



TRABAJO FIN DE GRADO  
GRADO EN INGENIERÍA INFORMÁTICA  
MENCIÓN EN INGENIERÍA DEL SOFTWARE



## **Simplex: aplicación móvil para la organización personal**

**Estudiante:** Joaquín Solla Vázquez  
**Dirección:** Fernando Bellas Permuy

A Coruña, agosto de 2023.



*A mi abuelo César*



## **Agradecimientos**

Para comenzar, me gustaría agradecer a mis padres y a mi hermana el apoyo que me han brindado sin cesar, a pesar de no comprender los disgustos y preocupaciones que puede llegar a brindar el mundo de la informática. Gracias por haberme ayudado a seguir adelante.

A mis amigos, por su curiosidad en mis proyectos y su predisposición a probar todo aquello que haya desarrollado.

A Clara, por acompañarme de principio a fin en este viaje de 4 años en el que ha padecido de mis errores y disfrutado de mis éxitos tanto como yo mismo.

Finalmente, a mi abuelo César, que de haber podido ver cómo me gradúo habría sido la persona más orgullosa del mundo.

## **Resumen**

El proyecto consiste en una aplicación móvil dedicada a la organización personal, de forma que su objetivo principal es mejorar la vida diaria de cualquier usuario. La aplicación se centra en la gestión de cuatro tipos de elementos: eventos, tareas, notas y rutinas. Con una ventana dedicada a cada tipo de elemento, el usuario puede realizar acciones como crear, editar, eliminar, ver sus detalles o compartir. A diferencia de las aplicaciones convencionales de organización personal, esta dispone de un calendario en el que se sitúan los 4 tipos de elementos para tener una vista más organizada de cada día.

El usuario deberá crearse una cuenta para usar la aplicación, de esta forma podrá disfrutar de la sincronización de cambios en tiempo real en diferentes dispositivos con la misma cuenta. La prioridad número uno de la aplicación es el usuario, por lo que se ha trabajado exhaustivamente para conseguir una interfaz intuitiva a la vez que sencilla y agradable para la vista. También se han implementado diversos ajustes de accesibilidad como soporte multi-idioma o formatos de fecha y hora.

Con un enfoque [Backend as a Service \(BaaS\)](#), la aplicación trabaja con [Google Firebase](#) y [Firestore Database](#) para gestionar tanto los usuarios como las bases de datos en la nube.

## **Abstract**

The project consists of a mobile application dedicated to personal organization, so that its main objective is to improve the daily life of any user. The application focuses on managing four types of items: events, to-dos, notes, and routines. With a window dedicated to each type of element, the user can perform actions such as create, edit, delete, view its details or share. Unlike the applications used for personal organization, this one has a calendar in which the 4 types of elements are located to have a more organized view of each day.

The user will need to create an account to use the application, so he/she can enjoy the change synchronization in real time on different devices with the same account.

The first priority of the application is the user, so extensive work has been done to achieve an intuitive interface that is simple and pleasing to the eye at the same time. Various accessibility definitions have also been implemented, such as multilanguage support or date and time formats.

With a [Backend as a Service \(BaaS\)](#) approach, the application works with [Google Firebase](#) and [Firestore Database](#) to manage both users and databases in the cloud.

**Palabras clave:**

- Aplicación móvil
- Android
- Flutter
- Dart
- Google Firebase
- Calendario
- Evento
- Tarea
- Nota
- Rutina

**Keywords:**

- Mobile application
- Android
- Flutter
- Dart
- Google Firebase
- Calendar
- Event
- To-Do
- Note
- Routine

# Índice general

---

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1	Contexto . . . . .	1
1.2	Objetivos . . . . .	2
1.3	Visión global del sistema . . . . .	4
<b>2</b>	<b>Estado del arte</b>	<b>6</b>
2.1	Google Calendar . . . . .	6
2.2	Any.do . . . . .	7
2.3	Evernote . . . . .	7
2.4	Otras aplicaciones . . . . .	8
2.5	Detección de elementos clave . . . . .	8
<b>3</b>	<b>Metodología</b>	<b>10</b>
3.1	Metodología escogida . . . . .	10
3.2	Detalles de la metodología . . . . .	11
3.2.1	Tablero Kanban . . . . .	11
3.2.2	Pruebas en dispositivos . . . . .	12
3.2.3	Seguimiento del tiempo . . . . .	12
3.3	Razonamiento de elección . . . . .	13
<b>4</b>	<b>Análisis de requisitos global</b>	<b>14</b>
4.1	Actores . . . . .	14
4.2	Casos de uso . . . . .	14
<b>5</b>	<b>Planificación</b>	<b>21</b>
5.1	Iteraciones . . . . .	21
5.1.1	Iteración 0 . . . . .	21
5.1.2	Iteración 1 . . . . .	21

5.1.3	Iteración 2 . . . . .	22
5.1.4	Iteración 3 . . . . .	22
5.1.5	Iteración 4 . . . . .	23
5.1.6	Iteración 5 . . . . .	23
5.1.7	Iteración 6 . . . . .	24
5.1.8	Iteración 7 . . . . .	24
5.1.9	Iteración 8 . . . . .	24
5.1.10	Iteración 9 . . . . .	25
5.2	Planificación temporal . . . . .	25
5.3	Cálculo de costes . . . . .	25
<b>6</b>	<b>Fundamentos tecnológicos</b>	<b>28</b>
6.1	Tecnologías usadas para el código fuente . . . . .	28
6.1.1	Android Studio . . . . .	28
6.1.2	Dart . . . . .	28
6.1.3	Flutter . . . . .	28
6.1.4	Gradle . . . . .	29
6.1.5	XML . . . . .	30
6.1.6	JSON . . . . .	30
6.1.7	Application Resource Bundle . . . . .	30
6.2	Servicios remotos . . . . .	30
6.2.1	Google Firebase Authentication . . . . .	30
6.2.2	Google Firestore Database . . . . .	31
6.2.3	Google Play Console . . . . .	31
6.3	Otras tecnologías . . . . .	32
6.3.1	Git . . . . .	32
6.3.2	Taiga . . . . .	32
6.3.3	Adobe Photoshop . . . . .	32
<b>7</b>	<b>Desarrollo iterativo</b>	<b>33</b>
7.1	Estructura del proyecto . . . . .	33
7.2	Modelo de datos . . . . .	36
7.3	Iteración 2 . . . . .	37
7.3.1	Análisis . . . . .	37
7.3.2	Diseño e implementación . . . . .	37
7.4	Iteración 3 . . . . .	41
7.4.1	Análisis . . . . .	41
7.4.2	Diseño e implementación . . . . .	41

---

7.5	Iteración 4 . . . . .	47
7.5.1	Análisis . . . . .	47
7.5.2	Diseño e implementación . . . . .	47
7.6	Iteración 5 . . . . .	52
7.6.1	Análisis . . . . .	52
7.6.2	Diseño e implementación . . . . .	52
7.7	Iteración 6 . . . . .	55
7.7.1	Análisis . . . . .	55
7.7.2	Diseño e implementación . . . . .	56
7.8	Iteración 7 . . . . .	59
7.8.1	Análisis . . . . .	59
7.8.2	Diseño e implementación . . . . .	59
7.9	Iteración 8 . . . . .	62
7.9.1	Análisis . . . . .	62
7.9.2	Diseño e implementación . . . . .	62
7.10	Iteración 9 . . . . .	66
7.11	Otros aspectos del desarrollo . . . . .	67
7.11.1	Presentación introductoria . . . . .	67
7.11.2	Depuración del código . . . . .	67
7.11.3	Colores de la aplicación . . . . .	68
7.11.4	Índices de la base de datos . . . . .	68
<b>8</b>	<b>Acceso a la aplicación</b>	<b>69</b>
8.1	Google Play Store . . . . .	69
8.2	GitHub . . . . .	70
8.3	Política de privacidad . . . . .	70
<b>9</b>	<b>Conclusiones y trabajo futuro</b>	<b>71</b>
9.1	Conclusiones . . . . .	71
9.2	Trabajo futuro . . . . .	71
<b>Lista de acrónimos</b>		<b>74</b>
<b>Glosario</b>		<b>75</b>
<b>Bibliografía</b>		<b>78</b>

# Índice de figuras

---

1.1	Mockup del calendario y la edición de un evento . . . . .	3
1.2	Mockup de la sección de ajustes . . . . .	4
1.3	Arquitectura del sistema . . . . .	5
2.1	Presentación de Google Calendar en Google Play Store . . . . .	6
2.2	Presentación de Any.do en Google Play Store . . . . .	7
2.3	Presentación de Evernote en Google Play Store . . . . .	8
3.1	Tablero Kanban durante el desarrollo del proyecto . . . . .	11
3.2	Gestor de tiempo durante el desarrollo del proyecto . . . . .	13
4.1	Mockup de las ventanas previas a autenticarse . . . . .	15
4.2	Mockup de ejemplo de notificaciones . . . . .	16
4.3	Mockup de gestión de las tareas . . . . .	17
4.4	Mockup de una tarea compartida como mensaje de Whatsapp . . . . .	18
4.5	Mockup de la vista de rutina . . . . .	19
4.6	Mockup de las vistas de ayuda, estadísticas y reportar un problema . . . . .	20
6.1	Vista web de administrador del servicio Google Firebase Authentication . . . . .	31
6.2	Vista web de administrador del servicio Google Firestore Database . . . . .	31
6.3	Vista web de la configuración de la ficha de Google Play Store . . . . .	32
7.1	Modelo de datos de la aplicación . . . . .	36
7.2	Estructura de la interfaz de las páginas . . . . .	38
7.3	Sección de ajustes con los temas claro y oscuro . . . . .	39
7.4	Página de ajuste del tamaño del texto . . . . .	40
7.5	Página de ayuda y alguna de sus secciones . . . . .	41
7.6	Flujo de la ejecución en función del estado de autenticación del usuario . . . . .	43
7.7	Widget Snackbar que indica errores al usuario . . . . .	44

---

7.8	Ventana de Iniciar sesión . . . . .	45
7.9	Mensaje de confirmación tras cambiar la contraseña . . . . .	45
7.10	Envío del email para restablecer la contraseña . . . . .	46
7.11	Email para restablecer la contraseña . . . . .	46
7.12	Calendario de Simplex . . . . .	48
7.13	Vista principal de las tareas . . . . .	54
7.14	Vista principal de las notas . . . . .	56
7.15	Formulario de creación de notas . . . . .	57
7.16	Eventos, tareas y notas en el calendario . . . . .	58
7.17	Lista de rutinas . . . . .	60
7.18	Selector de tipo de evento . . . . .	60
7.19	Selector de días de rutina de la nota . . . . .	61
7.20	Página de reporte de problemas . . . . .	64
7.21	Reporte visto desde el gestor de la base de datos . . . . .	64
7.22	Estadísticas para testers . . . . .	65
7.23	Paquete l10n para la internacionalización . . . . .	65
7.24	La aplicación en los 3 idiomas disponibles (español, gallego e inglés) . . . . .	66
7.25	Diagrama UML de la arquitectura general de la aplicación . . . . .	67
7.26	Presentación introductoria a la aplicación . . . . .	67
8.1	Simplex en Google Play Store . . . . .	69
8.2	Simplex en GitHub . . . . .	70
8.3	Política de privacidad . . . . .	70

# Índice de tablas

---

3.1	Dispositivos utilizados para las pruebas de la aplicación . . . . .	12
5.1	Casos de uso de la iteración 2 . . . . .	22
5.2	Casos de uso de la iteración 3 . . . . .	22
5.3	Casos de uso de la iteración 4 . . . . .	23
5.4	Casos de uso de la iteración 5 . . . . .	23
5.5	Casos de uso de la iteración 6 . . . . .	24
5.6	Casos de uso de la iteración 7 . . . . .	24
5.7	Casos de uso de la iteración 8 . . . . .	25
5.8	Planificación temporal del proyecto y tiempo real invertido . . . . .	26
5.9	Desglose del coste total del proyecto . . . . .	27

## Capítulo 1

# Introducción

---

### 1.1 Contexto

Siempre me he preocupado por ser una persona eficiente, es decir, aprovechar el tiempo lo mejor posible y tener todos mis planes bien organizados con antelación. A raíz de esto, surge la necesidad personal de una aplicación móvil (ya que es el dispositivo que tenemos siempre a mano) para llevar un seguimiento de mi vida diaria y sus diferentes eventos.

Tras probar varias aplicaciones dedicadas a la mejora de la organización personal, entre ellas aquellas que vienen integradas en todos los teléfonos móviles como las de Google y Apple, no encontraba nada que cumpliese todas mis necesidades y que me resultase cómodo de usar.

A todo esto se añade una observación que he hecho durante mi tiempo en el grado. Muchas personas, tanto dentro como fuera de la universidad, organizan sus notas, tareas pendientes y fechas importantes en un sólo lugar, muy habitualmente en Whatsapp [1]. Y es que gran parte de la gente organiza su día a día en un grupo de Whatsapp [1] sin participantes de forma que escribe en un chat vacío lo que considera necesario como recordatorio. No hay que explicar demasiado por qué esto no es eficiente, ya que se trata de un chat en el que los mensajes no se pueden reorganizar ni visualizar de forma ordenada.

Sumando mi necesidad personal y dicha observación, llegué a la decisión de desarrollar mi propia aplicación para móvil dedicada a la organización personal: Simplex. En ella se cumplirían los requisitos que no han podido cumplir las otras aplicaciones que había probado y sería una alternativa a todas las personas que usan el *método Whatsapp*. Para ser una propuesta de alternativa seria en el mercado de aplicaciones, mi aplicación tendrá que tener aspectos que la diferencien del resto y presenten una ventaja ante sus competidores, esto se detallará en los siguientes apartados.

## 1.2 Objetivos

Considerando lo anterior, se procede a detallar los objetivos que la aplicación debería cumplir.

La aplicación debe estar disponible para dispositivos móviles, y en ella cada usuario podrá organizar su vida diaria. Para usar la aplicación cada usuario deberá registrarse en el sistema de forma que su trabajo queda asociado a una cuenta en servidores remotos y está accesible en tiempo real en cualquier dispositivo con esa cuenta. De esta forma no existe limitación de usar un sólo dispositivo ni el riesgo de pérdida de los datos a causa de depender del almacenamiento local.

Una vez creada una cuenta, el usuario podrá gestionar 4 tipos de elementos:

- **Eventos:** Cualquier acontecimiento en una fecha y horas determinadas. Los eventos siempre aparecen en el calendario de la aplicación.
- **Tareas:** Representan recordatorios o trabajos que deben realizarse, por lo que cuentan con dos estados: pendiente y completado. También pueden tener una fecha límite, de esta forma también aparecerán en el calendario.
- **Notas:** Cualquier recopilación de texto que sea necesaria para el usuario. Pueden vincularse con una fecha determinada, de forma que sean accesibles desde el calendario.
- **Rutinas:** Dedicadas a las actividades recurrentes todas las semanas, permiten agregar eventos y notas que se van a repetir todas las semanas en uno o más días concretos. Desde el calendario se contará con un acceso directo a la rutina asociada a cada día. Un ejemplo de uso de las rutinas es una asignatura de la universidad que ocurre todos los lunes a las 8:30.

En esta lista se pueden ver dos factores diferenciadores de la aplicación. El primero, la integración de los 4 tipos de elementos en un mismo lugar, el calendario; de esta forma se dispone de una visión más completa de cada día, pudiendo ver sus eventos, notas asociadas, tareas que tienen de límite esa fecha y un acceso a las rutinas para el día. A mayores cada uno de los 4 elementos contará con su página dedicada. El segundo factor diferenciador son las rutinas, que nos permiten separar la recurrencia de la semana y el trabajo de los acontecimientos puntuales y únicos.

Con gestión de los diferentes tipos de elementos por parte del usuario nos referimos a la creación, edición, eliminación, visualización en detalle y compartición. En el caso de las tareas, debe añadirse también el cambio de estado de la tarea. En la ilustración [1.1](#) (página 3) se muestra un [mockup](#) de ejemplo de la visualización del calendario y de la edición de un evento.



Figura 1.1: Mockup del calendario y la edición de un evento

Además, la aplicación brinda al usuario una serie de ajustes para personalizar la interfaz y formatos a su gusto. Estos ajustes son:

- Globalización
  - Formato de horas
  - Formato de fechas
  - Formato de calendario
- Apariencia
  - Tema (oscuro/claro)
  - Tamaño de texto
- Idioma de la aplicación (gestionado automáticamente)

Por último, estarán a disposición del usuario apartados de ayuda sobre el uso de la aplicación, envío de errores y soporte de cuenta como la recuperación de la contraseña en caso de olvidársela o la verificación del correo electrónico vinculado a la cuenta. En la figura 1.2 (página 4) se muestran mockups de estas opciones.

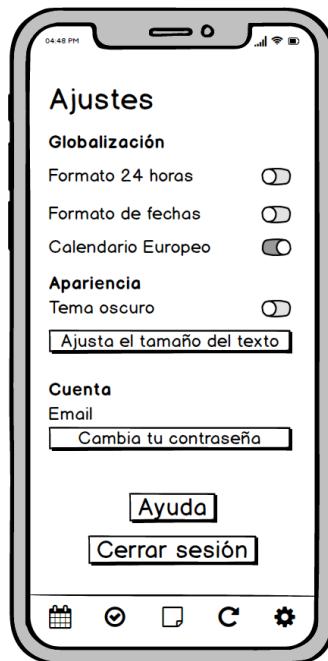


Figura 1.2: Mockup de la sección de ajustes

### 1.3 Visión global del sistema

La arquitectura del proyecto está orientada a un enfoque *Backend as a Service* [2] o *backendless*. En la figura 1.3 (página 5) se muestra un diagrama de la arquitectura del sistema. Desarrollado en *Dart* [3] y con el *framework Flutter* [4] de Google, el código fuente de la aplicación está compuesto por la interfaz, además de métodos auxiliares y llamadas a los servicios remotos.

Los servicios remotos pertenecen a Google y en este caso son: *Firebase Authentication* [5] y *Firestore Database* [6], en ellos se gestionan los usuarios y las bases de datos respectivamente.

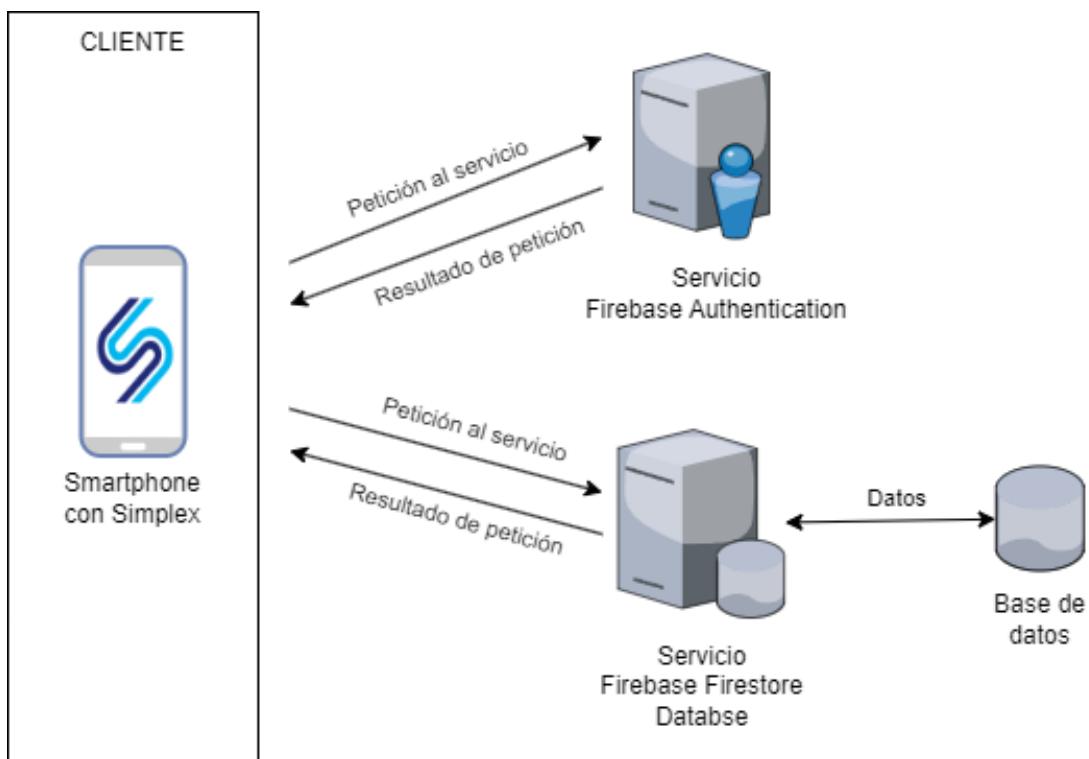


Figura 1.3: Arquitectura del sistema

## Capítulo 2

# Estado del arte

OBVIAMENTE la gestión del tiempo personal es algo que existe prácticamente desde el origen de los tiempos, por lo que el mercado de aplicaciones con dicha finalidad está muy expandido.

