



MÓDULO PROYECTO

CFGS Desarrollo de Aplicaciones
Multiplataforma
Informática y Comunicaciones



Tutor individual: Rodrigo Iglesias Gorron

Tutor colectivo: Cristina Silvan Pardo

Año: 2024/2025

Fecha de presentación: 23 de mayo 2025

Nombre y Apellidos: Estela de Vega Martín
Email: estela.vegmar@educa.jcyl.es



Tabla de contenido

1	Identificación proyecto.....	7
2	Organización de la memoria	7
3	Descripción general del proyecto	8
3.1	Objetivos	8
3.2	Cuestiones metodológicas	9
3.3	Entorno de trabajo (tecnologías de desarrollo y herramientas)	10
4	Descripción general del producto	11
4.1	Visión general del sistema: límites del sistema, funcionalidades básicas, usuarios y/o otros sistemas con los que pueda interactuar.	11
4.1.1	Visión general del sistema	11
4.1.2	Límites del sistema.....	11
4.1.3	Funcionalidades básicas.....	11
	Usuarios	12
	Compatibilidad con sistemas	12
4.2	Descripción breve de métodos, técnicas o arquitecturas(m/t/a) utilizadas.	13
4.3	Despliegue de la aplicación indicando plataforma tecnológica, instalación de la aplicación y puesta en marcha	14
4.3.1	Plataforma tecnológica	14
4.3.2	Instalación de la aplicación	14
5	Planificación y presupuesto.....	15
5.1	Planificacion	15
5.2	Presupuesto.....	18
5.2.1	Coste de desarrollo de codigo.....	18
5.2.2	Coste de software y herramientas de desarrollo.....	18
5.2.3	Coste del hardware	19
5.2.4	Coste de hosting.....	19
5.2.5	Coste materiales auxiliares	19

5.2.6	Coste material promocional.....	20
6	Documentación Técnica: análisis, diseño, implementación y pruebas	20
6.1	Especificación de requisitos.....	20
6.1.1	Requisitos funcionales	20
6.1.2	Requisitos no funcionales	21
6.2	Análisis del sistema.....	22
6.2.1	1. Gestión y manejo de tiempos	22
6.2.2	2. Gestión de penalizaciones	23
6.2.3	3. Búsquedas filtradas.....	23
6.2.4	4. Gestión de tipos de cubo y sesiones	23
6.2.5	5. Generación de scrambles.....	24
6.2.6	6. Generación de informes PDF	24
6.2.7	7. Autenticación de usuarios.....	24
6.2.8	8. Gestión del perfil de usuario	24
6.2.9	9. Estadísticas visuales	25
6.2.10	10. Configuración avanzada.....	25
6.3	Diseño del sistema:	26
6.3.1	Diseño de la Base de Datos	26
6.3.2	Diseño de la Interfaz de usuario.....	31
6.3.3	Diseño de la Aplicación.....	52
6.4	Implementación:.....	57
6.4.1	Entorno de desarrollo.	57
6.4.2	Estructura del código.	57
6.4.3	Cuestiones de diseño e implementación reseñables.	59
6.5	Pruebas.....	77
6.5.1	Pruebas de la aplicación.....	77
6.5.2	Seguridad	78

7	Manuales de usuario	82
7.1	Manual de usuario	82
7.1.1	INICIO DE LA APLICACIÓN	82
7.1.2	Inicio de sesión	82
7.1.3	Crear cuenta	83
7.1.4	PÁGINA PRINCIPAL: TIMER	84
7.1.5	Seleccionar o crear tipo de cubo	85
7.1.6	Seleccionar, crear o eliminar sesión	86
7.1.7	Scramble.....	87
7.1.8	Timer	88
7.1.9	PANTALLA HISTORIAL.....	91
7.1.10	Seleccionar tipo de cubo o sesión.....	91
7.1.11	Añadir tiempo manualmente	91
7.1.12	Búsqueda de tiempos.....	91
7.1.13	Opciones avanzadas.....	92
7.1.14	Visualización detalles de un tiempo	92
7.1.15	Compartir tiempos de la sesión	93
7.1.16	Eliminación múltiple de tiempos	94
7.1.17	PANTALLA ESTADISTICAS.....	94
7.1.18	NAVEGACIÓN	96
7.1.19	Navegación inferior.....	96
7.1.20	Navegación con Drawer.....	97
7.2	Manual de instalación	102
7.2.1	Instalar herramientas básicas	102
7.2.2	Instalacion y configuración de Flutter.....	102
7.2.3	Configurar Android Studio	103
7.2.4	Configuracion en IntelliJ.....	105

7.2.5	Configurar Supabase	106
7.2.6	Clonar y ejecutar el proyecto	111
8	Conclusiones y posibles ampliaciones	113
9	Bibliografía	115
9.1	Documentacion oficial de flutter.....	115
-	Flutter Material Components.....	115
-	Flutter Widgets y Layouts.....	115
-	Accesibilidad e internacionalización	115
-	Flutter Drawer	115
9.2	Herramientas y paquetes externos (pub.dev).....	115
-	Navegación y diálogos	115
-	Personalización UI	115
-	Compartir y exportar contenido.....	116
-	Notificaciones, audio y vibración	116
9.3	Seguridad y cifrado	116
9.4	Supabase	116
-	Uso e integración.....	116
-	Tutoriales.....	116
9.5	Recursos prácticos (StackOverflow, Medium, etc.)	116
-	UI personalizada y lógica	116
-	Manejo de formularios y datos	116
-	Acciones del sistema	117
-	Manejo de formularios y datos	117
-	Widget Icon	117
-	Problemas en paquetes.....	117
-	Selector de imágenes (imagen_picker)	117
-	Waves y CustomPainter	117

9.6	YouTube – Tutoriales y soluciones	117
-	Interfaces y navegación.....	117
-	Funciones avanzadas.....	117
9.7	Widget avanzados (Popover, Tooltip, Search, Charts)	118
-	Tooltip y Popover	118
-	Charts:	118
-	SearchDelegate.....	118
9.8	Herramientas externas	118
-	Generación de voz para archivos MP3	118
-	Logica del scramble del Square-1.....	118
9.9	Recursos personales y clases.....	119
-	Ejercicios y exámenes.....	119
-	Ayuda mediante Inteligencia Artificial	119
-	Documentación interna del IDE	119
10	Anexos	120
10.1	Anexo I – Código fuente de la aplicación	120
10.2	Anexo II: Backlog del Proyecto	120
10.3	Anexo III – Prototipo de diseño (Figma).....	121
10.4	Anexo V – Tutorial de bienvenida	121
10.5	Registro de tiempo de desarrollo.....	126
10.6	Página de documentación	127

1 Identificación proyecto

Este proyecto se centra en el desarrollo de una aplicación multiplataforma destinada a personas que practican la resolución de cubos de Rubik. La app integra un sistema de cronometraje preciso, generación automática de scrambles válidos según el tipo de cubo, y funcionalidades de personalización y análisis de resultados, todo ello orientado a ofrecer una herramienta práctica y completa para el entrenamiento y mejora diaria.

A lo largo del desarrollo se han abordado distintos aspectos técnicos clave como la programación de scrambles aleatorios según las reglas de cada modalidad (3x3, Skewb, Megaminx, etc.), el diseño de una interfaz adaptable a distintos usuarios, la gestión de notificaciones personalizadas, y la persistencia de datos mediante almacenamiento local y sincronización con servicios externos.

El objetivo principal ha sido crear una aplicación estable, funcional y ajustada a las necesidades reales de los usuarios que practican speedcubing, aplicando conocimientos adquiridos en el grado.

2 Organización de la memoria

Esta memoria está dividida en varios apartados para explicar de forma clara todo el desarrollo del proyecto. A continuación, se explica brevemente qué contiene cada parte:

- ❖ **1. Identificación del proyecto:** Aquí se presenta el título del proyecto, el autor, el curso y otros datos básicos que lo identifican en la portada y una breve descripción.
- ❖ **3. Descripción general del proyecto:** Se habla sobre los objetivos principales, cómo se ha planteado el desarrollo y qué herramientas se han utilizado.
- ❖ **4. Descripción general del producto:** Se explica qué hace la aplicación, para quién está pensada y cómo se conecta con otras partes del sistema .
- ❖ **5. Planificación y presupuesto:** Se muestra cómo se ha organizado el tiempo para desarrollar el proyecto y se hace una estimación de los costes (software, hardware, horas de trabajo...).
- ❖ **6. Documentación técnica:** Aquí se incluyen los requisitos, el análisis del sistema, el diseño de la base de datos y la interfaz, cómo se ha implementado todo, y las pruebas que se han realizado.

- ❖ **7. Manuales de usuario e instalación:** Se explican paso a paso las pantallas de la app y cómo usarla, además de cómo instalar y ejecutar el proyecto en distintos entornos.
- ❖ **8. Conclusiones y posibles ampliaciones:** Se hace una reflexión final sobre el trabajo realizado y algunas ideas que se podrían añadir en el futuro.
- ❖ **9. Bibliografía:** Lista de webs, tutoriales y documentación utilizada como apoyo durante el desarrollo.
- ❖ **10. Anexos:** Material adicional que complementa el contenido principal: código fuente, prototipos en Figma, el tutorial guiado con Cubix, etc.

3 Descripción general del proyecto

3.1 Objetivos

El proyecto tiene como objetivo principal desarrollar una aplicación que integre un **temporizador avanzado para la resolución de cubos de Rubik**, ofreciendo a los usuarios una experiencia personalizable y herramientas para registrar, analizar y mejorar sus tiempos de resolución.

A nivel profesional y personal, se persiguen los siguientes objetivos:

Objetivos principales:

- **Desarrollar un timer avanzado:** Implementar un temporizador preciso y funcional que permita gestionar tiempos, generar scrambles aleatorios y muestre estadísticas de resolución.
- **Aplicar y ampliar conocimientos técnicos:** Poner en práctica los conocimientos adquiridos durante el ciclo formativo, profundizando en el desarrollo móvil, bases de datos, programación de APIs y autenticación de usuarios.
- **Crear una aplicación estable y funcional:** Implementar una aplicación que garantice el correcto funcionamiento de la aplicación en distintas plataformas, priorizando la estabilidad, el rendimiento y la usabilidad.

Objetivos secundarios:

- **Diseñar una interfaz atractiva, intuitiva y accesible:** Crear una experiencia de usuario agradable, accesible para principiantes pero funcional para usuarios avanzados. Además de facilidad de usar para todo tipo de personas, incluyendo aquellas con dificultades visuales o de accesibilidad donde se pueden utilizar lectores de pantalla como TalkBack.
- **Facilitar la evolución personal del usuario:** permitir el seguimiento del progreso mediante estadísticas y gráfico evolutivo mensual, anual y diariamente.

3.2 Cuestiones metodológicas

Forma de realizar el proyecto:

El desarrollo de este proyecto ha seguido un enfoque mixto, comenzando con una **metodología en cascada** y luego evolucionando hacia un enfoque ágil inspirado en **Scrum**, en todo este proceso se siguió la metodología **Git Flow**, basada en la gestión de ramas de Git. A continuación, se detallan las fases y el enfoque metodológico adoptado en cada una de ellas:

1. Fase inicial (Metodología en Cascada - diciembre a febrero):

En las primeras fases del proyecto, desde diciembre hasta febrero, se adoptó el modelo **en cascada**, el cual es adecuado cuando los requisitos son claros y se busca un enfoque estructurado. Esta fase incluyó la elaboración de un **prototipo en Figma**, que permitió diseñar la interfaz de usuario de la aplicación de manera visual. A continuación, se realizaron los **casos de uso** y el **diagrama UML** para analizar la arquitectura del sistema y las interacciones entre sus componentes.

Una vez definidos los diseños y la arquitectura, se procedió a la **implementación** de la aplicación. Durante esta fase, también se realizaron pruebas **unitarias, de integración y de widget** para asegurar que cada módulo funcionara correctamente y que las integraciones entre ellos fueran exitosas.

2. Fase de implementación ágil (Metodología Scrum - febrero a mayo):

A partir de **febrero**, se adoptó un enfoque más ágil, inspirado en **Scrum**, para la gestión del desarrollo de funcionalidades. Durante esta fase, el proyecto se dividió en pequeñas tareas y se asignaron a diferentes ramas dentro del sistema de control de versiones utilizando **Git Flow**.

Además, se utilizó un [**backlog**](#) como herramienta para gestionar el progreso de las tareas, como se explica en el Anexo II.

3. Fase final (Ajustes y despliegue - mayo):

En las últimas semanas del proyecto, se realizaron los **ajustes finales** a la aplicación. También se llevaron a cabo pruebas de **instalación** y se verificó el correcto funcionamiento de la aplicación en los entornos.

3.3 Entorno de trabajo (tecnologías de desarrollo y herramientas)

Tecnologías utilizadas:

El proyecto sigue un modelo **cliente-servidor**, donde la aplicación móvil actúa como cliente y se conecta a un backend gestionado por **Supabase**, que cumple el rol de servidor. A pesar de no tratarse de una aplicación web, la arquitectura distribuida permite gestionar datos y autenticación de forma remota a través de una API REST.

Herramientas software y lenguajes de programación

Herramienta o lenguaje	Versión usada	Propietario o desarrollador	Tipo de licencia	Uso en el proyecto
Dart	Última versión estable 3.6.1	Google	Licencia BSD (código abierto permisivo, similar a MIT)	Lenguaje principal de programación
Flutter	3.27.2	Google	Licencia BSD (código abierto permisivo)	Framework para app multiplataforma
IntelliJ IDEA	IntelliJ IDEA 2022.3.3 (Community Edition)	JetBrains	Licencia de código abierto (Community Edition, gratuita)	IDE usado para el desarrollo
Supabase	Supabase no tiene una versión definida, pero en este proyecto se utiliza supabase_flutter: ^2.8.4.	Supabase Inc	Licencia Creative Commons BY-NC-SA (uso no comercial, con atribución y distribución bajo la misma licencia)	Backend: base de datos y autenticación
Figma	(versión web – última estable)	Figma, Inc.	Software propietario, (plan gratuito)	Diseño de interfaz y prototipado para el wireframe.

Git	git version 2.40.0.windows.1	Linus Torvalds	Licencias Open Source: MIT, Apache, GPL	Control de versiones local
GitHub	(versión web – cuenta personal) web (última versión)	Microsoft (anteriormente: Chris Wanstrath, P. J. Hyett, Tom Preston-Werner, Scott Chacon)	Software propietario	Repositorio remoto y gestión de tareas.

4 Descripción general del producto

4.1 Visión general del sistema: límites del sistema, funcionalidades básicas, usuarios y/o otros sistemas con los que pueda interactuar.

4.1.1 Visión general del sistema

La aplicación desarrollada es un sistema destinado a la medición, gestión y análisis de tiempos en la resolución de cubos de Rubik. Está dirigida a personas aficionadas o competitivas que desean registrar su rendimiento y progresar en su práctica.

4.1.2 Límites del sistema

La aplicación es **autónoma**, aunque depende de servicios externos (como Supabase) para funcionalidades como el almacenamiento de datos y la autenticación. No forma parte de un sistema más amplio ni se integra con otras aplicaciones externas, pero sí puede interactuar con otros sistemas a través de funciones como la **generación y compartición de archivos PDF**.

4.1.3 Funcionalidades básicas

- ✿ **Gestión y manejo de tiempos:** Registrar, eliminar o editar tiempos de resolución.

- ❖ **Gestión de penalizaciones:** Añadir o modificar penalizaciones (ninguna, +2 segundos o DNF).
- ❖ **Búsquedas filtradas:** Filtrar por tiempo o comentarios asociados a cada tiempo de resolución.
- ❖ **Gestión de tipos de cubo y sesiones:** Gestionar tipos de cubos y sus sesiones referenciadas.
- ❖ **Generación de scrambles:** Generación automática de scrambles adaptados al tipo de cubo.
- ❖ **Generación de informes PDF:** Exportar los tiempos de una sesión en formato PDF y compartirlos desde la app.
- ❖ **Autenticación de usuarios:** Registro y verificación por correo electrónico y login.
- ❖ **Gestión de usuarios:** Configuración del perfil del usuario.
- ❖ **Estadísticas visuales:** Gráficas de rendimiento por sesión con apartados como tiempo medio, mejor tiempo, distintas medias, porcentaje de distintas penalizaciones, el uso de tiempo total de la aplicación, entre otras.
- ❖ **Gestión de opciones avanzadas:** Personalización de la experiencia de usuario con configuraciones como sonidos, vibraciones, visibilidad del tiempo o de la inspección, etc.
- ❖ **Notificaciones:** Envío de notificaciones diarias.

Usuarios

Existe un único tipo de usuario registrado. Para acceder a la aplicación, es obligatorio crear una cuenta con una dirección de correo electrónico válida. El proceso de autenticación incluye el envío de un correo de verificación; el usuario no podrá acceder hasta confirmar su cuenta. Esta verificación garantiza mayor seguridad y control de acceso.

Compatibilidad con sistemas

Gracias a Flutter, la aplicación es **multiplataforma** y puede instalarse en Android, iOS, Windows, macOS y Linux.

4.2 Descripción breve de métodos, técnicas o arquitecturas(m/t/a) utilizadas.

En el desarrollo de esta aplicación se ha utilizado una arquitectura basada en el patrón **MVVM (Model - View - ViewModel)**, adaptada al entorno de Flutter con una separación entre los datos, la lógica de la aplicación y la interfaz de usuario. Este enfoque facilita el mantenimiento del código y la escalabilidad del proyecto.

Como técnicas principales, destacan:

- ❖ **Autenticación por correo electrónico** mediante Supabase.
- ❖ **Almacenamiento en la nube** con base de datos relacional gracias a Supabase.
- ❖ **Generación de informes PDF** y uso de mecanismos para compartir.
- ❖ **Control de versiones con Git** y gestión del proyecto con GitHub.

Estos puntos se han desarrollado con más detalle en el apartados anterior del documento.

Durante el desarrollo del proyecto se aplicaron diversos métodos y técnicas de trabajo que facilitaron la organización y la construcción progresiva de la aplicación:

- ❖ **Desarrollo por fases:** El proyecto se estructuró en ramas separadas, cada una centrada en una funcionalidad específica. Estas ramas se integraban progresivamente en la rama principal de desarrollo (dev), permitiendo un control claro del avance y una integración continua.
- ❖ **Desarrollo incremental:** A lo largo del proceso se generaron distintas versiones del proyecto (v1.0, v1.1 y una versión final), cada una con nuevas funcionalidades incorporadas, hasta completar la aplicación.
- ❖ **Prototipado previo y evolución del diseño:** Antes de iniciar el desarrollo, se elaboró un prototipo funcional en Figma para definir la interfaz de usuario y facilitar la planificación visual del producto inicial. A lo largo del desarrollo, se añadieron nuevas funcionalidades que no estaban contempladas en el diseño original, por lo que se volvió a actualizar ese prototipo para reflejar el resultado final.
- ❖ **Control de versiones:** Se utilizó Git como sistema de control de versiones y GitHub como plataforma para alojar el repositorio y gestionar los cambios.

- ❖ **Organización personal:** La planificación del trabajo se apoyó en la herramienta GitHub Projects, donde se gestionaron las tareas mediante tableros, y en una lista de control personal elaborada en un bloc de notas.

Estos métodos permitieron un desarrollo ordenado, facilitando tanto la organización como la implementación técnica del proyecto.

4.3 Despliegue de la aplicación indicando plataforma tecnológica, instalación de la aplicación y puesta en marcha

4.3.1 Plataforma tecnológica

Informacion del sistema operativo

Herramienta o lenguaje	Tipo
Nombre del sistema operativo	Microsoft Windows 10 Home
Versión del sistema operativo	10.0.19045 N/D Compilación 19045
Tipo de sistema	x64-based PC
Procesador(es):	1 Procesadores instalados. [01]: Intel64 Family 6 Model 158 Stepping 13 GenuineIntel ~2592 Mhz
RAM	16.262 MB

4.3.2 Instalación de la aplicación

La aplicación puede instalarse fácilmente desde el repositorio del proyecto en GitHub, dentro del apartado “Releases”. Pasos:

- Acceder al repositorio:
https://github.com/estelaV9/TFG_CubeX
- Entrar en la sección “Releases”: Allí se encuentran las versiones compiladas para cada plataforma.
- Descargar el archivo correspondiente:
 - o .apk para Android.
 - o .exe para Windows.

Nota: Aunque no ha sido posible generar ni probar los ejecutables para **iOS** y **macOS** debido a la falta de acceso a un equipo Apple con macOS y Xcode, la aplicación ha sido **configurada para soportar ambas plataformas**.

Todo el código ha sido desarrollado con compatibilidad multiplataforma en mente utilizando **Flutter**. Además, la integración con **Supabase** está implementada, lo que debería permitir que la aplicación funcione sin problemas en iOS y macOS una vez compilada en un entorno adecuado.

5 Planificación y presupuesto

5.1 Planificación

La planificación del proyecto se estructuró a lo largo de diferentes fases, detallando las actividades y tareas que se realizaron mes a mes. A continuación se presenta la planificación dividida por meses y tareas clave, incluyendo las migraciones de la base de datos, la integración de nuevas funcionalidades y la actualización del diseño de la aplicación.

1. Diciembre a febrero (Fase inicial de desarrollo):

Durante esta fase, se completaron las tareas iniciales del proyecto, que incluyeron:

- **Diseño del prototipo** en Figma, que sirvió como base para la interfaz de usuario.
- **Elaboración de los casos de uso y diagrama UML** para definir la estructura y el flujo de la aplicación.
- **Implementación inicial** de la aplicación, donde se desarrollaron las primeras funcionalidades y la estructura básica.
- **Pruebas unitarias, de integración y de widget** para asegurar que las funcionalidades iniciales se implementaron correctamente.

Objetivo principal: Completar el diseño del prototipo, la implementación inicial y realizar las primeras pruebas para garantizar la estabilidad de la aplicación.

