Desarrollo de Aplicaciones Multiplataforma
Curso 2020/2021

LastWords

Proyecto de Fin de Ciclo

Belal Benmousa
Antonio Tomás

Índice de contenidos

A Blanca, porque sin su ayuda y apoyo incondicional no habría me habría sido posible conseguir todo lo que he conseguido. A Lorenzo y Juan Fran, que me han soportado cuando les hablaba de programación. A mi compañero y amigo Belal, por todas sus noches sin dormir para que este proyecto saliera adelante. A mis compañeros del grupo B (de Buenos), por compartir sus conocimientos. A Sandra López, Rubén Arroyo, Joaquim Sabrià, Fernando Porrino, Daniel Martínez, Óscar Torrente, Jordi Montserrat, Jordi Hernández, Gerard Falcó y David Rodríguez, a los que estaré eternamente agradecido por haberme enseñado todo lo que sé.

Antonio Tomás

En primer lugar me gustaría agradecer a mi compañero Antonio Tomás el cual conocí este año, que más que un compañero lo considero un gran amigo, él tuvo esta gran idea de proyecto y gracias a él se ha podido llevar a cabo. Seguidamente, me gustaría también agradecer a todos mis compañeros y en especial al profesorado de DAM B, es difícil encontrar un centro con unos profesores tan atentos y dedicados con lo que hacen, gracias a Jordi, Fer, David, Dani y en especial a Gerard, ya que es su último año en el centro y he podido tener el honor de tenerlo como profesor. Les estaré eternamente agradecido y les deseo todo lo mejor.

Belal Benmousa

Datos del proyecto y resumen

Resumen del proyecto:

En la actualidad, no existen herramientas tecnológicas que permitan a las personas el tratamiento de su información más sensible cuando fallecen. Cuando las personas llegan al final de su vida, no hay manera de decidir qué se debe hacer con todas las cuentas de todos los servicios digitales a los que dicha persona está suscrita si no se opta por un testamento tradicional o alguna otra opción parecida. Tampoco existe la posibilidad de automatizar el envío de un último mensaje dirigido a sus seres queridos.

Last Words es una aplicación para dispositivos móviles que quiere dar a las personas la capacidad de decidir cuáles serán sus últimas palabras. Nadie está exento de sufrir un accidente que merme sus capacidades cognitivas o, incluso peor, que le provoque la muerte. Y lo peor de esta posibilidad es que no es posible saber cuándo ocurrirá. En un momento de la historia de la humanidad en que la tecnología está en auge y una pandemia azota el planeta, no existen muchas opciones que permitan manejar nuestra información cuando ya no estemos. Algunas empresas tecnológicas, como Facebook, permiten que ciertas personas autorizadas se hagan cargo de la cuenta que tenemos en el servicio o crear un obituario online, pero en ningún caso se va más allá.

El objetivo de Last Words es, por tanto, permitir a las personas crear mensajes personalizados con texto, imágenes, vídeos o archivos que se almacenarán de forma segura y que se enviarán a los contactos que elijan después de un tiempo que ellas mismas habrán establecido durante la creación del mensaje.

Palabras clave:

Android, Flutter, Java, Spring Boot, Heroku, muerte, mensajes, información.

Abstract:

Currently, there are no technological tools that allow people to deal with their most sensitive information when they pass away. When people reach the end of their lives, there is no way to decide what should be done with all the accounts of all the digital services to which that person is subscribed if they do not opt for a traditional will or some other similar option. There is also no way to automate the sending of a last message addressed to loved ones.

Last Words is an application for mobile devices that wants to give people the ability to decide what their last words will be. No one is exempt from suffering an accident that undermines their cognitive abilities or, even worse, causes them to

die. And the worst thing about this possibility is that it is not possible to know when it will happen. At a time in human history when technology is booming and a pandemic is sweeping the planet, there are not many options for managing our information after we are gone. Some technology companies, such as Facebook, allow certain authorized people to take over the account we have on the service or create an online obituary, but in no case does it go beyond that.

The aim of Last Words is therefore to allow people to create personalized messages with text, images, videos or files that will be stored securely and sent to the contacts of their choice after a period of time that they themselves will have set during the creation of the message.

Keywords:
Android, Flutter, Java, Spring Boot, Heroku, death, messages, information.

Introducción

La sociedad está cada vez más acostumbrada a utilizar todo tipo de servicios digitales. Es habitual registrarse y compartir los datos personales con todo tipo de servicios online: tiendas, redes sociales, periódicos digitales, aplicaciones de todo tipo... Estamos conectados al mundo a través del dispositivo móvil que todo el mundo tiene en el bolsillo.

Con respecto a la gestión de la información personal en Internet, la legislación avanza de manera mucho más lenta que la tecnología, lo que da lugar a numerosas lagunas que permiten que nuestra información vague por Internet sin ser conscientes en muchas ocasiones. Algunos avances, como el derecho al olvido, permiten al usuario solicitar a los motores de búsqueda la supresión de la divulgación de su información personal. Sin embargo, esta posibilidad tiene una limitación que en muchas ocasiones no se tiene en cuenta: el usuario tiene que estar vivo y tiene que ser el que ejerza este derecho.

Esta falta de control de la información, unida al auge de los dispositivos móviles, es la que ha propiciado la idea que se pretende desarrollar en este proyecto. Las empresas tecnológicas y los servicios digitales que éstas desarrollan están enfocándose cada vez más en obtener nuestros datos para conseguir réditos económicos. Sin embargo, no existen soluciones digitales para gestionar nuestros qué pasa con nuestros datos después de fallecer.

Last Words es un proyecto que pretende poner solución a este problema, dando la oportunidad a los usuarios de poder crear mensajes personalizados que se almacenarán de manera segura y que se enviarán a los contactos que elijan una vez fallezcan. Para conseguirlo, el objetivo es desarrollar una aplicación móvil

que actúe a la vez como interfaz y como "dispositivo de hombre muerto" o "dispositivo de presencia". Esta aplicación, a su vez, se comunicará diariamente con un servidor, que será el encargado de enviar los mensajes del usuario a los destinatarios elegidos una vez se cumplan las condiciones para ello.

Las posibilidades de este proyecto son muy interesantes, y permitirían a los usuarios crear mensajes con todo tipo de información que de otra manera se perdería tras el fallecimiento. Por ejemplo, las personas que hicieran uso de la aplicación podrían redactar unas últimas palabras para sus seres queridos y acompañarlas de alguna imagen o vídeo que rememorara un momento especial. O el usuario podría crear un mensaje con un archivo PDF que contuviera las credenciales de los diversos servicios a los que estuviera suscrito para que aquellas personas que elija pudieran gestionarlas sin problemas. Las posibilidades serían innumerables y permitirían a los usuarios de la aplicación gestionar en vida qué pasará con sus datos más importantes cuando fallezcan.

Objetivos

Objetivos generales

Durante el desarrollo de esta aplicación son varios los objetivos marcados, tanto tecnológicos como funcionales.

En primer lugar, y como objetivos tecnológicos, se pretende utilizar Flutter como lenguaje para programar el Front-end de la aplicación. Esto implica aprender desde cero un lenguaje de programación muy joven y en constante cambio, en el que las formas de hacer las cosas poco tienen que ver con lo aprendido durante el Ciclo Formativo. Para la interacción entre la aplicación y la persistencia de los datos, el objetivo es crear una API utilizando Spring Boot. Este framework de Java nos permitirá crear desde cero una API que comunique la aplicación móvil con una base de datos alojada en un servidor. En cuanto a la persistencia de los datos, el objetivo es usar PostgreSQL para almacenar los datos de los usuarios de manera segura.

En segundo lugar, y como objetivos funcionales, la intención del equipo de desarrollo es que la aplicación permita a los usuarios registrarse, crear nuevos mensajes con texto, imágenes, vídeos y/o archivos (que se guardarán en el servidor de manera segura) y que estos se envíen pasado un tiempo estipulado previamente. Para conseguir este último paso, la aplicación y el servidor donde están alojados los datos se deben poder comunicar una vez al día para que, en caso de que la comunicación no se pueda establecer en ninguno de los días, el servidor envíe los mensajes a los destinatarios. Este último paso, el del envío

automático de mensajes tras un tiempo estipulado, creemos que está fuera del alcance de este proyecto, pero es un aspecto fundamental que se desarrollará en el futuro más próximo utilizando conocimientos de Administración de Sistemas Informáticos en Red.

Objetivos específicos

A continuación, se repasa la lista de objetivos específicos que debe cumplir el proyecto:

- Desarrollar una aplicación funcional en Flutter
- Desarrollar una API funcional en Spring Boot
- Comunicar la aplicación con la API
- Almacenar los datos generados por la aplicación de manera segura en una base de datos
- Permitir a los usuarios de la aplicación registrarse e iniciar sesión
- Permitir a los usuarios de la aplicación modificar sus datos personales

Desarrollar una aplicación móvil

Uno de los objetivos principales del proyecto es la creación de una aplicación para dispositivos móviles. La tecnología elegida para este propósito es el framework Flutter, que permite compilar el mismo código en varias plataformas diferentes. Esta aplicación debe poder brindar al usuario la posibilidad de registrarse, iniciar sesión, modificar su perfil y crear, consultar, editar y eliminar mensajes que pueden contener texto, imagen, vídeo y ficheros. Además, estos mensajes deben poder enviarse de manera segura a un servidor donde se alojarán hasta la eliminación o el envío a los destinatarios.

Desarrollar una API

Otra de las piezas fundamentales de este trabajo es la creación de una API que permita a la aplicación móvil comunicarse con la base de datos que almacene los datos de los usuarios. Debido a sus proyecciones profesionales, la tecnología elegida para este menester es Spring Boot. La API debe ser capaz de recibir peticiones de la aplicación y enviar la información, debidamente cifrada, a la base de datos. Entre otras, esta interfaz debe poder aceptar peticiones de registro, de inicio de sesión y de creación, consulta, edición y eliminación de mensajes de los usuarios.

Garantizar la persistencia de datos

El tercer elemento esencial para el funcionamiento de la aplicación es la persistencia de los datos. Para ello, se ha elegido el SGBD PostgreSQL debido a su amplio soporte de la comunidad y a los conocimientos previos de este sistema. El

sistema gestor de bases de datos debe poder almacenar en diversas tablas la información relativa a los usuarios, sus mensajes y los contactos a los que van dirigidos.

Integrar la aplicación con la API y la base de datos

El siguiente objetivo, tras la creación de los tres elementos anteriores, es la integración de estos tres componentes. Esto permitiría obtener un servicio funcional, pero todavía incompleto.

Crear un servicio de envío automático de mensajes

Tras integrar la aplicación con la API y la base de datos, el siguiente paso es encontrar la manera de desarrollar un sistema que compruebe si el teléfono móvil ha estado inactivo durante un tiempo determinado. Este servicio también debe ser capaz de comunicarse con la base de datos para obtener y enviar la información a los destinatarios si el dispositivo móvil ha permanecido inactivo durante el tiempo que el usuario estableció durante la creación del mensaje.

Establecer un sistema de avisos por SMS y/o email

Como medida de seguridad y privacidad, se debe establecer un sistema de avisos que informe a usuarios y destinatarios de los diversos acontecimientos que pueden suceder en la aplicación. En cuanto a los usuarios, el sistema de información debe poder avisar al usuario de que si no se detecta actividad en el dispositivo móvil en el plazo de unos días el mensaje se enviará a los destinatarios elegidos. En lo que respecta a los destinatarios, el sistema de información debe ser capaz de enviar un SMS informando de que ha sido elegido como destinatario de un mensaje y que si quiere poder recibirlo es necesario que disponga de la aplicación o esté registrado.

Implantar un sistema de pago por mensaje

Para que la aplicación y los servicios asociados a ella sean viables económicamente se deberá implantar un sistema de pago por mensaje. Debido a los costes derivados del mantenimiento del código, de los servidores y de los servicios, es imprescindible establecer un modelo económico que sea rentable y que permita la subsistencia de la aplicación. Debido a la temática del proyecto, creemos que establecer un pago simbólico por mensaje es mucho más acertado moralmente que introducir publicidad. Esto permitiría además llegar a aquellas personas realmente interesadas en las funcionalidades de la aplicación.

Entorno del proyecto

Contexto

¿Qué pasa con nuestros datos después de la muerte? La tecnología avanza a pasos agigantados. Escuchar a otros hablar más de una vez al día de criptomonedas, de coches eléctricos y autónomos o de robots que aspiran, cocinan y hasta bailan es algo mucho más que habitual. La potencia de cálculo de los procesadores traspasa cada año los límites impuestos poco tiempo atrás. Muchas fábricas se ven condicionadas por la escasez de componentes electrónicos y algunas, como Seat, se ven obligadas a dejar de fabricar coches y prescindir, aunque sea de manera temporal, de algunos de sus trabajadores. Los informativos han incluido entre sus noticias el vídeo viral de turno de la red social de moda.

