> extraerNoticiasPagina2(){
        try{
            Document document = Jsoup.connect(urlReal).get();

            Element element = document.getElementsByAttributeValue("id","td_uid_9_62975726aa3c0").first();
            Elements elements = element.getElementsByTag("img");

            Elements prueba =  new Elements();

            prueba.addAll(elements);


            for (Element ads: prueba){
                Noticia responseDTO = new Noticia();

                responseDTO.setTitulo(ads.attr("title"));
                responseDTO.setImagen(ads.attr("data-img-url"));

                if (!Objects.equals(responseDTO.getTitulo(), "") && !Objects.equals(responseDTO.getImagen(), "")) {
                    responseDTOS.add(responseDTO);
                }
            }
```

2.2.2 Aspectos Interesantes Android

Ahora voy a comentar los aspectos más interesantes de la parte front, de la parte Android.

Build.Graddle, Imports necesarios para el correcto funcionamiento:

- RetroFit 2
- Glide
- OkHttp 3
- OkHttp 3 Loggin Interceptor

También es imprescindible que en el fichero AndroidManifest.xml asignemos permisos de acceso a internet al usuario para poder realizar las peticiones a nuestro servidor.

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Clase de configuración para conectar a la API.

```
public class ApiConfig {
    private static final String URL="http://10.0.2.2:8080";

    public static Retrofit getClient(){
        HttpLoggingInterceptor loggingInterceptor = new HttpLoggingInterceptor();

        loggingInterceptor.setLevel(HttpLoggingInterceptor.Level.BODY);

        OkHttpClient okHttpClient = new OkHttpClient.Builder()
                .addInterceptor(loggingInterceptor)
                .build();


        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl(URL)
                .addConverterFactory(GsonConverterFactory.create())
                .client(okHttpClient)
                .build();

        return retrofit;
    }
}
```

Clase que contiene todas las llamadas a nuestra API.

```
@GET("/piloto/all")
Call<List<Piloto>> obtenerTodosPilotos();

@GET("/circuito/all")
Call<List<Circuito>> obtenerTodosCircuitos();

@GET("/noticia/all")
Call<List<Noticia>> obtenerNoticias();

@GET("/piloto/all")
Call<List<Piloto>> obtenerPilotos();

@GET("/circuito/all")
Call<List<Circuito>> obtenerCircuitos();

@GET("/clasificacion/all")
Call<List<ClasificacionPiloto>> obtenerClasificacionPilotos();

@POST("/cliente/post")
Call<Usuario> crearUsuario(@Body Usuario usuario);

@PUT("/cliente/put")
Call<Usuario> reservaUsuario(@Body Usuario usuario);
```

Ejemplo de una petición a nuestro servidor, en este caso voy a mostrar la petición para obtener todos los pilotos.

```
Call<List<Piloto>> allPilotos = api.obtenerPilotos();

allPilotos.enqueue(new Callback<List<Piloto>>() {
    @Override
    public void onResponse(Call<List<Piloto>> call, Response<List<Piloto>> response) {
        pilotosList = response.body();
        adapter = new PilotoRecyclerAdapter(pilotosList,getActivity());
        recyclerView.setAdapter(adapter);
    }