A continuación se muestran algunas aplicaciones dedicadas a la organización personal, de las cuales se han sacado ideas o contienen elementos que han servido de inspiración para la creación de Simplex [7].

### 2.1 Google Calendar

Google Calendar [8] es la aplicación de Google dedicada a la organización personal, y viene instalada por defecto en los dispositivos Android [9]. Su principal característica es su calendario, en el que se pueden registrar eventos, citas y recordatorios.

Su punto fuerte es la integración con diversos servicios de Google como Gmail [10]. La infor-

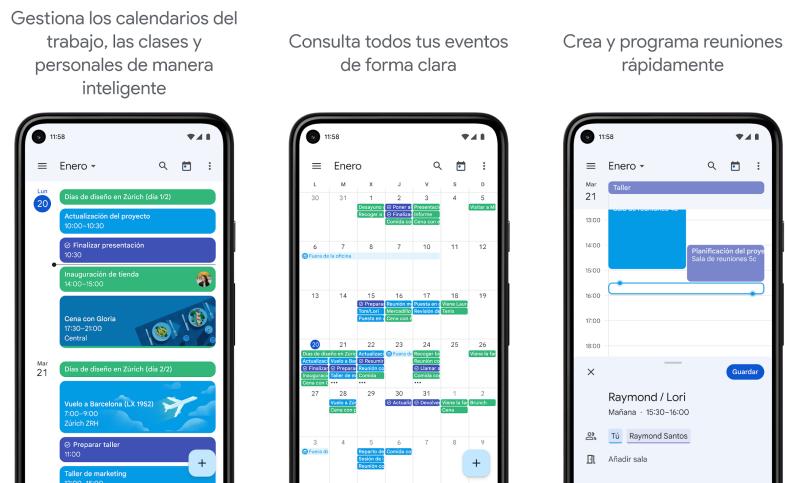


Figura 2.1: Presentación de Google Calendar en Google Play Store

mación se mantiene sincronizada con todos los dispositivos y el usuario puede recibir notificaciones de los eventos. Esto último es muy importante para la experiencia del usuario, por lo que también está implementado en Simplex [7].

A mayores dispone de características como múltiples calendarios o recordatorios inteligentes mediante inteligencia artificial. En la ilustración 2.1 (página 6) se muestra su presentación en Google Play Store [11].

## 2.2 Any.do

Any.do [12] es una aplicación multiplataforma (Android [9], iOS [13], web...) dedicada a la gestión de tareas, aunque también cuenta con un calendario.

Integrable con gran cantidad de servicios de terceros como GitHub [14] o Microsoft Teams [15], está centrada principalmente a la organización entre equipos de trabajo.

Es por esto que una de sus fortalezas es la sincronización en tiempo real, de forma que varios usuarios puedan trabajar en una misma lista a la vez.

Gracias a su calendario, se brinda al usuario la posibilidad de planificar eventos y citas.

En la ilustración 2.2 (página 7) se muestra su presentación en Google Play Store [11].

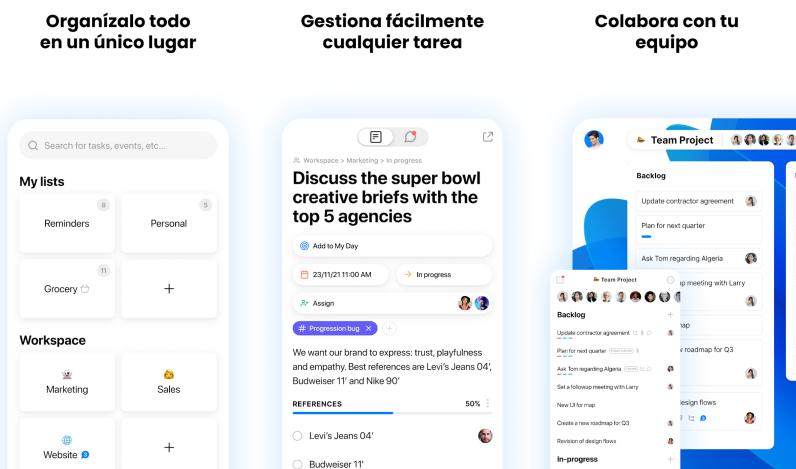


Figura 2.2: Presentación de Any.do en Google Play Store

## 2.3 Evernote

Evernote [16] es una de las aplicaciones de notas más usadas en el mundo.

Se centra exclusivamente en el manejo de notas, por lo que disponemos de un gran abanico de herramientas dedicadas a ellas. Existen las notas multimedia, que pueden contener audio, imágenes o archivos adjuntos. También es posible el etiquetado de notas para su clasificación.

Un punto muy relevante de esta aplicación que se ha aplicado también en Simplex [7] es la búsqueda de notas mediante cadenas de texto, de forma que sean más accesibles.

En la ilustración 2.3 (página 8) se muestra su presentación en Google Play Store [11].

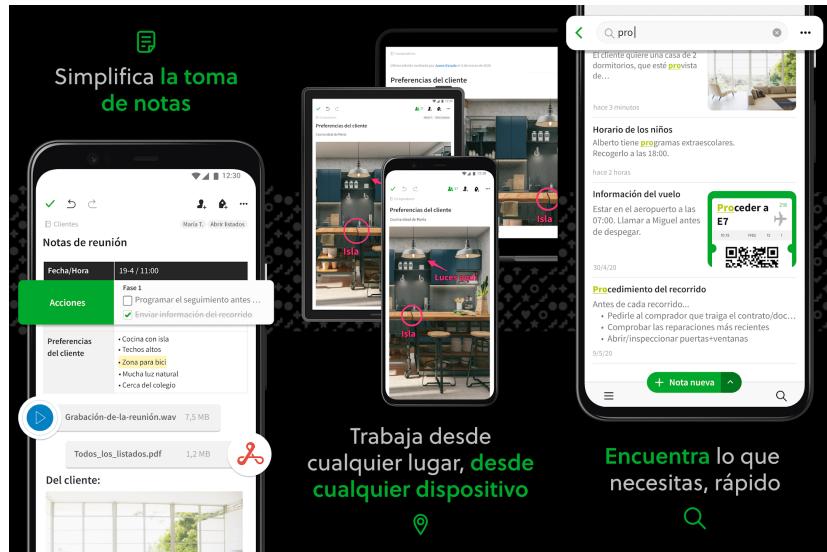


Figura 2.3: Presentación de Evernote en Google Play Store

## 2.4 Otras aplicaciones

Se han tomado de referencia muchas más aplicaciones, algunas de las más importantes y usadas en la actualidad son:

- Microsoft To Do [17]
- Todoist [18]
- Fantastical [19]

## 2.5 Detección de elementos clave

Tras el análisis de las aplicaciones expuestas, es necesario extraer los puntos clave que supongan una gran ventaja de cada aplicación. Además, también se deben detectar todos aquellos elementos básicos y comunes para que una aplicación dedicada a la organización personal sea funcional y cumpla los requisitos mínimos de cualquier usuario.

Aquí se listan algunos de los elementos clave extraídos del análisis y que Simplex [7] debe cumplir:

- Disponer de un calendario
- Sincronización en tiempo real
- Búsqueda de elementos mediante cadenas de texto
- Almacenamiento remoto y gestión de cuentas de usuario
- Sistema de notificaciones

## Capítulo 3

# Metodología

---

**E**n este capítulo se detalla la metodología de trabajo empleada para el desarrollo del proyecto, así como las razones de dicha elección.

### 3.1 Metodología escogida

Para este proyecto se ha escogido una metodología iterativa, en la que se entrega el producto software por funcionalidades de forma que se va completando poco a poco.

El primer paso en la aplicación de la metodología es realizar un análisis de requisitos global, en el que se recogen todos los requisitos que el producto debe cumplir, así como qué características debería tener.

Una vez recopilados los requisitos se planifica el proyecto, de forma que se dividirá en iteraciones o entregas en las que cada una abarca un conjunto de funcionalidades.

Para cada fase se realizan tareas de análisis, diseño, implementación y pruebas en ese orden. Una vez completadas todas las fases de la iteración, esta se considera completada, de forma que se añade al resto del proyecto siendo esto un incremento de las funcionalidades de la aplicación.

Este tipo de metodología presenta una gran cantidad de ventajas, algunas de las cuales son:

- **Entrega progresiva:** Dado que cada iteración supone un incremento de funcionalidad del producto, los usuarios pueden comenzar a usar el sistema en una fase temprana e ir recibiendo nuevas funcionalidades progresivamente sin necesidad de esperar a que el producto esté totalmente acabado.
- **Adaptabilidad:** Es más fácil aplicar cambios y añadir o eliminar requisitos en mitad del desarrollo gracias que el proyecto se realiza en varias entregas y no en una sola.
- **Mejora continua:** Gracias a la revisión periódica y la retroalimentación, se pueden detectar áreas de mejora en el proyecto.

- **Mayor facilidad de planificación:** Ya que cada iteración es planificada individualmente, se realiza una planificación periódica a corto plazo y con mayor precisión, ya que los objetivos se definen para cada ciclo y estos son mucho más claros y precisos que aquellos que se definan para todo el proyecto global.
- **Calidad del producto y reducción de riesgos:** Con las pruebas en cada iteración, se está contribuyendo a la mejora continua y con esto a su vez reduce las posibilidades de errores significativos, lo cual aumenta la calidad del producto y le da mayor valor.

## 3.2 Detalles de la metodología

Partiendo de que se emplea una metodología iterativa, es necesario aclarar varios puntos del desarrollo del proyecto.

### 3.2.1 Tablero Kanban

Para una mejor organización de las tareas de cada iteración, éstas se han distribuido en un tablero Kanban empleando la herramienta web Taiga. El tablero consta de 5 fases para las tareas: Nueva, Preparada, En Curso, Reparación/Optimización y Hecha.

A continuación, en la figura 3.1 (página 11), se muestra una instantánea del tablero Kanban en pleno desarrollo de la aplicación:

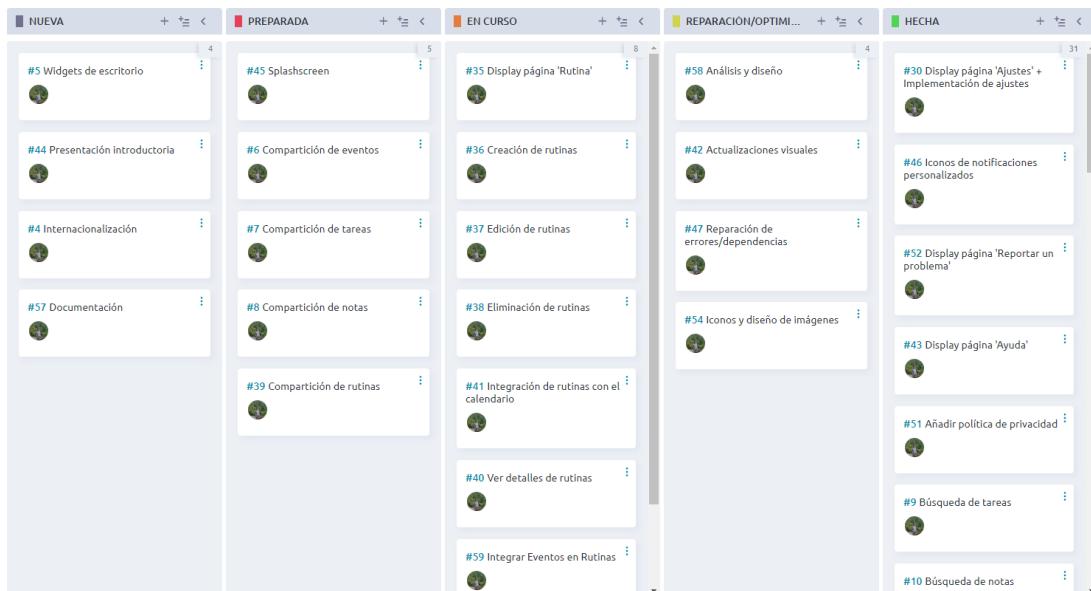


Figura 3.1: Tablero Kanban durante el desarrollo del proyecto

### 3.2.2 Pruebas en dispositivos

Las pruebas automatizadas para aplicaciones en el entorno de [Flutter](#) [4] siguen los fundamentos de las pruebas de interfaz web con herramientas como [Selenium](#) [20]. Dado esto, se ha llegado a la conclusión de que la realización de estas pruebas en toda la aplicación supondría un aumento muy considerable del tiempo de desarrollo, lo cual no es estrictamente necesario para este caso.

La decisión tomada en cuanto a las pruebas ha sido crear pruebas automatizadas para verificar la comunicación con los servicios de Google y la ejecución de los métodos. Para comprobar el funcionamiento de la interfaz gráfica, se han hecho pruebas manuales en varios tipos de dispositivos móviles. La elección de estos sistemas se basa en la diversidad de características, ya que, siendo todos sistemas [Android](#) [9], cada uno de ellos es de una generación distinta, con diferentes versiones de sistema operativo, protocolos de funcionamiento, prestaciones y proporciones de pantalla.

La lista de dispositivos en los que se ha probado la aplicación tras cada iteración está disponible en la tabla 3.1 (página 12).

Dispositivo	Año	Pantalla	Android
<i>Xiaomi Redmi Note 8T</i>	2019	6,3"	Android 11
<i>Google Pixel 4a</i>	2020	5,81"	Android 11
<i>Xiaomi Redmi Note 9 Pro</i>	2020	6,67"	Android 12
<i>Vivo x51 5g</i>	2020	6,56"	Android 10
<i>Xiaomi POCO X3 Pro</i>	2021	6,67"	Android 11
<i>Xiaomi Redmi Note 11 Pro</i>	2022	6,67"	Android 13

Tabla 3.1: Dispositivos utilizados para las pruebas de la aplicación

### 3.2.3 Seguimiento del tiempo

Adicionalmente se ha decidido llevar un seguimiento del tiempo invertido en el proyecto, por lo que se ha modelado una hoja de cálculo en la que se registra el tiempo invertido en cada tarea cada día de desarrollo a modo de [gestor de tiempo](#).

Las tareas han sido numeradas con un identificador único en el tablero [Kanban](#), por lo que en la hoja de cálculo se hace referencia a estos identificadores para referirse a cada tarea y asignarle una entrada de trabajo.

Las entradas diarias de trabajo son recopiladas en la misma hoja de cálculo y procesadas para calcular métricas como tiempo total invertido, veces realizadas tareas de análisis o promedio

de horas invertidas cada día de trabajo. Gracias a estas métricas es posible estimar el ritmo de avance del proyecto y detectar posibles mejoras en la forma de trabajo.

En la figura 3.2 (página 13) se muestra una vista del documento de seguimiento durante el desarrollo del proyecto:

A	B	C	D	E	F	G
Fecha	Tarea (#)	Tiempo (h)	Tipo de tarea	Total día (h)		
2			2022			
3	31/05/22	58	4	Análisis/diseño	4	Tiempos (h)
4						2022
5	01/06/22	58	4	Análisis/diseño	4	143,5
6						2023
7	02/06/22	58	3	Análisis/diseño	3	142,3
8						
9	03/06/22	53	6	Desarrollo	6	Total
10						285,8
11		30	2	Desarrollo		
12	04/06/22	53	2	Desarrollo		Promedio diario
13		17	1	Desarrollo	6	3,6
14		12	1	Desarrollo		
15						Tareas
16	05/06/22	12	4	Desarrollo		Análisis/diseño
17		47	1	Arreglo/corrección	5	4
18						
19		31	2	Desarrollo		Desarrollo
20	06/06/22	53	1	Desarrollo		108
21		22	2	Desarrollo		
22		15	2	Desarrollo	7	Arreglo/corrección
23						52
24	07/06/22	17	2	Desarrollo	3	

Figura 3.2: Gestor de tiempo durante el desarrollo del proyecto

### 3.3 Razonamiento de elección

En la elección de metodología se han valorado 3 opciones: Metodología en Cascada, [Scrum](#) y la metodología iterativa.

La metodología en cascada ha sido descartada prácticamente desde un principio dado que se considera un desarrollo lineal de todas las funcionalidades del proyecto, de forma que no se puede disfrutar de algunas de las ventajas mencionadas anteriormente del desarrollo iterativo. Además, un desarrollo continuo puede dar lugar a situaciones como el arrastre de errores a lo largo del desarrollo o una mala adaptación a cambios en los requisitos.

[Scrum](#) sería la metodología idónea de tratarse de un proyecto conjunto, ya que esta metodología está pensada para agilizar el proyecto dentro de grupos de trabajo empleando elementos como tableros [Kanban](#) e iteraciones (llamadas sprints) además de contar con reuniones para determinar el estado del proyecto y roles dentro de dicho grupo. Pero dado que se trata de un proyecto individual, debe descartarse esta metodología.

Finalmente se decide aplicar una metodología iterativa basada en [Scrum](#), pero con algunas diferencias que la hacen más sencilla. Como [Scrum](#) se queda grande para este proyecto, no se emplean roles, reuniones o artefactos [Scrum](#), pero sí se realiza un análisis de requisitos global y posteriormente se divide el desarrollo en varias iteraciones. Cada iteración representa un conjunto de requisitos y también conlleva un análisis detallado, implementación y pruebas.

## Capítulo 4

# Análisis de requisitos global

---

**A**quí se detallarán los diferentes actores que pueden interactuar con el sistema. También se listarán los requisitos globales del sistema recogidos como casos de uso.

## 4.1 Actores

Sólo existen 3 tipos de actores en esta aplicación, y los 3 son usuarios dado que el sistema no está pensado para poseer ningún rol de administración dentro de la propia aplicación. Los actores mencionados son:

- **Usuario no registrado:** Lo único que puede hacer este actor es crearse una cuenta para comenzar a usar la aplicación mediante el formulario de registro y verificando su dirección de correo electrónico.
- **Usuario registrado:** Cualquier persona que se haya registrado en la aplicación. Puede disfrutar de todas las funcionalidades que ofrece Simplex [7] sin límite alguno. También tiene la posibilidad de crearse más cuentas con otros correos electrónicos si lo desea.
- **Tester:** Posee las mismas capacidades que un usuario registrado, ya que también lo es. Adicionalmente tendrá acceso a una sección de métricas sobre el rendimiento de la aplicación a nivel de desarrollador desde la sección de ajustes.