2. Febrero a mayo (Fase de implementación ágil y mejoras):

En esta fase, se adoptó un enfoque más **ágil**, permitiendo que el desarrollo avanzara de manera más flexible mediante tareas iterativas. A partir de febrero, se organizó el trabajo mediante un **backlog** de tareas, en el cual se estructuraron y priorizaron las actividades. Las tareas principales fueron:

- **Migración de la base de datos a Supabase:** Este fue uno de los cambios más importantes, ya que se trasladó la gestión de la base de datos local a una API externa. Esto permitió mejorar la escalabilidad y la accesibilidad de los datos.
- **Desarrollo de nuevas funcionalidades e integración con Supabase:** Se implementaron nuevas características y funcionalidades dentro de la aplicación, adaptando el código para trabajar con la nueva base de datos.
- **Actualización del prototipo en Figma:** Tras la integración de las nuevas funcionalidades, se actualizó el prototipo para reflejar los cambios en la interfaz de usuario y los nuevos flujos de la aplicación.
- **Pruebas de integración y unitarias:** A medida que se completaban las tareas de desarrollo, se realizaron pruebas para asegurar que la integración entre las nuevas funcionalidades y el sistema fuese correcta.

Objetivo principal: Completar la migración de la base de datos, añadir nuevas funcionalidades y actualizar el diseño de la aplicación.

3. Mayo (Fase final de pruebas y despliegue):

Durante la fase final, se llevaron a cabo las siguientes tareas:

- **Ajustes finales al diseño:** Se realizaron pequeños ajustes en el diseño de la aplicación para asegurar que fuera coherente con la nueva funcionalidad implementada.
- **Pruebas de instalación y validación:** Se realizaron pruebas para garantizar que la aplicación funcionara correctamente en los dispositivos finales y que el proceso de instalación fuera sencillo y sin errores.
- **Preparación para el despliegue:** Se completaron los últimos pasos para la implementación final y el despliegue de la aplicación.

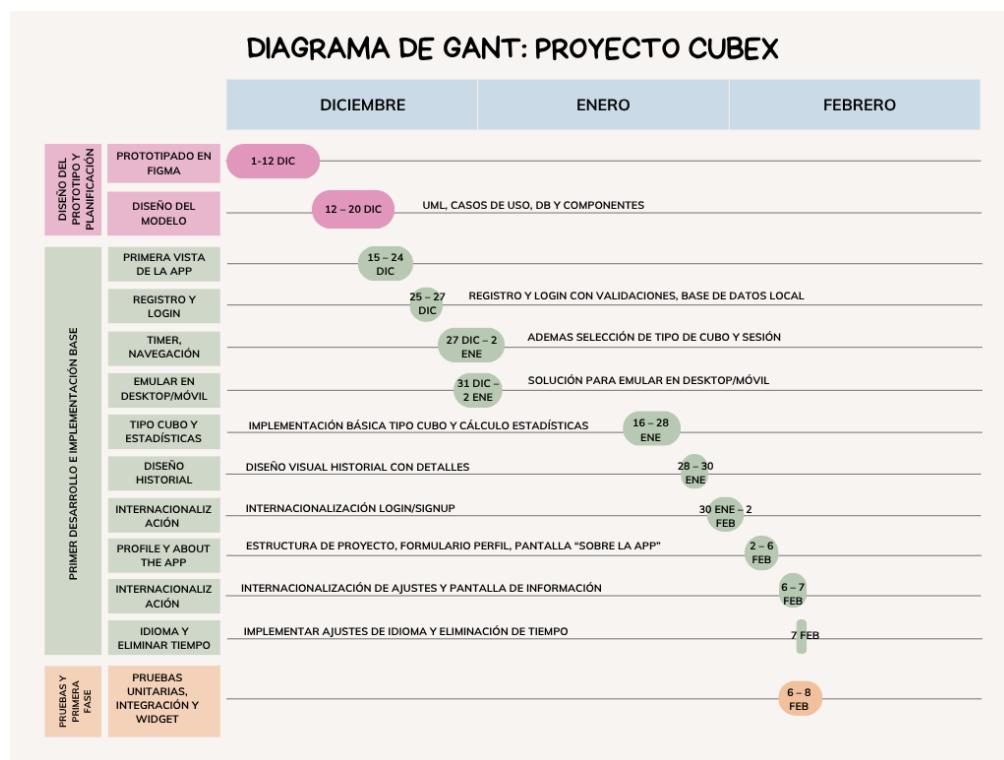
Objetivo principal: Finalizar las pruebas y preparar la aplicación para su lanzamiento.

Herramienta de planificación utilizada:

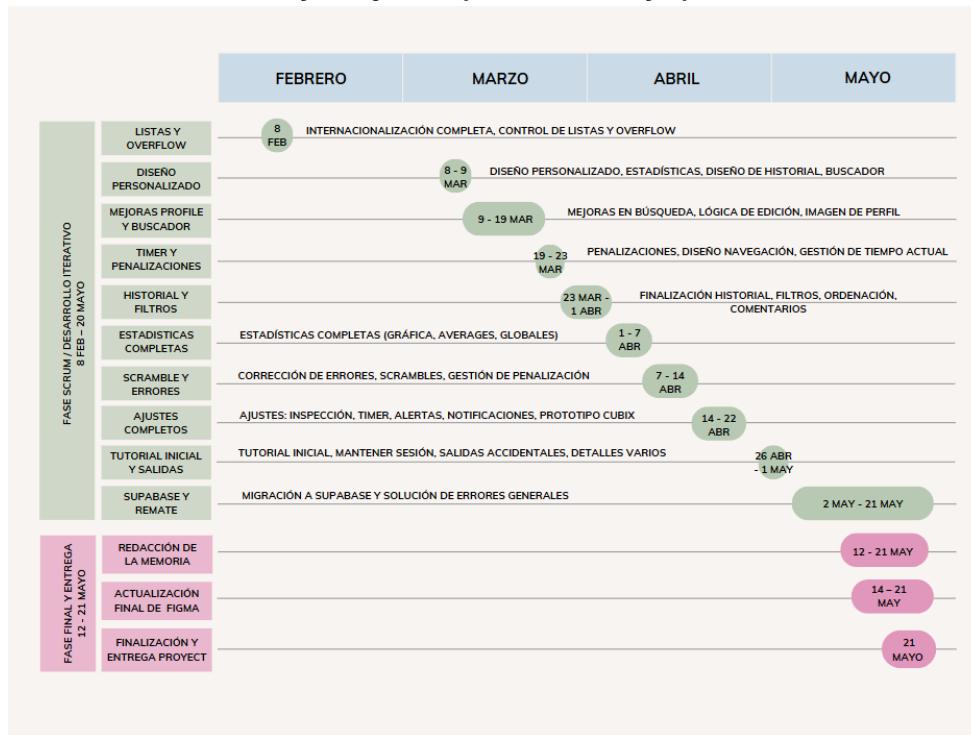
Para gestionar y planificar las tareas del proyecto, se utilizó un **backlog**. Este backlog permitió estructurar el flujo de trabajo mediante la creación de tareas, su priorización y asignación a las distintas iteraciones del proyecto. A lo largo de las fases de implementación, se fue añadiendo y actualizando las tareas en el backlog conforme avanzaba el desarrollo.

A continuación, se muestra el cronograma del proyecto en forma de diagrama de Gantt, dividido por las dos etapas:

Fase inicial (diciembre a febrero):



Fase de desarrollo y mejoras (marzo a mayo):



5.2 Presupuesto

5.2.1 Coste de desarrollo de código

El desarrollo total del proyecto ha requerido por ahora unas 400 horas de trabajo. Considerando un coste medio de programador junior en España, cuyo salario puede oscilar entre 20.000€ y 25.000€ anuales, se puede estimar un coste por hora de unos 12€/hora (bruto aproximado).

- **Horas dedicadas:** 300 horas picando código + 60 aprox entre prototipado y memoria + 40 aprox de documentación sobre los distintos temas.
- **Coste por hora:** 12€/hora.
- **Total estimado de desarrollo:** 4.800€.

5.2.2 Coste de software y herramientas de desarrollo

Se han utilizado herramientas de desarrollo principalmente de **distribución gratuita**, excepto el sistema operativo y Microsoft Word, que forman parte de un paquete de software adquirido previamente.

Herramienta o lenguaje	Tipo	Coste estimado	Licencia
Flutter + Dart	Framework y lenguaje principal	0€	Gratis
IntelliJ IDEA	IDE para el desarrollo	0€	Gratis
Supabase (plan gratuito)	Backend (autenticación y BD)	0€	Gratis
Figma (plan gratuito)	Diseño de interfaces	0€	Gratis
Microsoft Word	Redacción de memoria	Incluido en Windows	Pago
Git + GitHub (plan gratuito)	Control de versiones y gestión	0€	Gratis
Windows 10	Sistema operativo	100€	Pago

5.2.3 Coste del hardware

Para el desarrollo y pruebas se ha utilizado el siguiente equipo

Dispositivo	Descripción	Coste estimado
Ordenador portátil HP Omen	Desarrollo, pruebas y ejecución	1000€
Teléfono móvil (Android)	Pruebas en dispositivo real	200€ aprox

5.2.4 Coste de hosting

La aplicación utiliza **Supabase**, que dispone de un **plan gratuito** suficiente para las necesidades del proyecto (autenticación y base de datos).

Además, la documentación técnica del proyecto fue desplegada utilizando **GitHub Pages**, un servicio gratuito que permite publicar sitios web estáticos directamente desde un repositorio. Este despliegue incluye las páginas generadas por dart doc.

5.2.5 Coste materiales auxiliares

Se adquirió un soporte para teléfono con el fin de facilitar la visualización y pruebas de la aplicación móvil durante el desarrollo: 7€

5.2.6 Coste material promocional

- Camisetas con el logo y mascota del proyecto: 30 €
- Pegatinas de la app: 10 €
- Folletos explicativos: 15 €

Resumen de costes

Concepto	Coste estimado
Desarrollo del código	4.800€
Herramientas y software	100€
Hardware	1.200€
Hosting	0€
Coste material auxiliar	7€
Coste material promocional	55€
Total estimado	6.162€

6 Documentación Técnica: análisis, diseño, implementación y pruebas.

6.1 Especificación de requisitos

En este apartado se detallan los requisitos del sistema definidos durante el análisis del proyecto. Se distinguen entre requisitos funcionales y no funcionales.

6.1.1 Requisitos funcionales

Los requisitos funcionales definen las capacidades que proporciona mi aplicación en base a las necesidades del usuario. Estos son, como se trató en el punto 4.1 vision general del sistema:

- **Gestión y manejo de tiempos:** permite al usuario registrar, editar, eliminar y ver tiempos de resolución de cubos.
- **Gestión de penalizaciones:** el usuario puede añadir penalizaciones a sus tiempos, como "+2 segundos" o "DNF" (Did Not Finish).
- **Búsquedas filtradas:** se pueden realizar búsquedas y filtrados por tiempo registrado o comentarios asociados.

- **Gestión de tipos de cubo y sesiones:** la aplicación permite crear distintos tipos de cubo y agrupar los tiempos dentro de sesiones asociadas.
- **Generación automática de scrambles:** se generan scrambles aleatorios adecuados al tipo de cubo seleccionado por el usuario.
- **Generación y exportación de informes PDF:** el usuario puede exportar los tiempos de una sesión en un archivo PDF y compartirlo.
- **Autenticación de usuarios:** el sistema requiere el registro de usuario mediante correo electrónico, incluyendo verificación del mismo, así como acceso mediante login.
- **Gestión del perfil de usuario:** el usuario puede configurar su perfil, incluyendo nombre, imagen u otros datos.
- **Estadísticas visuales:** se presentan gráficas que reflejan el rendimiento del usuario, incluyendo estadísticas como mejor tiempo, media, distribución de penalizaciones, tiempo total de uso, entre otros.
- **Opciones de configuración avanzada:** la aplicación permite personalizar la experiencia del usuario mediante opciones como activar/desactivar sonidos, vibraciones, la visibilidad del cronómetro, el tiempo de inspección, etc.
- **Notificaciones:** la aplicación puede enviar notificaciones periódicas, como recordatorios diarios.

6.1.2 Requisitos no funcionales

Los requisitos no funcionales definen las cualidades y restricciones del sistema que no están directamente relacionadas con su funcionalidad, pero que afectan a su rendimiento, seguridad o experiencia de usuario. Se definen los siguientes:

- **Usabilidad:** interfaz amigable, clara y accesible para usuarios sin conocimientos técnicos.
- **Seguridad:** los datos del usuario están protegidos mediante autenticación segura y verificación por correo electrónico.
- **Portabilidad:** el uso de Flutter permite escalar la aplicación fácilmente a otras plataformas, como Windows, iOS y macOS.
- **Mantenibilidad:** el código está documentado para facilitar futuras ampliaciones o modificaciones.

- ❖ **Escalabilidad:** el diseño contempla la posibilidad de añadir nuevas funcionalidades sin comprometer la estabilidad del sistema actual.

6.2 Análisis del sistema

6.2.1 Gestión y manejo de tiempos

Esta funcionalidad permite al usuario añadir, modificar o eliminar tiempos registrados de resolución de cubos.

❖ Alta de tiempos

- Se registra automáticamente al detener el cronómetro.
- Campos:
 - Tiempo en segundos (obligatorio)
 - Fecha y hora de resolución (auto-generado)
 - Tipo de cubo (obligatorio)
 - Comentario del usuario (opcional)
 - Penalización (opcional: ninguna, +2, DNF)
 - ID de sesión (obligatorio)
- Validaciones:
 - Tiempo no puede ser mayor que 59 minutos 59.9 segundos.
 - Se requiere que haya asociado una sesión con un tipo de cubo existentes.

❖ Edición de tiempos

- Campos modificables:
 - Comentario
 - Penalización

❖ Eliminación de tiempos

- Requiere selección de un tiempo específico.
- Confirmación por diálogo emergente.

6.2.2 Gestión de penalizaciones

- ❖ Tipos disponibles:
 - Ninguna
 - +2 segundos
 - DNF (Did Not Finish)
- ❖ Aplicación desde el panel de detalles del tiempo y desde el cronometro despues de hacer un tiempo.
- ❖ Recalcula el valor final mostrado.

6.2.3 Búsquedas filtradas

- ❖ Filtros disponibles:
 - Por comentarios.
 - Por tiempo.

6.2.4 Gestión de tipos de cubo y sesiones

- ❖ **Tipos de cubo:**
 - Campos: nombre (obligatorio), descripción (opcional), idUser (obligatorio).
 - Validaciones: nombre e idUser único.
- ❖ **Sesión:**
 - Asociada a un tipo de cubo.
 - Campos: nombre, fecha de creación, id de tipo de cubo y de usuario.
 - Acciones: crear, mostrar, eliminar.
 - Validaciones: nombre de sesion, id de usuario y de tipo de cubo tienen que ser únicos, es decir, un usuario no puede tener dos sesiones con el mismo nombre de distinto tipo de cubo.

6.2.5 Generación de scrambles

- ❖ Automática al iniciar resolución.
- ❖ Adaptada al tipo de cubo seleccionado.
- ❖ Sigue los algoritmos oficiales WCA.

6.2.6 Generación de informes PDF

- ❖ Exporta los tiempos de una sesión.
- ❖ Contenido del PDF:
 - Lista de tiempos
 - Comentarios y penalizaciones
- ❖ Opciones:
 - Compartir por correo
 - Guardar en el dispositivo

6.2.7 Autenticación de usuarios

- ❖ Registro:
 - Campos: nombre (unico), correo electrónico (unico), contraseña (minímo 8 caracteres que contenga un número y un carácter especial)
 - Verificación vía correo obligatorio
- ❖ Login:
 - Campos: correo electrónico y contraseña
 - Solo usuarios verificados pueden acceder

6.2.8 Gestión del perfil de usuario

- ❖ Acciones permitidas:
 - Cambiar nombre.
 - Cambiar de contraseña.

- Cambiar imagen.
- Eliminar cuenta.
- ❖ Acceso desde el menú de configuración.
- ❖ Validaciones: si el usuario se va sin guardar, se muestra una alerta y si pulsa el botón de guardar tendrá que proporcionar su contraseña anterior.

6.2.9 Estadísticas visuales

❖ Por sesión:

- Mejor tiempo
- Tiempo promedio
- Ao5, Ao12, Ao50, Ao100 y AoTotal
- Evolución gráfica.
- Porcentaje de penalizaciones

❖ General:

- Tiempo total usando la app

6.2.10 Configuración avanzada

❖ Opciones configurables:

- Sonido, vibración o ambas a los 8 y 12 segundos de la inspección.
- Tiempo de inspección o cronómetro visible/oculto

❖ Guardadas localmente (SharedPreferences)

6.2.11 Notificaciones

- ❖ Notificaciones locales diarias y semanales.
- ❖ Configurables por el usuario

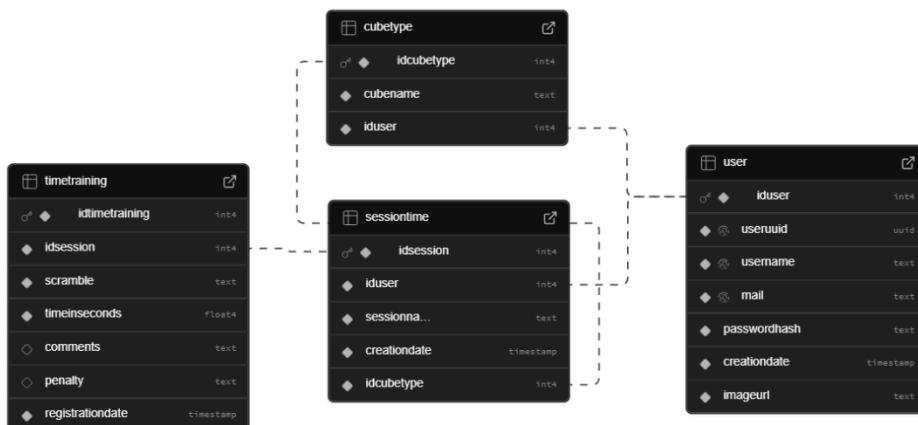
6.3 Diseño del sistema:

6.3.1 Diseño de la Base de Datos

Anteriormente la base de datos era local con SQQLite, pero luego de migro toda la gestión de la base de datos PostgreSQL, proporcionada por Supabase.

6.3.1.1 Modelo Entidad-Relacion (E/R)

Para el diseño de la base de datos del proyecto, se ha utilizado Supabase, que genera automáticamente un diagrama E/R a partir de las tablas creadas. Este diagrama permite visualizar cómo se relacionan los distintos elementos del proyecto: los usuarios, los tipos de cubos que pueden registrar, las sesiones que crean y los tiempos registrados en cada una:



6.3.1.2 Modelo Relacional

Las relaciones clave del modelo son las siguientes:

Relacion	Tipo de Relacion	Descripcion
User -> cubeType	1 : N	Un usuario puede definir múltiples tipos de cubo y cada tipo de cubo pertenece a un único usuario.
User -> sessionTime	1 : N	Un usuario puede tener múltiples sesiones, pero cada sesión está asociada a un único usuario.

CubeType -> sessionTime	1 : N	Un tipo de cubo puede estar asociado a varias sesiones, sin embargo, cada sesión solo puede tener un tipo de cubo.
SessionTime -> timeTraining	1 : N	Cada sesión puede contener múltiples tiempos de resoluciones registrados.

6.3.1.3 Diagrama de Tablas

Se presenta cada tabla detallada para cada entidad:

6.3.1.3.1 Tabla user

Campo	Tipo de dato	Restricción
idUser	SERIAL	PRIMARY KEY
userUUID	UUID	UNIQUE, NOT NULL
username	TEXT	UNIQUE, NOT NULL
mail	TEXT	UNIQUE, NOT NULL
passwordHash	TEXT	NOT NULL
creationDate	TIMESTAMP	NOT NULL
imageUrl	TEXT	NOT NULL

6.3.1.3.2 Tabla CubeType

Campo	Tipo de dato	Restricción
idCubeType	SERIAL	PRIMARY KEY
cubeName	TEXT	NOT NULL

idUser	INTEGER	Fk -> user(idUser), NOT NULL ON DELETE CASCADE
UNIQUE(idUser, cubeName)		

6.3.1.3.3 Tabla sessionTime

Campo	Tipo de dato	Restricción
idSession	SERIAL	PRIMARY KEY
idUser	INTEGER	FK -> user(idUser), NOT NULL
sessionName	TEXT	NOT NULL
creationDate	TIMESTAMP	NOT NULL
idCubeType	INTEGER	Fk -> CubeType(idCubeType), NOT NULL
UNIQUE(idUser, sessionName, idCubeType)		

6.3.1.3.4 Tabla timeTraining

Campo	Tipo de dato	Restricción
idTimeTraining	SERIAL	PRIMARY KEY
idSession	INTEGER	FK -> user(idUser), NOT NULL
scramble	TEXT	NOT NULL
timeInSeconds	REAL	NOT NULL
comments	TEXT	Opcional (NULL por defecto)
penalty	TEXT	CHECK('none', 'DNF', '+2') DEFAULT 'none'
registrationDate	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP, NOT NULL

6.3.1.4 Esquema de la base de datos

El esquema de la base de datos se ha creado utilizando SQL, donde se definen todas las tablas, claves primarias, foráneas y sus respectivas restricciones. Se muestra el script completo utilizado para crear la estructura en Supabase:

-- SE USA COMILLAS YA QUE ES UNA PALABRA RESERVADA

```
CREATE TABLE "user" (
```

```
    idUser SERIAL PRIMARY KEY,
```

```
    userUUID UUID UNIQUE NOT NULL,
```

```
    username TEXT NOT NULL UNIQUE, -- EL NOMBRE DEL USUARIO SERA  
UNICO Y DE LONGUITUD DE 12
```

```
    mail TEXT NOT NULL UNIQUE, -- EL MAIL SERA UNICO
```

```
    passwordHash TEXT NOT NULL,
```

```
    creationDate TIMESTAMP NOT NULL,
```

```
    imageUrl TEXT NOT NULL /* nota: por defecto tendra una imagen  
predeterminada */
```

```
);
```

```
CREATE TABLE cubeType (
```

```
    idCubeType SERIAL PRIMARY KEY,
```

```
    cubeName TEXT NOT NULL, -- EL NOMBRE DEL CUBO SERA UNICO
```

```
    -- CADA USUARIO TENDRA SUS TIPOS DE CUBOS
```

```
    idUser INTEGER NOT NULL REFERENCES "user"(idUser) ON DELETE  
CASCADE,
```

```
    -- ASEGURAR QUE UN USUARIO NO TENGA MAS DE UN TIPO DE CUBO  
CON EL MISMO NOMBRE
```

```
    UNIQUE (idUser, cubeName)
```

```
);
```

```

CREATE TABLE sessionTime (
    idSession serial PRIMARY KEY,
   idUser INTEGER NOT NULL REFERENCES "user"(idUser),
    sessionName TEXT NOT NULL,
    creationDate TIMESTAMP NOT NULL,
    idCubeType INTEGER NOT NULL REFERENCES cubeType(idCubeType),
    -- ASEGURAR QUE UN USUARIO NO PUEDA TENER DOS SESIONES CON EL
    MISMO NOMBRE DE DISTINTO TIPO DE CUBO
    UNIQUE (idUser, sessionName, idCubeType)
);