    @Override
    public void onFailure(Call<List<Piloto>> call, Throwable t) {
        Toast.makeText(getActivity(), text: "Fallo al comunicar con la base de datos", Toast.LENGTH_SHORT).show();
    }
});
```

2.3 IMPLANTACIÓN

El empaquetado de la aplicación será mediante un fichero .apk ya que al ser una aplicación móvil es elformato que ha de tener para su distribución. Para generar este fichero apk basta con acceder a las opciones que trae por defecto Android Studio.

El servidor de la base de datos será desplegado mediante Docker, ya que nos permite lanzar el contenedor cuando queramos sin necesidad de instalarlo dentro de nuestro equipo y además podemos lanzarlo en cualquier equipo simplemente ejecutando el fichero .yml

La api será desplegada desde nuestro propio IDE, en mi caso IntelliJ ya que cuenta con un montón de funcionalidades y su perfecto funcionamiento es ideal para el desarrollo del backend.

2.4 DOCUMENTACIÓN

Toda la documentación del proyecto se encuentra disponible en el repositorio del proyecto:

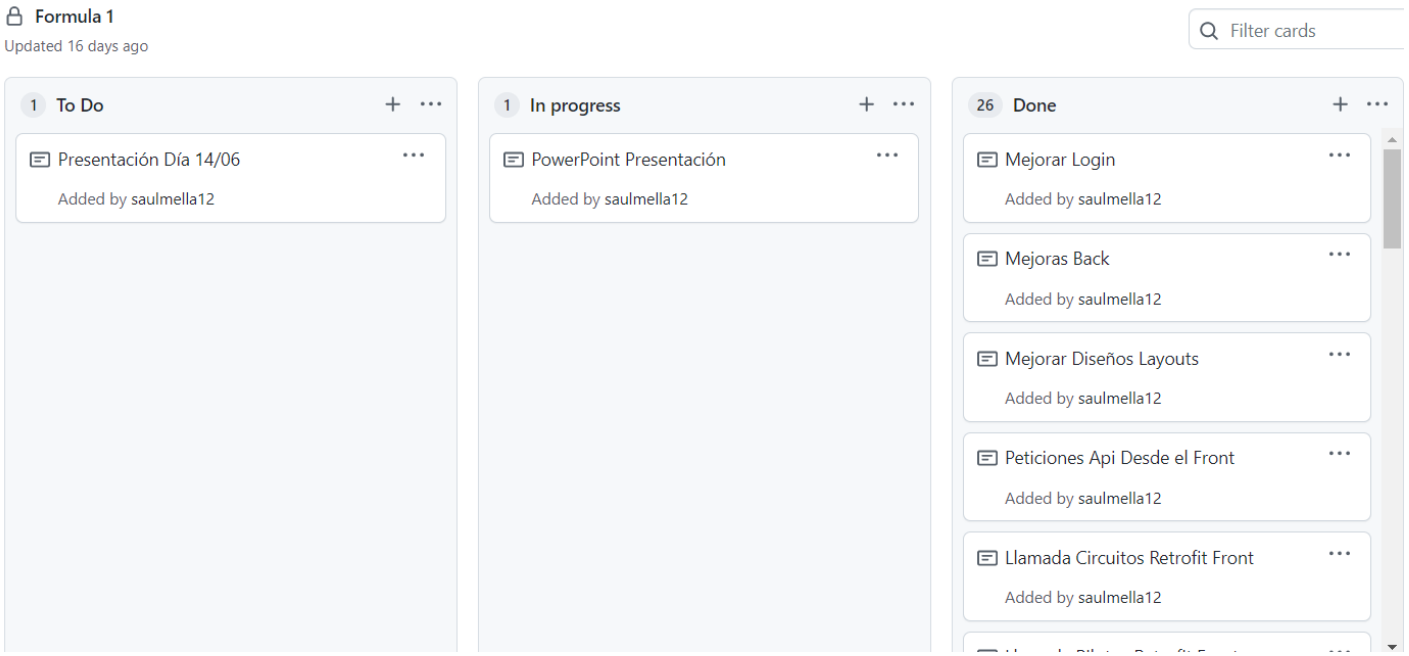
- https://github.com/saulmella12/Formula1

Aquí podrás encontrar la siguiente información:

- WIKI del proyecto en la que encuentras la misma información que en este documento.
- Además también se encuentra el diagrama de casos de uso.
- El diagrama de modelo de datos también se encuentra allí.
- También puedes ver los bocetos de la aplicación.
- Manual de usuario.
- Apartado Projects donde puedes consultar todas las tareas asignadas del proyecto.

3 RESULTADOS Y DISCUSIÓN

El desarrollo me lo planifiqué para que me llevara más o menos hasta la primera semana de Junio. Me organicé mediante el apartado de Proyects de GitHub, en el que puedes ir asignando tareas e ir moviendolas por el tablero si ya las has completado o si están en desarrollo o si todavía no las has empezado.



La mayor dificultad que me he encontrado durante el desarrollo ha sido sin ninguna duda el sacar tiempo para dedicarle al proyecto. Al estar haciendo las FCT a la vez que el TFG se complicaba muchos días el poder continuar realizando el proyecto.

Tuve también problemas en alguna petición en la API que me fallaba al implementar el Security Config.

A la hora de cifrar la contraseña en la base de datos también tuve problemas con el @Bean ya que me daba una referencia múltiple. Tarde varias horas en encontrar el error.

En la parte del front mis mayores problemas quizás han sido darle una buena apariencia.
También tuve un problema a la hora de hacer las peticiones a la API, me daban error constantemente hasta que me di cuenta de que el error era que no había asignado permisos de internet en el AndroidManifest.

4 TRABAJO FUTURO

En un trabajo fututo para seguir implementando funcionalidades y mejoras a la aplicación me gustaría realizar las siguientes actualizaciones:

- Opcionalmente me gustaría realizar un sistema de notificaciones para cuando se publique una nueva noticia (Probablemente sería realizado con Firebase).
- Posibilidad de poder filtrar esas notificaciones por los usuarios, es decir, que los usuarios puedan elegir qué tipo de notificaciones desean recibir (cuando empieza una carrera, noticias sobre un cierto piloto / escudería etc.).
- Además me gustaría realizar una aplicación de escritorio orientado a administradorespara poder gestionar los usuarios.
- También me gustaría que los usuarios pudieran iniciar sesión con sus propias cuentas deGoogle sin necesidad de tener que registrarse mediante un correo y contraseña especifico (Probablemente OAuth 2).
- Me gustaría también agregar un calendario de carreras con un contador que te indique cuanto tiempo falta para la siguiente.
- Además me gustaría implementar una funcionalidad para poder seguir la carrera en directo, es decir, ver en tiempo real la clasificación de la carrera, vueltas realizadas, tiempos de vuelta etc.…
- Mostrar mas detalles acerca de cada piloto, por ejemplo todas sus carreras ganadas, numero de mundiales, histórico de escuderías, numero de pódiums etc.
- Además me gustaría mostrar más información acerca de los circuitos, mas fotos, localización de donde está situado, numero de carreras realizadas en ese circuito.

4 CONCLUSIONES

Al principio cuando me di cuenta de que tenía que realizar el proyecto no sabía ni sobre que iba a hacerlo. Tras estar días pensando y hablar con profesores decidí hacerlo sobre uno de mis hobbies como es la Formula 1.

