**DISEÑO E IMPLEMENTACIÓN DE UNA APP DE GESTIÓN DE ACTIVIDADES DEPORTIVAS**

**José Martín Sánchez Jordán**

**Tutor: Francisco José Martínez Zaldívar**

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingeniería de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2021-22

Valencia, 8 de marzo de 2022

Resumen

En la actualidad, las condiciones sanitarias y tecnológicas han propiciado un auge en el sector del soporte tecnológico, debido a la necesidad de realizar tareas comúnmente presenciales de forma telemática.

El objetivo de este trabajo es simular este escenario, desarrollando una aplicación de gestiones deportivas destinada a profesionales del sector del entrenamiento personal que no tengan la capacidad económica o los medios para hacer llegar la actividad deportiva a sus usuarios a través de una plataforma online.

Realizar una aplicación de este tipo supone una oportunidad para aprender tecnologías modernas, por lo que para el desarrollo se ha utilizado Flutter**,** *framework* del lenguaje de programación Dart que permite desarrollar una App tanto para iOS como Android con un único código entre otras muchas ventajas. Tanto el servicio de autenticación como de almacenamiento de datos se ha implementado con Firebase. Estas tecnologías han sido desarrolladas en los últimos años por Google, poniéndose a la cabeza del desarrollo de aplicaciones multiplataforma en un contexto en el que el acceso instantáneo a cualquier servicio e información es primordial para el usuario.

Resum

En la actualidad, las condiciones sanitarias y tecnológicas han propiciado un auge en el sector del soporte tecnológico en ámbitos

Abstract

The memory of the TFG begins with a short abstract from 150 to 200 words, writen in Spanish, Valencian and English. These pages are not numbered.

Listado de Tablas

[Tabla 1. Ejemplo de tabla. 3](#_Toc97647495)

Listado de Figuras

[Figura 1: Extensiones de Visual Studio Code utilizadas. 4](#_Toc97673828)

[Figura 2: ‘DartPad’, medio de aprendizaje. 4](#_Toc97673829)

[Figura 3: Funcionamiento de flutter 5](#_Toc97673830)

[Figura 4: Estructuración con widgets 5](#_Toc97673831)

[Figura 5: Distintos sistemas operativos 6](#_Toc97673832)

[Figura 6: Añadir variable de entorno de Flutter 6](#_Toc97673833)

[Figura 7: Verificación instalación Flutter 7](#_Toc97673834)

[Figura 8: Creación del workspace 7](#_Toc97673835)

[Figura 9: Workspace generado por Flutter 8](#_Toc97673836)

[Figura 10: Simulador de Android 8](#_Toc97673837)

[Figura 11: Dispositivo seleccionado 8](#_Toc97673838)

[Figura 12: Ejecución de la aplicación 9](#_Toc97673839)

[Figura 13: Primeros pasos con Firebase 10](#_Toc97673840)

[Figura 14: Registro app. 10](#_Toc97673841)

[Figura 15: google-services. 11](#_Toc97673842)

[Figura 16: Dependencias de Google-services. 11](#_Toc97673843)

[Figura 17: Comando pub add de Flutter 11](#_Toc97673844)

Índice

[Capítulo 1. Introducción 2](#_Toc97673607)

[1.1 Motivación 2](#_Toc97673608)

[1.2 Objetivos 2](#_Toc97673609)

[1.3 Estructura de la memoria 3](#_Toc97673610)

[Capítulo 2. Tecnologías utilizadas 4](#_Toc97673611)

[2.1 Visual Studio Code 4](#_Toc97673612)

[2.2 Dart 4](#_Toc97673613)

[2.3 Flutter 5](#_Toc97673614)

[2.3.1 ¿Qué es Flutter? 5](#_Toc97673615)

[2.3.2 Instalación y configuración del proyecto 6](#_Toc97673616)

[2.4 Firebase 9](#_Toc97673617)

[2.5 Bibliografía 12](#_Toc97673618)

[Capítulo 3. Apartados de la Memoria 14](#_Toc97673619)

# Introducción

## Motivación

En los últimos años, se ha visto un aumento exponencial del uso de dispositivos móviles (smartphones, tablets etc) en la población en general. El acceso a información, noticias o cualquier tipo de servicio de forma inmediata y desde cualquier lugar es cada vez más imprescindible y demandado por la sociedad.

