



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

Flutter



Presentado por Samuel Casal Cantero
en Universidad de Burgos — 18 de julio
de 2020

Tutor: César García Osorio y Francisco Javier
Diez Pastor



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



D. José Francisco Díez Pastor y D. César García Osorio. profesores del departamento de Ingeniería Informática. área de Lenguajes y Sistemas Informáticos. Expone:

Que el alumno D. Samuel Casal Cantero, con DNI 71301273p, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Flutter.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 18 de julio de 2020

Vº. Bº. del Tutor:

Vº. Bº. del co-tutor:

D. nombre tutor

D. nombre co-tutor

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Android, Flutter, Dart, VisualStudio Code, Github, ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

Android, Flutter, Dart, VisualStudio Code, Github, ...

Índice general

Índice general	III
Índice de figuras	V
Índice de tablas	VI
Introducción	1
1.1. Estructura de la memoria	2
Objetivos del proyecto	5
2.1. Objetivos generales	5
2.2. Objetivos técnicos	5
2.3. Objetivos personales	6
Conceptos teóricos	7
3.1. Aplicaciones nativas	7
3.2. Aplicaciones híbridas mediante Framework	8
3.3. Flutter	11
Técnicas y herramientas	15
4.1. Sistema Operativo	15
4.2. Control de versiones	16
4.3. Gestión de proyecto	17
4.4. Entorno de desarrollo integrado (IDE)	18
4.5. Entorno de virtualización	19
4.6. Herramientas de comunicación	20
4.7. Documentación	20

4.8. Herramientas	21
4.9. Metodologías	22
4.10. Bibliotecas	23
Aspectos relevantes del desarrollo del proyecto	25
5.1. Comienzo del trabajo final de grado	25
5.2. Flutter es asíncrono	26
5.3. Nuevas formas de hacer if	28
5.4. Play Store	28
5.5. AdMob	28
Trabajos relacionados	31
Conclusiones y Líneas de trabajo futuras	33
7.1. Conclusiones	33
7.2. Líneas de trabajo futuras	34
Bibliografía	37

Índice de figuras

1.1. Cuota de mercado	2
3.2. Encuesta en el foro	9
3.3. Inicio de sesión	13
3.4. Ejemplo de Analytics	14
4.5. Herramienta Zenhub	18
4.6. Desarrollo en cascada	22
4.7. Cuaderno	23
5.8. Ejemplo de Future	27
5.9. Nuevas formas if	28
5.10. Banner	29
5.11. Métricas usuarios	29
5.12. Pagos adMob no van	30

Índice de tablas

3.1. Herramientas y lenguaje nativos	7
--	---

Introducción

Comentar brevemente que la idea inicial para el proyecto no era la de crear una aplicación para dispositivos móviles. Si no que se pensó en otros trabajos:

- A finales de 2019 la idea era hacer el proyecto dentro de la empresa donde me encontraba haciendo las prácticas curriculares (ITCL). El trabajo consistía en una aplicación de reconocimiento de la mosca de la fruta. Pero no prosperó.
- La siguiente propuesta de mis tutores fue la de trabajar con la herramienta *Knime* junto con *Moodle*. Dedique muchas horas a este proyecto, pero era de investigación y no me motivaba, por lo que se acabó descartando esta opción.

Finalmente para no perder la convocatoria me decante por hacer una aplicación Android mediante Flutter [?]. El por qué de esta decisión es que la cuota de mercado para este sistema operativo es enorme [?], con un fuerte crecimiento hasta 2018, estabilizándose y actualmente es el líder del mercado como podemos ver en la siguiente imagen 1.1.

También debemos de remarcar que la cuota de mercado de los sistemas operativos tradicionales no para de descender, lo que implica que los móviles han venido para quedarse, pero no para sustituir a los ordenadores. Por lo que es una apuesta de futuro aprender el desarrollo de estas aplicaciones móviles, ya que la demanda es muy grande.



Figura 1.1: Cuota de mercado

Básicamente me decanté por Flutter porque permite la creación de aplicaciones para dos sistemas operativos (Android e iOS) y web. Todo ello programando en un mismo lenguaje, con las ventajas que esto conlleva, además de que es de Google con el respaldo que ofrece.

El contenido de la aplicación estaría basado en una librería / colección de videojuegos. De tal manera que sería el hilo conductor con el que poder tocar gran cantidad de tecnologías y aprender en profundidad del funcionamiento de Flutter.

1.1. Estructura de la memoria

La memoria sigue la siguiente estructura:

- **Introducción:** breve descripción del problema a resolver y la solución propuesta. Estructura de la memoria y listado de materiales adjuntos.
- **Objetivos del proyecto:** exposición de los objetivos que persigue el proyecto.
- **Conceptos teóricos:** explicación de los conceptos teóricos clave para el entendimiento de la aplicación.

- **Técnicas y herramientas:** listado de técnicas metodológicas y herramientas utilizadas para gestión y desarrollo del proyecto.
- **Aspectos relevantes del desarrollo:** exposición de aspectos destacables que tuvieron lugar durante la realización del proyecto.
- **Conclusiones y líneas de trabajo futuras:** conclusiones obtenidas tras la realización del proyecto y posibilidades de mejora o expansión de la solución aportada.

Objetivos del proyecto

A continuación, se detallarán los objetivos que han motivado la realización de este proyecto así como los resultados que se desean conseguir.

2.1. Objetivos generales

- Desarrollar una aplicación para *smartphone*.
- Implementación y despliegue de la app en la tienda de de Google.
- Obtener el rendimiento económico mediante publicidad.
- Hacer que los usuarios pasen un buen rato.
- Mostrar quien es el creador de la aplicación, con el fin de poder usar esta como herramienta o portfolio.
- Desarrollar juegos.

2.2. Objetivos técnicos

- Aprender una alternativa moderna a javascript mediante Dart.
- Comprender el funcionamiento de Flutter.
- Control de versiones con la herramienta GitHub, mediante comandos a través de GitBash.
- Generar documentación de todo el proceso en \LaTeX .

- Realizar una planificación mediante *Scrum* eficiente, a través de la herramienta ZenHub, integrada en GitHub.
- Comenzar a usar las herramientas telemáticas para las reuniones con los tutores del trabajo final de grado.
- Comunicación de la aplicación mediante WebServices.
- Conocer y aprender a usar las herramientas que proporciona el *Cloud Services* de Google.
- Diseñar la arquitectura de la aplicación.