```

```

CREATE TABLE timeTraining (
    idTimeTraining SERIAL PRIMARY KEY,
    idSession INTEGER NOT NULL REFERENCES sessionTime(idSession),
    scramble TEXT NOT NULL,
    timeInSeconds REAL NOT NULL,
    comments TEXT DEFAULT NULL,
    -- RESTRINGIMOS LOS VALORES DEL CAMPO penalty (tipo ENUM)
    penalty TEXT CHECK(penalty IN ('none', 'DNF', '+2')) DEFAULT 'none',
    -- POR DEFECTO ES EL TIEMPO ACTUAL
    registrationDate TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);

```

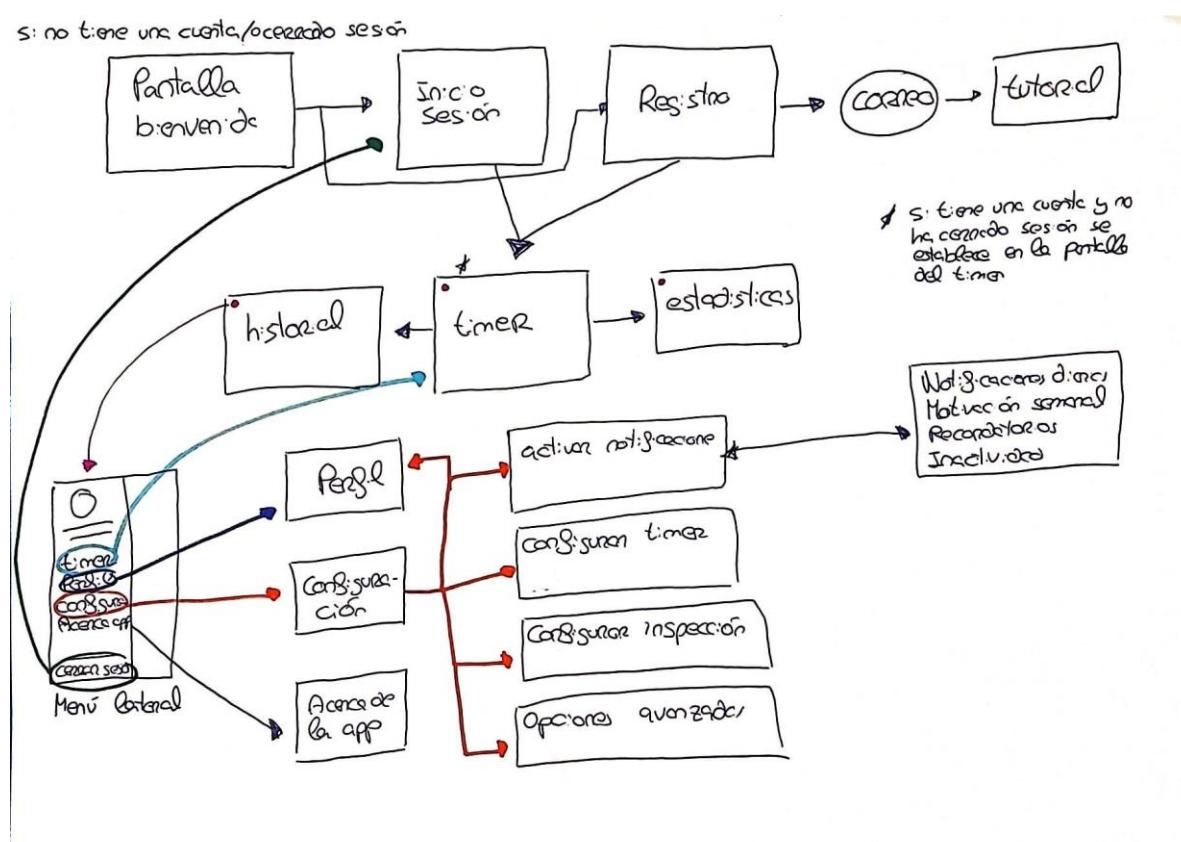
6.3.2 Diseño de la Interfaz de usuario.

La interfaz de usuario de la aplicación se ha diseñado en **Figma** con un enfoque centrado en la experiencia del usuario. La app presenta un diseño con tonos violetas, sombras suaves, botones con esquinas redondeadas y una navegación fluida entre pantallas.

El diseño se adapta tanto a dispositivos móviles como de escritorio, y sigue una estructura organizada en tres vistas principales: **Timer**, **Historial** y **Estadísticas**, todas accesibles desde la barra de navegación inferior. Además, se integra un menú lateral (Drawer) que permite acceder a secciones adicionales como ajustes, perfil y cierre de sesión.

6.3.2.1 Diagrama y flujo entre pantallas

La navegación de la aplicación se basa en el siguiente flujo:



6.3.2.2 Pantallas y componentes clave

6.3.2.2.1 Pantalla de bienvenida

La pantalla de bienvenida es lo primero que ve el usuario al abrir la app (si no tiene sesión iniciada).



Estructura visual

Está compuesta por cinco elementos principales colocados en vertical y centrados en pantalla:

1. **Fondo visual:** Se usa un fondo oscuro con formas en tonos violetas, que es el color principal de la app.
2. **Nombre de la app:** Aparece en la parte superior, en un tamaño grande.
3. **Logo de la app:** Justo debajo del nombre, se muestra el logo.
4. **Frase de bienvenida:** Debajo del logo hay una frase breve para dar la bienvenida: *“Take your skills to the next level.”*
 - **Botones de acción:** En la parte inferior hay dos botones grandes, tienen esquinas redondeadas, con sombras suaves y colores violetas.
 - “Iniciar sesión”
 - “Registrarse”

5. Comportamiento

- ❖ Si el usuario ya tiene sesión iniciada, esta pantalla no se muestra y va directo al cronómetro.
- ❖ Al tocar cualquiera de los botones, se redirige hacia la pantalla de login o de registro.
- ❖ Todo el contenido está adaptado al idioma del usuario, y si no ha elegido idioma previamente, se usa inglés por defecto.

6.3.2.2.2 Pantalla de inicio **de sesión**

La pantalla de login está diseñada para que el usuario pueda identificarse y entrar en la aplicación.



Estructura visual

La pantalla se compone de los siguientes elementos principales, distribuidos verticalmente:

1. **Título de la pantalla:** En la parte superior aparece el texto "Iniciar sesión" con una tipografía grande y centrado en pantalla.
2. **Formulario de acceso:** Se muestran dos campos de texto, ambos campos tienen validación, lo que impide continuar si están vacíos o contienen datos inválidos:
 - Nombre de usuario
 - Contraseña
3. **Icono para mostrar/ocultar contraseña:** Junto al campo de contraseña hay un pequeño ícono de ojo, que permite al usuario ver o esconder la contraseña mientras escribe.
4. **Botón de acceso:** Justo debajo del formulario hay un botón de "Iniciar sesión", que se activa solo si los campos se han llenado correctamente.
5. **Mensaje de error o validación:** Si el usuario introduce datos incorrectos, se muestra un mensaje en pantalla indicando que las credenciales no son válidas.
6. **Enlace a registro:** Debajo del botón hay un pequeño texto con enlace: "¿No tienes cuenta? Regístrate aquí". Al pulsarlo, se redirige a la pantalla de registro.

Comportamiento

- ❖ Al completar el formulario y pulsar el botón, se valida que las credenciales coincidan con un usuario existente.
- ❖ Si son correctas, se redirige directamente a la pantalla principal del cronómetro. Si son incorrectas o el usuario no existe, se muestra un mensaje informativo.

6.3.2.2.3 Pantalla de registro

La pantalla de registro tiene como funcionalidad principal encargarse del proceso de crear una cuenta. Su diseño sigue la misma estética que la pantalla de inicio de sesión.



Estructura visual

El formulario de registro está compuesto por varios elementos ordenados verticalmente con diseño centrado:

1. **Título de la pantalla:** En la parte superior se muestra el texto “Regístrate” en un tamaño grande y centrado.
2. **Campos del formulario**
 - Nombre de usuario
 - Correo electrónico
 - Contraseña
 - Confirmar contraseña
 - Todos los campos son obligatorios y están acompañados de iconos que indican su función.
3. **Validación visual**
 - Si el usuario deja un campo vacío o introduce un dato no válido, aparece un mensaje justo debajo del campo correspondiente.
 - La validación se aplica al pulsar el botón de registrar.
4. **Requisitos de la contraseña:** La contraseña debe tener al menos 8 caracteres, incluir un carácter especial y un número y coincidir con la confirmación. Si no se cumple alguna de estas condiciones, el sistema muestra un mensaje indicando qué falta.
5. **Mostrar/ocultar contraseñas:** Junto a los campos de contraseña hay un ícono para mostrar u ocultar el contenido.
6. **Sugerencia automática de nombre:** Si el nombre de usuario ya está registrado, se ofrece una sugerencia automática con una variante disponible añadiendo un número.

7. **Botón de registro:** Cuando todo es correcto, el botón “Regístrate” manda una alerta diciendo que confirme el correo que ha mandado.
8. **Enlace a login:** Al final del formulario aparece un pequeño texto: “¿Ya tienes cuenta? Inicia sesión”, que lleva de vuelta a la pantalla de login.
- 9. Confirmación por correo electrónico**

Una vez completado el registro, la interfaz informa al usuario de que se ha enviado un correo de confirmación. Ese mensaje contiene un botón para validar la cuenta.

Desde la propia app se ofrecen accesos rápidos para abrir directamente Gmail o la versión web del correo, lo que facilita el proceso.

Una vez el usuario confirma el correo, la app lo redirige automáticamente a la pantalla principal del cronómetro, mostrando un mensaje de éxito.

6.3.2.2.4 Pantalla del tutorial

Al abrir la aplicación por primera vez, el usuario es recibido por **Cubix**, la mascota oficial de CubeX, en una pantalla especialmente diseñada para crear una **primera impresión cercana**. Esta pantalla no solo presenta al personaje principal, sino que da inicio a un **tutorial guiado** que recorre las pantallas y funcionalidades principales de la app.

Presentación de Cubix y selección de idioma

En esta primera pantalla, Cubix se presenta como el **asistente personal del usuario** dentro de la aplicación. Se muestra una ilustración animada de Cubix, acompañada de un breve texto introductorio.

Antes de comenzar el recorrido, se ofrece la posibilidad de elegir el **idioma del tutorial** entre **español e inglés**. Esta elección mejora la accesibilidad desde el primer momento y adapta toda la experiencia guiada a la preferencia del usuario.



Inicio del recorrido guiado

Una vez seleccionado el idioma, Cubix pregunta al usuario si desea realizar el recorrido el usuario puede optar por:

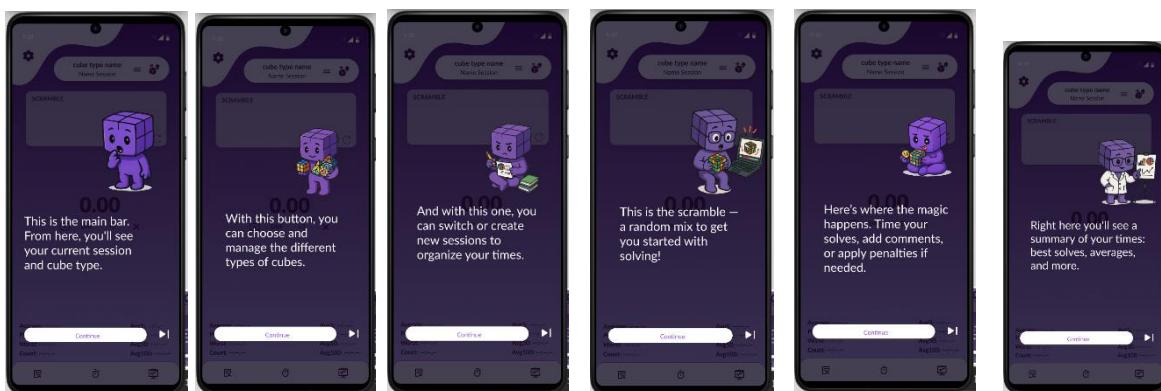
- ⦿ **Comenzar el tutorial.**
- ⦿ **Omitirlo y acceder directamente a la app.**

Esta decisión se presenta en forma de botones accesibles y bien diferenciados.

Recorrido por las principales pantallas

El tutorial está dividido en **cuatro secciones** principales, cada una correspondiente a una parte clave de la aplicación:

- **Pantalla del cronómetro:** La primera parada del recorrido es la pantalla principal de la app, donde se explica el funcionamiento del temporizador. Cubix va señalando cada componente mediante resaltes visuales:
 - **Barra superior:** muestra el tipo de cubo y la sesión activa.
 - **Gestión de tipos de cubo:** botón para cambiar, crear o eliminar cubos.
 - **Gestión de sesiones:** botón para crear o cambiar sesiones según el cubo activo.
 - **Scramble:** se muestra la secuencia generada aleatoriamente adaptada al cubo.
 - **Temporizador:** Cubix explica el sitio donde esta el cronometro así como donde añadir comentarios, aplicar penalizaciones o eliminar un tiempo.
 - **Estadísticas rápidas:** finalmente, se presentan los datos de rendimiento como PB, peor tiempo, conteo total y medias (Ao5, Ao12, Ao50, Ao100 y total)



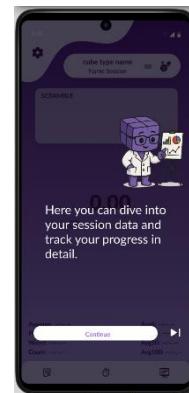
- **Pantalla de historial**

Después del cronómetro, Cubix guía al usuario a través del botón correspondiente en la barra inferior hacia la pantalla de **historial**: Aquí se explica que el usuario puede ver todos los tiempos registrados en la sesión activa y agregar, modificar o eliminarlos.



- **Pantalla de estadísticas**

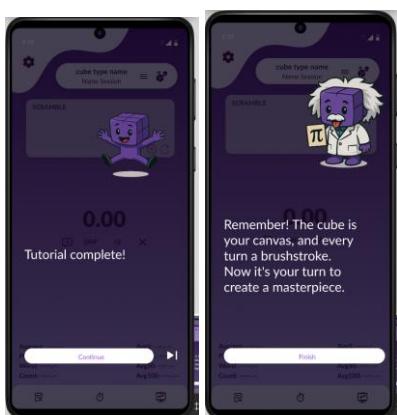
En la tercera sección del recorrido, el usuario es dirigido hacia la pantalla de **estadísticas**, también mediante la barra de navegación inferior: Cubix indica donde estan las **estadísticas** disponibles



Final del recorrido guiado

Una vez visitadas las tres pantallas principales, el recorrido concluye volviendo a la pantalla del cronómetro. Cubix se despide con una **frase**: “*¡Recuerda! El cubo es tu lienzo, y cada giro una pincelada. Ahora es tu turno de crear tu obra.*”

Se muestra un botón de "**Finalizar**", tras el cual se da paso al uso libre de la app con todas las funciones ya desbloqueadas y configuradas.



6.3.2.2.5 Pantalla del temporizador

La pantalla del temporizador constituye el **núcleo de la aplicación**, donde el usuario puede registrar, gestionar y analizar sus tiempos de resolución de cubos. Esta vista está diseñada para ser visualmente atractiva, centrada en la acción del cronómetro, e intuitiva tanto para principiantes como para usuarios avanzados.

Acceso y tutorial inicial

Si el usuario **accede por primera vez** tras el registro, se activa un **tutorial guiado por la mascota Cubix**, el cual explicamos anteriormente.

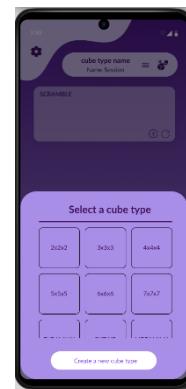
Al completar el registro, la app **establece automáticamente el tipo de cubo 3x3** y una sesión por defecto para que el usuario pueda comenzar a cronometrar sin configuraciones previas. Si el usuario ya ha iniciado sesión previamente, se carga **el último tipo de cubo y sesión activos**, respetando el estado anterior del usuario.



Gestión de tipos de cubo y sesiones

En la parte superior de la pantalla se encuentran dos accesos:

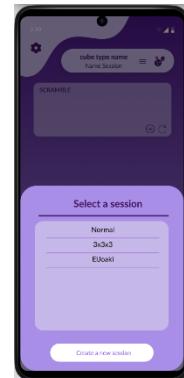
- **Selector de tipo de cubo** (ícono en la parte superior derecha): al pulsarlo, se muestra un menú desplegable con todos los tipos de cubo registrados. Desde ahí se puede:
 - **Cambiar** entre tipos de cubo.
 - **Crear uno nuevo** mediante un formulario con nombre personalizado.
 - **Eliminar** uno existente (manteniendo pulsado), siempre que no sea el único tipo de cubo registrado.



- **Gestión de sesiones:** junto al selector de cubos, el usuario puede gestionar sesiones asociadas al cubo activo:

- **Seleccionar** sesiones existentes.
- **Crear** nuevas sesiones (nombre personalizado).
- **Eliminar** sesiones manteniendo pulsado (con confirmación). No se puede eliminar la última sesión disponible.

Una vez seleccionado el tipo de cubo y la sesión, ambos se muestran siempre visibles en la interfaz como referencia.



Scramble

Inmediatamente debajo de estos selectores, se encuentra la **sección de scramble**:

- ❖ Se genera automáticamente cada vez que el usuario entra a la pantalla del timer o termina una resolución.
- ❖ Se adapta al tipo de cubo actual (por ejemplo, movimientos válidos distintos para 2x2, 3x3, Megaminx, etc.).
- ❖ El usuario puede:
 - **Generar uno nuevo** manualmente (ícono de recarga).
 - **Introducir uno personalizado**, mediante un formulario.
 - Si el campo está vacío, se lanza una alerta de error.

Cronómetro central

El elemento principal de la pantalla es el **cronómetro**, inicialmente marcado con 0.00, centrado visualmente. Su diseño destaca en tamaño y contraste para facilitar su lectura en todo momento. **Funcionamiento:**

- **Un toque simple** sobre el cronómetro inicia la pantalla de inspección (si está activada).
- **Mantener pulsado** inicia la cuenta atrás de inspección o directamente el cronómetro si no está activa.
- **Al soltar**, comienza la resolución.

- **Un toque final** detiene el cronómetro y redirige a la pantalla principal con el tiempo registrado.

Límites: El cronómetro tiene un límite máximo de **59 minutos y 59.9 segundos**, tras lo cual se detiene automáticamente.



Opciones tras una resolución

Una vez registrado un tiempo, se activan varios botones bajo el cronómetro:

- **Añadir comentario:** permite escribir una anotación sobre el tiempo. Si se añade, aparece un ícono visual que indica que ese tiempo tiene un comentario.
 - **Añadir penalización:** se puede aplicar +2 o DNF (Did Not Finish). No se pueden aplicar ambas. Si hay penalización y comentario, se muestran los dos iconos.
- **Eliminar tiempo:** permite borrar el último tiempo registrado (con confirmación visual).

Estadísticas rápidas

En la parte inferior de la pantalla, se muestran estadísticas de la sesión actual en tiempo real:

- ❖ **Mejor tiempo**
- ❖ **Peor tiempo**
- ❖ **Total de resoluciones**
- ❖ **Media general de la sesión**
- ❖ **Medias recientes:** Ao5, Ao12, Ao50, Ao100

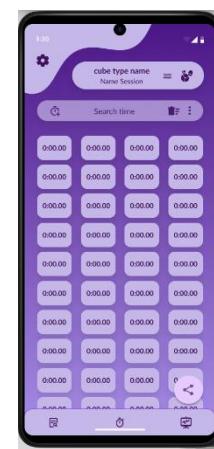
💡 Consideraciones especiales:

Los tiempos **marcados como DNF no se incluyen** en los cálculos de media ni en el PB. Si hay menos de X tiempos válidos (por ejemplo, para Ao12), se muestra "--:--.--".

Si hay múltiples DNF en las últimas X resoluciones, se omiten y se buscan tiempos anteriores para completar la media. Si no se puede completar, se considera la media como DNF.

6.3.2.2.6 Pantalla del historial

La pantalla de historial permite al usuario **consultar y gestionar todos los tiempos registrados** en una sesión, asociada a un tipo de cubo concreto. Está diseñada para ofrecer herramientas de búsqueda avanzada y opciones de edición y exportación.



Selección de tipo de cubo y sesión

En la parte superior de la pantalla se encuentra la **barra de selección** con menús desplegables para:

- Elegir el **tipo de cubo** (2x2, 3x3, Skewb, etc.).
- Seleccionar la **sesión** correspondiente a ese tipo de cubo.

Este componente de selección es el mismo que se utiliza en la pantalla del temporizador. Al seleccionar un tipo y sesión, la lista de tiempos registrados se actualiza automáticamente.

Añadir tiempo manualmente

A la izquierda de la barra de búsqueda hay un botón que permite al usuario **añadir un tiempo manual**.

Al pulsarlo, se abre un formulario emergente donde debe introducir:

- 💡 El **scramble** utilizado.
- 💡 El **tiempo de resolución**.

Una vez confirmado, el nuevo tiempo se inserta directamente en la lista correspondiente.

Búsqueda de tiempos

En el centro de la interfaz, la **barra de búsqueda** permite filtrar por tiempo o por comentario asociado al tiempo. Este sistema de filtrado se actualiza en tiempo real a medida que el usuario escribe permitiendo encontrar rápidamente registros específicos.

