Aplicación para visualizar datos y estadísticas de los campeonatos de F1 en dispositivos Android

Jorge Núñez Vázquez
Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles
Desarrollo de aplicaciones para dispositivos Android

Consultor/a: Francesc D´Assís Giralt Queralt
Profesor/a responsable de la asignatura: Carles Garrigues Olivella
05/06/2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Aplicación para visualizar datos y estadísticas de los campeonatos de F1 en dispositivos Android*
Nombre del autor:	*Jorge Núñez Vázquez*
Nombre del consultor/a:	*Francesc D´Assís Giralt Queralt*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	06/2019
Titulación:	*Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles*
Área del Trabajo Final:	*Trabajo Final de Máster - Desarrollo de Aplicaciones para Dispositivos Android*
Idioma del trabajo:	*Castellano*
Palabras clave	*Android, F1, GP*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Este trabajo de fin de máster tiene como objetivo poner en práctica los conocimientos y competencias adquiridas a lo largo de toda la titulación a través de la elaboración de una aplicación para dispositivos móviles.

La temática escogida es la creación de una aplicación para dispositivos móviles de la plataforma Android que mostrará al usuario datos y estadísticas históricas de los campeonatos de Fórmula 1 (F1), así como de sus pilotos y escuderías.

Para obtener los datos, la aplicación se conecta a través de la red con un Servicio Web existente de uso libre no comercial (https://ergast.com/mrd/), que proporciona datos de los campeonatos de F1 desde el inicio del campeonato del mundo, en 1950.

Para desarrollar el código de la aplicación se han implementado las arquitecturas CLEAN y MVP, en el que mediante la definición de capas de abstracción en el código se han creado diferentes módulos siguiendo los principios SOLID recomendados por la comunidad de desarrolladores, con el objeto de que la aplicación sea más fácil de testear, ampliar, modificar y mantener. El lenguaje seleccionado para el desarrollo ha sido Kotlin, por sus múltiples ventajas de seguridad y minimización de código con respecto a Java.

Tras la ejecución del proyecto, se ha obtenido un producto funcional y con una buena aceptación por parte de los usuarios, logrando los objetivos marcados.

Abstract (in English, 250 words or less):

This project aims to put into practice the knowledge and skills acquired throughout the degree through the development of an application for mobile devices.

The chosen theme is the creation of an application for mobile devices of the Android platform that will show the user data and historical statistics of the Formula 1 (F1) championships, as well as their pilots and teams.

To obtain the data, the aplicación will connect through the network with an existing non-commercial Web Service (https://ergast.com/mrd/) that provides data of the F1 championships since the start of the world championship, in 1950.

To implement the aplicaciónlication code we will use a CLEAN architecture and a MVP pattern, in which by defining layers of abstraction in the code we will create different modules following the SOLID principles recommended by the developer community, with the aim that the aplicación is easier to test, expand, modify and maintain.

The language selected for development will be Kotlin, due to its multiple security advantages and code minimization with respect to Java.

After the execution of the project, a functional product has been obtained and with a good acceptance by the users, achieving the marked objectives.

Índice de Contenidos

Índice de Tablas

Índice de Imágenes

1. Introducción

1.1 Contexto y justificación del Trabajo

Este trabajo de fin de máster tiene como objetivo poner en práctica los conocimientos y competencias adquiridas a lo largo de toda la titulación a través de la elaboración de una aplicación para dispositivos móviles (*app*).

La temática escogida es la creación de una aplicación para dispositivos móviles de la plataforma Android que mostrará al usuario datos y estadísticas históricas de los campeonatos de Fórmula 1 (F1), así como de sus pilotos y escuderías.

Para obtener los datos, la aplicación se conectará a través de la red con un Servicio Web (WS) existente de uso libre no comercial (https://ergast.com/mrd/), que proporciona datos de los campeonatos de F1 desde el inicio del campeonato del mundo, en 1950.

La F1 es un deporte con millones seguidores en todo el planeta, alcanzando una cifra de audiencia global acumulada de 1.758 millones de telespectadores en el campeonato de 2018, por lo que es un mercado interesante de potenciales usuarios. El objetivo de la aplicación será mostrar los datos del campeonato de F1 actual, así como de las escuderías y sus pilotos.

Actualmente existen varias aplicaciones en el mercado con un formato similar, algunas de ellas con muy buenas valoraciones, las cuales obtienen los datos del mismo WS[1]. Para aportar valor a la aplicación, se hará un breve estudio de mercado para identificar los puntos fuertes y débiles de la posible competencia, y definir acciones en el resto de las fases del proyecto en función de las conclusiones obtenidas.

Para desarrollar la aplicación la plataforma escogida será Android, la plataforma desarrollada por Google. Con una cuota mundial de casi el 75% de los usuarios[2], Android es actualmente el sistema operativo móvil más empleado del mundo, y dado que el mercado potencial de esta aplicación se encuentra muy repartido a nivel mundial, parece casi obligada su elección, más teniendo en cuenta de que se trata de la plataforma más empleada en países como Brazil[3] o China[4], que tienen los mayores índices de audiencia de este deporte[5].

[1] https://ergast.com/mrd/gallery/
[2] http://gs.statcounter.com/os-market-share/mobile/worldwide
[3] http://gs.statcounter.com/os-market-share/mobile/brazil
[4] http://gs.statcounter.com/os-market-share/mobile/china
[5] https://www.formula1.com/en/latest/article.formula-1s-tv-and-digital-audiences-grow-for-the-second-year-running.OqTPVNthtZKFbKqBaimKf.html

También se ha valorado la posibilidad de realizar una aplicación híbrida con frameworks Javascript como Ionic o React, pero a pesar de su gran evolución, la fluidez de ejecución de una aplicación con estas plataformas aún no consigue igualar a la de una ejecutada de forma nativa.

Otra de las razones para seleccionar esta plataforma, es el nivel de experiencia del autor con la plataforma Android, el cual es bastante mayor que con el resto, que también es un factor a tener en cuenta a la hora de plantear un proyecto con importantes requisitos de calidad.

1.2 Objetivos del Trabajo

El principal objetivo de este proyecto es el desarrollo de una aplicación para dispositivos móviles de la plataforma Android que muestre al usuario los datos del campeonato de F1, así como datos históricos de este, los cuales obtendrá de un WS de acceso público, y que a mayores incluya un comparador de datos de escuderías y pilotos.

Tras un breve estudio de mercado que desarrollaremos en el siguiente capítulo, hemos definido los siguientes objetivos:

1. Objetivos **funcionales** de la aplicación:
    - Proporcionar información del campeonato de F1 actual, o del último disputado, mostrando un listado de los Grandes Premios (GP).
    - Mostrar el detalle de un GP al pulsar sobre uno de ellos, incluyendo datos como la configuración del podio o la vuelta rápida.
    - Mostrar la evolución del campeonato con la clasificación de puntos tanto del campeonato de pilotos como de constructores
    - Mostrar el detalle de un piloto o de un constructor al pulsar sobre uno de ellos, proporcionando los datos más relevantes de la temporada actual, como el nº de victorias, puntos acumulados, carreras finalizadas, mejor posición, etc.
    - Proporcionar un comparador de estadísticas en el que el usuario pueda seleccionar entre dos pilotos o dos escuderías, actuales o históricas, y mostrar gráficamente una comparativa entre ambos de los datos más relevantes entre los muchos que nos proporciona el WS, como pueden ser: campeonatos ganados, victorias en GP, pole positions, vueltas rápidas, puntos ganados, etc.

2. Objetivos **no funcionales** del proyecto:
    - Crear una aplicación para dispositivos de la plataforma Android, empleando algunos de los patrones más recomendados en la actualidad por la comunidad de desarrolladores, como las arquitecturas CLEAN[6] y MVP[7], que ayudan a desarrollar aplicaciones más robustas, ampliables, y más fáciles de testear y mantener.
    - Desarrollar la aplicación empleando Kotlin[8], el lenguaje desarrollado por JetBrains, ya que proporciona interesantes características de legibilidad, seguridad y minimización de código.
    - Desarrollar test unitarios que faciliten el testeo automático de la aplicación aprovechando las características de las arquitecturas anteriormente citadas
    - Crear una aplicación atractiva, que sea sencilla de usar, y con un diseño agradable e intuitivo centrado en la experiencia de usuario (UX).

1.3 Enfoque y método seguido

Dentro de las posibles estrategias para el desarrollo de la aplicación, destacan el desarrollo híbrido y el nativo. Las aplicaciones híbridas se ejecutan sobre un WebView (Ionic, PhoneGap,…) o una máquina virtual JavaScript (React Native, Nativescript…) y ofrecen la ventaja de que se puede emplear el mismo código para varias plataformas, con la consiguiente reducción de costes, mientras que el desarrollo nativo aporta una mayor fluidez de ejecución al ejecutarse de forma *nativa* en el dispositivo, es decir, compilando a código máquina. Esta diferencia de fluidez es más notable en dispositivos de gama media y baja.

Para desarrollar este proyecto, se ha optado por el desarrollo nativo en la plataforma Android porque se ajusta al contexto del proyecto, proporcionará una mayor fluidez de ejecución al usuario y es ideal para poner en práctica todas las enseñanzas adquiridas a lo largo de la titulación.

Una vez seleccionados los objetivos y la plataforma de desarrollo, el siguiente paso será definir una planificación del proyecto en función de los requisitos del mismo, las tareas a realizar, los recursos necesarios y el tiempo disponible.

Antes de desarrollar la aplicación, se realizará una fase de diseño del producto basado en la experiencia de usuario. Para ello, se estudiarán en detalle los

[6] http://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html
[7] https://www.raywenderlich.com/7026-getting-started-with-mvp-model-view-presenter-on-Android
[8] https://kotlinlang.org/

posibles perfiles de usuario de la aplicación y las *apps* existentes en el mercado. A partir de los datos obtenidos se definirá el listado de acciones que la aplicación ejecutará, y en base a éstos se realizarán los *wireframes* del diseño de la aplicación. Primero en baja definición para poder realizar cambios más rápidamente, y después con un software de alta definición, con el que se pondrán exportar los diseños que servirán como guía para el desarrollo.

Dentro de las metodologías de desarrollo posibles, el desarrollo basado en la filosofía *agile*[9] y los principios SOLID[10] proporcionará simplicidad y calidad tanto al código como al proceso de desarrollo de la aplicación. Para ello, se emplearán patrones de arquitectura como CLEAN y MVP, intentando aprovechar las características que ofrecen de cara a testear la aplicación, y para su posterior mantenimiento o ampliación.

Se empleará un repositorio GIT para llevar un control de los cambios que se realicen en el código, que permitirá estructurar el desarrollo en fases concretas y proporcionará una mayor seguridad en caso de tener que revertir cambios a una versión anterior.

El diseño UX y el desarrollo *agile* son algunas de las prácticas más recomendadas entre la comunidad de desarrolladores, por su comprobada efectividad en proyectos de desarrollo de software y la gestión de los mismos, en términos de calidad, velocidad de desarrollo, solidez de código y seguridad, por lo que se considera una estrategia adecuada para el desarrollo de este proyecto.

[9] https://agilemanifesto.org/
[10] https://es.wikipedia.org/wiki/SOLID

1.4 Planificación del Trabajo

A continuación se van a describir los recursos necesarios y las tareas a realizar durante el desarrollo del proyecto, así como los riesgos y las medidas preventivas, para realizar una planificación temporal de cada uno de los hitos del proyecto:

Recursos Necesarios para ejecutar el proyecto	
Humanos	Diseñador UX Desarrollador Android con conocimientos de Kotlin, testing, y arquitecturas CLEAN y MVP
Materiales	Un ordenador portátil Un dispositivo Android para pruebas Papel, lápiz y bolígrafo para elaborar los sketches
Software	Mock/Sketches: Ninjamock Diseño HD: Axure Desarrollo: Android Studio Control de versiones: Bitbucket, Sourcetree (GIT) Librerías: Room, Picasso, Retrofit2, Mockito,… (se detallarán todas las librerías empleadas en el capítulo 3. Desarrollo)

Tabla 1. Recursos necesarios para ejecutar el proyecto

El mayor coste de los recursos necesarios recae en la especialización de los perfiles técnicos, ya que deben dominar varias disciplinas y tecnologías, y en el coste del equipo informático. La mayoría de los programas de software citados son de uso abierto y no tienen ningún tipo de coste anual, lo cual es una gran ventaja a la hora de plantearse el desarrollo de un proyecto de estas características con carácter emprendedor. Axure es un software que sí requiere una suscripción anual, pero por suerte dispone de una licencia gratuita para estudiantes que será la empleada en este proyecto.

Una vez identificados los recursos necesarios, se realizará una estimación de los posibles riesgos que pueden provocar desviaciones en la planificación, así como su probabilidad e impacto, y las posibles medidas planteadas para compensar dichas desviaciones, para tenerlas en cuenta a la hora de realizar nuestra planificación temporal:

Tipo de Riesgo	Causas	Prob.	Impacto	Medidas preventivas
Humano	Poca experiencia con tecnologías como Koltin, Axure o testing unitario	Media	Medio	Preparación y estudio previo de estas tecnologías. En caso de bloqueo buscar tecnologías alternativas que aseguren la mayor calidad posible
	Trabajo semanal a tiempo completo, lo que puede dificultar el rendimiento de trabajo por exigencias del trabajo, horas extra, fatiga, etc.	Alta	Alto	Realizar una planificación con cierto margen temporal para poder compensar dichas desviaciones
	Posibles bajas por enfermedad, parones vacacionales, que puedan provocar desviaciones en la planificación	Baja	Medio	Realizar una planificación con cierto margen temporal para poder compensar dichas desviaciones
Técnico	Fallo del WS, el cual no es gestionado por el equipo de este proyecto y podría dificultar las pruebas y la ejecución	Baja	Alto	Realizar la aplicación mediante capas de abstracción para que los módulos sean lo más independientes posible en caso de tener que implementar un nuevo WS
	Problemas con el equipo, pérdida de trabajo en formato digital	Media	Alto	Desarrollar el proyecto mediante el empleo de un control de versiones que se actualizará con cada nueva característica o *bug fixing* y que guardará siempre una versión en la nube

Tabla 2. Evaluación de Riesgos del Proyecto

Teniendo en cuenta los datos anteriores y los plazos de entrega marcados, se pueden planificar las tareas a realizar en orden cronológico, de forma que se disponga de un calendario con el que poder realizar un seguimiento de la evolución del proyecto y realizar las acciones necesarias en caso de posibles desviaciones:

Nombre de la tarea	Fecha de inicio	Fecha final	Duración
1 - Definición y planificación de la aplicación	**20/02/19**	**13/03/19**	**16d**
Definición del objeto de la aplicación	20/02/19	22/02/19	3d
Estudio de mercado entre las aplicaciones existentes	25/02/19	26/02/19	2d
Primera definición de la funcionalidad de la aplicación	27/02/19	28/02/19	2d
Planificación detallada y estimación de costes de los trabajos a realizar	01/03/19	11/03/19	7d
Estudio de riesgos con medidas de contingencia	12/03/19	13/03/19	2d
2 - Diseño Centrado en el Usuario	**14/03/19**	**03/04/19**	**15d**
Investigación de requisitos de UX y contextos de uso	14/03/19	18/03/19	3d
Diseño conceptual	19/03/19	21/03/19	3d
Prototipado	22/03/19	03/04/19	9d
Evaluación	26/03/19	01/04/19	5d
Definición de los casos de uso	26/03/19	28/03/19	3d
Diseño de la arquitectura	29/03/19	01/04/19	2d
3 - Implementación	**04/04/19**	**15/05/19**	**30d**
Codificación de la aplicación	04/04/19	30/04/19	19d
Creación de POJO/BBDD	04/04/19	08/04/19	3d
Integración con API	09/04/19	12/04/19	4d
Codificación de Casos de Uso	13/04/19	24/04/19	9d
Detalles Gráficos	25/04/19	30/04/19	4d
Test de la aplicación	01/05/19	15/05/19	11d
Definición conjunto de pruebas	01/05/19	08/05/19	6d
Elaboración de test unitarios	09/05/19	15/05/19	5d
4 - Entrega Final	**16/05/19**	**04/06/19**	**14d**
Finalización de la memoria y el resto de entregables	16/05/19	04/06/19	14d
Entrega TFM	**20/02/19**	**05/06/19**	**75d**

Tabla 3. Calendario de tareas

Diagrama de Gantt - Aplicación para visualizar datos y estadísticas de los campeonatos de F1 en dispositivos Android