2.3. Objetivos personales

- Adquirir el conocimiento necesario para desarrollar aplicaciones móviles y multiplataforma. Es decir, para tres entornos: Android, iOS y web.
- Aprobar el trabajo de fin de grado, ya que es un reto.
- Estudiar como generar documentación en \LaTeX .
- Aprender que sin esfuerzo no hay recompensa.
- Comprender el funcionamiento de la comunicación entre la una aplicación y los servicios en la nube.
- Reforzar los conocimientos adquiridos durante la carrera.
- Investigar diferentes herramientas para solventar los problemas que salgan.
- Adquirir las nociones necesarias para poder llevar proyectos.
- Conseguir manejarme en entornos de incertidumbre.

Conceptos teóricos

La parte del proyecto más importante es el proceso de lanzar una aplicación Android desde cero, para ello se ha tenido que realizar una investigación para determinar que herramientas, lenguajes de programación, entornos de desarrollo son los más apropiados para un despliegue ágil.

Todo esto se sitúa en un mercado altamente competitivo, con una gran variabilidad (usuarios, empresas, desarrolladores ...) y con un constante cambio.

3.1. Aplicaciones nativas

Estas se denominan así porque se desarrollan en el lenguaje nativo de cada uno de los sistemas operativos. Dependiendo de la plataforma en la que se desee crear esta nueva aplicación requerirá conocer un lenguaje de programación y un entorno de desarrollo en concreto. A continuación se muestra una tabla 3.1 con las herramientas y lenguajes nativos:

Sistema Operativo	Lenguaje Programación	Entorno
Android	Java	Android Studio
iOS	Objective C, Swift	Xcode

Tabla 3.1: Herramientas y lenguaje nativos

Las ventajas principales son: que están desarrolladas directamente sobre la capa nativa del dispositivo, el rendimiento será óptimo y todas las funcionalidades y características estarán disponibles desde el primer momento.

Entre las desventajas: el elevado coste y mantenimiento, ya que requiere personal más especializado, con mayor tiempo destinado al desarrollo. Pero el mayor *gap* lo encontramos en que el código nativo de una plataforma no puede ser reutilizado para la otra. Algo que los frameworks actuales están empezando a ofrecer.

3.2. Aplicaciones híbridas mediante Framework

Un framework (de origen anglosajón, marco de trabajo), según lo que dice la wikipedia [?], es una estructura conceptual y tecnológica de soporte definido, normalmente por módulos de software concretos, que sirve de base para la organización y el desarrollo de software. Las ventajas que ofrece son varias, entre las que se puede destacar:

- Único código fuente: desarrollo de aplicaciones multiplataforma con un único lenguaje, reduciendo los costes de creación / mantenimiento y recursos destinados.
- **Evita repetición de código:** las partes más usadas, pasan a ser algo del *core* del framework.
- **Uso de buenas prácticas:** muchos de estos están basados en patrones de diseño que nos obligan a usar.
- **Elementos avanzados integrados:** dispone de librerías nativas que ofrecen *Widgets* o funcionalidades de gran calidad, que llevarían mucho tiempo implementar, o que la creación de los mismo desde cero, no llegaría a obtener una calidad del mismo nivel.
- **Desarrollo ágil:** por los factores anteriores, podemos centrarnos más en la lógica de negocio de la aplicación que se desea hacer, de una manera más rápida y segura.

Por lo tanto, es necesario trabajar mediante frameworks, ya que nos garantizar una aplicación de mayor calidad, que si la hacemos desde cero, en código nativo.

Opciones disponibles

En el mercado existen muchos frameworks disponibles para el desarrollo de aplicaciones móviles, por lo que decantarse por uno no es tarea sencilla, ya que como se aprecia cada uno tiene sus pros y contras. En una encuesta realizada en un foro [?], a mediados del 2019, la opinión de los más recomendados era Flutter como se puede ver en la imagen 3.2:



Figura 3.2: Encuesta en el foro

Algo ideal es que se intento buscar un framework *crossplataform*, con el que abarcar un mayor mercado, a nivel de usuarios y dispositivos.

Ionic

Ionic [?] se basa en lenguaje de programación javascript, html y css. Internamente esta basado en otro framework, como es Angularjs. La licencia Open source, fue lanzado en el 2013, permite el desarrollo para iOS y Android. A sufrido gran cantidad de cambios desde entonces.

Ventajas:

- Una amplia comunidad de usuarios.
- Documentación extensa y de calidad.

Algunas de las desventajas:

- Rendimiento algo menor, ya que no se desarrolla de forma nativa.
- Bibliotecas en constante cambio y evolución, más que ser algo bueno, puede que deje a la aplicación fuera de versión y se tenga que hacer desde cero.

React native

React native [?] fue creado en 2015, de la mano de Facebook. La licencia que tiene es MIT. La diferencia que tiene con React, es que no manipula el DOM, por lo que tampoco usa HTML o CSS. Se programa en javascript. Las aplicaciones más conocidas que usan este framework es instagram o facebook.

Ventajas:

- Gran comunidad de usuarios y herramientas.
- Mejora de las funcionalidades constantemente.

Algunas de las desventajas:

- Rendimiento algo menor.
- No es código nativo, pero casi, por lo que no tiene soporte oficial de Google y Apple.

Xamarin

Xamarin [?] creado por Microsoft en el 2011, por lo que está desarrollado en .NET, siendo propietario.

Ventajas:

- Permite el desarrollo de iOS, Android, web pero nativas de escritorio también.
- Puede llamar a fragmentos de código usados en otras plataformas.
- Soporte para wearables.

Algunas de las desventajas:

- Acceso limitado a las bibliotecas.
- Soporte y actualizaciones lento / tarde.
- Comunidad grande pero pequeña comparada con otras.
- Aplicaciones de mayor tamaño.
- Coste.

Otros

Hay muchos otros, como kotlin, Apache Cordoba, jQuery Mobile, Native script ... Pero no me convencieron por diversas razones.