## 4.2 Casos de uso

La siguiente lista representa los casos de uso de la aplicación. Cada caso de uso reúne una serie de funcionalidades que serán descritas para cada apartado y, en algunos casos, se adjuntará también algún [mockup](#) de su diseño.

Cabe mencionar que las tareas de Taiga [21] no se relacionan directamente con los casos de uso, sino que un caso de uso puede estar desgranado en varias tareas en Taiga [21].

- **CU-01 Registro de usuarios:** Un usuario podrá registrarse indicando su correo electrónico y una contraseña. Una vez verificado su dirección de correo electrónico ya podrá hacer uso de la aplicación. En la ilustración 4.1 (página 15) se muestran los mockups de las ventanas de autenticación.
- **CU-02 Autenticación de usuarios:** Para iniciar sesión, el usuario deberá indicar el correo electrónico y contraseña con los que se ha registrado.
- **CU-03 Cierre de sesión:** Un usuario autenticado puede cerrar sesión en la aplicación. Posteriormente será redirigido a la página de inicio de sesión y registro.
- **CU-04 Cambio de contraseña:** Un usuario autenticado puede cambiar su contraseña desde la sección de ajustes.
- **CU-05 Recuperación de contraseña:** Si un usuario no autenticado no recuerda su contraseña, puede recuperarla mediante un email de recuperación a su dirección de correo electrónico.

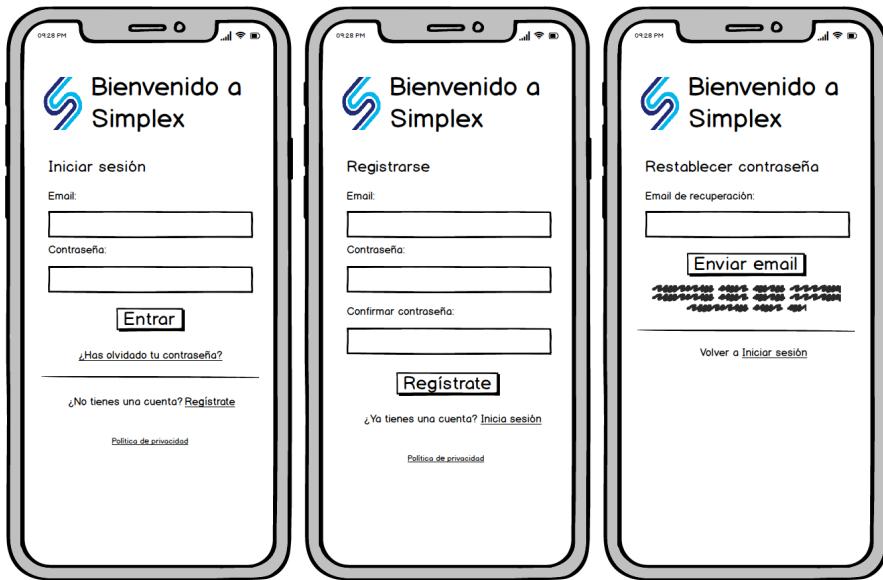


Figura 4.1: Mockup de las ventanas previas a autenticarse

- **CU-06 Ver calendario y eventos:** Cada usuario tendrá acceso al calendario y una lista de los eventos de cada día, mostrando su título, descripción y hora.
- **CU-07 Ver detalles de eventos:** Si un usuario selecciona un evento, podrá ver todos sus detalles. Esto incluye los campos: título, descripción, tipo (una vez o rutina), fecha, hora, color y lista de notificaciones.

- **CU-08 Crear eventos:** El usuario podrá crear eventos indicando, al menos, todos sus campos obligatorios.
- **CU-09 Editar eventos:** El usuario también podrá editar los campos de un evento de la misma forma que durante su creación. También podrá cancelar los cambios.
- **CU-10 Eliminar eventos:** Podrá eliminarse un evento seleccionándolo y escogiendo dicha opción. También será posible acceder a esa opción mediante un menú desplegable manteniendo pulsado en el evento.
- **CU-11 Recibir notificaciones de eventos:** El usuario podrá programar notificaciones a modo de recordatorio sobre el evento en las fechas que desee. En la figura 4.2 (página 16) se muestra un ejemplo de notificaciones.



Figura 4.2: Mockup de ejemplo de notificaciones

- **CU-12 Compartir eventos:** Se podrá compartir un evento como un bloque de texto, que incluirá su título y demás información relevante. Las aplicaciones disponibles para compartir son todas las que el usuario tenga instaladas y soporten introducción de texto.
- **CU-13 Ver lista de tareas:** Cada usuario tendrá acceso a una lista con sus tareas, agrupadas por su estado: pendiente o completada. De cada tarea se muestra su título, descripción y fecha límite.
- **CU-14 Ver detalles de tareas:** Si un usuario selecciona una tarea, podrá ver todos sus detalles. Esto incluye los campos: título, descripción, estado (pendiente o completada), prioridad y fecha límite.
- **CU-15 Crear tareas:** El usuario podrá crear tareas indicando, al menos, todos sus campos obligatorios.
- **CU-16 Editar tareas:** El usuario también podrá editar los campos de una tarea de la misma forma que durante su creación. También podrá cancelar los cambios.

- **CU-17 Eliminar tareas:** Podrá eliminarse una tarea seleccionándola y escogiendo dicha opción. También será posible acceder a esa opción mediante un menú desplegable manteniendo pulsado en la tarea. Adicionalmente también habrá un botón para eliminar todas las tareas completadas.
- **CU-18 Recibir notificaciones de tareas:** El usuario recibirá notificaciones de una tarea conforme ésta se acerca a su fecha límite (si tiene una).
- **CU-19 Compartir tareas:** Se podrá compartir una tarea como un bloque de texto, que incluirá su título y demás información relevante. Las aplicaciones disponibles para compartir son todas las que el usuario tenga instaladas y soporten introducción de texto. En la ilustración 4.4 (página 18) se muestra un ejemplo de tarea compartida.
- **CU-20 Buscar tareas por texto:** El usuario puede filtrar las tareas por título mediante una entrada de texto.
- **CU-21 Añadir tareas al calendario:** Podrán añadirse tareas al calendario indicando su fecha límite. Estarán disponibles en la lista de elementos del día indicado y se podrá acceder a sus detalles desde allí también. En la ilustración 4.3 (página 17) se muestran los mockups de las ventanas tareas.



Figura 4.3: Mockup de gestión de las tareas

- **CU-22 Ver lista de notas:** Cada usuario tendrá acceso a una lista con sus notas, ordenadas por su fecha de modificación. De cada nota se muestra su título, contenido y fecha de edición.

- **CU-23 Ver detalles de notas:** Si un usuario selecciona una nota, podrá ver todos sus detalles. Esto incluye los campos: título, contenido, fecha en el calendario y relación con las rutinas.
- **CU-24 Crear notas:** El usuario podrá crear notas, no tienen ningún campo obligatorio.
- **CU-25 Editar notas:** El usuario también podrá editar los campos de una nota de la misma forma que durante su creación. También podrá cancelar los cambios.
- **CU-26 Eliminar notas:** Podrá eliminarse una nota seleccionándola y escogiendo dicha opción. También será posible acceder a esa opción mediante un menú desplegable manteniendo pulsado en la nota.
- **CU-27 Recibir notificaciones de notas:** El usuario recibirá notificaciones de una nota conforme ésta se acerca a su fecha límite (si tiene una).
- **CU-28 Compartir notas:** Se podrá compartir una nota como un bloque de texto, que incluirá su título y demás información relevante. Las aplicaciones disponibles para compartir son todas las que el usuario tenga instaladas y soporten introducción de texto.
- **CU-29 Buscar notas por texto:** El usuario puede filtrar las notas por título mediante una entrada de texto.
- **CU-30 Añadir notas al calendario:** Podrán añadirse notas al calendario indicando una fecha de calendario. Estarán disponibles en la lista de elementos del día indicado y se podrá acceder a sus detalles desde allí también.

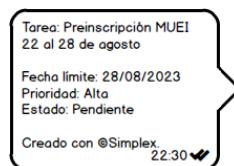


Figura 4.4: Mockup de una tarea compartida como mensaje de Whatsapp

- **CU-31 Ver lista de rutinas:** Cada usuario tendrá acceso a una lista con sus rutinas, que estarán clasificadas por días de la semana. En cada día de la semana se mostrarán los eventos y notas pertinentes. En la figura 4.5 (página 19) se muestra un mockup de la vista de rutina.
- **CU-32 Crear rutinas:** El usuario podrá crear eventos y notas de rutinas, deberá indicar a qué días de la semana pertenecen.

- **CU-33 Editar rutinas:** El usuario también podrá editar los campos de una rutina de la misma forma que edita los de un evento o nota normal.
- **CU-34 Eliminar rutinas:** Podrá eliminarse una rutina seleccionándola y escogiendo dicha opción. También será posible acceder a esa opción mediante un menú desplegable manteniendo pulsado en la nota.
- **CU-35 Compartir rutinas:** Se podrá compartir una rutina del mismo modo que se comparten los eventos y notas normales.

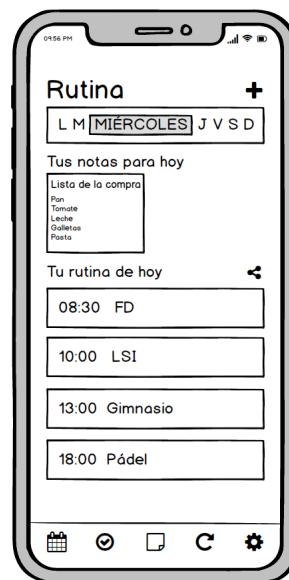


Figura 4.5: Mockup de la vista de rutina

- **CU-36 Ver sección de ajustes:** El usuario tendrá acceso a una sección de ajustes en la que podrá configurar algunos aspectos de la aplicación.
- **CU-37 Ajustar tamaño de texto:** Podrá modificarse el tamaño de la fuente de la aplicación al gusto del usuario.
- **CU-38 Ver sección de ayuda:** Estará disponible un apartado de ayuda para que el usuario consulte cualquier duda acerca del uso de la aplicación.
- **CU-39 Reportar un problema:** Podrán enviarse reportes de errores y problemas a través de un formulario. El usuario podrá decidir si adjuntar información de la cuenta para una revisión más eficaz del incidente.

- **CU-40 Ver estadísticas de tester:** Los usuarios designados como **testers** tendrán acceso a un apartado de estadísticas acerca del uso de la aplicación y de los reportes de errores. En la ilustración 4.6 (página 20) se pueden ver varios **mockups**, uno de ellos de la página de estadísticas.
- **CU-41 Internacionalización:** La aplicación determinará en qué idioma establecer el texto en función del idioma del dispositivo del usuario. Los idiomas disponibles son: inglés (por defecto), castellano y gallego.



Figura 4.6: Mockup de las vistas de ayuda, estadísticas y reportar un problema

# Capítulo 5

# Planificación

---

**E**n este apartado se detallarán las diferentes iteraciones que componen al proyecto, listando los casos de uso y actividades realizadas en cada una de ellas. También se hará un cálculo del coste del proyecto.

## 5.1 Iteraciones

El proyecto se ha dividido en un total de 10 iteraciones, que serán detalladas a continuación. Cabe mencionar que en todas las iteraciones se han realizado correcciones y mejoras de otras iteraciones anteriores como parte de la mejora continua.

### 5.1.1 Iteración 0

La primera iteración tiene como objetivo la definición del producto. Se ha realizado un análisis global de los requisitos que ha de cumplir la aplicación y se han dividido dichos requisitos en las siguientes iteraciones. También se han hecho [mockups](#) como los adjuntados a lo largo de esta memoria para tener una imagen provisional del aspecto que debería tener la interfaz del sistema. Esta iteración supone el comienzo del proyecto y se ha iniciado el 31 de mayo de 2022.

### 5.1.2 Iteración 1

La segunda iteración del proyecto consiste en la formación en las tecnologías para el proyecto. En este caso, a pesar de haber trabajado con alguna de ellas con anterioridad en alguna asignatura, ha supuesto un trabajo muy exhaustivo. En realidad la formación no ha durado solamente el tiempo de la iteración, sino que ha sido un proceso continuo de aprendizaje a lo largo de todo el proyecto. La iteración 1 supone la formación en las tecnologías hasta estar listo para comenzar a desarrollar la base de la aplicación.

Las tecnologías en las que se ha necesitado formación han sido: Dart [3] (framework de Flutter [4]), Gradle [22] (para las configuraciones del proyecto Android [9]) y la plataforma Google Firebase [23].

### 5.1.3 Iteración 2

Esta es la primera la primera iteración de desarrollo. Su principal objetivo es montar el esqueleto del proyecto y construir la base de la aplicación móvil. Además, se desarrolla la ventana de ajustes y toda su funcionalidad. Es necesario mencionar que, a pesar de implementar la vista de *Ayuda* en esta iteración, sus secciones acerca de cada parte de la aplicación se han ido completando con sus respectivas iteraciones. La lista de casos de uso de la iteración se muestra en la tabla 5.1 (página 22):

Identificador	Caso de uso
CU-36	Ver sección de ajustes
CU-37	Ajustar tamaño de texto
CU-38	Ver sección de ayuda

Tabla 5.1: Casos de uso de la iteración 2

### 5.1.4 Iteración 3

El siguiente paso a la construcción de la base de la aplicación es enlazarla con los servicios de Google Firebase [23]. En esta iteración se ha configurado todo el proyecto en Google Firebase [23], se han establecido las bases de datos, se ha activado la gestión de usuarios y se han definido las reglas de acceso a los servicios remotos.

Además, también se ha implementado toda la funcionalidad relacionada con la gestión de usuarios en la aplicación. En la tabla 5.2 (página 22) se muestran los casos de uso realizados:

Identificador	Caso de uso
CU-01	Registro de usuarios
CU-02	Autenticación de usuarios
CU-03	Cierre de sesión
CU-04	Cambio de contraseña
CU-05	Recuperación de contraseña

Tabla 5.2: Casos de uso de la iteración 3

### 5.1.5 Iteración 4

A partir de esta iteración comienza a implementarse la gestión de los elementos de trabajo de la aplicación. En esta iteración se desarrolla la gestión de eventos y se implementa también el calendario de la aplicación. En la tabla 5.3 (página 23) se muestran sus casos de uso:

Identificador	Caso de uso
CU-06	Ver calendario y eventos
CU-07	Ver detalles de eventos
CU-08	Crear eventos
CU-09	Editar eventos
CU-10	Eliminar eventos
CU-11	Recibir notificaciones de eventos
CU-12	Compartir eventos

Tabla 5.3: Casos de uso de la iteración 4

### 5.1.6 Iteración 5

En esta iteración se ha desarrollado el manejo de tareas. En la tabla 5.4 (página 23) se listan los casos de uso desarrollados:

Identificador	Caso de uso
CU-13	Ver lista de tareas
CU-14	Ver detalles de tareas
CU-15	Crear tareas
CU-16	Editar tareas
CU-17	Eliminar tareas
CU-18	Recibir notificaciones de tareas
CU-19	Compartir tareas
CU-20	Buscar tareas por texto
CU-21	Añadir tareas al calendario

Tabla 5.4: Casos de uso de la iteración 5

### 5.1.7 Iteración 6

En la iteración 6 se han implementado todas las funcionalidades que rodean a las notas. Los casos de uso se pueden consultar en la tabla 5.5 (página 24):

Identificador	Caso de uso
CU-22	Ver lista de notas
CU-23	Ver detalles de notas
CU-24	Crear notas
CU-25	Editar notas
CU-26	Eliminar notas
CU-27	Recibir notificaciones de notas
CU-28	Compartir notas
CU-29	Buscar notas por texto
CU-30	Añadir notas al calendario

Tabla 5.5: Casos de uso de la iteración 6

### 5.1.8 Iteración 7

Aquí se ha desarrollado la gestión de rutinas, que integran un nuevo tipo de eventos y notas. En la tabla 5.6 (página 24) se listan sus casos de uso:

Identificador	Caso de uso
CU-31	Ver lista de rutinas
CU-32	Crear rutinas
CU-33	Editar rutinas
CU-34	Eliminar rutinas
CU-35	Compartir rutinas

Tabla 5.6: Casos de uso de la iteración 7

### 5.1.9 Iteración 8

La última iteración de desarrollo, en ella se han desarrollado varios puntos como las estadísticas de [testers](#) o la internacionalización de la aplicación.

Es importante mencionar que en esta iteración también se han realizado todos los trámites para el lanzamiento de la aplicación a Google Play Store [11]. En la tabla 5.7 (página 25) se muestran los casos de uso de la iteración:

Identificador	Caso de uso
CU-39	Reportar un problema
CU-40	Ver estadísticas de tester
CU-41	Internacionalización

Tabla 5.7: Casos de uso de la iteración 8

### 5.1.10 Iteración 9

En la última iteración del proyecto se ha redactado esta memoria, desarrollada en [Lamport TeX \(LaTeX\)](#) [24].

## 5.2 Planificación temporal

Al comienzo de cada iteración se ha calculado una estimación de su duración en horas. Al final de la iteración se sumaban todas las entradas correspondientes y se comparaba el tiempo estimado con el real.

Algunas iteraciones se alargan en el tiempo ya que el proyecto ha sufrido parones a lo largo del curso. El caso más extremo es el de la iteración 6, que comenzó en julio de 2022 y terminó en febrero de 2023, fecha que coincide con el fin de los exámenes del primer cuatrimestre del curso.

Posteriormente se ha trabajado en el proyecto a lo largo del segundo cuatrimestre y se ha terminado en verano de este año.

En la tabla 5.8 (página 26) se muestran la duración estimada y real del proyecto.

## 5.3 Cálculo de costes

A continuación se hará un cálculo del coste del proyecto, teniendo en cuenta el tiempo invertido y el equipo necesario.

El precio por hora de trabajo es una aproximación basada en los costes del mercado real, siendo de 25€ la hora.

$$\text{Coste de horas} = 352,75 \text{ horas} * 25 \text{ €/hora} = \mathbf{8.818,75 \text{ €}}$$

Iteración	F. inicio	F. fin	Estimación	T. real
Iteración 0	31/05/2022	01/06/2022	10 h	11 h
Iteración 1	02/06/2022	04/06/2022	10 h	8 h
Iteración 2	05/06/2022	15/06/2022	40 h	29,5 h
Iteración 3	16/06/2022	29/06/2022	30 h	26 h
Iteración 4	30/06/2022	13/07/2022	40 h	45,75 h
Iteración 5	14/07/2022	26/07/2022	30 h	42,75 h
Iteración 6	27/07/2022	25/02/2023	30 h	26,25 h
Iteración 7	26/02/2023	11/04/2023	50 h	66 h
Iteración 8	12/04/2023	02/08/2023	50 h	57,5 h
Iteración 9	03/08/2023	23/08/2023	30 h	40 h
<b>TOTAL</b>	<b>31/05/2022</b>	<b>15/08/2023</b>	<b>320 h</b>	<b>352,75 h</b>

Tabla 5.8: Planificación temporal del proyecto y tiempo real invertido

Para simular un caso real, deberíamos considerar que el proyecto se desarrolla a tiempo completo sin períodos de pausa. Para llegar a este cálculo debemos tener en cuenta el artículo 34 del Real Decreto Legislativo 2/2015 [25], en el que se dice que en España la duración máxima que puede tener una jornada de trabajo es de 40 horas semanales, por lo que consideraremos una jornada de 160 horas al mes:

$$\text{Duración real} = 352,75 \text{ horas} / 160 \text{ horas/mes} = 2,2 \text{ meses}$$

Es decir, la duración real del proyecto sería de 2 meses y una semana aproximadamente. El equipamiento y recursos utilizados para el proyecto han sido:

- **Equipamiento informático:** Se ha empleado un ordenador portátil de altas prestaciones para poder emular correctamente la aplicación durante el proceso de desarrollo, éste ha tenido un coste de 1.250 €. Además, también se han adquirido un monitor y un teclado y ratón inalámbricos, con un coste de 200 €. Como este equipamiento se aprovechará para otros proyectos, a este proyecto se le atribuirá el 25% del coste, lo que corresponde a **362,5 €** en equipamiento informático.
- **Acceso a Internet:** Contratando una tarifa básica de Internet con un coste de 29,95 €/mes, el coste del acceso a Internet durante 3 meses es de **89,85 €**.