Si a esta premisa le añadimos el factor de la actual situación pandémica global, nos encontramos en un contexto en el que gran parte de los sectores laborales que no estaban tecnológicamente adaptados a los requisitos actuales, se hayan visto obligados a renovar su *modus operandi*. Por esta razón, se ha optado por plantear un escenario en el que, por ejemplo, un entrenador personal que antes realizaba los entrenamientos presencialmente con su cliente y ahora requiere un medio moderno, rápido e intuitivo para gestionar, planificar y controlar los entrenamientos de forma telemática.

He aprovechado esta oportunidad para aprender un nuevo lenguaje de programación: Dart, y su *framework* Flutter, una novedosa tecnología enfocada a la programación multiplataforma, con un gran desarrollo por parte de Google. Además, he ampliado los conocimientos ya adquiridos en la carrera sobre Firebase (también desarrollado por Google) para el almacenamiento de datos, ya que al estar desarrollados por la misma compañía tienen una gran compatibilidad.

## Objetivos

Los objetivos propuestos inicialmente son los siguientes:

* Crear una aplicación móvil multiplataforma
* Emprender el desarrollo de una aplicación con vistas a ser comercializada
* Ampliar mis habilidades como desarrollador de aplicaciones
* Adquirir más conocimientos sobre aplicaciones para dispositivos móviles
* Enfrentarme a un proyecto desde cero y totalmente creado por mi
* Aprender un lenguaje de programación nuevo con vistas al futuro laboral
* Conocer nuevas tecnologías punteras en el sector del desarrollo de software
* Mejorar en el aspecto de diseño y estructura de interfaz gráfica

## Estructura de la memoria

Para la confección de este documento, he planteado los siguientes capítulos, pensados a su vez en un orden coherente al transcurrido durante el desarrollo del proyecto:

* **Capítulo 1**: Pensado para introducir al lector en el contexto social en el que nos encontramos y por ende la motivación que me ha llevado a realizar este proyecto, asumiendo el aprendizaje de nuevas tecnologías muy interesantes con vistas al futuro laboral o incluso emprendedor mejorando esta aplicación.
* **Capítulo 2**: Profundizaré en los aspectos técnicos y funcionales de las distintas tecnologías que he utilizado para el desarrollo de la aplicación añadiendo además la parte de configuración del *workspace* así como herramientas utilizadas para mejorar el proceso de programación.
* **Capítulo 3:** Analizaré la metodología que me ha ayudado a terminar el proyecto dentro de unos plazos y la importancia de una organización de tareas para ser más productivo.
* **Capítulo 4:** En este apartado se trata el análisis del proyecto, tratando cuales han sido los requisitos iniciales para la aplicación, roles (actores) pensados y los casos de uso en la utilización de la aplicación. Otros puntos para tratar serán como se ha tratado *back end* a través de diagramas de flujo y del *front end* haciendo uso además de capturas de la propia aplicación.
* **Capítulo 5:** Adjuntaré los resultados de las pruebas realizadas una vez terminado el proceso de desarrollo.
* **Capítulo 6:** Reflexionaré sobre las conclusiones acerca de las tecnologías utilizadas para el desarrollo de la aplicación y el margen de mejora y ampliación de esta.

# Tecnologías utilizadas

## Visual Studio Code

Como entorno de desarrollo, he optado por Visual Studio Code frente a otras alternativas contempladas ya que, pese a no estar familiarizado con este editor, me parecía un buen momento para hacer aprender a utilizarlo.

Otro motivo por el cual lo he elegido para este proyecto es debido a la implementación sencilla de una serie de extensiones, adjuntadas en la figura 1, como las propias de Dart y Flutter que permiten una serie de auto rellenados de código, visualización de acabados o requisitos de *widgets* de una forma intuitiva. Las extensiones restantes son meramente estéticas, pero ayudan a realizar a tareas de formateo de código y organización de *workspace*.

Texto

Descripción generada automáticamente

Figura 1: Extensiones de Visual Studio Code utilizadas.