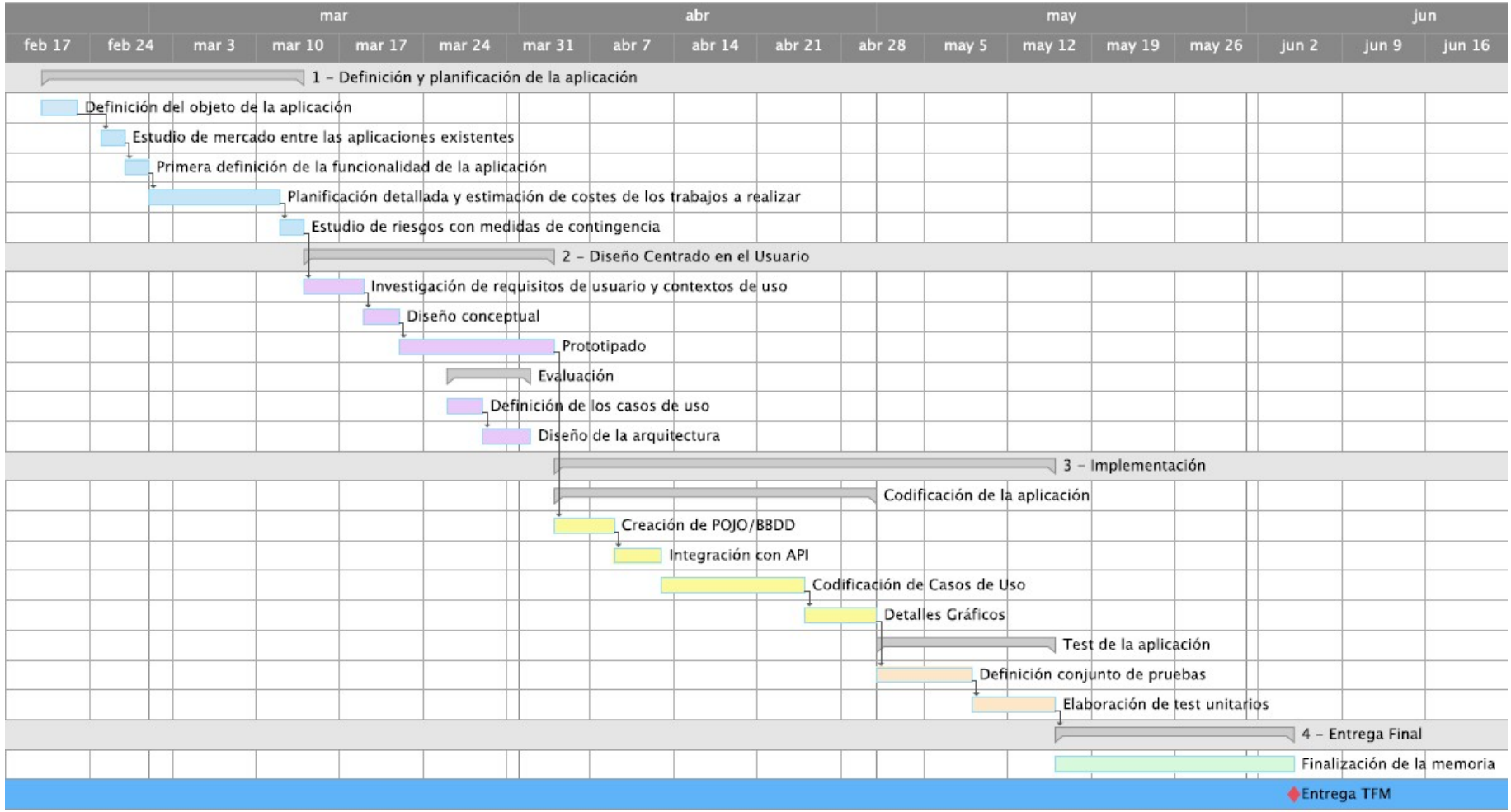


Ilustración 1. Diagrama de Gantt del proyecto

La cantidad de horas disponibles para ejecutar el trabajo, teniendo en cuenta días libres y posibles bajas, responden a la siguiente estimación: 2h/día de lunes a viernes y 4h/día el fin de semana, con una media de 2,6h/día. En un período de 75 días, eso resulta en un total de 200h, pudiendo aumentar la carga de trabajo en caso de posibles desviaciones.

Con todos estos datos podemos realizar una estimación en función de los recursos, las tareas a realizar y el nº de horas necesarias, así como el grado de especialización del equipo de desarrollo:

Recursos Necesarios		**Coste**
Humanos	Diseñador UX Desarrollador Android con conocimientos de Kotlin, testing, y arquitecturas CLEAN y MVP	60h * 15€/h = 900€ 140h * 18€/h = 2.520€
Materiales	Un ordenador portátil Un dispositivo Android para pruebas Papel, lápiz y bolígrafo para elaborar los sketches	1.000€ 200€ 10€
Software	Licencia Android Developer para publicar en Play Store	20 €
	Total	**4.650 €**

Tabla 4. Estimación de recursos necesarios

1.5 Breve sumario de productos obtenidos

Los productos obtenidos tras la finalización de este proyecto serán una memoria, el código de la aplicación, un archivo apk, que contendrá los archivos ejecutables para instalar la aplicación en un dispositivo de la plataforma Android, y un vídeo de presentación con un resumen del trabajo realizado.

En la memoria se definirán: el contexto y en el enfoque empleados, los objetivos al inicio del proyecto, una planificación de los tiempos y fases de ejecución de este, una descripción detallada de los procesos seguidos a lo largo de los diversos capítulos, y una conclusión con los resultados obtenidos y las posibles desviaciones con respecto a la planificación inicial.

El código de la aplicación seguirá un formato concreto, que permita leerlo de forma limpia y legible, y se añadirán los comentarios que sean necesarios para una mejor comprensión del mismo. También incluirá test unitarios, para poder realizar un *testing* automático de las principales funcionalidades de la aplicación, con el objeto de que sea más fácil de ampliar y mantener, ya que

con cada nueva funcionalidad que se añada se podrá comprobar rápidamente si el código se ha visto afectado en alguna otra parte de la aplicación.

El archivo apk será generado a partir de las herramientas Gradle y Android Studio, e incluirá todos los archivos necesarios para la correcta ejecución de la aplicación en un cualquier dispositivo Android con su simple instalación.

1.6 Breve descripción de los otros capítulos de la memoria

El siguiente capítulo se centrará en el Diseño UX de la aplicación. Partiendo de un estudio del contexto y las posibles necesidades de los potenciales usuarios, se elaborarán unas fichas de éstos. Después se hará un detallado estudio de mercado entre las aplicaciones existentes para examinar sus puntos fuertes y débiles, y en función de las conclusiones obtenidas se concretará la funcionalidad de la app. Una vez hecho esto se podrá proceder a realizar los diseños en baja y alta resolución, que servirán como guía para el desarrollo final

En el capítulo de Implementación, se definirá la estrategia a seguir durante la codificación del producto final, explicando los paradigmas de arquitectura empleados, así como las librerías externas incluidas y su objetivo. También se comentarán las estructuras de bases de datos locales y remotas, y se incluirán las partes del código más relevantes para su compresión.

En el capítulo de Testing y Revisión del Diseño se hará un análisis de la ejecución de la app recopilando la información a través de herramientas como Firebase o por la propia experiencia de los usuarios a los que se distribuirá la aplicación.

En el capítulo de Conclusiones se resumirán las conclusiones finales del proyecto, valorando la evolución del desarrollo con respecto a la planificación inicial, las desviaciones en la planificación y las medidas de contingencia empleadas y una valoración del resultado final.

Por último, en los capítulos de Glosario y Bibliografía se incluirán las referencias técnicas empleadas junto con su definición y las fuentes consultadas durante la elaboración de este proyecto.

2. Diseño Centrado en la experiencia de Usuario (UX)

En este capítulo se detallará el proceso llevado a cabo para definir el diseño de la aplicación centrado en la experiencia de usuario (UX) indicado en el Plan de Trabajo, para posteriormente desarrollar la aplicación en base a este diseño. Los objetivos principales de este capítulo serán:

1) Investigar los usuarios de la aplicación y recoger requisitos, tanto cuantitativos como cualitativos, que ayudarán a conocer los usuarios y definir perfiles.
2) Examinar y analizar las condiciones en que se utilizará el sistema para definir su contexto de uso.
3) Estudiar los escenarios de uso.
4) Definir los flujos de interacción en el sistema.
5) Diseñar y construir un prototipo de alto nivel de la aplicación.

2.1. Usuarios y contexto de uso

El objetivo de la primera fase es el de conocer las características de los usuarios, sus necesidades y objetivos, así como el contexto de uso. Para ello, se abarcará el estudio desde dos frentes: primero se hará un breve estudio de mercado entre las mejores aplicaciones similares disponibles en la actualidad, para determinar cuáles son sus puntos fuertes y los puntos débiles que nuestra aplicación puede mejorar, y a continuación se realizarán dos fichas de potenciales usuarios de la aplicación, detallando su identidad, contexto, motivaciones, las posibles necesidades que pretende cubrir la aplicación y cualquier información que pueda ser relevante para entender los comportamientos del usuario.

Benchmarking o estudio de la posible competencia

Para llevar a cabo este estudio, se han seleccionado tres aplicaciones para la plataforma Android disponibles actualmente en Google Play Store[11], elaboradas en torno a los datos el mismo WS. Se ha examinado su diseño, navegabilidad y experiencia de uso, así como las opiniones de los usuarios del *market,* para obtener una visión del contexto de nuestra aplicación dentro el mercado actual.

[11] https://play.google.com/store

Nombre	**GP Companion**	
Desarrollador	indigo-dev	
Valoración usuarios	4,9	
https://play.google.com/store/apps/details?id=com.indigodev.gp_companion		

Esta es posiblemente una de las mejores apps existentes en el mercado. A pesar de no tener un número de descargas demasiado significativo, los que la han probado se muestran muy satisfechos con la experiencia, tiene la valoración más alta en la tienda de aplicaciones y parece un buen modelo a seguir para definir los casos de uso y el diseño UX de nuestra aplicación. Dentro de sus aspectos mejor valorados, se citan los siguientes:

- Su diseño: un diseño agradable e intuitivo que permite disfrutar de la aplicación desde el primer momento de uso
- La navegación entre diferentes pantallas, que permite acceder al detalle de un piloto o una escudería simplemente pulsando sobre alguno de ellos en cualquier pantalla de la aplicación que aparezcan
- La cantidad de información mostrada, ya que se pueden obtener una gran cantidad de datos históricos de los campeonatos

Dentro de los puntos de mejora, encontramos algunas opiniones de usuarios que agradecerían disponer de un comparador de estadísticas históricas:

> ★★★★★ 21 July 2016
>
> **A treat for the F1 geeks** There's all the information you could possibly want here. The only thing that's really missing is what the drivers had for breakfast on race day. Only thing I'd love to see is some side by side comparisons of timings from different years. At the moment its a bit cumbersome to jump through several sc

Este será uno de los potenciales puntos de mejora de la aplicación objeto de este proyecto, ya que se trata de una necesidad identificada por parte de los usuarios. Otro posible punto de mejora será intentar buscar un nombre para la aplicación que se ajuste mejor a los parámetros de búsqueda de los usuarios, ya que a pesar de lo bien valorada que está cuesta encontrarla entre las apps de Play Store.

Nombre	Total Formula 1 stats	
Desarrollador	Fattarsi Gianluca	
Valoración usuarios	4,4	
https://play.google.com/store/apps/details?id=com.gmail.fattazzo.formula1world		

Esta app tiene un contenido similar a GP Companion, aunque su diseño no es tan intuitivo como en ésta. El apartado de estadísticas sólo permite obtener un ránking histórico de todos los pilotos/escuderías ordenado en función de las victorias, pódiums, etc… y tampoco ofrece un comparador de estadísticas como el que se plantea incluir en nuestra app. Uno de los posibles puntos fuertes a seguir son los elementos gráficos empleados para mostrar los datos, que la hacen visualmente atractiva para el usuario:

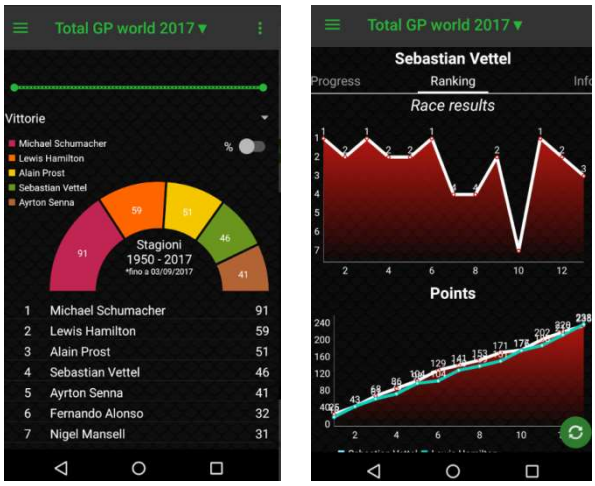


Ilustración 2. Detalles gráficos de otras aplicaciones

Nombre	Formula Live 24 Racing 2019	
Desarrollador	Genera	
Valoración usuarios	4,2	
https://play.google.com/store/apps/details?id=com.gm.racing.main		

Entre sus puntos fuertes se incluyen su galería de imágenes, su apartado de noticias o la posibilidad de iniciar sesión con la cuenta de Twitter del usuario. Entre las deficiencias reseñadas por los usuarios destaca fundamentalmente el abuso del empleo de anuncios (Ads) durante la navegación entre las diferentes pantallas.

El WS seleccionado para elaborar la app no incluye el servicio de urls de imágenes de los pilotos o las escuderías, pero sí que sirve un enlace a Wikipedia que podemos emplear para obtener las imágenes de los pilotos de a través de su propia API[12], tal como lo hacen en algunas de las aplicaciones anteriormente analizadas.

Por motivos de tiempo y recursos disponibles, no entrará dentro del alcance de este proyecto el incluir en la app un número de características tan elevado como las que puede ofrecer algunas de estas aplicaciones, como puede ser incluir un apartado de noticias, o detalles de tiempos de todas las tandas de calificación, que se planificarían para versiones posteriores de la aplicación.

Tras este análisis, se puede definir un listado de requisitos para la aplicación propuesta, destacando los siguientes:

- Diseño atractivo, simple e intuitivo, proporcionando una grata experiencia de usuario en cuanto a navegabilidad. El empleo de colores como el rojo o el negro parece un buen punto de partida, ya que se identifican con los colores del logo oficial del campeonato de F1. Una de las posibles mejoras visuales puede ser emplear la API de Wikipedia para poder obtener imágenes de los pilotos.
- Un nombre que facilite el posicionamiento ASO en el mercado de aplicaciones. El uso de las palabras clave correctas es fundamental para que la aplicación se posicione lo más arriba posible.
- Incluir un comparador de estadísticas como punto de mejora, en el que se ofrezca al usuario la opción de seleccionar entre dos pilotos o escuderías de cualquier época de la historia y se muestre una comparativa entre ambos de los logros más relevantes, como: victorias en GP, podios, número de puntos, vueltas rápidas, etc.
- Mostrar los datos empleando algún tipo de librería que permita crear elementos gráficos a partir de éstos de forma que el diseño sea más atractivo y refleje visualmente las diferencias.

Para nuestra aplicación se ha seleccionado el nombre **Formula Stats**, teniendo en cuenta las restricciones por motivos de derechos de autor[13] del Campeonato Mundial de Fórmula Uno de la FIA, que no permiten posibles combinaciones de nombres que quizá podrían tener un mejor posicionamiento ASO en el mercado.

[12] https://www.mediawiki.org/w/api.php
[13] https://www.formula1.com/en/toolbar/legal-notices.html

2.2 Fichas de usuario

Alberto

"Apasionado de los deportes de motor, en especial de la F1"

Edad: 29
Trabajo: Java developer
Estado Civil: Soltero
Ubicación: A Coruña, España
Carácter: Fan de los deportes

Personalidad

Introvertido	Extrovertido
Frío	Sentimental
Razón	Intuición

Introvertido | Racional | Independiente | Gamer

Objetivos

- Encontrar un trabajo mejor remunerado
- Encontrar una pareja que comparta sus aficiones
- Disfrutar de sus deportes favoritos en su tiempo libre

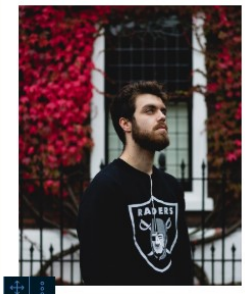
Frustraciones

- Poca afinidad con la empresa en la que trabaja
- Vivir sólo y tener menos vida social de la que le gustaría
- Perderse algún GP de F1 por motivos de trabajo

Bio

Alberto es un joven técnico programador, que desde siempre ha sido un apasionado de los deportes, especialmente de los deportes de motor. Vive sólo y dispone de un Smartphone Android y un ordenador portátil que emplea para visualizar contenidos multimedia, gestionar su correo y seguir la prensa deportiva. Es un apasionado de la F1, al que le gusta jugar a juegos como MotorSport Manager en el que poder gestionar su propia escudería, ya que le gusta trabajar con datos y extraer conclusiones a partir de ellos. A Alberto le gustaría disponer de una app móvil para consultar el estado del campeonato, así como la evolución de sus pilotos favoritos durante el año. Alberto se instala nuestra app, y comprueba que dispone de un calendario de las carreras así como la clasificación del campeonato con un sólo click. También puede acceder al detalle de los pilotos y de las escuderías, con estadísticas que le permiten realizar su propio análisis de la evolución del campeonato,

Brais



"Fan de la música, los cómics y la Fórmula 1"

Edad: 26
Trabajo: Técnico de sonido
Estado Civil: Soltero
Ubicación: Madrid, España
Carácter: Alternativo

Personalidad

Introvertido	Extrovertido
Frío	Sentimental
Razón	Intuición

Sentimental | Extrovertido | Analítico | Curioso

Objetivos

- Poder trabajar en la música
- Lograr reconocimiento
- Poder tener tiempo para disfrutar de sus aficiones

Frustraciones

- Poca conexión con el sistema político
- No poder compaginar todas sus aficiones
- Pocas expectativas de progresar económicamente

Bio

Brais es un joven técnico de sonido que vive en Madrid con dos compañeros de piso. Dentro de sus aficiones se encuentran la música, tocar con su grupo de rock, leer cómics y hacer sus propias grabaciones caseras. Es bastante aficionado a la tecnología, y dispone de un PC y un smartphone de la plataforma Android que emplea para trabajar, ver sus películas y series favoritas, y para navegar por las redes sociales.