- **Oficina de trabajo:** Se ha alquilado una oficina para trabajar en el proyecto. El coste del alquiler ha sido de 200 € al mes, lo que supone **600 €** ya que es necesario un contrato de 3 meses.
- **Electricidad:** La electricidad tiene un coste de **90 €** en 3 meses.

Una vez recabados todos los gastos, se suman para calcular el coste total del proyecto. En la tabla 5.9 (página 27) se muestra el desglose de costes y el coste total del proyecto:

Concepto	Coste
<i>Horas de trabajo</i>	8.818,75 €
<i>Equipamiento informático</i>	362,5 €
<i>Acceso a Internet</i>	89,85 €
<i>Oficina de trabajo</i>	600 €
<i>Electricidad</i>	90 €
<b>COSTE TOTAL</b>	<b>9.961,1 €</b>

Tabla 5.9: Desglose del coste total del proyecto

El coste total del proyecto ha sido de **9.961,1 €**.

## Capítulo 6

# Fundamentos tecnológicos

---

A continuación se mostrarán todas las tecnologías que se han usado durante el desarrollo del proyecto. Estas tecnologías pueden ser divididas en 3 conjuntos:

## 6.1 Tecnologías usadas para el código fuente

### 6.1.1 Android Studio

Android Studio [26] es un IDE creado por JetBrains [27] y Google destinado al desarrollo de aplicaciones Android [9], aunque gracias a frameworks como Flutter [4] es posible que con el mismo código fuente las aplicaciones sean funcionales en más plataformas.

Para un correcto desarrollo de las aplicaciones, este IDE permite probarlas en tiempo real en dispositivos móviles emulados por el ordenador. Además también permite probarlas en dispositivos móviles reales y depurarlas vía USB.

### 6.1.2 Dart

Dart [3] es el lenguaje de programación escogido para el desarrollo de la aplicación, ya que se integra con Flutter [4] para la creación de aplicaciones móviles en la actualidad. Es un lenguaje compilado, y se ha empleado para la base de la aplicación, para el desarrollo de la interfaz y para la creación de funciones y llamadas a los servicios remotos.

Proporciona soporte para la Programación Orientada a Objetos (POO), lo cual ha sido ideal para el desarrollo del sistema.

### 6.1.3 Flutter

Flutter [4] es un framework desarrollado por Google. Tiene la capacidad de crear aplicaciones para móviles con distintos sistemas operativos con el mismo código fuente, incluso

permite crear aplicaciones de escritorio. Integra varios componentes para poder crear las aplicaciones, entre ellos están [Dart \[3\]](#) y un gran número de bibliotecas por defecto.

Una de sus fortalezas es la posibilidad de crear [widgets](#) personalizados en [Dart \[3\]](#), a continuación se muestra un ejemplo:

```

1 Column AuthHeader(BuildContext context){
2   return Column(
3     children: [
4       SizedBox(
5         height: deviceHeight * 0.01,
6       ),
7       Row(
8         children: [
9           Image.asset('assets/icon_no_background.png', scale:
deviceWidth*0.0175, ),
10          SizedBox(width: deviceWidth*0.02, ),
11          Container(
12            width: deviceWidth*0.625,
13            child: Text(AppLocalizations.of(context)!.welcome,
14              style: TextStyle(
15                color: colorMainText,
16                fontSize: deviceWidth * fontSize * 0.1,
17                fontWeight: FontWeight.bold)),
18            ),
19          ],
20        ),
21      ],
22    );
23 }
```

#### 6.1.4 Gradle

[Gradle \[22\]](#) es la herramienta utilizada para la automatización de las construcciones de la aplicaciones, así como del manejo de sus dependencias. En [Android Studio \[26\]](#) ésta es la herramienta estándar de construcción de los proyectos.

El proyecto contiene dos archivos de configuración [Gradle \[22\]](#):

- **android/build.gradle:** Destinado a la configuración de aspectos del proyecto como conjunto. Por ejemplo, la configuración de versiones de [Gradle \[22\]](#) o declaración de los repositorios de las dependencias.
- **android/app/build.gradle:** Para la configuración de aspectos específicos de la aplicación, como su construcción o dependencias.

### 6.1.5 XML

eXtensible Markup Language (**XML**), es un lenguaje que se emplea para organizar y listar elementos. Se ha usado en el proyecto para varios aspectos:

- **Configuración de la aplicación:** El archivo **android/app/src/main/AndroidManifest.xml** es uno de los más importantes del proyecto y en él se configura la aplicación y sus características, se solicitan permisos de uso del sistema y se definen las actividades principales y secundarias entre otras cosas.
- **Recursos y aspecto de la aplicación:** Mediante archivos en las carpetas como **android/app/src/main/res/drawable** o **android/app/src/main/res/mipmap...** se establecen los recursos utilizados para el ícono de la aplicación o sus imágenes de inicio. También se definen otros campos visuales como paletas de colores por defecto o estructura de las notificaciones.

### 6.1.6 JSON

JavaScript Object Notation (**JSON**) [28], es un formato dedicado al intercambio de datos. Se ha empleado este formato para el intercambio de objetos e información con las bases de datos remotas de la aplicación.

### 6.1.7 Application Resource Bundle

Los archivos **Application Resource Bundle (ARB)** son un formato dedicado íntegramente a la **internacionalización** de las aplicaciones. Se ha creado un fichero de este tipo por cada idioma soportado, y en él se encuentran sus respectivas entradas de texto.

## 6.2 Servicios remotos

### 6.2.1 Google Firebase Authentication

**Firebase Authentication** [5] es uno de los servicios empleados en Simplex [7], contenido dentro de **Google Firebase** [23]. Este servicio permite gestionar los usuarios, desde inicios de sesión hasta verificación de correos electrónicos o cambios de contraseña.

Los métodos que proporciona permiten que la autenticación de los usuarios se haga de forma completamente segura. Todas las solicitudes al servicio son resueltas en tiempo real.

En la ilustración 6.1 (página 31) se muestra una vista del servicio en la web.

The screenshot shows the 'Authentication' section of the Google Firebase console. At the top, there are tabs for 'Users', 'Sign-in method', 'Templates', 'Usage', and 'Settings'. Below the tabs is a search bar and a button labeled 'Agregar usuario'. A table lists five users with the following data:

Identificador	Proveedores	Fecha de creación	Fecha de acceso	UID de usuario
[REDACTED]@gmail.com	[REDACTED]	9 ago 2023	9 ago 2023	dsXVgg[REDACTED]
[REDACTED]@gm..	[REDACTED]	9 ago 2023	9 ago 2023	OubWTI[REDACTED]
[REDACTED]@gmail.com	[REDACTED]	8 ago 2023	8 ago 2023	Bfy4lp8[REDACTED]
[REDACTED]@gmail.com	[REDACTED]	8 ago 2023	8 ago 2023	AEKhna[REDACTED]
[REDACTED]@gmail.com	[REDACTED]	8 ago 2023	8 ago 2023	YTjgYeig[REDACTED]

Figura 6.1: Vista web de administrador del servicio Google Firebase Authentication

### 6.2.2 Google Firestore Database

[Firestore Database](#) [6] es la base de datos escogida para el almacenamiento de los datos en la nube. También está integrada en [Google Firebase](#) [23] y permite sincronización en tiempo real, cacheado de datos en momentos sin conexión a Internet, acceso seguro mediante reglas y una organización por jerarquía de documentos y colecciones.

En la ilustración 6.2 (página 31) se puede ver el portal de administración web del servicio.

The screenshot shows the 'Cloud Firestore' interface. At the top, there are tabs for 'Datos', 'Reglas', 'Índices', 'Uso', and 'Extensões'. Below the tabs is a navigation bar with 'Vista del panel' and 'Compilador de consultas'. The main area shows a hierarchical view of documents under 'users' and a detailed view of a specific document '6007795' with fields like 'description', 'done', 'id', 'limitDate', 'limited', 'name', and 'priority'.

Figura 6.2: Vista web de administrador del servicio Google Firestore Database

### 6.2.3 Google Play Console

[Google Play Console](#) [29] es la plataforma a la que se sube la aplicación para su publicación en [Google Play Store](#) [11]. Desde la web se pueden crear canales de pruebas para [testers](#), gestionar las versiones de producción, configurar la apariencia de la aplicación en la tienda o consultar gran variedad de métricas acerca de la aplicación y sus usuarios.

En la ilustración 6.3 (página 32) se muestra una vista de la plataforma.

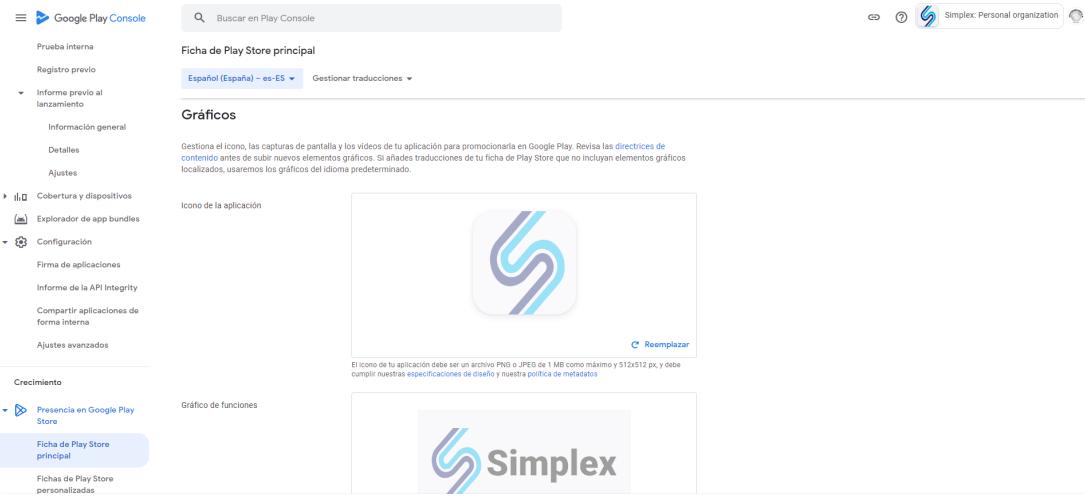


Figura 6.3: Vista web de la configuración de la ficha de Google Play Store

## 6.3 Otras tecnologías

### 6.3.1 Git

Git [30] es un Sistema de Control de Versiones (SCV) en el que se guarda el repositorio remoto del proyecto. En él se encuentra el historial de commits y la lista de versiones de lanzamiento de la aplicación. El repositorio es de acceso público para que cualquier interesado pueda consultar el código fuente.

### 6.3.2 Taiga

La herramienta web Taiga [21] se ha empleado para gestionar las tareas de desarrollo del proyecto. Se ha empleado un tablero Kanban como se ha mencionado anteriormente, este tablero se puede ver en la figura 3.1 (página 11).

### 6.3.3 Adobe Photoshop

Finalmente, se ha usado la herramienta gráfica Adobe Photoshop [31] para el diseño de iconos y logotipos, además de preparar los gráficos necesarios para la publicación de la aplicación en Google Play Store [11].

## Capítulo 7

# Desarrollo iterativo

---

EN este capítulo se detallará el desarrollo de la aplicación desde los puntos de vista técnico y funcional. Se describirá la estructura del proyecto, su modelo de base de datos, su colección de usuarios y también se tratará en profundidad el desarrollo de cada iteración.

## 7.1 Estructura del proyecto

En la estructura del proyecto no se consideran los servicios de [Firebase Authentication](#) [5] y [Firestore Database](#) [6].

Ya se ha mencionado anteriormente que no existe una división tradicional de [backend](#) y [frontend](#), por lo que la estructura del proyecto queda de la siguiente forma:

- **/android:** Es el directorio que contiene la base y la configuración de la aplicación para que funcione correctamente en [Android](#) [9]. Aquí se han configurado los archivos [Gradle](#) [22] para la construcción de la aplicación, se han gestionado las dependencias y los SDKs para la compilación del proyecto. También se han modificado aspectos visuales de la aplicación como su ícono.
- **/assets:** En esta ubicación se hallan todas las imágenes y recursos que necesita el sistema. Contiene más de 30 archivos necesarios para una correcta visualización de la aplicación.
- **/lib:** El paquete **lib** contiene todo el código fuente de la aplicación. Éste está organizado por carpetas según el área al que pertenece. El método **main** de la aplicación también se encuentra dentro de este paquete. Los archivos de este paquete son de extensión .dart. La estructura de las carpetas es la siguiente:
  - **/lib/classes:** Aquí se encuentran todas las entidades del proyecto (event.dart, todo.dart, note.dart y report.dart), además de una carpeta **/stats** que contiene las

entidades pertenecientes a las estadísticas de `tester`. Para una mayor simplicidad en las importaciones del proyecto, se ha creado un archivo `all_classes.dart` que apunta a todas las clases, de forma que importando sólo este archivo se están importando en realidad todas las clases a la vez, su código tiene la siguiente forma:

```

1  export 'package:simplex/classes/event.dart';
2  export 'package:simplex/classes/todo.dart';
3  export 'package:simplex/classes/note.dart';
4  export 'package:simplex/classes/report.dart';

```

- **/lib/common:** Contiene código que es común a varias partes del proyecto, por lo que se decide evitar duplicados, además de que se agiliza el desarrollo. En este directorio se encuentran documentos con funciones comunes, los colores usados para los elementos y el texto, enlaces externos, o variables globales de la aplicación. También contiene una carpeta `/widgets` en la que hay 6 archivos Dart [3] con `widgets` de diferentes tipos y temáticas. Este enfoque para el uso de `widgets` funciona de la siguiente manera, primero se define el `widget`, en este caso un contenedor, en el archivo común, y después lo construimos en el lugar deseado con los parámetros necesarios:

```

1  import 'package:flutter/material.dart';
2  import 'package:simplex/common/all_common.dart';
3  import 'package:flutter_gen/gen_110n/app_localizations.dart';
4
5  Container NoItemsContainer(String items, double
6    heightProportion, BuildContext context){
7    return Container(
8      height: deviceHeight*heightProportion,
9      alignment: Alignment.center,
10     child: Text(AppLocalizations.of(context)!.without + items,
11       textAlign: TextAlign.center,
12       style: TextStyle(
13         color: colorSecondText,
14         fontSize: deviceWidth * fontSize * 0.05,
15         fontWeight: FontWeight.normal)),
16   );
}

```

```

1  if (events.length==0 && todos.length==0 && notes.length==0){
2    NoItemsContainer(AppLocalizations.of(context)!.events, 0.2,
3    context)
}

```

- **/lib/l10n:** Aquí se encuentran los documentos ARB correspondientes a cada idioma para el proceso de internacionalización, también existe un archivo **untranslated-messages.txt** en el que se indicarían las traducciones que no se han realizado o que tienen algún problema, en caso de haberlas.
- **/lib/pages:** Contiene todas las páginas de la aplicación. También existe un documento **all\_pages.dart**. Hay un total de 27 archivos de páginas y están repartidos por las siguientes carpetas:
  - \* **/lib/pages/events**
  - \* **/lib/pages/todos**
  - \* **/lib/pages/notes**
  - \* **/lib/pages/routines**
  - \* **/lib/pages/settings**
  - \* **/lib/pages/stats**
  - \* **/lib/pages/help**
- **/lib/services:** Este paquete contiene los archivos con los métodos y llamadas a los 4 servicios usados por la aplicación, los 2 remotos son **Firebase Database** [6] y **Firebase Authentication** [5] y los otros 2 son el servicio de manejo de notificaciones móviles [32] y el de grabado de variables en el almacenamiento local (**shared preferences** [33]).
- **main.dart:** El archivo principal en donde comienza la ejecución de la aplicación. Dentro del método **main** se ejecuta la operación **runApp()**, en donde se definen, entre otras cosas, las regiones soportadas (es, gl, en, todas sin código de país) y las rutas a todas las páginas disponibles de la aplicación:

```

1   ...
2   supportedLocales: [
3     Locale('es', ''), // Spanish, no country code
4     Locale('gl', ''), // Galician, no country code
5     Locale('en', ''), // English, no country code
6   ],
7   initialRoute: '/auth',
8   routes: {
9     '/auth': (context) => const Auth(),
10    '/home': (context) => const Home(),
11    '/services/change_password_service': (context) => const
12      ChangePasswordService(),
13      ...

```

## 7.2 Modelo de datos

Firestore Database [6] es una base de datos de tipo Non SQL / Not only SQL (NoSQL). Concretamente, es una base de datos orientada a documentos, por lo que su jerarquía consiste en documentos y colecciones. Existen 3 colecciones principales: Reportes, Estadísticas y Usuarios. En la colección de usuarios, cada documento (que representa a cada usuario) contiene a su vez 3 colecciones más, que recogen los eventos, tareas y notas de ese usuario en particular.

Por lo tanto, la estructura de la base de datos del proyecto es la adjuntada en la figura 7.1 (página 36).

COLECCIÓN	DOCUMENTO	COLECCIONES	DOCUMENTO	ATRIBUTOS
Reports	keep 5007692 6492054 9425476 ...			
		ATRIBUTOS		
		id: string active: bool problem: String date: DateTime userEmail: String userId: String		
COLECCIÓN	DOCUMENTO	COLECCIONES	DOCUMENTO	ATRIBUTOS
Stats	events notes todos			
		ATRIBUTOS		
		active: int deleted: int total: int		
COLECCIÓN	DOCUMENTO	COLECCIONES	DOCUMENTO	ATRIBUTOS
Users	AEKhnAdfgk... Bfy41p8cd5s... SkBBaP35u4... ...	Events	4707690 9492055 1125474 ...	id: int name: String description: String date: DateTime time: DateTime color: int routineEvent: bool notificationsList: List[[int:DateTime]] routinesList: List[int]
		Todos	4007696 5592054 8325474 ...	id: int name: String content: String modificationDate: DateTime onCalendar: bool calendarDate: DateTime routineNote: bool routinesList: List[int]
		Notes	6207696 5692053 9725478 ...	id: int name: String description: String done: bool priority: int limited: bool limitDate: DateTime
		ATRIBUTOS		
		userId: String emailVerified: bool tester: bool		

Figura 7.1: Modelo de datos de la aplicación

## 7.3 Iteración 2

### 7.3.1 Análisis

Una vez completada la iteración 1, correspondiente a la formación en las tecnologías necesarias para el proyecto, comienza la primera iteración de desarrollo. El primer paso es crear la base de la aplicación, posteriormente definimos la estructura de páginas principales y finalmente se desarrollan la sección de ajustes, tamaño de texto y ayuda.