Opciones avanzadas

A la derecha de la barra de búsqueda se encuentran dos botones:

- ❖ **Eliminar todos los tiempos:** Representado por un ícono de papelera, permite borrar **todos los tiempos de la sesión actual**. Al pulsarlo, aparece una **alerta de confirmación** y si se confirma, la lista queda vacía y se actualiza automáticamente.
- ❖ **Menú de filtros adicionales** (ícono de tres puntos verticales): Al pulsarlo, se abre un cuadro de diálogo con opciones para ordenar la lista:
 - Por **fecha** (más reciente o antiguo)
 - Por **tiempo** (de menor a mayor o viceversa)
 - Botón de **Cerrar** para salir del diálogo

Visualización y gestión de tiempos individuales

Cada tiempo registrado aparece en un contenedor que muestra:

- ❖ El valor del tiempo.
- ❖ La penalización aplicada (si existe).
- ❖ El scramble utilizado y el comentario asociado a ese tiempo.
- ❖ La fecha de cuando se realizó.



Si el usuario pulsa sobre un tiempo específico, se abre un cuadro de detalles donde puede:

- ❖ **Eliminar el tiempo** (botón de papelera en la esquina superior).
- ❖ **Copiar el scramble** (botón con ícono de copiar junto al scramble).

Ambas acciones muestran un mensaje de éxito al completarse, y la lista se actualiza automáticamente.

Compartir tiempos de la sesión

En la esquina inferior derecha de la pantalla se encuentra un **botón flotante de compartir**, representado por un ícono de compartir. Al pulsarlo, se abren dos opciones:

1. **Descargar PDF:** Se genera un archivo PDF con todos los tiempos de la sesión, mostrando:
 - Tipo de cubo y nombre de la sesión.
 - Fecha de creación.
 - Tabla con columnas de tiempo, penalización y scramble.
2. **Compartir PDF:** Abre el menú del sistema para compartir el archivo generado con otras aplicaciones.

6.3.2.2.7 Pantalla de las estadísticas

La pantalla de estadísticas permite al usuario analizar su rendimiento con detalle, obteniendo una visión de su evolución y progresos en la resolución de cubos de Rubik.

Estructura visual general

La pantalla está dividida en tres bloques principales:

1. Estadísticas generales
2. Tabla de promedios
3. Gráfico de evolución



1. Selección de tipo de cubo y sesión

En la parte superior se encuentra el **selector de tipo de cubo y sesión**, que permite al usuario cambiar entre distintas combinaciones.

Este componente se mantiene visible de forma consistente con otras pantallas como el Timer o el Historial.

2. Estadísticas generales

El primer bloque presenta una **vista resumida de los datos más importantes** de la sesión seleccionada.

Se utilizan contenedores con estilo para mostrar:

- ❖ **PB (Mejor tiempo personal)**
- ❖ **Peor tiempo**
- ❖ **Cantidad total de resoluciones**
- ❖ **Número de penalizaciones:**
 - DNF (Did Not Finish)
 - +2 segundos
- ❖ **Porcentaje de penalizaciones sobre el total**
- ❖ **Tiempo total de uso de la aplicación**

3. Análisis de promedios (Average Analysis)

A continuación, se muestra una **tabla de promedios**, donde el usuario puede analizar su rendimiento en diferentes tramos:

- ❖ **Ao5** (media de 5)
- ❖ **Ao12**
- ❖ **Ao50**
- ❖ **Ao100**
- ❖ **AoTotal** (media de todos los tiempos de la sesión)

Para cada tipo de media se muestran tres métricas:

- ❖ **Mejor media**
- ❖ **Peor media registrada**
- ❖ **Media actual**

Estas medias se calculan siguiendo el estándar: eliminando el mejor y peor tiempo del bloque y haciendo la media del resto.

4. Gráfico de evolución de tiempos

En la parte inferior, se muestra un **gráfico interactivo** que permite visualizar los tiempos a lo largo del tiempo, con diferentes opciones de personalización:

❖ **Escalas temporales:**

- Diaria
- Mensual
- Anual



❖ **Tipos de línea:**

- **Línea continua:** representa todos los tiempos registrados.
- **Línea discontinua:** representa los mejores tiempos alcanzados en cada periodo.

❖ **Opciones visuales:**

- Cambiar entre orientación horizontal o vertical.
- Mostrar/ocultar los valores numéricos sobre los puntos.

❖ **Interacción:** Al tocar un punto de la gráfica, aparece un mensaje o tooltip con el tiempo correspondiente y su fecha.

El gráfico se genera automáticamente si existen datos registrados en la sesión seleccionada. En caso contrario, se muestra un mensaje informativo.

6.3.2.2.8 Navegación

La navegación dentro de la app se ha diseñado para ser intuitiva y rápida. Se utilizan dos principales funciones de navegación: **una barra inferior fija** para las funciones principales y **un Drawer lateral** para opciones complementarias.

Navegación inferior

En la parte inferior de la pantalla se encuentra una **barra de navegación permanente**, que permite cambiar entre las tres secciones principales de la aplicación:

- ❖ Temporizador
- ❖ Historial
- ❖ Estadísticas

Esta barra se mantiene visible en todo momento y su diseño sigue la línea estética de la app:

- ❖ El botón activo se **resalta en color blanco** y se encierra en un **círculo** que lo diferencia visualmente del resto.
- ❖ Los botones inactivos permanecen en **color morado oscuro**.
- ❖ Los iconos son simples y fácilmente reconocibles, lo que facilita la comprensión rápida incluso para nuevos usuarios.

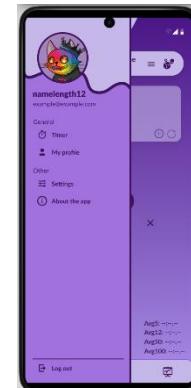
Además de los botones, la navegación también admite **gestos horizontales** (swipe) para desplazarse entre estas tres secciones deslizando hacia la izquierda o derecha. Esta funcionalidad es especialmente útil en dispositivos móviles mejorando la fluidez a la hora de usarlo.

Navegación mediante Drawer (menú lateral)

La aplicación también cuenta con un **Drawer lateral** (menú desplegable), accesible desde un ícono de ajustes ubicado en la esquina superior izquierda de la pantalla.

Este menú incluye:

- ❖ **Foto de perfil, nombre de usuario y correo:** mostrados en la parte superior del menú para identificar rápidamente al usuario activo.
- ❖ **Opciones disponibles:**
 - **Temporizador:** vuelve a la pantalla principal.
 - **Mi perfil:** permite acceder a la gestión de la cuenta.
 - **Configuración:** ajustes generales de la aplicación.
 - **Sobre la app:** información del proyecto y acceso al repositorio.
 - **Cerrar sesión:** finaliza la sesión actual y vuelve a la pantalla de inicio de sesión.



El Drawer aparece mediante una animación lateral desde la izquierda y se puede cerrar tocando fuera del menu.

Comportamiento general

- ⦿ El sistema de navegación se adapta correctamente a distintos tamaños de pantalla.
- ⦿ Al cambiar de sección, se actualiza el contenido principal, manteniendo los estados importantes (como el tipo de cubo actual o las selecciones activas).

6.3.2.2.9 Pantalla del Acerca de la aplicación

La pantalla "Sobre la aplicación" ofrece al usuario información general y de contacto relacionada con el proyecto. Su diseño es simple, donde el usuario puede identificar fácilmente el desarrollo y acceder al código fuente si lo desea.



Estructura visual

El contenido está centrado verticalmente y distribuido en tres secciones principales:

1. **Título y descripción general:** En la parte superior aparece el texto “Sobre la aplicación” con un diseño igual al de las otras opciones, con un fondo degradado de distintos morados. Justo debajo, se muestra un pequeño párrafo explicativo donde se indica de que va la app.
2. **Enlace al código fuente:** Se muestra un ícono para abrir el código fuente. Al pulsarlo, el usuario es redirigido al repositorio público del proyecto, alojado en GitHub.
3. **Nombre de usuario de GitHub:** En la parte inferior de la pantalla, se muestra el nombre del desarrollador en GitHub. Este texto actúa también como enlace: al pulsarlo, se abre el perfil del desarrollador directamente en el navegador.

Comportamiento

- Al pulsar cualquiera de los dos enlaces, se abre el navegador del sistema y se dirige a la página correspondiente (el repositorio o el perfil del desarrollador).
- El botón de retroceso o la barra de navegación permiten volver a la pantalla anterior fácilmente.

6.3.2.2.10 Pantalla de editar los datos del perfil

La pantalla de perfil permite al usuario gestionar los datos asociados a la cuenta del usuario.



Estructura visual

La pantalla está organizada en secciones verticales, permitiendo editar múltiples campos:

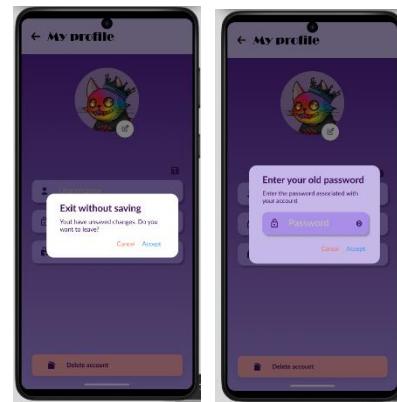
1. **Foto de perfil:** En la parte superior se muestra la imagen de perfil del usuario en forma circular. Al pulsar sobre ella, se abre un diálogo para seleccionar una nueva imagen:
 - En ordenadores, se abre el explorador de archivos.
 - En móviles, se accede a la galería del sistema.
 - La imagen nueva se carga de forma inmediata en la interfaz, pero no se guarda hasta confirmar los cambios.
2. **Campos de edición**
 - **Nombre de usuario.**
 - **Contraseña:**
 - Se muestra un formulario con dos campos: nueva contraseña y confirmación de la nueva contraseña.
 - Los campos de contraseña incluyen iconos para mostrar/ocultar el texto.
 - Las contraseñas deben cumplir los requisitos mínimos (longitud, caracteres especiales, coincidencia).

2. Campos de edición

- **Nombre de usuario.**
- **Contraseña:**
 - Se muestra un formulario con dos campos: nueva contraseña y confirmación de la nueva contraseña.
 - Los campos de contraseña incluyen iconos para mostrar/ocultar el texto.
 - Las contraseñas deben cumplir los requisitos mínimos (longitud, caracteres especiales, coincidencia).

3. Botones de acción

- **Guardar cambios:** Aparece habilitado solo si se detectan cambios. Al pulsarlo, se solicita la contraseña actual para validar la identidad de usuario antes de actualizar.
- **Cancelar:** Si el usuario intenta salir o retroceder sin guardar, se muestra una alerta informando que hay cambios pendientes. Permite confirmar si desea salir sin guardar o quedarse en la pantalla.



4. **Eliminar cuenta:** Al final de la pantalla, se muestra un botón con estilo diferenciado (en rojo y con icono de advertencia). Esta opción está protegida con una alerta de confirmación.

5. Comportamiento general

- ✖ Los cambios se guardan solo si todos los campos son válidos.
- ✖ El usuario puede cambiar nombre, imagen y contraseña en una misma acción.
- ✖ La validación se realiza en tiempo real, y se informa al usuario si hay errores en los campos antes de guardar.

6.3.2.2.11 Pantalla de los ajustes

La pantalla de ajustes está diseñada para que el usuario pueda personalizar su experiencia con la aplicación, desde aspectos visuales y de idioma, hasta funcionalidades avanzadas del cronómetro y del sistema de notificaciones. Su estructura esta organizada en diferentes secciones.

Estructura visual general

La pantalla se organiza en una lista vertical de secciones, cada una con su propio título y opciones agrupadas.

Cada opción está representada por una fila con texto, ícono y un botón.



6.3.2.2.11.1 Idioma

Al pulsar la opción de idioma, se despliega un **diálogo de selección** con los dos idiomas disponibles: **Español e Inglés**.

Una vez seleccionado, la interfaz se **actualiza automáticamente**.



6.3.2.2.11.2 Perfil y cuenta

Se incluye un acceso directo a la pantalla de "**Mi perfil**" desde esta sección.

El botón abre directamente la vista donde se puede modificar nombre, imagen y contraseña.

6.3.2.2.11.3 Notificaciones

Esta sección permite gestionar todas las notificaciones integradas en la app.

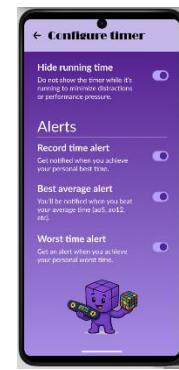
- ❖ Si las notificaciones aún no tienen permisos, se muestra una **alerta con opción de ir a los ajustes del sistema**.
- ❖ Una vez activadas, la app envía una **notificación de prueba** confirmando que se han activado correctamente.
- ❖ El diseño incluye interruptores para activar o desactivar individualmente los tipos de notificación:
 - Notificación diaria (09:00)
 - Mensaje motivacional semanal (lunes, 09:00)
 - Recordatorio para entrenar (18:00)
 - Notificación por inactividad (20:00)
- ❖ El **botón de reinicio rápido** activa por defecto solo:
 - Notificación diaria
 - Notificación de nuevo récord
 - Notificación de inactividad



6.3.2.2.11.4 Configuración del cronómetro

Permite modificar el comportamiento visual del temporizador durante la resolución.

- ❖ **Mostrar/Ocultar tiempo durante la resolución:** útil para evitar distracciones.
- ❖ **Alertas de rendimiento:** opciones para notificar si se ha superado un PB, un peor tiempo o la mejor media (actualmente en desarrollo).



6.3.2.2.11.5 Configuración de inspección

Desde aquí se gestiona la lógica del tiempo de inspección previa a la resolución.

- ❖ **Habilitar o deshabilitar la inspección.**
- ❖ Si está habilitada, aparece un campo para definir el **número de segundos** (por defecto 15), con validaciones:
 - Campo obligatorio
 - Solo números
 - Máximo: 59 segundos
- ❖ Si se desactiva la inspección, este campo queda bloqueado visualmente.



6.3.2.2.11.6 Opciones avanzadas

Opciones adicionales relacionadas con la experiencia durante la inspección:

- ❖ **Activar alertas de inspección:** permiten recibir alertas durante los segundos 8 y 12 de la cuenta regresiva.
- ❖ **Tipo de alerta** (se habilita solo si las alertas están activadas):
 - Vibración
 - Sonido (en función del idioma actual)
 - Ambos



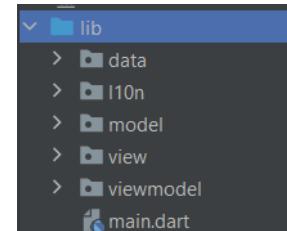
Los controles se presentan como interruptores o menús desplegables según el tipo de opción.

Comportamiento general

- Las configuraciones se guardan automáticamente tras cada cambio.
- La interfaz se adapta correctamente tanto a escritorio como a móviles, con scroll vertical y separación entre secciones.

6.3.3 Diseño de la Aplicación.

El diseño de la aplicación se ha utilizado la arquitectura **MVVM (Model-View-ViewModel)**, que se adapta a Flutter y permite una separación clara.



6.3.3.1 Arquitectura general

La estructura del proyecto se organiza en varias carpetas que separan la lógica de presentación (pantallas y widgets), la lógica y el modelo de datos (entidades). También se dispone de componentes reutilizables y archivos de configuración general como la internacionalización.

6.3.3.2 Configuración del entorno de desarrollo

Se instaló y configuró el entorno necesario con Flutter + Dart, y se utilizó IntelliJ IDEA como entorno de desarrollo principal. Se conectó el proyecto a un repositorio Git, gestionado desde GitHub, utilizando ramas y GitHub Projects para la planificación de tareas.

6.3.3.3 Configuración de Supabase y Deep Linking

La aplicación utiliza **Supabase** como backend completo. Esta herramienta proporciona funcionalidades de autenticación, base de datos en tiempo real, almacenamiento de archivos (por ejemplo, imágenes de perfil) y funciones de servidor.

Uno de los aspectos clave en el diseño ha sido configurar correctamente el **flujo de verificación de cuenta mediante correo electrónico**. Por defecto, Supabase envía al usuario un enlace de verificación tras el registro. Ese enlace incluye una redirección a una URL que se debe definir previamente en el panel de administración de Supabase.

Dado que la aplicación está pensada para funcionar en dispositivos móviles (Android e iOS), ha sido necesario configurar el sistema de **Deep Linking**, de modo que, tras confirmar el correo, el enlace redirija a la propia aplicación en lugar de a una URL web (como localhost).

Configuración realizada:

En el panel de Supabase, en el apartado Authentication > URL Configuration, se ha definido una URL personalizada del tipo:

[myapp://login-callback](#)

Esta URL es la encargada de abrir la aplicación Flutter cuando el usuario confirma su cuenta desde el correo electrónico.

- En **Android**, se ha añadido un filtro intent-filter en el archivo `AndroidManifest.xml`, de la siguiente forma:

```
<intent-filter>

    <action android:name="android.intent.action.VIEW"/>

    <category android:name="android.intent.category.DEFAULT"/>

    <category android:name="android.intent.category.BROWSABLE"/>

    <data android:scheme="myapp" android:host="login-callback"/>

</intent-filter>
```

- En **iOS y macOS**, se ha configurado el archivo `ios/Runner/Info.plist` para declarar el esquema personalizado:

```
<key>CFBundleURLTypes</key>
```

```
<array>
    <dict>
        <key>CFBundleTypeRole</key>
        <string>Editor</string>
        <key>CFBundleURLSchemes</key>
        <array>
            <string>myapp</string>
        </array>
    </dict>
</array>
```

De esta forma, cuando un usuario se registra, la aplicación muestra un mensaje indicando que debe revisar su bandeja de entrada para confirmar su correo. Una vez hecho esto, al pulsar en el enlace, se redirige automáticamente a la aplicación móvil mediante el esquema definido (myapp://login-callback), iniciando sesión de forma automática y redirigiendo al usuario a la pantalla principal: el **temporizador**.

El sistema de Deep links en Windows es mucho más complejo, por lo que no se pudo implementar.

Se integró:

- ❖ **Autenticación de usuarios** (registro, verificación de email).
- ❖ **Base de datos PostgreSQL** para almacenar usuarios, tiempos, tipos de cubos y sesiones.
- ❖ **Storage de imágenes** para los avatares de usuario.
- ❖ **Deep Links** para el proceso de verificación por email en apps móviles, configurando:
 - URLs de redirección en Supabase.
 - AndroidManifest.xml para Android.
 - Info.plist para iOS y macOs.

6.3.3.4 Desarrollo del sistema de autenticación

Se implementaron formularios con validaciones para el registro e inicio de sesión. El sistema incluye verificación por correo electrónico y hasta que no se confirma no redirigirá a la página principal.

6.3.3.5 Desarrollo del temporizador (núcleo de la aplicación)

Se implementó un sistema de gestión de tipos de cubos y sesiones. Cada tipo de cubo puede tener múltiples sesiones, y dentro de cada sesión, los usuarios pueden registrar tiempos, añadir comentarios, aplicar penalizaciones (DNF, +2) y ver estadísticas.

6.3.3.6 Implementación del sistema de scrambles

Se programó la generación automática de scrambles para cada tipo de cubo, permitiendo al usuario resetear o crear uno nuevo antes de comenzar cada resolución.

6.3.3.7 Gestión del historial de tiempos

Se creó una pantalla para consultar todos los tiempos registrados, permitiendo búsquedas, filtros, ordenación, eliminación individual o total y exportación en formato PDF. Además de poder ver en detalle el tiempo y modificarlo según las preferencias del usuario.

6.3.3.8 Implementación del módulo de estadísticas

Se desarrolló una sección que ofrece al usuario:

- ✿ Análisis de medias (ao5, ao12...).
- ✿ Estadísticas generales: Mejor y peor tiempo, cantidad de penalizaciones, además de un sistema para visualizar el tiempo de uso total de la app, entre otras.
- ✿ Gráficas interactivas (por día, mes o año) con filtros personalizables.

6.3.3.9 Personalización de la experiencia del usuario

Desde el apartado de ajustes, el usuario puede configurar:

- Idioma de la app (internacionalización).
- Comportamiento del temporizador.
- Tiempo de inspección (con validaciones).
- Alertas en momentos clave (8 y 12 segundos).
- Permisos y preferencias de notificaciones.

6.3.3.10 Gestión del perfil de usuario

Se implementó una sección para modificar la imagen, nombre y contraseña del usuario, con validaciones y protección mediante contraseña actual. Las imágenes se almacenan en Supabase gracias a su sistema de Storage.

6.3.3.11 Implementación de notificaciones programadas

Se añadió un sistema de notificaciones locales configurables, con distintas opciones: diarias, motivacionales semanales, recordatorios y notificaciones por inactividad.

6.3.3.12 Incorporación de asistente y tutorial inicial

Se diseñó un recorrido guiado con la mascota Cubix para explicar las funciones principales al nuevo usuario tras su primer acceso, con selección de idioma.

6.4 Implementación:

6.4.1 Entorno de desarrollo.

Herramientas utilizadas exclusivamente en la programación:

- **Dart:** Lenguaje de programación principal utilizado para desarrollar la lógica y funcionalidades de la aplicación móvil.
- **Flutter:** Framework para el desarrollo multiplataforma, permitiendo la creación de la aplicación tanto para dispositivos Android como iOS.
- **IntelliJ IDEA:** IDE utilizado para escribir y depurar el código en Dart y Flutter.
- **Supabase:** Herramienta utilizada para gestionar la base de datos y autenticación de usuarios, que facilita la creación de un backend escalable y accesible.
- **Git:** Sistema de control de versiones utilizado para gestionar el código fuente, facilitando el seguimiento de cambios.
- **GitHub:** Plataforma para alojar los repositorios de código.

6.4.2 Estructura del código.

Librerías descargadas:

A continuación, se detallan las librerías utilizadas en el proyecto para las distintas funcionalidades:

- **android_intent_plus:** Para interactuar con otras aplicaciones del sistema Android, como abrir el Gmail.
- **supabase_flutter:** Para integrar la base de datos y la autenticación de usuarios en la aplicación mediante Supabase.
- **flutter_dotenv:** Para manejar variables de entorno de manera segura, como las claves API de Supabase.
- **image_cropper:** Para permitir la edición y recorte de imágenes dentro de la aplicación.
- **animated_text_kit:** Para crear animaciones de texto personalizadas.
- **vibration:** Para proporcionar efectos de vibración en la aplicación.
- **audioplayers:** Para la reproducción de audio dentro de la app.