## Dart

Llamado originalmente Dash, es un lenguaje de programación open source desarrollado por Google desde 2011, apostando por una alternativa al gigante JavaScript.

La decisión de optar por Dart realmente ha sido por conveniencia, ya que Flutter es un *framework* de Dart, aunque he decir que me ha gustado mucho ya que, si estás iniciado en Java (como es el caso gracias al grado) es prácticamente igual, pero con atajos por lo que es muy fácil adaptarse.

Además, Dart cuenta con una gran documentación, así como de un entorno web de prácticas tal y como se muestra en la figura 2, el cual ha sido mi medio de aprendizaje previo a introducirme en el maravilloso mundo de Flutter.

Captura de pantalla de un celular

Descripción generada automáticamente

Figura 2: ‘DartPad’, medio de aprendizaje.

## Flutter

### ¿Qué es Flutter?

Flutter es una “herramienta” que permite crear aplicaciones multiplataforma (iOS, Android) con un único código base, lo cual es su principal ventaja respecto de, por ejemplo, Android Studio donde debes saber Java o Kotlin para desarrollar una aplicación que no te servirá para iOS.

¿Cómo puede hacerlo? Flutter realmente es la suma de dos componentes, un *SDK (Software Development Kit)* con las herramientas que permiten que el código sea compilado en ambas plataformas y un *framework* o librería de *widgets* (bloques) que nos brinda un abanico de opciones, desde utilizar los propios *widgets* o funciones, customizarlos o crear los tuyos propios.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Figura 3: Funcionamiento de flutter

La forma que tiene Flutter de implementar estos widgets es muy sencilla e intuitiva, especialmente para programadores con problemas para el *frontend* como es mi caso, siendo el código compilado y mostrando los errores como si fuera parte del *backend*, lo que para mi es una gran característica de cara al *debug*.

En Flutter todo es un *widget*, desde un simple botón hasta el ‘*Container*’ o contenedor en el que se añaden otros *widgets*. En la figura 4 se encuentran los principales tipos de *widgets* que se utilizan para estructurar, como *Column()* (columna) o *Row()* (fila) donde se añadirán otros *widgets* con funciones específicas, como el citado botón, una imagen o un texto.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 4: Estructuración con widgets

Uno de los principales potenciales de Flutter, y por lo que cada vez es una opción más utilizada por programadores es que, compilado el código, este no tiene ningún tipo de contacto con el sistema operativo del dispositivo, sino que ‘pinta’ los pixeles de la pantalla, es decir, tiene su propio motor que controla toda la pantalla. Esta funcionalidad es la que permite que con un único código puedas implementar una aplicación tanto para iOS como Android, además de convertir Flutter en una gran decisión en términos de rendimiento debido a la gran velocidad para renderizar *layouts*.

En los siguientes capítulos profundizaré más sobre la forma de programar de Flutter, a través de casos concretos de mi proyecto. Por ahora, nos centraremos en el proceso de integración con el *workspace* para comenzar a trabajar.

### Instalación y configuración del proyecto

Comenzar a trabajar con Flutter es muy sencillo, en primer lugar, debemos instalar Flutter desde su página web. En la figura 4 vemos que Flutter ofrece distintas versiones para el sistema operativo de nuestro equipo de trabajo.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 5: Distintos sistemas operativos

Aunque parece un punto sin importancia, hay que tener en cuenta que, si optamos por utilizar por ejemplo un Mac, no podremos realizar desarrollos para Android debido a la limitación del propio sistema operativo. En mi caso he optado por Windows.

El proceso de instalación no es relevante para este trabajo, pero sí cabe mencionar que para que todo funcione, se debe añadir la clásica variable de entorno en nuestro equipo, en el caso de Windows vemos en la figura 3 cómo es agregada junto al *Path* del directorio correspondiente donde se haya realizado la instalación:

Texto

Descripción generada automáticamente con confianza baja

Figura 6: Añadir variable de entorno de Flutter

Para verificar que la instalación se ha llevado a cabo correctamente, basta con ejecutar *$flutter* desde cualquier directorio. En caso de que, como en la figura 7, no muestre los posibles comandos sino un mensaje de error significará que algo no ha sido configurado de forma apropiada. Otro comando más específico es *$flutter doctor¸* que nos especifica por qué no se está llevando a cabo la instalación.