El resultado de esta iteración es una aplicación ejecutable pero sin apenas contenido, tan solo una sección de ajustes y otra de ayuda vacía. Se han desarrollado los siguientes casos de uso:

- CU-36 Ver sección de ajustes
- CU-37 Ajustar tamaño de texto
- CU-38 Ver sección de ayuda

### 7.3.2 Diseño e implementación

El desarrollo comienza con la base de la aplicación, es decir, crear una aplicación vacía pero totalmente ejecutable, configurando todos los aspectos necesarios. Para ello se ha creado un proyecto nuevo en [Android Studio](#) [26], y se ha definido el esqueleto de paquetes. También se ha definido un método **main** y la página principal llamada **home** que contendrá las 5 ventanas principales. Por último se han configurado los archivos [Gradle](#) [22] con las dependencias y configuraciones de ejecución necesarias.

Como una de los objetivos principales es proporcionar la mejor interfaz posible al usuario, se han desarrollado todos los [widgets](#) y páginas de la aplicación empleando unos multiplicadores en función de la proporción de pantalla del dispositivo. Estos valores son leídos en el arranque de la aplicación con la función **\_check\_device()** y se guardan como dos variables globales, de esta forma cada vez que se crea un objeto se evita emplear dimensiones fijas y el tamaño de este se adapta al dispositivo.

```

1 void _check_device() {
2     setState(() {
3         var padding = MediaQuery.of(context).padding;
4         verticalDevice =
5             MediaQuery.of(context).orientation ==
Orientation.portrait;
6
7         deviceHeight = max(MediaQuery.of(context).size.height,
8             MediaQuery.of(context).size.width) -
9             padding.top -
10            padding.bottom;

```

```

11     deviceWidth = min(MediaQuery.of(context).size.height,
12                         MediaQuery.of(context).size.width);
13
14     if (deviceHeight != 0 || deviceWidth != 0) {
15         debugPrint('[OK] Device checked.');
16         deviceChecked = true;
17     } else
18         _check_device();
19     });
20 }

```

Un ejemplo de `widget` usando este método para el tamaño es el siguiente:

```
1   SizedBox(height: deviceHeight*0.02,),
```

Además, se ha definido una estructura estándar para todas las páginas de la aplicación. Las páginas que pertenezcan al menú principal contarán con una barra de navegación, mientras que las demás no. En la figura 7.2 (página 38) se puede consultar la estructura general de la interfaz de las páginas de la aplicación.

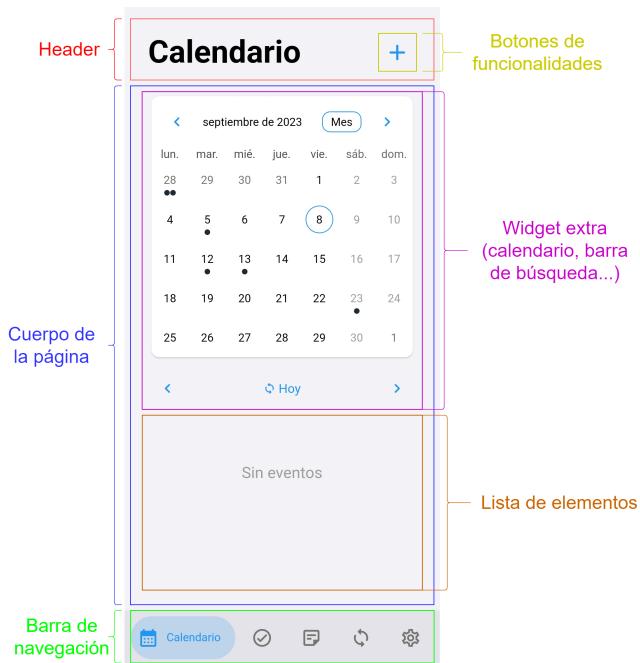


Figura 7.2: Estructura de la interfaz de las páginas

### CU-36 Ver sección de ajustes

Una vez definida la página de inicio, que cuenta con 5 secciones (Calendario, Tareas, Notas, Rutina y Ajustes), comienza el desarrollo de la ventana de ajustes. El primer paso ha sido

definir las variables globales de la aplicación que representan los valores de los ajustes: 4 booleanos para los formatos de fechas y el tema, un double para el tamaño de la fuente y un Locale para el formato del calendario. Para el menú de ajustes se han creado widgets comunes para agilizar el proceso y evitar duplicados. En la ilustración 7.3 (página 39) se muestra dicha sección.

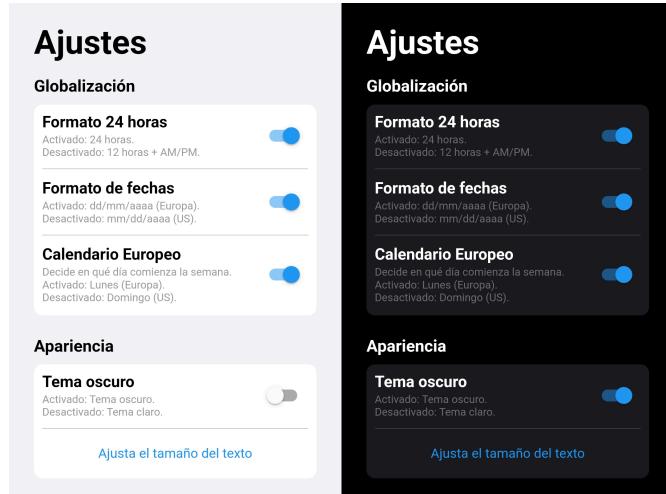


Figura 7.3: Sección de ajustes con los temas claro y oscuro

El último paso para la implementación de estos ajustes es que sean permanentes, es decir, que perdure la elección del usuario una vez cerrada la aplicación. Para ello se ha añadido la dependencia `shared_preferences` [33], que proporciona los métodos para almacenar en el almacenamiento local del dispositivo el valor de estas variables. Esto se hace asignando una clave identificadora a cada valor, y se actualizan con los siguientes métodos:

```

1 saveSettingBool(String key, bool value) async {
2   final prefs = await SharedPreferences.getInstance();
3   prefs.setBool(key, value);
4   debugPrint('[OK] Saved setting $key to $value');
5 }
```

En el arranque de la aplicación se llama a la función `readSettings()`, que se encarga de leer los valores de las claves y actualizar las variables del sistema con dichos valores.

### CU-37 Ajustar tamaño de texto

Este caso de uso ha resultado bastante sencillo. Todas las líneas de texto de la aplicación siguen un paradigma similar al de las dimensiones de los widgets, su tamaño de fuente se basa en 3 factores: un valor constante (es el que el programador define para que sea más grande o más pequeño que el otro texto), el ancho del dispositivo (para guardar una relación con el

tamaño de los [widgets](#)) y una variable global llamada **fontSize**, que funciona como un multiplicador sobre el cual el usuario decide.

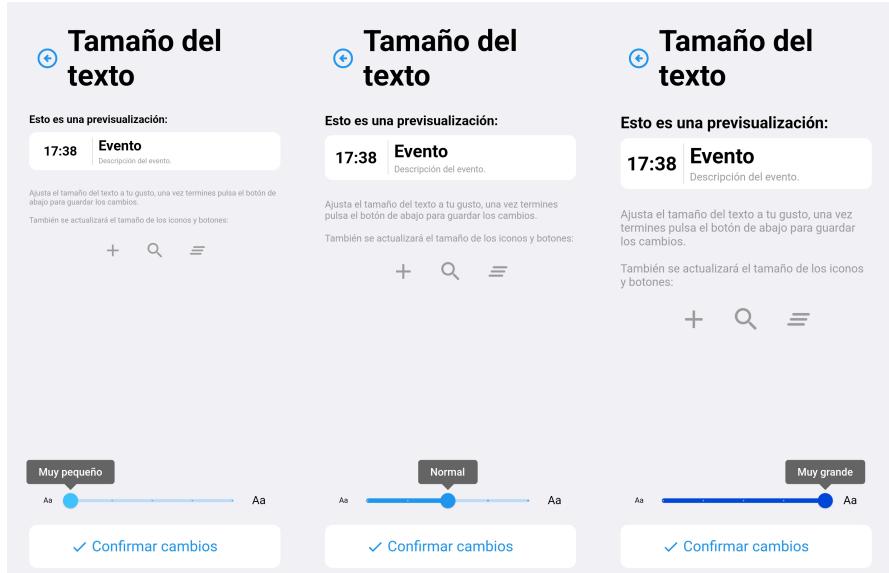


Figura 7.4: Página de ajuste del tamaño del texto

En la ventana para definir el tamaño del texto hay un texto y unos iconos de muestra para que el usuario tenga referencia del tamaño que tendrá una vez confirmados los cambios.

El usuario tan solo tiene que deslizar una barra horizontal a izquierda o a derecha para disminuir o agrandar el texto respectivamente. Se han establecido 5 valores para el multiplicador, que van desde 0.8 para *Muy pequeño* hasta 1.2 para *Muy grande*. Se muestra esta sección en la figura 7.4 (página 40).

### CU-38 Ver sección de ayuda

La sección de ayuda consiste en un menú con 5 botones que dan acceso a una página diferente de información acerca de un área de la aplicación. En una de estas páginas se ponen ejemplos y se explican todas las herramientas del sistema para que el usuario pueda sacarle el máximo partido posible.

A mayores la sección de ayuda proporciona acceso al apartado de reporte de problemas (que será tratado en otro caso de uso) y también a la política de privacidad [34] de la aplicación, que está pública y disponible para todo el mundo en la web. Pueden verse algunas partes de esta sección en la ilustración 7.5 (página 41).



Figura 7.5: Página de ayuda y alguna de sus secciones

## 7.4 Iteración 3

### 7.4.1 Análisis

En esta iteración se ha implementado toda la gestión de los usuarios, desde la autenticación hasta las funciones auxiliares como la recuperación de contraseña. Los casos de uso a completar en esta iteración son:

- CU-01 Registro de usuarios
- CU-02 Autenticación de usuarios
- CU-03 Cierre de sesión
- CU-04 Cambio de contraseña
- CU-05 Recuperación de contraseña

### 7.4.2 Diseño e implementación

Para comenzar esta parte primero ha sido necesario crear un proyecto en [Google Firebase](#) [23], y enlazarlo con el proyecto de [Android Studio](#) [26], esto se ha hecho en varios pasos, como descargar el fichero **googleservices.json** y añadir las dependencias a los archivos **android/build.gradle** y **android/app/build.gradle**:

```

1 repositories {
2     google()
3     ...
4 }
5
6 dependencies {
7     ...
8     classpath 'com.google.gms:google-services:4.3.15'
9 }
10
11 apply plugin: 'com.google.gms.google-services'
```

Una vez enlazados los dos proyectos, se comienza con el fichero **auth.dart**, en el que se realiza una comprobación de si el usuario está autenticado o no y se redirige a la página que corresponda en cada caso:

```

1 if (snapshot.connectionState == ConnectionState.waiting){
2     return Center(child: CircularProgressIndicator(color:
3         colorSpecialItem,));
4 } else if (snapshot.hasError){
5     if (loginIndex == 0) return LogInService();
6     else if (loginIndex == 1) return SignUpService();
7     else return RecoverPasswordService();
8 } else if(snapshot.hasData){
9     return VerifyEmailService();
10 }else{
11     if (loginIndex == 0) return LogInService();
12     else if (loginIndex == 1) return SignUpService();
13     else return RecoverPasswordService();
14 }
```

En la base de la aplicación en **main.dart** se ha definido como página por defecto la del fichero **auth.dart**, por lo que siempre se realiza esta comprobación al iniciar la aplicación, de forma que no es posible entrar sin autenticarse. En el servicio **verifyEmailService** se redirecciona automáticamente a la página **Home** en caso de que el email del usuario ya esté verificado. El flujo de la ejecución se puede observar en la figura 7.6 (página 43).

### CU-01 Registro de usuarios

Si el usuario no dispone de cuenta de Simplex [7], deberá crearse una. El registro se realiza mediante correo electrónico y contraseña.

El formulario de registro funciona del mismo modo que todos los del sistema: siempre se comprueba que los campos cumplan el formato indicado en local, de forma que la petición no se enviará a los servidores remotos hasta que todas las entradas de texto respeten sus

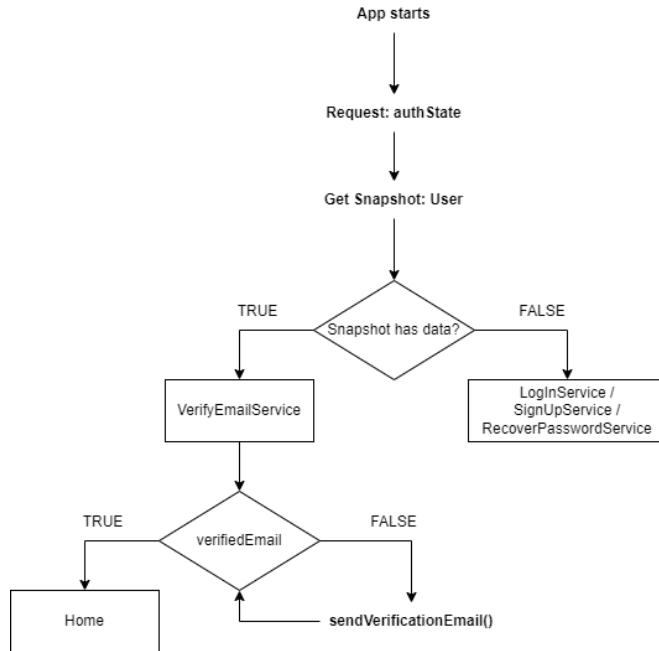


Figura 7.6: Flujo de la ejecución en función del estado de autenticación del usuario

formatos. Una vez se cumple esta premisa, se llama a la función **signUp()**, que comprueba que el email proporcionado no suponga ningún error o que no esté ya registrado:

```

1 Future signUp() async {
2     showDialog(
3         context: context,
4         barrierDismissible: false,
5         builder: (context) => Center(child:
6             CircularProgressIndicator(),
7         );
8
8     try{
9         await FirebaseAuth.instance.createUserWithEmailAndPassword(
10             email: emailController.text.trim(),
11             password: passwordController.text.trim());
12     } on FirebaseAuthException catch (e){
13         // TRATAMIENTO DE EXCEPCIONES
14     }
15     createUserDoc();
16     navigatorKey.currentState!.pop();
17 }
```

Cuando el formulario se procesa, se llama a la función **createUserDoc()**, que se encarga de generar una entrada en la base de datos para el usuario con las colecciones vacías, tan solo

con un documento vacío llamado *keep* para que ésta no se borre.

Tanto en este caso como en todos los demás de la aplicación, siempre que ocurra un error se le indicará al usuario por pantalla mediante un **widget Snackbar** como el de la figura 7.7 (página 44).

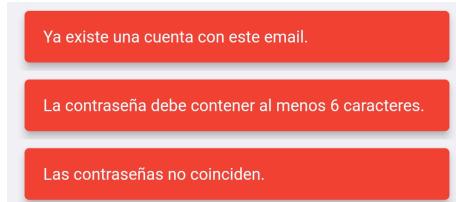


Figura 7.7: Widget Snackbar que indica errores al usuario

Una vez cubierto y verificado el formulario de registro, el servicio de **Firebase Authentication** [5] enviará un email de verificación a la dirección indicada por el usuario. En cuanto el usuario verifique su cuenta accediendo al enlace del email, la aplicación entrará a la ventana principal, el calendario. Esto es posible ya que la aplicación, tras enviar el email, se queda en un bucle comprobando el estado de la verificación cada 3 segundos. De esta forma se evita la creación de cuentas de forma masiva o malintencionada.

```

1  if (!verifiedEmail) {
2      sendVerificationEmail();
3      timer = Timer.periodic(
4          Duration(seconds: 3),
5          (_) => checkVerifiedEmail(),
6      );
7  }

```

## CU-02 Autenticación de usuarios

Si el usuario dispone de una cuenta de Simplex [7], debe autenticarse para acceder a la aplicación. Esto se hace rellenando el formulario de inicio de sesión, que solicita el correo electrónico y la contraseña con los que se ha realizado el registro.

La ventana de *Iniciar de sesión* tiene accesos directos a las ventanas de *Registrarse* y *Restablecer contraseña*. Del mismo modo que el formulario de registro, primero se comprueba el formato de las entradas en local y, si todo está en orden, se envía la petición al servidor.

Si las credenciales introducidas son correctas, se mostrará una introducción sobre la aplicación y posteriormente se redirigirá a la página principal, el calendario. En la figura 7.8 (página 45) se muestra el formulario de autenticación.



Figura 7.8: Ventana de Iniciar sesión

### CU-03 Cierre de sesión

Se ha colocado un botón al fondo de la página de *Ajustes*, de esta forma al pulsarlo se mostrará un diálogo para confirmar que el usuario quiere cerrar la sesión. Una vez pulsado, se ejecuta el método `FirebaseAuth.instance.signOut()`.

### CU-04 Cambio de contraseña

Es posible que en algún momento el usuario quiera cambiar la contraseña de su cuenta, por lo que se ha añadido un apartado en la sección *Cuenta* de la ventana *Ajustes* para este propósito. En la página se solicitan al usuario su contraseña actual, su nueva contraseña y que repita la nueva contraseña.

Si el formulario cumple el formato necesario y las credenciales son correctas, se indicará al usuario mediante un mensaje en pantalla que su contraseña ha sido actualizada correctamente, como se ve en la figura 7.9 (página 45). No es necesario que se cierre la sesión actual del usuario para que se cambie la contraseña.

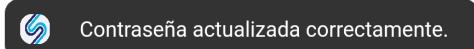


Figura 7.9: Mensaje de confirmación tras cambiar la contraseña

### CU-05 Recuperación de contraseña

Como el usuario puede llegar a olvidarse de su contraseña, se ha implementado una ventana para poder restablecerla. Todo lo que debe hacer el usuario es indicar el correo electrónico de la cuenta de Simplex [7] y el servidor de Firebase Authentication [5] se encargará de enviarle un email con un enlace de recuperación.

Esta solicitud se realiza con la función **resetPassword**, que posteriormente indicará al usuario si se ha enviado el email o, por el contrario si existe algún error con la dirección de correo proporcionada.

En la figura 7.10 (página 46) se muestra esta sección en la aplicación y en la figura 7.11 (página 46) se muestra el email de recuperación.



Figura 7.10: Envío del email para restablecer la contraseña

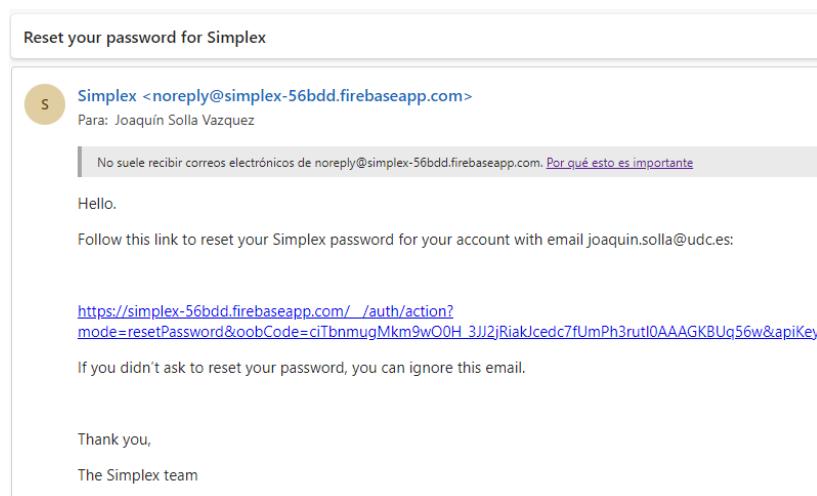


Figura 7.11: Email para restablecer la contraseña

## 7.5 Iteración 4

### 7.5.1 Análisis

Esta iteración supone el comienzo del desarrollo de los elementos con los que va a trabajar el usuario, en este caso hablamos de los eventos y el calendario. Antes de comenzar con el código fuente, es necesario crear las bases de datos necesarias, definir sus estructuras de documentos y enlazarlas con el proyecto. Los casos de uso a cubrir en esta iteración son:

- CU-06 Ver calendario y eventos
- CU-07 Ver detalles de eventos
- CU-08 Crear eventos
- CU-09 Editar eventos
- CU-10 Eliminar eventos
- CU-11 Recibir notificaciones de eventos
- CU-12 Compartir eventos

### 7.5.2 Diseño e implementación

La implementación comienza con la creación de las colecciones de datos en [Firestore Database](#) [6]. También se tuvo que establecer una serie de reglas de acceso para la base de datos, implementadas en el lenguaje específico para esto, [FSR](#):

```

1 rules_version = '2';
2 service cloud.firestore {
3     match /databases/{database}/documents {
4
5         match /users/{userId} {
6             allow create, update, delete: if
7                 request.auth.uid == userId;
8             allow read: if
9                 true == true;
10            match /events/{eventId} {
11                allow create, read, update, delete: if
12                    request.auth.uid == userId;
13                }
14                ...
15            }
16            ...
17        }
}

```

Después tan solo quedaría enlazar la base de datos con el proyecto a través de los archivos [Gradle](#) [22], al igual que en las anteriores veces.

