

UNIVERSIDAD POLITÉCNICA DE MADRID

E.T.S. DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

PROYECTO FIN DE MÁSTER

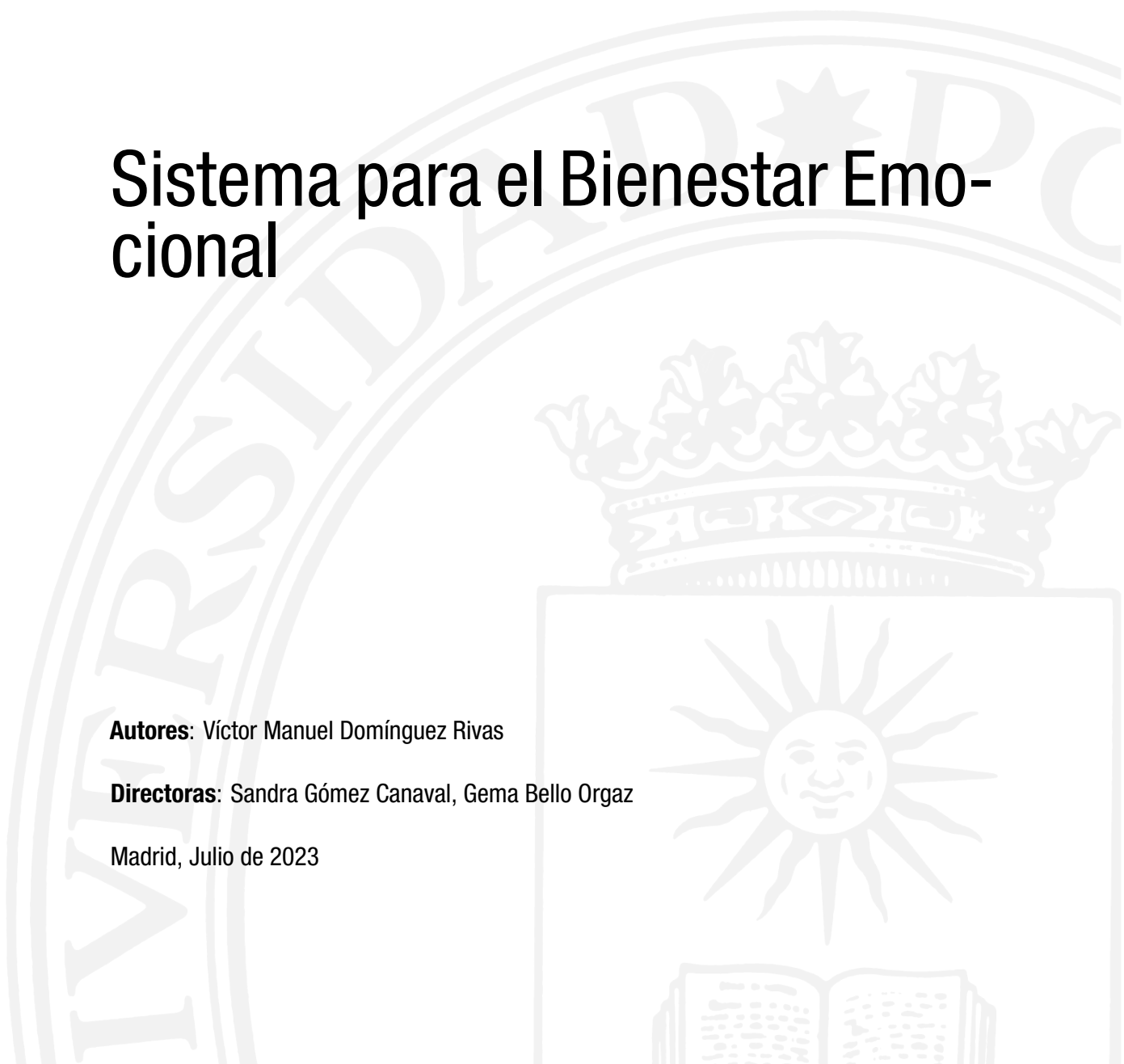
**MÁSTER UNIVERSITARIO EN SOFTWARE DE SISTEMAS DISTRIBUIDOS Y
EMPOTRADOS**

Sistema para el Bienestar Emocional

Autores: Víctor Manuel Domínguez Rivas

Directoras: Sandra Gómez Canaval, Gema Bello Orgaz

Madrid, Julio de 2023



Víctor Manuel Domínguez Rivas

Sistema para el Bienestar Emocional

Proyecto Fin de Máster, 28 de junio de 2023

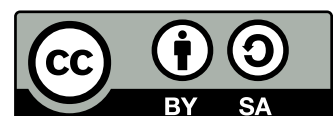
Directoras: Sandra Gómez Canaval, Gema Bello Orgaz

E.T.S. de Ingeniería de Sistemas Informáticos

Campus Sur UPM, Carretera de Valencia (A-3), km. 7

28031, Madrid, España

Esta obra está bajo una licencia **Creative Commons «Atribución-CompartirIgual 4.0 Internacional»**.



Resumen

Palabras clave: Salud Mental, Android, Wearables, Jetpack Compose, Health Connect

Abstract

Keywords: Mental Health, Android, Wearables, Jetpack Compose, Health Connect

Agradecimientos

Este proyecto es el final de un camino de siete años en la escuela, el broche de una etapa tan importante. De alguna manera estos casi once meses han sido la síntesis de un largo viaje, tanto a nivel técnico como humano. Y como en todo viaje que se precie, lo mejor no es necesariamente el destino, sino llegar hasta allí.

No habría sido posible llegar hasta aquí sin la ayuda de mis amigos, por su apoyo en los momentos más delicados y sobre todo por hacer disfrutar el camino partiendo desde la más absoluta nada. Nunca nos hemos conformado con nada, siempre hemos empujado más allá y eso lo ha hecho todo mucho más especial y profundo. Todos han tenido su contribución y la conocen perfectamente, pero por hacer esta memoria finita; quiero acordarme especialmente de Juan Luis y Rocío por haber estado juntos en todo momento, habernos complementado en tantísimos niveles y siendo poco menos que una familia.

A mi familia por haber apoyado en el día a día desde la distancia, a pesar de vivir una etapa tan complicada. En los momentos donde todo cambia, en los que tienes que decidir sobre tantas consecuencias; es donde aparece la calidad humana de las personas.

Obviamente esto tampoco sería posible sin mis tutoras. A Sandra por todo su apoyo, tanto profesional como especialmente personal. A pesar de ser un año cuanto menos complicado para todos ha estado ahí, intentando sacar lo mejor de cada momento y situación. Por supuesto también a Gema por su enorme labor a nivel técnico, ayudándome a decidir sobre tantas y tantas cosas, aportando consejos sobre cómo mejorar... y sobre todo por su faceta personal, por estar ahí y entender situaciones delicadas y también por aportar la ilusión en el día a día. Sin pasión no hay vida.

Por supuesto a todas las personas que han colaborado con su granito de arena. A Cristina y Miriam por su increíble labor en los consejos y cuestionarios. de la aplicación. También a LevelUp por ayudarme con la identidad gráfica del proyecto; sin ellos el proyecto no tendría alma.

También quiero agradecer a todas las personas que se dedican a la investigación y a la ciencia. Una sociedad que no cuida a sus científicos e investigadores está condenada a la decadencia, ya que sin ellos no sería posible el progreso.

Asimismo quiero agradecer humildemente el trabajo de personas como Aaron Swartz y Alexandra Elbakyan, e iniciativas como la Open Access por facilitar el acceso a la ciencia [1]. El acceso libre y gratuito al conocimiento científico es vital para el desarrollo de cualquier sociedad democrática. Una sociedad que lo promueve es una más formada y menos manipulable; algo vital en nuestros tiempos.

Índice general

Índice general	i
Índice de figuras	iii
Índice de cuadros	iv
1 Introducción	1
1.1 Contexto	1
1.2 Motivación	1
1.3 Objetivos	1
1.4 Justificación	2
1.5 Estructura del documento	2
2 Marco teórico y contexto tecnológico	3
2.1 Marco Teórico	3
2.2 Contexto tecnológico	3
3 Estado de la cuestión	23
3.1 Análisis de la situación actual	23
4 Metodología y materiales	24
4.1 Metodologías de desarrollo del <i>software</i> de Sistemas	24
5 Diseño del sistema propuesto	25
5.1 Descripción del caso de estudio	25
5.2 Diseño del sistema	25
6 Desarrollo del sistema	26
6.1 Project Blastoff	26

6.2	Setup del proyecto	37
6.3	Implementación de la aplicación móvil	37
6.4	Implementación de la API del servidor	37
7	Resultados	38
7.1	Preparación del entorno	38
7.2	Resultados del caso de estudio	38
8	Impacto social y medioambiental	39
8.1	Contexto medioambiental	39
8.2	Aspectos éticos, sociales y económicos	39
9	Presupuesto	40
10	Conclusiones	41
11	Líneas futuras	42
12	Temporal	43
12.1	Setup del proyecto	43
12.2	Sprint #0	43
12.3	Desarrollo	44
	Bibliografía	45
A	Código fuente del proyecto	50
A.1	Aplicación móvil	50
A.2	API del servidor	52

Índice de figuras

2.1	Extracto de los documentos alojados en Teams	4
2.2	Extracto del registro de horas	5
2.3	Logo actual de Android.	7
2.4	Capas de Android.	8
2.5	HTC Dream en funcionamiento.	9
2.6	Estadísticas acumulativas de las versiones de Android	10
2.7	Logo de Kotlin.	11
2.8	Ejemplo de uso de la herramienta <i>Material Design Builder</i>	13
2.9	Categorías de pantalla según anchura	14
2.10	Algunos elementos gráficos de Material Design 3	15
2.11	Logo de Salud Conectada.	15
2.12	Arquitectura básica de Salud Conectada.	17
2.13	Visualización de datos de distancia.	18
2.14	Visualización del acceso a los datos.	18
2.15	Granularidad de los permisos.	19
2.16	Logo de Python.	20
2.17	Logo de MongoDB.	21
6.1	Componentes básicos del sistema	31
6.2	Pila con la arquitectura del sistema	32
6.3	Dimensionamiento aproximado del proyecto	34

Índice de cuadros

6.1	Lista de NOes del proyecto	30
6.2	Riesgos del proyecto	33
6.3	Cuestiones y prioridades del proyecto	35

1.

Introducción

Si he logrado ver más lejos ha sido porque he subido a hombros de gigantes.

Isaac Newton

1.1 Contexto

1.2 Motivación

1.3 Objetivos

Objetivo general

Objetivos específicos

El objetivo de un [Proyecto Fin de Grado \(PFG\)](#), [Proyecto Fin de Máster \(PFM\)](#) y [Tesis Doctoral \(TD\)](#) es una de las piezas clave a plantear, y a su vez una de las más complicadas. Se considera la **finalidad** del proyecto en cuestión a realizar y suele encajar dentro de una de las siguientes categorías:

- **Contraste** o validación de una hipótesis. Este es típico de [TDs](#), aunque algunos [PFMs](#) y (muy raramente) [PFGs](#) pueden caer dentro de esta categoría.
- **Desarrollo** o diseño de algo (e.g. Software, hardware, sistema, edificio). Suele ser el más común en la rama de la ingeniería, tanto [PFMs](#) como [PFGs](#).
- **Estudio** de un tema que deduce o descubre nuevo conocimiento. Éste suele ser más común en las ramas de las ciencias puras y humanidades, tanto [PFMs](#) como [PFGs](#).

Decimos que es una pieza clave porque sirve como primer indicador de la consecución del proyecto. Si nos planteamos un objetivo, en las conclusiones podemos indicar si se ha cumplido o no el objetivo planteado. Por eso es necesario que el objetivo esté bien definido, porque si se acepta como objetivo válido en un proyecto, y éste se concluye como cumplido, el proyecto habrá sido ejecutado correctamente.

Ahora bien, ¿cómo determinamos que el objetivo se ha cumplido? pues intentando definirlo para que se pueda cumplir, es decir, intentando que sea:

- **Acotado en el tiempo**, así es más fácil establecer un marco temporal para su realización y programar temporalmente las partes de las que se compone.
- **Medible**, para saber cómo de lejos estamos de llegar a un resultado aceptable.
- **Específico**, de manera que esté bien acotado y sea difícil embarcarse en tareas que no nos acerquen a su consecución.
- **Alcanzable**, porque si no lo es, por mucha intención y esfuerzo que le pongamos no se va a terminar.
- **Relevante**, porque si, en un **PFG** para Ingeniería del Software, desarrollamos un producto mecánico para sexar pollos, pues por muy importante que sea, poco tiene que ver con lo que se ha estudiado durante todos estos años.

Y sí, para acordarnos de cuáles son estas características podemos usar el acrónimo

1.4 Justificación

1.5 Estructura del documento

2. Marco teórico y contexto tecnológico

La ciencia amigo, está compuesta de errores; pero son errores que es útil cometer ya que nos acercan poco a poco hacia la verdad.