Desde joven, es aficionado a la Fórmula 1 porque disfruta con la emoción de los grandes premios y la estrategia necesaria para ganar una carrera. Estas últimas temporadas no está consiguiendo compaginar las carreras con sus otras aficiones, lo que le impide ver algunas de ellas y echa de menos seguir el campeonato como lo solía hacer. Buscando en la tienda encuentra nuestra aplicación, la prueba, y comprueba como dispone de un completo seguimiento que le permite estar más conectado y compensar esa carencia de tiempo.

2.3 Diseño conceptual

En esta segunda fase se elaborarán los escenarios de uso a partir de la información recopilada en la primera fase. Estos serán de gran utilidad para conceptualizar la estructura de la aplicación y los flujos de interacción.

EU1 – Consultar el calendario de GP	
Perfil de usuario	Usuario nuevo
Personaje	Alberto
Contexto	Es jueves y Alberto se encuentra tomando algo al salir del trabajo cuando de repente le viene a la cabeza que este fin de semana es posible que haya carrera, aunque no recuerda en qué circuito ni a qué horario. Como es algo que le sucede a menudo, se pregunta si habrá alguna app que le permita hacer un seguimiento del campeonato de F1. Dentro de las apps existentes, le llama la atención una app que además de incluir estos datos también tiene estadísticas de pilotos y escuderías e incluso un comparador para comparar protagonistas de diferentes épocas de la F1. Tras descargarla y consultar la aplicación, Alberto ya sabe que tiene que reservar la mañana del domingo si quiere ver la próxima carrera en directo
Objetivo	Consultar los datos del próximo GP
Funcionalidades	Mostrar listado de GP Mostrar detalles del GP, incluyendo el horario

Tabla 5. EU1 - Consultar el calendario de GP

EU2 – Consultar la clasificación de pilotos o escuderías	
Perfil de usuario	Usuario habitual
Personaje	Brais
Contexto	Se acerca el final del campeonato de F1 y la clasificación está bastante ajustada, con dos equipos con opciones de ganar. A Brais le gustaría consultar la clasificación de pilotos y escuderías para hacer sus propias cábalas sobre qué resultados tendrían que obtener para poder optar a ganar el campeonato
Objetivo	Consultar la clasificación de pilotos y escuderías
Funcionalidades	Mostrar clasificación del campeonato de pilotos Mostrar clasificación del campeonato de escuderías

Tabla 6. EU2 - Consultar la clasificación de pilotos o escuderías

EU3 – Consultar las estadísticas de pilotos o escuderías	
Perfil de usuario	Usuario habitual
Personaje	Brais
Contexto	A Brais le gustaría analizar la evolución de los resultados y sobretodos comprobar la progresión de algunos de sus pilotos y escuderías favoritas, especialmente de los más noveles, ya que son los futuros campeones de la F1. Para ello dispone de una app que le muestra gráficamente una evolución de las principales estadísticas y de esta forma puede estudiar quién mejora sus posiciones de salida, cuál es más regular o qué circuitos del campeonato se le dan mejor a quién.
Objetivo	Consultar datos de estadísticas de pilotos y escuderías
Funcionalidades	Mostrar estadísticas de los pilotos y escuderías al hacer click sobre alguno de ellos

Tabla 7. EU3 – Consultar las estadísticas de pilotos o escuderías

EU4 – Consultar el comparador de estadísticas	
Perfil de usuario	Usuario habitual
Personaje	Brais
Contexto	Brais es un forofo de la F1 al que le gusta seguir las noticias del campeonato y participar en los foros de discusión. Uno de los otros foreros ha respondido a uno de sus comentarios argumentando en contra de uno de sus pilotos favoritos utilizando datos que él cree que son erróneos, ya que el otro forero defiende que el piloto que le gusta a él tuvo mejores números. Brais sabe que la app que usa para seguir el campeonato habitualmente incluye un comparador de estadísticas entre pilotos/escuderías, así que selecciona su piloto favorito y el del otro forero y comprueba que efectivamente las estadísticas del piloto de Brais muestran que obtuvo resultados mejores
Objetivo	Comparar las estadísticas entre pilotos y escuderías de cualquier época
Funcionalidades	Incluir un comparador de estadísticas entre pilotos y escuderías, proporcionando la opción de seleccionar una pareja cualquiera de ellos para que la aplicación muestre automáticamente una comparativa de los datos más relevantes de ambos

Tabla 8. EU4 – Consultar el comparador de estadísticas

2.3 Prototipado

Tomando los flujos de interacción definidos en la segunda fase, se ha realizado un prototipo horizontal de la aplicación. La versatilidad del prototipo hace que sea sencillo y económico introducir modificaciones en el diseño e iterar incorporando mejoras fruto de los resultados obtenidos en la evaluación.

Primero se han realizado unos bocetos a mano definiendo la ubicación del menú de navegación, el contenido de las pantallas, y los principales datos a mostrar:

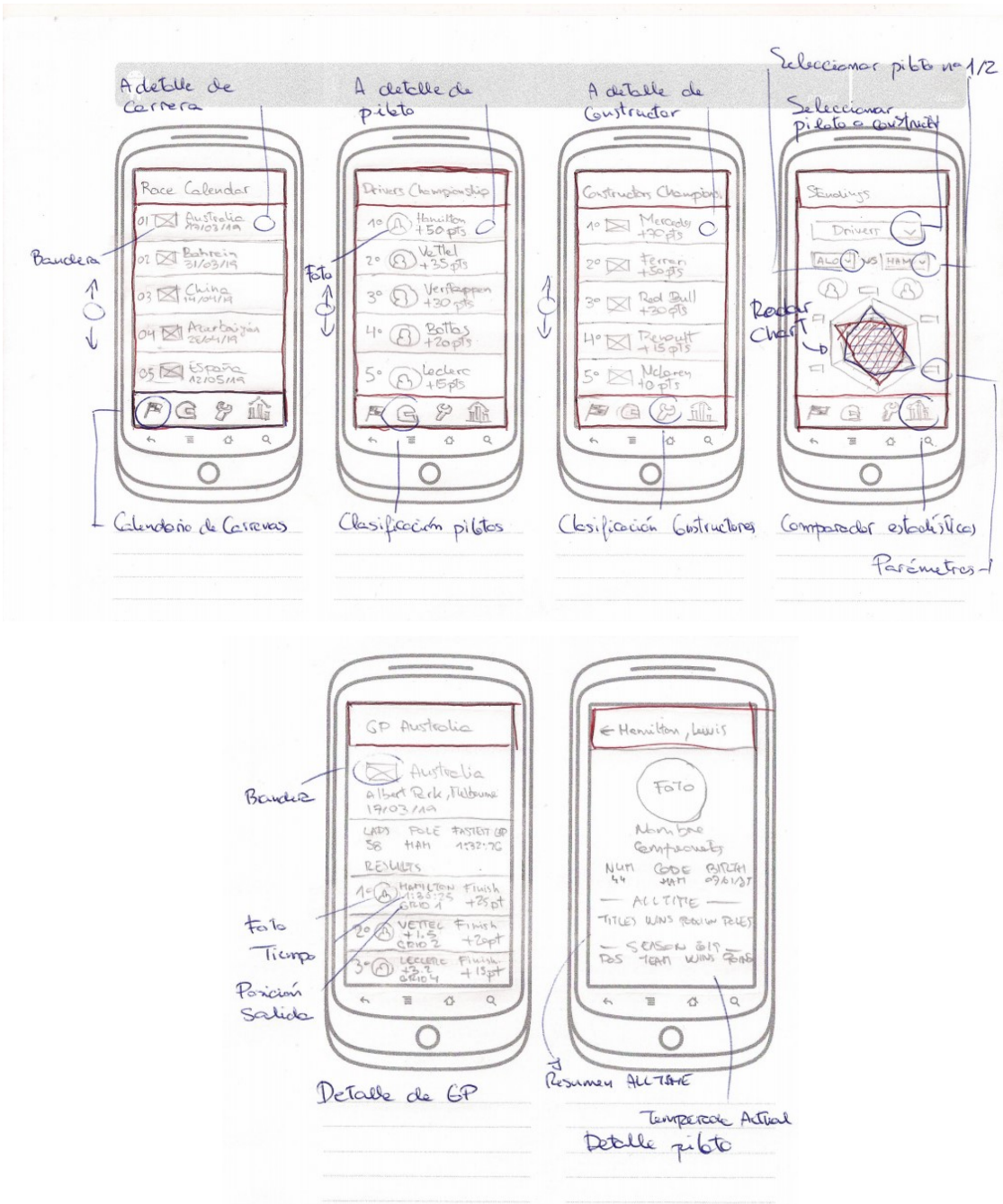


Ilustración 3. Bocetos de la aplicación realizados a mano

Empleando dichos bocetos como base, se ha realizado un prototipo en alta definición empleando el software Axure RP[14], que se muestran a continuación.

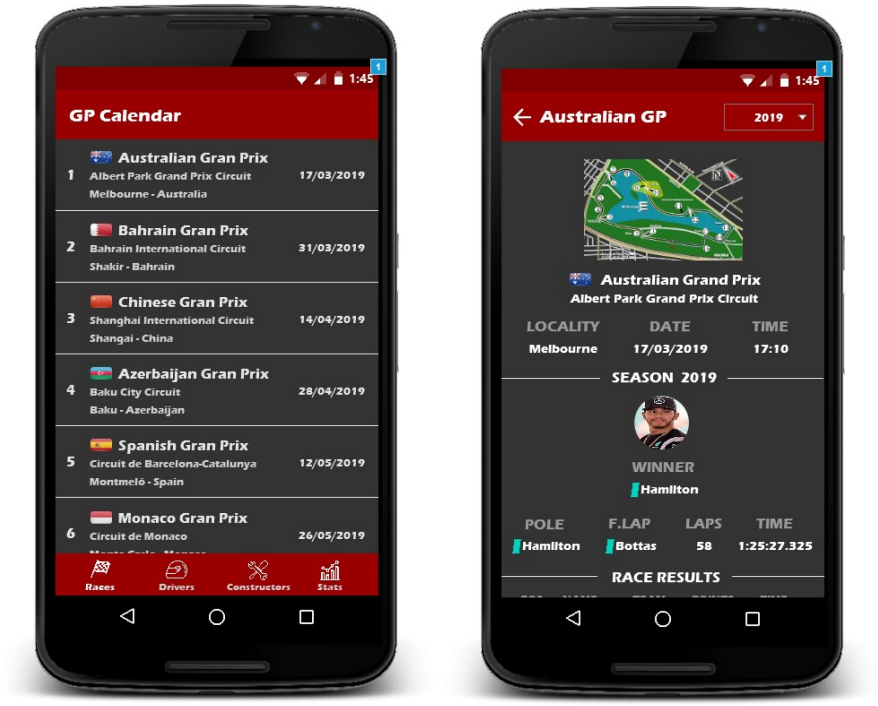


Ilustración 4. Diseño de Calendario de GP y Detalle de carrera (parte superior)

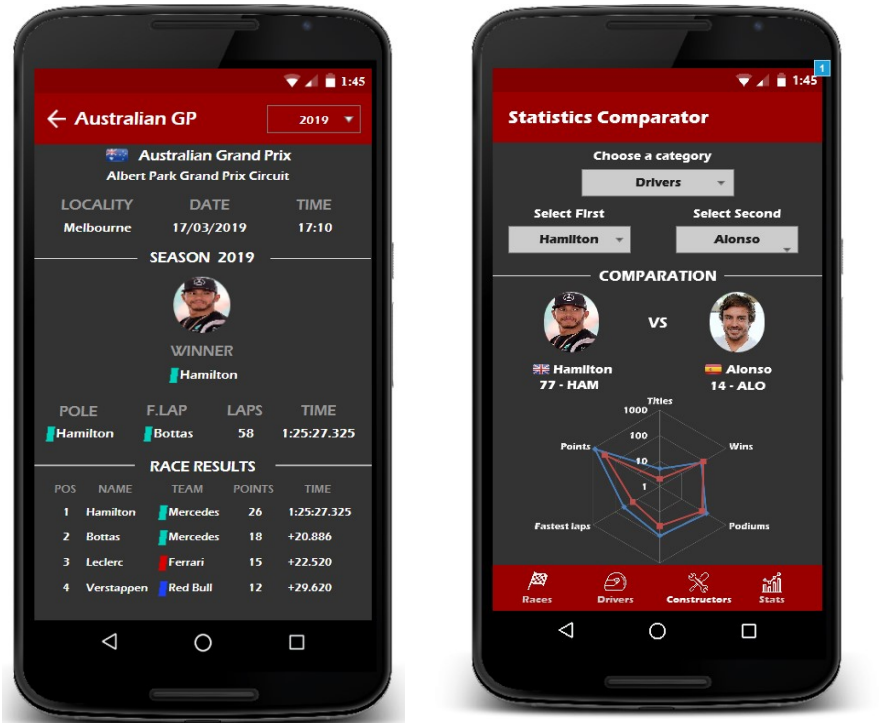


Ilustración 5. Diseño de Detalle de Carrera y Comparador de Estadísticas

[14] https://www.axure.com/

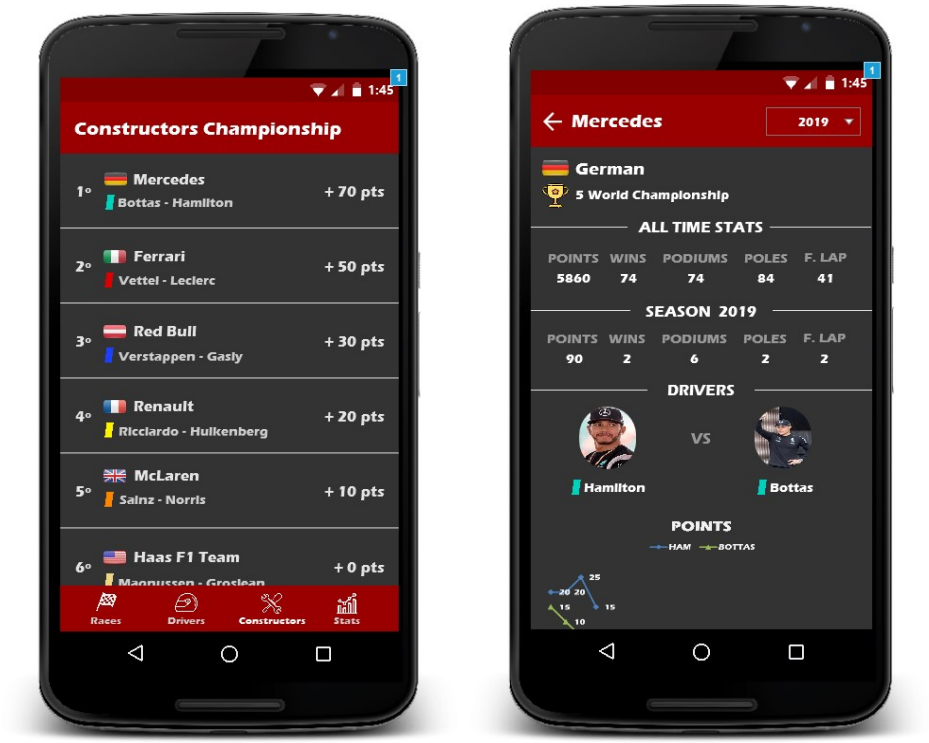


Ilustración 6. Diseño de Clasificación de Constructores y Detalle de Constructor

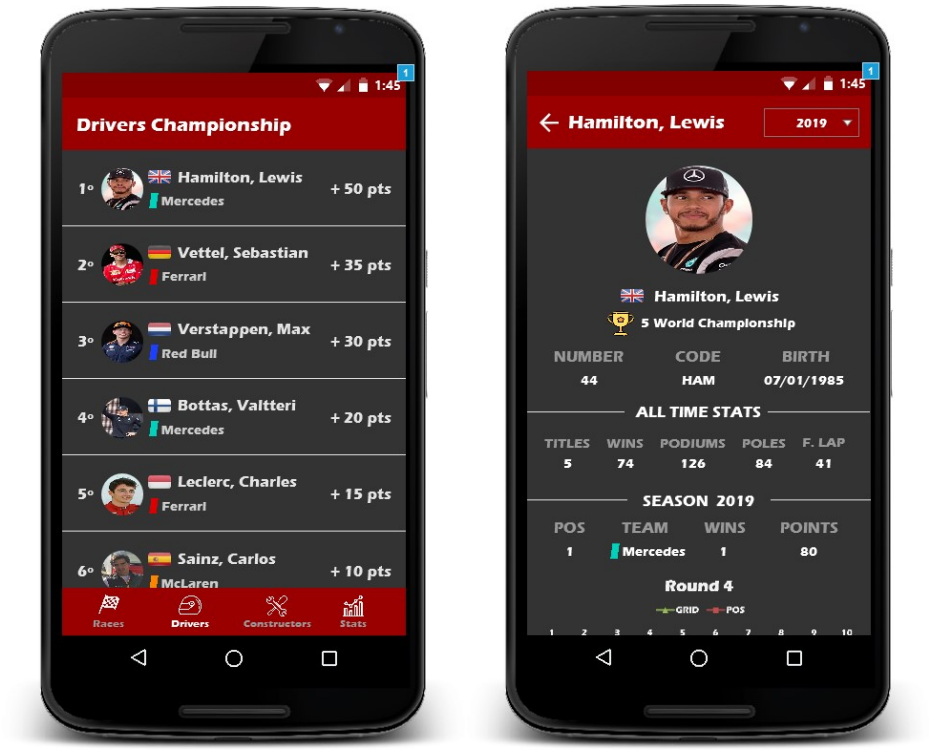


Ilustración 7. Diseño de Clasificación de Pilotos y Detalle de Piloto

Los iconos han sido obtenidos de FlatIcon[15] bajo licencia "Creative Commons BY 3.0" de los autores photo3idea_studio, DinosoftLabs, Retinaicons y Freepik

[15] https://www.flaticon.com/

2.4. Evaluación

El objetivo de la última fase es planificar la evaluación del prototipo. El proceso de UX es un proceso iterativo y, por tanto, hay que ir evaluando los diseños y corrigiendo los errores de manera iterativa.