### CU-06 Ver calendario y eventos

Esta iteración tiene 3 partes, primero comenzamos con el calendario, que se ha implementado a partir de la dependencia [table\\_calendar](#) [35], de forma que se ha implementado un widget **TableCalendar**. En la ilustración 7.12 (página 48) se muestra este calendario.

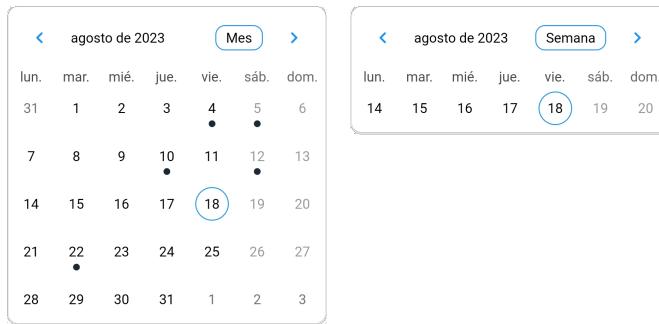


Figura 7.12: Calendario de Simplex

La segunda parte supone la implementación de los eventos, de forma que se define una clase en [Dart](#) [3] que representa a cada evento, cada evento contará con los atributos definidos en dicha clase. También se definen los métodos **toJson()** y **fromJson()** para intercambiar estos objetos con la base de datos. La clase **Event** tiene la misma estructura que tendrán las demás:

```

1 import 'package:cloud_firestore/cloud_firestore.dart';
2
3 class Event{
4   final int id;
5   final String name;
6   final String description;
7   final DateTime date;
8   final DateTime time;
9   final int color;
10  final List<dynamic> notificationsList;
11  final List<dynamic> routinesList;
12  final bool routineEvent;
13
14  Event({
15    required this.id,
16    required this.name,
17    required this.description,
18    required this.date,
19    required this.time,
```

```

20     required this.color,
21     required this.notificationsList,
22     required this.routinesList,
23     required this.routineEvent,
24   );
25
26   Map<String, dynamic> toJson() => {
27     'id': id,
28     'name': name,
29     'description': description,
30     'date': date,
31     'time': time,
32     'color': color,
33     'notificationsList': notificationsList,
34     'routinesList': routinesList,
35     'routineEvent': routineEvent,
36   };
37
38   static Event fromJson(Map<String, dynamic> json) => Event(
39     id: json['id'],
40     name: json['name'],
41     description: json['description'],
42     date: (json['date'] as Timestamp).toDate(),
43     time: (json['time'] as Timestamp).toDate(),
44     color: json['color'],
45     notificationsList: json['notificationsList'],
46     routinesList: json['routinesList'],
47     routineEvent: json['routineEvent'],
48   );
49 }

```

El tercer paso es el más elaborado, ya que consiste en que la aplicación solicite a la base de datos los eventos de ese usuario específico y, una vez recogidos, enviarlos al calendario para que aparezcan en sus respectivos días. Además, si el usuario selecciona un día en el calendario deberán aparecer debajo de este los eventos ordenados por hora. El calendario también debe actualizarse en el momento en el que la base de datos notifique un cambio en los eventos.

La base de este procedimiento es la siguiente:

1. Se define un [widget StreamBuilder](#), que se encarga de solicitar la lista completa de eventos a la base de datos.
2. Una vez el [widget](#) recibe la lista, comprueba si ocurrió algún error, de ser así se indicará al usuario, si la lista está vacía o contiene eventos, se llama a la función **loadEventsToCalendar(...)**, que se encarga de crear una lista que a su vez contiene listas que representan cada día con eventos, esto será lo que lea el calendario.

3. Se genera el calendario, que marcará los días que tengan elementos con un punto ya que lee la lista de días con elementos.
4. Otro **StreamBuilder** recibe también la lista de días con elementos y se encarga de mostrar bajo el calendario los eventos del día seleccionado en el calendario.

### CU-07 Ver detalles de eventos

Cuando un usuario selecciona un evento, se le redirige a la vista de sus detalles. Lo primero que hace el sistema al seleccionar un evento es establecerlo en la variable global **selectedEvent**, de esta forma al cargar la página se definen los valores a mostrar a partir de los parámetros del evento.

Todos los parámetros obedecen a los formatos y preferencias definidos en la sección *Ajustes* por el usuario.

También se muestra un botón que redirige a la ventana de edición del evento.

### CU-08 Crear eventos

La creación de un evento supone llenar un formulario indicando los campos requeridos, algunos obligatorios y otrosopcionales.

Además de entradas de texto, también se recogen otros campos:

- **Fecha y hora:** Se definen dos objetos **DateTime** separados para representar cada cosa, se recogen mediante unos diálogos de calendario (**DatePicker**) y reloj (**TimePicker**).
- **Color:** El color se guarda localmente como un objeto **Color**, pero como en *Firestore Database* [6] no existe este objeto, debe realizarse la conversión de su código hexadecimal a un objeto **String**.
- **Notificaciones:** Se genera una lista de mapas, cada mapa contiene una referencia, que es el identificador de la notificación (generado a partir de la fecha en milisegundos de creación), y también contiene la fecha y hora de notificación.

Una vez se confirma el formulario, el sistema realiza las conversiones necesarias de los parámetros y convierte el objeto **Event** nuevo a **JSON** [28] para a continuación enviárselo a la base de datos. El nuevo evento aparecerá automáticamente en el calendario en su día.

### CU-09 Editar eventos

La edición de eventos recoge dos aspectos de los dos casos de uso anteriores. Por una parte se guarda el elemento a editar en la variable **selectedEvent**, y por la otra se genera un formulario igual al de la creación del evento, pero ya con los datos del evento seleccionado. El

usuario tan solo tiene que actualizar los campos que deseé y confirmar los cambios, después el sistema envía la petición de actualización al servidor.

### CU-10 Eliminar eventos

Cuando un usuario selecciona la eliminación de un evento, se lanzan las órdenes **cancelAllEventNotifications(eventId)** para eliminar las notificaciones pendientes que pudiese tener el evento y **deleteEventById(eventId)** para eliminar tanto local como remotamente el evento.

```

1  deleteEventById(int eventId) async {
2      final doc = FirebaseFirestore.instance.collection('users')
3          .doc(FirebaseAuth.instance.currentUser!.uid)
4          .collection('events').doc(eventId.toString());
5
6      await doc.delete();
7      debugPrint('[OK] Event deleted');
8      await subtractElement("events");
9 }
```

### CU-11 Recibir notificaciones de eventos

Para la recepción de notificaciones móviles acerca de los elementos de la aplicación, se ha añadido la dependencia **flutter\_local\_notifications** [32] y se ha definido un servicio **notification\_service.dart** en donde se definen los métodos para programar notificaciones. Cada vez que se crea un evento con notificaciones, estas se envían al servicio para su programación. Si un evento es editado, se eliminan todas las notificaciones que tenía programadas anteriormente y se programan las nuevas.

### CU-12 Compartir eventos

Un evento puede ser compartido en prácticamente cualquier aplicación como un mensaje de texto. Cuando se selecciona la opción de compartir, se construye un objeto String a partir de los parámetros del evento, desués el paquete **social\_share** [36] recibe la cadena de texto y permite compartirlo en las aplicaciones:

```

1 Evento: Evento de prueba
2 Esto es la descripción del evento
3
4 El 23/08/2023
5 A las 18:30
6
7 Creado con ©Simplex.
```

## 7.6 Iteración 5

### 7.6.1 Análisis

Esta iteración recoge todas las funcionalidades relacionadas con las tareas. Gran parte del funcionamiento de esta parte tiene la misma base y fundamentos que la iteración 4 con los eventos, por lo que no se añadirán detalles repetitivos. Los casos de uso completados en esta iteración son:

- CU-13 Ver lista de tareas
- CU-14 Ver detalles de tareas
- CU-15 Crear tareas
- CU-16 Editar tareas
- CU-17 Eliminar tareas
- CU-18 Recibir notificaciones de tareas
- CU-19 Compartir tareas
- CU-20 Buscar tareas por texto
- CU-21 Añadir tareas al calendario

### 7.6.2 Diseño e implementación

Esta iteración comienza con la definición de la entidad **Todo**, por lo que se ha creado su fichero con su método constructor y sus métodos de conversión **JSON**.

#### CU-13 Ver lista de tareas

La lista de tareas se encuentra en el documento **todos\_main.dart**, y consiste, de la misma forma que el calendario y la lista de eventos, en un **widget StreamBuilder**. Este **widget** lanza 3 veces un mismo método (pero con diferentes parámetros de cada vez) que solicita a la base de datos la lista de tareas con una prioridad concreta, el método también admite la inclusión de una línea de texto para filtrar las tareas por su título:

```
1 Stream<List<Todo>> readPendingTodosByPriorityAndName(int priority,  
    String content) {  
2     if (content == "") return  
3         FirebaseFirestore.instance.collection('users')  
            .doc(FirebaseAuth.instance.currentUser!.uid)
```

```

4   .collection('todos')
5   .where('done', isEqualTo: false)
6   .where('priority', isEqualTo: priority)
7   .orderBy('limited', descending: true)
8   .orderBy('limitDate', descending: false)
9   .orderBy('name', descending: false)
10  .snapshots().map((snapshot) =>
11    snapshot.docs.map((doc) => Todo.fromJson(doc.data())).toList());
12
13 else // CONSULTA CON PALABRAS CLAVE
14 }
```

Cuando el **StreamBuilder** recibe las 3 respuestas, fusiona los resultados en una lista de listas. De no haber ninguna tarea, se indicará por pantalla. De haber tareas, estas se clasifican primeramente en dos contenedores, uno de tareas pendientes y otro de tareas completadas. Dentro de las tareas pendientes también se hará una división por niveles de prioridad (alta, media y baja).

```

1 StreamBuilder<List<List<Todo>>>(
2   stream: CombineLatestStream.list([
3     readDoneTodosByName(keywords.trim()),
4     readPendingTodosByPriorityAndName(1, keywords.trim()),
5     readPendingTodosByPriorityAndName(2, keywords.trim()),
6     readPendingTodosByPriorityAndName(3, keywords.trim()),
7   ]),
8   builder: (context, snapshot) {
9     if (snapshot.hasError) // CONTENEDOR DE ERROR
10    else if (snapshot.hasData) // LISTA DE TAREAS
```

También existe un contenedor con una barra de búsqueda, de forma que el usuario puede activarla y filtrar las tareas por su título. Cuando el usuario introduce un nuevo carácter en la barra de búsqueda, el sistema vuelve a ejecutar la consulta a la base de datos (en caso de no haberse modificado la base de datos remota se ejecuta la consulta en el contenido cacheado por la aplicación) y se actualiza la lista de tareas automáticamente. La búsqueda de elementos por texto funciona de la misma forma para la parte de notas.

Puede verse la vista de tareas en la figura 7.13 (página 54).

#### CU-14 Ver detalles de tareas

Este caso de uso funciona de la misma forma que para los eventos, solamente se adapta a los parámetros específicos de las tareas. Se emplea la variable **selectedTodo** para guardar la tarea seleccionada. También se puede acceder a la vista de edición desde la página de detalles.

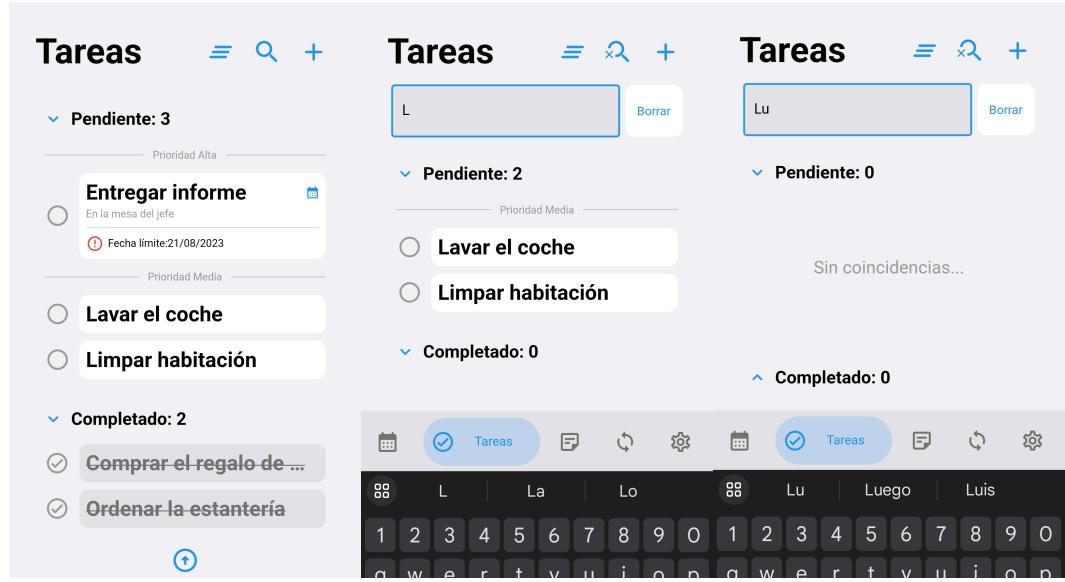


Figura 7.13: Vista principal de las tareas

### CU-15 Crear tareas

Se emplea un formulario de la misma forma que para los eventos, pero se añade un campo diferente: la prioridad de la tarea. Este parámetro se puede ajustar mediante una barra horizontal que contiene los 3 niveles de prioridad disponibles.

También está el campo de *Tarea con fecha límite*, que puede activarse de forma que el usuario seleccionará una fecha con un diálogo con un calendario.

### CU-16 Editar tareas

La edición de tareas funciona de la misma forma que su creación, pero la página se abre con el formulario con los campos cubiertos con sus datos actuales. Una vez confirmados por el usuario y comprobado que el formulario no contiene errores, la secuencia de acciones consiste en: crear un nuevo **Todo** con los datos del formulario, actualizar la antigua tarea con los datos de la nueva, borrar sus notificaciones antiguas y programar sus nuevas notificaciones.

### CU-17 Eliminar tareas

La eliminación de tareas funciona de la misma forma que para todas las entidades, ya mencionada anteriormente. Las tareas tienen una funcionalidad adicional que es la eliminación de todas las tareas marcadas como completadas. Esto se realiza a través de la función **deleteDoneTodos()**, que envía la petición de borrado a todos los documentos de tarea con el parámetro **done** como verdadero.

### CU-18 Recibir notificaciones de tareas

La programación de notificaciones de tareas funciona de forma automática, el usuario no puede decidir cuándo recibirlas. Sólo las tareas con fecha límite pueden tener notificaciones, y estas serán el día límite que tienen marcado y el día siguiente, es decir, cuando la tarea ya ha expirado.

### CU-19 Compartir tareas

Las tareas se comparten de la misma forma que todas las entidades, con un resumen de texto con sus campos más relevantes.

### CU-20 Buscar tareas por texto

El filtrado de tareas por texto se consigue pasando como parámetro a la petición a la base de datos la variable **keywords**, que es la que guarda el texto introducido en la entrada de texto. La búsqueda por texto se puede activar y desactivar con un botón, que cambia el valor de la variable de tipo booleano **showSearchbar** y actualiza la vista de forma instantánea. Por defecto la búsqueda por texto está desactivada, para lograr una interfaz menos cargada.

### CU-21 Añadir tareas al calendario

La integración con el calendario se logra incluyendo las entidades **Todo** en el tipado de la función que recoge los eventos para el calendario, como son varias las entidades a recoger, se define como **dynamic** para que las admita todas. En la visualización de elementos del día, bajo el calendario, se organizan varios contenedores: uno para eventos, otro para tareas y un último para notas. El orden con el que aparecen los elementos para cada día es el siguiente: Tareas, Notas y Eventos (estos últimos ordenados cronológicamente). Sólo aparecen en el calendario las tareas con una fecha límite definida.

## 7.7 Iteración 6

### 7.7.1 Análisis

En esta iteración se desarrollan las funcionalidades en torno a la tercera entidad del sistema, las notas. Las explicaciones serán breves dado que el funcionamiento de éstas es muy similar a la de las otras dos entidades. Los casos de uso completados en la iteración son:

- CU-22 Ver lista de notas
- CU-23 Ver detalles de notas

- CU-24 Crear notas
- CU-25 Editar notas
- CU-26 Eliminar notas
- CU-27 Recibir notificaciones de notas
- CU-28 Compartir notas
- CU-29 Buscar notas por texto
- CU-30 Añadir notas al calendario

### 7.7.2 Diseño e implementación

El primer paso una vez más ha sido crear la entidad **Note**, con sus campos correspondientes, entre los que destaca **content**, que contiene todo el texto de la nota y es su parte esencial.

#### CU-22 Ver lista de notas

La vista de notas es más simple que la de tareas, ya que sólo se ordenan por su fecha de edición (de más reciente a menos). La distribución de las notas por pantalla se lleva a cabo con el widget **MasonryGridView.count** [37], de esta forma se organizan según su tamaño sin dejar huecos libres, logrando así una interfaz un poco más moderna de lo que será una simple cuadrícula con campos del mismo tamaño tal como se ve en la figura 7.14 (página 56).