La tecnología, en definitiva, ha dejado de ser un elemento ajeno al grueso de la población para convertirse en una parte fundamental en la vida de las personas. Sin embargo, aún hay áreas en las que la tecnología ha puesto tímidamente sus manos, y una de estas áreas es la muerte. Sí que se invierten cientos de miles de millones en evitar el último e inevitable paso que damos en la vida, pero pocos son los esfuerzos tecnológicos que se emplean en gestionar aquello que dejamos atrás cuando ya no estamos.

Justificación

La motivación para elegir esta idea como proyecto ha sido, principalmente, la falta de soluciones existentes para este propósito. La idea de poder desarrollar una herramienta que permita a los usuarios crear mensajes personalizados que se enviarán en algún momento del futuro también nos resultó atractiva e interesante.

La elección de Flutter y Spring Boot para la creación del Front-end y el Back-end respectivamente responde al amplio abanico de posibilidades que estas dos tecnologías brindan en el desarrollo de aplicaciones para móviles.

Por un lado, Flutter permite crear vistosas aplicaciones para iOS, Android y web con un sólo código fuente, lo que sin duda hace que la aplicación esté mucho más preparada para futuras ampliaciones. También nos brinda la posibilidad de lanzar la aplicación no sólo en Google Play Store, sino también en la App Store de Apple.

En la parte del Back-end, Spring Boot es una potente herramienta que permite crear una API RESTful que controle todas aquellas acciones que necesita

la aplicación para almacenar y gestionar toda la información que los usuarios van generando.

Soluciones existentes

Tras analizar la existencia de soluciones similares a Last Words, hemos llegado a la conclusión de que no hay ninguna aplicación similar en el mercado. Las herramientas digitales que tratan algún tipo de información de las personas fallecidas se centran en la creación de memorias y obituarios. De entre este tipo de herramientas, podemos destacar algunas como Memories.net o Onceivegone.com. Sin embargo, y como hemos comentado, estas herramientas no permiten a los usuarios la creación de mensajes ni su envío automático pasado un tiempo, como sí pretende Last Words.

Alcance

Situación actual

A lo largo del curso, un integrante del equipo ha ido desarrollando la aplicación utilizando Figma, Java y Android Studio. A nivel de interfaz, el diseño realizado con Figma está completo al 95%, faltando sólo algunos ajustes para hacer la aplicación más atractiva.

La implementación del diseño realizado en Figma en la interfaz de la aplicación está completa al 85% en la aplicación Java. Hay algunas características del diseño que no se han podido llevar a cabo por falta de tiempo y de conocimientos.

Las funcionalidades en esta versión de la aplicación están implementadas al 60%. Faltan algunas funcionalidades como el guardado de imágenes, vídeos y archivos en la base de datos y el envío automático de mensajes transcurridos los días establecidos por el usuario.

Alcance y posibles obstáculos

Aunque al inicio del proyecto la aplicación está completa al 60% utilizando Java, hemos querido usar un nuevo enfoque utilizando Flutter, un SDK utilizado para desarrollar aplicaciones móviles y web desarrollado por Google. Este framework utiliza Dart, un lenguaje de programación creado también por Google.

La decisión de utilizar Flutter y comenzar el desarrollo de nuevo, dejando de lado todo lo desarrollado hasta ahora, ha sido tomada en base a las ventajas que

ofrece esta herramienta. Entre las más destacadas, las más importantes para tomar la decisión de cambiar Java por Flutter han sido:

- Un código, múltiples plataformas: aunque Java es un lenguaje multiplataforma, depende de que la máquina virtual sea compatible con la plataforma donde se quiere ejecutar el código. En el mercado de las aplicaciones móviles, iOS y Android son los dos actores principales, con una cuota de mercado cercana al 100%. Aunque las aplicaciones desarrolladas en Java son 100% compatibles con Android, no lo son con iOS, por lo que estaríamos perdiendo una parte del mercado si hubiésemos seguido con el desarrollo en Java.

- Rendimiento nativo: Flutter ofrece a los desarrolladores un rendimiento cercano al nativo, ya que se compila a código máquina ARM o x86 utilizando los compiladores nativos de Dart. Flutter en su configuración básica utiliza widgets Material o Cupertino que son propios e independientes del sistema en el que se ejecutan, lo que permite renderizarlos directamente en la GPU. Esto proporciona un rendimiento muy elevado. Otras soluciones multiplataforma, como React Native, en su configuración básica ofrecen sólo los componentes nativos del sistema, lo que supone instalar componentes no nativos si lo que se busca es personalizar la interfaz y el comportamiento. Esto, unido a que además utiliza una capa JavaScript, provoca que el rendimiento pueda ser ligeramente peor.

- Código más simple y fácil de entender: Java y Android son herramientas muy sólidas, robustas y asentadas en el mercado. Sin embargo, esto provoca ciertas limitaciones a la hora de desarrollar aplicaciones. Por ejemplo, el código necesario para desarrollar una aplicación utilizando Java y Android se vuelve largo y complicado de entender, con multitud de clases y métodos que hacen necesario, en nuestra opinión, tener unos conocimientos sólidos de estas herramientas. Flutter y Dart, por su parte, simplifican el código necesario para desarrollar las mismas aplicaciones, lo que hace que el código sea más inteligible.

La decisión de empezar de cero conlleva, sin embargo, múltiples obstáculos que deberemos afrontar.

Por un lado, aprender Dart y Flutter en las postrimerías del curso, desechando gran parte del trabajo realizado hasta ahora, conlleva tener mucho menos tiempo para implementar funcionalidades. A lo largo del curso y de sus asignaturas hemos ido aprendiendo las particularidades de Java y nos hemos ido haciendo con el lenguaje de manera óptima gracias a su uso intensivo. Con Dart y Flutter la

situación es completamente opuesta: no hay ninguna asignatura que explique los fundamentos básicos y cada línea de código creada es aprendizaje para el equipo desarrollador. Cambiar Java y Android por Dart y Flutter ha sido una decisión arriesgada y sin duda nos pondrá en aprietos en más de una ocasión, pero creemos que merece la pena intentar aprender estas herramientas tan interesantes.

Por otro lado, la decisión de crear desde cero una API con Spring Boot para comunicar la aplicación con la base de datos en lugar de utilizar Google Firebase también mermará el tiempo disponible para crear Last Words. Sin embargo, y sabiendo que Spring Boot es una herramienta muy utilizada en el mundo laboral, creemos que es más que adecuado aprender a desarrollar una API con este framework de Java.

En relación al punto anterior, la creación de una base de datos desde cero que se comunique con la aplicación mediante la API en lugar de utilizar Firebase también puede generar obstáculos que deberemos salvar, aunque el alcance de los mismos en el momento de redactar la situación inicial del proyecto es desconocido.

Por último, creemos que la funcionalidad de enviar los mensajes a los destinatarios de manera automatizada deberá quedar fuera del proyecto debido a que plantea unos retos que no seremos capaces de asumir, dado que el resto de componentes de la aplicación serán desarrollados desde cero y partiendo de unos conocimientos nulos sobre las herramientas que se van a utilizar.

Metodología, validación y herramientas de seguimiento

La metodología elegida para el desarrollo de este proyecto ha sido una interpretación libre de Scrum. En lugar de seguir un desarrollo tradicional, con documentación extensa y unos plazos muy marcados, se ha decidido realizar reuniones diarias por Discord y comunicaciones frecuentes por WhatsApp.

Para realizar compartir los progresos y trabajar de manera colaborativa se ha elegido Git, utilizando GitHub como lugar de almacenamiento de los repositorios. Las ventajas de este sistema son claras, ya que además del trabajo colaborativo permite salvaguardar el código en caso de que las máquinas utilizadas para desarrollar el código fuente sufran algún tipo de fallo.

Para llevar a cabo el seguimiento de las tareas se ha optado por utilizar una hoja de cálculo de Google, en la que se comparan las horas estimadas durante la planificación del proyecto con las que se han realizado hasta el momento.

Presupuesto

Elementos de la estimación inicial

Elemento	Cantidad	Precio unitario	Precio Total
Hosting Heroku	1	94.90€/año	94.90€
Dominio lastwords.es	1	6.99€/año	6.99€
Cuota desarrollador Google Play	2	25€	50€
Cuota desarrollador App Store	2	99€	198€
Cuota subida app a Google Play	1	25€	25€
Honorarios diseñador	20	20€/hora	400€
Honorarios programador	90	20€/hora	1800€
		Total	2.574,89€

Justificación de los costes estimados

Para la estimación de costes se han consultado diversas soluciones de hosting, así como los espacios para desarrolladores de Google Play Store y Apple App Store. También se han tenido en cuenta las estimaciones que algunos sitios web de trabajo y de programación hacen sobre el salario de los programadores.

Plan de control del presupuesto

Para ir controlando nuestro presupuesto utilizaremos una hoja de cálculo de las que nos facilita Google, con dicho documento, podemos ir visualizando la trayectoria de nuestro presupuesto para así poder prever alguna futura necesidad de aumento del mismo razones de mejoras o productividad.

Valoración de la viabilidad económico-financiera

La viabilidad económica de este proyecto está supeditada a la publicación de la aplicación en las diferentes tiendas digitales de fabricantes como Google o Apple.

En el caso de seguir adelante con el proyecto después de terminar el CFGS, el proyecto sólo puede ser viable si se cobra un precio simbólico por mensaje de 5

euros. Teniendo en cuenta que los servidores deben permanecer activos para poder gestionar el envío de mensajes, es fundamental obtener un rédito económico de cada uno de ellos para poder asumir los costes del servicio.

La posibilidad de mantener los servicios haciendo uso de la publicidad es inviable, ya que moralmente consideramos desacertado incluir publicidad en una aplicación que pretende gestionar información sensible.

Planificación temporal

Fases del proyecto

El desarrollo del proyecto se llevará a cabo en varias fases. En primer lugar, en la fase de análisis se recogerán los requisitos, funcionales y no funcionales, que debe cumplir la aplicación para considerar que está terminada. También se marcarán los objetivos que se deben cumplir y se analizarán las distintas tecnologías disponibles para alcanzarlos.

El diseño de los distintos componentes de la aplicación y su entorno será el siguiente paso en el proyecto. Durante esta fase se decidirá la arquitectura de la aplicación y de los servicios que ésta necesita para funcionar, así como la estructura de los datos y su almacenamiento y persistencia. Debido a que el diseño de la interfaz ya se realizó en un momento anterior del curso, el tiempo dedicado al diseño de la interfaz se dedicará a pulir aspectos que eran mejorables. Por último, se debe diseñar cómo implementar técnicas de criptografía para proteger los datos de los usuarios.

En la fase de desarrollo se trabajará en la creación del código necesario para crear la aplicación, así como para crear el Back-end que permita la correcta comunicación entre la aplicación para Android y los diferentes servidores, servicios y bases de datos que son imprescindibles para que funcione correctamente.

Si el tiempo lo permite, la última fase relacionada con la aplicación será la de pruebas. En caso de ser posible se realizarán distintas pruebas, priorizando aquellas de usabilidad que permitan obtener feedback por parte de usuarios ajenos al desarrollo de software.

Planificación inicial

En la siguiente tabla se puede observar la estimación de horas necesarias para desarrollar los objetivos marcados en el proyecto.

	Horas Planificadas	Horas Reales	Estado	Observaciones
Proyecto Integrado	198	0	-198	
Aplicación	178	0	-178	
Análisis	24	0	-24	
Requisitos	2		-2	
Objetivos	2		-2	
Tecnología	20		-20	
Diseño	20	0	-20	
Arquitectura - Diseño de clases	4		-4	
Interfaz - Mockups	2		-2	
Persistencia - Diseño conceptual	4		-4	
Criptografía	10		-10	
Desarrollo	128	0	-128	
Estrategia de desarrollo	2		-2	
Interfaz	20		-20	
Backend	90		-90	
BD (SQL)	16		-16	
Pruebas	6	0	-6	
Usabilidad	6		-6	
Memoria	12	0	-12	
Redacción	10		-10	
Revisión	2		-2	
Presentación oral	8	0	-8	
Presentación oral	2		-2	
Materiales (Power Point, etc.)	6		-6	

Soluciones a desviaciones eventuales

Como solución a posibles problemas y desviaciones trataremos de buscar información en internet, apoyándonos siempre en el conocimiento de los profesores. En el inicio del proyecto no es posible saber todas las circunstancias que influirán en el desarrollo, pero sí podemos prever que habrá problemas relacionados con las tecnologías elegidas, que son nuevas en su mayoría.

Justificación de la finalización del tiempo

La planificación elegida está condicionada por la elección de Flutter y Spring Boot, dos tecnologías nuevas que requerirán de un aprendizaje constante para poder obtener un resultado aceptable del proyecto.

Punto de control

Cambios respecto a la planificación inicial

Después de un mes, el proyecto no avanza todo lo rápido que sería deseable. Las entregas de otras asignaturas, unido al lento avance tanto en la parte de Front-end como en la parte de Back-end están condicionando el desarrollo del proyecto. En este momento se decide dejar fuera del alcance del proyecto aquellas funcionalidades que más dependen de la rama de administración de sistemas.

En este punto, el método para encriptar y desencriptar los mensajes no está decidido y está suponiendo un reto saber cómo solucionar este problema.