A. Definición de los casos de uso

El objetivo de este apartado, una vez acabada la fase de UX, es el de definir formalmente los casos de uso. Estos casos de uso servirán para establecer las funcionalidades de la aplicación en función de los escenarios de uso y el prototipo definidos durante el UX.

Para ello, se han elaborado varios diagramas que ayudarán a definir los flujos de acción de la aplicación y las entidades de datos a manejar y servirán como guía a la hora de implementar el código de la aplicación y definir la mejor arquitectura a emplear para ejecutar nuestro diseño. A continuación se muestra el flujo de acción desde la pantalla de inicio de la aplicación, que será el calendario de carreras, por ser una de las más relevantes.

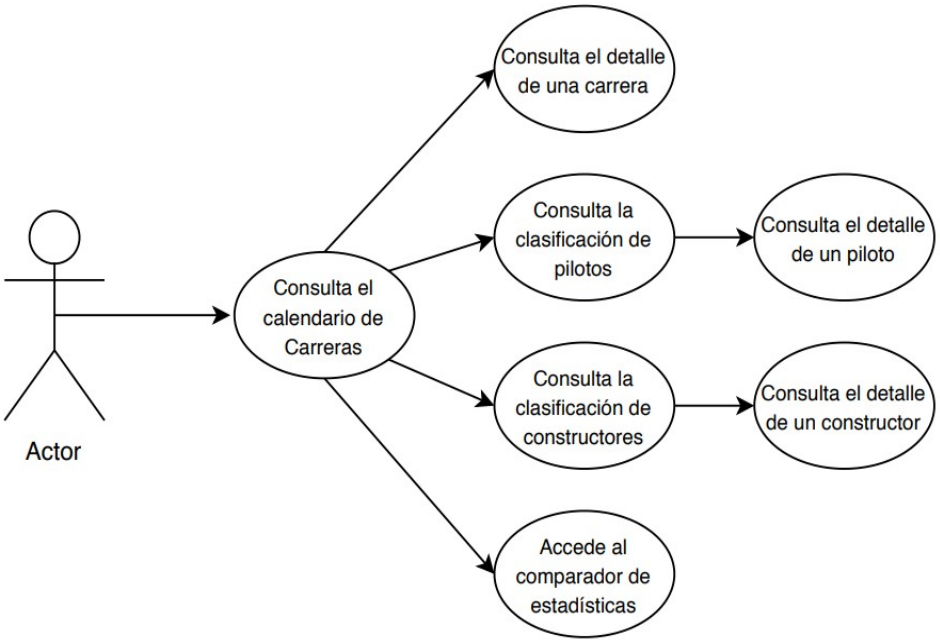


Ilustración 8. Diagrama de flujo de la aplicación

Actores	Caso de Uso	Precondiciones	Flujo	Postcondiciones
Usuario	**CU-1 Consultar el calendario de carreras**	Tener la aplicación apagada, o acceder a la pantalla principal si se está en otra pantalla sin acceso al menú principal	Activar la aplicación, ya que será la pantalla por defecto, o pulsar el icono de carreras de la barra de navegación	El usuario visualiza un listado con el calendario de carreras
	CU-2 Consultar el detalle de una carrera	Seguir alguno de los pasos para consultar el calendario de carreras	Pulsar sobre una de las carreras del listado	El usuario visualiza el detalle de la carrera seleccionada
	CU-3 Consultar la clasificación de pilotos	Activar la aplicación, o acceder a la pantalla principal si se está en otra pantalla sin acceso al menú principal	Pulsar sobre el icono de pilotos de la barra de navegación	El usuario visualiza un listado ordenado con la clasificación de pilotos
	CU-4 Consultar el detalle de un piloto	Seguir alguno de los pasos para consultar la clasificación de pilotos	Pulsar sobre uno de los pilotos del listado	El usuario visualiza el detalle del piloto seleccionado
	CU-5 Consultar la clasificación de constructores	Activar la aplicación, o acceder a la pantalla principal si se está en otra pantalla sin acceso al menú principal	Pulsar sobre el icono de constructores de la barra de navegación	El usuario visualiza un listado ordenado con la clasificación de constructores
	CU-6 Consultar el detalle de un constructor	Seguir alguno de los pasos para consultar la clasificación de constructores	Pulsar sobre uno de los constructores del listado	El usuario visualiza el detalle del constructor seleccionado
	CU-7 Consultar el comparador de estadísticas	Activar la aplicación, o acceder a la pantalla principal si se está en otra pantalla sin acceso al menú principal	Pulsar sobre el icono de estadísticas de la barra de navegación. Seleccionar si se prefiere ver la comparativa entre dos constructores o dos pilotos. Seleccionar los dos constructores o pilotos	El usuario visualiza una comparativa entre los dos constructores o pilotos

Tabla 9. Listado de Casos de Uso de la aplicación

B. Diseño de la arquitectura

El objetivo de este apartado es el de definir la arquitectura del sistema, identificando las entidades que se representarán en la base de datos (si la hay), las clases y objetos que se utilizarán para gestionar los diferentes procesos y la estructura de la API que servirá para realizar las peticiones enviados al servidor desde un cliente.

Diagrama UML correspondiente al diseño de la Base de Datos (BBDD)

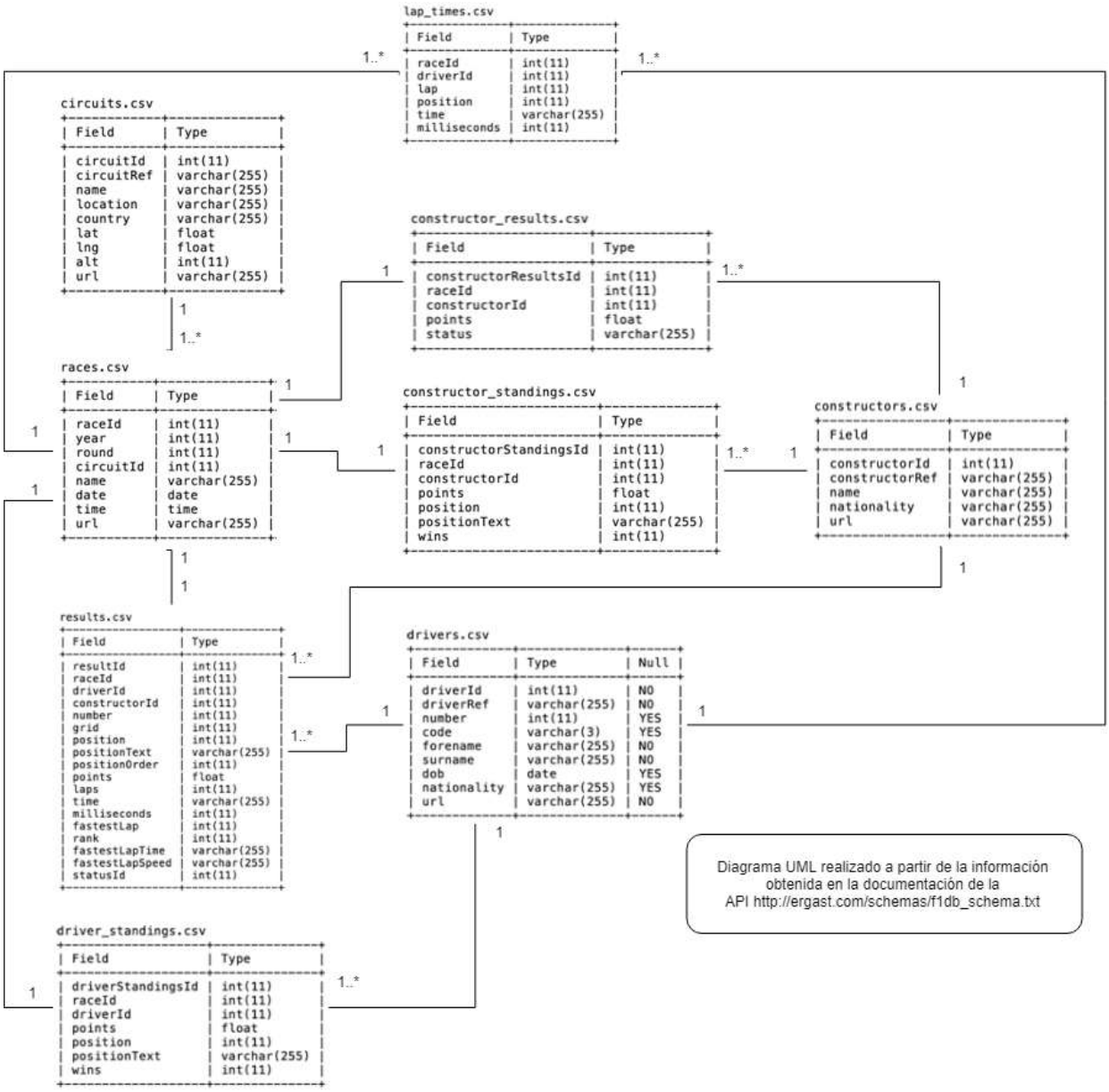


Ilustración 9. Diagrama UML de la Base de Datos de la aplicación

Arquitectura del sistema

Las arquitecturas permiten desacoplar diferentes unidades del código de una manera organizada. De esa forma el código se vuelve más fácil de entender, modificar y probar. Para el desarrollo de este proyecto se emplearán los fundamentos de arquitectura **CLEAN**, que resumiremos a continuación:

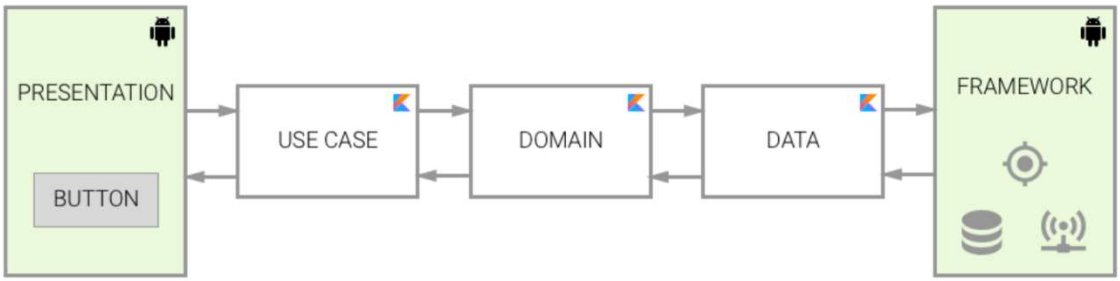


Ilustración 10. Diagrama de la arquitectura CLEAN

a) Presentation

Es la capa que interactúa con la interfaz de usuario. Esta capa de presentación generalmente consiste en la interfaz de usuario de Android (activities y fragments) y presenters o modelos de vista, según el patrón de presentación que se decida usar. En nuestro caso se empleará el patrón **MVP**, que permite separar la capa de presentación de la lógica para que el funcionamiento de la interfaz de usuario sea independiente de cómo lo representamos en la pantalla. Idealmente, el patrón MVP lograría que la misma lógica tenga vistas completamente diferentes e intercambiables.

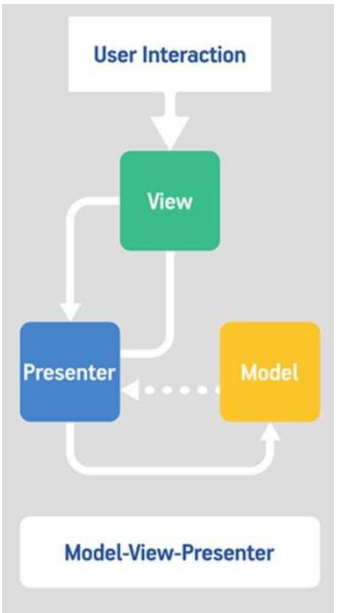


Ilustración 11. Diagrama del patrón MVP

b) Use Case

Por lo general, también se llama interactors. Estas son principalmente las acciones que el usuario puede activar. Esas pueden ser acciones activas (el usuario hace clic en un botón) o acciones implícitas (la aplicación navega a una pantalla).

c) Domain

Contiene todos los modelos de negocio. Idealmente, debería ser la capa más grande, aunque en general las aplicaciones de Android tienden a dibujar una API en la pantalla de un teléfono como es nuestro caso, por lo que la mayor parte de la lógica central consistirá en solicitar y conservar datos.

d) Data

En esta capa, tiene una definición abstracta de los diferentes orígenes de datos, y cómo deben usarse. En este caso, se utilizará un patrón de repositorio que, para una solicitud determinada, puede decidir dónde encontrar la información.

e) Framework

Encapsula la interacción con el framework, de modo que el resto del código puede ser agnóstico y reutilizable en caso de que desee implementar la misma aplicación en otra plataforma (una opción real hoy en día con los proyectos multiplataforma Kotlin). Esta capa debe ser lo más simple posible, ya que toda la lógica debe abstraerse en la capa de datos.

C. Evaluación del diseño mediante el empleo de Google Analytics

Una vez definidos los perfiles de usuarios, el prototipo y los casos de uso, se está en disposición de realizar una primera versión de la aplicación. Al tratarse de un proceso iterativo y continuo, es necesario obtener algún tipo de feedback de los usuarios para poder seguir mejorando el diseño en versiones posteriores. Para ello, se ha estudiado la opción de introducir Google Analytics[16] en la app, lo que permitirá recoger eventos enviados por los usuarios desde la propia app que se podrán resumir en un informe que refleja las pantallas más visitadas, o las funcionalidades más empleadas, para en función de estos datos ir progresivamente refinando el diseño de la aplicación.

Esta aplicación funciona con Firebase, una plataforma integrada para desarrolladores de aplicaciones de Google. La implementación de este servicio se detallará en el siguiente capítulo de este proyecto.

[16] https://analytics.google.com/analytics/web/#/a137357769w0p0/admin

3. Implementación

3.1. Herramientas empleadas

A. Entorno de desarrollo

Para el desarrollo de la aplicación se ha empleado Android Studio 3.4 como entorno de desarrollo, ejecutado en un PC con Windows 10.

B. Lenguaje de programación

La aplicación se ha desarrollado íntegramente en Kotlin, un lenguaje fuertemente tipado que opera enteramente con Java y que aporta una mayor seguridad frente a sus problemáticas Null Poiter Exceptions. También simplifica el desarrollo gracias a su inferencia de tipos y tiene características interesantes a la hora de trabajar con colecciones de objetos complejas.

C. Control de Versiones

Se ha empleado un repositorio en BitBucket para llevar un control de cambios del proyecto, subiendo los cambios con cada evolución de la aplicación, y llevar un seguimiento del mismo. Para gestionar la comunicación con el repositorio se ha empleado SourceTree como interfaz GIT.

3.2. Características de la aplicación

- compileSdkVersion 28: Se elige esta versión para compilar la app contra la más versión de Android reciente durante el momento de desarrollo
- minSdkVersion 21: A partir de esta versión se cambió la máquina virtual Dalvik del sistema Android por ART (Android Runtime), que es más potente y mejora los tiempos de compilación[17].
- targetSdkVersion: 28 Se elige esta versión para compilar la app contra la más versión de Android reciente durante el momento de desarrollo

3.3. Librerías empleadas

Room[18]

Esta útil librería del Jetpack de Android aporta una capa de abstración sobre SQLite y se emplea para desarrollar y acceder fácilmente a una base de datos local en el dispositivo.

[17] http://blog.safedk.com/technology/solving-androids-65k-limit-part-2-the-lollipop-generation/
[18] https://developer.android.com/topic/libraries/architecture/room

Koin[19]

Esta librería se integra con Kotlin permitiendo implementar el patrón de inyección de dependencias y de esta forma desacoplar las clases unas de otras.

Picasso[20]

Esta librería se emplea para cargar imágenes web en la app a través de la url de la imagen.

CircleImageView[21]

Esta librería facilita añadir imágenes con un marco circular a la aplicación.

ExpandableHeightListView[22]

Con esta librería se añade una lista de un tamaño fijo dentro de un ScrollView evitando problemas de compatibilidad.