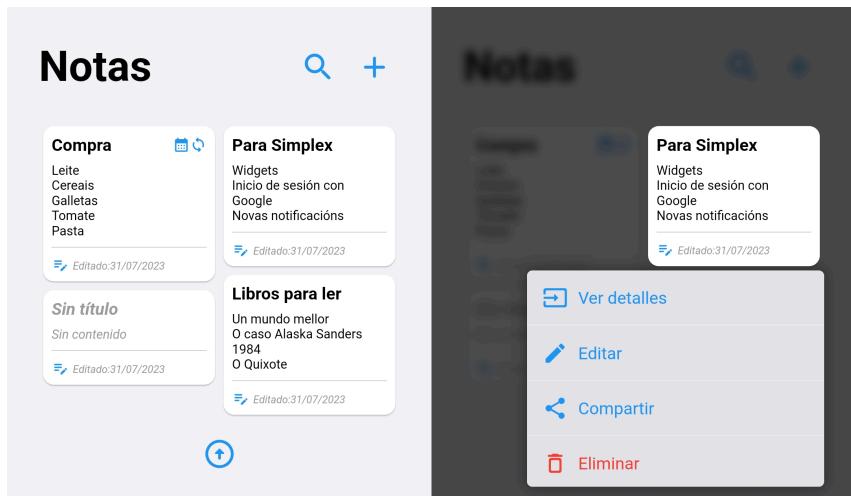


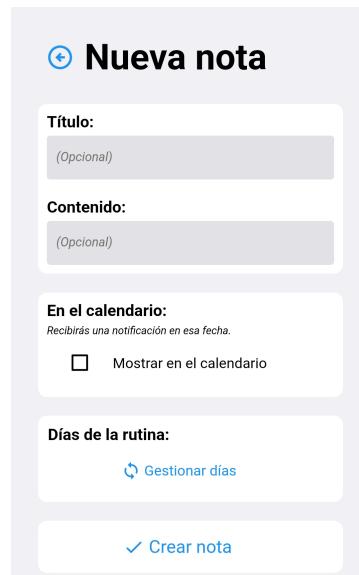
Figura 7.14: Vista principal de las notas

### CU-23 Ver detalles de notas

Los detalles de las notas funcionan de la misma forma que los de todas las entidades. Para esta entidad se utiliza la variable **selectedNote**.

### CU-24 Crear notas

La creación de notas también es un formulario, en él se encuentra la entrada para el contenido de la nota, que puede aumentar de tamaño en función de la cantidad de texto introducida para un manejo y lectura más cómodos. También se puede marcar la opción *Mostrar en el calendario* si queremos que aparezca en una fecha específica. En la figura 7.15 (página 57) se muestra el formulario de creación para las notas.



El formulario se titula "Nueva nota". Contiene los siguientes campos:

- Título:** Un campo de texto con placeholder "(Opcional)".
- Contenido:** Un campo de texto con placeholder "(Opcional)".
- En el calendario:** Una sección que indica: "Recibirás una notificación en esa fecha." Con un checkbox para "Mostrar en el calendario".
- Días de la rutina:** Un botón "Gestionar días" que abre una modal.
- Botones de acción:** "Crear nota" (en azul) y "Cancelar" (en gris).

Figura 7.15: Formulario de creación de notas

### CU-25 Editar notas

Las notas se editan con un formulario idéntico al empleado en su creación. El procedimiento para actualizar la nota es también similar al empleado para actualizar eventos y tareas.

### CU-26 Eliminar notas

Se pueden eliminar notas desde los mismos puntos que las demás entidades, pero a diferencia de las tareas, no existe un método para eliminarlas todas de una sola vez.

**CU-27 Recibir notificaciones de notas**

Las tareas con una fecha en el calendario tendrán programada una notificación para ese día. El usuario no puede definir notificaciones en fechas personalizadas para las tareas.

**CU-28 Compartir notas**

Las notas se comparten, al igual que todas las entidades, como texto. De forma que se compartirá todo el contenido de la nota, además de su título.

**CU-29 Buscar notas por texto**

También se ha implementado la búsqueda por texto para las notas, al igual que para las tareas. El filtrado por texto sólo aplica a los títulos de las notas, por lo que no se puede buscar en función de su contenido.

**CU-30 Añadir notas al calendario**

Las notas con una fecha establecida aparecerán en el calendario, junto con los eventos y las tareas. Por lo tanto se podrá acceder al contenido de una nota a través del calendario. En la figura 7.16 (página 58) se puede ver cómo eventos, tareas y notas comparten el mismo espacio.



Figura 7.16: Eventos, tareas y notas en el calendario

## 7.8 Iteración 7

### 7.8.1 Análisis

A continuación toca implementar las rutinas, y también todas sus funcionalidades. A una rutina pueden pertenecer los eventos y las notas, pero no las tareas. Cada uno de los 7 días de la semana tendrá su rutina con sus elementos recurrentes. Por lo tanto en esta iteración se llevarán a cabo los siguientes casos de uso:

- CU-31 Ver lista de rutinas
- CU-32 Crear rutinas
- CU-33 Editar rutinas
- CU-34 Eliminar rutinas
- CU-35 Compartir rutinas

### 7.8.2 Diseño e implementación

Esta iteración se divide en dos partes. La primera es el desarrollo de la página de rutinas, y la otra la expansión de los eventos y notas a las rutinas. La base de este trabajo consiste en añadir varios atributos a las entidades, además de modificar sus formularios de creación y edición.

#### CU-31 Ver lista de rutinas

La página principal de las rutinas se divide en 3 elementos principales:

1. **Selector de día de la semana:** El primer contenedor consiste en una fila horizontal con los días de la semana, de forma que el usuario puede cambiar de rutina rápidamente. Cada vez que el usuario cambia de día de la semana, la vista se actualiza ya que se cambia el índice de la variable **routineDay**.
2. **Lista de notas de la rutina:** Un **StreamBuilder** se encarga de solicitar las notas con el día de rutina indicado y las muestra en una fila con desplazamiento en horizontal. En caso de que la rutina no contenga notas este contenedor no se mostrará al usuario.
3. **Lista de eventos de la rutina:** Otro **StreamBuilder** solicita a la base de datos los eventos de rutina que tengan ese día de la semana marcado, y las ordena cronológicamente por sus horas. Este contenedor también contiene un botón para compartir toda la rutina de cada día en particular.

En la figura 7.17 (página 60) se muestra la vista de rutinas.

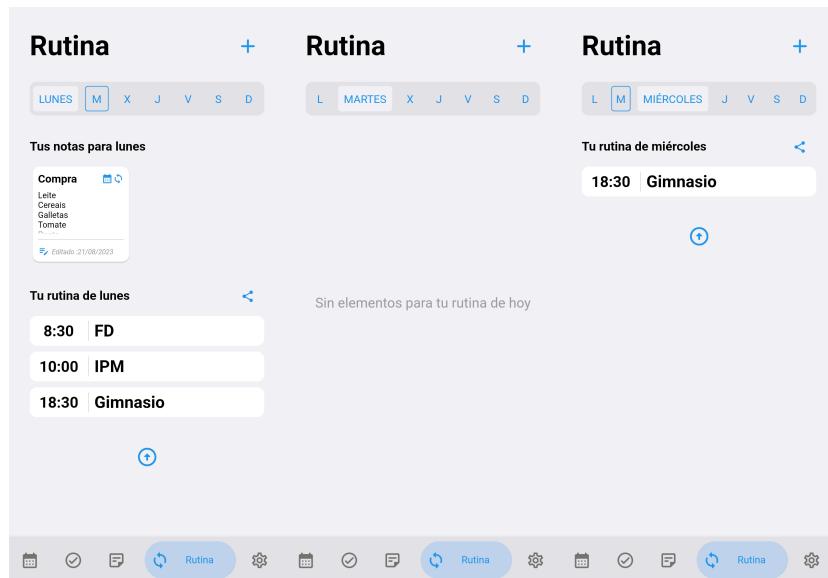


Figura 7.17: Lista de rutinas

### CU-32 Crear rutinas

La creación de rutinas consiste en marcar los eventos o notas como rutinas. Esto puede hacerse de 3 formas:

1. **Crear un evento de rutina:** En la página de creación de eventos existe un botón para indicar si el evento a crear es de *una vez* o de *rutina*, tal y como se ve en la ilustración 7.18 (página 60). En función del tipo de evento se mostrarán y ocultarán algunos campos del formulario y se llevarán a cabo acciones diferentes al confirmar la creación.

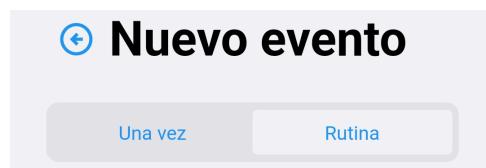


Figura 7.18: Selector de tipo de evento

La actualización del formulario se lleva a cabo con el control de una variable de tipo **booleano** y condicionales:

```

1  if(routineEvent) // CAMPO PARA EVENTO DE RUTINA
2  if(!routineEvent) // CAMPO PARA EVENTO DE UNA VEZ

```

2. **Crear una nota con días de rutina:** Desde la página de creación de notas se pueden marcar días de rutina en los que aparecerá la nota (no existe una diferenciación de

tipo de nota como en los eventos). El usuario seleccionará en un diálogo los días de la semana en los que la nota estará disponible, este diálogo se muestra en la ilustración 7.19 (página 61).

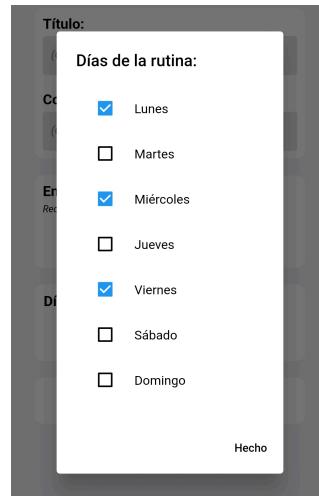


Figura 7.19: Selector de días de rutina de la nota

3. **Crear un elemento desde las rutinas:** También se puede crear un elemento directamente desde la página de rutinas. El usuario tendrá un botón para seleccionar si quiere crear un evento o una nota, y el formulario se actualizará automáticamente.

### CU-33 Editar rutinas

La edición de un elemento de rutina se realiza del mismo modo que en los anteriores casos. En el caso de los eventos, el usuario tendrá disponible la opción de convertir un evento de *rutina* en un evento de *una vez* y viceversa.

### CU-34 Eliminar rutinas

Las rutinas se eliminan de la misma forma que las demás entidades, ya que no es necesario cambiar nada respecto a los métodos de borrado anteriores.

### CU-35 Compartir rutinas

La compartición de rutinas es más elaborada que la de las entidades individuales. Se procesa toda la rutina en un bucle de forma que a medida que se leen sus eventos se va montando el texto del mensaje a compartir:

```
1 | String dayText = dayToString(day, context);
```

```

2 String text = AppLocalizations.of(context)!.routine + ":";
3   $dayText\n";
4
5 if (selectedRoutineEvents.length > 0){
6   for(int i=0; i<selectedRoutineEvents.length; i++){
7     text += ("\n" + timeToString(selectedRoutineEvents[i].time) +
8       " - "
9       + selectedRoutineEvents[i].name);
10  }
11 } else text += AppLocalizations.of(context)!.noEvents + '.';

```

```

1 Rutina: Lunes
2
3 08:30 - FD
4 10:00 - IPM
5 18:30 - Gimnasio
6
7 Creado con ©Simplex.

```

## 7.9 Iteración 8

### 7.9.1 Análisis

La última iteración de desarrollo, que corresponde a los detalles finales (pero no por ello menos importantes) de la aplicación. Además de desarrollar los 3 casos de uso que se mencionarán, también se ha preparado la aplicación para su publicación en Google Play Store [11], se han diseñado todos los gráficos y miniaturas necesarios y se han cubierto los formularios y configuraciones para solicitar la publicación. Los casos de uso que se completarán en esta iteración son:

- CU-39 Reportar un problema
- CU-40 Ver estadísticas de tester
- CU-41 Internacionalización

### 7.9.2 Diseño e implementación

Antes de detallar el desarrollo de los casos de uso se describirán de forma simple los pasos para que la aplicación haya sido publicada en Google Play Store [11]:

1. **Política de privacidad:** Se ha generado una política de privacidad [34] adaptada a la aplicación, ya que es obligatorio para publicar cualquier tipo de aplicación en la tienda

de Google. La política de privacidad [34] está disponible desde la aplicación [7] y en la web: [https://joaquinsolla.com/privacy\\_policies/simplex.html](https://joaquinsolla.com/privacy_policies/simplex.html).

2. **Alta como desarrollador:** Me he dado de alta como desarrollador de Google, esto supone hacer un pago único en la plataforma [29].
3. **Configuración Gradle [22], Application Armor (AppArmor) y claves de desarrollador:** Se ha configurado el archivo `android/app/build.gradle` con una versión de lanzamiento, en donde se incluyen los identificadores de versión (se han utilizado el código de versión 5 y el nombre de versión 1.0.0). También se han obtenido unas claves de desarrollador y se han enlazado con el proyecto para verificar el origen de los ejecutables generados.
4. **AppBundle:** Se ha generado un archivo **AppBundle (AAB)** con el contenido de la aplicación.
5. **Configuración de Google Play Store:** En el panel de control para desarrolladores de Google Play Store [11] se ha creado el proyecto Simplex y se han llevado a cabo todas las configuraciones necesarias como añadir los gráficos de la ficha de Google Play Store [11]. También se han rellenado y firmado formularios conforme se recogen y tratan los datos de los usuarios de forma responsable y cumpliendo las leyes vigentes.
6. **Nuevo lanzamiento:** Se ha creado un nuevo lanzamiento con el archivo **AppBundle** generado anteriormente y con el título *Simplex: Organización personal*. Tras una semana de revisión, la aplicación se ha publicado [7] sin ninguna observación o problema.

### CU-39 Reportar un problema

Desde el apartado de ayuda será posible reportar problemas que el usuario pueda experimentar con la aplicación. La página del reporte de problemas consiste en un formulario en el que se recogen los posibles problemas. Se brinda al usuario la decisión de adjuntar automáticamente información adicional acerca de su identificador de cuenta y otros campos que pueden agilizar la solución del problema. Este formulario puede consultarse en la figura 7.20 (página 64).

Los reportes se almacenan también en la base de datos, en una colección dedicada. Cada reporte tendrá un identificador único y recoge toda la información necesaria para abordar dicho problema. También contienen un atributo `booleano` llamado **active** para indicar si ya se han solucionado o si todavía no se han abordado. En la figura 7.21 (página 64) se muestra la vista de reporte desde el gestor de base de datos.



Figura 7.20: Página de reporte de problemas

reports > 8061656			Más funciones en Google Cloud
simplex-56bdd	reports	8061656	⋮
+ Iniciar colección	+ Agregar documento	+ Iniciar colección	
reports	5007692	+ Agregar campo	
stats	8061656	active: true	
users	keep	date: 22 de agosto de 2023, 09:28:52 UTC+2	
		id: "8061656"	
		problem: "Cuando la aplicación se inicia el teléfono móvil se queda bloqueado por 5 segundos."	
		userEmail: "joaquin.solla@udc.es"	
		userId: "Yydhg22oFsatml7GdkNSpXsloVN2"	

Figura 7.21: Reporte visto desde el gestor de la base de datos

### CU-40 Ver estadísticas de tester

Aquellos usuarios que tengan el atributo **tester** como verdadero, tendrán disponible en la página de ajustes un espacio de estadísticas acerca de la aplicación. Un usuario solo puede ser **tester** si el desarrollador lo marca como tal desde el gestor de base de datos.

Para cada tipo de estadística se ha creado su respectiva entidad, algunas tienen además métodos auxiliares. Existen dos secciones de estadísticas: de uso de la aplicación y de reportes; en el primer tipo se pueden consultar análisis sobre los usuarios que utilizan la aplicación y métricas de las entidades más usadas por ellos, en el segundo tipo se encuentran estadísticas acerca del número de problemas reportados por los usuarios.

La mayoría de los datos se muestran en forma gráficos, para esto se ha usado el paquete de gráficos **syncfusion\_flutter\_charts** [38]. Puede verse un ejemplo de esta sección en la figura

7.22 (página 65).



Figura 7.22: Estadísticas para testers

### CU-41 Internacionalización

Se han establecido 3 idiomas para la aplicación: Inglés, español y gallego. El primer paso para internacionalizar la aplicación ha sido establecer dichos idiomas en la base de la aplicación con el apartado **supportedLocales**.

Posteriormente se ha realizado la configuración de **Localization (L10N)**, que es el método que se empleará para internacionalizar la aplicación. El idioma definido por defecto para la aplicación será el inglés. Una vez configurado correctamente, el paquete resultante para este trabajo es el mostrado en la figura 7.23 (página 65):

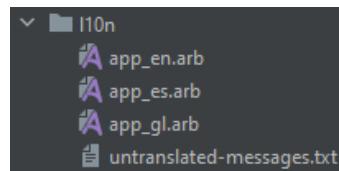


Figura 7.23: Paquete l10n para la internacionalización

El último paso (y el más laborioso) es completar los archivos de traducciones y reemplazar las entradas de texto literales en el código fuente de la aplicación por estas entradas. De esta forma la aplicación se mostrará automáticamente en el idioma en el que esté el dispositivo y, en caso de no ser ninguno de los definidos en la aplicación, se mostrará en inglés. Las entradas de cada idioma se encuentran en su respectivo archivo **ARB**. A continuación se muestra la estructura de un archivo **ARB**:

```

1 {
2   "accept": "Aceptar",
3   "cancel": "Cancelar",
4   "calendar": "Calendario",
5 }
```

Esta es la sintaxis utilizada previamente a la [internacionalización](#):

```
1 Text("Lunes", ...)
```

Y esta es la nueva sintaxis con [L10N](#) implementado:

```
1 Text(AppLocalizations.of(context)!.dayMonday, ...)
```

En la figura 7.24 (página 66) se muestra la aplicación en los 3 idiomas disponibles:

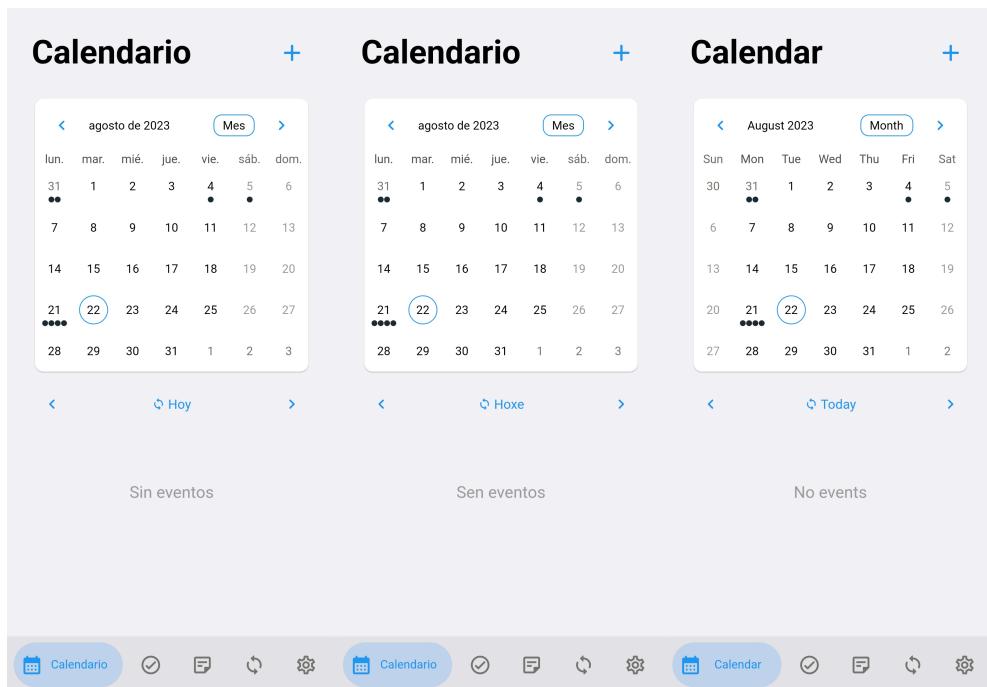


Figura 7.24: La aplicación en los 3 idiomas disponibles (español, gallego e inglés)

Tras finalizar todas las iteraciones que suponen programación, la arquitectura resultante de páginas y sus relaciones es la que se muestra en el diagrama de la figura 7.25 (página 67).

## 7.10 Iteración 9

La última iteración del proyecto corresponde a la elaboración de esta memoria, desarrollada en [LaTeX](#) [24]. En las métricas de tiempo del proyecto también se han tenido en cuenta las horas invertidas en esta iteración.