- ❖ **timezone**: Para gestionar y convertir tiempos de zonas horarias.
- ❖ **permission_handler**: Para gestionar permisos de acceso a funciones como a los ajustes.
- ❖ **flutter_local_notifications**: Para implementar notificaciones locales en la aplicación.
- ❖ **syncfusion_flutter_charts**: Para generar gráficos interactivos dentro de la app.
- ❖ **share_plus**: Para compartir contenido desde la aplicación a otras aplicaciones.
- ❖ **pdf**: Para generar archivos PDF.
- ❖ **printing**: Para imprimir el contenido generado en la aplicación.
- ❖ **dropdown_button2**: Para agregar menús desplegables personalizados.
- ❖ **flutter_smart_dialog**: Para mostrar diálogos interactivos y mensajes emergentes.
- ❖ **flutter_speed_dial**: Para implementar botones flotantes de acción rápida, utilizado para el botón de compartir o descargar el pdf con los tiempos.
- ❖ **popup_menu** y **custom_pop_up_menu**: Para implementar menús emergentes personalizados.
- ❖ **popover**: Para mostrar menús tipo popover.
- ❖ **curved_navigation_bar**: Para crear barras de navegación con bordes curvados.
- ❖ **stroke_text**: Para aplicar efectos de texto.
- ❖ **sqflite**: Para gestionar bases de datos SQLite en la aplicación.
- ❖ **sqflite_common_ffi**: Para la integración con SQLite en plataformas de escritorio.
- ❖ **path**: Para trabajar con rutas de archivos en el sistema.
- ❖ **crypto**: Para la encriptación y manejo seguro de datos.
- ❖ **logger**: Para facilitar la depuración con un sistema de registro.
- ❖ **provider**: Para gestionar el estado de la aplicación.
- ❖ **flutter_markdown**: Para mostrar contenido en formato Markdown.

- **url_launcher**: Para abrir URLs externas desde la app.
- **image_picker**: Para seleccionar imágenes desde la galería o la cámara.
- **shared_preferences**: Para almacenar datos pequeños localmente.
- **flutter_localizations**: Para gestionar la localización y traducción de la app.
- **intl**: Para la internacionalización y la gestión de formatos de fechas y números.
- **flutter_launcher_icons**: Para personalizar los iconos de la aplicación.
- **flutter_test** y **integration_test**: Para realizar pruebas unitarias e integración de la aplicación.

librería de funciones desarrolladas por el usuario:

En este proyecto, no se ha desarrollado una librería de funciones personalizadas. Se han utilizado principalmente librerías externas para aprovechar funcionalidades específicas que facilitan el desarrollo, como la autenticación, el manejo de base de datos y la generación de gráficos, entre otras.

El código es orientado a objetos, orientado a eventos, o estructurado/procedimental:

El código de este proyecto es principalmente **orientado a objetos**, siguiendo la arquitectura **MVVM (Modelo-Vista-ViewModel)**, que es el común en aplicaciones con Flutter. La aplicación está estructurada de tal forma que se divide entre el **Modelo** (gestión de datos), la **Vista** (interfaz de usuario) y el **ViewModel** (lógica de la vista).

6.4.3 Cuestiones de diseño e implementación reseñables.

6.4.3.1 Gestión de notificaciones locales

La aplicación utiliza notificaciones locales para recordar al usuario actividades importantes como **entrenamientos**, **motivación** semanal, y **recordatorios** por inactividad. Este sistema está gestionado por varias clases. A continuación, se detalla cómo se gestiona esta funcionalidad.



Inicialización del sistema de notificaciones

La clase *NotificationService* gestiona la parte principal de notificaciones. Su método *initNotification()* inicializa el plugin y solicita los permisos.

Este método se llama al iniciar la aplicación. Si el usuario no ha concedido permisos, se le solicitarán automáticamente.

```
static Future<bool> initNotification() async {
    // SI NO ES ANDROID NI IOS, NO HACE NADA POR AHORA
    if (!Platform.isAndroid && !Platform.isIOS) {
        return false;
    }

    // CONFIGURACION PARA ANDROID
    const AndroidInitializationSettings initSettingsAndroid =
        AndroidInitializationSettings('@mipmap/ic_launcher');

    // CONFIGURACION PARA IOS
    const DarwinInitializationSettings initSettingsIOS =
        DarwinInitializationSettings(
            requestAlertPermission: true,
            requestBadgePermission: true,
            requestSoundPermission: true,
        );

    // CONFIGURACION GENERAL DE INICIALIZACION
    const InitializationSettings initSettings = InitializationSettings(
        android: initSettingsAndroid,
        ios: initSettingsIOS,
    );

    // INICIALIZAMOS EL PLUGIN
    await flutterLocalNotificationsPlugin.initialize(initSettings);

    // LUEGO PEDIMOS LOS PERMISOS
    return await requestNotificationPermissions();
}
```

Solicitud y comprobación de permisos

Se utiliza el paquete *permission_handler* para comprobar si el usuario ha concedido permisos de notificación. Esto se hace tanto al solicitar como al mostrar o programar notificaciones:

```
static Future<bool> checkNotificationPermissions() async {
    if (Platform.isAndroid) {
        return await Permission.notification.status.isGranted;
    } else if (Platform.isIOS) {
        // EN IOS NO SE PUEDE VERIFICAR FÁCILMENTE, ASÍQUE ASUMIMOS QUE ES TRUE TRAS PEDIRLOS
        return true;
    }
    return false;
}
```

Mostrar una notificación simple

Para mostrar una notificación inmediata, se llama a *showNotification()*:

```
// MOSTRAMOS LA NOTIFICACION
await flutterLocalNotificationsPlugin.show(
    id,
    title,
    body,
    platformDetails,
);
```



Mostrar una notificación programada

Para mostrar una notificación programada utilizamos el método `showNotificationSchedule`. Se hace con `zonedSchedule` para que respete la zona horaria del usuario:

```
static Future<void> showNotificationSchedule(BuildContext context, int id, String title, DateTime scheduleDate) async {
  const AndroidNotificationDetails androidDetails =
    AndroidNotificationDetails("cubeX_channel", "CubeX Notification",
      importance: Importance.high, priority: Priority.high);

  const NotificationDetails platformDetails =
    NotificationDetails(android: androidDetails);

  await flutterLocalNotificationsPlugin.zonedSchedule(
    id,
    Internationalization.internationalization.getLocalizations(context, "${title}.title"),
    Internationalization.internationalization.getLocalizations(context, "${title}.content"),
    tz.TZDateTime.from(scheduleDate, tz.local),
    platformDetails,
    matchDateTimeComponents: DateTimeComponents.time,
    androidScheduleMode: AndroidScheduleMode.exactAllowWhileIdle);
}
```

La pantalla

`ManageDisabledNotificationScreen` se muestra si las notificaciones están desactivadas. Aquí el usuario puede conceder permisos desde la aplicación o ir a Ajustes:

```
// SE SOLICITA EL PERMISO
var status = await Permission.notification.request();
DatabaseHelper.logger.i(status);

if (status.isGranted) {
  if (context.mounted) {
    // SI EN LOS AJUSTES ESTA PERMITIDO, SE CAMBIA EL VALOR DE LAS PREFERENCIAS A TRUE
    // POR LO QUE SE MUESTRAN LA PANTALLA CON LAS NOTIFICACIONES
    configuration.changeValue("isActive", true);
    AlertUtil.showSnackBarInformation(
      context, "active_notifications");

    // SE ENVIA UNA NOTIFICACION DE QUE SE ACTIVARON LAS NOTIFICACIONES
    NotificationService().showNotification(
      id: 0,
      titleKey: "notification",
      bodyKey: "notifications_activated",
      context: context);
  }
} else {
  // SI ESTA DENEGADO EL PERMISO SE MUESTRA UNA ALERTA QUE ABRE LOS SETTINGS
  if (context.mounted) {
    AlertUtil.showAlertSimple(
      context, "allow_permissions");
  }
}
```

Además, se detecta si el usuario ha vuelto desde Ajustes para re-evaluar los permisos:

```
@override
void didChangeAppLifecycleState(AppLifecycleState state) {
  if (state == AppLifecycleState.resumed) {
    // CUANDO LA APP VUELVE AL PRIMER PLANO SE CHEQUEAN LOS PERMISOS
    _checkNotificationPermissions();
  }
} // ESTE METODO SE LLAMARA CUANDO LA APP CAMBIE SU ESTADO DE CICLO DE VIDA
// ES DECIR, CUANDO EL USUARIO SE VAYA DE LA APP Y VUELVA SE CHEQUEARA LOS PERMISOS
// POR SI HA IDO A QUITAR EL PERMISO DE AJUSTES
```

6.4.3.2 Diseño gráfico con CustomPainter

Para mejorar el diseño visual de las pantallas de autenticación (login y signup), entre otras, se utiliza una implementación personalizada de gráficos en Flutter mediante la clase `CustomPainter`. Se hace uso de la clase `WaveContainerPainter`, que permite dibujar una figura ondulada decorativa (wave) en la parte superior de la pantalla, proporcionando un mejor estilo a la interfaz para hacer mas atractiva la experiencia del usuario. Este diseño se logra a través del dibujo manual de curvas sobre un `Canvas`.

Uso de la clase WaveContainerPainter

```
/// Clase que dibuja una wave personalizada con un degradado y una sombra violeta.
///
/// Esta clase herada de [CustomPainter], permitiendo dibujar waves personalizadas en un widget.
/// La onda tiene tres secciones: una onda pequeña, una onda intermedia y una onda grande.
/// Además, se dibuja una sombra desplazada ligeramente hacia arriba para dar un contorno.
class WaveContainerPainter extends CustomPainter {
  @override
  void paint(Canvas canvas, Size size) {...}

  @override
  bool shouldRepaint(covariant CustomPainter oldDelegate) {
    // DEVUELVE FALSE PORQUE ESTA WAVE NO CAMBIA Y NO NECESITA REPINTARSE
    return false;
}
```

Esta clase sobrescribe el método `paint` para construir una forma compuesta por varias ondas y una sombra. Está diseñada para ser estática por lo que `shouldRepaint` retorna false.

Estructura del Dibujo

- **Gradiente de color (Degrado superior-inferior):**
 Se establece un degradado vertical que va del color superior al inferior, definido en la clase `AppColors`.

```
// 1. SE CREA UN OBJETO PAINT QUE SE USA PARA DIBUJAR LA WAVE PRINCIPAL
// SE LE ASIGNA UN DEGRADADO LINEAL QUE CAMBIA ENTRE DOS COLORES
final paint = Paint()
  ..shader = const LinearGradient(
    begin: Alignment.topCenter, // EL DEGRADADO EMPIEZA DESDE ARRIBA
    end: Alignment.bottomCenter, // Y TERMINA ABAJO
    colors: [
      AppColors.upLinearColor, // COLOR SUPERIOR
      AppColors.downLinearColor, // COLOR INFERIOR
    ],
  ).createShader(Rect.fromLTWH(0, 0, size.width, size.height))
  ..style = PaintingStyle.fill; // RELLENATODO EL AREA
```

- **Sombra adicional:**

Se dibuja una sombra violeta ligeramente desplazada hacia arriba:

```
// 4. SE CREA OTRO PATH PARA DIBUJAR LA SOMBRA
final shadowPath = Path();
```

```
// 9. SE CREA LA SOMBRA DESPLAZANDO EL CAMINO PRINCIPAL HACIA ARRIBA
shadowPath.addPath(path, Offset(0, -10)); // MUEVE EL CAMINO 10 PIXELES ARRIBA

// 10. SE DIBUJA PRIMERO LA SOMBRA PARA QUE QUEDA DEBAJO DE LA WAVE
canvas.drawPath(shadowPath, shadowPaint);
```

- **Forma de la wave:**

La onda se compone de tres secciones usando curvas cúbicas:

- o Una onda pequeña inicial.
- o Una onda intermedia que suaviza la transición.
- o Una onda final más grande que aporta dramatismo visual.

```
// 5. DIBUJAMOS LA PRIMERA WAVE PEQUEÑA QUE EMPIEZA EN UN PUNTO MAS BAJO
// moveTo() POSICIONA EL PUNTO INICIAL DEL CAMINO
path.moveTo(0, size.height * 0.28);
path.cubicTo(
    size.width * 0.2, // PRIMER PUNTO DE CONTROL X
    size.height * 0.15, // PRIMER PUNTO DE CONTROL Y
    size.width * 0.3, // SEGUNDO PUNTO DE CONTROL X
    size.height * 0.23, // SEGUNDO PUNTO DE CONTROL Y
    size.width * 0.45, // PUNTO FINAL X
    size.height * 0.23 // PUNTO FINAL Y
);

// 6. DIBUJAMOS LA WAVE INTERMEDIA QUE DESCENDE UN POCO MAS
//(es la ola que baja un poco y que une las dos waves principales)
path.cubicTo(
    size.width * 0.45, // PRIMER PUNTO DE CONTROL X
    size.height * 0.22, // PRIMER PUNTO DE CONTROL Y
    size.width * 0.54, // SEGUNDO PUNTO DE CONTROL X
    size.height * 0.24, // SEGUNDO PUNTO DE CONTROL Y
    size.width * 0.6, // PUNTO FINAL X
    size.height * 0.2 // PUNTO FINAL Y
);

// 7. SE DIBUJA LA ONDA FINAL MAS GRANDE, QUE SUBE MAS ALTA
path.cubicTo(
    size.width * 0.85, // PRIMER PUNTO DE CONTROL X
    size.height * 0.05, // PRIMER PUNTO DE CONTROL Y
    size.width * 0.9, // SEGUNDO PUNTO DE CONTROL X
    size.height * 0.15, // SEGUNDO PUNTO DE CONTROL Y
    size.width, // PUNTO FINAL X (BORDE DERECHO)
    size.height * 0.18 // PUNTO FINAL Y
);
```

Font size: 10pt

```
// 8. SE CIERRA EL CAMINO BAJANDO HASTA LA PARTE INFERIOR
path.lineTo(size.width, size.height); // LLEVA LA LINEA AL FONDO DERECHO
path.lineTo(0, size.height); // LLEVA LA LINEA AL FONDO IZQUIERDO
path.close(); // CIERRA EL CAMINO
```

- **Dibujo final sobre el Canvas:**

```
// 10. SE DIBUJA PRIMERO LA SOMBRA PARA QUE QUEDA DEBAJO DE LA WAVE
canvas.drawPath(shadowPath, shadowPaint);

// 11. SE DIBUJA LA WAVE PRINCIPAL ENCIMA DE LA SOMBRA
canvas.drawPath(path, paint);
```

6.4.3.3 Tutorial con SmartDialog

Con el fin de mejorar la **experiencia de usuario (UX)** y facilitar la **adaptación inicial a la aplicación**, se desarrolló un **sistema de tutorial guiado por la mascota de la aplicación, Cubix**. Este componente tiene como objetivo **introducir al usuario en las principales funcionalidades** de la interfaz tras registrarse por primera vez. Para ello, se utilizó la librería *SmartDialog*, que permite mostrar **ventanas animadas y resaltados**.

Diseño y estructura del componente

El tutorial fue encapsulado en la clase *StartGuideComponent*, la cual gestiona la lógica de los pasos, la navegación entre pantallas y el resaltado de elementos clave.

Cada paso del tutorial se define mediante una instancia de la clase *GuideStep*, que contiene:

- Claves de texto internacionalizadas (*textKeys*).
- Imagen asociada de Cubix que actúa como asistente visual (*mascotAsset*).
- Posición del elemento a resaltar (*highlightPosition*).
- Indicación opcional de si se debe navegar a otra pantalla (*navigationIndex*).
- Una bandera booleana que determina si es el último paso (*isFinalStep*).

Por ejemplo:

```
const GuideStep = (function() {
    'use strict';
    var GuideStep = function(textKeys, mascotAsset, highlightPosition, navigationIndex, isFinalStep, mascotPosition) {
        this.textKeys = textKeys;
        this.mascotAsset = mascotAsset;
        this.highlightPosition = highlightPosition;
        this.navigationIndex = navigationIndex;
        this.isFinalStep = isFinalStep;
        this.mascotPosition = mascotPosition;
    };
    return GuideStep;
})();
```

En este paso, se resalta la cabecera de la aplicación, se muestra una imagen de la mascota, y se presenta un texto animado con la clave *tour_header*.

Internacionalización y selección de idioma

Desde el primer paso del tutorial, se le da al usuario la opción de seleccionar el idioma (español o inglés), lo cual es gestionado mediante dos *ValueNotifier<bool>*:

```
/// Atributos que indican los botones de selección de idioma.
ValueNotifier<bool> isSelectedSpainNotifier = ValueNotifier<bool>(false);
ValueNotifier<bool> isSelectedEnglishNotifier = ValueNotifier<bool>(true);
```

Dependiendo la selección del usuario, se refleja visualmente mediante bordes animados en las banderas, al cambiar se actualiza el idioma actual de la app

Animación del texto

El texto mostrado en cada paso está animado con un efecto de máquina de escribir, utilizando el widget AnimatedTextKit. El texto que se muestra está completamente adaptado al idioma que ha elegido el usuario anteriormente.

```
// MAPEA EL TEXTO A UN TEXTO ANIMADO
List<AnimatedText> animatedTexts = texts.map((text) {
  return TypewriterAnimatedText(
    Internationalization.internationalization
      .getLocalizations(context, text),
    speed: _animationSpeed,
  ); // TypewriterAnimatedText
}).toList();

// APLICA UN ESTILO AL TEXTO ANIMADO
return DefaultTextStyle(
  style: const TextStyle(fontSize: 30.0, fontFamily: 'Console'),
  child: AnimatedTextKit(
    // SE REPITE SIEMPRE
    repeatForever: true,
    // PAUSA ESTABLECIDA ENTRE CADA TEXTO
    pause: _pauseDuration,
    // LISTA DE TEXTOS A MOSTRAR
    animatedTexts: animatedTexts,
  ), // AnimatedTextKit
); // DefaultTextStyle
```

Navegación y resaltado

El tutorial destaca componentes clave del paso en concreto del tutorial mediante resaltados blancos, definidas por la clase *HighlightPosition*.

Además, si un paso requiere navegar a otra pantalla (como ir al historial o las estadísticas), se define con *navigationIndex*, y la transición se gestiona desde la función *_moveToNextStep()*:

```
if (step.navigationIndex != null) {
  ChangeScreen.changeScreen(
    BottomNavigation(index: step.navigationIndex!), context);
} // CAMBIA DE PANTALLA SI EL PASO LO INDICA
```

Avance del tutorial

El tutorial se puede avanzar mediante un botón de "Continue", y en cualquier momento puede ser omitido con el botón "Skip":

Al llegar al paso final, el botón cambia su comportamiento y texto a "Finish", cerrando completamente la guía.

```
child: ElevatedButton(
  onPressed: () async {
    SmartDialog.dismiss();
    await _moveToNextStep(context);
  },
  child: const Text("Continue"),
), // ElevatedButton
```

Contenido completo del paso

Cuando se activa un paso del tutorial, se muestra un SmartDialog.showAttach() que contiene:

- ❖ Imagen de la mascota.
- ❖ Texto animado.
- ❖ Caja de resaltado si aplica.
- ❖ Botones de navegación.

```
// MUESTRA EL DIALOGO DEL PASO ACTUAL
await _replaceDialog();
return SmartDialog.showAttach(
    // CONTEXTO DONDE SE EJECUTARA EL DIALOGO
    targetContext: context,
    alignment: Alignment.center,
    // EVITA QUE AL CLICKAR FUERA SE CIERRE EL DIALOGO
    clickMaskDismiss: false,
    // ASEGURA QUE SOLO HAY UN DIALOGO ACTIVO (importante)
    keepSingle: true,
    // USA LA ANIMACION DE DESFALCIMIENTO AL MOSTRAR EL DIALOGO
    animationType: SmartAnimationType.fade,
    // SI HAY UNA POSICION DE RESALTADO LO CONSTRUYE EN ESA PARTE PROPORCIONADA
    highlightBuilder: positionedHighlight != null
        ? (_, __) => positionedHighlight!
        : null,
    builder: () =>
        return SizedBox(
            height: 200,
            width: 300,
            child: Column(
                mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                children: [
                    // MASCOTA
                    Image.asset('assets/images/icono.png'),
                    // TITULO
                    Text(
                        'PASO 1',
                        style: TextStyle(
                            color: Colors.white,
                            fontSize: 16,
                            fontWeight: FontWeight.bold),
                    ),
                    // TEXTO
                    Text(
                        'Este es el primer paso del tutorial. Aquí se muestra una descripción detallada de lo que debes hacer. Puedes navegar entre los pasos usando los botones de navegación en la parte inferior.',  
...  
...  
...
```

6.4.3.4 Generación y exportación de PDF

Una de las funcionalidades importantes de la aplicación desarrollada es la capacidad de **exportar los tiempos registrados en una sesión como un archivo PDF**, el cual puede ser descargado o compartido por el usuario. Esto resulta especialmente útil para compartir resultados con terceros (amigos, otros speedcubers, etc.).

Para ello, se ha implementado la clase PdfGenerator, que se encarga de orquestar todo el proceso de generación, construcción y exportación del documento PDF.

Arquitectura y flujo general

El proceso completo se divide en tres etapas principales:

- ❖ **Recopilación de datos de sesión**
- ❖ **Generación del contenido PDF**
- ❖ **Exportación o compartición del archivo**

Todo el flujo es coordinado por el método público convertPdf.

```
Future<void> convertPdf(BuildContext context, ShareOption shareOption) async {
```

Este método:

- ❖ Obtiene el usuario actual, su sesión activa y el tipo de cubo seleccionado.
- ❖ Verifica que los datos necesarios existen y que hay tiempos registrados.
- ❖ Llama internamente a _generatePdf(...) para construir el documento.
- ❖ Por último, ejecuta _pdfExport(...) para guardarlo o compartirlo.

1. Generación del contenido del PDF

El contenido del documento se construye mediante el paquete pdf de Flutter. Se utiliza el método privado `_generatePdf(...)`:

```
Future<pw.Document> _generatePdf(CurrentSession currentSession,
| CurrentCubeType currentCube, List<TimeTraining> times) async {
```

Este método genera un documento de tipo `pw.Document`, el cual se compone de:

- ❖ Un encabezado con el nombre de la sesión, tipo de cubo y fecha.
- ❖ Una tabla con los tiempos registrados, mostrando:
 - Tiempo en segundos
 - Penalización (si la hubo)
 - Scramble utilizado

Si no se han registrado tiempos, se muestra un mensaje informativo.