Texto

Descripción generada automáticamente

Figura 7: Verificación instalación Flutter

Resaltar que en la figura 7 si bien se ve un mensaje de error, es debido a no tener la versión de Visual Studio con C++ lo cual no interfiere en la realización del proyecto y se puede continuar sin ningún problema.

Ahora sí, toca comenzar un proyecto nuevo, para ello simplemente debemos ejecutar el comando *$flutter create* *nombre\_proyecto* situándonos en nuestro *workspace* o directo de trabajo

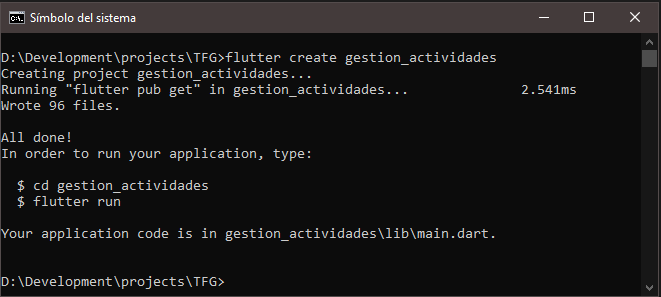


Figura 8: Creación del workspace

Como se observa en la figura 8, se ha creado una subcarpeta dentro del directorio “TFG” con el nombre del proyecto. Dentro Flutter ya se ha encargado de configurar todo. Si abrimos esta carpeta creada desde Visual Studio Code, vemos el proyecto debe quedar como en la figura 9.

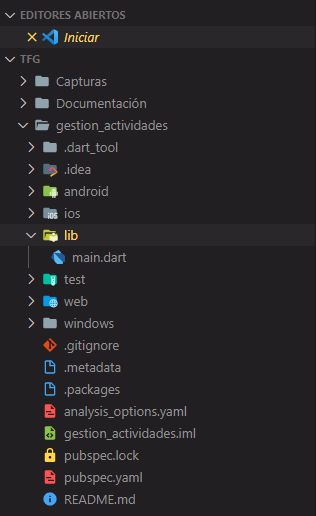


Figura 9: Workspace generado por Flutter

El fichero ‘*main.dart’*  es generado de forma automática para que veamos un ejemplo de como debe escribirse una aplicación funcional en la que puedes pulsar un botón y un contador se va incrementando. Este punto es perfecto para introducir el medio en el que vamos a emular a través de un simulador nuestra aplicación.

Si bien Visual Studio Code ofrece varias alternativas para arrancar nuestra aplicación desde el mismo navegador web o incluso desde nuestro propio dispositivo móvil, yo he optado por hacer uso de los simuladores de Android Studio. Simplemente debemos instalar este IDE y sin ni siquiera arrancarlo, podemos elegir entre varios tipos de dispositivos, resoluciones y versiones de Android a nuestro gusto desde el AVD Manager, como en la figura10.

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Figura 10: Simulador de Android

Para que el simulador arranque debemos pulsar el botón de play verde mostrado en la figura 11.

Captura de pantalla de un celular

Descripción generada automáticamente

Figura 11: Dispositivo seleccionado

Con el simulador de Android arrancado, basta con ejecutar el comando *$flutter run*. Aquí tienes la opción de hacerlo desde el CMD o desde el propio Visual Studio Code, el cual viene con un terminal integrado, mostrado en la figura 12, que facilita esta tarea.

Texto

Descripción generada automáticamente

Figura 12: Ejecución de la aplicación

Como respuesta, además de arrancar la aplicación, Flutter proporciona unas funciones asignadas a unas teclas, para limpiar la pantalla (c), quitar la aplicación (q), o terminar el proceso de simulación (d). Sin duda la más utilizada es (r) *‘Hot reload’,* que permite ahorrarnos el proceso de compilación y que se “carguen” solamente los últimos cambios aplicados. Aunque realmente no hace falta utilizar el terminal para esto ya que, por defecto, cuando salvas un fichero Dart formatea todo el código y además si el simulador está arrancado, ejecuta esa instrucción automáticamente.