3.3. Flutter

Flutter [?] es un SDK de código abierto creado por Google a finales del 2018. Permite que los desarrolladores puedan crear aplicaciones para iOS, Android y web.

Este se encuentra escrito en Dart, que es un lenguaje de programación creado por Google en 2011. Este lo que hace es una mejora del lenguaje javascript, pero sin pretender sustituirlo. Es decir, ofrecer mejores resultados y alternativas para algunos problemas, siendo una herramienta mejor, para proyectos más grandes, como es el caso de flutter.

Por lo tanto Flutter es una herramienta mi nueva, con la que se pueden hacer aplicaciones comerciales para varias plataformas sin tener que programar exclusivamente para cada una de ellas. Ya que nos ofrece la compilación nativa directamente, reduciendo los costes a la hora de tener que llevar proyectos en los que sea necesario estar en los dos mercados.

Las características más importantes son :

- Desarrollo rápido de las aplicaciones. Cuando se dice rápido es porque permite *hot reload*, esta es la carga en caliente durante la fase de desarrollo. Implica que nos es necesario tener que compilar todo el código, si no aquellas partes que fueron modificadas. Lo que permite en tiempo de ejecución ver los cambios.
- Está muy optimizado, con una evolución y soporte constante.
- Es un lenguaje de programación respaldado por Google, con lo que esto conlleva. Seguridad, confianza, fiabilidad, soporte, cursos y un montón de herramientas(más de 300 apis distintas: mapas, reconocimiento facial, traductor ...).
- La integración con el sistema operativo Android es mucho más eficaz, ya que este también pertenece a Google.
- Cuando se compila, lo hace a nativo, siendo una ganancia de rendimiento.

- La curva de aprendizaje es baja, en el caso de que sepamos javascript, typescript o ecmaScript.
- La calidad de las animaciones.

Las desventajas que tiene son:

- La mayor parte de la documentación se encuentra en inglés, pero es más problema de desarrollador, que de la propia documentación del framework.
- Las aplicaciones ocupan más espacio que las nativas, ya que suelen incluir el SDK al completo.
- Al ser tan nuevo, tiene algunos problemas, no ofrece todas las funcionalidades, como las que ofrecen otras plataformas. Google lo sabe y está apostando mucho por ello.
- Es una comunidad pequeña, pero a crecido en dos años más que otras, en tiempos similares.

Al final me decanto por este Framework básicamente porque es algo nuevo y disruptivo. El respaldo de Google se me presentaba como garantía, con la tranquilidad que eso conlleva. Es decir, todo el *cloud services de Google* se puede integrar con gran facilidad. Otra de las cosas que fueron de agrado es que la comunidad lo a recibido con los brazos abiertos, como se puede ver en la encuesta 3.2, se encuentra entre los favoritos de los desarrolladores, por algo será.

Firestore

Firestore [?] es una plataforma para el desarrollo de aplicaciones web, Android e iOS, creada por Google en el 2014. Esta en la nube, formando el *Google Cloud Platform*, que consta de un conjunto de herramientas para dotar de una calidad enorme a los proyectos. Ya que permite la integración del ecosistema de Google como un todo.

Fue integrada en el proyecto por el hecho de ser un servicio gratuito y que aportaba gran valor a la aplicación. Este pasa a ser de pago cuando la app tiene que escalar a un nivel más grande, por el tema de usuarios o el número de peticiones que se realicen a la misma. Algunas de las herramientas que integra son de pago, otras no.

Por lo tanto al ser multiplataforma es un backend con el que poder controlar todo, ganancias, gastos, escalabilidad, nuevos productos, herramientas ...

Entre los servicios que ofrece tenemos:

- **Real Time Data Base:** base de datos simples en tiempo real, si fuera necesario tener que guardar música, video o fotos, tiene otra parte que es la de Storage.
- **Crash reporting:** herramienta para el reporte de errores que se producen en la aplicación.
- **Autentication:** integración con la mayor cantidad de aplicaciones de redes sociales en las que su API permite esto. Obviamente la propia Google es una de ellas. Lo podemos ver en la siguiente imagen 3.3:

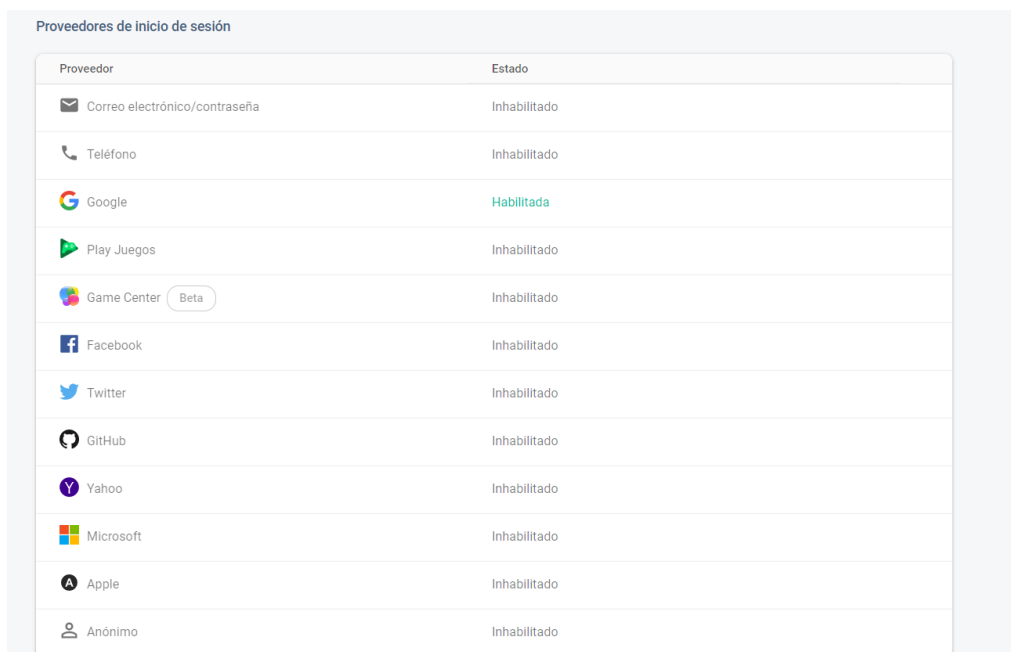


Figura 3.3: Inicio de sesión

- **Test lab:** probar la aplicación antes de realizar el despliegue de la misma.