Al principio no sabía muy bien si iba a ser capaz de realizar este proyecto solo cumpliendo con el plazo de entrega y abordando toda la funcionalidad que debía realizar y con la que me comprometí en el anteproyecto.

Alguna de estas funcionalidades ya las habíamos visto en clase por lo que no tenía mucho miedo de realizarlas, pero otras como por ejemplo el scrapper web me asustaba un poco ya que no tenía ningún tipo de conocimiento sobre ello.

Al final estamos en un mundo que está en constante desarrollo y cada día salen tecnologías y librerías nuevas al mercado y es nuestra responsabilidad como programadores el aprender a dominar todas estas tecnologías y no tener temor por el echo de no entender al principio cómo funciona.

Las primeras semanas me estuve formando acerca de estas tecnologías y librerías y la verdad que lo que parecía que iba a ser complicado resultó en una experiencia super gratificante. No aprendí solamente a dominarla si no que también aprendí que al final todo es ponerse y al final lo sacas.

Al inicio del desarrollo del proyecto primero prototipé la aplicación, esto me ayudó mucho a decidir que partes tenía que realizar en el backend para obtener el objetivo que yo tenía.

Una vez que tenía claro que necesitaba comencé con el desarrollo del backend. Al haber utilizado Spring Boot en clase no se me hizo muy complicado realizar toda la lógica de la API.

Cree las clases que necesitaba ajustándose a mi diagrama de modelo de datos y realicé todas las peticiones que iba a necesitar. Añadí alguna opcional para una posterior actualización de la aplicación.

Resumidamente el desarrollo del backend no me resultó muy complicado. Simplemente un poco tedioso cuando fallaba alguna consulta a la API por algún motivo.

El desarrollo del front me resultó más pesado, elegí realizarlo en Android ya que fue una asignatura que durante el curso no me llamaba demasiado la atención y me apetecía reforzar conocimientos en ella.

Lo primero que hice fue todos los layouts que iban a tener mi aplicación siguiendo los bocetos que realicé al principio.
Una vez que cree todos los layouts comencé con el desarrollo de la aplicación, ajuste toda la navegabilidad dependiendo de donde hiciera click el usuario.

Por último realicé todas las llamadas a la API para mostrar los datos que quería. Ya que el backend me aseguré de que funcionara a la perfección no tuve demasiados problemas a la hora de hacer las consultas.

Resumidamente el desarrollo del front no fue difícil pero si fue aburrido en mi opinión, pero me ha servido bastante para afianzar conocimientos obtenidos durante el curso y conocimientos nuevos que he ido aprendiendo durante el desarrollo.

En conclusión, ha sido una experiencia en la que me he dado cuenta de que aunque no supiese hacer todo al principio, con esfuerzo y dedicación todo se puede lograr. Estoy contento por haber podido realizar una aplicación, incluyendo tanto la parte front como la parte backend.

Es cierto que el desarrollo ha sido bonito pero quizás un poco agobiante en algunos aspectos ya que combinar las FCT con el desarrollo del TFG puede ser complicado ya que no dispongo de tanto tiempo para realizarlo. Y al ser un proyecto FullStack se hacía mucho más complejo.

5 BIBLIOGRAFÍA

- https://blog.codmind.com/que-es-spring-boot/#:~:text=Spring%20Boot%20es%20una%20tecnolog%C3%ADa,servidor%20de%20aplicaciones%20y%20enfocarnos
- https://www.arquitecturajava.com/que-es-spring-boot/
- https://openwebinars.net/blog/que-es-mongodb/
- https://jarroba.com/scraping-java-jsoup-ejemplos/
- https://developer.android.com/guide/topics/ui/dialogs?hl=es-419
- https://iconos8.es/icons/set/casco
- https://developer.android.com/studio/write/resource-manager?hl=es-419
- https://mscdroidlabs.es/como-crear-un-boton-con-bordes-redondos-en-android/
- https://www.iteramos.com/pregunta/5630/como-crear-edittext-con-esquinas-redondeadas
- http://www.pmoinformatica.com/2017/02/requerimientos-funcionales-ejemplos.html
- http://www.pmoinformatica.com/2015/05/requerimientos-no-funcionales-ejemplos.html
- https://es.slideshare.net/glendita77/requisitos-de-la-informacin
- https://www.uup.es/blog/10-principios-de-usabilidad/
- https://htmlcolorcodes.com/es/