MPAndroidChart[23]

Esta librería se emplea para añadir gráficas a la aplicación.

AssertJ[24]

Para facilitar las aserciones de los test unitarios de la aplicación.

Barista[25]

Para facilitar las aserciones de los test instrumentales de la aplicación.

3.4. Bases de datos

Los datos que se muestran en la aplicación se obtienen de las siguientes bases de datos:

Remotas

- Estadísticas de F1: https://ergast.com/mrd/
- Imágenes de banderas: https://countryflags.io/
- Códigos de país: https://restcountries.eu/
- Imágenes de Wikipedia: https://www.mediawiki.org/wiki/API:Main_page

[19] https://github.com/InsertKoinIO/koin
[20] https://square.github.io/picasso/
[21] https://github.com/hdodenhof/CircleImageView
[22] https://github.com/paolorotolo/ExpandableHeightListView
[23] https://github.com/PhilJay/MPAndroidChart
[24] https://joel-costigliola.github.io/assertj/
[25] https://github.com/SchibstedSpain/Barista

Locales

- CountriesLocal: Para obtener las direcciones url de las imágenes de las banderas primero fue necesario obtener los códigos alpha2 de dicho país, pero la api que sirve los datos de F1 sólo devuelve la nacionalidad del piloto o el país del constructor o la carrera, por lo que se optó por descargar una base de datos con todos los códigos de país, junto con su nacionalidad y el nombre de su país, durante la primera instalación, y de esta forma poder buscar el código en función del dato que nos devuelva el servidor de F1. Posteriormente se carga la url de la bandera con Picasso.

- Caché: Se emplea la caché del dispositivo para guardar algunas de las consultas más habituales y de esta forma hacer menos llamadas a red.

3.5. Desarrollo de la arquitectura

La estructura de la aplicación se divide en tres carpetas principales (data, domain y presentation) que representarán las capas de la arquitectura CLEAN definida en el capítulo de Diseño, y cada una se encargará de un cometido diferente.

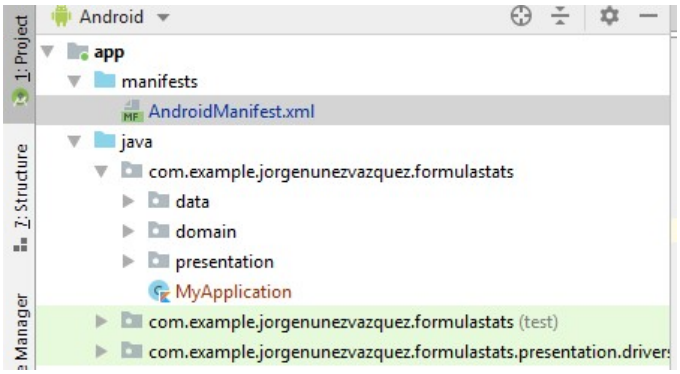


Ilustración 12. Estructura de la aplicación

A. Capa Data

Esta capa se encargará de gestionar todo lo relacionado con los datos de la aplicación. Sus principales funciones serán:

- La gestión de llamadas a red para obtener datos de una fuente de datos remota.
- La creación y el acceso a una base de datos local en el dispositivo
- La gestión de la caché del dispositivo para almacenar archivos temporales que puedan ser desechados en caso de falta de memoria.

Esta capa define la implementación de las diferentes fuentes de datos de la aplicación, que son la red (remote), una base de datos local y la caché del dispositivo. También se ha definido un repositorio por cada elemento principal de la aplicación que se dividen en cinco:

1. Countries: Gestiona una base de datos que nos permitirá obtener el código alpha de un país bien sea por el nombre de dicho país o por su nacionalidad, que se emplearán para cargar la imagen de la bandera de los pilotos o de los constructores.
2. Drivers: Gestiona la base de datos que permite obtener las estadísticas, los resultados, y las imágenes relacionadas con los pilotos.
3. Constructors: Gestiona la base de datos que permite obtener las estadísticas, los resultados, y las imágenes relacionadas con los constructores.
4. Races: Gestiona la base de datos que permite obtener el calendario de carreras y los resultados de las mismas una vez concluidas.
5. Stats: Gestiona la base de datos que permite obtener el listado histórico de pilotos y constructores.

B. Capa Domain

Esta capa se encargará de definir los contratos de la aplicación, que son las interfaces que definirán las operaciones a ejecutar y serán implementadas en el resto de capas, sirviendo de enlace entre la vista y el modelo. En ella también se implementarán los casos de uso de la app, que serán las clases encargadas de solicitar la información a la capa Data por petición del presenter de la capa Presentation y darle una respuesta al mismo entregando los datos si la operación ha tenido éxito o devolviendo un error en caso contrario.

Estos casos de uso se ejecutarán siempre en segundo plano para liberar el hilo de ejecución principal de la app. Para ello implementarán la interfaz Runnable, que permite ejecutar operaciones en hilos paralelos, y extenderán la clase BaseUseCase, que será genérica y definirá los métodos a ejecutar y un Callback para devolver los datos al hilo principal una vez se complete la operación:

```
abstract class BaseUseCase<P, R> : Runnable {

    protected var params: P? = null
    lateinit var invoker: Invoker
    var callback: ((Result<R>) -> Unit)? = null

    fun execute(params: P? = null, invoker: Invoker, callback: ((Result<R>) -> Unit)? = null) {
        this.params = params
        this.callback = callback
        this.invoker = invoker
        invoker.execute( baseUseCase: this)
    }

    protected fun notify(result: Result<R>) {
        invoker.notify(callback, result)
    }
}
```

Ilustración 13. Clase BaseUseCase

Para ejecutar los casos de uso se empleará un ThreadPoolExecutor[26] como se recomienda en la guía de desarrolladores de Android, que permitirá gestionar y ejecutar varios hilos a la vez, lanzando una nueva operación desde la cola siempre que haya un hilo disponible. Este mánager de operaciones en segundo plano se encuentra implementado en la clase UseCaseInvoker, y es inyectado en todos los presenters de la aplicación para que puedan ejecutar los casos de uso con él.

```
class UseCaseInvoker : Invoker {

    private val handler: Handler by lazy { Handler(Looper.getMainLooper()) }
    private val threadPoolExecutor: ThreadPoolExecutor
    private val TIME_UNIT = TimeUnit.SECONDS
    private val WORK_QUEUE = LinkedBlockingQueue<Runnable>()

    init {
        threadPoolExecutor = ThreadPoolExecutor(
            CORE_POOL_SIZE,
            MAX_POOL_SIZE,
            KEEP_ALIVE_TIME.toLong(),
            TIME_UNIT,
            WORK_QUEUE
        )
    }
```

Ilustración 14. Clase UseCaseInvoker

B. Capa Presentation

Esta capa se encargará de implementar las vistas de la aplicación (activities y fragments), y los presenters que se encargarán de gestionar la navegación entre vistas y la ejecución de los casos de uso en respuesta a los eventos enviados por las vistas. Una vez recibida la respuesta de la capa Data, el presenter devuelve los datos a la vista para que los reproduzca en pantalla. Gracias a la inyección de dependencias, es posible lograr el desacoplamiento deseado entre capas para depender lo menos

[26] https://developer.android.com/training/multiple-threads/create-threadpool

posible del framework de Android, y así poder desarrollar más test unitarios que corran sobre la máquina virtual de Java en vez de depender de un dispositivo físico o el emulador Otra de las ventajas es la modularización del código y la posibilidad de reutilización del mismo. Las siguientes capturas han sido tomadas con un dispositivo Android modelo Xiaomi MiA1:

- Main Activity: Es la activity principal y se encarga de gestionar los eventos y la navegación de los fragments de un BottomNavigationView con cuatro categorías:
    - Races: Muestra un listado con el calendario de carreras del campeonato actual, ordenado por fecha, indicando su país y el circuito. Haciendo click sobre alguna de las carreras se accede a su pantalla de detalle.

    - Drivers: Muestra un listado con la clasificación de pilotos del campeonato actual, ordenado por posición en el campeonato, indicando su país y su equipo. Haciendo click sobre alguno de los pilotos se accede a su pantalla de detalle.

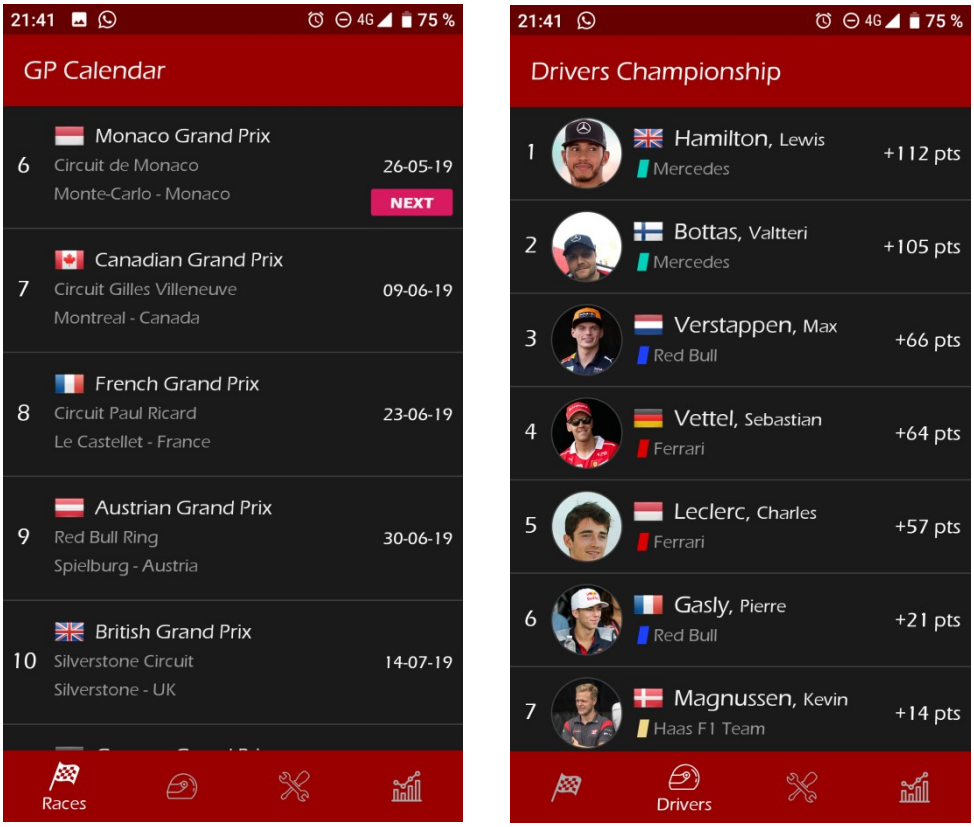


Ilustración 15. Capturas de Calendario de Carreras y Clasificación de Pilotos

- Constructors: Muestra un listado con la clasificación de constructores del campeonato actual, ordenado por posición en el campeonato, indicando su país y sus pilotos. Haciendo click sobre alguno de los constructores se accede a su pantalla de detalle.

- Stats: Muestra unos spinners que permiten seleccionar dos pilotos o dos conductores de los que se muestra un gráfico de radar comparando algunos de sus números más relevantes, como: número de victorias, campeonatos, poles, etc...

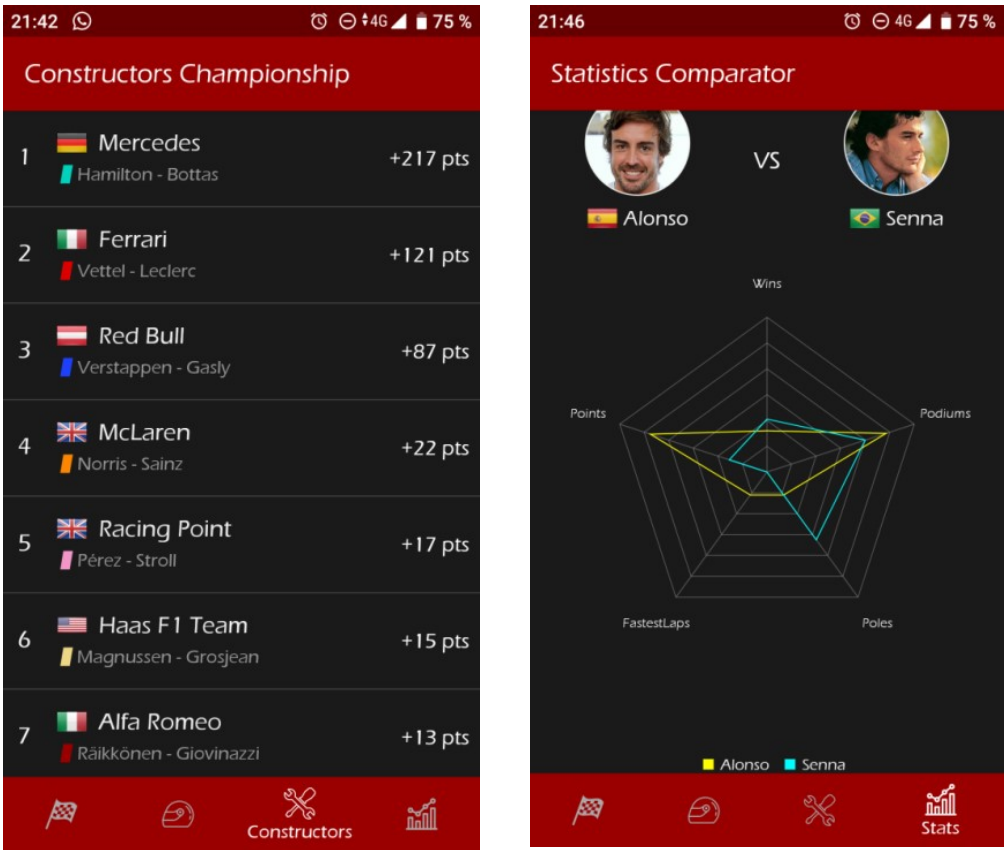


Ilustración 16. Capturas de Clasificación de Constructores y Comparador de Estadísticas

- RaceDetailActivity: Muestra una pantalla de detalle de la carrera con el circuito, la fecha y horario local y el resultado en caso que la carrera ya haya concluido[27].

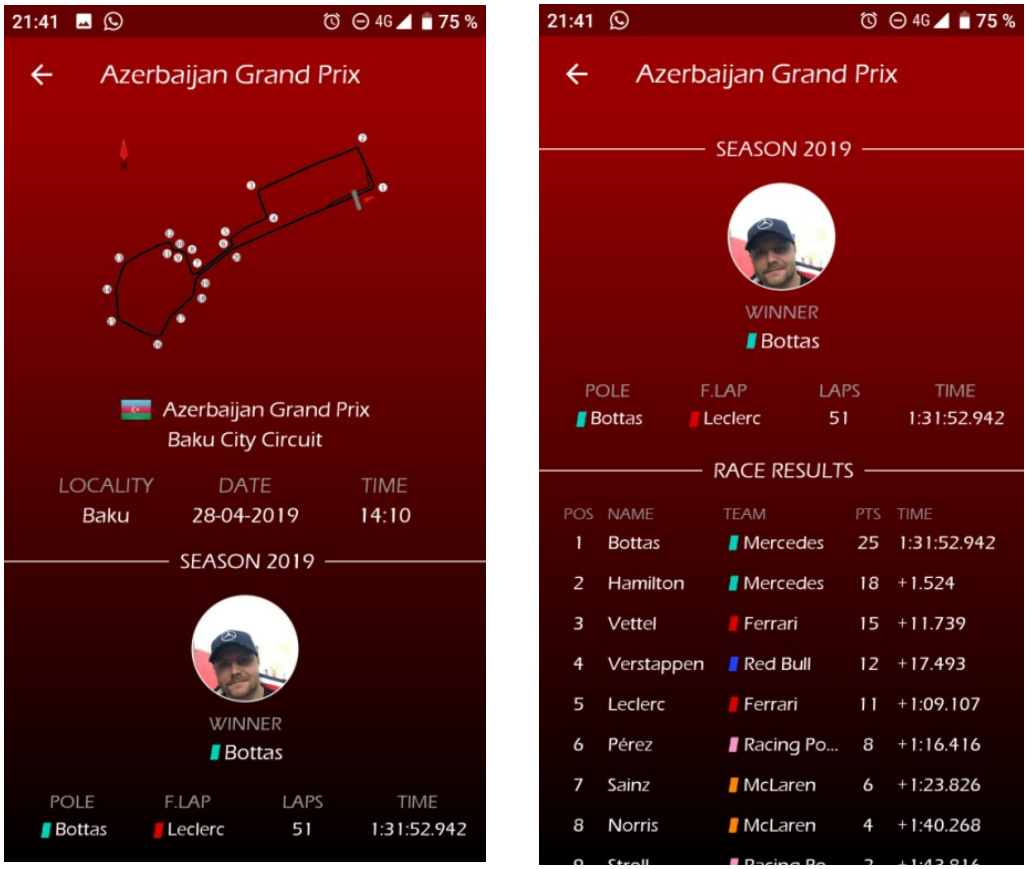


Ilustración 17. Capturas de Detalle de Carrera

- DriverDetailActivity: Muestra una pantalla de detalle del piloto con su imagen, un apartado con estadísticas históricas, el detalle de su posición en el campeonato actual y dos gráficas que muestran su evolución: una comparando sus posiciones de salida con el resultado de la carrera, y otra mostrando el progreso de los puntos ganados.