- **Remote config:** hacer cambios internos del funcionamiento de la aplicación sin tener que recompilar o actualizarla.
- **Hosting:** servidor donde podemos publicar una página web.
- **Análisis:** mediante las analíticas que ofrece 3.4, sirve como herramienta de toma de decisiones. Pudiendo optimizar a que mercados dirigirse mediante una estrategia de marketing.

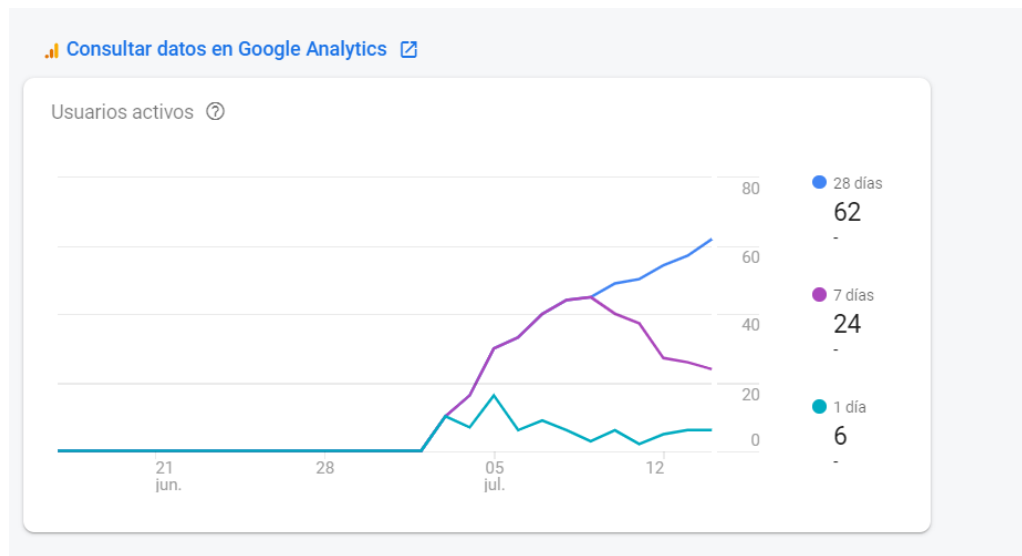


Figura 3.4: Ejemplo de Analytics

Técnicas y herramientas

Esta parte de la memoria tiene como objetivo presentar las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto. Se comentará de manera breve, las diferentes opciones y la razón porque éstas fueron descartadas.

4.1. Sistema Operativo

Opciones elegidas

Microsoft

Microsoft es el sistema operativo generalista por excelencia. Es de sobra conocido, con sus pros y contras. Fue elegido por comodidad, ya que es el que tengo instalado en mis ordenadores. Además el IDE usado **18** para el desarrollo, será de la misma compañía, por lo que hay mayor optimización para este sistema.

Android

Andorid es un sistema operativo móvil desarrollado por Google. Está basado en el kernel de Linux. Tiene casi toda la cuota de mercado de telefonía. Este se ha usado en mi *smartphone* personal y en las máquinas virtuales, con el fin de probar las aplicaciones en dispositivo físicos y virtualiazdos respectivamente.

Alternativas descartadas

Ubuntu

Ubuntu es un sistema operativo de código abierto, distribuido bajo una licencia libre. Está basado en Debian, distribución de Linux. Se desarta porque no iba a instalar otro sistema operativo en mis ordenadores personales.

macOS X

macOs sistema operativo creado por Apple, basado en Unix. Se descarta por que se quiere trabajar con dispositivos Android. Pero en el caso de que se quiera desarrollar para iOS **16**, es indispensable usarlo aquí, porque a la hora de compilar, tira de las librerías internas de este sistema operativo.

iOS

iOS sistema operativo para *smartphones* creado por Apple, basado en Unix. Se descarta por que se quiere trabajar con dispositivos Android. Pero en el caso de que se quiera desarrollar para iOS **16**, es indispensable usarlo aquí, porque a la hora de compilar tira de las librerías internas del sistema operativo.

4.2. Control de versiones

Opciones elegidas

Git

Git es un software de control de versiones, pensado para trabajar con gran cantidad de archivos, con el fin de llevar el registro de los cambios y coordinar a las personas que los comparten. Es gratuito y de código abierto.

Me he decantado por él, porque lo usé durante las prácticas curriculares y extracurriculares de forma intensa, mediante la herramienta Git Bash, ya que mediante comando me siento más cómodo que con una interfaz. Aunque también la dispone mediante el comando *gitk*.

GitHub

Github es una plataforma web, recientemente comprada por Microsoft, usada para el control de versiones con las funciones de Git. Entre las

diferentes herramientas a destacar: wiki para cada uno de los proyectos, gráficos, funcionalidades de red social, gestor de proyectos, entre otras.

Se escogió esta porque la hemos usado durante el grado y es de sobra conocida. Además ofrece la posibilidad de integración con la herramienta Zenhub [17](#), para la gestión del proyecto, teniendo las dos cosas centralizadas en el mismo lugar, lo que facilita el proceso de desarrollo.

Alternativas descartadas

Gitlab

[Gitlab](#) es igual que Github pero de código abierto ya que tiene licencia MIT. También lo usé en la empresa durante las prácticas pero fue descartado porque me parece de menor calidad. En cuanto a espacio este tiene 10 GB a favor, en contra del 1 GB de Github.

Bitbucket

[Bitbucket](#) es otra web más para el control de versiones. Esta enfocado más a la empresa privada, ya que se suele integrar muy bien con otras herramientas de gestión de proyectos. Se descarto por el poco uso que he tenido con este.

Extensiones Visual Studio Code

Hay una gran cantidad de herramientas para el control de versiones en la tienda de este editor de código, puede ser práctico, pero las interfaces pueden ser liosas. Por eso me gusta más mediante comando, descartando rápidamente esta opción.