Julio Verne

2.1 Marco Teórico

2.2 Contexto tecnológico

En esta sección se describirán las tecnologías y herramientas más relevantes que se utilizarán a lo largo del desarrollo proyecto. El uso de las mismas se describirá en la sección 6.

Gestión del proyecto

Notion

Notion es una plataforma online parcialmente gratuita, utilizada para el seguimiento de proyectos y de equipos. Esta herramienta es fácilmente personalizable a partir de una serie de elementos predefinidos, entre los que se encuentran bases de datos embebidas (y vistas visuales como las mismas, como tableros Kanban o diagramas Gantt), soporte para imágenes, vídeo y audio, integración con otras plataformas como Slack, Zoom, Twitter...

referenciar
sección
corres-
pondien-
te

En esta herramienta se pueden construir páginas, las cuales pueden contener texto formateado con el lenguaje de marcado Markdown o cualquiera de los bloques anteriores, por lo que fue elegida para conformar el centro neurálgico del seguimiento y documentación del proyecto, como se expone en la sección de Metodología.

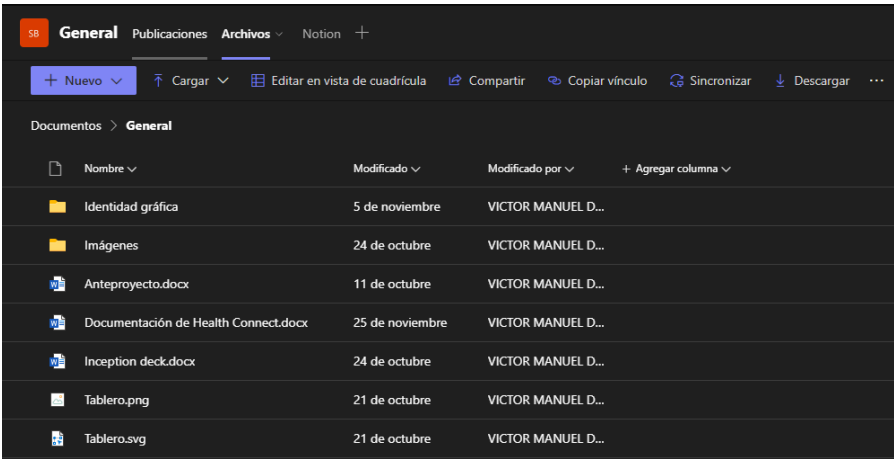
meter
aquí la
ref

Microsoft Teams

Microsoft Teams es una herramienta de comunicación y colaboración para equipos propiedad de Microsoft. Es principalmente utilizada para alojar canales de comunicación y videollamadas.

Para este proyecto lo hemos seleccionado por dos de sus funcionalidades: desplegar un sistema de archivos en la nube para cada canal de comunicación y registro de turnos de trabajo.

En cuanto a la primera de ellas, la hemos elegido por delante de otras opciones de alojamiento ya que por ser usuarios de la UPM, disponemos de 1 TB de almacenamiento, por lo que podemos alojar todo tipo de contenido multimedia prácticamente sin restricciones y de forma fácil y sencilla.



General			
+ Nuevo			
Cargar			
Editar en vista de cuadrícula			
Compartir			
Copiar vínculo			
Sincronizar			
Descargar			
Documents > General			
Nombre	Modificado	Modificado por	+ Agregar columna
Identidad gráfica	5 de noviembre	VICTOR MANUEL D...	
Imágenes	24 de octubre	VICTOR MANUEL D...	
Anteproyecto.docx	11 de octubre	VICTOR MANUEL D...	
Documentación de Health Connect.docx	25 de noviembre	VICTOR MANUEL D...	
Inception deck.docx	24 de octubre	VICTOR MANUEL D...	
Tablero.png	21 de octubre	VICTOR MANUEL D...	
Tablero.svg	21 de octubre	VICTOR MANUEL D...	

Figura 2.1: Extracto de los documentos alojados en Teams

Asimismo, la funcionalidad de registro de turnos es cómoda de usar y nos genera automáticamente un documento Excel con las horas registradas por parte de todos los miembros del equipo, ideal si queremos registrar el total de horas empleadas en el proyecto.

	A	B	C	D	E
1	Leyenda	Fecha	Nombre del empleado	Hora de entrada	Hora de salida
2	Rojo = editado	09/04/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/04/2022 01:38 PM	09/04/2022 08:01 PM
3	Púrpura = agregado	09/04/2022	VICTOR MANUEL DOMINGUEZ RIVAS		
4	Azul = eliminado	09/05/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/05/2022 12:56 PM	09/05/2022 02:50 PM
5		09/05/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/05/2022 04:35 PM	09/05/2022 06:43 PM
6		09/06/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/06/2022 04:01 PM	09/06/2022 05:16 PM
7		09/07/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/07/2022 12:18 PM	09/07/2022 02:08 PM
8		09/07/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/07/2022 04:54 PM	09/07/2022 06:59 PM
9		09/09/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/09/2022 11:58 AM	09/09/2022 01:28 PM
10		09/12/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/12/2022 10:44 AM	09/12/2022 02:33 PM
11		09/14/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/14/2022 01:07 PM	09/14/2022 02:17 PM
12		09/14/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/14/2022 05:10 PM	09/14/2022 06:05 PM
13		09/15/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/15/2022 08:45 PM	09/15/2022 09:52 PM
14		09/16/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/16/2022 01:30 PM	09/16/2022 02:46 PM
15		09/16/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/16/2022 03:55 PM	09/16/2022 05:45 PM
16		09/16/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/16/2022 11:50 PM	09/17/2022 12:31 AM
17		09/17/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/17/2022 07:50 PM	09/17/2022 10:52 PM
18		09/18/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/18/2022 12:13 AM	09/18/2022 01:19 AM
19		09/20/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/20/2022 11:34 AM	09/20/2022 07:24 PM
20		09/21/2022	VICTOR MANUEL DOMINGUEZ RIVAS	09/21/2022 11:59 AM	09/21/2022 01:27 PM

Figura 2.2: Extracto del registro de horas

Zotero

Zotero es una plataforma open-source (bajo la licencia AGPL) de gestión de referencias y citas bibliográficas, que permite recopilar y almacenar referencias de libros, artículos, sitios web y otros documentos cómodamente. Una vez registradas, permite exportar dichas citas de acuerdo a diferentes estilos de citación.

En este proyecto ha sido seleccionada por su facilidad de uso, por ser open-source y disponer de una gran cantidad de plugins, como *Zotero Connector*, que nos permite agregar referencias desde el navegador.

Ingeniería del Software

En esta sección vamos a describir brevemente algunos de los elementos de Ingeniería del Software más importantes que se usarán a lo largo del proyecto.

Inyección de dependencias

La Inyección de Dependencias

Integración y entrega continua

CI/CD es un concepto de Ingeniería de Software, relacionado estrechamente con la filosofía *DevOps* y con las metodologías ágiles; en el que se implementan flujos de trabajo que permitan distribuir las aplicaciones a los clientes de forma rápida, continua y fiable, a través de la automatización de ciertas etapas del desarrollo software.

El nombre es una combinación de dos siglas, las cuales tienen diferentes significados: CI hace referencia a *Continuous Integration* (integración continua), mientras que CD puede indicar dos procesos distintos: bien *Continuous Delivery* (entrega continua) o *Continuous Deployment* (despliegue continuo), los cuales en ocasiones se utilizan indistintamente [2].

Ahondando en dichos conceptos, la integración continua es un proceso por el cual se establece un único repositorio central, el cual contiene tanto el código del producto software como sus pruebas unitarias y de integración. Tras ello se establece una capa de automatización que permite realizar la integración, construcción y pruebas del software en cada actualización del repositorio.

Por otra parte, la entrega y despliegue continuo tienen en común la automatización de las etapas de puesta en marcha del nuevo software. No obstante, la entrega continua aboga por un despliegue manual de los artefactos al no necesitarse o no ser beneficioso cambios frecuentes en el entorno de producción, mientras que el despliegue continuo automatiza dicho despliegue. Por tanto, el uso de un proceso u otro reside en las necesidades del proyecto.

La implementación de una infraestructura CI/CD permite a los desarrolladores entregar cambios de forma rápida al reducir enormemente el tiempo de integración y despliegue, incrementando la eficiencia y eficacia en la implementación de software [3].

Asimismo, permite a los miembros del equipo trabajar de forma simultánea en distintas partes del software, sin tener que parar para la integración de todos los cambios y resolver los conflictos que puedan aparecer.

Control de versiones

Un sistema de control de versiones es una herramienta utilizada para controlar y gestionar el código fuente de un proyecto, si bien estos sistemas se pueden utilizar casi para cualquier tipo de fichero. Estos sistemas se encargan de monitorizar las modificaciones de los archivos de una carpeta, llamada repositorio, permitiendo guardar instantáneas de dicha carpeta; a las cuales se pueden volver en cualquier momento.

Con estas herramientas se protegen los archivos ante accidentes

Aplicación móvil

Android

A grandes rasgos, Android es un Sistema Operativo orientado a dispositivos móviles basado en el núcleo Linux, diseñado para ser independiente de la arquitectura hardware de dichos dispositivos. Si bien originalmente fue planteado para teléfonos móviles, con el avance de la industria ha adoptado un enfoque más amplio y es compatible con más dispositivos: tabletas, relojes inteligentes, televisores, pantallas de automóviles... aunque, excepto en el caso de las tabletas, se trata de versiones basadas en Android con su propia idiosincrasia.



Figura 2.3: Logo actual de Android. Imagen extraída de [4]

Normalmente cuando nos referimos a Android, no nos referimos únicamente al sistema operativo, sino a la plataforma creada entorno al mismo; como haremos a lo largo de este proyecto. Dicha plataforma o *framework* consta de numerosas capas, siendo el sistema operativo una parte de ellas. El sistema operativo como tal es denominado AOSP o *Android Open Source Project*, siendo su código fuente público. Cualquier persona puede acceder a él, descargarlo y modificarlo [5].

No obstante, en la inmensa mayoría de los teléfonos móviles el sistema operativo es complemen-

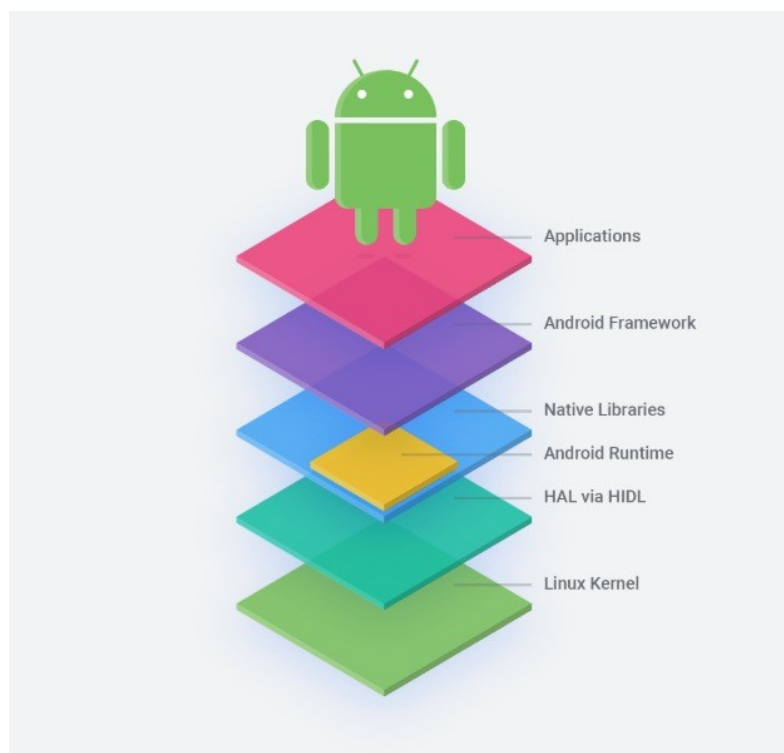


Figura 2.4: Capas de Android. Imagen extraída de [6]

tado con, entre otros, los GMS (*Google Mobile Services*, o servicios de Google), los cuales solo están disponibles bajo licencia; otorgada a los fabricantes que cumplen con una serie de requisitos. Los GMS se utilizan para tareas como la gestión de notificaciones, servicios de geolocalización... además de para acceder a las herramientas de Google, como la tienda de aplicaciones Play Store. Los fabricantes también pueden personalizar y añadir funciones al sistema operativo, lo que explica que dos terminales con la misma versión puedan verse tan diferentes entre sí.

Por otra parte, Android fue inicialmente desarrollado por la empresa homónima, si bien fue comprada en 2005 por Google por 50 millones de dólares. La salida del sistema operativo se produciría dos años después, el 5 de noviembre de 2007, si bien el primer terminal que lo utilizaba (HTC Dream, también conocido como T-Mobile G1) fue comercializado el 23 de septiembre de 2008 [7] [8].