Ejemplo de estructura:

Esto permite generar un documento claro, organizado y exportable con formato estándar (A4, márgenes, texto centrado, etc.).

```
if (times.isNotEmpty)
  pw.Table.fromTextArray(
    // SE AÑADE BORDES
    border: pw.TableBorder.all(),
    headerDecoration:
      const BoxDecoration(color: PdfColors.grey300),
    headerStyle: pw.TextStyle(fontWeight: pw.FontWeight.bold),
    // ENCABEZADO DE LA TABLA
    headers: ['Time (s)', 'Penalty', 'Scramble'],
    // LOS TIEMPOS
    data: times
      .map((time) => [
        time.timeInSeconds.toString(),
        time.penalty ?? 'No Penalty',
        time.scramble
      ])
      .toList(),
    // SE CENTRA
```

2. Exportación del archivo

La exportación del documento se realiza mediante el método `_pdfExport(...)`:

```
Future<void> _pdfExport(
  BuildContext context, pw.Document pdf, String sessionName,
  {required ShareOption shareOption}) async {
```

Este método detecta la **plataforma del dispositivo** y adapta el comportamiento:

- ❖ **Windows/macOS:** se abre un explorador de archivos donde el usuario puede elegir dónde guardar el archivo.
- ❖ **Android/iOS:** el PDF se guarda en el directorio temporal del sistema, y se puede compartir directamente con otras apps si el usuario así lo solicita.

Para compartir el archivo, se utiliza share_plus:

```
// ESTE METODO DEL PAQUETE 'share_plus' PERMITE COMPARTIR
// MULTIPLES ARCHIVOS SIMULTANEAMENTE
await Share.shareXFiles(
    // EN ESTE CASO, SE PASA UNA LISTA CON UN SOLO ARCHIVO
    [XFile(filePath)],
    text: 'Tiempos de la sesión',
);
```

Esto permite integrar el sistema de compartición del dispositivo (WhatsApp, Drive, Gmail, etc.).

Enum ShareOption

Se define un enum sencillo que permite especificar la acción del usuario:

- ❖ **download**: el archivo se guarda localmente.
- ❖ **moreOptions**: se comparte con otras apps.

```
/// Enum que define las opciones de compartir un archivo PDF.
///
/// - [download]: Guarda el archivo sin compartirlo.
/// - [moreOptions]: Permite compartir el archivo con otras aplicaciones.
enum ShareOption { download, moreOptions }
```

6.4.3.5 Implementación de vibración y sonido personalizado

Durante la fase de inspección previa a la resolución, se incorporan alertas configurables por el usuario que permiten advertir los segundos 12 y 8 de la inspección mediante vibración, sonido o ambos (o ninguno si lo desactiva).

Esta funcionalidad ayuda al usuario a anticiparse al final del tiempo de inspección, siguiendo con las normas de competición oficiales, mejorando la usabilidad de la app con una experiencia adaptable a las preferencias del usuario.

Alertas mediante vibración

Si el usuario ha seleccionado vibración como tipo de alerta (o ambas), el dispositivo vibrara durante un segundo en los segundos 12 y 8:

```
if (confi.inspectionAlertType == InspectionAlertType.vibrant ||
    confi.inspectionAlertType == InspectionAlertType.both) {
    // HACE QUE VIBRE UN SEGUNDO
    Vibration.vibrate(duration: 1000);
} // SI EL TIPO DE ALERTA ES VIBRACION O AMBAS
```

Alertas mediante sonido

En caso de tener activo la alerta con sonido, se reproducirá un archivo de audio correspondiente al segundo específico. Además, se adapta al idioma de la aplicación (español o inglés)

```

if (_currentTime == 8) {
    WidgetsBinding.instance.addPostFrameCallback(_)
        async {
            await audioPlayer.play(AssetSource(
                // DEPENDIENDO DEL IDIOMA, REPRODUCE UNO U OTRO
                locale.languageCode == "en"
                    ? "sound_alerts/eight_seconds.mp3"
                    : "sound_alerts/ocho_segundos.mp3") // AssetSource
            );
        });
} // SI EL TIEMPO DE INSPECCIÓN ES DE 8 SEGUNDOS

if (_currentTime == 12) {
    WidgetsBinding.instance.addPostFrameCallback(_)
        async {
            await audioPlayer.play(AssetSource(
                // DEPENDIENDO DEL IDIOMA, REPRODUCE UNO U OTRO
                locale.languageCode == "en"
                    ? "sound_alerts/twelve_seconds.mp3"
                    : "sound_alerts/doce_segundos.mp3")); // AssetSource
        });
} // SI EL TIEMPO DE INSPECCIÓN ES DE 12 SEGUNDOS

```

6.4.3.6 Generación aleatoria de scrambles válidos

Uno de los aspectos clave de la aplicación es la generación de secuencias aleatorias (*scrambles*) válidas para diferentes tipos de cubos, como el 3x3, Skewb, Megaminx o Square-1. Esta funcionalidad es fundamental para proporcionar al usuario una resolución aleatoria antes de iniciar el cronómetro.

Para ello, se ha diseñado una clase específica llamada ScrambleGenerator, que contiene lógica adaptada para cada tipo de cubo.

Selección de movimientos válidos

Cada tipo de cubo tiene un conjunto distinto de movimientos válidos según sus capas. Por ejemplo:

```
static List<String> moves = ["U", "R", "B", "L", "D", "F"]; // ARRAY CON LOS MOVIMIENTOS POSIBLES

static List<String> movesSkewb = ["U", "R", "B", "L", ];

static List<String> movesMegaminx = ["U", "R", "D", ];

static List<String> moves4x4 = ["U", "D", "L", "R", "F", "B", "Uw",
    "Dw", "Lw", "Rw", "Fw", "Bw"];
```

Esto permite adaptar la generación de movimientos a las características físicas de cada puzzle.

Prevención de movimientos consecutivos repetidos

En el caso del cubo 3x3 (y derivados), se evita repetir la misma cara dos veces seguidas, mejorando la utilidad del *scramble*:

```
while (move.contains(lastMove)) {
    move = moves[
        Random().nextInt(isSkewb ? movesSkewb.length : moves.length)];
} // CUANDO LA CAPA SEA IGUAL A LA ANTERIOR CAPA, SE COGE OTRA CAPA ALEATORIAMENTE
```

Generación con modificadores '1' y '2'

En cada movimiento, se introduce aleatoriamente un giro ' (inverso), 2 (doble) o ninguno, con probabilidades dependiendo del cubo:

```
int randomMove = Random().nextInt(100);
if (isSkewb ? randomMove <= 50 : randomMove <= 30) {
    // SI ES MENOR O IGUAL A 30 SE LE COLOCA '1'
    scrambleList.add("${move}'");
    // EN LASTMOVE SOLO SE GUARDA LA CAPA, SIN EL GIRO ADICIONAL
    lastMove = move;
} else if (isSkewb == false) {
    // SI NO ES EL SCRAMBLE DEL SKEWB ENTONCES SE LE AÑADE EL MOVIMIENTO ADICIONAL '2'
    if (randomMove <= 60) {
        // SI NO ES MENOR O IGUAL A 30 Y ES MENOR O IGUAL A 60 SE LE COLOCA UN 2
        scrambleList.add("${move}2");
        lastMove = move;
    }
} else {
    // Y SI NO, SE AGREGA SOLO LA CAPA
    scrambleList.add(move);
    lastMove = move;
}
```

- Generación de *scramble* para 3x3

El método `generateScramble32Skewb()` es el núcleo para los cubos 2x2, 3x3, Skewb y Pyraminx, con adaptaciones según el parámetro `isSkewb`.

```
case "2x2x2":
    // USA LAS MISMAS CAPAS QUE EL 3x3 PERO CON 11 MOVIMIENTOS
    return generateScramble32Skewb(11);
case "3x3x3":
    // RANGO ENTRE 20 A 25 MOVIMIENTOS DE CAPA PARA GENERAR EL SCRAMBLE
    int random = (Random().nextInt(25 - 20 + 1) + 20);
    return generateScramble32Skewb(random);
case "skewb":
    // EL SCRAMBLE DEL SKEWB USA SOLO LAS CAPAS U, R, B, L Y SON 11 MOVIMIENTOS
    return generateScramble32Skewb(11, true);
case "pyraminx":
    // EL SCRAMBLE ES EL MISMO QUE EL SKEWB SOLO QUE CON 9 MOVIMIENTOS
    return generateScramble32Skewb(9, true);
```

- Otros casos: Square-1 y Megaminx

Estos cubos requieren una lógica distinta debido a sus sistemas de movimiento:

Square-1

Se genera un par de movimientos con rotaciones superior/inferior y un giro de eje /:

```
// AÑADE EL MOVIMIENTO EN FORMATO (top,bottom)
scramble.add("($top,$bottom)");

// AÑADE UN GIRO DE EJE "/"
scramble.add("/");

```

Megaminx

Para el Megaminx, se priorizan los movimientos "R" y "D", y se utilizan variantes como ++, -- y ':

```
move == "U" ? scramble.add("$move'") : scramble.add("$move++");
```

6.4.3.7 Gestión de la interfaz: ocultar teclado y barra de estado

Para mejorar la experiencia de usuario se implementaron dos funcionalidades a destacar: el cierre del teclado mediante un toque y la ocultación de la barra de estado del dispositivo.

Ocultar el teclado al tocar fuera

Se permite que el teclado se cierre automáticamente cuando el usuario toca fuera de un campo de texto. Para ello, se utiliza un GestureDetector que detecta toques en cualquier parte de la pantalla y elimina el foco del campo activo

```
child: GestureDetector(
    // CUANDO SE TOCA EN CUALQUIER LADO DE LA PANTALLA
    onTap: () {
        // SE QUITA EL FOCO DEL ELEMENTO ACTUAL, LO QUE CIERRA EL TECLADO SI ESTA ABIERTO
        FocusManager.instance.primaryFocus?.unfocus();
    },
)
```

Ocultar la barra de estado

Ademas, se elimino la barra de notificaciones y la de la navegación del sistema mediante el SystemUiMode

```
// QUITA EL STATUS BAR EN EL MOVIL  
SystemChrome.setEnabledSystemUIMode(SystemUiMode.immersiveSticky);
```

Aunque, esta implementación ha llevado un inconveniente visual, al ocultar la barra aparece una franja negra en el lugar de las notificaciones, que no fue posible solucionar correctamente.

Flutter ofrece distintos modos para gestionar la visibilidad del sistema UI:

- **leanBack**: Oculta las barras del sistema temporalmente; reaparecen al tocar cualquier parte de la pantalla.
- **immersive**: Oculta completamente las barras del sistema, pero reaparecen si el usuario desliza desde el borde de la pantalla.
- **immersiveSticky**: Similar a immersive, pero las barras del sistema reaparecen brevemente al hacer un gesto y luego se ocultan de nuevo automáticamente.
- **edgeToEdge**: Permite que el contenido de la aplicación se extienda detrás de las barras del sistema, pero no las oculta.
- **manual**: Permite controlar manualmente qué elementos del sistema se muestran u ocultan mediante funciones específicas.

6.4.3.8 Confirmación doble para cerrar la aplicación

Se implementa un mecanismo para evitar salidas accidentales mediante una confirmación doble al pulsar el botón de retroceso del dispositivo. Esta funcionalidad se ha desarrollado con el objetivo de controlar salidas involuntarias que podrían implicar volver al login.

La clase DoubleTapExitDialog envuelve cualquier pantalla (se aplica al BottomNavigation para que englobe tanto la pantalla principal, como el historial y las estadísticas) e intercepta si el usuario pulsa el botón de retroceso. Ese comportamiento detecta si el usuario ha pulsado dos veces el botón de ir para atrás en un intervalo inferior a dos segundos, si es así, se lanza una alerta de confirmación para cerrar la aplicación. Si no, se muestra un mensaje informativo para indicarle que debe pulsar nuevamente el botón para salir



```
Future<bool> _confirmExit() async {
    final now = DateTime.now();

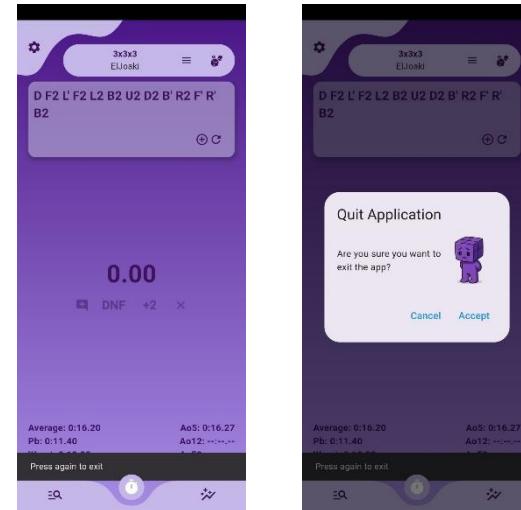
    if (_lastBackButtonTime == null ||
        now.difference(_lastBackButtonTime!) > _exitLimitTime) {
        // PRIMERA PULSACION MOSTRAR MENSAJE
        _lastBackButtonTime = now;
        AlertUtil.showSnackBarWithDuration(context, 2, "press_again_to_exit");
        return false;
    } // SE VERIFICA SI ES LA PRIMERA VEZ O YA PASO EL TIEMPO LIMITE ENTRE PULSACIONES

    // LA SEGUNDA PULSACION MUESTRA LA ALERTA DE SALIDA
    AlertUtil.showExitAlert(context);
    return true;
}
```

Esta función se activa mediante el componente PopScope, que se encarga de captar el intento de navegar hacia atrás y evitar que la app se cierre directamente.

```
Widget build(BuildContext context) {
    return PopScope(
        // EVITAR QUE SE PUEDA SALIR DIRECTAMENTE CON EL BOTON DE ATRAS
        canPop: false,
        onPopInvoked: (bool didPop) async {
            if (!didPop) {
                // MOSTRAR ALERTA O MENSAJE DE CONFIRMACION
                await _confirmExit();
            } // SI NO SE HIZO POP, SE MANEJA MANUALMENTE LA SALIDA
        },
        // CONTENIDO PRINCIPAL QUE SE MUESTRA EN LA PANTALLA
        child: widget.child,
    );
}
```

Además, gracias a que la sesión del usuario se mantiene activa, al volver a abrir la aplicación tras cerrarla, el usuario continua en el mismo punto.



6.4.3.9 Cálculo del tiempo de uso en pantalla

Se ha implementado el sistema de contador del tiempo de uso de la aplicación, el cual permite contar el tiempo mientras la aplicación permanece activa en primer plano. Este sistema se basa en un temporizador que solo se activa cuando la app está en primer plano y se detiene cuando la app pasa a segundo plano o es minimizada.

La clase **CurrentUsageTimer** es la encargada de gestionar este sistema. Utiliza la funcionalidad de **WidgetsBindingObserver** para detectar cambios en el ciclo de vida de la aplicación y gestionar de manera adecuada el inicio y la detención del contador.

- **Contador de tiempo**

Se implementa un temporizador que incrementa el tiempo de uso cada segundo mientras la aplicación está en primer plano:

```
/// Método que inicia el contador solo si la app está activa
void start() {
    if (!_isRunning) {
        _isRunning = true;
        _timer = Timer.periodic(const Duration(seconds: 1), (timer) {
            _secondsUsed++; // INCREMENTA EL TIEMPO CADA SEGUNDO
            notifyListeners(); // NOTIFICA A LA UI SOBRE EL CAMBIO DE ESTADO
        }); // Timer.periodic
    }
}
```

El temporizador se detiene cuando la aplicación pasa a segundo plano, usando los métodos de **WidgetsBindingObserver** para gestionar los cambios en el ciclo de vida de la app:

```
/// Método que detecta cuando la app se abre o se minimiza
@Override
void didChangeAppLifecycleState(AppLifecycleState state) {
    if (state == AppLifecycleState.resumed) {
        start(); // SI LA APP SE ABRE, SE REANUDA EL CONTADOR
    } else if (state == AppLifecycleState.paused ||
               state == AppLifecycleState.inactive ||
               state == AppLifecycleState.detached) {
        stop(); // SI LA APP SE MINIMIZA, SE DETIENE EL CONTADOR
    }
}
```

- **Persistencia de datos**

El uso total se guarda en el **SharedPreferences**, asegurando que, aunque la aplicación se cierra y se vuelva a abrir, el tiempo de uso no se pierda:

```
/// Método que carga el tiempo almacenado en el SharedPreferences.
Future<void> _loadTime() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    _secondsUsed =
        prefs.getInt('usage_time') ?? 0; // OBTIENE EL TIEMPO ALMACENADO
    notifyListeners(); // NOTIFICA A LA UI SOBRE EL CAMBIO DE ESTADO
}
```

Y para **guardar** el tiempo de uso cuando la app se detiene:

```
/// Método que guarda el tiempo en el SharedPreferences.
Future<void> _saveTime() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    await prefs.setInt('usage_time', _secondsUsed); // ALMACENA EL TIEMPO ACTUAL
}
```

- Formato de tiempo

Para mostrar el tiempo de uso se implementa un metodo que formatea el tiempo en años, meses, días y horas:

```
/// Método para formatear el tiempo en años, meses, días y horas.
static String formatTime(int seconds) {
    int years = seconds ~/ (365 * 24 * 3600); // CALCULA LOS AÑOS
    int remainingSeconds = seconds % (365 * 24 * 3600);

    int months = remainingSeconds ~/ (30 * 24 * 3600); // CALCULA LOS MESES
    remainingSeconds %= (30 * 24 * 3600);

    int days = remainingSeconds ~/ (24 * 3600); // CALCULA LOS DIAS
    remainingSeconds %= (24 * 3600);

    int hours = remainingSeconds ~/ 3600; // CALCULA LAS HORAS

    // DEVUELVE EL TIEMPO FORMATEADO EN UNA CADENA
    if (years > 0) {
        return "${years}a ${months}m ${days}d";
    } else if (months > 0) {
        return "${months}m ${days}d";
    } else if (days > 0) {
        return "${days}d ${hours}h";
    } else {
        return hours == 1 ? "$hours hora" : "$hours horas";
    }
}
```

- Gestión del ciclo de vida de la app

El sistema de observación del ciclo de vida es clave para iniciar y detener el timer cuando la app cambie de estado, por eso se usa WidgetBindingObserver para saber los diferentes estados de la app:

```
/// Método que detecta cuando la app se abre o se minimiza
@Override
void didChangeAppLifecycleState(AppLifecycleState state) {
    if (state == AppLifecycleState.resumed) {
        start(); // SI LA APP SE ABRE, SE REANUDA EL CONTADOR
    } else if (state == AppLifecycleState.paused || state == AppLifecycleState.inactive || state == AppLifecycleState.detached) {
        stop(); // SI LA APP SE MINIMIZA, SE DETIENE EL CONTADOR
    }
}
```

Actualmente, esta implementación solo cuenta el tiempo de uso de la aplicación de manera global, es decir, no distingue entre diferentes usuarios. Esta limitación está redactada como una de las posibles ampliaciones a desarrollar.

6.4.3.10 Visualización de datos con gráficas interactivas

Para ofrecer al usuario un análisis visual de su rendimiento en una sesión de un tipo de cubo a lo largo del tiempo, se ha implementado el widget GraphicPerformanceContainer, una **gráfica interactiva** basada en la biblioteca syncfusion_flutter_charts. Esta gráfica permite representar los **tiempos de resolución registrados** en la sesión actual, así como una línea discontinua con diferentes puntos que muestra la evolución de los **mejores tiempos personales (PB)**.



Carga de datos

Los datos se obtienen desde Supabase según la sesión seleccionado por el usuario. Se utiliza el método `_loadTimes()` para cargar todos los tiempos de la sesión actual:

```
// SI NO SE HA INTRODUCIDO NI COMENTARIO NI TIEMPO, SE BUSCAN TODOS LOS TIEMPOS DE LA SESION
times = await timeTrainingDaoSb.getTimesOfSession(session.idSession, null,
    null, currentTime?.dateAsc, currentTime?.timeAsc);
```

Cada tiempo se convierte en un objeto `_TimeData`, que contiene el tiempo en segundos y su fecha de registro:

```
for (TimeTraining timeInList in _listTimes) {
    _TimeData timeData =
        _TimeData(timeInList.timeInSeconds, timeInList.registrationDate);
    _listTimesData.add(timeData);
} // ANADIMOS EL TIEMPO Y EL DATE A LA LISTA
```

Los **mejores tiempos personales (PB)** se calculan de forma incremental usando `_loadPbList()`, agregando un nuevo punto solo si es mejor que el anterior mínimo:

```
void _loadPbList() {
    if (_listTimesData.isNotEmpty) {
        // EL PRIMER MINIMO PB ES EL PRIMER TIEMPO QUE HIZO
        double minimoPb = _listTimesData[0].time;
        _TimeData timeData = _TimeData(minimoPb, _listTimesData[0].date);
        pbTimes.add(timeData);

        for (_TimeData c in _listTimesData) {
            if (c.time < minimoPb) {
                // SI EL TIEMPO ES MEJOR QUE EL MINIMO PB SE AGREGA A LA LISTA
                minimoPb = c.time;
                _TimeData timeDataPb = _TimeData(minimoPb, c.date);
                pbTimes.add(timeDataPb);
            }
        }
    } // VALIDAMOS QUE LA LISTA TIENE AL MENOS 1 TIEMPO PARA ESTABLECER EL MEJOR PB
}
```

Representación con SfCartesianChart

La visualización se realiza con el widget `SfCartesianChart`, usando **dos líneas**:

1. **Línea de tiempos generales:** muestra todos los tiempos registrados.
2. **Línea de PB:** representa la evolución de los mejores tiempos con una línea discontinua y puntos visibles.

El eje X muestra fechas, ajustándose al filtro actual (diario, mensual o anual), y el eje Y representa los tiempos en segundos.

```
// GRAFICA DE TIEMPOS
LineSeries<_TimeData, String>(
    // DATOS DE LOS TIEMPOS
    dataSource: _listTimesData,
    xValueMapper: (_TimeData timeData, _) {...},
    // VALOR TIEMPO
    yValueMapper: (_TimeData timeData, _) => timeData.time,
    // SE QUITA LA LEYENDA
    name: '',
    isVisibleInLegend: false,
    // SI ESTA ACTIVO SE MUESTRA EL TIEMPO
    dataLabelSettings:
        DataLabelSettings(isVisible: isNumberActive)), // LineSeries

// GRAFICA CON LOS PBS
LineSeries<_TimeData, String>(
    dataSource: pbTimes,
    xValueMapper: (_TimeData timeData, _) {...},

    // HACER LA LINEA DISCONTINUA
    dashArray: const <double>[12, 12],
    markerSettings: const MarkerSettings(...), // MarkerSettings
    yValueMapper: (_TimeData timeData, _) => timeData.time,
    name: '',
    isVisibleInLegend: false,
    dataLabelSettings:
        DataLabelSettings(isVisible: isNumberActive)) // LineSeries
) // <CartesianSeries<_TimeData, String>[]>. SfCartesianChart
```

Interactividad

El usuario puede personalizar la gráfica con varias opciones:

- ❖ Cambiar entre **orientación vertical y horizontal**.
- ❖ Activar o desactivar **etiquetas de tiempo** en los puntos.
- ❖ Filtrar los datos por **día, mes o año** con botones tipo TextButton con hover.