4.3. Gestión de proyecto

Opción elegida

Zenhub

[Zenhub](#) es una herramienta de gestión de proyectos, que viene por defecto integrada en Github [16](#), lo que implica a usarla sin pensarlo mucho. En mi caso lo que más usé fue el tablero de *kanban* donde poder ver las *issues* que me planificaba para cada *sprint*. Ofrece diferentes tipos de gráficos, el que mejor se encajaba a la planificación fue de *burndown*, ya que me permite ver

lo ideal del proyecto y la progresión que llevo. No me gustó que las tareas las cierra por días en vez de por horas.

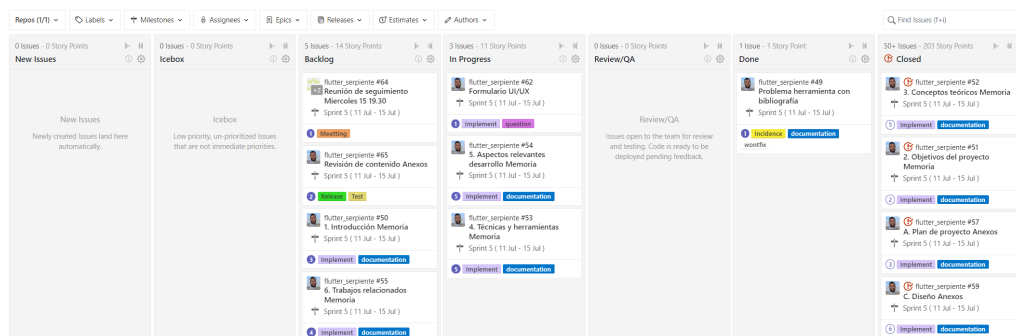


Figura 4.5: Herramienta Zenhub

Alternativas descartadas

Jira

Jira es una herramienta web propietaria, para el control, seguimiento de errores e incidencias, dentro de la gestión de proyectos. La conozco porque la usé durante las prácticas curriculares, me gustaba mucho, pero no me parece cómoda para lo que quería hacer. Ya que esta herramienta esta orientada en la mejora de procesos en la empresa, que al desarrollo de software.

Trello

Trello es una herramienta de gestión de proyectos, con interfaz móvil y web. Usa el sistema kanban como Zenhub [17](#) y tiene integración con Github [16](#), pero me parece que no encaja en proyectos unipersonales, como era mi caso. Ya que el enfoque es más colaborativo, con grandes grupos de trabajo, donde es necesario compartir documentos con los requisitos, etc ...

4.4. Entorno de desarrollo integrado (IDE)

Opción elegida

Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft, cuya licencia es MIT. Esta es totalmente gratuita, con una

cantidad enorme de extensiones o *plugins*, que minimizan esfuerzos, además de la integración con git, que permite ver los cambios en tiempo real.

Esta facilidad de uso hicieron que me decantase por él.

Alternativas descartadas

Android Studio

Android Studio es la herramienta oficial de desarrollo de aplicaciones móviles para el sistema operativo Android. La licencia es Apache 2.0. y sustituye a Eclipse **19** como el entorno de desarrollo preferido por Google.

Este IDE se descarta para la creación de código, pero no para las máquinas virtuales.

Eclipse

Eclipse es la plataforma software compuesta de varias herramientas de programación de código abierto. Se descartó por ser un IDE no recomendado por parte de Google. Además que a mi parecer es algo tosco.

4.5. Entorno de virtualización

Opción elegida

Android Studio

Android Studio 19 fue elegido para el despliegue de la aplicación en las máquinas virtuales que ofrece este IDE. Es eficiente y simple, centrado en Android. La creación de las VM es muy cómoda, ya que tiene todo integrado, (con cuatro clicks de ratón se puede lanzar una).

Sumado a que algunas herramientas de Visual Studio Code tiran de estas máquinas, hace que sea la opción más recomendable.

Alternativas descartadas

Virtual Box

Virtual Box es un software de virtualización desarrollado por Oracle. Es de sobra conocido, ya que lo hemos usado durante el grado. Puede desplegar Android, pero al no estar integrado en el desarrollo de aplicaciones, se descartó.

4.6. Herramientas de comunicación

Opciones elegidas

MS Teams

Teams es la herramienta de comunicación y colaboración integrado en el paquete ofimático de *Office* de Microsoft. Se usó esta herramienta porque con la cuenta de la universidad tenemos acceso a ella y mis tutores de proyecto fueron quienes me la propusieron.

Outlook

Outlook es una herramienta de Microsoft para la gestión de la información personal, integrado en la *suite* de Microsoft Office. Se ha usado porque la cuenta de la universidad esta integrada con la plataforma.

Alternativas descartadas

Gmail

Es un servicio de correo electrónico proporcionado por Google. Se descarta porque no tiene integración con la plataforma de la universidad de Burgos.

4.7. Documentación

Opciones elegidas

TexStudio

TexStudio es un editor de Latex de código abierto, licencia GNU, muy similar a texmaker [21](#), ya que es un *fork* de este. Lo que hizo decantarme por el fue la corrección ortográfica interactiva y resaltado de la sintaxis.

Zotero

Zotero es un gestor de referencias bibliográficas, cuya licencia es AGPL, siendo gratuito. Tiene una extensión para el navegador que ayuda mucho.

Alternativas descartadas

TexMaker

Texmaker es un IDE gratuito para escribir documentos en Latex, su licencia es GPL.

Overleaf

Overleaf herramienta web para la edición de documentos escritos en Latex. Esta opción fue descartada porque al ser web, no me permite llevar un seguimiento del versionado como lo puedo hacer, si lo tengo en local con git.

MS Word

Editor de documentos de Microsoft, muy conocido, dentro del paquete de ofimática. Se descartó porque la universidad ofrece una plantilla en Latex, de gran calidad y que este no permite el control de versiones.

4.8. Herramientas

Opción elegida

Google Chrome

Es un navegador web de código cerrado desarrollado por Google, siendo gratuito. Se ha usado este sin plantear alternativas, porque lo llevo usando desde hace muchos años. Fue usado como 'navaja suiza', ya que es una herramienta de herramientas, que ha sido usado para multitud de cosas, como:

- Para descargar programas, paquetes, librerías...
- Para la búsqueda de información.
- Edición de imágenes.
- Generador de iconos **romannurik**
- Creación de diagramas **creately**
- Armonía de color **Adobe color**

4.9. Metodologías

Scrum

Marco de trabajo para el desarrollo ágil de software, implementando una estrategia iterativa incremental, con revisiones y reuniones con los tutores de proyecto cada 5 días.