Consecuencias de los cambios respecto a los objetivos y desarrollo del proyecto

En este punto, se empieza a asumir que el proyecto es demasiado ambicioso. Sin embargo, la voluntad del equipo desarrollador es continuar con el desarrollo de esta propuesta una vez termine el curso, ya que tiene potencial para ser un producto viable.

Situación del proyecto en el punto de control

Después de un mes, la aplicación objeto de este proyecto está todavía en una fase temprana de desarrollo. Las pantallas que conforman su interfaz no están terminadas, las funcionalidades no se han empezado a desarrollar y la API y la base de datos están lejos todavía de poder desarrollarse. Sin embargo, y conforme el equipo desarrollador se vaya liberando de algunas tareas pendientes, creemos posible alcanzar un porcentaje elevado de los objetivos marcados.

	Horas Planificadas	Horas Reales	Estado	Observaciones
Proyecto Integrado	198	54	-144	
Aplicación	178	54	-124	
Análisis	24	6	-18	
Requisitos	2	2	0	

Objetivos	2	2	0	
Tecnología	20	2	-18	
Diseño	20	12	-8	
Arquitectura - Diseño de clases	4	2	-2	
Interfaz - Mockups	2	2	0	
Persistencia - Diseño conceptual	4	2	-2	
Criptografía	10	6	-4	
Desarrollo	128	30	-98	
Estrategia de desarrollo	2	2	0	
Interfaz	20	12	-8	
Backend	90	12	-78	
BD (SQL)	16	4	-12	
Pruebas	6	6	0	
Usabilidad	6	6	0	
Memoria	12	0	-12	
Redacción	10	0	-10	
Revisión	2	0	-2	
Presentación oral	8	0	-8	
Presentación oral	2	0	-2	
Materiales (Power Point, etc.)	6	0	-6	

Planificación final

Agotado el tiempo de desarrollo, los objetivos marcados al inicio del proyecto no se han alcanzado en su mayoría. El continuo aprendizaje de las tecnologías elegidas ha provocado un retraso irrecuperable en la implementación de las funcionalidades principales, lo que ha conllevado que la aplicación se encuentre en un estado no óptimo de terminación.

Diversos problemas surgidos de comenzar desde cero la creación de la aplicación utilizando un lenguaje de programación y un framework totalmente desconocidos como son Dart y Flutter, unido al cambio de paradigma que el uso de estas tecnologías requieren, ha supuesto que el tiempo de desarrollo haya sido a su vez un arduo proceso de aprendizaje. Problemas como el control de estados dentro de la aplicación, que a su vez afecta al uso de variables entre los ámbitos de la misma, han sido todo un desafío que poco a poco se han conseguido superar. Sin embargo, y a pesar de los avances logrados en los últimos compases del curso, aún queda mucho trabajo por hacer para dar por terminada la aplicación.

Por otro lado, la decisión de crear una API desde cero utilizando Spring Boot también ha supuesto todo un desafío. La complejidad de Java y de las posibilidades que brinda Spring para la creación de APIs han requerido de mucho esfuerzo, sobre todo en lo que se refiere a autenticación mediante el uso de tokens, una característica moderna que ha sido difícil de implementar sin los conocimientos adecuados.

A continuación se muestra la tabla con las horas finales dedicadas a este proyecto:

	Horas Planificadas	Horas Reales	Estado	Observaciones
Proyecto Integrado	198	278	80	
Aplicación	178	258	80	
Análisis	24	32	8	
Requisitos	2	2	0	
Objetivos	2	2	0	
Tecnología	20	28	8	
Diseño	20	34	14	
Arquitectura - Diseño de clases	4	12	8	
Interfaz - Mockups	2	2	0	
Persistencia - Diseño conceptual	4	4	0	
Criptografía	10	16	6	
Desarrollo	128	186	58	
Estrategia de desarrollo	2	2	0	
Interfaz	20	82	62	
Backend	90	94	4	
BD (SQL)	16	8	-8	
Pruebas	6	6	0	
Usabilidad	6	6	0	
Memoria	12	16	4	
Redacción	10	14	4	
Revisión	2	2	0	
Presentación oral	8	4	-4	
Presentación oral	2	2	0	
Materiales (Power Point, etc.)	6	2	-4	

Para poner en perspectiva este resultado, cada integrante del equipo desarrollador ha dedicado una media de 17,4 horas a la semana al proyecto durante las 8 semanas en que se ha realizado.

Leyes y normativa

Last Words es una aplicación cuya razón de ser es la gestión de datos personales de los usuarios que la utilizan. Por este motivo, es muy importante asegurarse de que la aplicación cumple las distintas leyes que regulan el tratamiento de los datos personales y los derechos que los usuarios pueden ejercer. No es el objetivo de este documento hacer un análisis exhaustivo de las leyes que afectan a la aplicación. Sin embargo, sí se revisarán algunos aspectos fundamentales que la aplicación debe cumplir para adecuarse a las normativas vigentes.

LSSI

La Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico establece los derechos que los usuarios tienen cuando utilizan cualquier servicio digital. También establece las obligaciones de las empresas para cumplir con estos derechos.

Así, por ejemplo, los prestadores de los servicios digitales deben mostrar de forma clara y accesible toda aquella información básica que garantice su identidad a los usuarios. También debe mostrarse gratuitamente la información previa a la contratación de los servicios de manera que sea clara, accesible, comprensible e inequívoca. En el caso de los dispositivos móviles, esta obligación se dará por cumplida si el prestador del servicio incluye un enlace a una página web con esta información.

Los prestadores de servicios también están obligados a recabar explícitamente el consentimiento de los usuarios, siempre después de haber informado de manera clara y completa la utilización, finalidad y tratamiento de sus datos. Esta acción puede llevarse a cabo con los medios de que disponen los navegadores y/o las aplicaciones.

LOPD y RGPD

Para cumplir con la Ley Orgánica de Protección de Datos y con el Reglamento General de Protección de Datos debemos cumplir una serie de requisitos. Algunos de estos requisitos son los siguientes:

- Mantener un registro de actividades del tratamiento de datos personales

- ➢ Realizar un análisis de riesgos y adoptar medidas de seguridad
- ➢ Proteger los datos desde el diseño y por defecto
- ➢ Establecer un código de conducta
- ➢ Notificar las brechas de seguridad que pongan en riesgo la privacidad de los usuarios
- ➢ Designar un delegado de protección de datos

Análisis

Especificación de requisitos

Last Words es una aplicación enfocada al tratamiento, almacenamiento y envío de información privada de los usuarios. Por este motivo, debe poder cumplir con una serie de requisitos funcionales y no funcionales que se describen a continuación.

Funcionales

Las principales funcionalidades de la aplicación Last Words son el poder crear un mensaje que irá destinado a los contactos deseados, asegurándose de que llegarán a ellos de forma segura transcurrido el tiempo deseado.

Esta aplicación debe permitir el registro de diversas maneras según el usuario desee. Principalmente, debe ser posible utilizar un email, el nombre y una contraseña. La información que utilice el usuario para registrarse debe pasar los parámetros exigidos. Otra posibilidad será el registro e inicio de sesión mediante una cuenta de Google o una cuenta de Facebook. Se ha propuesto hacer el login de esta manera, para poder facilitar al usuario todas las comodidades posibles.

Una vez dentro de la aplicación, debe ser posible para el usuario crear mensajes que contengan texto, imagen, vídeo o un fichero de otro tipo. También debe ser posible editar y eliminar estos mensajes, así como consultarlos.

También es un requisito fundamental el permitir al usuario editar sus datos personales, así como también darle la posibilidad de cambiar la contraseña.

No funcionales

Debido a que la aplicación trabaja principalmente con los datos sensibles de los usuarios, la seguridad debe ser una máxima en el desarrollo de la misma. Para garantizar la seguridad, todos los datos personales del usuario que puedan ser susceptibles de ocasionar mermas en su intimidad deben ser tratados con el máximo respeto y rigor, por lo que es fundamental asegurar la información alojada en las bases de datos.

Para ofrecer una experiencia óptima al usuario, la aplicación debe ser rápida y clara en su propósito, no siendo aceptable que el usuario tenga que pasar por pantallas innecesarias para conseguir su objetivo. Así mismo, la ayuda debe ser accesible y fácil de entender, para proporcionar al usuario un apoyo en caso de duda.

Relacionado con el requisito anterior, la aplicación debe ser fácil de aprender, para llegar al máximo de usuarios posible. Para ello, una interfaz agradable y sencilla también debe ser un requisito indispensable.

Arquitectura

A continuación, se describe la arquitectura que debe tener la aplicación para ser completamente funcional y que pueda ser publicada en las tiendas digitales.

Descripción general

La arquitectura de este proyecto consta de los siguientes elementos fundamentales:

- **Aplicación móvil:** Realizada en Flutter, la aplicación tiene como objetivo proveer al usuario de una interfaz amigable y sencilla que le permita crear, guardar, modificar y eliminar mensajes.
- **API**: La API consta de dos partes, una para el almacenamiento de datos y otra para el almacenamiento de ficheros. Spring Boot es la tecnología elegida para comunicar la aplicación con la persistencia de los datos, mientras que Node JS es la elegida para la comunicación de la aplicación con la persistencia de los ficheros.
- **Bases de datos:** Usando PostgreSQL como SGBD, se usará una base de datos relacional para almacenar la información de los usuarios y sus mensajes y otra base de datos para almacenar los ficheros relacionados con los mensajes de los usuarios.
- **API de Google:** Utilizada para permitir el registro e inicio de sesión a los usuarios utilizando sus cuentas de Google.
- **API de Facebook:** Utilizada para permitir el registro e inicio de sesión a los usuarios utilizando sus cuentas de Facebook.
- **API de Twilio:** Esta API será la utilizada para el envío de mensajes informativos a los usuarios y destinatarios de los mensajes.
- **Google Pay:** Servicio de pagos de Google para integrar la compra de mensajes en la aplicación.

Diagramas

A continuación se definirán los diagramas de la aplicación de Last words con una breve descripción de cada una de ellas:

Diagrama de casos de uso

El primer diagrama en referencia a Last Words es el diagrama de casos de uso, en el cual se definen las opciones que el usuario tendrá disponibles para poder interactuar correctamente con el sistema. Dentro de estas interacciones se han definido desde registrarse o iniciar sesión, hasta poder crear un mensaje o editarlo.

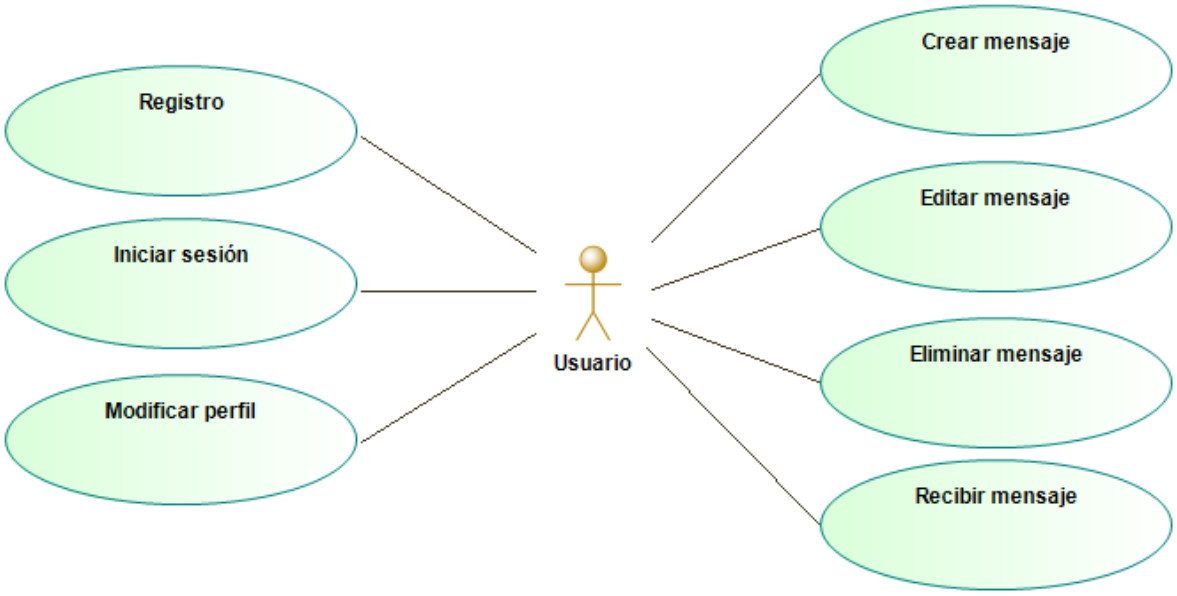


Diagrama de clases

A continuación se muestra el diagrama de clases que define cómo se almacenan dentro de la aplicación los datos necesarios para su funcionamiento, independientemente de la tecnología utilizada para su creación:

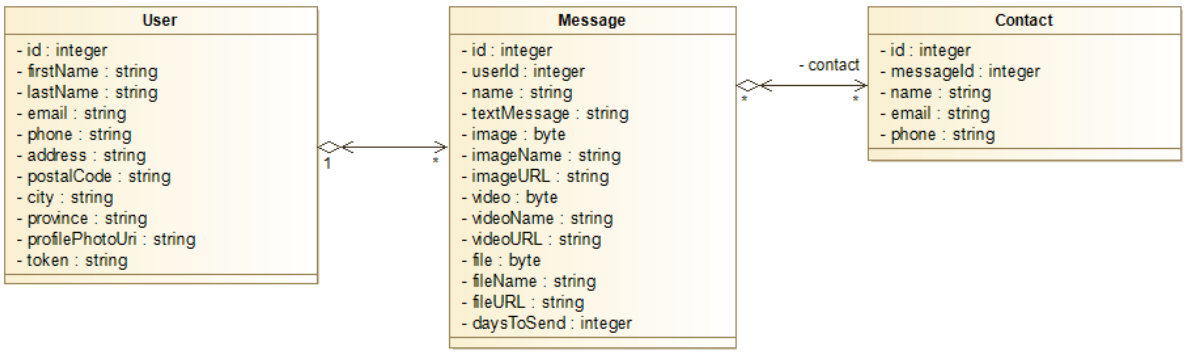
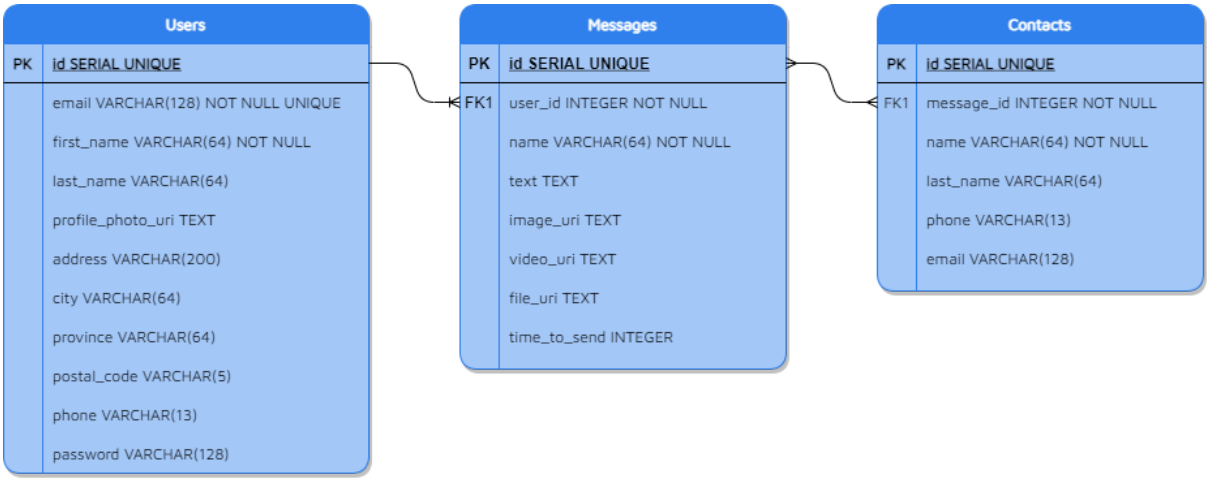
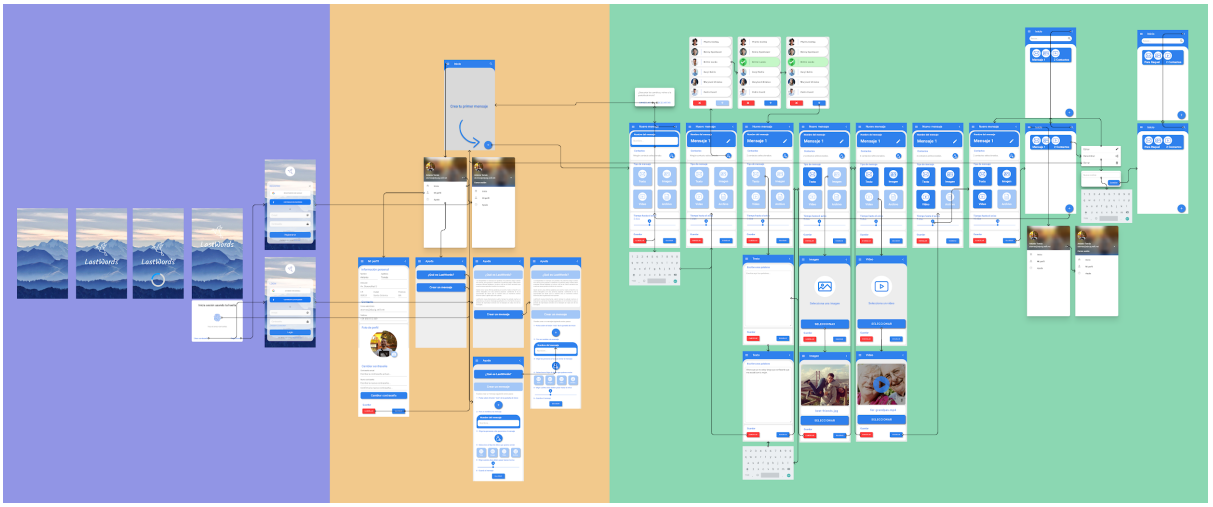


Diagrama de la base de datos

En el siguiente diagrama se puede observar la estructura de los datos dentro de la base de datos PostgreSQL.



Wireframe



Haz clic en la imagen para verla a tamaño completo

Seguridad

En una aplicación como Last Words, que se basa en el manejo de información sensible de los usuarios, la seguridad de los datos es fundamental. Por este motivo, es necesario tomar las medidas oportunas para garantizar que los datos de los usuarios permanezcan a salvo.

En primer lugar, el primer paso es garantizar que la contraseña de los usuarios se guarda encriptada en la base de datos. Para conseguirlo, Spring Boot, junto con OAuth, se encarga de cifrar la contraseña en el momento de recibirla y justo antes de almacenarla en la base de datos.

En segundo lugar, antes de publicar la aplicación en las tiendas digitales es imprescindible garantizar que toda la información que los usuarios utilicen para crear sus mensajes se almacene totalmente cifrada en la base de datos. Esto plantea un enorme reto dada la naturaleza de la aplicación, ya que como se ha comentado anteriormente, el objetivo de la aplicación es guardar mensajes que serán enviados a los destinatarios cuando los usuarios que los han creado fallezcan.

Tanto si se elige un método de criptografía asimétrica como si se elige uno de criptografía simétrica, existe el enorme reto de hacer llegar la clave de desencriptación al destinatario manteniendo el anonimato del usuario creador hasta el momento del fallecimiento. Es posible que pasen varios años hasta que el destinatario reciba el mensaje, por lo que no es baladí la cuestión que concierne al almacenamiento de la clave de desencriptación hasta llegado ese momento. Durante el desarrollo se han barajado distintas opciones, si bien la decisión queda fuera del alcance de este proyecto. Las opciones para el almacenamiento de la clave de desencriptación son las siguientes:

- **Avisar a los destinatarios y obligar a la descarga de la aplicación para el intercambio de claves:** Una de las opciones barajadas es la de obligar a la descarga de la aplicación a los destinatarios en el momento en que se crea el mensaje.

  Un sistema de información por SMS o email avisaría al destinatario de que alguien, sin mencionar el nombre del autor, le ha elegido para recibir información importante en el momento en que esa persona fallezca.

  En caso de que el destinatario aceptara la recepción del mensaje, se le instaría a descargar la aplicación y registrarse. En el momento en que este proceso se completara, las aplicaciones del autor y del destinatario se intercambiarían las claves para que el mensaje pueda ser descifrado en un momento posterior.

  Esta opción presenta dos grandes inconvenientes: el primero es que obliga al destinatario a descargar la aplicación y registrarse, lo que puede generar rechazo. El segundo es que la tecnología es muy cambiante y dinámica, y no es posible garantizar que el destinatario vaya a conservar el mismo teléfono durante un largo periodo de tiempo ni que vaya a mantener instalada una aplicación que no va a usar hasta que reciba el mensaje de la persona fallecida.

- **Cifrar los datos con una contraseña generada aleatoriamente:** Otra de las soluciones que se han tenido en mente ha sido la de cifrar los datos del

autor con una contraseña generada aleatoriamente. En el momento de crear el mensaje, los datos se cifran con una contraseña segura generada de manera aleatoria. En ese momento, existen dos posibilidades:

- **Almacenar la contraseña en una base de datos**
- **Enviar la contraseña al destinatario**

En el caso de **almacenar la contraseña**, cuando el autor creara el mensaje se cifrarían los datos con una contraseña generada aleatoriamente y ésta se almacenaría en un servidor, donde permanecería hasta el fallecimiento del autor. En ese momento, la contraseña se enviaría al destinatario y se le facilitaría un enlace que podría utilizar para consultar el mensaje. El principal inconveniente de esta solución es el riesgo potencial a la seguridad de los datos almacenados, ya que estarían expuestos en caso de sufrir un ataque informático en la base de datos.

En el caso de **enviar la contraseña al destinatario**, cuando el autor creara el mensaje, éste se cifraría con una contraseña aleatoria y, para evitar pasar por el servidor, la aplicación del autor la enviaría al destinatario, que podría usarla para descifrar el mensaje cuando el autor falleciera. Los inconvenientes de esta solución son evidentes, ya que se perdería el anonimato del autor y, además, se delega en el destinatario la conservación de la contraseña durante un tiempo indeterminado.

➢ **Entregar a una notaría la clave de descifrado:** Por último, existe la posibilidad de cifrar el mensaje en el momento de la creación y entregar la clave de descifrado a una notaría, que la custodiaría hasta el momento en que el mensaje deba ser descifrado. La ventaja de esta solución radica en que la responsabilidad del almacenamiento de la clave se deja en manos de una institución legal controlada. El principal inconveniente es el coste que esta solución puede conllevar, ya que los servicios notariales tendrían un coste elevado que podría hacer que la aplicación fuera inviable económicamente.

Persistencia

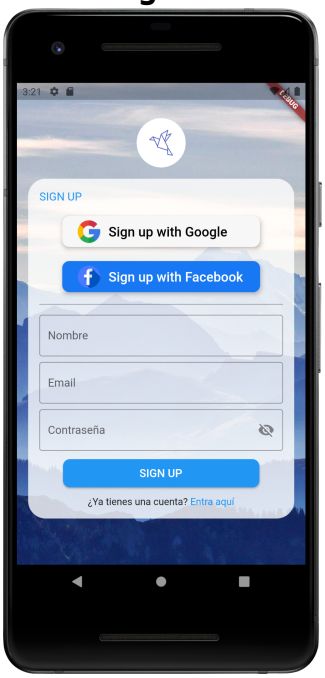
Para la persistencia de los datos de los usuarios, sus mensajes y sus contactos se ha optado por utilizar una base de datos relacional SQL. Dada la baja complejidad de la estructura de los datos, creemos que utilizar una base de datos relacional es lo más adecuado para mantener la integridad y la consistencia de los datos. Además, con las acciones adecuadas, la seguridad de los mismos estaría garantizada y podrían mantenerse durante mucho tiempo debido a la gran estabilidad que otorgan este tipo de bases de datos.

Tal y como se puede observar en la sección de diagramas, la base de datos de Last Words consta de tres tablas relacionadas entre sí. En la tabla *users* se almacenan los datos relativos a los usuarios de la aplicación, como el nombre, la dirección o la foto de perfil. En la tabla *messages* se guarda la información de los mensajes que los usuarios crean con la aplicación. Entre otra información, en esta tabla se encuentra el nombre de los mensajes, así como la dirección de los archivos de imagen o vídeo que se pueden adjuntar. Por último, en la tabla *contacts* se almacena la información de los destinatarios a los que van dirigidos los mensajes. Esta información se obtiene directamente de la agenda de contactos de los usuarios, por lo que la información que se almacena se limita al nombre, apellidos, teléfono y email de estos contactos.

Interfaz

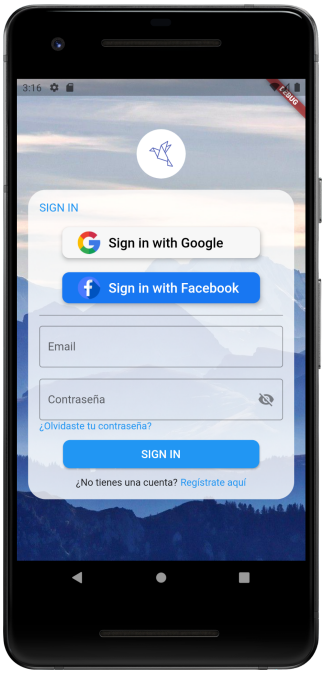
En una aplicación para dispositivos móviles, la interfaz de usuario es tan importante como la parte que no se ve. En el caso de Last Words, la interfaz, creada con Flutter, pretende ser sencilla y fácil de usar. En la sección de diagramas se puede ver el *wireframe* realizado con Figma en las fases tempranas del desarrollo. La interfaz consta de las siguientes pantallas:

Registro



Permite al usuario crear una nueva cuenta.