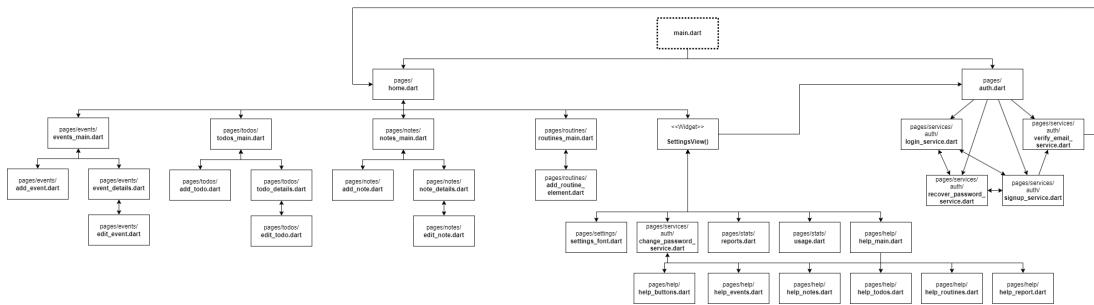


Figura 7.25: Diagrama UML de la arquitectura general de la aplicación

## 7.11 Otros aspectos del desarrollo

A continuación se mencionarán aspectos del desarrollo que son relevantes pero que no constituyen un caso de uso por sí solos.

### 7.11.1 Presentación introductoria

Empleando el paquete **introduction\_screen** [39] se presenta al usuario una introducción a la aplicación la primera vez que se instala la aplicación tal y como se ve en la figura 7.26 (página 67). De esta forma se muestra un resumen rápido y sencillo acerca de las funcionalidades que presta el sistema.

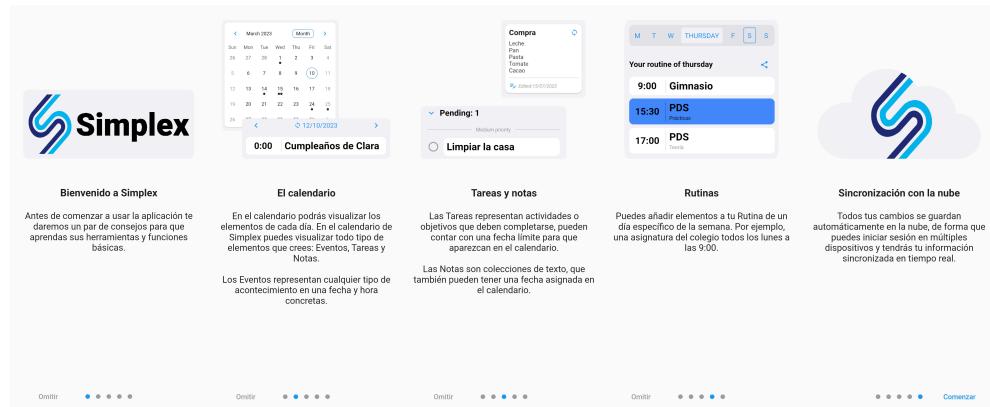


Figura 7.26: Presentación introductoria a la aplicación

### 7.11.2 Depuración del código

Para un mejor entendimiento de la ejecución del código fuente durante el desarrollo, se implementaron mensajes de depuración en todos los métodos y funciones relevantes del sistema para el seguimiento de variables y del flujo de ejecución en tiempo real. Estas líneas de

código sólo funcionan en tiempo de desarrollo, por lo que en la versión de lanzamiento no penalizan en el rendimiento. Este es un ejemplo de salida de depuración:

```
1 I/flutter (23796): [WRN] Cannot show notification 15684938: Time  
out of range
```

### 7.11.3 Colores de la aplicación

La aplicación mantiene unos estándares y reglas establecidos previamente al desarrollo, y eso también incluye a los colores que se mostrarán en la interfaz. Se ha definido una paleta de colores para el tema claro y otra para el tema oscuro. Los [widgets](#) y el texto de la aplicación emplean siempre estas paletas de colores para lograr una interfaz atractiva y coherente.

### 7.11.4 Índices de la base de datos

Para que la base de datos funcione de forma eficiente se han definido [índices](#) para cada consulta que se realiza, ya que estas se llevan a cabo un gran número de veces y sin variaciones.

## Capítulo 8

# Acceso a la aplicación

La aplicación se ha desarrollado con la intención de ser totalmente pública, ya que su propósito es mejorar la organización del día a día de todas las personas y, además, que sea una inspiración y un ejemplo para otros desarrolladores. Es por esto que se puede descargar gratuitamente en Google Play Store [11]. También está disponible todo su código fuente en GitHub [14] para que cualquiera pueda probarla y experimentar con ella, además de ver los patrones de desarrollo utilizados y la estructura de todo el proyecto.

## 8.1 Google Play Store

Se puede descargar la aplicación a través del enlace: Simplex: Organización personal. También se puede obtener a través del código QR de la ilustración 8.1 (página 69).

### Simplex: Organización personal

joaquinssolla

5+  
Descargas | PEGI 3

Instalar en más dispositivos | Compartir

Esta aplicación está disponible para tu dispositivo

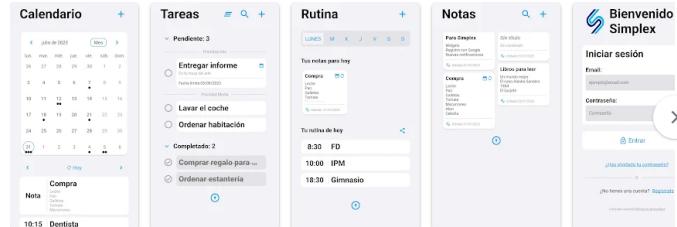


Figura 8.1: Simplex en Google Play Store

## 8.2 GitHub

Se puede acceder al código fuente de la aplicación y a sus versiones de lanzamiento en [GitHub](#).

También se puede acceder a través del código QR de la ilustración 8.2 (página 70).



Figura 8.2: Simplex en GitHub

## 8.3 Política de privacidad

Cualquier usuario que quiera acceder a la política de privacidad [34] podrá hacerlo a través del siguiente enlace: [Política de privacidad](#).

También podrá acceder a través del código QR de la ilustración 8.3 (página 70).



Figura 8.3: Política de privacidad

## Capítulo 9

# Conclusiones y trabajo futuro

---

### 9.1 Conclusiones

Concluido el proyecto, se realiza una visión con retrospectiva del trabajo realizado. Los principales objetivos a cumplir eran: desarrollar una aplicación móvil para que la gente organice su vida diaria y que esta disponga de una interfaz sencilla y atractiva. Podría decirse que se han cumplido los objetivos establecidos.

El desarrollo de este proyecto ha sido largo, durando más de un año, pero se ha adquirido un abanico inmenso de conocimientos, desde refuerzo de las tecnologías [JSON](#) [28] hasta la integración de [Google Firebase](#) [23] con [Flutter](#) [4]; incluso el desarrollo de esta memoria ha participado en la adquisición de conocimientos acerca de [LaTeX](#) [24]. También se han adquirido habilidades con un esquema de bases de datos que nunca había empleado, y se han puesto en práctica muchos conocimientos adquiridos en el grado: diseño de interfaces, diseño de esquemas de datos, estimación de proyectos, patrones de desarrollo, etc.

La aplicación va a mantenerse pública y gratuita en Google Play Store [11], así como su código fuente en GitHub [14] para que siempre esté disponible para todas aquellas personas que quieran usarla.

En la actualidad son varias las personas que ya usan la aplicación a diario, lo que supone toda una satisfacción y un orgullo haber desarrollado algo realmente útil para la gente.

### 9.2 Trabajo futuro

Otra forma de ver el proyecto es como un sistema en evolución, ya que la aplicación tiene mucho potencial y margen de mejora. En el futuro se podrían añadir nuevas funcionalidades y elementos, algunas mejoras a añadir son:

- **Widgets de escritorio:** Elementos que el usuario pueda colocar en el escritorio de su dispositivo móvil y que funcionen como visual rápida a su calendario y listas de

elementos sin necesidad de abrir la aplicación.

- **Creación de eventos mediante enlaces compartidos:** Permitir que el usuario pueda compartir un evento a otra persona mediante un enlace, y que este enlace abra la aplicación con el formulario de creación de evento completado con los datos del evento compartido, esperando la confirmación de creación.
- **Soporte de multimedia:** Soporte por parte de la aplicación para elementos de audio o imagen para añadir a las notas y demás elementos.
- **Una aplicación web:** Una aplicación web con las mismas funcionalidades que la aplicación móvil. La cuenta sería la misma en ambos servicios y la información estaría sincronizada entre ellos.

# **Apéndices**

# Lista de acrónimos

---

**AAB** AppBundle. [63](#)

**AppArmor** Application Armor. [63](#)

**ARB** Application Resource Bundle. [30](#), [35](#), [65](#)

**BaaS** Backend as a Service. [1](#), [4](#)

**FSR** Firestore Security Rules. [47](#)

**IDE** Integrated Development Environment. [28](#)

**JSON** JavaScript Object Notation. [30](#), [50](#), [52](#), [71](#)

**L10N** Localization. [65](#), [66](#)

**LaTeX** Lamport TeX. [25](#), [66](#), [71](#)

**NoSQL** Non SQL / Not only SQL. [36](#)

**POO** Programación Orientada a Objetos. [28](#)

**SCV** Sistema de Control de Versiones. [32](#)

**XML** eXtensible Markup Language. [30](#)

# Glosario

---

**Adobe Photoshop** Editor de fotografías desarrollado por Adobe Systems Incorporated. Usado principalmente para el retoque de fotografías y gráficos. [32](#)

**Android** Sistema operativo móvil perteneciente a Google. [6](#), [7](#), [12](#), [22](#), [28](#), [33](#)

**Android Studio** IDE dedicado al desarrollo de aplicaciones móviles, en especial aplicaciones Android. [28](#), [29](#), [37](#), [41](#)

**backend** Parte de un sistema que maneja los procesos y la gestión de datos que no son visibles directamente para el usuario final, pero que son esenciales para el funcionamiento de la aplicación. [33](#)

**booleano** Tipo de dato que sólo puede tener dos valores posibles: verdadero o falso. [39](#), [55](#), [60](#), [63](#)

**commit** Guardado y registro de los cambios realizados en el código fuente de un proyecto en un sistema de control de versiones. [32](#)

**Dart** Lenguaje de programación de código abierto, desarrollado por Google. [4](#), [22](#), [28](#), [29](#), [34](#), [48](#)

**depuración** Proceso de probar, identificar, analizar y corregir errores o defectos en un programa o sistema. [67](#), [68](#)

**Firebase Authentication** Servicio de autenticación de usuarios para aplicaciones ofrecido por Google, aparece contenido en Google Firebase. [4](#), [30](#), [33](#), [35](#), [44](#), [46](#)

**Firestore Database** Servicio de base de datos proporcionado por Google e integrado en Google Firebase. [1](#), [4](#), [31](#), [33](#), [35](#), [36](#), [47](#), [50](#)

**Flutter** Framework de código fuente abierto de desarrollo de aplicaciones móviles creado por Google. [4](#), [12](#), [22](#), [28](#), [71](#)

**framework** Conjunto de componentes, bibliotecas, estándares y patrones de diseño que permiten a los desarrolladores construir aplicaciones de manera rápida y organizada. [4](#), [22](#), [28](#)

**frontend** Es la interfaz de usuario y la capa de presentación de un sistema que permite a los usuarios interactuar con la funcionalidad y los datos proporcionados por el backend. [33](#)

**gestor de tiempo** Herramienta que se emplea para hacer un seguimiento del tiempo que se dedica en un proyecto. [12](#)

**globalización** Diseño y desarrollo de productos software para que sean adecuados y funcionales en diferentes contextos culturales, lingüísticos y regionales en todo el mundo. [3](#)

**Google Firebase** Plataforma para el desarrollo de aplicaciones web y aplicaciones móviles desarrollada por Google. [1](#), [22](#), [30](#), [31](#), [41](#), [71](#)

**Gradle** Sistema de automatización de construcción de código de software. [22](#), [29](#), [33](#), [37](#), [48](#), [63](#)

**internacionalización** Diseño y desarrollo de aplicaciones que se adapten a diferentes idiomas, formatos numéricos o formatos de fecha y hora. [24](#), [30](#), [35](#), [66](#)

**iOS** Sistema operativo móvil perteneciente a Apple. [7](#)

**kanban** Método de gestión de tareas que se utiliza en diversos contextos, incluyendo el desarrollo de software. Su elemento principal es un tablero en donde se colocan las tareas. [11–13](#), [32](#)

**Locale** Tipo de dato Dart que representa un idioma y una región. [39](#)

**mockup** Representación visual o boceto que simula la apariencia y funcionamiento de un producto como una página web o una aplicación móvil. [2](#), [3](#), [14](#), [15](#), [17](#), [18](#), [20](#), [21](#)

**Scrum** Metodología de trabajo ágil utilizada en el desarrollo de software y en la gestión de proyectos en general. [13](#)

**Selenium** Entorno de pruebas automáticas de software para aplicaciones basadas en la web. [12](#)

**tester** Persona designada para probar una aplicación, tanto versiones preliminares como lanzamientos ya públicos con el objetivo de detectar posibles fallos y mejoras. [20](#), [24](#), [31](#), [34](#), [64](#)

**widget** Objeto Dart que forma parte de la interfaz de usuario. [29](#), [34](#), [37–40](#), [44](#), [48](#), [49](#), [52](#), [56](#), [68](#), [71](#)

**índice** Estructura utilizada en sistemas de gestión de bases de datos para mejorar la velocidad de búsqueda y recuperación de datos en una tabla o colección. [68](#)

# Bibliografía

---

- [1] “Whatsapp | mensajería y llamadas gratuitas privadas, seguras y confiables.” [En línea]. Disponible en: [https://www.whatsapp.com/?lang=es\\_LA](https://www.whatsapp.com/?lang=es_LA)
- [2] “Serverless architectures.” [En línea]. Disponible en: <https://martinfowler.com/articles/serverless.html>
- [3] “Dart programming language | dart.” [En línea]. Disponible en: <https://dart.dev/>
- [4] “Flutter - build apps for any screen.” [En línea]. Disponible en: <https://flutter.dev/>
- [5] “Firebase authentication.” [En línea]. Disponible en: <https://firebase.google.com/docs/auth?hl=es-419>
- [6] “Cloud firestore - firebase.” [En línea]. Disponible en: <https://firebase.google.com/docs/firestore?hl=es-419>
- [7] J. S. Vázquez, “Simplex: Organización personal.” [En línea]. Disponible en: <https://play.google.com/store/apps/details?id=com.joa.simplex>
- [8] “Google calendar.” [En línea]. Disponible en: <https://calendar.google.com/calendar/u/0/r>
- [9] “La plataforma que amplía los límites de lo posible - android.” [En línea]. Disponible en: [https://www.android.com/intl/es\\_es/](https://www.android.com/intl/es_es/)
- [10] “Gmail: Correo electrónico sin coste, privado y seguro - google.” [En línea]. Disponible en: <https://www.google.com/intl/es/gmail/about/>
- [11] “Aplicaciones de android en google play.” [En línea]. Disponible en: <https://play.google.com/store/games?hl=es&gl=US>
- [12] “Any.do.” [En línea]. Disponible en: <https://play.google.com/store/apps/details?id=com.anydo&hl=es&gl=US>
- [13] “ios - apple (es).” [En línea]. Disponible en: <https://www.apple.com/es/ios>

- [14] “Github: Let’s build from here · github.” [En línea]. Disponible en: <https://github.com/>
- [15] “Microsoft teams.” [En línea]. Disponible en: <https://www.microsoft.com/es-es/microsoft-teams/log-in>
- [16] “Evernote.” [En línea]. Disponible en: <https://play.google.com/store/apps/details?id=com.evernote&hl=es&gl=US>
- [17] “Microsoft to do.” [En línea]. Disponible en: [https://play.google.com/store/apps/details?id=com.microsoft.todos&hl=es\\_419&gl=US](https://play.google.com/store/apps/details?id=com.microsoft.todos&hl=es_419&gl=US)
- [18] “Todoist: Lista de tareas.” [En línea]. Disponible en: [https://play.google.com/store/apps/details?id=com.todoist&hl=es\\_419&gl=US](https://play.google.com/store/apps/details?id=com.todoist&hl=es_419&gl=US)
- [19] “Fantastical calendar 2023.” [En línea]. Disponible en: [https://play.google.com/store/apps/details?id=com.calendarview.todomanage&hl=es\\_419&gl=US](https://play.google.com/store/apps/details?id=com.calendarview.todomanage&hl=es_419&gl=US)
- [20] “Selenium.” [En línea]. Disponible en: <https://www.selenium.dev/>
- [21] “Your opensource agile project management software - taiga.” [En línea]. Disponible en: <https://taiga.io/es>
- [22] “Gradle build tool.” [En línea]. Disponible en: <https://gradle.org/>
- [23] “Firebase | google’s mobile and web app development platform.” [En línea]. Disponible en: <https://firebase.google.com>
- [24] “Latex - a document preparation system.” [En línea]. Disponible en: <https://www.latex-project.org/>
- [25] “Real decreto legislativo 2/2015, de 23 de octubre, por el que se aprueba el texto refundido de la ley del estatuto de los trabajadores.” [En línea]. Disponible en: <https://www.boe.es/buscar/act.php?id=BOE-A-2015-11430#a34>
- [26] “Download android studio.” [En línea]. Disponible en: <https://developer.android.com/studio>
- [27] “Jetbrains.” [En línea]. Disponible en: <https://www.jetbrains.com/es-es/>
- [28] “Introducción a json (javascript object notation).” [En línea]. Disponible en: <https://www.json.org/json-es.html>
- [29] “Google play console.” [En línea]. Disponible en: <https://play.google.com/intl/es/console/about/>

- [30] “Git.” [En línea]. Disponible en: <https://git-scm.com/>
- [31] “Adobe photoshop oficial.” [En línea]. Disponible en: <https://www.adobe.com/es/products/photoshop.html>
- [32] “flutter\_local\_notifications | flutter package.” [En línea]. Disponible en: [https://pub.dev/packages/flutter\\_local\\_notifications](https://pub.dev/packages/flutter_local_notifications)
- [33] “shared\_preferences | flutter package.” [En línea]. Disponible en: [https://pub.dev/packages/shared\\_preferences](https://pub.dev/packages/shared_preferences)
- [34] “Política de privacidad de simplex.” [En línea]. Disponible en: [https://joaquinssolla.com/privacy\\_policies/simplex.html](https://joaquinssolla.com/privacy_policies/simplex.html)
- [35] “table\_calendar | flutter package.” [En línea]. Disponible en: [https://pub.dev/packages/table\\_calendar](https://pub.dev/packages/table_calendar)
- [36] “social\_share | flutter package.” [En línea]. Disponible en: [https://pub.dev/packages/social\\_share](https://pub.dev/packages/social_share)
- [37] “flutter\_staggered\_grid\_view | flutter package.” [En línea]. Disponible en: [https://pub.dev/packages/flutter\\_staggered\\_grid\\_view](https://pub.dev/packages/flutter_staggered_grid_view)
- [38] “syncfusion\_flutter\_charts | flutter package.” [En línea]. Disponible en: [https://pub.dev/packages/syncfusion\\_flutter\\_charts](https://pub.dev/packages/syncfusion_flutter_charts)
- [39] “introduction\_screen | flutter package.” [En línea]. Disponible en: [https://pub.dev/packages/introduction\\_screen](https://pub.dev/packages/introduction_screen)