Desarrollo en cascada

Es un desarrollo en secuencia, es decir, una vez se pasa de etapa, no se puede volver atrás, se divide en *steps* muy diferenciados, que se adapta muy bien con las metodologías ágiles.

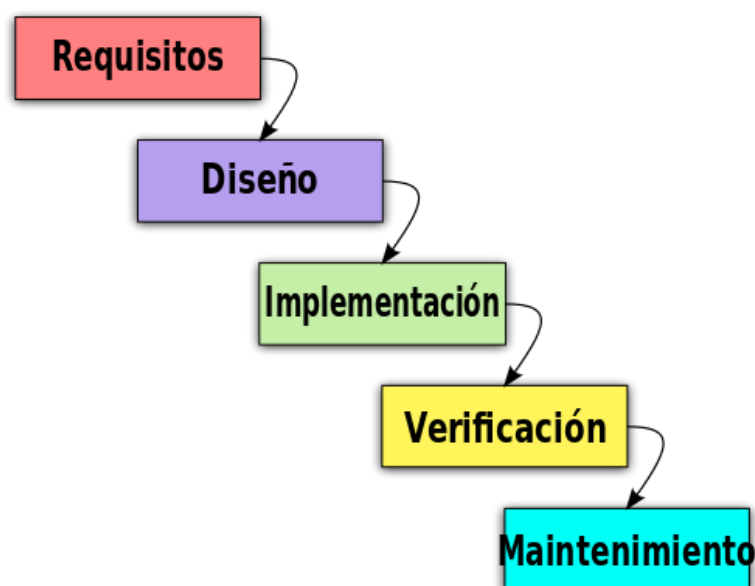


Figura 4.6: Desarrollo en cascada

Bolígrafo y cuaderno

Para mi la parte más importante de toda la gestión del proyecto, reuniones, diseño de las interfaces, control de las horas, planificación, caja negra ...

Siempre lo he tenido a mi lado como soporte, como se puede ver [4.7](#):

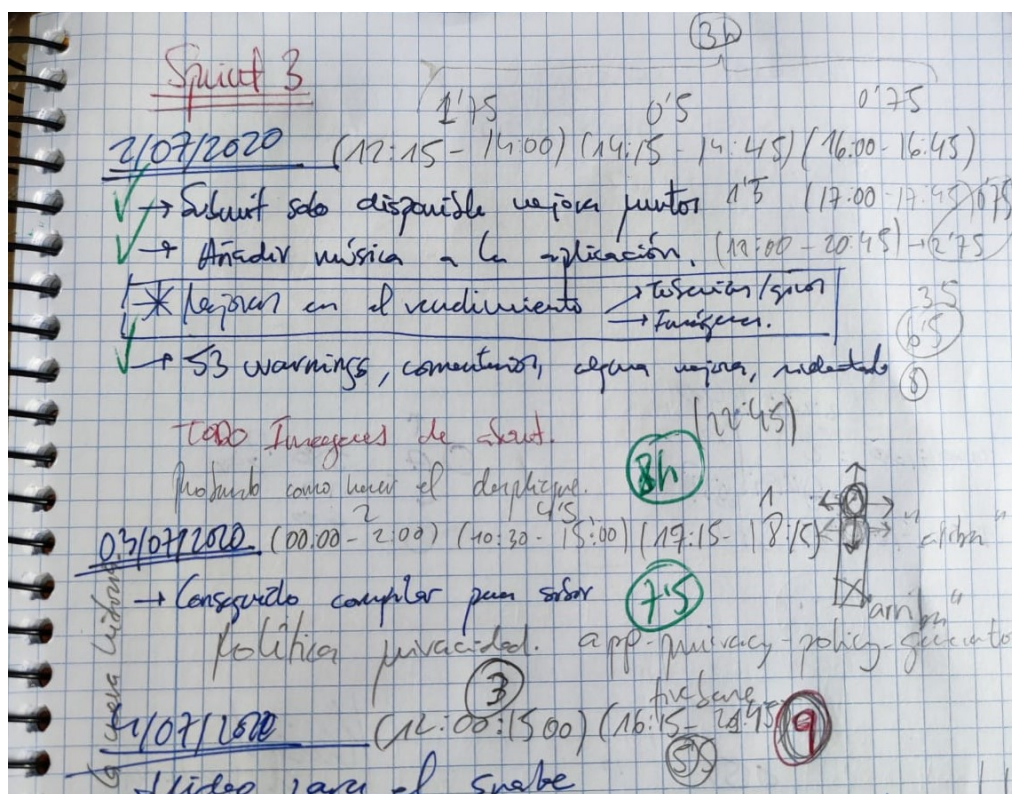


Figura 4.7: Cuaderno

4.10. Bibliotecas

Material Design

Es una normativa de diseño o guía de estilos para garantizar una homogeneidad en la visualización dentro de Android. Está desarrollada por Google.

Firestore

Plataforma para el desarrollo de aplicaciones móviles, creada por Google en el 2014. Dispone de una gran cantidad de herramientas de mucha calidad.

Play Services

Librería que permite usar a terceros todas las herramientas ofrecidas por Google.

Aspectos relevantes del desarrollo del proyecto

En este apartado se recogen los aspectos más importantes o relevantes del desarrollo del proyecto. Desde las decisiones tomadas y las implicaciones que conlleva. Además de los problemas que fueron surgiendo.

5.1. Comienzo del trabajo final de grado

La idea de mi proyecto, surge de la noche a al mañana, porque el problema es que era 20 de Julio y no tenía nada que entregar, ya que iba a hacer otro proyecto en Knime, dejándolo para otro curso.

Pero no podía dejar pasar una oportunidad. Fue más una falta de motivación y de malas sensaciones que otra cosa.

Por lo que me decidí estudiar como estaba el mercado de la programación móvil, descubriendo Flutter. Me gustó mucho, todo lo que se comentaba sobre este *Framework*, eran cosas buenas, que era algo nuevo y sobretodo lo que más me decidió a cogerlo, es el respaldo que le daba Google.