Inicio de sesión

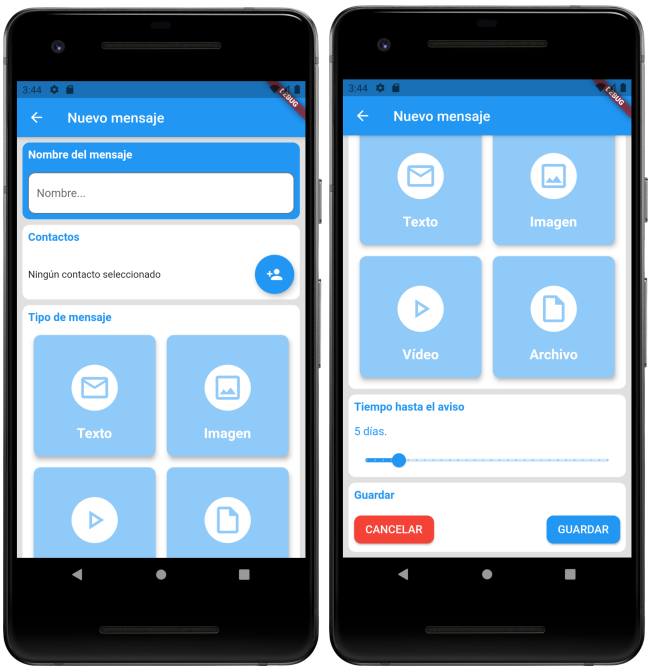


Permite al usuario iniciar sesión mediante email, Google o Facebook.

Inicio

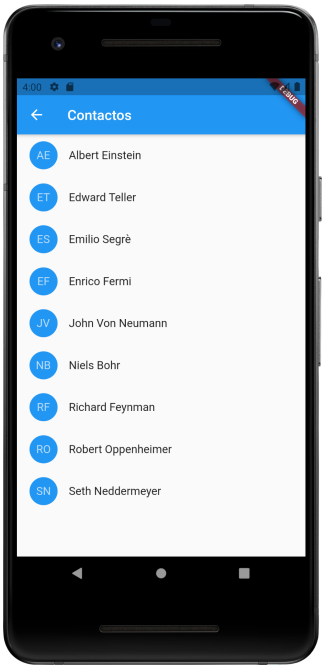
Permite al usuario ver los mensajes creados y acceder a la pantalla de creación de nuevo mensaje.

Nuevo mensaje



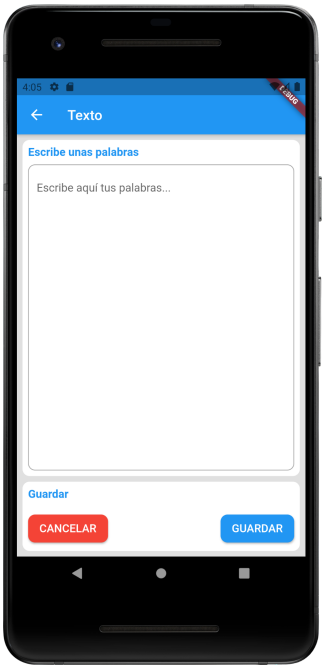
Permite al usuario crear un nuevo mensaje.

Selección de contactos



Permite al usuario seleccionar los contactos a los que va dirigido el nuevo mensaje.

Mensaje de texto



Permite al usuario escribir unas palabras que serán guardadas dentro del mensaje.

Adjunción de imagen

Adjunción de vídeo

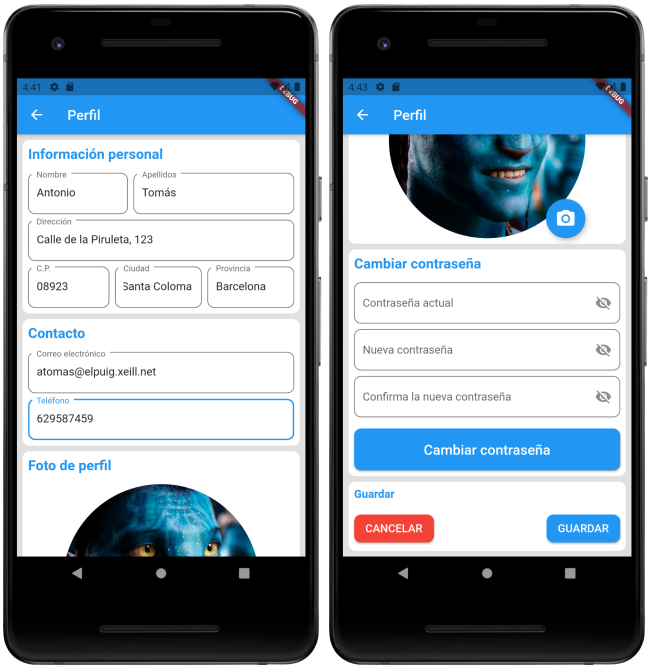
Permite al usuario seleccionar una imagen que será guardada dentro del mensaje.

Permite al usuario seleccionar un vídeo que será guardado dentro del mensaje.

Adjunción de archivo

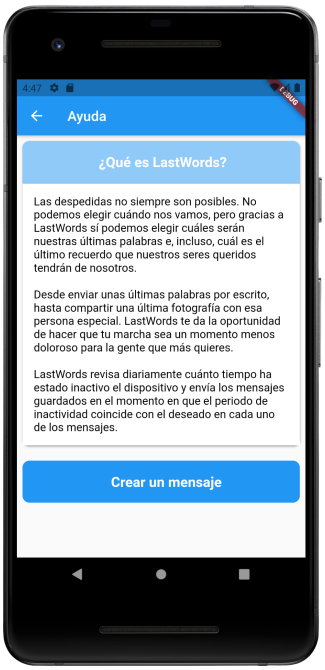
Permite al usuario seleccionar un archivo que será guardado dentro del mensaje

Perfil



Permite al usuario añadir y modificar datos personales, así como cambiar la contraseña.

Ayuda



Permite al usuario conocer el objetivo de la aplicación y cómo se utiliza.

Tecnología

El desarrollo de una aplicación de estas características requiere del uso de varias tecnologías que deben facilitar la carga de trabajo al equipo desarrollador. En el caso que nos ocupa, la siguiente tabla muestra las tecnologías elegidas, por qué se han elegido con respecto a su posible competencia y su función dentro del desarrollo de este proyecto.

Tecnología	Finalidad
Flutter	Framework creado por Google para la creación de aplicaciones multiplataforma. Para el desarrollo de la aplicación existía la posibilidad de utilizar el SDK de Android, pero las posibilidades de llevar el código a otras plataformas nos hizo decantarnos por este framework de desarrollo de aplicaciones. En el proyecto, Flutter es usado para la creación de la interfaz de la aplicación.

 Dart	Lenguaje de programación creado por Google como alternativa moderna a JavaScript. Debido a que Flutter hace uso de este lenguaje no ha sido una opción trabajar con Java como alternativa. Es utilizado para la creación de funciones y la declaración de variables en Flutter.
 Spring	Framework de Java de código abierto con múltiples funcionalidades. Se ha elegido Spring sobre otras alternativas como Struts debido a que se ha obtenido una base en Spring al final del curso y no había tiempo suficiente como para aprender de cero más tecnologías. En el proyecto se utiliza el componente Boot para crear la API que conecta la aplicación con la base de datos.
 Heroku	Plataforma como servicio basada en contenedores que permite desplegar y almacenar aplicaciones de manera muy rápida conectando con GitHub. No se han valorado alternativas debido a que ofrece soluciones gratuitas y de pago asequibles para desarrolladores. En este proyecto se utiliza para desplegar y alojar la API de Spring Boot y la base de datos PostgreSQL.
 PostgreSQL	Sistema Gestor de Bases de Datos relacionales SQL de código abierto. No se han valorado alternativas debido a que ya se tenían conocimientos sólidos de este SGBD y, además, es una alternativa muy válida a otras soluciones de pago como SQL Server. En el proyecto se utiliza para almacenar la información de los usuarios, de los mensajes y de los contactos. También se utiliza para almacenar ficheros.
 Postman	Plataforma para desarrollo de APIs que permite crear y probar APIs de manera eficiente y rápida. No se han valorado alternativas como cURL debido a que Postman es muy fácil de utilizar y la versión gratuita es suficiente para el propósito de este proyecto. En el proyecto se utiliza para probar la API de Spring Boot.

 IntelliJ Idea	Entorno de desarrollo integrado de la compañía JetBrains destinado a la programación con Java. Se valoró utilizar Eclipse como alternativa, pero finalmente se decidió usar IntelliJ por su interfaz más moderna y su rendimiento superior, además del soporte que tiene por parte de la compañía desarrolladora. En este proyecto se utiliza para crear el código fuente de la API Spring Boot.
 Android Studio	Entorno de desarrollo integrado de la compañía JetBrains utilizado para desarrollar aplicaciones Android. Es el entorno de desarrollo oficial para la creación de aplicaciones para este sistema operativo móvil. No se valoraron alternativas como Visual Studio Code debido a la gran integración que este IDE tiene con Android y con IntelliJ. En este proyecto se utiliza para crear el código fuente de la aplicación desarrollada con Flutter.
 GitHub	Plataforma de desarrollo colaborativo de Microsoft para alojar proyectos utilizando el sistema de control de versiones Git. No se han valorado otras opciones como GitLab porque GitHub se integra a la perfección con Heroku y todos los repositorios del curso se han ido alojando aquí. En el proyecto se utiliza para alojar y compartir los códigos fuente de la API Spring Boot y la aplicación realizada con Flutter.
 Google	Compañía de servicios digitales que ofrece todo tipo de aplicaciones web. En el proyecto se utiliza para alojar la aplicación en Google Play, para facilitar el inicio de sesión y el registro en la aplicación con cuentas de Google y para realizar copias de seguridad del código fuente.
 Facebook	Red social con millones de usuarios en todo el mundo. En el proyecto se utiliza para facilitar el inicio de sesión y el registro en la aplicación con cuentas de Facebook.

 Twilio	Compañía de comunicaciones que ofrece servicios de mensajería basada en la nube. No se han valorado alternativas por desconocimiento. En el proyecto se usa para el envío automatizado de SMS y emails.
 Discord	Programa de mensajería con una amplia comunidad. No se han valorado otras alternativas porque es un programa ampliamente utilizado y fácil de usar que permite comunicaciones por texto y voz, además de compartir la pantalla con el resto de integrantes de la llamada en directo. En la aplicación se utiliza para las reuniones diarias del equipo desarrollador.
 WhatsApp	Programa de mensajería instantánea muy utilizado para comunicaciones entre personas. No se ha valorado otra alternativa como Signal o Telegram por la costumbre en el uso diario de esta aplicación. En el proyecto se ha utilizado para que el equipo desarrollador se comunicara de manera rápida y directa, además de para compartir capturas de pantalla y ficheros de pequeño tamaño.

Desarrollo

Estrategia de desarrollo

Para poder llevar a cabo el proyecto se ha tenido que acordar cuál será la mejor manera de llevar a cabo el proyecto, ahorrando tiempo y costos, con la cual se ha partido primeramente de la búsqueda de programas gratuitos o de software libre para economizar al máximo dicho proyecto.

Figma

- En primera instancia el prototipo de la app tras buscar varias alternativas se ha definido con el programa figma. Con dicho software al poderse guardar todo en la nube y hacerse los cambios en tiempo real, ha sido utilizado para poder crear tanto el prototipo como el diseño de la aplicación y así poder facilitar las tareas de maquetación.

Flutter

➢ Para la maquetación y creación de la aplicación se utiliza Flutter.
Dicho framework utiliza el lenguaje de programación Dart y se ha optado por este framework por sus tiempos de carga y a su complicación nativa, la cual mejora notablemente su rendimiento. También se ha utilizado dicho framework porque tiene la característica de que es multiplataforma, un mismo código puede ejecutar el proyecto tanto en Android como en los.

Heroku

➢ La aplicación ha sido subida a un servidor local de heroku, debido a que heroku contiene un plan gratuito para poder subir el proyecto en la nube y a demás cabe destacar que es compatible tanto con las PostgreSQL como con Spring Boot, facilitando así las tareas de automatización y poder permitir el conectar cualquier elemento de la app de manera muy sencilla y en la nube. Tanto la api en Spring Boot, como el front-end con Flutter, la Base de datos y el storage el cual almacena el contenido de los mensajes, han sido subidos a esta plataforma.

Spring Boot

➢ Para el backend de la aplicación se ha optado por hacerse con el framework de Java Spring Boot. Dentro de dicho framework se han utilizado diversas librerías para poder llevar a cabo todas las necesidades de la aplicación, para poder crear controladores y comunicarse con las aplicaciones rest, se ha utilizado la dependencia de Web, ha sido vital ejerciendo roles como la creada de los endpoints los cuales comunicaban el front-end de flutter con el backend.

También para ejercer la conexión a la base de datos en este caso postgres se han utilizado sus pertinentes dependencias ejerciendo así la conexión.