  En la parte superior derecha, en la Toolbar, se carga un spinner que al pulsarlo muestra un listado de todas las temporadas del piloto. Al seleccionar alguna de ellas, la app se actualiza mostrando el detalle de su posición final en dicho campeonato, así como las gráficas con el progreso del piloto durante toda la temporada.

[27] Nota: la api empleada para obtener los datos en red se actualiza sobre tres horas después de la carrera

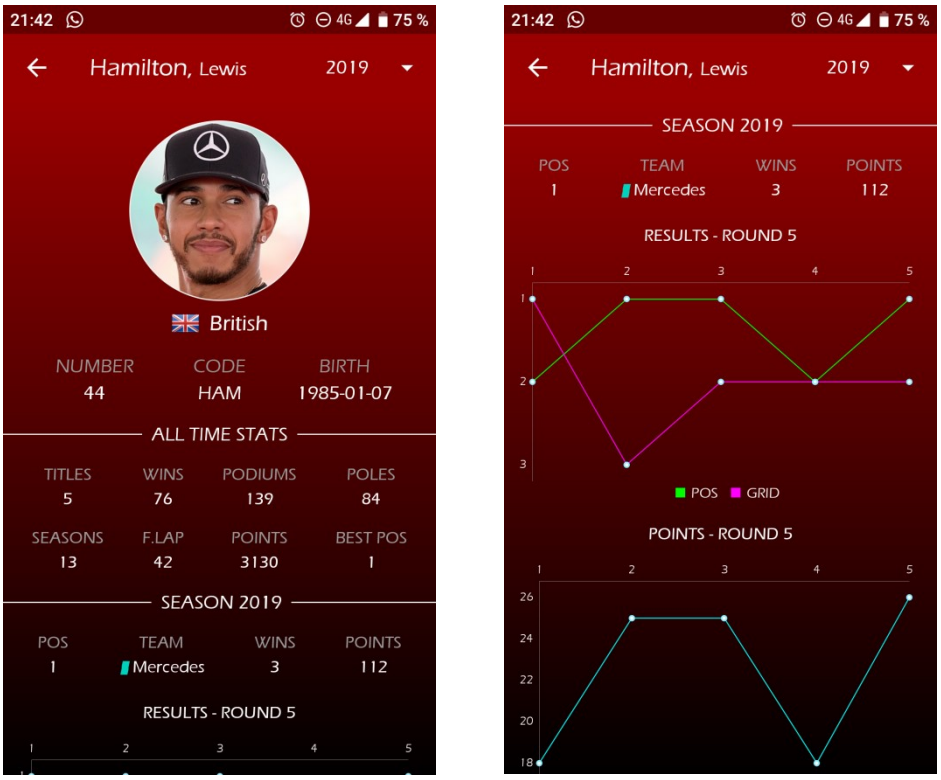


Ilustración 18. Capturas de Detalle de Piloto en la temporada 2019

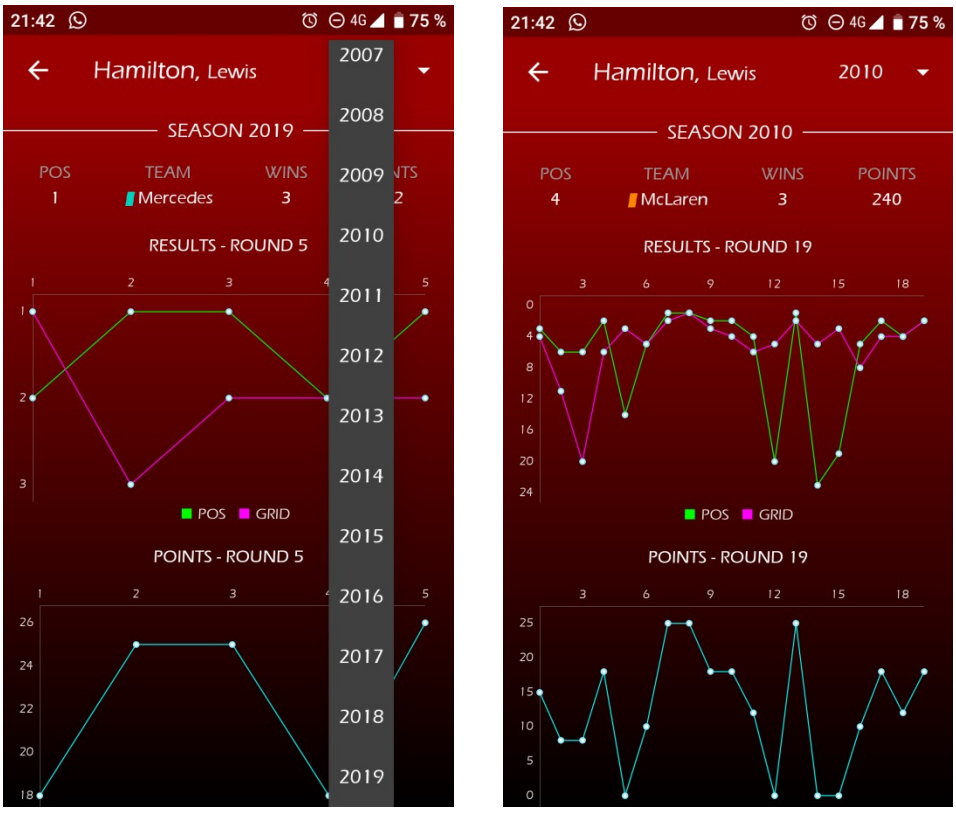


Ilustración 19. Capturas de Detalle de Piloto en la temporada 2010

- ConstructorDetailActivity: Muestra una pantalla de detalle del constructor con su nacionalidad, un apartado con estadísticas históricas, el detalle de su posición en el campeonato actual y dos gráficas que muestran una comparativa de la evolución de sus pilotos: una comparando sus resultados de carrera, y otra del progreso de los puntos ganados por ambos.

  En la parte superior derecha, en la Toolbar, se carga un spinner que al pulsarlo muestra un listado de todas las temporadas del constructor. Al seleccionar alguna de ellas, la app se actualiza mostrando el detalle de su posición final en dicho campeonato, así como las gráficas con la comparativa de sus pilotos durante toda la temporada.

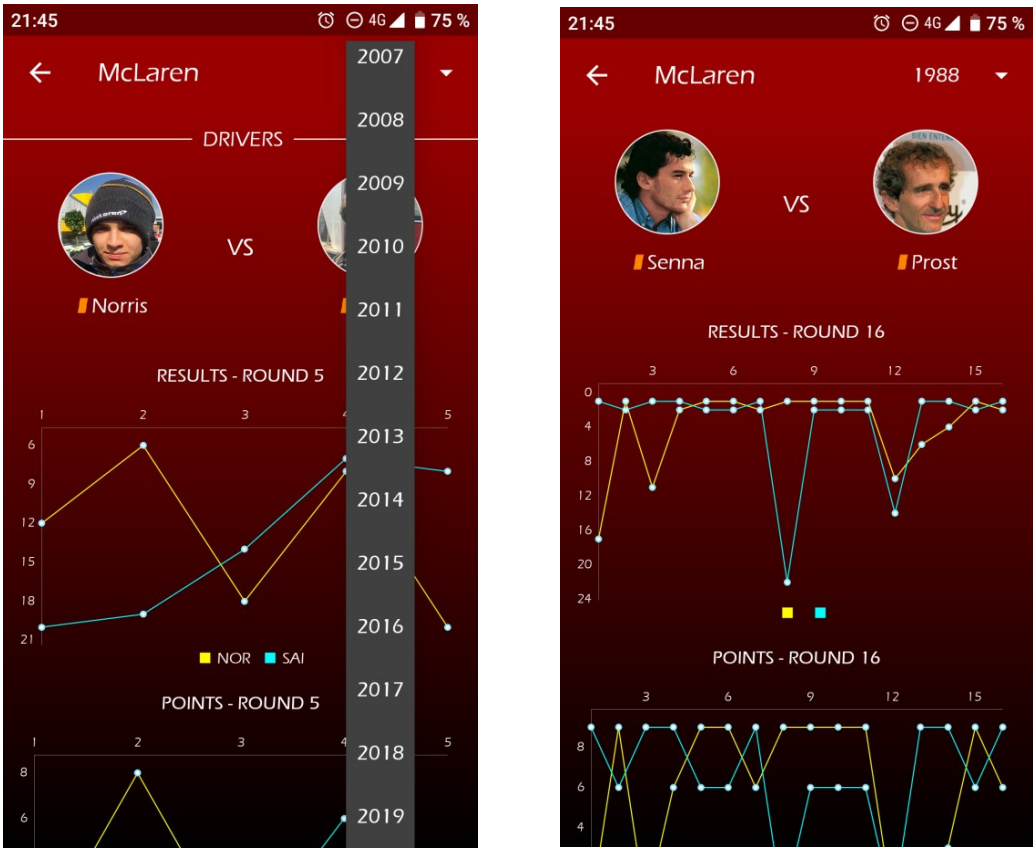


Ilustración 20. Capturas de Detalle de Constructor en la temporada 2019 y en 1988

3.6. Testing

Se han implementado algunos test unitarios en la aplicación, aprovechando las ventajas que ofrece el desacoplamiento de clases y la inyección de dependencias. Para ello se han mockeado las dependencias de las clases empleando los contratos definidos previamente en lugar de sus implementaciones, y así poder devolver los resultados que nos interesen para nuestros propósitos de testing.

Ejemplos de test instrumentales (Ejecutados en un Xiaomi MiA1)

```
@Test
fun on_activity_started_has_correct_content() {
    assertDisplayed(R.id.bottomNavigationView)
    assertDisplayed(R.id.container)
    assertDisplayed(R.id.myToolbar)
    assertDisplayed(R.id.racesRecyclerView)
}

@Test
fun on_constructors_clicked_shows_constructors()
    clickOn(R.id.action_constructors)
    assertDisplayed(R.id.constructorsRecyclerView
}

@Test
fun on_drivers_clicked_shows_drivers() {
    clickOn(R.id.action_drivers)
    assertDisplayed(R.id.driversRecyclerView)
}

@Test
fun on_races_clicked_shows_races() {
    clickOn(R.id.action_races)
    assertDisplayed(R.id.racesRecyclerView)
}

@Test
fun on_stats_clicked_shows_stats() {
    clickOn(R.id.action_stats)
    assertDisplayed(R.id.statsScrollView)
    assertDisplayed(R.id.selectSpinner)
    assertDisplayed(R.id.firstSpinner)
    assertDisplayed(R.id.secondSpinner)
}
```

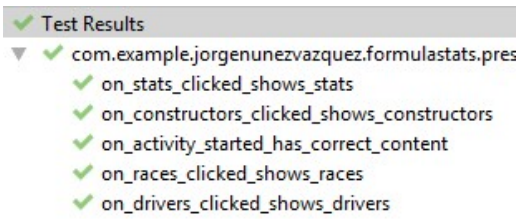


Ilustración 21. Test instrumentales de la aplicación

Ejemplos de tests unitarios ejecutados con JUnit4

```
@Test
fun `given drivers repository, when getDriverSeasonStandings returns list of driver season standings,
    getDriversSeasonStandingsResult = Success(driverStandingsWrapper)
    getDriverImageFromWikiResult = Success( data: "imageUrl")
    val result = driversRepository.getDriverSeasonStandings( season: "2019")
    Assertions.assertThat(result).isInstanceOf(Success::class.java)
    Assertions.assertThat((result as Success).data).isNotNull()
}

@Test
fun `given drivers repository, when getDriverSeasonStandings returns an Error, then error result`() {
    getDriversSeasonStandingsResult = Error()
    val result = driversRepository.getDriverSeasonStandings( season: "2019")
    Assertions.assertThat(result).isInstanceOf(Error::class.java)
}
```

Ilustración 22. Ejemplos de test unitarios de la aplicación

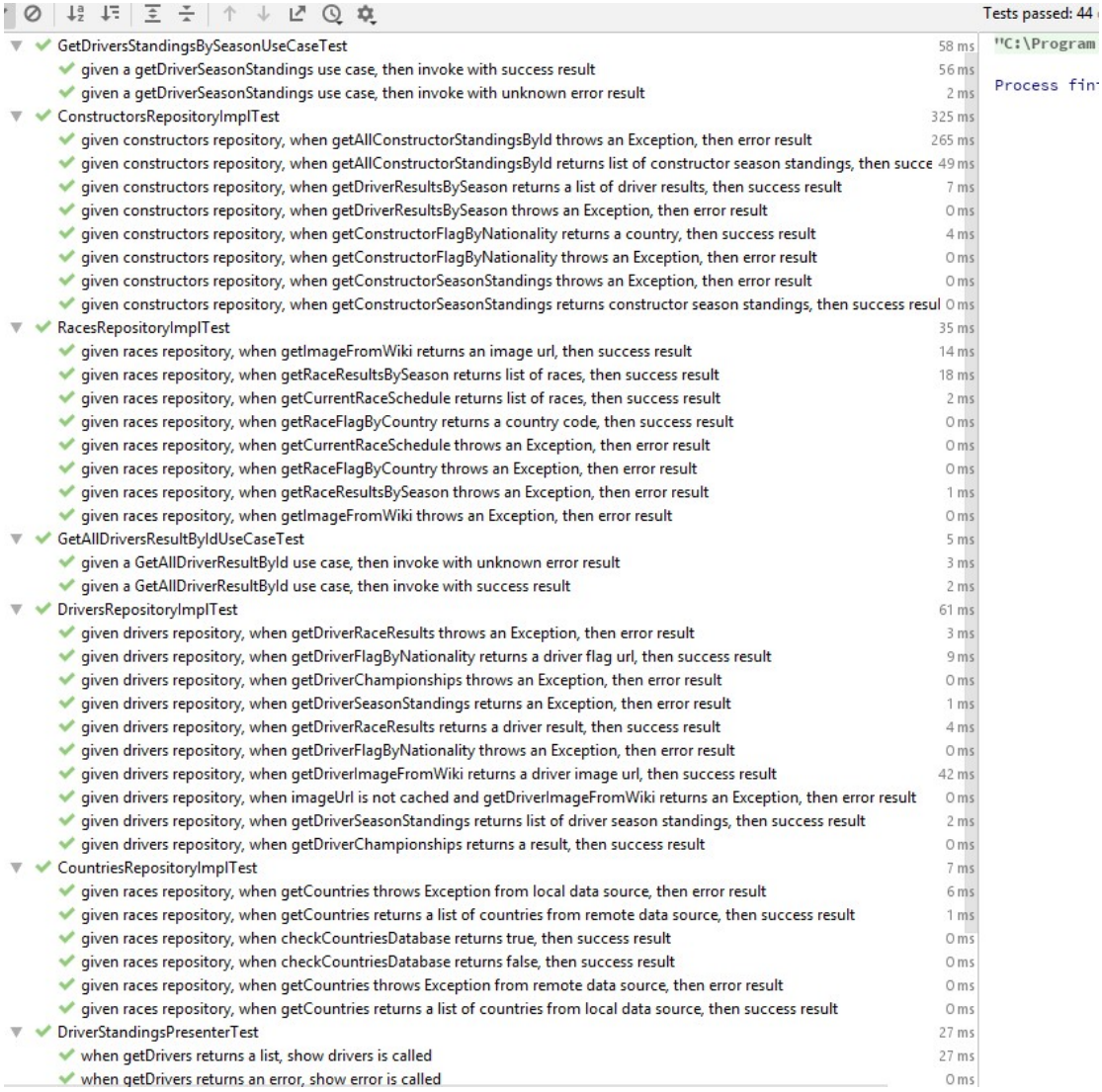


Ilustración 23. Listado de test unitarios ejecutados con la aplicación

3.7. Estado del proyecto

Esta fase del proyecto ha sido la de mayor complejidad y la que más retos ha propuesto a la hora de afrontarla. A pesar de que la versión de la aplicación en esta entrega es estable y posee todas las funcionalidades diseñadas, sí que ha habido una desviación con respecto a la planificación inicial, ya que por falta de tiempo no se ha logrado implementar la cantidad de test unitarios deseada, y el código necesita una mayor revisión antes de poder plantearse el lanzamiento de la aplicación a un nivel de producción, por lo que requerirá un esfuerzo extra en las semanas restantes. Entre los motivos de la desviación se pueden encontrar son los siguientes:

- Una estimación incorrecta de la carga de trabajo necesaria para desarrollar todas las funcionalidades de la aplicación, ya que la gran cantidad de clases necesarias y de datos manejados ha implicado más tiempo de desarrollo del esperado.

- La escasa experiencia en el uso de tecnologías como Kotlin, el patrón de arquitectura CLEAN, o algunas de las librerías empleadas, que han obligado a un esfuerzo mayor de estudio y desarrollo del autor.