Esto se lo comenté a mis tutores de trabajo final de grado, dándome el visto bueno, pero que tenía que trabajar mucho, para llegar a la calidad esperada. Por lo que era un reto enorme al que me enfrentaba, pero como digo, no se pueden dejar pasar las oportunidades, además que por otra parte lo que más me incentivaba, es a la hora de buscar trabajo, no decir que solo me quedaba el TFG.

Ya que muchas de las empresas me decían que me esperaban a que terminase, para luego cogerme. Y yo no podía esperar, porque si no hacía el

TFG, debía de trabajar para no tener que pensar en esto. Por lo que me encontraba entre la 'espada y la pared'.

Por lo que muchas gracias a mis tutores, por haberme dejado cambiar de proyecto, por animarme a conseguirlo, por el feedback constante y darme el soporte necesario.

Finalmente hice un curso en Udemy, de unas 40 horas en dos días y medio, para conocer los elementos más básicos, ya que empezaba con los conocimientos que aporta la carrera nada más.

5.2. Flutter es asíncrono

Uno de los mayores retos cuando empiezas a trabajar con este *framework* (ya que en el curso que hice ni se tocan prácticamente), es que se trabaja de manera asíncrona.

Es decir, todo la aplicación se ejecuta en un único hilo, por lo que si un bloque de código se queda congelado, se cuelga todo. Entonces surgen las operaciones asíncronas, que permiten crear funciones o fragmentos de código que no detengan la aplicación entera. Ya que para algunas situaciones es necesario quedarse a la espera como puede ser la llegada de datos de un servidor muy lejano, por lo que no se puede dar al cliente la sensación de que la aplicación se a *freezeado*.

Dentro de Dart son los llamados Future<Object>. Básicamente son tareas que se quedan a la espera, con el fin de que se las devuelva el objeto pedido. Un ejemplo de esto lo podemos ver en [5.8](#).

```
///Método para guardar/actualizar los campos del usuario
Future<void> _guardarBD() async {
  // Dentro de guardar mostrar
  print("Dentro de guardar");
  // Cogemos el documento por el correo del user, en el caso de que no
  DocumentReference doc =
    firestoreDB.collection("ranking").document(_correoForm);
  Map<String, dynamic> data = {
    "imageUrl": imageUrlGoogle,
    "nombre": _nombreForm,
    "puntos": _puntosForm,
    "fecha": FieldValue.serverTimestamp() //Guarda la fecha del server
  };

  try {
    /*
     * Crea el documento en el caso de que no este en la base de datos
     * con los datos anteriores, de data.
     */
    doc.setData(data);
  } catch (err) {
    print("El error del update es: $err");
  }
}
```

Figura 5.8: Ejemplo de Future

Para ello tenemos que declarar las funciones que son asíncronas con la palabra reservada *async*, de tal manera que devolverá los resultados en un tiempo x.

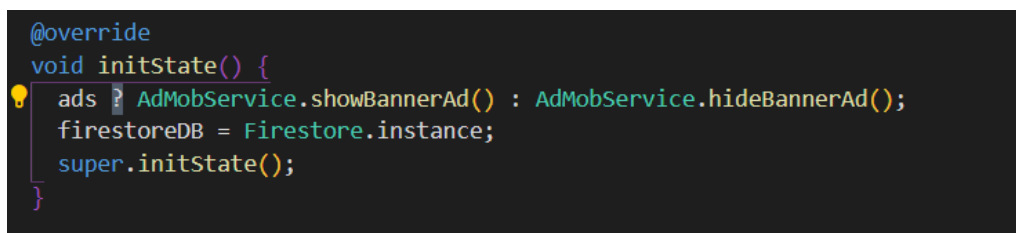
Cuando realicemos la llamada a estas indicaremos que se queden a la espera de respuesta con *await*.

Pero podemos encontrarnos con otros objetos que también manejan tareas futuras que son los Streams. Haciendo que la aplicación de flutter sea reactiva, como es el caso del cuatro en raya o el ranking.

Estos se definen en un lugar, cogen datos de otro lugar y se queda a la escucha de que se produzcan cambios. En el caso de disparen esos cambios, el Stream se encargará de rediseñar la interfaz del usuario o de actualizar la variables, según el caso.

5.3. Nuevas formas de hacer if

Desconocía totalmente esta nueva forma de hacer los condicionales, ya que deja un código más limpio, que ocupa menos líneas e intuitivo. Un ejemplo de esto se puede ver en la imagen siguiente 5.9:



```
@override
void initState() {
  ads ? AdMobService.showBannerAd() : AdMobService.hideBannerAd();
  firestoreDB = Firestore.instance;
  super.initState();
}
```

Figura 5.9: Nuevas formas if

5.4. Play Store

Uno de los problemas encontrados fue a la hora de tener que generar las claves de seguridad, para que la aplicación se pueda subir a la tienda y que ciertos de los servicios estén operativos.

La incidencia estaba que no estaba generando una clave para *debug* y no para *release*, imposibilitando la subida.

Esto era debido a que en la documentación de Flutter siempre se realiza para el modo de test. Luego otra de las cosas a vigilar, es la ruta en la que dejamos la clave y la configuración de algunos ficheros, como los *.gradle* o el *manifest*.

Es muy importante tener la clave bien guardada porque en el caso de que se pierda, vamos a tener que volver a realizar el proceso desde cero y es algo tedioso.

5.5. AdMob

Otro de los problemas que he tenido es que a la hora de mostrar los anuncios en la aplicación, no me deja usar los que no son test. He revisado cada uno de los Ids 5.10, pero nada.


Configuración del bloque de anuncios	
Nombre del bloque de anuncios ?	Banner Bottom Ads
Formato del anuncio ?	 Banner
ID del bloque de anuncios ?	ca-app-pub-7462396340145780/9142410233

Figura 5.10: Banner

Y las métricas de usuario si que funcionan 5.11, por lo que la instancia de adMob es la correcta.