Seguidamente y no menos importante debido a un tema de seguridad ha sido necesario utilizar tanto las dependencias de Spring Security, como las de Spring Oauth2 la cual sigue los protocolos de generación de tokens para así poder evitar el que la app funcione con sesiones y controlar la entrada de terceros. Cabe destacar también que con el método de inyección de dependencias se facilita la programación contra interfaz, permitiendo a los distintos componentes depender únicamente de interfaces y produciendo así un código menos acoplado. No solo eso, también permite implementar el patrón singleton de una forma extremadamente sencilla, por defecto, las dependencias que inyectamos son singletons).

PostgreSQL

- En lo que a Bases de datos se refiere, se ha optado por la utilización de una base de datos relacional, en este caso en concreto con PostgreSQL. Como característica de PostgreSQL se ha optado por su utilización gracias a su flexibilidad ya que es multiplataforma, es de código abierto y gratuito. Por estas características y al ser compatible al cien por cien tanto como con heroku como con Spring y Flutter, se ha desarrollado el proyecto en dicho software.

Android Studio

- En relación con la aplicación, se ha utilizado el entorno de desarrollo Android Studio de JetBrains, ya que cuenta con herramientas que son necesarias para crear aplicaciones. Esto incluye desde el código, al diseño de la interfaz de usuario de la aplicación. El mismo entorno guía al creador en todo el proceso para poder obtener de esta manera una aplicación funcional.

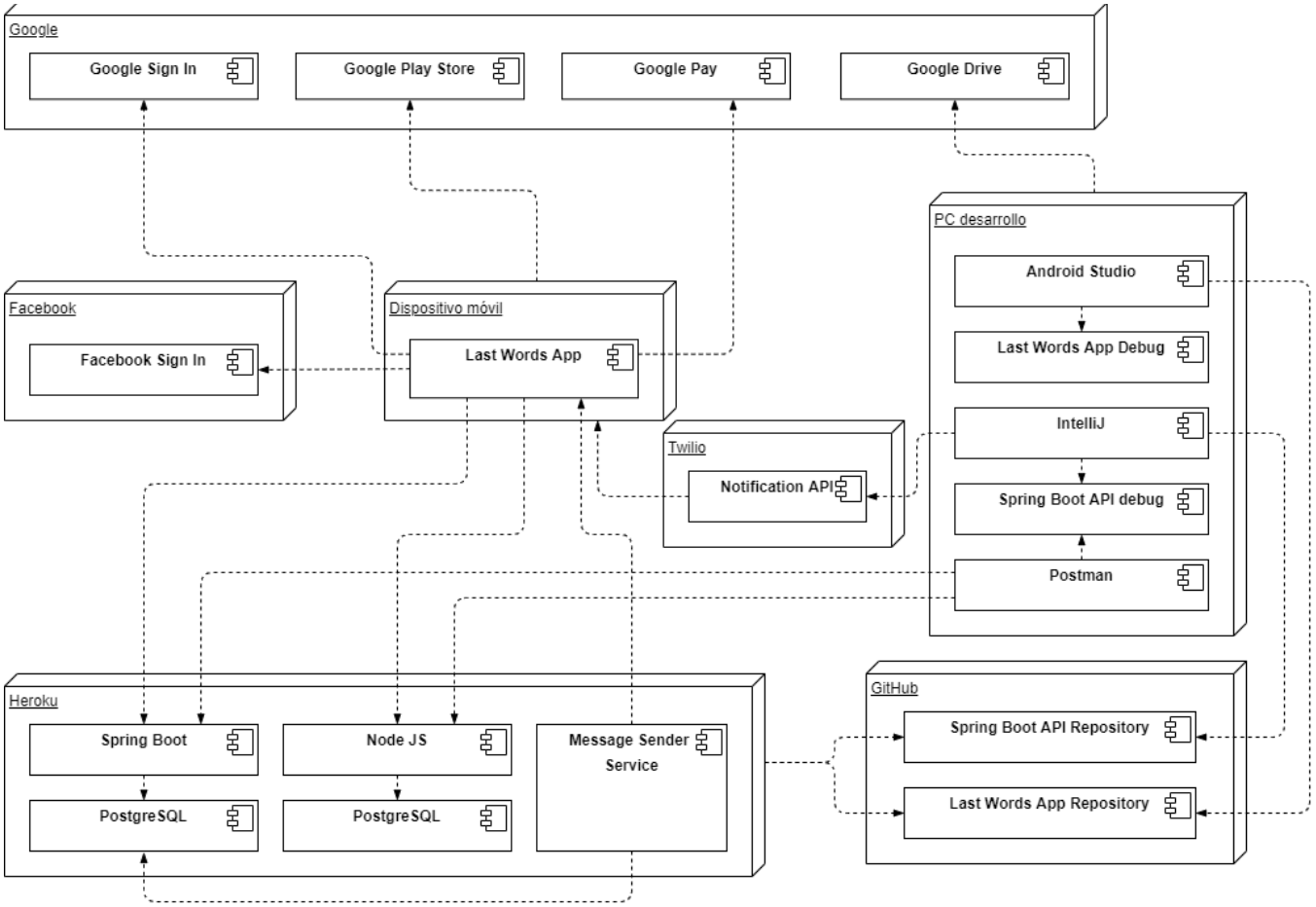
IntelliJ IDEA

- Para desarrollar esta aplicación, se ha optado por utilizar el entorno de desarrollo IntellIJ de JetBrains, debido a su facilidad de utilización, sus funcionalidades y de que la versión gratuita cumple con creces con las necesidades del usuario, se ha optado por desarrollar íntegramente el backend en dicho entorno de desarrollo.

Github

- En Last Words para poder hacer un control de versiones y backups en caso de error o posible pérdida de datos se ha utilizado el software de control de versiones Git. Gracias a este software se ha podido permitir en todo momento controlar lo que se va haciendo dando la posibilidad de recuperar cualquier cosa del código ya creada en caso de pérdida de algún elemento del mismo. En su defecto Git ha sido utilizado con la plataforma para alojar el código de cualquier aplicación GitHub, debido a su facilidad de uso y su interfaz tan intuitiva esta plataforma ha sido la perfecta para el desarrollo del proyecto.

Diagrama de despliegue



Pruebas

Durante el desarrollo del proyecto se han realizado diversas pruebas. Sin embargo, no se han podido realizar todas las que habrían sido necesarias. A continuación se describen las pruebas realizadas.

Funcionales

Durante el desarrollo de la aplicación y la API se han ido realizando algunas pruebas para garantizar que las funciones implementadas tenían el comportamiento esperado. A continuación, se describen estas pruebas y su resultado:

Front-end

Interfaz

A nivel de interfaz, se han realizado una serie de pruebas para asegurar la consistencia del aspecto visual. Algunas de estas pruebas han sido:

1. Comprobar si el uso de colores es adecuado y poco estridente.
2. Comprobar si el tamaño de los botones es adecuado para personas mayores.
3. Comprobar si el diseño es *responsive* y se adapta a todo tipo de pantallas
4. Comprobar si la interfaz cambia cuando se introducen cambios
5. Comprobar que se puede navegar hacia todas las pantallas de la aplicación

Inicio de sesión

A nivel de inicio de sesión, se ha realizado la siguiente prueba:

1. Comprobar que el usuario recibe un aviso cuando trata de introducir unas credenciales con formato inválido.
2. Comprobar que el usuario puede acceder a la aplicación cuando introduce unas credenciales válidas.

Registro

A nivel de registro, se ha realizado la siguiente prueba:

1. Comprobar que el usuario recibe un aviso cuando trata de introducir unos datos con formato inválido.
2. Comprobar que el usuario puede acceder a la aplicación cuando se registra correctamente.

Creación de mensajes

Con respecto a la creación de mensajes, algunas de las pruebas funcionales han sido las siguientes:

1. Comprobar que los botones de la pantalla de creación de un nuevo mensaje cambian de color cuando se ha escrito un texto y/o se ha elegido una imagen, un vídeo o un archivo.
2. Comprobar que el usuario obtiene un feedback háptico cuando desliza la barra que permite seleccionar los días que deben transcurrir para enviar el mensaje.

Perfil

Con respecto al perfil de usuario, se han realizado las siguientes pruebas:

1. Se ha comprobado que el formulario de entrada de datos de usuario funciona correctamente.

Ayuda

Con respecto a la ayuda que debe brindar la aplicación al usuario, se han realizado las siguientes pruebas:

1. Se ha comprobado que la letra es lo suficientemente grande como para ser leída sin problemas por todo tipo de usuarios.
2. Se ha comprobado que el pequeño tutorial incluido es claro e inequívoco.

Back-end

Login

Con respecto al inicio de sesión, se han realizado las siguientes pruebas para asegurar el correcto funcionamiento de esta característica:

1. Intentar hacer login con un usuario no registrado en la base de datos.
2. Probar loguearse con un usuario registrado pero con la contraseña incorrecta.
3. Intentar acceder con un usuario y una contraseña registrados correctamente.
4. Iniciar sesión con un usuario registrado para comprobar que se devuelve un token correctamente.
5. Probar el inicio de sesión con los parámetros vacíos.
6. Inicio sesión con un correo incorrecto pero con una contraseña correcta.
7. Login correcto con Google y Facebook.

Registro

En referencia al registro, se han llevado a cabo una serie de comprobaciones en en relación a este apartado:

1. Registro de un usuario con los campos vacíos.
2. Añadir un usuario ya existente.
3. Comprobar que al registrarse un usuario devuelve un token.
4. Incorporar una contraseña a un usuario existente creado mediante Google o Facebook.
5. Incluir un usuario mediante la Api de Google y Facebook.
6. Introducir un usuario con los datos correctos y completos.

Añadir mensaje

Durante el proceso de desarrollo de la aplicación en lo que añadir mensajes representa se han realizado las siguientes comprobaciones:

1. Añadir un mensaje con todos los campos necesarios.

2. Incluir uno sin los campos necesarios.
3. Insertar un mensaje sin contactos asignados.
4. Asegurarse de que solo permita hacerlo utilizando un token válido.

Eliminar mensaje

Para llevar a cabo esta funcionalidad en la aplicación se han llevado a cabo una serie de validaciones:

1. Eliminar un mensaje inexistente.
2. Borrar un mensaje existente.
3. Garantizar que solo deje hacerlo si se accede con un token correcto.

Editar un mensaje

En cuanto a la edición de de mensajes, ha sido necesario comprobar una serie de pruebas para verificar su comportamiento correcto:

1. Comprobar el editar un mensaje con un id inexistente.
2. Intentar cambiar datos no pre definidos previamente como el id.
3. Cambiar los campos autorizados de uno en uno.
4. Editar todos los campos permitidos a la vez.
5. Cerciorarse de que solo deje hacerlo si se accede con un token correcto.

Editar un usuario

Desde Last Words para comprobar dicha funcionalidad se han llevado a cabo las siguientes pruebas:

1. Intentar editar un usuario inexistente.
2. Probar el cambiar los valores no permitidos como el id de un usuario existente.
3. Editar los valores de un usuario existente.
4. Comprobar que no permita editar usuario sin acceder con un token válido.

Endpoints

En lo que respecta a los endpoints han habido una serie de comprobaciones con el fin de asegurarse la accesibilidad de la app, se detalla a continuación:

1. Comprobar que solo son públicos los accesos de registro y login.
2. Ocultar cualquier otra url que no sea la de registro o login si no se ha verificado el acceso con el token.
3. Intentar acceder a una url sin autenticación.
4. Acceder a una url después de autentificarse.

5. Verificar las redirecciones en caso de no estar autenticado.

Usabilidad

Para comprobar el nivel de usabilidad de la aplicación se solicitó a un usuario que probara la aplicación durante un corto periodo de tiempo. Los resultados de esta prueba se pueden ver en el apartado de anexos, al final de este documento.

Lanzamiento

El principal cometido de Last Words es el de llegar al máximo posible de usuarios por lo cual se optó por desarrollarla con Flutter seguidamente se relataran las tareas para su distribución y planificación con el fin de que la aplicación sea accesible para cualquier usuario:

Tareas para la distribución

Tras la comprobación de todas las funcionalidades y haber superado satisfactoriamente todas las pruebas, se procede a documentar todo el proceso y el funcionamiento de la aplicación. Posteriormente, tras hacerse lo nombrado previamente y antes de seguir con la subida la Play Store, gracias a la ejecución de test de usabilidad en distintos usuarios, se corrigen posibles errores o las recomendaciones encontrados por los testers para así poder mejorar la calidad de los futuros usuarios que la descarguen.

Publicación

Para poder publicar una aplicación tanto en la App Store de Apple como en Google Play Store se deben de cumplir una serie de requisitos.

Google Play Store

En el caso de Play Store, una vez terminada la aplicación, es necesario crear un certificado si es la primera vez que se hace por parte del desarrollador, el cual contendrá las claves de firma de la aplicación o aplicaciones creadas. Al acabar con este proceso y teniendo ya el certificado, se debe generar un APK con la clave creada en el certificado hecho previamente, y con esto, el usuario se cerciora de tener los derechos de dicha aplicación.