En caso de que no haya datos disponibles, se muestra un mensaje en lugar de la gráfica.

6.5 Pruebas.

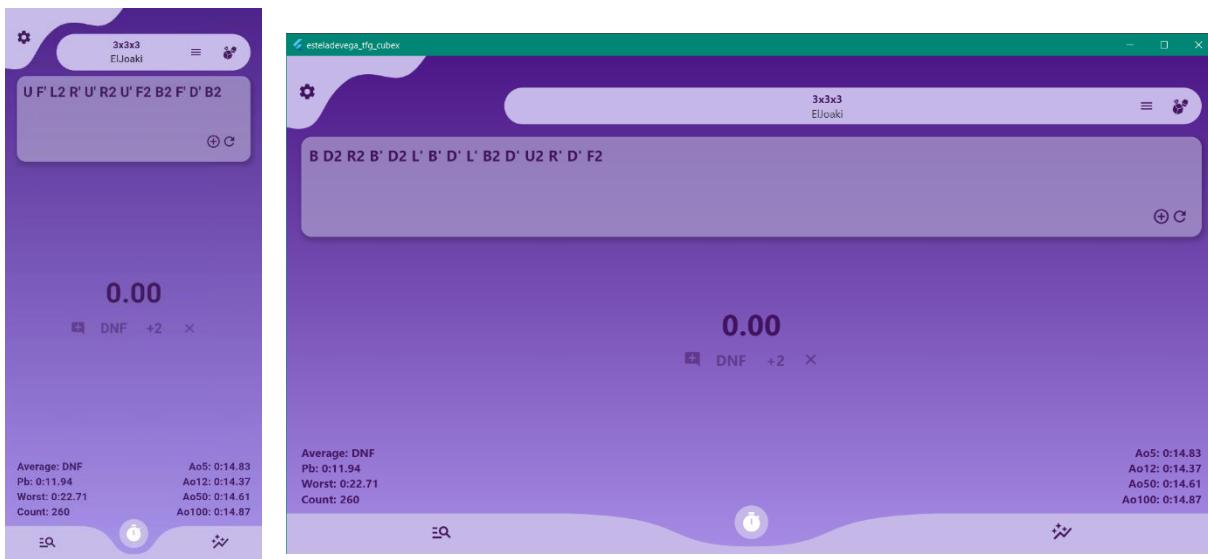
6.5.1 Pruebas de la aplicación

Una vez finalizado el desarrollo de la aplicación, se llevaron a cabo diferentes pruebas críticas para asegurar su correcto funcionamiento, rendimiento y estabilidad en los entornos disponibles.

Pruebas por plataforma

La aplicación fue diseñada para ser multiplataforma gracias a Flutter, y se realizaron pruebas específicas en los siguientes entornos:

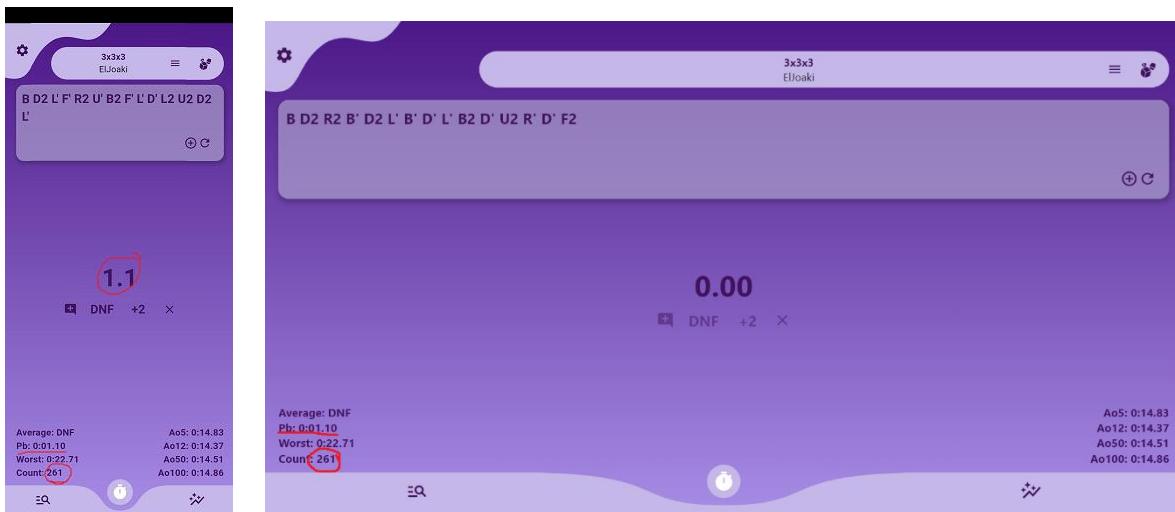
- ❖ **Android:** Se probó en varios dispositivos reales con diferentes versiones del sistema operativo. Se verificó que todas las funcionalidades —como el registro, inicio de sesión, temporizador, estadísticas, configuración, y navegación general, entre otras— funcionaran correctamente sin errores.
- ❖ **Windows:** Se realizaron pruebas en entornos de escritorio con Windows 10. La aplicación respondió bien y la interfaz se adaptó correctamente al formato de pantalla más amplio.
- ❖ **iOS y macOS:** Aunque no se han podido probar en dispositivos reales por falta de tiempo y medios, la configuración para estas plataformas se ha realizado. La conexión con Supabase (base de datos, autenticación y almacenamiento en la nube) está integrada y configurada, por lo que no debería de ocurrir errores.
- ❖ **Linux:** No se ha probado de forma práctica, aunque la compilación para esta plataforma es compatible con el entorno Flutter.



Pruebas de concurrencia:

Se simularon accesos desde distintos dispositivos con la misma cuenta de usuario. Gracias a Supabase y su arquitectura en tiempo real, los datos se sincronizaron de forma correcta entre todos los dispositivos conectados simultáneamente.

Por ejemplo, desde el móvil se realizará un tiempo y justo cuando se guarda se actualiza en el ordenador.



6.5.2 Seguridad

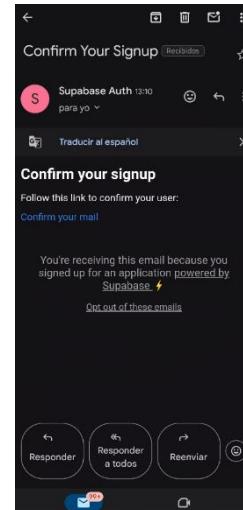
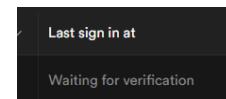
La aplicación implementa varias medidas de seguridad orientadas a garantizar el control de acceso, la protección frente a posibles ataques y una experiencia de uso segura para el usuario final. A continuación, se detallan las principales funcionalidades relacionadas con la seguridad, así como las pruebas realizadas.



6.5.2.1 Autenticación de usuarios

La autenticación en la aplicación se gestiona mediante Supabase, utilizando un sistema de inicio de sesión a través de correo electrónico y contraseña. Para completar el registro, el usuario debe verificar su dirección de correo electrónico mediante un enlace que se envía automáticamente tras completar el formulario de registro.

Hasta que el usuario no realiza dicha confirmación, su cuenta se encuentra en estado pendiente de verificación. Durante este tiempo, Supabase registra automáticamente la entrada en su tabla interna de usuarios (**auth.users**), indicando que la cuenta aún no ha sido activada.



Una vez confirmada la dirección de correo, Supabase emite una sesión válida y se lanza el evento **AuthChangeEvent.signedIn**. Esto se puede verificar por consola que el usuario ha sido autenticado correctamente, con información como el identificador del usuario, metadatos del perfil y la hora del último inicio de sesión.

```
I/flutter (15657): I/flutter (15657): #0  SupabaseHelper.getSession.<anonymous closure> (package:esteladevega_tfg_cubex/data/database/supabase_helper.dart:41:29)
I/flutter (15657): #1  _RootZone.runUnaryGuarded (dart:async/zone.dart:1609:10)
I/flutter (15657):
I/flutter (15657):   ▶ evento: AuthChangeEvent.signedIn, session: Session(providerToken: null, providerRefreshToken: null, expiresIn: 3600, tokenType: bearer, user: User(id: 25d76090-3003-4483-aa50-a7f8a6d222cb, appMetadata: {provider: email, providers: [email]}, userMetadata: {email: ██████████@gmail.com, email_verified: true, mail: ██████████@gmail.com, password: 1d769ecab3cd80089ba1871890c05a8b904092dfe5cb06d3047602aac529c7e, phone_verified: false, sub: 25d76090-3003-4483-aa50-a7f8a6d222cb, username: blueaa}, aud: authenticated, confirmationSentAt: 2025-05-17T11:10:54.953Z, recoverySentAt: null, emailChangeSentAt: null, newEmail: null, invitedAt: null, actionLink: null, email: ██████████@gmail.com, phone: , createdAt: 2025-05-17T11:10:54.902Z, confirmedAt: 2025-05-17T11:13:06.708016Z, emailConfirmedAt: 2025-05-17T11:13:06.708016Z, phoneConfirmedAt: null, lastSignInAt: 2025-05-17T11:13:08.334Z, role: authenticated, updatedAt: 2025-05-17T11:13:08.34775Z, identities: [UserIdentity(id: 25d76090-300
I/flutter (15657): |
```

6.5.2.2 Mantenimiento de la sesión

En un inicio, una vez iniciado el proceso de autenticación, el mantenimiento de la sesión se implementó utilizando SharedPreferences, que permite almacenar datos de manera persistente en el dispositivo. A través de esto, se guarda la información del usuario, de la sesión y el tipo de cubo seleccionado. Esto permite que, al cerrar y volver a abrir la aplicación, el usuario continúe accediendo sin necesidad de volver a iniciar sesión, siempre que no haya cerrado sesión manualmente.

Posteriormente, al integrar la gestión de la base de datos y el sistema de autenticación mediante Supabase, se planteó la idea de reemplazar SharedPreferences por el sistema de persistencia de sesión que Supabase ofrece de forma automática a través de tokens. Sin embargo, debido a limitaciones de tiempo y a la complejidad de la migración, no se logró lograr esa integración. Por ello, se mantuvo la gestión de sesión con SharedPreferences como solución funcional.

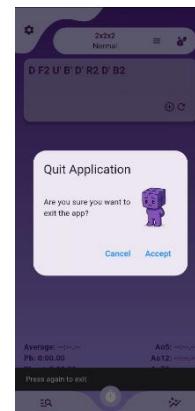
Cuando se ejecuta la aplicación, se realiza una comprobación inicial para determinar si existe un usuario guardado en SharedPreferences y si este se encuentra activo. Si el usuario está activo, el sistema redirige automáticamente a la pantalla principal del cronómetro, evitando así el paso por la pantalla de inicio o login. Si no, se carga el flujo de inicio normal.

```
// SI HAY UN USUARIO Y ESTA LOGEADO ENTONCES SE LE REDRIGE DE NUEVO A
// LA PANTALLA DEL TIMER, SI NO A LA PANTALLA DE INICIAR SESIÓN
home: newUser != null && newUser!.isLoggedIn
      ? const BottomNavigation(index: 1)
      : const IntroScreen(); // MaterialApp
```

Este comportamiento resulta especialmente útil en situaciones donde el usuario cierra la aplicación y desea volver más adelante, ya que se mantiene sus datos desde donde lo dejó.

6.5.2.3 Prevención de salidas accidentales

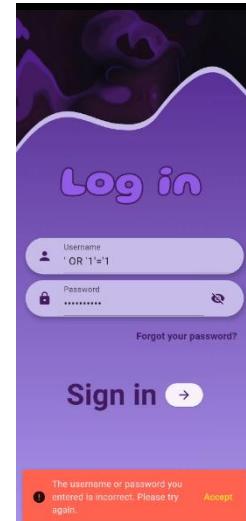
Desde la pantalla principal de la aplicación, el botón de retroceso del dispositivo no permite el retorno directo a la pantalla de inicio de sesión. En su lugar, se muestra un diálogo de confirmación preguntando si el usuario desea salir de la aplicación. Esta medida tiene como objetivo evitar salidas accidentales o no intencionadas que puedan afectar la experiencia del usuario.



6.5.2.4 Protección frente a inyecciones de código

Se han realizado pruebas introduciendo entradas maliciosas en los campos de texto, incluyendo intentos de inyección SQL (' OR '1'='1), etiquetas HTML y fragmentos de código JavaScript. La aplicación no procesa este tipo de entradas, ya que Flutter, por defecto, no interpreta HTML ni ejecuta scripts. Además, Supabase proporciona mecanismos de validación y protección frente a inyecciones si se utilizan correctamente sus funciones.

- Flutter no ejecuta ni interpreta HTML o JavaScript, por lo que este tipo de entradas no afectan visual ni funcionalmente a la aplicación.
- Supabase valida y restringe las operaciones según las políticas de seguridad definidas (como las RLS anteriormente explicadas), lo que añade una capa adicional de protección.



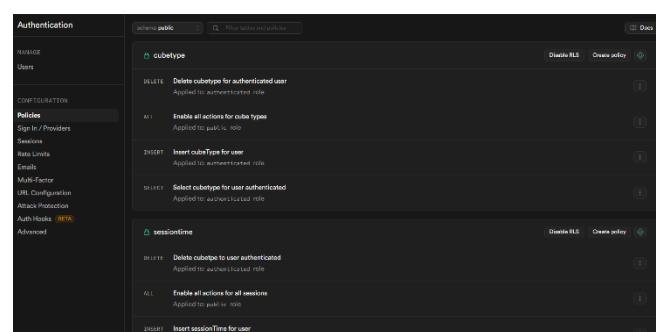
- Al tratarse como textos del propio campo, todas las entradas están sujetas a las validaciones definidas en los formularios. Esto impide que se puedan introducir comandos o estructuras que alteren la lógica del sistema.

Aunque actualmente no se implementa un sistema explícito para estas entradas maliciosas, las tecnológicas utilizadas ofrecen una protección básica suficiente.

6.5.2.5 Seguridad en el acceso a la base de datos (Row Level Security - RLS)

Para proteger el acceso a los datos almacenados en Supabase, se implementó un sistema de seguridad basado en **Row Level Security (RLS)**. Este mecanismo permite definir **políticas (policies)** a nivel de fila en cada tabla de la base de datos, lo que garantiza que los usuarios solo puedan acceder a los datos que les pertenecen.

En este proyecto, se activó RLS en las cuatro tablas principales y se definieron políticas que permiten el acceso únicamente cuando el uuid del usuario autenticado (obtenido desde la sesión) coincide con el uuid almacenado en los registros de cada tabla. Esto asegura que ningún usuario pueda consultar, modificar o eliminar información que no le corresponde.



The screenshot shows the Supabase RLS configuration interface. It displays two main sections: 'cubeType' and 'sessionTime'. Each section lists specific RLS policies:

- cubeType:**
 - DELETE:** Delete cubeType for authenticated user (Applied to: authenticated role)
 - ALL:** Enable all actions for cube type (Applied to: public role)
 - INSERT:** Insert cubeType for user (Applied to: authenticated role)
 - SELECT:** Select cubeType for user authenticated (Applied to: authenticated role)
- sessionTime:**
 - DELETE:** Delete sessionTime for user authenticated (Applied to: authenticated role)
 - ALL:** Enable all actions for all sessions (Applied to: public role)
 - INSERT:** Insert sessionTime for user

Estas políticas se aplican tanto para operaciones de lectura (SELECT) como de escritura (INSERT, UPDATE, DELETE), adaptándolas a cada necesidad específica.

De forma provisional, para permitir el correcto funcionamiento de la versión de desarrollo en Windows, se desactivaron temporalmente las restricciones en las tablas mediante una política de acceso libre (ALLOW ALL). Este ajuste es temporal y será revertido pronto para garantizar la seguridad total de los datos.

7 Manuales de usuario

7.1 Manual de usuario

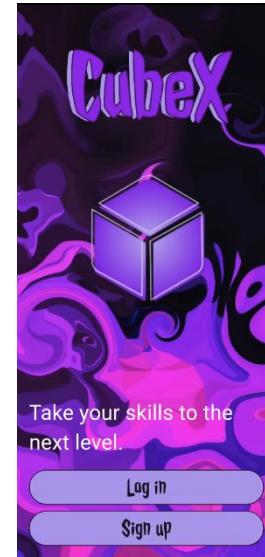
7.1.1 INICIO DE LA APLICACIÓN

Al abrir la aplicación, el usuario accede a la **pantalla de inicio**, desde donde puede comenzar su experiencia con la app. En esta pantalla se ofrecen dos opciones principales:

- **Iniciar sesión**, si ya se ha registrado con una cuenta.
- **Registrarse**, si es la primera vez que accede.

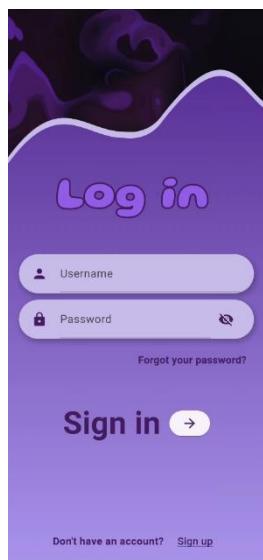
La aplicación está **adaptada al idioma seleccionado del usuario**. Si anteriormente seleccionó un idioma, la interfaz se mostrará automáticamente en ese idioma. En caso contrario, se utilizará el idioma predeterminado (inglés).

Si el usuario ya había iniciado sesión y **no cerró la sesión manualmente**, al abrir la aplicación será **redirigido directamente a la página principal**, conservando su información y sesión activa. En ese caso, no verá la pantalla de inicio.



7.1.2 Inicio de sesión

Al pulsar la opción **Iniciar sesión** en la pantalla de inicio, se mostrará un formulario donde el usuario debe introducir sus credenciales:



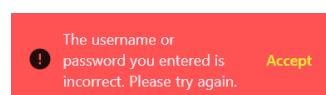
• Nombre de usuario

• Contraseña

Ambos campos son obligatorios.

Para facilitar su uso, se ofrece la opción de **mostrar u ocultar la contraseña** mediante un ícono o botón junto al campo.

Si el usuario introduce credenciales incorrectas o aún no tiene una cuenta, se mostrará un mensaje informativo.



En este último caso, también se ofrece un enlace directo a la pantalla de **registro** para crear una nueva cuenta.

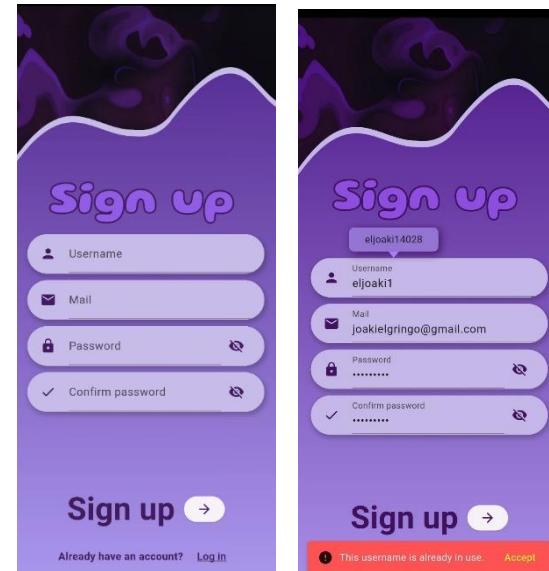
Si todo salió bien, redirigirá al usuario a la pantalla principal con el timer.

7.1.3 Crear cuenta

Al pulsar la opción de **registrarse**, se mostrará un formulario para que rellene con los datos correctos de nuevo usuario.

Dicho formulario deberá cumplir con los siguientes requisitos:

- ❖ Los campos del formulario son obligatorios.
- ❖ La contraseña introducida debe tener 8 caracteres y contener un carácter especial. Además debe coincidir con la confirmación. Para mayor comodidad, se ofrece la opción de mostrar o esconder las contraseñas.
- ❖ El nombre o el correo electrónico del usuario no esté previamente registrado.



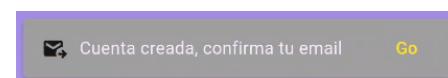
Si ya lo están, se mostrará un mensaje informando al usuario. En el caso del nombre de usuario, se le sugerirá automáticamente otra alternativa de nombre.



En la parte inferior del formulario, se incluye un enlace para los usuarios que **ya tienen una cuenta**, que les permite volver fácilmente a la pantalla de inicio de sesión.

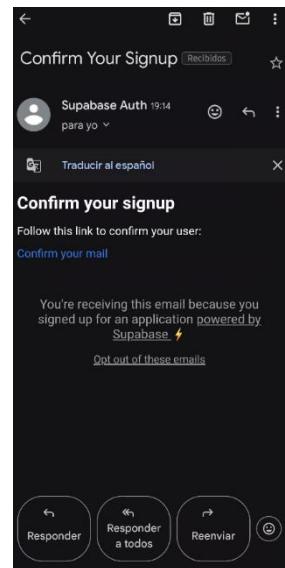
Confirmación del correo electrónico

Una vez completado correctamente el formulario, se enviará un **correo de confirmación** a la dirección proporcionada. Este correo contiene un botón para validar la cuenta.



El usuario podrá abrir el mensaje directamente desde la app, con acceso rápido tanto a la aplicación de Gmail como a la versión web.

Después de confirmar el correo, el usuario será **redirigido automáticamente a la pantalla principal**, donde verá un mensaje indicando que el registro se ha completado con éxito.



7.1.4 PÁGINA PRINCIPAL: TIMER

Una vez accede a la aplicación, el usuario llega a la **pantalla principal**, donde puede cronometrar sus tiempos de resolución para diferentes tipos de cubos. Esta funcionalidad se organiza por **sesiones**, que permiten al usuario guardar y revisar sus tiempos para cada tipo de cubo de forma independiente.