Figura 5.11: Métricas usuarios

Tras investigar y realizar gran cantidad de pruebas creo que el problema puede estar en que no tengo métodos de cobro de ingresos. Por lo tanto la propia Google ni se molesta en poner anuncios. Pero es que la propia consola web de admob, no me deja introducir un método de pago, como se puede ver en la imagen 5.12:

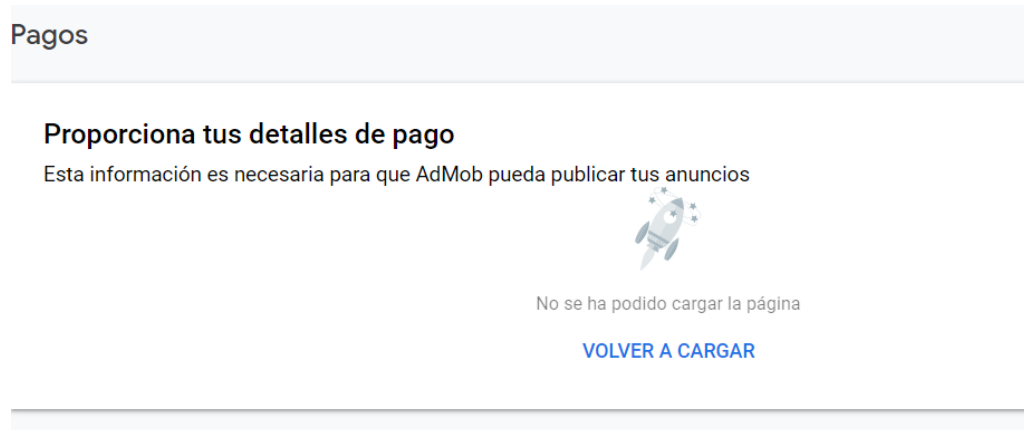


Figura 5.12: Pagos adMob no van

Trabajos relacionados

Al tratarse de una herramienta muy joven, salió en el 2018, por lo que lleva apenas dos años en el mercado. Cada vez empieza a ver más documentación al respecto, cursos y foros en los que se tratan los temas. Entiendo que esto se debe a que Google esta apostando fuerte por el lenguaje.

No me he basado en trabajo anteriores, pero si que he buscado mucha información y me he documentado en gran cantidad de sitios, algunos de los cuales son :

- Flutter sdk , donde está toda la documentación oficial [web](#).
- Canal oficial de [flutter](#) en youtube, donde se comenta el Widget de la semana.
- Todo lo referente a [dart](#).
- Canal de youtube sobre [firebase](#).

Conclusiones y Líneas de trabajo futuras

En este apéndice se expone las conclusiones tras la finalización del proyecto final de grado, así como todas las posibles ideas o líneas de trabajo futuras, con el fin de que el proyecto mantenga una continuidad.

7.1. Conclusiones

- El objetivo general del proyecto se ha cumplido, ya que todos los requisitos funcionales, como los no funcionales, fueron realizados de manera satisfactoria. Aunque es revisable la mejora de ciertos aspectos, como el diseño de algunas ventanas de la aplicación.
- Todo el proceso que engloba crear una aplicación, desde la adquisición de los conocimientos, el desarrollo, el diseño, revisiones de producto, la planificación, hasta el despliegue final.

Indica que se han tocado la mayoría de los conocimientos adquiridos durante el grado, pero que aún así es necesario el constante avance formativo.

- Adquirir los conocimientos necesarios para la creación de una aplicación en Flutter, ya que es un SDK de gran versatilidad, permitiendo hacer aplicaciones en un tiempo menor.

Aprender también las herramientas que ofrecidas por parte de Google como Firebase o adMob. *Tools* que desconocía, pero que dan gran valor añadido al producto final.

- Durante todo el proyecto se usaron gran variedad de aplicaciones, herramientas o dispositivos que ayudaron a mejorar la calidad, rendimiento y funcionalidad del producto final o de algunos de los procesos intermedios. Esto implica tener que especializarse en cada una de ellas, lo que consume recursos temporales.

Pero a la larga este conocimiento aprendido ayudará a tener mejores productos en el futuro.

- Lidar con gran incertidumbre, esto se debe que a la hora de planificar es difícil estimar correctamente las horas. Lo que en próximos proyectos puede ser una complicación. Por lo que es una cosa que tengo que mejorar.

Es de gran importancia ajustarse lo máximo posible a la realidad y no pecar de pesimista como es mi caso, ya que planifico más horas de las que luego realmente se invierten.

- La investigación como parte fundamental a la hora de añadir nuevas funcionalidades al producto como el *sign in* de la aplicación o la publicidad, entre otras.

En definitiva, personalmente estoy muy contento de haber trabajado duro y haber conseguido una aplicación, ya que desconocía la programación en Android, lo que me ha hecho crecer como persona, pero sobretodo en el ámbito profesional.

7.2. Líneas de trabajo futuras

- La idea es que la aplicación retenga el mayor tiempo posible a los usuarios para publicidad. Para ello la biblioteca de juegos tiene que seguir creciendo.

Estos nuevos juegos, se deben de planificar en base a la publicidad para sacar el mayor retorno económico.

- Internacionalización de la aplicación, para estar disponible en un mayor número de países, de tal manera que el mercado que abarque sea más amplio.
- Integración de test automáticos como de nuevas metodologías de desarrollo dirigido por test (TDD). Esto mejoraría sustancialmente el rendimiento del programador.

- Estudio de nuevas herramientas o frameworks destinados a la creación de videojuegos. De tal forma que se pueda migrar los dos juegos actuales, con el fin de ganar un mayor rendimiento.
- Añadir una inteligencia artificial al juego del cuatro en raya, de tal manera que no solo se pueda jugar online, si no contra la CPU. Como idea esta podría integrar tres niveles de complejidad: fácil, medio y experto. Dentro de la propia Firebase de Google tenemos herramientas de *Cloud computing*, como el *ML kit (Machine learning kit)*.

Este kit tiene herramientas muy conocidas de Google como puede ser el reconocimiento de texto, etiquetado de las imágenes o detección de las caras, pero se pueden añadir nuestro propios modelos.

- Realizar el despliegue de la aplicación en la tienda de Apple *App Store*, para dispositivos iOS. Y la creación de un hosting donde podamos alojar la aplicación web.
- Integración y exploración de las herramientas de videojuegos como *Play juegos* de Google y la de Apple con su *Game Center*. Esto podría permitirnos las compras in-app.
- Implementación de algún juego de realidad aumentada con la integración de la publicidad también.

Bibliografía