Tras terminar el proceso descrito anteriormente y subir la aplicación a la plataforma Play Store, es necesario hacerse una cuenta en la consola de desarrollador Google Play Store. Dicha cuenta, si es la primera vez que se hace, se debe abonar un pago único de por vida de 25€. Al finalizar el abono de la cantidad

descrita previamente, se tiene que rellenar el perfil de desarrollador con los datos personales del propietario de mencionada cuenta.

Después de finalizar este transcurso, se pide seleccionar el idioma de origen de la aplicación y rellenar los datos de información de la app, y, obligatoriamente, añadir un mínimo de 2 y un máximo de 8 capturas o imágenes en referencia a la aplicación. Una vez hecho esto, se debe de añadir el logo y importar el APK a la consola de Play Store.

A continuación, hay que especificar si la aplicación será de pago o gratuita, y definir clasificación de contenido para poder calcular la edad mínima permitida de los usuarios. En caso de que la aplicación sea de pago, el rango de precios permitidos en España es de 0,50€ a 350€.

Para finalizar, es necesario especificar el contenido y su audiencia objetiva. Esto se hará mediante un formulario donde se calculará la edad objetiva de la aplicación, y, tras finalizar este último paso, se le enviará la aplicación a Google para una revisión y en un plazo de 24h, se definirá si es apta o no apta para la plataforma.

App Store

Para poder publicar una aplicación en la App store es necesario disponer de una cuenta en App que se obtiene de forma gratuita registrándote en la plataforma con tu correo electrónico. Después de la obtención de dicha cuenta se deberá de dar de alta en el programa de desarrolladores de Apple el cual, a diferencia de Google Play Store, tendrá un coste bastante más elevado, en concreto es de 99€ anuales. Tras la verificación por parte de App conforme esté todo bien, tanto el pago como los datos, pasadas 48 horas después de la solicitud, se activará la cuenta y podrá ser funcional para poder subir las aplicaciones en la plataforma.

Una vez la cuenta esté activa, se debe de configurar el proyecto con Xcode que es el entorno de desarrollo de Apple. En dicho entorno se debe de ingresar con la cuenta de desarrollador creada previamente, para que automáticamente se generen los certificados de apple firmados necesarios para su compilación para que se pueda subir a la tienda. De igual manera que con Google Play Store, se necesita añadir el icono de la aplicación y las capturas que contendrá. También permite la opción opcional de añadir un promocional. Tras finalizar este proceso, sólo quedará añadir la aplicación mediante Itunes Connect donde se debe acceder con la cuenta creada previamente e ir siguiendo los pasos que van saliendo en la interfaz gráfica proporcionada por Apple, en la cual se debe definir: la descripción, el título, la versión, el copyright de conformidad del creador, entre otras cosas.

Cuando se haya terminado el proceso de subida de la aplicación, es necesario validarla para comprobar que todo está en orden, si después de la validación no se encuentran errores, se debe de validar mediante el Itunes Connect.

Conclusiones

Para la realización de este proyecto apostamos por una idea que consideramos innovadora y con futuro, pero creemos que la envergadura del proyecto es demasiado grande como para tener un producto terminado completamente y comercialmente viable. Después de un periodo de desarrollo bastante apretado, podemos concluir que el proyecto elegido ha supuesto toda una serie de desafíos. Algunos hemos conseguido superarlos, mientras que la falta de tiempo y de conocimientos ha impedido que salváramos otros muchos.

Con la perspectiva que da el tiempo podríamos pensar que la decisión de comenzar desde cero una aplicación para Android, desechando el trabajo realizado previamente, ha sido un error. Sin embargo, analizando con detenimiento esta decisión y considerando la trayectoria que el uso de Java tiene en este Sistema Operativo y su incierto futuro, creemos que la decisión de aprender y utilizar Dart, Flutter y Spring Boot para realizar una aplicación funcional en un plazo de dos meses no podría haber sido más acertada.

No hemos conseguido los objetivos previstos y podemos afirmar que eran demasiado ambiciosos, pero a cambio hemos aprendido tres tecnologías para el desarrollo de aplicaciones con mucho futuro que de otra manera no habríamos utilizado. Haber aprendido los fundamentos de Dart, Flutter y Spring Boot sin duda nos abre un mundo de posibilidades en el desarrollo de aplicaciones multiplataforma, que al final era el objetivo del Ciclo Formativo.

Como conclusión, creemos que tras el trabajo realizado en este proyecto tenemos una base sólida desde la que poder trabajar para, algún día, publicar esta prometedora propuesta en las tiendas de aplicaciones de Google y Apple.

Objetivos alcanzados

A pesar de que la aplicación no es completamente funcional, hemos conseguido crear una interfaz muy aparente con Flutter y conectarla con una API Spring Boot capaz de almacenar los datos en una base de datos PostgreSQL. Para el futuro queda implementar el resto de funcionalidades.

Trabajo futuro

Puesto que algunas de las funcionalidades se escapaban del alcance de este proyecto, sobre todo aquellas relacionadas con la administración de sistemas, el

trabajo futuro estará enfocado en implementar todas aquellas funcionalidades que han quedado fuera.

Entre otras, las funcionalidades en las que se trabajará en el futuro son:

- **Terminar la integración de la API con la aplicación móvil:** Las limitaciones de tiempo provocadas por el proceso de aprendizaje de tecnologías como Flutter y Spring Boot han provocado que algunas de las funcionalidades, como el envío de mensajes a la base de datos, se hayan quedado fuera. El trabajo más inmediato será terminar de integrar la aplicación móvil realizada en Flutter con la API desarrollada en Spring Boot. Esto nos permitirá ampliar las funcionalidades y el alcance de la aplicación.

- **Encriptar la información relativa a los mensajes que los usuarios crean y guardan en la base de datos:** Es importante que los mensajes sólo puedan ser descifrados por el usuario y los contactos que él ha elegido.

- **Implementar el envío automático de los mensajes del usuario cuando éste fallece:** En el momento de entregar el proyecto, la aplicación sólo se comunica con los servidores cuando el usuario se registra o inicia sesión, por lo que debemos trabajar en implementar la comunicación y comprobación diaria de la aplicación con el servidor, para que cuando llegue el momento se envíen los mensajes a los destinatarios.

- **Implementar un sistema de avisos por SMS y email para avisar a los destinatarios:** Por cuestiones técnicas, es necesario que los destinatarios dispongan de la aplicación para poder ver los mensajes que van dirigidos a ellos. Por este motivo, es fundamental animar a los destinatarios a que descarguen la aplicación mediante un SMS o email que les avise, de manera anónima, de que han sido elegidos para recibir información de algún otro usuario, facilitando un enlace de descarga.

- **Implementar un sistema de pago por mensaje:** En la actualidad, la aplicación no dispone de un sistema de pago que haga rentable el despliegue público de la aplicación. Como trabajo para el futuro está previsto implementar un sistema de pago como Google Pay o PayPal que permita vender la creación de mensajes.

- **Crear una versión para iOS:** Cuando todas las funcionalidades descritas anteriormente estén acabadas, el trabajo estará enfocado en adaptar la aplicación para poder lanzarla en dispositivos Apple. Flutter tiene la enorme ventaja de utilizar un sólo código fuente para todos los dispositivos en los

que es compatible, como Android, iOS y navegadores web, por lo que es el siguiente paso lógico en el ciclo de vida de la aplicación.

Bibliografía

Spring

Spring | Home 05/03/2021
¿Qué es Spring Boot? 05/03/2021
Tutorial 02/04/2021
Spring Security 5 - OAuth2 Login 02/04/2021
Spring Boot Token based Authentication with Spring Security & JWT 05/03/2021
How to Change the Default Port in Spring Boot 05/03/2021
How to Change the Default Port in Spring Boot 07/04/2021
Spring Boot OAuth2 Social Login with Google, Facebook, and Github - Part 1 07/04/2021
Connect a Spring Boot app to Cloud SQL 10/04/2021
Cómo mostrar datos en Flutter usando el servicio REST de Spring Boot 12/04/2021
How to return an array of objects in json format in spring boot? 12/04/2021
Spring Boot Consuming and Producing JSON 13/04/2021
Spring Security – Customize the 403 Forbidden/Access Denied Page 13/04/2021
https://www.baeldung.com/spring_redirect_after_login 15/04/2021
https://www.baeldung.com/spring_redirect_after_login 17/04/2021
Postman and Spring security - Help - Postman 17/04/2021
https://stackoverflow.com/questions/30502962/testing-spring-security-with-postman 17/04/2021
Querys avanzadas con JPA en Spring Boot :: Documentacion en español 18/04/2021
https://docs.spring.io/autorepo/docs/spring-android/1.0.x/reference/html/rest-template.html 18/04/2021
No entendí muy bien como funciona el @Autowired de Spring 18/04/2021
Registration + Login Example using Spring Boot, Spring Security, Spring Data JPA, Hibernate, MySQL, Thymeleaf 18/04/2021
Spring Security Oauth2: Google Login 19/04/2021
Spring Boot JPA y su configuración 19/04/2021
Integrar el login de Google en tu App con OAuth2 y Spring Security 26/04/2021
6. Spring JDBC Prev Next 26/04/2021
Getting Started With Google Sign-In and Spring Boot 03/05/2021
Enviar email con Spring 03/05/2021
▸ Crear API REST con Spring 03/05/2021
▸ Crear API REST con Spring 10/05/2021
Solucionar CORS en Spring 10/05/2021
46. Heroku 11/05/2021
ene 8, 2019 | Sebastian Castillo Rodríguez Autenticación de APIs basada en tokens con Spring y JWT 15/05/2021
Que es Spring Boot y su relación con los microservicios 17/05/2021
Using Spring ResponseEntity to Manipulate the HTTP Response 17/05/2021
ResponseEntity 18/05/2021
Rest basic authentication via spring security without form-login 22/05/2021

Testing spring security with Postman 22/05/2021
Using JWT with Spring Security OAuth 24/5/2021
Deploy Spring Boot PostgreSQL CRUD REST API Application to Heroku 24/05/2021
Spring Boot + Spring Security + JWT from scratch - Java Brains 24/05/2021
Despliegue automático de aplicación Spring Boot en Heroku desde GitHub parte 1 25/05/2021
Securizando mi Api con Spring Security y Oauth 2.0 25/05/2021
Spring Boot, Spring Security, PostgreSQL: JWT Authentication example 25/05/2021

Google Play Store

Te puede interesar: Cómo crear y publicar una app en Google Play 28/05/2021

App Store

Cómo publicar una app en App Store – DevsDNA 28/05/2021
Cómo colgar una aplicación en la APP Store 28/05/2021
Guía para publicar una aplicación en la App Store de Apple 28/05/2021

Flutter

Flutter - Beautiful native apps in record time 22/03/2021
Install Flutter SDK 22/03/2021
Dart packages 22/03/2021
An introduction to the dart:io library 22/03/2021
Flutter Awesome 22/03/2021
Learn Flutter and Dart to create Android and IOS apps 04/04/2021
Free Google Flutter Tutorial - Learn Flutter - Beginners Course 04/04/2021
7 Free Flutter Courses for Beginners in 2021 | by javinpaul | Javarevisited 04/04/2021
Free Google Flutter Tutorial - Learn Flutter - Beginners Course 04/04/2021
Build Flutter Android & iOS Chat Application with Firebase – Course Catalog 05/04/2021
Adding assets and images 06/04/2021
https://michaeladesola1410.medium.com/input-validation-flutter-dfe433caec5c 06/04/2021
font_awesome_flutter | Flutter Package The Font Awesome Icon pack available as Flutter Icons. Provides 1500 additional icons 06/04/2021
Flutter how to handle image with fixed size inside box? 12/04/2021
Animate a page route transition 12/04/2021
Cookbook 12/04/2021
Navigation and routing 12/04/2021
Navigation 12/04/20201
Navigate with named routes 12/04/2021
Flutter documentation 12/04/2021
Adding interactivity to your Flutter app 12/04/2021
How do I Set Background image in Flutter? 12/04/2021
How do I stretch an image to fit the whole background (100% height x 100% width) in Flutter? 12/04/2021
How To Use Images In Flutter — To The Point 12/04/2021
How to show fullscreen image in flutter 12/04/2021
ElevatedButton class - material library - Dart API 12/04/2021
Circular widget with wrap text inside · Issue #49878 · flutter/flutter 12/04/2021
Wrapping text inside a circle (Flutter) 12/04/2021