Desde entonces, numerosas versiones de Android han sido lanzadas, siendo la última versión estable Android 13; estableciéndose por parte de Google la *costumbre* de lanzar cada año una nueva versión principal. En cada una de ellas se introducen nuevas características, pero esto no significa que todos los dispositivos puedan actualizar. Los fabricantes no están obligados a actualizar sus



Figura 2.5: HTC Dream en funcionamiento. Imagen extraída de [9]

terminales, lo que en la práctica supone que las nuevas versiones no son utilizadas masivamente y que los programadores deben de tener en cuenta las versiones antiguas en sus aplicaciones.

Debido a que Google dejó de publicar oficialmente las estadísticas de uso de su sistema operativo, no es posible conocer con plena exactitud dichas cifras. La comunidad se ha encargado de estimar dicha información [10]; relevando que a fecha de mayo de 2023 sólo el 20% de los dispositivos tienen la última versión, mientras que las versiones 12,11 y 10 están presentes en el 20,8%, 21,1% y 16,6% respectivamente.

Version	SDK / API level	Version code	Codename	Cumulative usage ¹	Year
Android 14 ^{DEV}	Level 34	UPSIDE_DOWN_CAKE	Upside Down Cake	—	TBD
Android 13	Level 33	TIRAMISU	Tiramisu ²	20.0%	2022
	▪ targetsdk will need to be 33+ for new apps and app updates by August 2023.				
Android 12	Level 32 Android 12L	S_V2	Snow Cone ²	40.8%	2021
	Level 31 Android 12	S			
	▪ targetsdk must be 31+ for new apps and app updates.				
Android 11	Level 30	R	Red Velvet Cake ²	61.9%	2020
Android 10	Level 29	Q	Quince Tart ²	78.5%	2019
Android 9	Level 28	P	Pie	86.6%	2018
Android 8	Level 27 Android 8.1	O_MR1	Oreo	91.3%	2017
	Level 26 Android 8.0	O		92.9%	
Android 7	Level 25 Android 7.1	N_MR1	Nougat	94.0%	2016
	Level 24 Android 7.0	N		96.2%	
Android 6	Level 23	M	Marshmallow	97.9%	2015
Android 5	Level 22 Android 5.1	LOLLIPOP_MR1	Lollipop	99.1%	2015
	Level 21 Android 5.0	LOLLIPOP, L		99.3%	2014
	▪ Jetpack Compose requires a minSdk of 21 or higher.				
Android 4	Level 20 Android 4.4W ³	KITKAT_WATCH	KitKat	No data	2013
	Level 19 Android 4.4	KITKAT			
	▪ Google Play services do not support Android versions below API level 19.				

Figura 2.6: Estadísticas acumulativas de las versiones de Android. Imagen extraída de [10]

Por último, a fecha de marzo de 2023, Android dispone de una cuota de mercado del 71% en el segmento de sistemas operativos para dispositivos móviles, teniendo su mayor rival, el sistema operativo iOS (propiedad de Apple) un 28%. Entre ambos acaparan el mercado, con un 99% de cuota de mercado. En cuanto a España, el porcentaje de Android asciende hasta el 77,73% por el 21,81 de iOS [11].

Kotlin

Durante el diseño de Android se estableció que el lenguaje principal para desarrollar aplicaciones sería Java, si bien incorpora soporte para utilizar código C y C++ [12]. No obstante, al ser Java un lenguaje interpretado sobre una máquina virtual (JVM o *Java Virtual Machine*) se abrió la puerta para utilizar otros lenguajes que utilizasen la JVM. En la conferencia de Google I/O de 2017 fue anunciado el soporte oficial y completo de Kotlin dentro de Android.



Figura 2.7: Logo de Kotlin. Imagen extraída de [13]

Kotlin es un lenguaje de programación desarrollado por JetBrains¹ y publicada su primera versión estable el 15 de febrero de 2016. El objetivo de este lenguaje fue tan sencillo como ambicioso: crear un lenguaje conciso (permitiendo reducir la cantidad de código *boilerplate*), con soporte de nuevas funcionalidades; pero sin renunciar a la rapidez de compilación de Java ni al todo el código escrito en él [14].

Sus principales características son las siguientes [15] [16]:

- Interoperable al 100% con Java, lo que facilita la reutilización de código ya existentes. Es interoperable en ambos sentidos.
- Permite escribir código más seguro, ya que en el diseño del lenguaje se solucionaron problemas crónicos de Java como las *Null Pointer Exception*. Según datos internos de Google, las aplicaciones escritas en Kotlin tienen un 20% de probabilidades menos de fallar.
- Soporte nativo y estructurado para la programación concurrente y asíncrona mediante construcciones como las *corrutinas* y los flujos.
- Desarrollo multiplataforma, no solo para Android: aplicaciones web, *backend* e iOS.
- Permite desarrollos en varios paradigmas: orientada a objetos, funcional, imperativa...

Asimismo, al convertirse en la *I/O* de 2019 en el lenguaje de referencia para el desarrollo de Android [17], los desarrollos de librerías y herramientas relacionadas con Android están escritas en este lenguaje, aprovechando al máximo sus nuevas características. Por tanto, si bien es interoperable con Java, está recomendado que los nuevos desarrollos lo utilicen [18], como se ha realizado en este proyecto.

¹JetBrains es una empresa muy reconocida dentro de la industria por crear una serie de entornos de desarrollo muy populares, como PhpStorm, CLion o IntelliJ IDEA. Sobre este último está construido Android Studio, el entorno de desarrollo oficial dentro de Android.

Jetpack Compose

Jetpack Compose es un conjunto de herramientas *modernas* de Android para el desarrollo de interfaces gráficas, lanzado en su primera versión estable el 28 de julio de 2021 por Google [19]. Este kit de librerías permite desarrollar en el ecosistema Android de forma nativa interfaces gráficas de manera declarativa como en los sistemas React, Flutter o SwiftUI; siguiendo las tendencias actuales de la industria en el desarrollo de aplicaciones móviles.

Este enfoque declarativo nos permite describir cómo queremos que sea nuestra interfaz gráfica. Asimismo, las interfaces que construimos con este sistema pueden estar interconectadas a un estado que definamos; describiendo cómo será nuestra interfaz para cada posible estado. Cuando ese estado cambie, nuestra interfaz gráfica cambiará automáticamente para mostrar el nuevo estado, simplificando enormemente el desarrollo y reduciendo el código necesario [20].

Hasta la aparición de Jetpack Compose, el desarrollo interfaces gráficas nativas en Android se realizaba con el enfoque conocido como programación imperativa. En este tipo de desarrollo es necesario especificar paso por paso cómo se va a construir dicha interfaz gráfica exhaustivamente. En dicho proceso (conocido en Android como sistema de vistas) se codificaba un fichero XML, en el que se describían todos los elementos gráficos (botones, textos...); para en el código Java/Kotlin de la aplicación se accediera a dichos elementos y se le aplicaran manualmente modificaciones y transformaciones [21].

Además, en Jetpack Compose, a diferencia del sistema anterior, los componentes gráficos están desacoplados del sistema operativo; por lo que no dependemos de la versión del terminal para mostrar correctamente nuestra interfaz gráfica. Eso ocurría anteriormente y como ya vimos, la fragmentación en Android es un problema endémico, lo que complicaba bastante el desarrollo. Asimismo, es compatible con los componentes XML del sistema anterior, lo que facilita la migración de los proyectos antiguos a este nuevo paradigma.

No obstante, como ya vimos en el apartado anterior, está diseñado para ser utilizado desde Kotlin, por lo que en la práctica obliga a usar dicho lenguaje; lo que en algunos casos puede resultar en un pico de dificultad hasta que se domina el lenguaje.

Material Design 3

Material Design 3 (también conocido como *Material You*) es la tercera iteración del conjunto de principios y directrices de diseño de Google, como respuesta a la creciente ubicuidad de Android: móviles con pantallas plegables, *smartwatch*, televisores [22]... Su primera implementación estable para Jetpack Compose fue lanzada el 24 de octubre de 2022 [23].

Al ser utilizado por Google para la creación de elementos gráficos tanto en sus aplicaciones como en el sistema operativo, es la guía de diseño de facto dentro del ecosistema Android.

Sus principales características son las siguientes [24]:

- Centrado en la personalización de la interfaz gráfica. Los diseñadores definirán tres colores principales, los cuales serán utilizados para los elementos gráficos de forma totalmente transparente al programador. A partir de dichos colores, se elaborará mediante la herramienta *Material Design Builder* [25] una paleta de colores con variantes de los mismos, diseñada para cumplir estándares de accesibilidad², garantizando el nivel de contraste correcto.

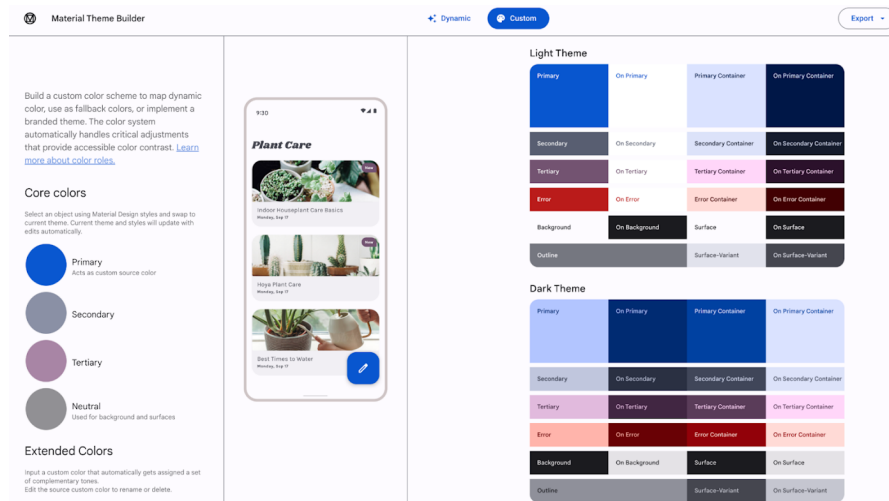


Figura 2.8: Ejemplo de uso de la herramienta *Material Design Builder*. Imagen extraída de [23]

Además, si el dispositivo dispone de Android 12 o superior, pueden tomarse dichos colores desde el fondo de pantalla del usuario, incrementando exponencialmente la personalización; si bien se permite establecer colores *fijos* para ciertos contenidos.

²Describir dichos estándares está fuera del alcance del proyecto, pero es un proceso basado en proporciones de luminancia

- Soporte nativo para categorizar el tamaño de la pantalla del dispositivo, tanto en altura como en anchura.

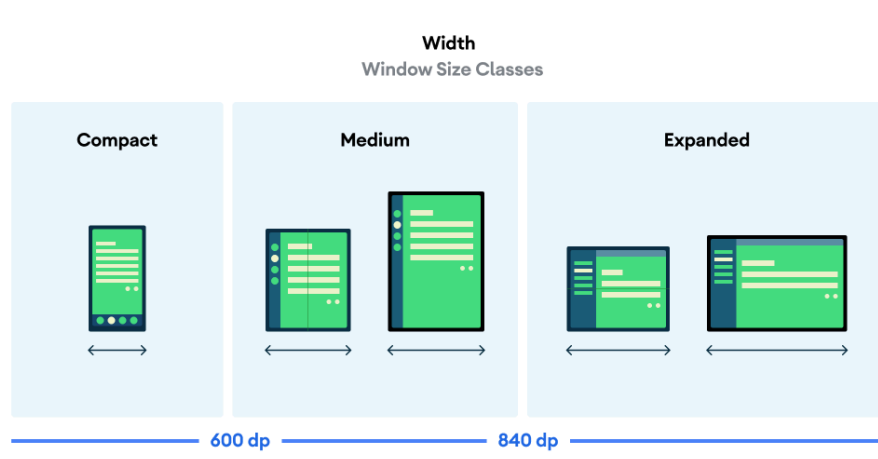


Figura 2.9: Categorías de pantalla según anchura. Imagen extraída de [23]

- Sistema de fuentes basado en estilos principales para cada tipo de contenido: desde titulares hasta etiquetas, pasando por títulos, cuerpos de texto...
- Soporte nativo para animaciones, las cuales ya son utilizadas en los componentes gráficos nativos, como los *switch*.
- Evolución de muchos elementos gráficos, como las tarjetas, botones, selectores de fechas...
- Sistema de formas o *redondeo* multinivel para modernizar nuestros elementos y hacerlos más distinguibles entre sí.
- Mejora el concepto conocido como *elevación*, basándose en colores y no en sombras. Este elemento permite superponer elementos y transmitir gráfica y visualmente la importancia de cada uno de ellos.
- Soporte nativo para tema claro y oscuro, ya que en los últimos años los temas oscuros se han hecho cada vez más populares en las aplicaciones y no estaba presente previamente de forma nativa.