Para poder aplicar medidas correctivas en el futuro se realizará un análisis de los bloqueos con respecto a la planificación inicial para intentar detectar en qué partes del desarrollo ha habido más desviaciones. Para ello se puede emplear el registro de nuestro repositorio GIT:

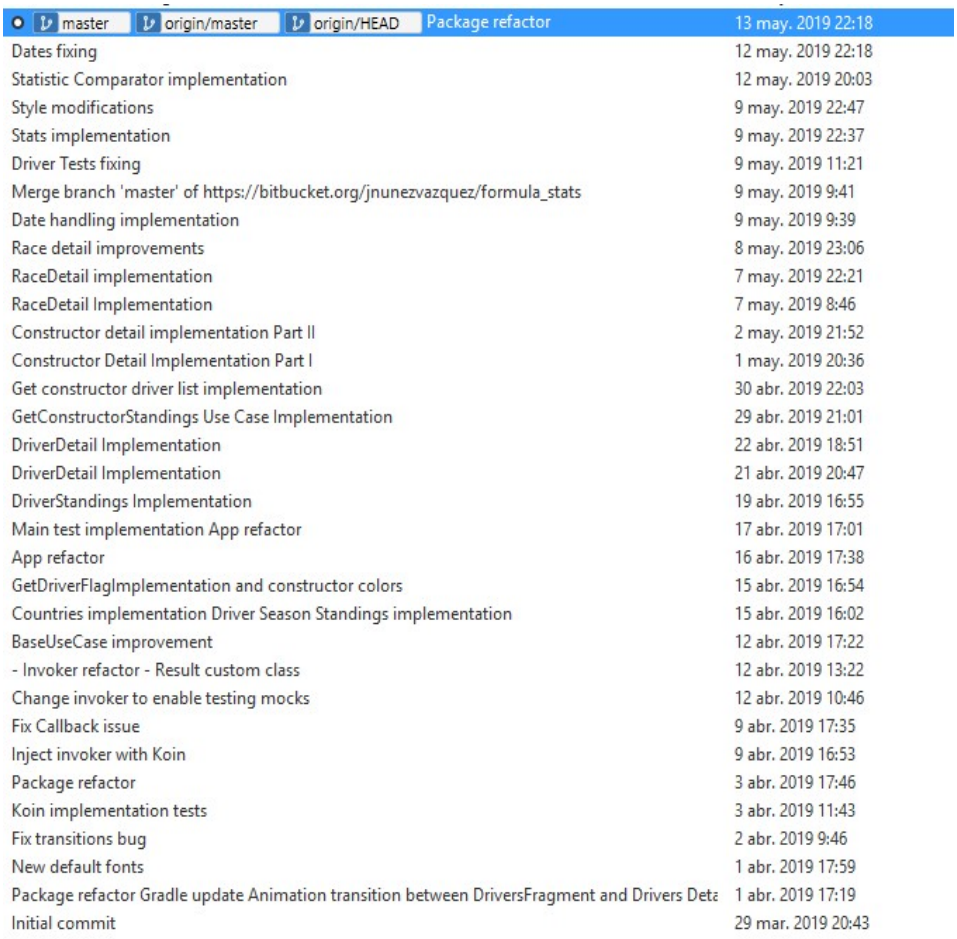


Ilustración 24. Captura del Control de Versiones del proyecto

Gracias a este registro podemos comprobar que los detalles gráficos han llevado más tiempo del estimado en la planificación, y deberá de tenerse en cuenta para futuros proyectos.

4. Testeo de la Aplicación y Revisión del Diseño

Una vez finalizado el desarrollo de la aplicación se dispone de una primera versión de la misma con todas las características especificadas en el capítulo de Diseño, pero antes de poder lanzarla a un nivel de producción (como puede ser publicarla en Google Play Store[28]) es necesario realizar un testeo a fondo para localizar y corregir las posibles fallas o *bugs* de la app.

Por otra parte, para intentar ofrecer la mejor experiencia de usuario posible debemos seguir investigando entre los potenciales usuarios de la aplicación, para recursivamente ajustar el diseño de la aplicación a sus necesidades e intentar que en cada nueva versión de la app podamos satisfacer y atraer a un público cada vez mayor. Para ello, se distribuirá una primera versión de la aplicación entre un pequeño número de usuarios (5-10) para que la prueben un par de semanas y después se les solicitará rellenar una breve encuesta sobre la misma.
Para ejecutar esta fase fundamental del proyecto, se emplearán dos herramientas de Google muy útiles para obtener los datos necesarios para poder testear la ejecución de la app y afinar su diseño: Firebase y Forms

4.1 Integración con Firebase

Al integrar la app con Firebase, se dispone de acceso a un gran número herramientas que son de gran utilidad a la hora de corregir errores y obtener datos de uso de los usuarios. Tras crear el proyecto en Firebase, descargar el archivo google-services.json que nos proporciona la plataforma y añadir las dependencias necesarias al archivo build.gradle de nuestro proyecto ya podemos acceder a todas las herramientas disponibles. A continuación se enumeran las más relevantes de cara al testeo de nuestra app:

- **Crashlytics:** Al integrar Crashlytics en la app se obtiene un registro muy valioso de los crashes de la misma, ya que indica en qué parte del código se produjo la falla y el motivo, número de usuario afectados, tipos de dispositivo, etc… y de esta forma es sencillo localizar y corregir un error y llevar un seguimiento del mismo. Durante las últimas semanas de desarrollo se han corregido los siguientes errores del proyecto gracias a esta herramienta.

[28] https://play.google.com/store

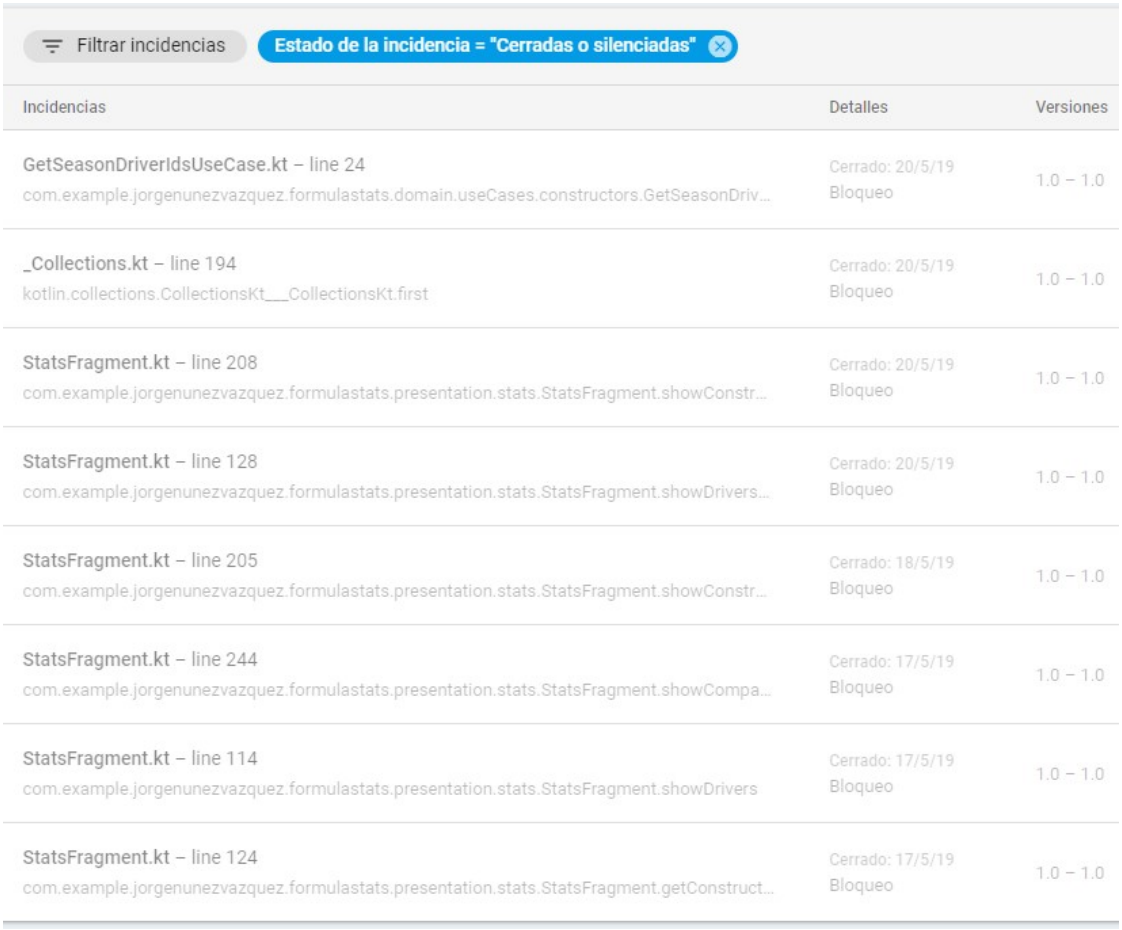


Ilustración 25. Incidencias cerradas en Crashlytics

Como se puede apreciar en la siguiente gráfica proporcionada por Crashlytics, una vez solucionados los anteriores errores no se han vuelto a detectar fallas en las últimas dos semanas.

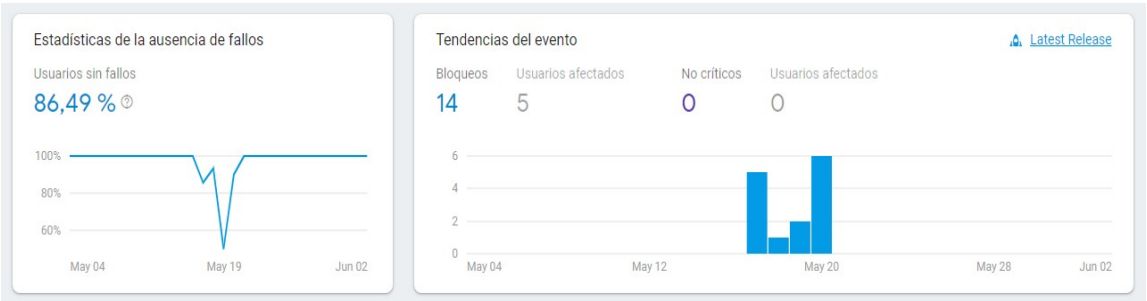


Ilustración 26. Bloqueos de los usuarios solucionados

- **Performance**: Gracias a esta herramienta podemos detectar, entre otros, los posibles problemas de renderizado de pantallas y de respuesta de red de nuestra aplicación. Tras varias semanas de uso no se han detectado incidencias relevantes en este sentido, como se puede apreciar en las siguientes gráficas aportadas por Firebase:

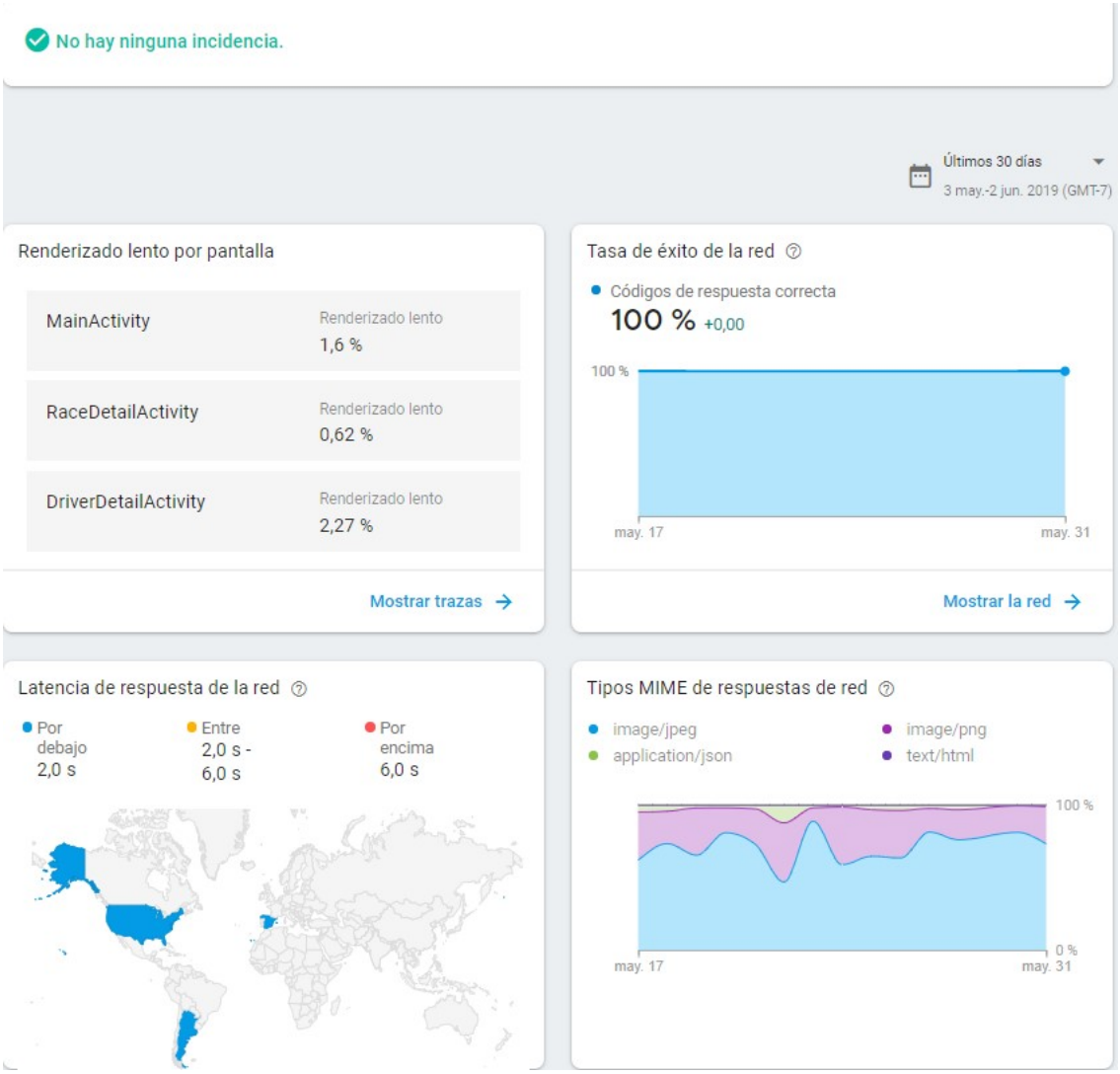


Ilustración 27. Gráficas principales de Firebase Performance

- **Test Lab**: Test Lab permite probar la app en un variado número de diferentes dispositivos Android con simplemente subir nuestro apk o bundle para su análisis. Estas pruebas se ejecutan en un laboratorio de dispositivos de Google, y permiten detectar posibles errores de ejecución. Nuestra aplicación ha superado estas pruebas en los siguientes dispositivos:

    - Pixel 3, nivel de API Q-beta-3
    - Pixel 2 XL, nivel de API 26
    - Galaxy J7 (SM-J710MN), nivel de API 23
    - Nexus 6, nivel de API 21
    - Galaxy Note 9 USA, nivel de API 27
    - Galaxy J1 ace SM-J111M, nivel de API 22
    - Razer Phone, nivel de API 25
    - Nexus 7 (2013), nivel de API 21
    - OnePlus One, nivel de API 22
    - Xperia XZ1 Compact, nivel de API 26
    - Low-resolution MDPI phone, Virtual, nivel de API 23
    - Huawei P8 lite, nivel de API 21

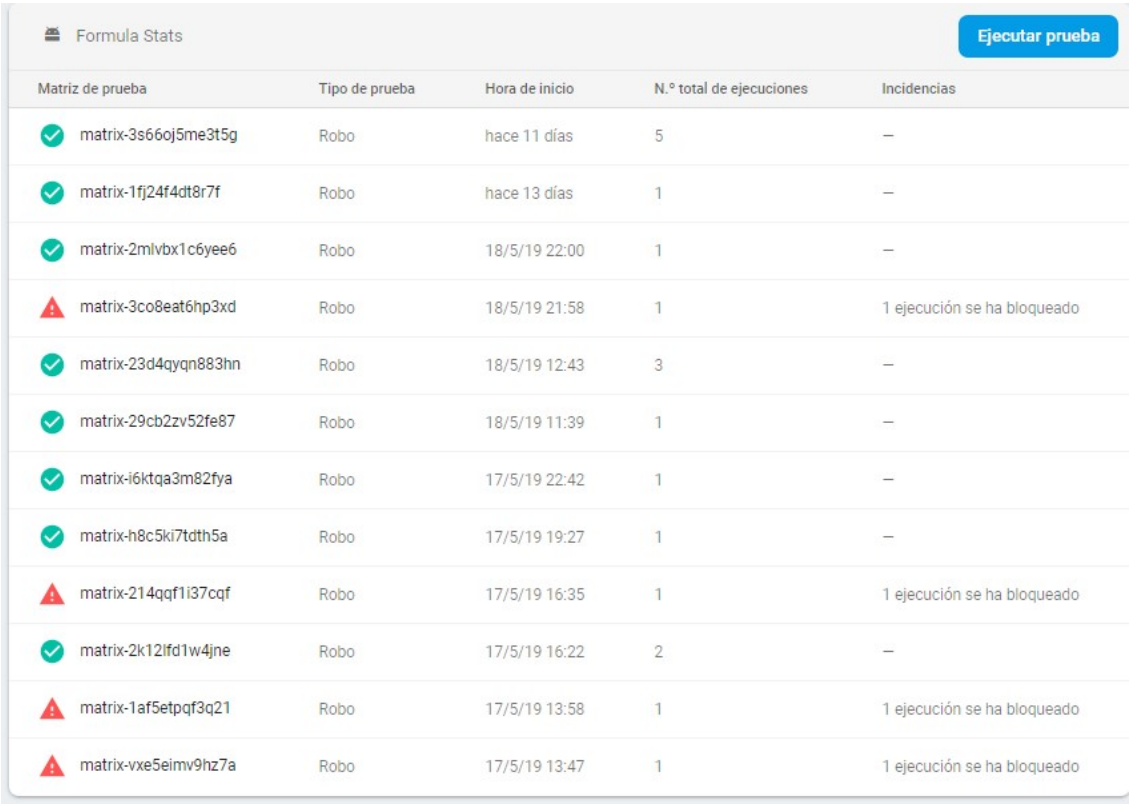


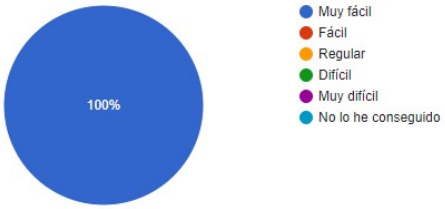
Ilustración 28. Pruebas Robo ejecutadas en Test Lab