Texto

Descripción generada automáticamente

Figura 13: Aplicación demo de Flutter

## Firebase

Firebase no es un servicio de almacenamiento al uso, aporta un sistema de autenticación compatible con plataformas externas como Facebook o Twitter, actualización de registros en tiempo real, visualización de los accesos, análisis y estadísticas sobre los usuarios, clicks en anuncios o el uso de la aplicación. Todo ello gestionado y almacenado en servidores de Google que nos aportan una seguridad añadida y una rapidez para configurar el entorno de trabajo.

Firebase es un servicio gratuito, pero a partir de un límite de accesos y operaciones (bastante amplio para el caso que nos atañe en este proyecto) el servicio pasa a ser de pago por lo que debe ser revisado con cuidado en caso de tener una aplicación a gran escala.

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 14: Primeros pasos con Firebase

Adentrarse en Firebase (figura 3) y sus ventajas es sencillo a través de su página web, siguiendo sus pasos es fácil iniciarse en los conceptos. Además, Firebase cuenta con una documentación muy extensa sobre como integrar este servicio en tu aplicación, sea app o aplicación web y para todo tipo de lenguajes de programación, incluido Flutter con el que comparte un manual específico para utilizar ambas tecnologías. La integración en mi proyecto se ha realizado de la siguiente forma:

1. Registrar la app con un nombre del paquete de Android como se muestra en la figura 4.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 15: Registro app.

1. Tal y como se indica en la figura 5, se añadirá el fichero *google-services.json* al proyecto.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 16: google-services.

\* Como nota, comprobar que en las dependencias se haya añadido correctamente junto a la versión propia.

Texto

Descripción generada automáticamente

Figura 17: Dependencias de Google-services.

1. Por último, para hacer uso de cualquiera de las funciones de Firebase, se debe hacer uso de sus distintas librerías. La instalación de versión correcta de estas es un proceso que normalmente tedioso y da quebraderos de cabeza en otros entornos, Flutter sin embargo lo soluciona con un simple comando en el terminal, ejemplificado a través de la instalación de unas de las distintas librerías de Firebase que he utilizado en mi proyecto en la figura 7.

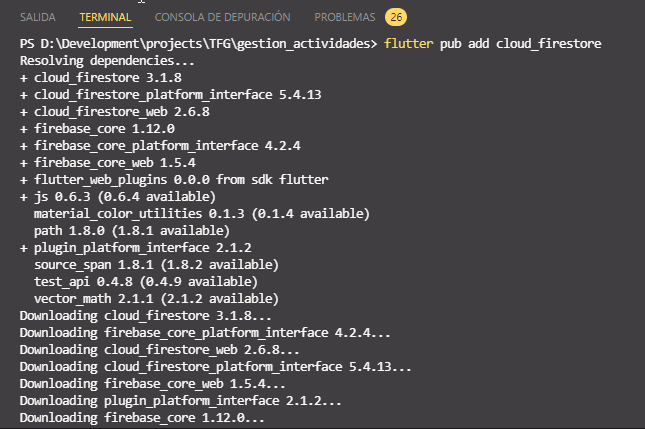


Figura 18: Comando pub add de Flutter

## Bibliografía

La Memoria debe incluir una relación bibliográfica de las fuentes consultadas. Las referencias bibliográficas siempre deberán estar convenientemente citadas en el texto de la memoria. Se deberá dedicar una sección de la misma para listarlas con la siguiente notación:

[1] Wilkins, R.; Little, J. “Solution of electrostatic field problems using Sander's function,” *IEEE Transactions on Microwave Theory and Techniques,* vol. 55, no. 9, pp. 1880–1886, September 2007.

[2] Ansoft Corporation, “HFSS: 3D high-frequency electromagnetic simulation,” http://www.ansoft.com/products/hf/hfss/index.cfm. [Online].

[3] Matthaei, G.; Young, L. and Jones, E. M. T. *Microwave Filters, Impedance-Matching Networks, and Coupling Structures*. Artech House Inc., 1980.