En pocas palabras, en esta versión se han dedicado a modernizar el lenguaje de diseño, haciéndolo más atractivo y completo; alejándose de lo puramente funcional, mejorando la accesibilidad y explorando el mundo de la personalización para el usuario.

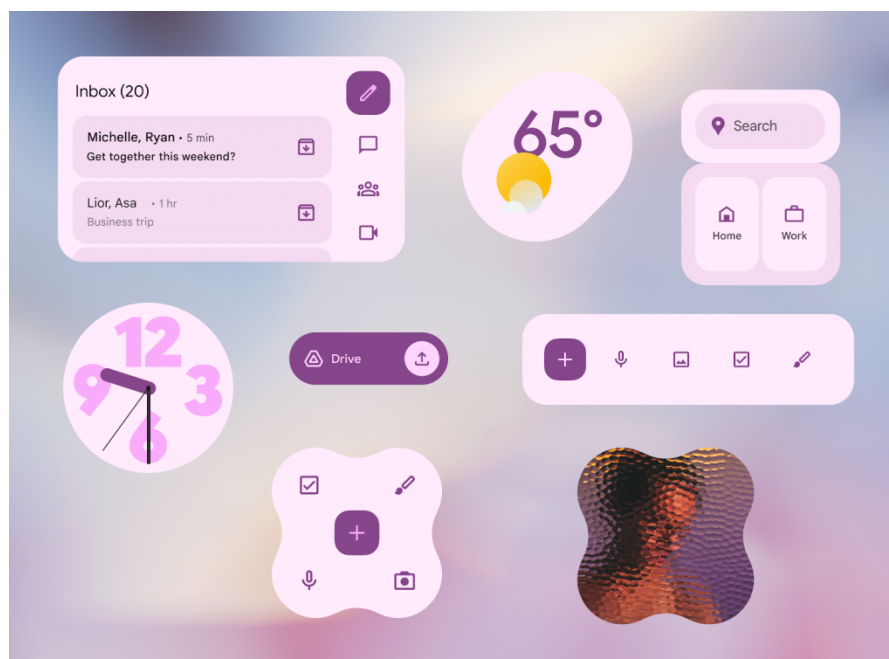


Figura 2.10: Algunos elementos gráficos de Material Design 3. Imagen extraída de [26]

Salud Conectada

En la conferencia I/O (sí, otra vez) de 2022 se anunció *Health Connect* (o Salud Conectada en castellano), una aplicación creada por Google (junto con Samsung [27]) que aglutinará todos los datos relacionados con salud dentro del ecosistema Android. La aplicación (en beta desde noviembre de 2022) es compatible con Android 9 o superior, mientras que está previsto que para Android 14 venga incorporada aplicación de fábrica o preinstalada³ [28].



Figura 2.11: Logo de Salud Conectada. Imagen extraída de [29]

Esta herramienta viene a solucionar una problemática importante y es la ausencia del reaprovechamiento de los datos relacionados con la salud. Hasta ese momento, la posición casi unánime

³Recordar aquí que esta aplicación es parte de los servicios de Google y no del sistema operativo propiamente dicho o AOSP.

dentro la industria se basaba en aislar los datos recoletados por sus dispositivos *wearables* o aplicación dentro de su ecosistema [30] [31].

La única manera utilizar dichos datos en otras aplicaciones estaba restringida a sistemas propietarios como Google Fit, cuyos datos se almacenaban en la nube con los problemas de privacidad asociados, o proyectos *open source* como Gadget Bridge [32]; que accedían a datos mediante ingeniería inversa.

A fecha de mayo de 2023, se estima que más de 100 aplicaciones han integrado Salud Conectada, incluyendo las aplicaciones de las pulseras Fitbit y Samsung, Peloton, Oura [33]...

Centrándonos en Salud Conectada, se trata tanto de una plataforma como una API para los desarrolladores. La plataforma puede registrar datos como la actividad física, el sueño, la nutrición o incluso el ciclo menstrual⁴, siendo un intermediario entre las aplicaciones que generan o escriben dichos datos y las que quieren acceder a esos datos.

Esta API estandariza el acceso a todas las fuentes de datos, independientemente de su procedencia; lo que simplifica enormemente tanto la lectura como escritura de los datos. Además, el sistema permite consultar datos agregados, pudiendo ser una agregación acumulativa (como el total de distancia caminada en un intervalo de tiempo) o estadística (las pulsaciones mínimas, máximas o promedio en un intervalo de tiempo).

Por otra parte, está diseñada con la privacidad en mente: los datos se almacenan localmente, mientras que el acceso a los mismos está fuertemente granularizado: en la que el usuario puede decidir qué aplicaciones tienen acceso (tanto lectura como escritura) a cada tipo de registro [35].

Asimismo, las aplicaciones solo pueden leer datos con una antigüedad de hasta 30 días previos a su instalación. [36], registrándose además todas las lecturas y escrituras en el sistema, las cuales puede visualizar el usuario.

En definitiva, si bien está aún en desarrollo, esta plataforma nos permite abstraernos del hardware

⁴La lista completa de los datos que puede registrar Salud Conectada se encuentra disponible en [34]

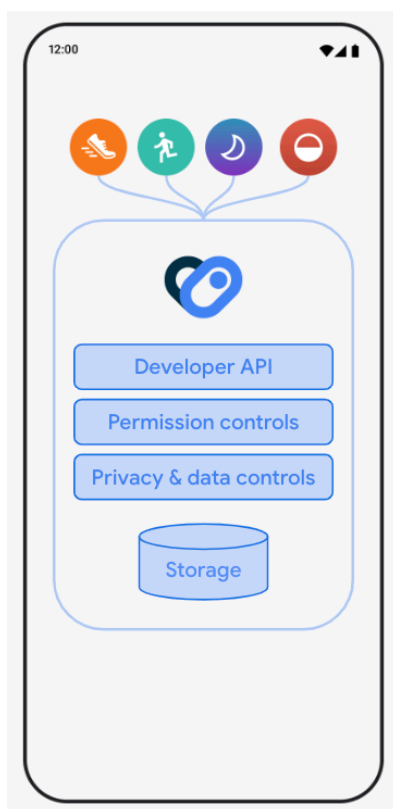


Figura 2.12: Arquitectura básica de Salud Conectada. Imagen extraída de [27]

de recolección de datos biométricos, reduciendo la complejidad de nuestro desarrollo; y además permite al usuario un mayor control sobre sus datos sensibles.

Room

Uno de los servicios que provee Android es SQLite, un sistema de gestión de base de datos relacionales de bajo nivel. Este sistema es usado por la aplicación que lo utiliza, ahorrando configuración y complejidad [37].

No obstante para muchas aplicaciones es un modelo demasiado rígido. Para simplificar el uso de SQLite Google creó en 2017 esta librería, la cual hace de intermediario entre el propio SQLite y nuestra aplicación [38]. Técnicamente hablando se trata de una capa de abstracción sobre dicho sistema gestor, la cual permite reducir la complejidad de los usos comunes de la base de datos; sin perder el acceso a SQLite. Asimismo, esta librería brinda otras ventajas, como la verificación de las consultas SQL en tiempo de compilación [39].



Figura 2.13: Visualización de datos de distancia. Elaboración propia.

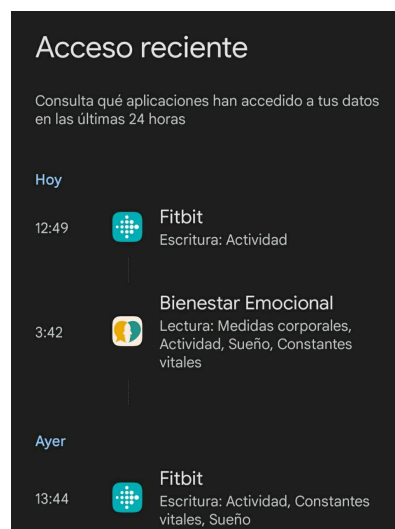


Figura 2.14: Visualización del acceso a los datos. Elaboración propia.

Los elementos básicos de esta plataforma son los siguientes:

- *Data Access Objects*, también conocido como DAO. En este elemento (implementado como una interfaz con la anotación `@Dao`) creamos nuestras operaciones sobre la base de datos: consultas, inserciones...

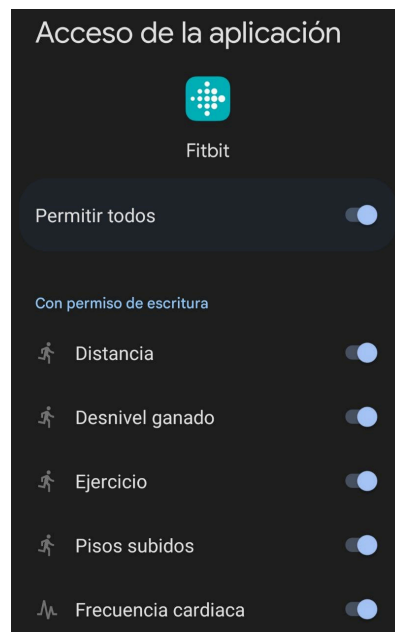


Figura 2.15: Granularidad de los permisos. Elaboración propia.

- Entidades, las cuales se corresponden con una clase en Kotlin (anotada como `@Entity`) y normalmente con una tabla de base de datos. Con esta estructura creamos las tablas de la base de datos, mientras que en nuestra aplicación disponemos de objetos que mapean dichos registros. Los atributos de dicha clase serán los campos de la tabla en cuestión, sobre los cuales podemos utilizar anotaciones para establecer claves, índices, etc
- Clase base de datos, donde especificamos las entidades a utilizar y cómo se instancia dicha base de datos (aquí podemos ajustar si estará cifrada o no por ejemplo). Se corresponde con una clase abstracta que hereda de `RoomDatabase` y que usa ciertas anotaciones para que se genere automáticamente el código *boilerplate*

Servidor

Python

Python es un lenguaje de programación interpretado creado por Guido Van Rossum en 1991, con un marcado enfoque de construir código de la manera más clara y legible posible. Es uno de los lenguajes más populares de la escena, gracias a su facilidad de uso y a su enorme comunidad de desarrolladores. Estos últimos han creado un enorme ecosistema de librerías que facilita la creación de nuevos proyectos, especialmente para prototipos.



Figura 2.16: Logo de Python. Imagen extraída de [40]

Actualmente es utilizado para todo tipo de tareas, tales como desarrollo web, aplicaciones científicas... pasando por inteligencia artificial o análisis de datos; casi siempre apoyándose en potentes librerías que mejoran notablemente el propio lenguaje.

Sus principales características son las siguientes:

- Soporte mutiparadigma: como Kotlin, se puede utilizar programación imperativa, orientada a objetos o funcional en cualquier momento.
- Gran biblioteca estándar que facilita muchas operaciones mundanas, como la lectura de archivos JSON.
- Soporte multiplataforma gracias a la compatibilidad de su intérprete con todos los sistemas operativos principales. Llega hasta tal punto que un subconjunto del lenguaje (MicroPython) es compatible con algunos microcontroladores.
- Sistema de tipos fuertemente tipado y dinámico, lo que permite que una variable pueda cambiar de tipo fácilmente.
- Facilidad de instalación de librerías gracias a la herramienta pip.

Flask

Flask es un popular entorno ligero o *microframework* de Python utilizado para construir aplicaciones web, tanto como páginas como API. Diseñado para ser ligero y fácil de usar, Flask proporciona las herramientas necesarias para crear dichas aplicaciones web de una manera minimalista, rápida y eficiente, en sintonía con la propia filosofía del lenguaje.

A diferencia de otros entornos más completos como Django, Flask se enfoca en brindar solo lo esencial, lo que permite crear prototipos lo más fácilmente posible, si bien puede quedarse corto para aplicaciones relativamente complejas [41]. Su sistema de enrutamiento HTTP es sencillo y claro, mientras que posee extensiones para autenticación de usuarios, manejo de base de datos, etc.

Gracias a esta simplicidad se usará en este proyecto para prototipar la parte del servidor. Asimismo, debido a su enfoque modular y a su sistema de plantillas podría expandirse su funcionalidad con más elementos web sin ninguna adaptación del código ya existente.

MongoDB

MongoDB es un sistema gestor de base de datos no relacional orientado a documentos, diseñado para manejar grandes volúmenes de datos de manera eficiente y escalable. A diferencia de las bases de datos relacionales tradicionales, no hay registros sino documentos, unos ficheros con una extensión muy similar a JSON: BSON (JSON Binarios) [42].



Figura 2.17: Logo de MongoDB. Imagen extraída de [43]

Este enfoque no relacional le permite ofrecer un modelo de datos flexible, sin un esquema. Esto se traduce en una mayor libertad para almacenar datos y cambiar el modelo de los mismos, ya que para cambiar cualquier atributo no es necesario reconstruir el modelo; muy útil para prototipos o modelos de datos dinámicos.