4.2 Revisión del Diseño a través de UX

Para continuar con el proceso iterativo de la revisión del diseño, empleando la herramienta Google Forms se ha realizado un cuestionario entre algunos usuarios de la app, para obtener datos relacionados con su experiencia como usuarios y de esta forma buscar soluciones que puedan mejorar esta experiencia en futuras versiones de la aplicación. Estos son los resultados de dicho cuestionario:

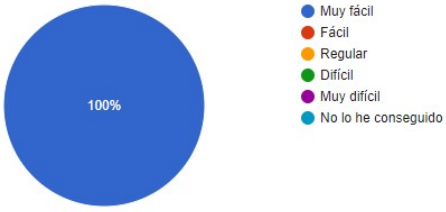
1. Intente averiguar la fecha y hora de la próxima carrera. Por favor, indique la dificultad que ha implicado:

5 respuestas



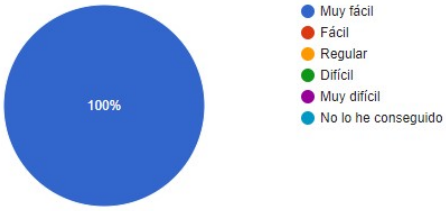
2. Intente averiguar qué piloto ha quedado en quinta posición en la última carrera disputada. Por favor, indique la dificultad que ha implicado:

5 respuestas



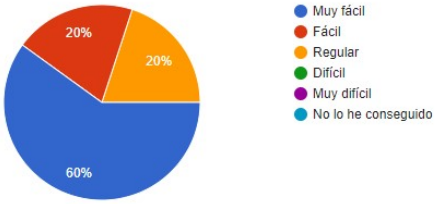
3. Intente averiguar qué piloto es el sexto clasificado en el campeonato actual. Por favor, indique la dificultad que ha implicado:

5 respuestas



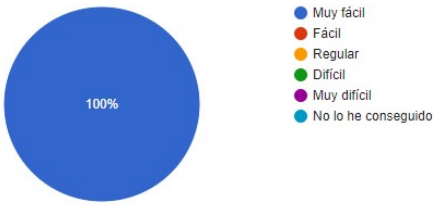
4. Intente averiguar en qué posición final quedó Carlos Sainz en el campeonato de 2017. Por favor, indique la dificultad que ha implicado:

5 respuestas



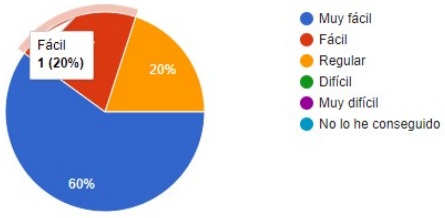
5. Intente averiguar qué constructor es el séptimo clasificado en el campeonato actual. Por favor, indique la dificultad que ha implicado:

5 respuestas



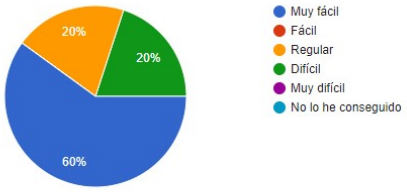
6. Intente averiguar qué pilotos corrían para MCLaren en 1988. Por favor, indique la dificultad que ha implicado:

5 respuestas



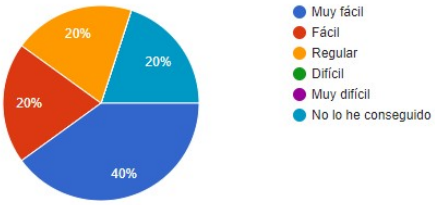
7. Intente averiguar qué piloto ha ganado más carreras: Fernando Alonso o Ayrton Senna. Por favor, indique la dificultad que ha implicado:

5 respuestas



8. Intente averiguar qué constructor ha ganado más puntos a lo largo de su historia: Ferrari o MCLaren. Por favor, indique la dificultad que ha implicado:

5 respuestas



9. Qué es lo que más te ha gustado de la aplicación:

5 respuestas

Intuitiva, buen diseño UI, buenos tiemoos de respuesta
Cumple perfectamente su función con una estética atractiva
La información
Las comparativas pilotos.
Las graficas

10. Qué es lo que menos te ha gustado de la aplicación:

4 respuestas

Los combos para elegir pilotos en la parte de estadisticas pueden ser eternos. Para buscar a Senna (que aparce duplicado) tardé casi un minuto
El desplegable del comparador de estadísticas es muy completo pero quizá demasiado extenso. Una buena opción sería añadirle un buscador de texto manual
Nada significativo
La falta de vaores numéricos comparativas

Firebase Analytics

Al integrar nuestro proyecto con Firebase, se dispone acceso a información muy valiosa acerca de nuestros usuarios y su experiencia de uso. Podemos añadir nuestros propios eventos a la aplicación para recopilar información sobre los contenidos más visitados o las funcionalides menos empleadas, y de esta forma poder analizar estos datos para intentar mejorar nuestro diseño.

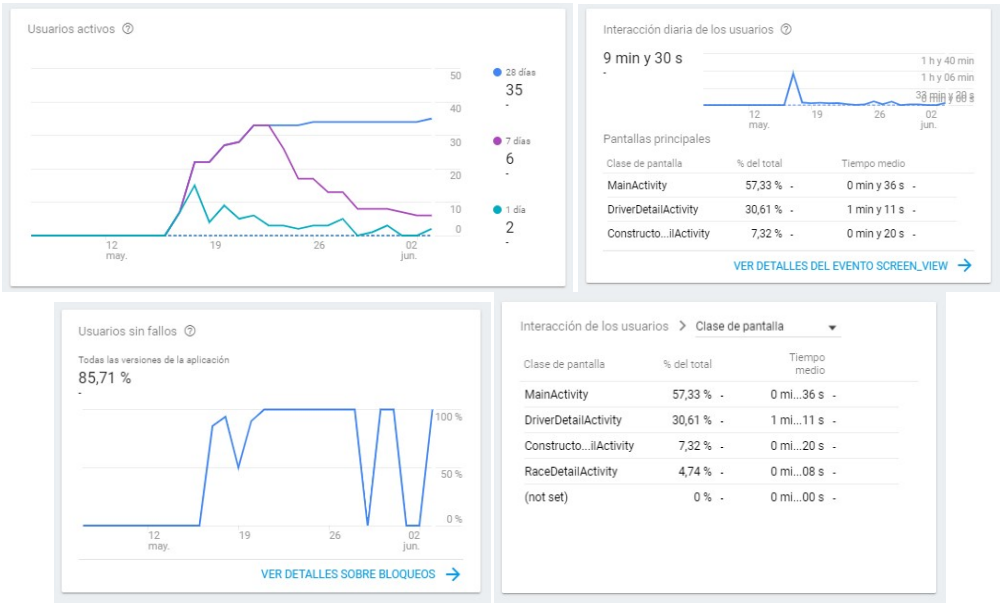


Ilustración 29. Gráficas de Firebase Analytics

5. Conclusiones

Mediante la ejecución de este proyecto se ha elaborado una aplicación completamente funcional, que cumple con los requisitos establecidos durante la fase de diseño, y que ha sido sólidamente testeada.

A través de las encuestas realizadas a los usuarios y de los datos obtenidos de las pruebas realizadas en Firebase, se ha podido confirmar que la experiencia de usuario es intuitiva y agradable, y que la aplicación muestra un funcionamiento correcto en multitud de dipositivos en cuanto a rendimiento, respuestas de red, ausencia de bloqueos o renderizado de pantallas.

Sin embargo, estos mismos datos indican que aún queda mucho margen de mejora para la aplicación, y el siguiente paso sería analizar el feedback de los usuarios para buscar formas de modificar el diseño en próximas versiones de cara a mejorar la experiencia de uso.

En cuanto al seguimiento de la planificación, en general ha sido el esperado en cuanto a recursos, tiempo y esfuerzo, encontrando los mayores problemas en la fase de implementación, principalmente debido a la escasa experiencia con el lenguaje de programación y la arquitectura, de lo cual se puede obtener una lección para futuros proyectos personales.

Con todo ello, se considera que se han cumplido el objetivo principal marcado al inicio del proyecto, que era fundamentalmente el poner en práctica los conocimientos adquiridos durante el desarrollo de este Máster mediante la creación de una aplicación móvil, desde su fase de concepción hasta llegar a un nivel de producción, empleando las herramientas y los paradigmas de programación más actuales, y buscando siempre una grata experiencia por parte de los usuarios gracias a la técnicas que proporciona el Diseño UX.

4. Glosario

Servicio Web: Un servicio web (en inglés, *web service* o *web services*) es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones

App híbrida: Una aplicación híbrida es un programa de software que combina elementos de aplicaciones nativas y web.

App nativa: Una aplicación nativa es un programa de software que se desarrolla para su uso en una plataforma o dispositivo en particular.

Plataforma: En informática, una plataforma de desarrollo es el ambiente o entorno de software común en el cual se desenvuelve la programación de un grupo definido de aplicaciones. Comúnmente se encuentra relacionada directamente a un sistema operativo

CLEAN: se basa en la premisa de estructurar el código en capas contiguas, es decir, que solo tienen comunicación con las capas que están inmediatamente a sus lados. Cada nivel debe realizar sus propias tareas y se comunica únicamente con sus niveles inmediatamente contiguos.

MVP: Model-View-Presenter es una derivación del modelo arquitectónico Model-View-Controller (MVC) que se utiliza principalmente para crear interfaces de usuario. En MVP, toda la lógica se envía al Presenter, y se aboga por separar la lógica de negocio y el Modelo de la Vista.

Kotlin: Lenguaje de programación orientada a objetos (OOP) de tipo estático que es interoperable con la máquina virtual Java, las bibliotecas Java y Android.

Wireframe: Guía visual que representa el marco esquelético de una de una aplicación móvil

API: La interfaz de programación de aplicaciones, conocida también por la sigla API, en inglés, *application programming interface*, es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción

Agile: La gestión ágil de proyectos es un enfoque iterativo para planificar y guiar los procesos del proyecto. Al igual que en el desarrollo de software ágil, un proyecto ágil se completa en pequeñas secciones llamadas iteraciones. Cada iteración es revisada y criticada por el equipo del proyecto, que puede incluir representantes de la empresa cliente, así como empleados

SOLID: Acrónimo introducido por Robert C. Martin a comienzos de la década del 2000 que representa cinco principios básicos de la programación orientada a objetos y el diseño

GIT: Software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la fiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente.

Bug fixing: Cambio realizado a un sistema o producto diseñado para manejar un error o fallo de programación.

POJO: Sigla creada por Martin Fowler, Rebecca Parsons y Josh MacKenzie en septiembre de 2000 y utilizada por programadores Java para enfatizar el uso de clases simples y que no dependen de un framework en especial

Apk: Variante del formato JAR de Java y se usa para distribuir e instalar componentes empaquetados para la plataforma Android para teléfonos inteligentes y tabletas

Benchmarking: Consiste en tomar "comparadores" o benchmarks de aquellos productos, servicios y procesos de trabajo que pertenezcan a organizaciones que evidencien las mejores prácticas sobre el área de interés, con el propósito de transferir el conocimiento de las mejores prácticas y su aplicación

Posicionamiento ASO: También conocido como Optimización ASO, es el proceso que se lleva a cabo para conseguir aparecer en las primeras posiciones o resultados de búsqueda en las tiendas de App móviles o App Store

Framework: Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar

Repositorio: Espacio centralizado donde se almacena, organiza, mantiene y difunde información digital, habitualmente archivos informáticos, que pueden contener trabajos científicos, conjuntos de datos o software

Mock: Objetos preprogramados con expectativas que conforman la especificación de lo que se espera que reciban las llamadas

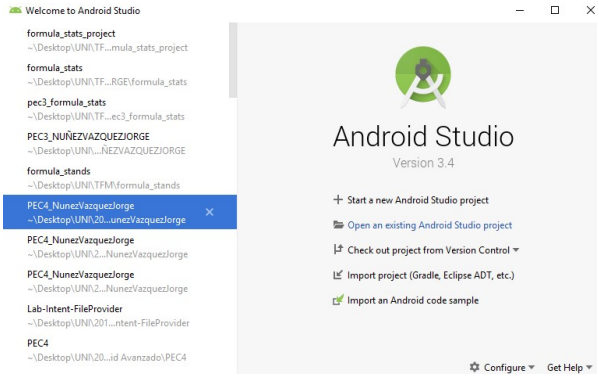
5. Bibliografía

[1] https://ergast.com/mrd/gallery/ [Visitado el 13/03/2019]
[2] http://gs.statcounter.com/os-market-share/mobile/worldwide [Visitado el 13/03/2019]
[3] http://gs.statcounter.com/os-market-share/mobile/brazil [Visitado el 13/03/2019]
[4] http://gs.statcounter.com/os-market-share/mobile/china [Visitado el 13/03/2019]
[5] https://www.formula1.com/en/latest/article.formula-1s-tv-and-digital-audiences-grow-for-the- second-year-running.OqTPVNthtZKFbKqBaimKf.html [Visitado el 13/03/2019]
[6] http://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html [Visitado el 13/03/2019]
[7] https://www.raywenderlich.com/7026-getting-started-with-mvp-model-view-presenter-on-Android [Visitado el 13/03/2019]
[8] https://kotlinlang.org/ [Visitado el 13/03/2019]
[9] https://agilemanifesto.org/ [Visitado el 13/03/2019]
[10] https://es.wikipedia.org/wiki/SOLID [Visitado el 03/04/2019]
[11] https://play.google.com/store [Visitado el 03/04/2019]
[12] https://www.mediawiki.org/w/api.php [Visitado el 03/04/2019]
[13] https://www.formula1.com/en/toolbar/legal-notices.html [Visitado el 03/04/2019]
[14] https://www.axure.com/ [Visitado el 03/04/2019]
[15] https://www.flaticon.com/ [Visitado el 03/04/2019]
[16] http://www.nosolousabilidad.com/manual/3_2.htm [Visitado el 03/04/2019]
[17] https://antonioleiva.com/clean-architecture-android/ [Visitado el 03/04/2019]
[18] https://medium.com/cr8resume/make-you-hand-dirty-with-mvp-model-view-presenter-eab5b5c16e42 [Visitado el 03/04/2019]
[19]http://blog.safedk.com/technology/solving-androids-65k-limit-part-2-the-lollipop-generation/
[Visitado el 14/05/2019]
[20] https://developer.android.com/topic/libraries/architecture/room
[Visitado el 14/05/2019]
[21] https://github.com/InsertKoinIO/koin [Visitado el 14/05/2019]
[22] https://square.github.io/picasso/ [Visitado el 14/05/2019]
[23] https://github.com/hdodenhof/CircleImageView [Visitado el 14/05/2019]
[24] https://github.com/paolorotolo/ExpandableHeightListView [Visitado el 14/05/2019]
[25] https://github.com/PhilJay/MPAndroidChart [Visitado el 14/05/2019]
[26] https://joel-costigliola.github.io/assertj/ [Visitado el 14/05/2019]
[27] https://github.com/SchibstedSpain/Barista [Visitado el 14/05/2019]
[28] https://developer.android.com/training/multiple-threads/create-threadpool
[Visitado el 14/05/2019]

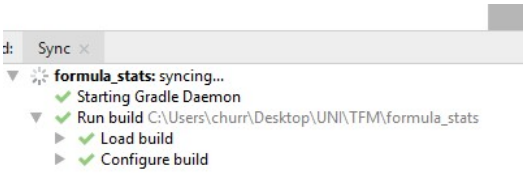
6. Anexos

ANEXO I. Instrucciones de compilación

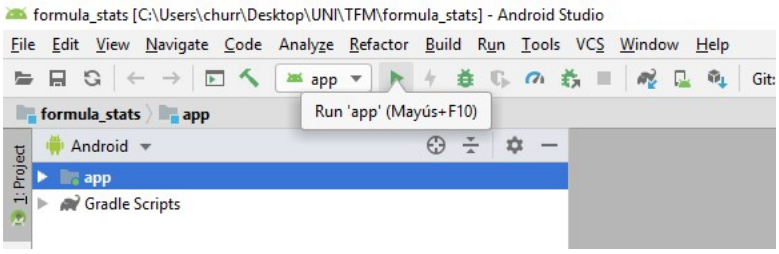
1) Abrir el Proyecto con Android Studio[29]



2) Esperar a que Gradle configure la Build del proyecto



3) Ejecutar “app” en el botón Play de Android Studio



[29] Nota: Testeado con Android Studio Versión 3.4

ANEXO II. Manual de Usuario