Padding class - widgets library - Dart API - Flutter 12/04/2021
ShapeDecoration class - painting library - Dart API 12/04/2021
Understanding constraints 12/04/2021
https://github.com/gerardfp/flutter_basic_http 12/04/2021
flutter ImageAsset Image extending outside BoxDecoration container 12/04/2021
Add border to a Container with borderRadius in Flutter 12/04/2021
how to change the opacity of the snackbar in flutter? 12/04/2021
Dart API docs 12/04/2021
Widget catalog 12/04/2021
Material Components widgets 12/04/2021
TextField class - material library - Dart API 12/04/2021
TextEditingController class - widgets library - Dart API 12/04/2021
Handle changes to a text field 12/04/2021
AppBar class - material library - Dart API 12/04/2021
Añadir un Drawer a la pantalla 12/04/2021
¿Cómo diseñar un menú drawer o menú lateral en Flutter? | FLUTTER 12/04/2021
Navigation Drawer using Flutter. In this article I'm gonna talk about... | by Ashish Rawat | CodeChai 12/04/2021
Scaffold class in Flutter with Examples 13/04/2021
Scaffold class - material library - Dart API 13/04/2021
Introduction to widgets Flutter widgets are built using a modern framework that takes inspiration from React 13/04/2021
New Buttons and Button Themes 13/04/2021
Create and style a text field 13/04/2021
Build a form with validation 13/04/2021
Text fields 13/04/2021
beamer | Flutter Package 13/04/2021
https://pub.dev/packages/showcase 13/04/2021
showcaseview | Flutter Package 13/04/2021
Flutter Change height of an OutlineButton inside an AppBar? 13/04/2021
https://pub.dev/packages/flutter_signin_button/ 13/04/2021
SizedBox class - widgets library - Dart API 13/04/2021
Flutter Inspector is showing in a weird way 19/04/2021
Flutter/Dart How to center sizedbox to top center of screen? 19/04/2021
https://stackoverflow.com/questions/51155208/make-container-widget-fill-parent-vertically 19/04/2021
Flutter — Container Cheat Sheet. A convenience widget that combines... | by Julien Louage | JLouage 19/04/2021
Error : the argument type int cannot be assigned to the parameter type string 19/04/2021
Create a rounded button / button with border-radius in Flutter 19/04/2021
Set the space between Elements in Row Flutter 19/04/2021
Space between Column's children in Flutter 19/04/2021
Flutter Gridview in Column. What's solution..? 20/04/2021
Flutter - GestureDetector onTapDown color changing 20/04/2021
GridView class - widgets library - Dart API 20/04/2021
Place a button after Gridview.count - Flutter 20/04/2021
ToggleButtons in GridView 20/04/2021
Scrollable Container inside a Column 22/04/2021
https://stackoverflow.com/questions/56326005/how-to-use-expanded-in-singlechildscrollview 22/04/2021
SingleChildScrollView class - widgets library - Dart API 22/04/2021

Flutter Problem: Listview Vs Column + SingleChildScrollView | by Jitesh Mohite | FlutterWorld 22/04/2021
seekbar | Flutter Package 22/04/2021
Contents in this project Flutter Create Material Seekbar Slider in Android iOS Example Tutorial 22/04/2021
Slider class - material library - Dart API 22/04/2021
Flutter align button to bottom of Drawer 22/04/2021
Flutter textfield background color on focus 22/04/2021
form | Flutter Package 22/04/2021
reactive_forms | Flutter Package (pub.dev) 22/04/2021
flutter_login | Flutter Package A login widget with login/signup functionalities to help speed up development pub 22/04/2021
dropdown_search | Flutter Package 26/04/2021
https://pub.dev/packages/anim_search_bar 26/04/2021
simple_search_bar | Flutter Package A simple yet functional flutter search bar. It's an AppBar that You can 26/04/2021
outline_search_bar | Flutter Package 26/04/2021
animations | Flutter Package 26/04/2021
Flutter change Text on Button click in Gridview 26/04/2021
List of state management approaches 27/04/2021
How to change the background color of raised button dynamically in onPressed() 27/04/2021
Change Background Color of button dynamically in Flutter / Dart 27/04/2021
Simple app state management 27/04/2021
provider | Flutter Package 27/04/2021
Building Layouts in Flutter 27/04/2021
Writing custom platform-specific code 03/05/2021
ClipRect class - widgets library - Dart API 03/05/2021
The equivalent of wrap_content and match_parent in flutter? 03/05/2021
How to Get Filename of a File In Flutter ? 03/05/2021
Sizing elements to percentage of screen width/height 03/05/2021
image_picker | Flutter Package 03/05/2021
Adding an Image Picker in a Flutter App - Pick images using Camera and Gallery / Photos 03/05/2021
video_player | Flutter Package 03/05/2021
Flutter Tutorial - ImagePicker - Pick Image & Video 03/05/2021
Flutter Image and Video Picker · GitHub 03/05/2021
Open default contacts app from my app in Flutter 04/05/2021
Open Contacts on device via icon tap flutter 04/05/2021
contacts_service | Flutter Package 04/05/2021
How To Access Contacts On Your Flutter App | by Arvinth | Xenon Labs 04/05/2021
How to Use the Provider Pattern in Flutter 04/05/2021
FLUTTER: PROVIDER PARA PRINCIPIANTES -GUIA DE INICIO | by Saul Ram 04/05/2021
File picker plugin for Flutter compatible with both iOS & Android 04/05/2021
is there a way to generate model class for firestore database on Flutter? 06/05/2021
Camera and Photo Gallery Using Image Picker with Flutter | Day 25 - #30DaysOfFlutter 09/05/2021
Flutter: Provider, una alternativa al BLoC 09/05/2021
https://heartbeat.fritz.ai/using-the-camera-gallery-in-flutter-apps-2f9e8e0e5899?gi=3f55d596dabc 09/05/2021
#Flutter Tutorial - Save Selected Image in App Directory in separate Thread (coderzheaven.com) 09/05/2021

Flutter 36: Provider 1 - Value 09/05/2021
Provider state management in flutter with Todo Application 09/05/2021
Flutter/Dart - How to update Image after Image Picker on other Screens 09/05/2021
ImageProvider class - painting library - Dart API 09/05/2021
Read and write files 09/05/2021
Reading and Writing Data and Files with Path Provider using Dart's Flutter Framework — Steemit 09/05/2021
Contacts List & Search Implementation in Flutter - Easy Peasy 09/05/2021
A Flutter plugin to retrieve and manage contacts on Android and iOS 09/05/2021
flutter_native_splash | Dart Package 10/05/2021
Expandable AppBar (onPressed) 10/05/2021
Flutter : Splash Screen. One of the first questions when... | by Diego Velasquez 10/05/2021
scoped_model | Flutter Package 10/05/2021
expansion_tile_card | Flutter Package 10/05/2021
How to get AppBar height in Flutter? 10/05/2021
Icons class - material library - Dart API 11/05/2021
MaterialButton class - material library - Dart API 11/05/2021
permission_handler | Flutter Package 13/05/2021
How to implement vibration with Flutter for both Android and iOS? 13/05/2021
How To Solve Flutter 2.0 upgrade, pub get failed with nonsense dependency behavior 15/05/2021
file_picker | Flutter Package A package that allows you to use a native file explorer to pick 15/05/2021
Dart 2: enums. Tipos enumerados | by Marcos Alejandro 15/05/2021
1 thought on "Como crear un AlertDialog en Flutter" 15/05/2021
AlertDialog class - material library - Dart API 15/05/2021
Close AlertDialog on Navigating back from second screen and refresh page flutter 15/05/2021
FileSystemException: FileSystemException: Cannot copy file, errno = 32 · Issue #28905 · flutter/flutter 15/05/2021
Add a Drawer to a screen 16/05/2021
Reset provider data - Flutter 16/05/2021
How to pass data from alertdialog to page in flutter? 16/05/2021
how to pass Context to show AlertDialog in flutter 16/05/2021
287 How to change the application launcher icon on Flutter? 16/05/2021
flutter_launcher_icons | Dart Package A package which simplifies the task of updating your Flutter app's launcher icon 16/05/2021
Flutter Docs on "JSON and serialization" 16/05/2021
Pass arguments to a named route 18/05/2021
How to Create a Context Menu in Flutter? 18/05/2021
Send data to the internet 20/05/2021
Obtener datos desde internet 20/05/2021
How to make HTTP requests in Flutter 20/05/2021
Shared Preferences: ¿Cómo guardar la configuración de la aplicación Flutter y las preferencias del... 24/05/2021
Flutter Session login and keep logged in 24/05/2021
Flutter Login With Database SQLite | by Ecco Suprastyo 24/05/2021
Flutter: How to do CRUD with PostgreSQL? Part 1 24/05/2021
Flutter Thursday Series. I came across flutter sometime this... | by Shuaib Afegbua 24/05/2021
Spring Boot Flutter Login Registration | Flutter Spring Boot Authentication | REST API | MySQL 24/05/2021
Flutter + API Spring Boot + MySql creación CRUD, explicación de la app desde cero 24/05/2021
Named Routes in Flutter 24/05/2021

Flutter Thursday 13: Building a User Registration and Login Process with provider and external API 24/05/2021
Flutter: How to get data show DrawerWidget as per the user or admin which is stored in sharedPreference? 26/05/2021

Anexos

Test de usabilidad

Autor	Lloc	Data
Antonio Tomás Franco	Institut Puig Castellar	18/02/2021

Objectiu Descripció de l'objectiu, raons i justificació de la prova

El objetivo de esta prueba es detectar errores en la navegación de la aplicación Last Words, comprobar si el modelo mental ideado por el equipo desarrollador corresponde con el del público objetivo y obtener opiniones sobre el aspecto visual de la aplicación.

Tipus de prova Testejar funcionament, capacitat de resolució de l'usuari, etc.	Abast Es concreta la part a provar (tota l'app, una o varies funcionalitats)
Comprobar el flujo natural de uso de la aplicación, lo que comprende el registro y login, modificación del perfil, consulta de la ayuda y creación, modificación y eliminación de mensajes. Comprobar la adecuación del estilo visual.	El ámbito de la prueba estará en la estructura de navegación de la aplicación y en su aspecto visual.

Participants Participants i tipus (expert/novell) que intervenen en la prova (stakeholders, desenvolupadors, extern, vinculat al projecte, etc.)

Participantes expertos en el uso de aplicaciones móviles.

Procediment Tasques a realitzar durant el test

Tasca	Temps minuts	Fet (%)	Descripció

1	1m	100%	Breve introducción al usuario de la aplicación y entrega de guion con pautas generalistas sobre la prueba.
2	30s	100%	Registro e inicio de sesión.
3	1m30s	100%	Descubrir el objetivo de la aplicación y el modo de uso mediante la sección de ayuda.
4	3m	100%	Creación de un mensaje.
5	1m	100%	Modificación y eliminación de un mensaje.
6	3m	100%	Recogida de impresiones y sugerencias.

Treball de camp Atributs que es volen provar, mètriques i/o preguntes relacionades, valors desitjats, reals i anotacions

Atributs	Mètriques/Preguntes	Tipus (subjectiva/obj.)	Valor desitjat	Valor real	Anotacions
Efectividad	Registro y Login correctos	Objetiva	Sí	Sí	Durante el registro y login, el usuario ha introducido incorrectamente los datos, pero se atribuye a la rapidez de la prueba.
	Creación de un mensaje	Objetiva	Sí	Sí*	El usuario ha creado un mensaje sin problemas pero ha realizado pulsaciones de manera intuitiva en sitios donde no había interacciones o eran diferentes a lo esperado (lista de contactos, selección de imagen/video/archivo).
Satisfacción	Aspecto visual agradable/atractivo	Subjetiva	Sí	Sí	El usuario ha encontrado agradable los colores elegidos y las proporciones de la aplicación.

		Ayuda clara y fácil de aprender	Subjetiva	Sí	Sí*	El usuario cree que la ayuda es útil, pero podría estar más accesible.
		Facilidad para crear mensajes	Subjetiva	Sí	Sí	Habría que introducir interacciones en algunos puntos de la aplicación, como el ImageView de seleccionar imagen/vídeo/archivo. El usuario ha pulsado en estos lugares esperando una interacción que no ha ocurrido.
	Eficiencia	Crear un mensaje	Objetiva	<3m	2:52m	El usuario ha podido crear un mensaje en un tiempo cercano al estimado.
		Registro y Login	Objetiva	<30s	40s	El usuario ha tardado un poco más de tiempo del esperado al introducir incorrectamente los datos de login.
	Errores	Errores aparecidos durante el uso	Objetiva	0	0	La aplicación no ha sufrido ningún error inesperado durante su uso.
	Diseño/Temática	Adecuación de la apariencia con la temática	Subjetiva	Sí	Sí	El usuario cree que la apariencia y los colores elegidos son muy adecuados para la temática de aplicación y su objetivo.

Conclusions Conclusions dels resultats de la prova amb els errors detectats, les solucions plantejades i el treball futur

La prueba ha sido satisfactoria y se han detectado algunas mejoras a realizar. Las tareas más inmediatas se centrarán en mejorar y añadir las interacciones que el usuario ha intentado de manera intuitiva pero que no estaban implementadas, como por ejemplo pulsar sobre los iconos de imagen/video/archivo para abrir la galería. También se tratará de mejorar la ubicación de la ayuda o, en su defecto, la manera de encontrarla. La pequeña guía inicial, que explica cómo

	crear un nuevo mensaje y modificar los existentes, se cambiará para asegurarse de que el usuario la lee. Se trabajará en mejorar el aspecto e interacción de los mensajes de la lista de la pantalla principal.											

Enlaces a los repositorios

https://github.com/BilBenAch/Api-rest
https://github.com/ATomas-Puig/lastwords