Por otra parte, el modelo de consultas ofrece posibilidades similares a las de SQL, por lo que no supone una desventaja. También implementa conceptos presentes en las base de datos relacionales, como los índices. Además, al disponer de una gran comunidad detrás, existen numerosos módulos que lo permiten integrar con lenguajes de programación, como pymongo para Python.

Asimismo, posee otras dos grandes fortalezas: un núcleo distribuido, lo que permite una alta disponibilidad y escalabilidad; y una licencia de uso gratuito desde octubre de 2018, bajo la licencia AGPL [44].

3.

Estado de la cuestión

En este capítulo se introduce una revisión del estado del arte y del estado de la cuestión en lo que respecta al marco teórico del proyecto que se propone en este TFM, así como una descripción sobre los sistemas que existen en el mercado o que han sido reportados en la literatura, los cuales presentan aspectos comunes con la solución propuesta.

3.1 Análisis de la situación actual

En esta sección, se pone en valor y/o contraste la información introducida en las secciones anteriores, de tal forma que pueda realizarse un análisis del entorno en el que se desarrolla este proyecto, así como también se pueda poner de manifiesto la relevancia y la idoneidad del desarrollo propuesto.

Proyectos relacionados

Contribución de la solución propuesta

4. Metodología y materiales

En este capítulo, se introducen algunas de las principales metodologías de desarrollo *software* que actualmente más se utilizan.

4.1 Metodologías de desarrollo del *software* de Sistemas

Metodología del desarrollo seleccionada

añadir
tableros
etc

hablar
de los
reposito-
rios git
y github

5. Diseño del sistema propuesto

5.1 Descripción del caso de estudio

5.2 Diseño del sistema

6.

Desarrollo del sistema

¿Quién dijo miedo habiendo hospitales?
Sabiduría popular de la ETSISI

6.1 Project Blastoff

descripción
del
blastoff

Inception Deck

El *Inception Deck* (también conocido como *Agile Inception*) [45] [46] es un conjunto de actividades propio de las metodologías ágiles, las cuales tienen como objetivo establecer inequívocamente los propósitos del proyecto y sus expectativas, a través de la comunicación entre todas las personas involucradas de alguna manera en el proyecto.

Más concretamente, el *Inception Deck* es un conjunto de 10 complejos ejercicios y preguntas, los cuales permiten unificar todas las visiones del producto hacia una sola, asegurando que el equipo avance en sintonía hacia una misma dirección.

Dichas dinámicas son las siguientes:

- ¿Por qué estamos aquí? (*Why are we here?*): describe los motivos principales por los que se realiza este proyecto, y cómo hemos llegado hasta este punto.
- *Elevator Pitch*: consiste en transmitir resumida y atractivamente la idea de nuestro proyecto condensada en unos 30 segundos (lo que podría durar un viaje en ascensor, de ahí el nombre) para captar la atención de otras personas. En este pequeño discurso se muestra el problema a resolver, cómo se pretende solventar y cuál es el valor diferencial que tiene nuestra idea y/o equipo, para alentar al oyente a colaborar con el mismo.

- Diseñar una caja de producto (*Design a Product Box*): obliga a imaginar nuestro proyecto encapsulado en un cartel publicitario, buscando para ello los mensajes e imágenes para promocionar el producto.
- Crear una lista de NOes (*Create a NOT list*): delimita los límites del proyecto estableciendo lo que no se va a realizar en el mismo.
- Conoce a tus vecinos (*Meet your neighbors*): esclarece las personas o equipos de los cuales depende el éxito de nuestro proyecto.
- Haz ver la solución (*Show the solution*): diseña a alto nivel la arquitectura técnica del producto, de forma legible para todas las partes del proyecto.
- ¿Qué nos quita el sueño? (*Ask what keep us up at night*): identifica los posibles riesgos a los que se enfrentará nuestro proyecto que pueden afectar a su desarrollo y éxito. Se diferencia entre los riesgos en los que podemos influir y en los que no, para decidir medidas para mitigar su posible impacto en el caso de los primeros y concienciar acerca de los segundos.
- Tómale las medidas (*Size it up*): estimación a grandes rasgos del orden de magnitud temporal del proyecto, para controlar las expectativas del mismo.
- Ser claros en qué vamos a dar (*Be clear on what's going to give*): determina las prioridades del proyecto en factores clave como el tiempo, alcance, presupuesto o calidad entre otros, junto a su flexibilidad para redimensionar el proyecto si es necesario.
- ¿Cuánto va a costar? (*Show what it's going to take*): desarrollo un presupuesto aproximado tanto en dinero como en personal y recursos si no se realizó previamente. En caso contrario, se discute la viabilidad el mismo y del proyecto.

A continuación se describe el resultado de dichas actividades para este proyecto.

Why are we here?

La Salud Mental es un área de la salud cuya visibilización aumenta cada día, pero que aún sigue recubierta de estigma y carece de recursos suficientes. Organizaciones como la Organización Mundial de la Salud y la Confederación de Salud Mental de España han realizado informes e infografías donde se presentan datos sobre este problema tan silencioso, grave y desconocido.

En nuestro país, el 6,7% de la población está afectada por la ansiedad, exactamente la misma cifra de personas con depresión. Casi la mitad de los españoles de entre 15 y 29 años (48,9%) considera

que ha tenido algún problema de salud mental, mientras que más de la mitad de las personas con trastorno mental que necesitan tratamiento no lo reciben.

A nivel mundial, el 12,5% de todos los problemas de salud está representado por los trastornos mentales, una cifra mayor a la del cáncer y los problemas cardiovasculares. Más de 300 millones de personas viven en el mundo con depresión, un problema que se ha aumentado en un 18,4% entre 2005 y 2015; mientras que 800.000 personas se suicidan cada año, siendo la segunda causa de muerte en personas de 15 a 29 años.

Con semejantes estadísticas, queda un largo camino por recorrer para que la sociedad considere a la salud mental como un área de salud igual de importante que las demás. Esto se manifiesta enormemente en la falta de atención a sus síntomas, lo que conlleva faltas crónicas de tratamiento entre la población.

Desde la Informática podemos acceder a numerosos datos, tanto de comportamiento de una persona con su móvil, como datos sobre su estado físico con la generalización de dispositivos conocidos como *wearables* (generalmente una pulsera o reloj inteligente equipado con sensores biométricos).

Bajo esta premisa se han publicado estudios [47] [48] [49] [50] que avalan que a partir de dichos datos se estimar por ejemplo si una persona tiene estrés, lo que supone un primer paso para la detección de problemas de salud mental.

Asimismo, los fabricantes de los dispositivos *wearables* ofrecen aplicaciones en las que se pueden consultar los datos recolectados, pero no ofrecen datos acerca de la salud mental, y tampoco son ofrecidos en los smartphones.

Por este motivo estamos aquí: para realizar una aplicación que aunando software y hardware pueda alertar de síntomas de problemas de salud mental, para que dicha persona sea consciente de que su salud mental puede estar deteriorándose y quizás necesite ayuda profesional.

Elevator pitch

Se estima que el 25% de las personas tendrá un trastorno mental a lo largo de su vida, y entre ellas, entre el 35% y el 50% no reciben tratamiento o no es el adecuado. Para la comunidad universitaria presentamos el Sistema para el Bienestar Emocional, una aplicación que permite obtener tu nivel de estrés teniendo en cuenta el uso del móvil y opcionalmente la información de *wearables*.

A diferencia de otras aplicaciones, no nos fijamos únicamente en el dato de un sensor invasivo ni comunicamos datos con terceras empresas, nuestro producto es una aplicación *open source*, por

lo que puedes modificarla a tu antojo, que proporciona resultados más elaborados y precisos y elabora recomendaciones para tu situación mental.

Product Box

TODO

NOT list

Dentro del alcance	Fuera del alcance
<ul style="list-style-type: none">• Desarrollo de la aplicación de usuario para dispositivos Android en el lenguaje de programación Kotlin.• Lectura de datos biométricos del usuario: ritmo cardíaco y sus variaciones, sueño, actividad física.• Obtención de datos del móvil: sueño, uso de aplicaciones.• Comunicación de resultados a los usuarios mediante notificaciones.• Visualización tanto del estado de bienestar emocional actual como de su evolución.• <i>Layout responsive</i> de la aplicación para adaptarla a todo tipo de dispositivos.• Desarrollo del modo oscuro de la interfaz de la aplicación.• Conexión de la aplicación con el modelo de Inteligencia Artificial.• Aporte de los datos de los usuarios para entrenar el modelo de Inteligencia Artificial.• Validación mediante tests con cuestionarios para corroborar los resultados obtenidos.• Realización profesional de la gestión de proyecto siguiendo metodologías ágiles.	<ul style="list-style-type: none">• Desarrollo de la aplicación para terminales con otro sistema operativo, como iOS.• Retro-aprendizaje del modelo de Inteligencia Artificial para un usuario en particular.• Evolución del modelo de Inteligencia Artificial tras la prueba piloto.• Utilización de mecanismos de <i>edge computing</i> en la aplicación de usuario.• Implementación de cuentas de usuario para mantener los datos entre dispositivos.• Realización de ingeniería inversa a las pulseras Xiaomi desde cero.• Inclusión de otras pulseras que no dispongan del sistema operativo Wear OS.• Uso de la herramienta Google Fit para obtener datos de dispositivos.• Lectura de sensores de estrés y oxígeno en sangre por ser invasivos de cara al usuario.• Utilización de escáneres cerebrales u otros dispositivos similares.
No resuelto	

nos falta la imagen cuando esté disponible

Cuadro 6.1: Lista de NOes del proyecto

Meet your neighbors

- Estudiantes de la universidad como usuarios del sistema.
- Profesores de la universidad como usuarios y/o promotores del sistema.
- Responsables de la infraestructura de la universidad para poder entrenar el modelo de Inteligencia Artificial.
- Fabricantes tanto de los *wearables* como de los smartphones.
- Desarrolladores del S.O. Android y sus API oficiales.
- Futuros desarrolladores del proyecto, como estudiantes de la escuela para sus proyectos Fin de Titulación; bien a nivel de aplicación o de Inteligencia Artificial.
- Subdirección de Asuntos Económicos de la ETSISI como responsables de la compra de material hardware para este proyecto.
- Consultores internos sobre desarrollo de aplicaciones móviles, ingeniería de software e integración de componentes hardware para transmitir los conocimientos necesarios para continuar el proyecto/resultados del mismo.

Show the solution

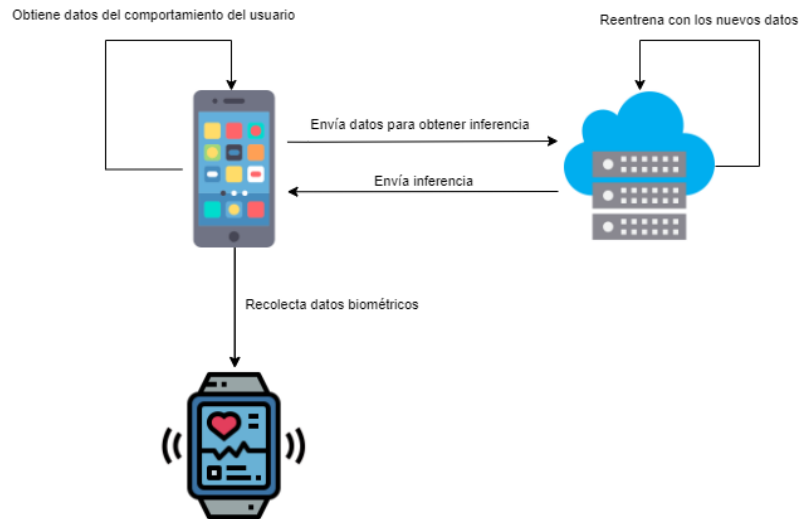


Figura 6.1: Componentes básicos del sistema



Figura 6.2: Pila con la arquitectura del sistema

Up at night

Riesgos en los que podemos influir	Riesgos en los que no podemos influir
<ul style="list-style-type: none"> • Realización de pruebas exhaustivas tanto del <i>backend</i> como de la interfaz gráfica de usuario. • Participación de voluntarios que aporten datos al estudio con los que mejorar la precisión y evitar sesgos. • Elaboración de unos términos y condiciones legales acerca de los datos obtenidos en el proyecto. • Accesibilidad correcta de la aplicación. • Elaboración de un diseño coherente a nivel UX/UI de la aplicación. • Comunicación y difusión eficaz del proyecto en la escuela, en la defensa de este TFM y en redes sociales. • Continuidad del proyecto después del presente TFM para, entre otros, publicar una aplicación en iOS. • Definición clara de la arquitectura y de los procesos de mantenimiento y ampliación de la aplicación. • Mantenimiento futuro de la aplicación con las nuevas versiones de Android. • Potencia de cómputo suficiente para entrenar el modelo de Inteligencia Artificial 	<ul style="list-style-type: none"> • Problemas de compatibilidad entre las versiones de Android. • Muestra de estudiantes variada y contrastada participando en el proyecto. • Soportes de los fabricantes de <i>wearables</i> (en especial Samsung) a Health Connect. • Funcionamiento de las pulseras Xiaomi en aplicaciones no oficiales. • Disponibilidad de los datos del móvil y de los <i>wearables</i>. • Fiabilidad suficiente de los datos obtenidos por el móvil y los <i>wearables</i>. • Incumplimiento de la planificación por circunstancias ajenas al proyecto. • Retrasos en la entrega del hardware necesario para el sistema. • Ausencia de fallos al publicar la aplicación en la Google Play Store. • Precisión insuficiente del modelo de Inteligencia Artificial para ser utilizado en un entorno real.

Cuadro 6.2: Riesgos del proyecto

Size it up

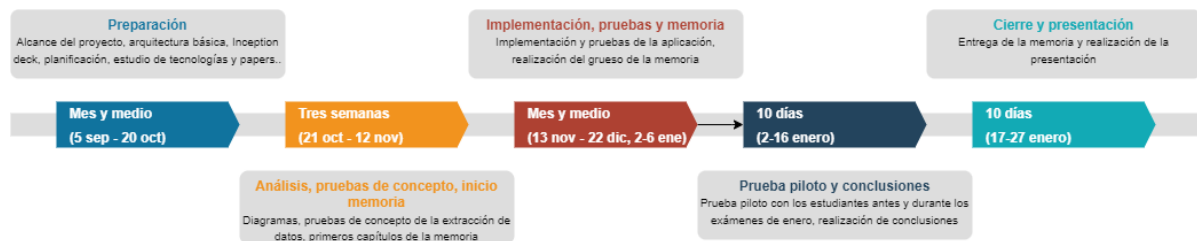


Figura 6.3: Dimensionamiento aproximado del proyecto

What's going to give

Aspecto	Importancia	Implicaciones
Detalle de la documentación	Media	Se desea que la memoria explique detalladamente las cuestiones importantes del proyecto y cubra todo lo realizado en el mismo, pero no es necesario en detalles menores ni se pretende que la memoria sea excesivamente larga.
Alcance de la implementación	Baja - media	Una vez realizada la conexión con las API de Android y con el hardware concedido, se puede establecer flexibilidad con el alcance del prototipo.
Calidad	Media - alta	Se deben realizar pruebas detalladas y obtener métricas de calidad del software implementado, pero se permite cierta flexibilidad en las mismas sobre el software que maneje el hardware del proyecto por su alta dificultad.
Comunicación del proyecto	Alta	Se necesitan usuarios para la prueba piloto y para obtener retroalimentación del proyecto. No obstante, si bien se desea que el proyecto sea continuado, no es estrictamente necesario.

Experiencia de usuario	Media	La aplicación debe ser intuitiva y accesible para el usuario, pero no se necesita que el prototipo tenga un diseño visual (elementos, animaciones, gráficas) muy elaborado y detallado al ser un prototipo.
Fiabilidad	Media	No están previstas pruebas exhaustivas con todos los dispositivos compatibles (versiones del SO, dispositivos hardware), por lo que <i>bugs</i> leves son admisibles.
Presupuesto (hardware)	Muy alta	El presupuesto está cerrado al aprobado, por lo que no se pueden incorporar nuevos componentes.
Rendimiento	Baja - media	La optimización del software queda fuera del proyecto, si bien se debe de garantizar un rendimiento mínimo para no afectar la experiencia de usuario.
Seguridad	Alta	Si bien los datos del móvil no son personales, los de los <i>wearables</i> sí lo son, por lo que deben ser debidamente protegidos tanto en su almacenamiento como en su transmisión con técnicas estándar.
Tiempo	Muy alta	El máster proporciona dos convocatorias fijas en el calendario, y por la situación personal del autor solo puede acudir a la de enero, por lo que el proyecto debe finalizar para esa fecha.

Cuadro 6.3: Cuestiones y prioridades del proyecto

What it's going to take

En el apartado de Tómale las medidas se establecieron las etapas del proyecto, por lo que ya conocemos el tiempo estimado de desarrollo del proyecto: cuatro meses y medio. Asimismo, al ser un Trabajo Final de Máster realizado por una persona se necesita a un único ingeniero para el proyecto.

Durante ese plazo las labores del desarrollador incluyen (pero no se limitan a):

- Análisis y planteamiento del problema a resolver.
- Diseño de un sistema que combine hardware y software.
- Implementación de una aplicación Android con el lenguaje de programación Kotlin.
- Diseño de interfaces de usuario y experiencia de usuario.
- Comunicación de smartphones Android con dispositivos *wearables* mediante Bluetooth.
- Realización de pruebas unitarias, funcionales y de integración.
- Uso de sistemas de control de versiones, como Github.
- Desarrollo de un pipeline CI/CD.
- Aplicación de metodologías ágiles.

No obstante, son labores que puede desarrollar un perfil con formación universitaria en Informática, ya que al ser un proyecto relativamente breve no se necesita de personal especializado en análisis, pruebas, etc.

Por tanto, se supone que dicho perfil es un Android Developer y que su salario es el promedio. Según la conocida web de empleo Glassdoor, dicho salario medio en Madrid es de 33.000 €/año, por lo que su salario prorrateado es de 12.375€. Asimismo, hay que sumar el presupuesto concedido por la universidad, que asciende a 460€.

Por tanto, la estimación del coste total del proyecto es de 12.835€.

6.2 Setup del proyecto

6.3 Implementación de la aplicación móvil

6.4 Implementación de la API del servidor

Subida de datos de usuario

Subida de los datos de los cuestionarios

Datos de la comunidad

7.

Resultados

¡Los números Mason! ¿Qué significan?

Jason Hudson

7.1 Preparación del entorno

En este apartado se enumeran los pasos seguidos para la preparación del entorno que ha sido utilizado en el desarrollo del sistema.

Sistema Operativo

Tecnología 1

Tecnología ...

Tecnología N

SectionImplementación del caso de estudio

7.2 Resultados del caso de estudio

En este apartado se muestran algunos ejemplos de los utilización y resultados que podrían obtenerse.

8.

Impacto social y medioambiental

En este capítulo se recogen los beneficios que la implantación del proyecto desarrollado podría generar tanto a nivel medioambiental como su impacto social.

8.1 Contexto medioambiental

Los Objetivos de Desarrollo Sostenible (ODS)....

8.2 Aspectos éticos, sociales y económicos

Hablar
aquí de
la pri-
vacidad
de los
datos
antes de
health
connect
(usa,
gpd)

Estamos
en la
cresta
de la ola
al venir
todo
para
Android
14

9.

Presupuesto

10.

Conclusiones

Lo perfecto es enemigo de lo bueno.

Cita atribuida a Voltaire

11.

Líneas futuras

Cada día me gusta levantarme porque hay otro reto.

Roger Penske

12.

Temporal

12.1 Setup del proyecto

12.2 Sprint #0

Este sprint se desarrolló entre el 5 y el 21 de septiembre, y consistió en la selección y puesta en marcha de las herramientas que se utilizarían para el desarrollo de este proyecto.

Git y Github

Git es sistema de control de versiones open-source (bajo la licencia GNU GPL v2) diseñado por Linus Torvalds, siendo muy popular entre los equipos de desarrollo software. Si bien Git almacena y gestiona cambios en archivos de cualquier tipo, está pensado para ser más eficiente en los archivos de código fuente.

Por otra parte, Github es una plataforma en línea que ofrece alojamiento y gestión de proyectos que utilicen Git. Para este proyecto, además del alojamiento, utilizamos sus herramientas de CI/CD (GitHub Actions), por la que a través de un fichero .yaml podemos configurar un pipeline de acciones con las que realizar los procesos que queramos automáticamente. Además, pueden establecerse flujos de trabajos por ramas de desarrollo u otras condiciones, tales como formatos de tags, tipos de eventos...

L^AT_EXy Overleaf

L^AT_EX como medio de creación de la presente memoria y Overleaf como editor de dicho lenguaje.

12.3 Desarrollo

Inyección de dependencias (Dagger Hilt)

Cifrado de la base de datos (SQLCipher)

Navegación entre ventanas (Compose Destination)

Planificación de tareas (Work Manager)

Comunicación con el servidor (Retrofit)

Ajustes del usuario (Datastore)

Localización de la app (Strings.xml y Lingliver)

Onboarding (Lottie)

Gráficas (Vico)

Flujos de integración continua (GitHub Actions)

Bibliografía

- [1] A. Salamanca, *La geopolítica de los papers: conocimiento libre contra la millonaria industria de las revistas académicas*, es, mar. de 2023. dirección: <https://elordenmundial.com/sci-hub-revistas-academicas-lucha-guerrillera-industria-millonaria/> (visitado 24-06-2023).
- [2] Redhat, *¿Qué son la integración/distribución continuas (CI/CD)?*, es, mayo de 2022. dirección: <https://www.redhat.com/es/topics/devops/what-is-ci-cd> (visitado 16-12-2022).
- [3] beServices, *Integración continua del software ¿Qué significan las siglas CI CD?* | beServices, es. dirección: <https://www.beservices.es/ci-cd-que-es-n-5488-es> (visitado 16-12-2022).
- [4] G. L. VulcanSphere vectorised by CMetalCore and optimised by, *English: The Android logo as of 2019*, ago. de 2019. dirección: [https://commons.wikimedia.org/wiki/File:Android_logo_2019_\(stacked\).svg](https://commons.wikimedia.org/wiki/File:Android_logo_2019_(stacked).svg) (visitado 26-06-2023).
- [5] C. Collado, *Qué es AOSP: así funciona el Android sin Google*, es, mayo de 2022. dirección: <https://www.lavanguardia.com/andro4all/android/que-es-aosp-asi-funciona-el-android-sin-google> (visitado 26-06-2023).
- [6] E. Pérez, *AOSP: así es el Android 'open source' sin Google que queda como opción para Huawei*, es, Section: aplicaciones, mayo de 2019. dirección: <https://www.xataka.com/aplicaciones/aosp-asi-al-android-open-source-google-que-queda-como-opcion-para-huawei> (visitado 26-06-2023).
- [7] R. Adeva, *Android: qué es, versiones, aplicaciones y cómo saber la versión instalada*, es, feb. de 2023. dirección: <https://www.adslzone.net/reportajes/software/que-es-android/> (visitado 26-06-2023).

- [8] J. Marquez, *Así era el HTC Dream, el primer teléfono de la historia con Android (y sí, tenía teclado QWERTY)*, es, Section: moviles, mayo de 2022. dirección: <https://www.xataka.com/moviles/asi-era-htc-dream-primer-telefono-historia-android-tenia-teclado-qwerty> (visitado 26-06-2023).
- [9] M. Oryl, *T-Mobile G1 Launch Event*, sep. de 2008. dirección: https://commons.wikimedia.org/wiki/File:T-Mobile_G1_launch_event_2.jpg (visitado 26-06-2023).
- [10] E. Belinski, *Android API Levels*. dirección: <https://apilevels.com/> (visitado 26-06-2023).
- [11] E. Press, *Así se reparten Android e iOS el mercado global de sistemas operativos móviles*, (SCHE-ME=ISO639) es, Publisher: Europa Press, abr. de 2023. dirección: <https://www.europapress.es/portaltic/software/noticia-asi-reparten-android-ios-mercado-global-sistemas-operativos-moviles-20230404123515.html> (visitado 26-06-2023).
- [12] A. Developers, *Cómo agregar código C y C++ a un proyecto / Android Studio*, es-419. dirección: <https://developer.android.com/studio/projects/add-native-code?hl=es-419> (visitado 26-06-2023).
- [13] *Kotlin brand assets / Kotlin*, en-US. dirección: <https://kotlinlang.org/docs/kotlin-brand-assets.html> (visitado 27-06-2023).
- [14] R. Rao K, *The history of Kotlin - Kotlin for Enterprise Applications using Java EE [Book]*, en, ISBN: 9781788997270. dirección: <https://www.oreilly.com/library/view/kotlin-for-enterprise/9781788997270/ea4ec584-db64-4026-89a8-2086301eb9c5.xhtml> (visitado 26-06-2023).
- [15] *Kotlin for Android / Kotlin*, en-US. dirección: <https://kotlinlang.org/docs/android-overview.html> (visitado 26-06-2023).
- [16] *Enfoque de prioridad de Kotlin en Android / Android Developers*, es-419. dirección: <https://developer.android.com/kotlin/first?hl=es-419> (visitado 26-06-2023).
- [17] M. Braun, *Celebrating 5 years of Kotlin on Android*, en, ago. de 2022. dirección: <https://android-developers.googleblog.com/2022/08/celebrating-5-years-of-kotlin-on-android.html> (visitado 26-06-2023).
- [18] F. Lardinois, *Kotlin is now Google's preferred language for Android app development*, en-US, mayo de 2019. dirección: <https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/> (visitado 26-06-2023).

- [19] A.-C. Bellini y N. Butcher, *Jetpack Compose is now 1.0: announcing Android's modern toolkit for building native UI*, en, jul. de 2021. dirección: <https://android-developers.googleblog.com/2021/07/jetpack-compose-announcement.html> (visitado 26-06-2023).
- [20] A. Leiva, *Qué es Jetpack Compose y cómo crear tu primer proyecto en Android*, es, sep. de 2021. dirección: <https://devexperto.com/jetpack-compose-que-es/> (visitado 26-06-2023).
- [21] *Programación imperativa vs declarativa: Google Jetpack Compose*, es-ES, mayo de 2021. dirección: <https://www2.deloitte.com/es/es/blog/todo-tecnologia/2021/programacion-imperativa-vs-declarativa-google-jetpack-compose.html> (visitado 26-06-2023).
- [22] I. Ramírez, *Qué es Material You y en qué se diferencia de Material Design*, es, Section: sistema-operativo, mar. de 2022. dirección: <https://www.xatakandroid.com/sistema-operativo/que-material-you-que-se-diferencia-material-design> (visitado 26-06-2023).
- [23] G. Singh, M. Design y D. Advocate, *Material Design 3 for Compose is now stable*, en, oct. de 2022. dirección: <https://material.io/blog/material-3-compose-stable> (visitado 26-06-2023).
- [24] *Material Design*, en. dirección: <https://m3.material.io/> (visitado 26-06-2023).
- [25] *Material Design Builder*. dirección: <https://m3.material.io/theme-builder#/dynamic> (visitado 26-06-2023).
- [26] A. Cerda, *Material Design 3: Novedades para el sistema de diseño de Google*, es-CL, mayo de 2022. dirección: <https://blog.ida.cl/disenio/material-design-3-novedades-sistema-diseno-google/> (visitado 26-06-2023).
- [27] C. Wilk, *Introducing Health Connect, a new API for Android app developers to securely access user health data*, en, mayo de 2022. dirección: <https://android-developers.googleblog.com/2022/05/introducing-health-connect.html> (visitado 26-06-2023).
- [28] R. Pandey, *Health Connect could be built right into Android 14*, en, Section: Operating Systems, feb. de 2023. dirección: <https://www.androidpolice.com/health-connect-built-right-into-android-14/> (visitado 26-06-2023).
- [29] *Health Connect | Desarrolladores de Android*, es-419. dirección: <https://developer.android.com/guide/health-and-fitness/health-connect?hl=es-419> (visitado 26-06-2023).

- [30] M. Ramírez, *Android 14 tendrá como eje central tus datos de salud y así arreglar el desaguado actual con tanta app*, es, Section: Actualizaciones Android, dic. de 2022. dirección: https://www.lespanol.com/elandroidelibre/noticias-y-novedades/actualizaciones-android/20221219/android-central-datos-salud-arreglar-desaguado-actual/727177645_0.html (visitado 26-06-2023).
- [31] M. Rahman, *Android 14 has built-in support for Google and Samsung's Health Connect platform*, en, Section: Mobile, feb. de 2023. dirección: <https://www.xda-developers.com/android-14-health-connect-built-in/> (visitado 26-06-2023).
- [32] Freeyourgadget, *Gadgetbridge*, es-ES. dirección: <https://codeberg.org/Freeyourgadget/Gadgetbridge> (visitado 26-06-2023).
- [33] A. Malik, *Google's Health Connect platform is coming to Android 14 with new features*, en-US, mayo de 2023. dirección: <https://techcrunch.com/2023/05/10/googles-health-connect-platform-is-coming-to-android-14-with-new-features/> (visitado 26-06-2023).
- [34] *Lista de tipos de datos / Desarrolladores de Android*, es-419. dirección: <https://developer.android.com/guide/health-and-fitness/health-connect/data-and-data-types/data-types?hl=es-419> (visitado 27-06-2023).
- [35] R. Sáez, *Google lanza Health Connect, el sitio donde podrás gestionar todas tus apps de salud y fitness*, es, Section: Aplicaciones, nov. de 2022. dirección: <https://www.lavanguardia.com/tecnologia/aplicaciones/20221117/8608956/google-lanza-health-connect-sitio-podras-gestionar-todas-apps-salud-fitness-pmv.html> (visitado 26-06-2023).
- [36] *Preguntas frecuentes / Desarrolladores de Android*, es-419. dirección: <https://developer.android.com/guide/health-and-fitness/health-connect/frequently-asked-questions?hl=es-419> (visitado 27-06-2023).
- [37] A. P. Recio, *Persistencia de datos en Android con Room*, es, mar. de 2019. dirección: <https://www.adictosaltrabajo.com/2019/03/04/persistencia-de-datos-en-android-con-room/> (visitado 27-06-2023).
- [38] A. Leiva, *Room, la librería de Base de datos de Android*, es, jul. de 2020. dirección: <https://devexperto.com/room-la-libreria-de-base-de-datos-de-android/> (visitado 27-06-2023).
- [39] *Cómo guardar contenido en una base de datos local con Room*, es-419. dirección: <https://developer.android.com/training/data-storage/room?hl=es-419> (visitado 27-06-2023).

- [40] *The Python Logo*, en. dirección: <https://www.python.org/community/logos/> (visitado 27-06-2023).
- [41] A. Rodríguez, *Flask: minimalismo para el desarrollo web en Python*, es, ago. de 2014. dirección: <http://hipertextual.com/2014/08/flask-python> (visitado 27-06-2023).
- [42] *JSON And BSON*, en-us. dirección: <https://www.mongodb.com/json-and-bson> (visitado 27-06-2023).
- [43] *MongoDB Brand Resources*, en-us. dirección: <https://www.mongodb.com/brand-resources> (visitado 27-06-2023).
- [44] *¿Qué Es MongoDB?*, es. dirección: <https://www.mongodb.com/es/what-is-mongodb> (visitado 27-06-2023).
- [45] J. Rasmusson, *The Agile Samurai*, English. Dallas: The Pragmatic Bookshelf, 2010, ISBN: 978-1-934356-58-6. (visitado 24-10-2022).
- [46] M. López Mendoza, *Agile Inception: Qué es y cómo ejecutarlo* / OpenWebinars, abr. de 2021. dirección: <https://openwebinars.net/blog/agile-inception-que-es-y-como-ejecutarlo/> (visitado 24-10-2022).
- [47] P. Schmidt, A. Reiss, R. Duerichen, C. Marberger y K. Van Laerhoven, «Introducing WESAD, a Multimodal Dataset for Wearable Stress and Affect Detection», *ICMI'18*, oct. de 2018.
- [48] M. Boukhechba, A. R. Daros, K. Fua, P. I. Chow, B. A. Teachman y L. E. Barnes, «DemonicSalmon: Monitoring mental health and social interactions of college students using smartphones», *Smart Health*, 9-10, 2018.
- [49] B. A. Hickey, T. Chalmers, P. Newton et al., «Smart Devices and Wearable Technologies to Detect and Monitor Mental Health Conditions and Stress: A Systematic Review», *Sensors*, 2021.
- [50] W. Rui, C. Fanglin, C. Zhenyu et al., «StudentLife: Assessing Mental Health, Academic Performance and Behavioral Trends of College Students using Smartphones», *Ubicomp*, 2014.

A. Código fuente del proyecto

A.1 Aplicación móvil

Paquete raíz

```
1 package es.upm.bienestaremocional
2
3 import android.os.Bundle
4 import androidx.activity.ComponentActivity
5 import androidx.activity.compose.setContent
6 import dagger.hilt.android.AndroidEntryPoint
7 import es.upm.bienestaremocional.data.settings.AppSettings
8 import es.upm.bienestaremocional.data.settings.ThemeMode
9 import kotlinx.coroutines.CoroutineScope
10 import kotlinx.coroutines.Dispatchers
11 import kotlinx.coroutines.flow.first
12 import kotlinx.coroutines.launch
13 import javax.inject.Inject
14 import kotlin.properties.Delegates
15
16 @AndroidEntryPoint
17 class MainActivity : ComponentActivity() {
18     @Inject
19     lateinit var appSettings: AppSettings
20
21     override fun onCreate(savedInstanceState: Bundle?) {
22         super.onCreate(savedInstanceState)
23
24         lateinit var darkTheme: ThemeMode
25         var dynamicColors by Delegates.notNull<Boolean>()
26     }
```

```
27     val job = CoroutineScope(Dispatchers.Default).launch {
28         darkTheme = appSettings.getTheme().first()
29         dynamicColors = appSettings.getDynamicColors().first()
30     }
31
32     CoroutineScope(Dispatchers.Main).launch {
33         job.join()
34         setContent {
35             BienestarEmocionalApp(
36                 darkTheme = darkTheme,
37                 dynamicColors = dynamicColors,
38             )
39         }
40     }
41 }
42 }
```

Listado A.1: MainActivity.kt

```
1 package es.upm.bienestaremocional
2
3 import android.app.Application
4 import androidx.hilt.work.HiltWorkerFactory
5 import androidx.work.Configuration
6 import dagger.hilt.android.HiltAndroidApp
7 import javax.inject.Inject
8
9
10 /**
11  * This class is needed to instantiate Hilt.
12  */
13 @HiltAndroidApp
14 class MainApplication : Application(), Configuration.Provider {
15     // Work Manager with Hilt
16     @Inject
17     lateinit var workerFactory: HiltWorkerFactory
18
19     override fun getWorkManagerConfiguration() =
20         Configuration.Builder()
21             .setWorkerFactory(workerFactory)
22             .build()
23 }
```

Listado A.2: MainApplication.kt

```
1 package es.upm.bienestaremocional
2
3 import androidx.compose.runtime.Composable
4 import com.ramcosta.composedestinations.DestinationsNavHost
5 import es.upm.bienestaremocional.data.settings.ThemeMode
6 import es.upm.bienestaremocional.ui.screens.NavGraphs
7 import es.upm.bienestaremocional.ui.theme.BienestarEmocionalTheme
8
9 @Composable
10 fun BienestarEmocionalApp(
11     darkTheme: ThemeMode,
12     dynamicColors: Boolean,
13 ) {
14     BienestarEmocionalTheme(darkTheme = darkTheme.themeIsDark(), dynamicColors =
15         ↪ dynamicColors)
16     {
17         DestinationsNavHost(navGraph = NavGraphs.root)
18     }
19 }
```

Listado A.3: SistemaBienestarEmocionalApp.kt

A.2 API del servidor

Paquete raíz

```
1 import json
2
3 from flask import Flask, request
4
5 from src.endpoints import Endpoints
6 from src.utils import obtain_logger
7
8 # Init
```

```
9 app = Flask(__name__)
10 endpoints = Endpoints()
11 logger = obtain_logger("API")
12
13
14 @app.route('/community')
15 def community():
16     """Returns the averages of the community"""
17     return endpoints.community_endpoint()
18
19
20 @app.route('/user_data', methods=["POST"])
21 def user_data():
22     """Save user data from the app"""
23     request_data = request.json
24     logger.info('A request has been received')
25     return endpoints.user_data_endpoint(request_data)
26
27
28 @app.route('/daily_questionnaires', methods=["POST"])
29 def daily_questionnaires():
30     """Save user daily questionnaires data from the app"""
31     request_data = request.json
32     logger.info('A request has been received')
33     return endpoints.daily_questionnaires_endpoint(request_data)
34
35
36 @app.route('/one_off_questionnaires', methods=["POST"])
37 def one_off_questionnaires():
38     """Save user one_off questionnaires data from the app"""
39     request_data = request.json
40     logger.info('A request has been received')
41     return endpoints.one_off_questionnaires_endpoint(request_data)
42
43
44 @app.route('/bg_data', methods=["POST"])
45 def user_databg():
46     """Save user background data from the app"""
47     request_databg0 = request.json
48     request_databg = json.loads(request_databg0)
49     logger.info(f'A request has been received with the following data: {
50 ↪ request_databg}')
51     return endpoints.user_databg_endpoint(request_databg)
```

```
52
53 # If this script is being executed and not imported, deploy the API
54 if __name__ == '__main__':
55     app.run(host="0.0.0.0")
```

Listado A.4: api-prod.py

```
1 import json
2
3 from flask import Flask, request
4
5 from src.endpoints import Endpoints
6 from src.utils import obtain_logger
7
8 # Init
9 app = Flask(__name__)
10 endpoints = Endpoints(database="bienestar_emocional_test")
11 logger = obtain_logger("API-test")
12
13
14 @app.route('/community')
15 def community():
16     """Returns the averages of the community"""
17     return endpoints.community_endpoint()
18
19
20 @app.route('/user_data', methods=["POST"])
21 def user_data():
22     """Save user data from the app"""
23     request_data = request.json
24     logger.info(f'A request has been received with the following data: {request_data}
25     ↪ ')
26     return endpoints.user_data_endpoint(request_data)
27
28 @app.route('/daily_questionnaires', methods=["POST"])
29 def daily_questionnaires():
30     """Save user daily questionnaires data from the app"""
31     request_data = request.json
32     logger.info(f'A request has been received with the following data: {request_data}
33     ↪ ')
34     return endpoints.daily_questionnaires_endpoint(request_data)
```



```
34
35
36 @app.route('/one_off_questionnaires', methods=["POST"])
37 def one_off_questionnaires():
38     """Save user one_off questionnaires data from the app"""
39     request_data = request.json
40     logger.info(f'A request has been received with the following data: {request_data}
    ↪ ')
41     return endpoints.one_off_questionnaires_endpoint(request_data)
42
43 @app.route('/bg_data', methods=["POST"])
44 def user_databg():
45     """Save user background data from the app"""
46     request_databg0 = request.json
47     request_databg = json.loads(request_databg0)
48     logger.info(f'A request has been received with the following data: {
    ↪ request_databg}')
49     return endpoints.user_databg_endpoint(request_databg)
50
51
52
53 @app.route('/bg_data', methods=["POST"])
54 def user_databg():
55     """Save user background data from the app"""
56     request_databg0 = request.json
57     request_databg = json.loads(request_databg0)
58     logger.info(f'A request has been received with the following data: {
    ↪ request_databg}')
59     return endpoints.user_databg_endpoint(request_databg)
60
61
62 # If this script is being executed and not imported, deploy the API
63 if __name__ == '__main__':
64     app.run(
65         host="0.0.0.0",
66         port=5000, # TODO change to 5001 when these port would be available
67         debug=True
68     )
```

Listado A.5: api-test.py

```
1 import threading
```

```
2 from typing import Dict
3
4 from pymongo import MongoClient
5 from pymongo.results import InsertOneResult, InsertManyResult
6
7
8 class Database:
9     """
10     Contains a singleton instance of mongo database to interact with
11     """
12     _instance = None
13     _database = None
14     _lock = threading.Lock()
15
16     def __new__(cls, *args, **kwargs):
17         if cls._instance is None:
18             with cls._lock:
19                 # Another thread could have created the instance
20                 # before we acquired the lock. So check that the
21                 # instance is still nonexistent.
22                 if not cls._instance:
23                     cls._instance = super().__new__(cls)
24
25                     if "host" in kwargs:
26                         host = kwargs["host"]
27                     else:
28                         host = "localhost"
29
30                     if "port" in kwargs:
31                         port = kwargs["port"]
32                     else:
33                         port = 27017
34
35                     if "database" in kwargs:
36                         database = kwargs["database"]
37                     else:
38                         database = "bienestar_emocional"
39
40                     cls._database = MongoClient(host, port).get_database(database)
41             return cls._instance
42
43     @staticmethod
44     def insert_user_data(data: Dict) -> InsertOneResult:
45         """
```

```
46         Insert data into user_data collection
47         :param data: data to insert
48         :return: InsertOneResult of the execution
49         """
50         collection = Database._database.get_collection('user_data')
51         return collection.insert_one(data)
52
53     @staticmethod
54     def insert_daily_questionnaires(data: Dict) -> InsertManyResult:
55         """
56         Insert data into daily_questionnaires collection
57         :param data: data to insert
58         :return: InsertManyResult of the execution
59         """
60         collection = Database._database.get_collection('daily_questionnaires')
61
62         processed_data = []
63
64         for key in data["data"]:
65             for element in data["data"][key]:
66                 values = element
67                 values.update({"type": key, "userId": data["userId"]})
68                 values.pop("id")
69                 processed_data.append(values)
70         return collection.insert_many(processed_data)
71
72     @staticmethod
73     def insert_one_off_questionnaires(data: Dict) -> InsertManyResult:
74         """
75         Insert data into one_off_questionnaires collection
76         :param data: data to insert
77         :return: InsertManyResult of the execution
78         """
79         collection = Database._database.get_collection('one_off_questionnaires')
80
81         processed_data = []
82
83         for key in data["data"]:
84             for element in data["data"][key]:
85                 values = element
86                 values.update({"type": key, "userId": data["userId"]})
87                 values.pop("id")
88                 processed_data.append(values)
89         return collection.insert_many(processed_data)
```

```

90
91 @staticmethod
92 def get_daily_questionnaires_average(start: int, end: int) -> Dict:
93     """
94         Obtain the average score of all scored measures filter by timestamps
95         :param start: filter by greater than or equals this value
96         :param end: filter by less than this value
97         :return: Dict with measure and score
98     """
99     collection = Database._database.get_collection('daily_questionnaires')
100     query = collection.aggregate(
101         [
102             {
103                 "$match":
104                     {
105                         "createdAt":
106                             {
107                                 "$gte": start,
108                                 "$lt": end
109                             },
110                         "score":
111                             {
112                                 "$gte": 0,
113                             }
114             },
115             {
116                 "$group":
117                     {
118                         "_id": "$type",
119                         "average":
120                             {
121                                 "$avg": "$score"
122                             }
123                     }
124             }
125         ]
126     )
127     return {item["_id"]: round(item["average"], 2) for item in query}
128
129
130 @staticmethod
131 def insert_user_databg(databg: Dict) -> InsertOneResult:
132     """
133     Insert background data into user_databg collection

```

```

134         :param databg: data to insert
135         :return: InsertOneResult of the execution
136         """
137         collection = Database._database.get_collection('user_databg')
138         return collection.insert_one(databg)

```

Listado A.6: database.py

```

1  from typing import Dict
2
3  from flask import Response, jsonify
4
5  from src.database import Database
6  from src.response.daily_questionnaires_response import
    ↪ build_daily_questionnaires_response
7  from src.response.one_off_questionnaires_response import
    ↪ build_one_off_questionnaires_response
8  from src.response.user_data_response import build_user_data_response
9  from src.time import get_timestamps_yesterday, get_timestamps_current_week,
    ↪ get_timestamps_last_seven_days
10 from src.response.user_databg_response import build_user_databg_response
11 from src.utils import data_not_empty
12 from src.validator.daily_questionnaires_validator import DailyQuestionnairesValidator
13 from src.validator.one_off_questionnaires_validator import
    ↪ OneOffQuestionnairesValidator
14 from src.validator.user_data_validator import UserDataValidator
15 from src.validator.user_databg_validator import UserDataBgValidator
16
17
18 class Endpoints:
19     def __init__(self, **kwargs):
20         self.udv = UserDataValidator()
21         self.dqv = DailyQuestionnairesValidator()
22         self.ooqv = OneOffQuestionnairesValidator()
23         self.udbgv = UserDataBgValidator()
24
25         if "database" in kwargs:
26             self.database = Database(database=kwargs["database"])
27         else:
28             self.database = Database()
29
30     def user_data_endpoint(self, request_data: Dict) -> Response:

```

```

31     if self.udv.validate(request_data):
32         if data_not_empty(request_data["data"]):
33             self.database.insert_user_data(request_data)
34             response = build_user_data_response(request_data["data"])
35             return jsonify(response)
36     else:
37         return Response("Bad request", 400)
38
39 def daily_questionnaires_endpoint(self, request_data: Dict) -> Response:
40     if self.dqv.validate(request_data):
41         if data_not_empty(request_data["data"]):
42             self.database.insert_daily_questionnaires(request_data)
43             response = build_daily_questionnaires_response(request_data["data"])
44             return jsonify(response)
45     else:
46         return Response("Bad request", 400)
47
48 def one_off_questionnaires_endpoint(self, request_data: Dict) -> Response:
49     if self.ooqv.validate(request_data):
50         if data_not_empty(request_data["data"]):
51             self.database.insert_one_off_questionnaires(request_data)
52             response = build_one_off_questionnaires_response(request_data["data"])
53             return jsonify(response)
54     else:
55         return Response("Bad request", 400)
56
57 def community_endpoint(self) -> Response:
58     response = {}
59
60     # Obtain all timestamps
61     yesterday_timestamps = get_timestamps_yesterday()
62     current_week_timestamps = get_timestamps_current_week()
63     last_seven_days_timestamps = get_timestamps_last_seven_days()
64
65     # Yesterday average
66     response["yesterday"] = self.database.get_daily_questionnaires_average(
67         yesterday_timestamps[0],
68         yesterday_timestamps[1]
69     )
70
71     # Current week average by day
72     response["current_week"] = [
73         self.database.get_daily_questionnaires_average(day[0], day[1]) for day in
↪ current_week_timestamps

```

```

74         ]
75
76         # Last seven days average
77         response["last_seven_days"] = self.database.get_daily_questionnaires_average(
78             last_seven_days_timestamps[0],
79             last_seven_days_timestamps[1]
80         )
81
82         return jsonify(response)
83
84     def user_databg_endpoint(self, request_databg: Dict) -> Response:
85         if self.udbgv.validate(request_databg):
86             if data_not_empty(request_databg["databg"]):
87                 self.database.insert_user_databg(request_databg)
88                 response = build_user_databg_response(request_databg["databg"])
89                 return jsonify(response)
90             else:
91                 return Response("Bad request", 400)

```

Listado A.7: endpoints.py

```

1  from datetime import datetime, timedelta
2
3
4  def __datetime_to_millisecond_timestamp(date: datetime) -> int:
5      """ Pass datetime to timestamp in milliseconds"""
6      return int(date.timestamp() * 1000)
7
8
9  def __get_start_day(date: datetime) -> int:
10     """ Get the timestamp of the start of the day"""
11     return __datetime_to_millisecond_timestamp(
12         date.replace(
13             hour=0,
14             minute=0,
15             second=0,
16             microsecond=0
17         )
18     )
19
20
21 def __get_end_day(date: datetime) -> int:

```

```
22     """ Get the timestamp of the end of the day"""
23     return __datetime_to_millisecond_timestamp(
24         date.replace(
25             hour=23,
26             minute=59,
27             second=59,
28             microsecond=999999
29         )
30     )
31
32
33 def get_timestamps_yesterday():
34     """ Get the timestamp of the start and end of yesterday"""
35     yesterday = datetime.now() - timedelta(days=1)
36     return __get_start_day(yesterday), __get_end_day(yesterday)
37
38
39 def get_timestamps_current_week():
40     """Get start and end timestamp of each day of the current week"""
41     now = datetime.now()
42     monday = now - timedelta(days=now.weekday())
43     sunday = monday + timedelta(days=6)
44     result = []
45
46     day = monday
47     while day ≤ sunday:
48         result.append((__get_start_day(day), __get_end_day(day)))
49         day += timedelta(days=1)
50
51     return result
52
53
54 def get_timestamps_last_seven_days():
55     """ Get the timestamp of the start of seven days before and the end of yesterday
56     ↪ """
57     now = datetime.now()
58     first_day = now - timedelta(days=7)
59     last_day = now - timedelta(days=1)
60     return __get_start_day(first_day), __get_end_day(last_day)
```

Listado A.8: time.py


```
1 import logging
2 import string
3 import sys
4 from pathlib import Path
5 from typing import Collection, Dict
6
7
8 def inner_contained_fully_outer(inner: Collection, outer: Collection) -> bool:
9     """
10     Checks if all elements of inner collection are present in outer collection
11     :param inner: Elements to check in outer
12     :param outer: Collection that should contain all inner elements
13     :return: true if all elements are present, otherwise false
14     """
15     return all(element in outer for element in inner)
16
17
18 def obtain_logger(name: string):
19     """
20     Builds a logger that writes on stdout and a file placed in logs' folder
21     :param name: Name of the logger
22     :return: Logger ready to use
23     """
24     logger = logging.getLogger(name)
25     logger.setLevel(logging.DEBUG)
26
27     standard_output_handler = logging.StreamHandler(sys.stdout)
28     file_handler = logging.FileHandler(Path(__file__).parent / "..../logs" / f'{name
↵ }.log')
29
30     formatter = logging.Formatter(
31         '[(asctime)s.%(msecs)d]\t %(name)s - %(levelname)s \t[(name)s.%(funcName)s
↵ :%(lineno)d]\t %(message)s',
32         datefmt='%d/%m/%Y %H:%M:%S'
33     )
34
35     standard_output_handler.setFormatter(formatter)
36     file_handler.setFormatter(formatter)
37
38     logger.addHandler(standard_output_handler)
39     logger.addHandler(file_handler)
40
41     return logger
```

```
42
43
44 def data_not_empty(data: Dict):
45     """
46     Check if any of values of data is not an empty list
47     :param data: dict to check
48     :return: true if any of these values is not empty, false if all values are empty
49     """
50     values_empty = list(map(lambda values: len(values) > 0, data.values()))
51     return any(values_empty)
```

Listado A.9: utils.py