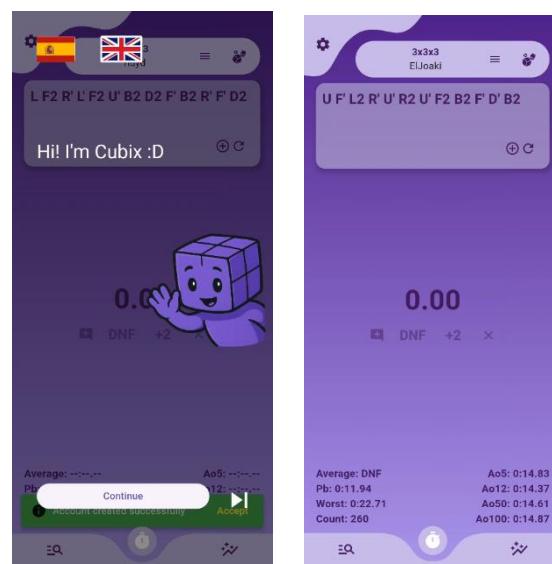
Primera vez en la aplicación

Cuando un usuario accede por primera vez tras registrarse, se muestra un **tutorial guiado por la mascota de la aplicación, Cubix**, explicado en el Anexo IV, que explica paso a paso los componentes que conforman las pantallas principales. Este tutorial puede ser completado o saltado.

Se aplica una **configuración inicial predeterminada**, en la que se crea automáticamente el tipo de cubo **3x3x3** con una sesión asociada. Esto permite al usuario comenzar a cronometrar sin necesidad de configurar nada manualmente.

Usuarios que ya han iniciado sesión anteriormente

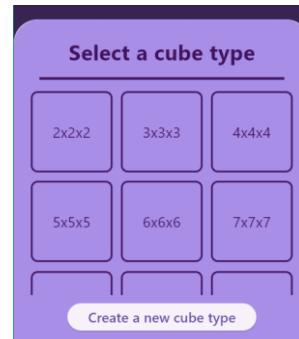
Si el usuario ya se ha registrado e inicia sesión, la aplicación cargará automáticamente el **primer tipo de cubo en la lista**, con su primera correspondiente sesión.



7.1.5 Seleccionar o crear tipo de cubo

En la pantalla principal, el usuario puede cambiar el tipo de cubo activo en cualquier momento. Para ello, debe pulsar el ícono situado en la parte superior derecha de la pantalla.

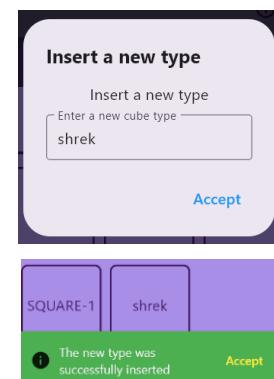
Al hacerlo, se despliega un menú con todos los tipos de cubo que el usuario posee. Desde ahí, se puede seleccionar cualquiera de ellos para empezar a cronometrar en su sesión correspondiente.



Crear un nuevo tipo de cubo

Además, desde este mismo menú, el usuario tiene la opción de **crear un nuevo tipo de cubo**. Para ello, deberá pulsar en “crear nuevo tipo de cubo” y llenar un breve formulario indicando el nombre del nuevo cubo y confirmar.

Una vez creado, el nuevo tipo de cubo aparecerá automáticamente en la lista y se le asignará una **sesión por defecto**, lista para ser utilizada.



Eliminar un tipo de cubo

Desde el menú de selección de tipos de cubo, el usuario puede **eliminar un tipo de cubo** manteniendo pulsado sobre el cubo que desea borrar.

Antes de completar la acción, se mostrará un mensaje de confirmación para evitar eliminaciones accidentales.

Si se confirma, el tipo de cubo seleccionado será eliminado junto con **todas las sesiones y los tiempos asociados** a él.

Si solo hay un tipo de cubo en la lista, no será posible eliminarlo ya que siempre habrá un tipo de cubo y una sesión activa como mínimo.

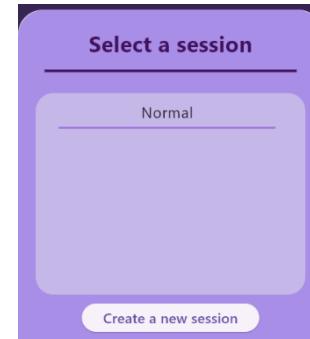
7.1.6 Seleccionar, crear o eliminar sesión

Junto al menú de tipos de cubo, el usuario encontrará el **ícono de gestión de sesiones**. Este permite acceder a una lista con todas las sesiones disponibles para el tipo de cubo actualmente seleccionado.



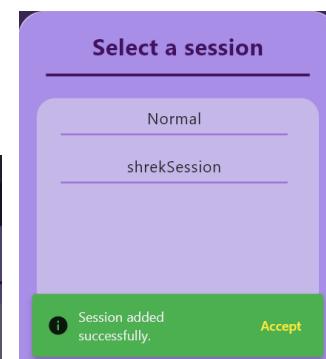
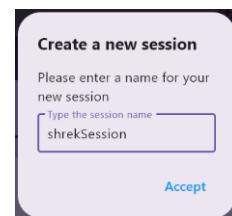
Seleccionar sesión

El usuario puede cambiar de sesión pulsando sobre cualquiera de las disponibles en la lista. Cada sesión almacena sus propios tiempos, asociados al tipo de cubo activo.



Crear una nueva sesión

Desde el mismo menú, se puede crear una nueva sesión. Para ello, el usuario debe pulsar el botón de “crear nueva sesión”, introducir un nombre de la sesión y confirmar. La nueva sesión se añadirá automáticamente a la lista.



Eliminar una sesión

Para eliminar una sesión, el usuario debe mantener pulsado sobre la sesión que desea borrar. Antes de eliminar, se mostrará un mensaje de confirmación. Si se acepta, la sesión se eliminará junto con todos los tiempos asociados.

Si solo hay una sesión disponible, no se podrá eliminar, ya que se requiere que haya siempre al menos una sesión activa por cada tipo de cubo.

Después de estos pasos, se mostrará que tipo de cubo y sesión es el actual:



7.1.7 Scramble

Justo debajo del selector de tipo de cubo y de sesión, se encuentra la sección de **scramble**.

Se trata de una **secuencia de movimientos generada automáticamente** que el usuario puede aplicar a su cubo antes de iniciar el cronómetro, con el objetivo de mezclarlo de forma aleatoria.

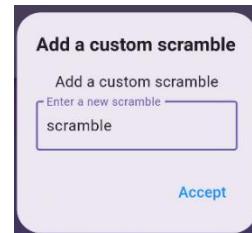
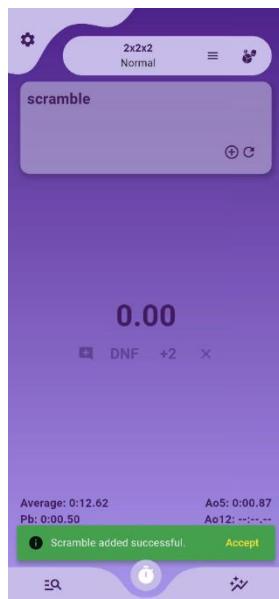
Funcionamiento automático

Cada vez que el usuario entra en la pantalla del timer o termina de registrar un tiempo, **se genera automáticamente un nuevo scramble**, correspondiente al tipo de cubo actualmente seleccionado.

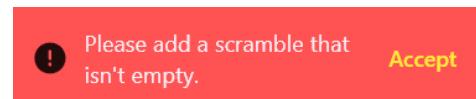
La aplicación incluye scrambles específicos para distintos tipos de cubo, como **2x2, 3x3, Megaminx, Pyraminx, Skewb y Square-1**, adaptando los movimientos según el cubo activo.

Scramble personalizado

El usuario también puede introducir manualmente un **scramble personalizado**. Para ello, debe pulsar el botón  y llenar el formulario con la secuencia de movimientos deseada.



Al confirmar, el nuevo scramble se establecerá automáticamente como el actual. Si el campo de scramble está vacío al confirmar, se mostrará un mensaje de error y no se aplicará ningún cambio.



Ademas, el usuario tambien tendrá la opcion de generar otro scramble pulsando el icono de recarga.

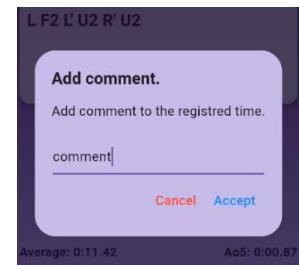
7.1.8 Timer

En el centro de la pantalla se encuentra el **cronómetro**, que se muestra inicialmente con el valor '**0.00**'. Este componente permite al usuario registrar sus tiempos de resolución de forma precisa y sencilla.

Funcionalidades adicionales:

Debajo del tiempo cronometrado, aparecen **botones de opciones** que permiten personalizar o gestionar la resolución. Estas opciones estarán **deshabilitadas hasta que el usuario registre un tiempo**. Una vez realizado una resolución, se habilitan las siguientes acciones:

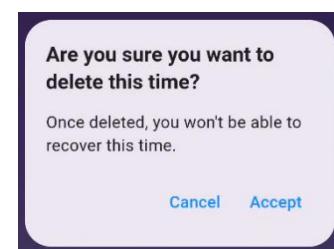
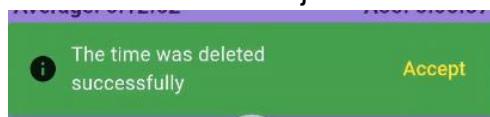
- ❖  **Añadir comentarios:** Permite escribir un comentario al tiempo realizado. Al añadir un comentario, en el tiempo aparecerá un ícono para indicar que el tiempo realizado lleva un comentario:



- ❖  **Añadir penalizaciones:** Se añade la penalización de Did Not Finish o la penalización que suma dos segundos al tiempo realizado. Solo se puede elegir una de las dos penalizaciones. Cuando se indica la penalización se muestra en el tiempo. Ademas, si el tiempo posee un comentario, aparecerán los dos iconos indicándolo:



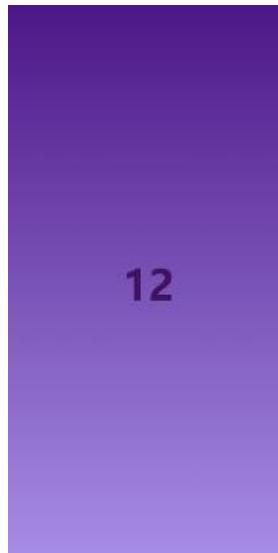
- ❖  **Eliminar tiempo:** Se elimina el tiempo realizado mostrando un mensaje de éxito.



7.1.8.1 Tiempo de inspección

La aplicación ofrece la opción de activar un **tiempo de inspección previo a la resolución**, tal como se hace en competiciones oficiales de speedcubing.

Cuando el tiempo de inspección está activado, el usuario deberá **tocar el cronómetro** para acceder a la pantalla de inspección. Para iniciar la cuenta atrás, se debe **mantener pulsado** el cronómetro: comenzará automáticamente la cuenta regresiva.



Penalizaciones (no implementadas aún)

Aunque no está disponible en la versión actual, se prevé que en futuras ampliaciones se apliquen penalizaciones automáticas según el tiempo de inspección:

- Si el usuario **superá el tiempo máximo en uno o dos segundos**, se añadirá una penalización de **+2 segundos** al tiempo de resolución.
- Si el usuario **superá los segundos establecidos más los dos segundos**, el tiempo será marcado como **DNF (Did Not Finish)** y se redirigirá automáticamente al cronómetro con ese resultado asignado.



+2



DNF

+ DNF +2 ×

7.1.8.2 Tiempo de resolución

Una vez finaliza el tiempo de inspección (si está activado), el usuario puede **iniciar la resolución del cubo**. Al **dejar de mantener pulsado** en la pantalla de inspección, el cronómetro comenzará a correr automáticamente, mostrando el tiempo transcurrido en segundos y décimas de segundo.

Cuando el usuario termina de resolver el cubo, debe **dar un toque en la pantalla** para detener el cronómetro.

Inmediatamente, se redirigirá a la **pantalla principal del timer**, donde se mostrará el **tiempo final de resolución** obtenido.

El cronómetro tiene un **límite de tiempo de 59 minutos y 59.9 segundos**. Si se alcanza este valor sin que el usuario detenga el cronómetro manualmente, se detendrá automáticamente.

7.5

7.1.8.3 Estadísticas

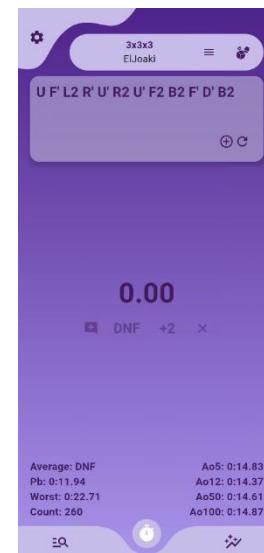
Ubicada en la parte inferior de la pantalla del timer, se encuentra una sección de **estadísticas** relacionadas con los tiempos registrados durante la sesión actual.

En esta sección se muestra el mejor y peor tiempo, la cantidad total de tiempos guardados en la sesión, y distintas medias. Estas incluyen la media total de todos los tiempos de la sesión, así como las medias de x tiempos más recientes registrados en la fecha actual.

Respecto a los **DNF** (*Did Not Finish*), tienen un trato aparte dentro del sistema de estadísticas. Los tiempos marcados como DNF no se tienen en cuenta para calcular el mejor o peor tiempo ni se incluyen en la media total (suele declararse como el peor tiempo).

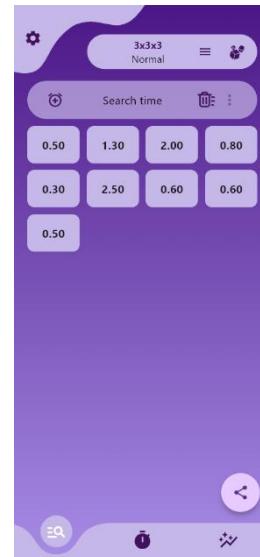
Además, para las medias de x tiempos más recientes, si entre los últimos x tiempos hay uno o más DNF, estos se omiten y se buscan tiempos anteriores válidos para completar el conjunto de x tiempos necesarios. Sin embargo, si hay dos o todos los últimos x tiempos son DNF, entonces la media se muestra como **DNF** también.

Si no hay x tiempos suficientes para la media de x tiempos, se mostrará ' $--:--.--$ '.



7.1.9 PANTALLA HISTORIAL

La pantalla de **historial** permite al usuario visualizar todos los tiempos registrados en cada sesión de un tipo de cubo.



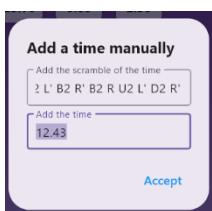
7.1.10 Seleccionar tipo de cubo o sesión

El sistema de gestión de tipos de cubos y sesiones es la misma que la pantalla del **timer**, por lo que el usuario podrá navegar entre los diferentes tipos de cubos y sesiones para ir visualizando el histórico de los tiempos de cada sesión elegida.

En la parte superior de la pantalla se encuentra la barra con los menús desplegables que permite al usuario navegar entre los distintos tipos de cubos (2x2, 3x3, Skewb, etc.) y sesiones, al igual que en la pantalla del temporizador. Al seleccionar una opción, se mostrará el listado correspondiente de tiempos registrados para esa sesión.

7.1.11 Añadir tiempo manualmente

Para registrar un nuevo tiempo de forma manual, el usuario  puede pulsar el botón ubicado a la izquierda de la barra de búsqueda. Se abrirá un formulario donde deberá introducir el **scramble** utilizado y el **tiempo de resolución**.

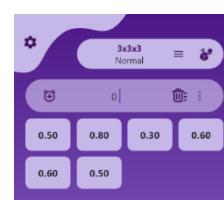


Una vez completado, al pulsar el botón de aceptar, el tiempo se añadirá automáticamente a la sesión actual y será visible en la lista. Esta funcionalidad es útil para registrar resultados obtenidos fuera de la aplicación o tiempos anteriores.



7.1.12 Búsqueda de tiempos

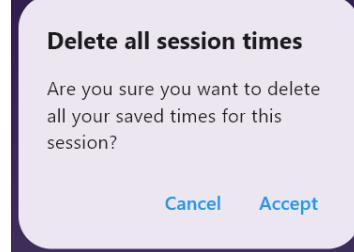
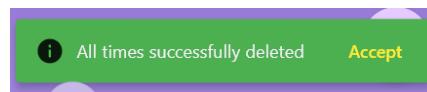
Junto al botón para añadir tiempos, existe una barra de búsqueda que permite filtrar los tiempos de la sesión actual. Al pulsar sobre ella, se activa un campo de texto con el mensaje "**Buscar por tiempo o comentario**". A medida que el usuario escribe, la lista de tiempos se actualiza en tiempo real mostrando solo aquellos que coincidan con el texto introducido, ya sea en el valor del tiempo o en algún comentario añadido.



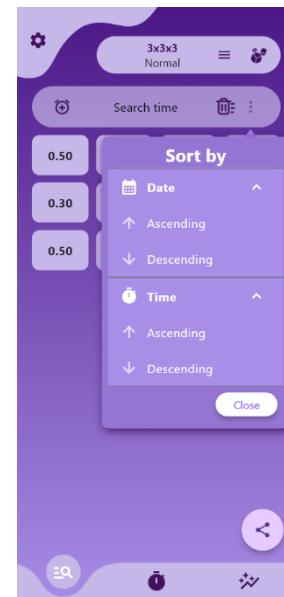
7.1.13 Opciones avanzadas

A la derecha del buscador se encuentran dos botones:

-  **Eliminar todos los tiempos de la sesión actual:** Representado por un ícono de papelera, permite borrar de forma inmediata todos los tiempos de la sesión actual. Aparecerá una alerta de confirmación y al confirmar se eliminarán los tiempos.



-  **Menú de filtros adicionales (ícono de tres puntos verticales):** Al pulsarlo, se despliega un cuadro de diálogo donde el usuario puede aplicar distintos filtros para ordenar la lista:
 - Por **fecha** (más reciente o antiguo)
 - Por **tiempo** (de menor a mayor o de mayor a menor)
 - Botón de **Cerrar** para salir del diálogo

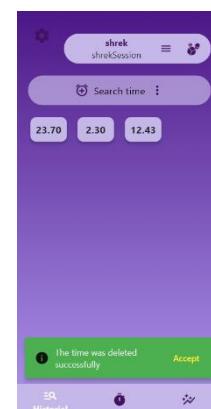


7.1.14 Visualización detalles de un tiempo

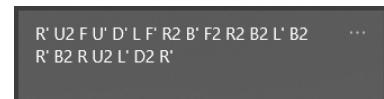
El usuario podrá ver detalles pulsando sobre un tiempo en concreto de la sesión actual. De esta forma, se visualizarán los detalles, con las opciones de **borrar** o **copiar** el scramble que se utilizó para realizar ese tiempo.



Para **borrar el tiempo**, es tan sencillo como pulsar el botón de la papelera ubicado en la esquina superior derecha y ya se eliminaría el tiempo. Se cierra el menú con los detalles y se actualiza la lista de tiempos.



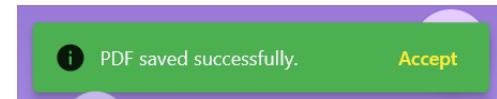
Para **copiar el scramble**, el usuario deberá pulsar el botón de copy ubicado  al lado del scramble. Se mostrará un mensaje de éxito y se podrá visualizar en el portapapeles.



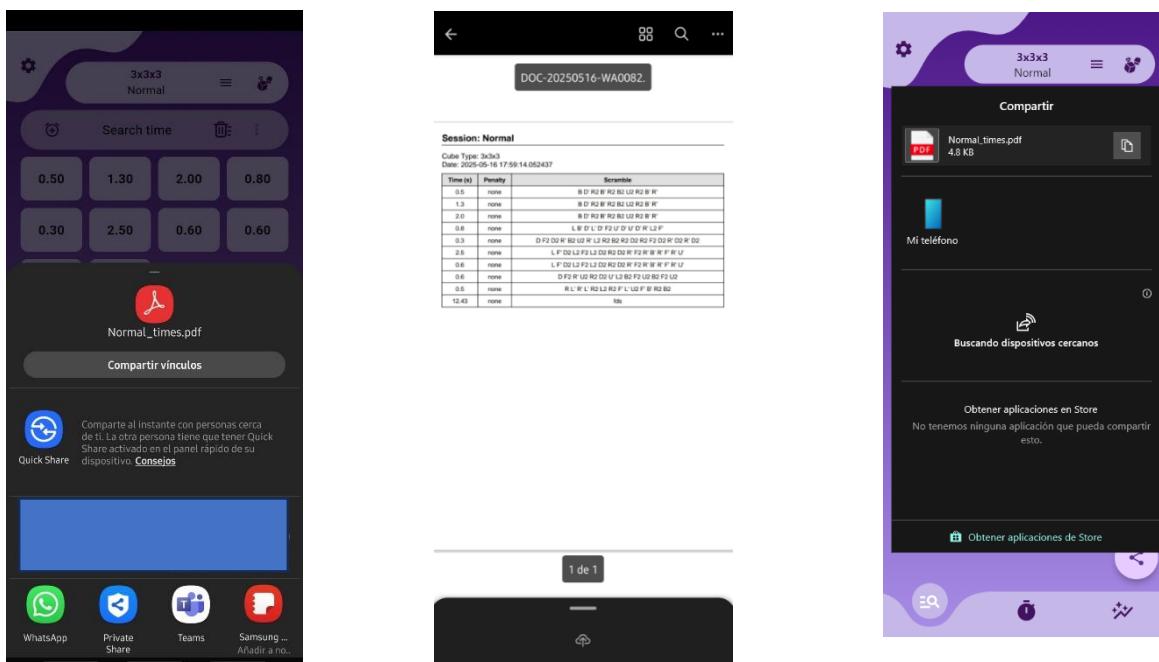
7.1.15 Compartir tiempos de la sesión

En la esquina inferior derecha, hay un botón flotante con el icono de **compartir**. Al pulsarlo, se abre una ventana con dos opciones:

- ❖ **Descargar en formato PDF:** Se genera un archivo PDF con todos los tiempos de la sesión actual. Con su mensaje de confirmación al descargar el PDF.
- ❖ **Compartir PDF:** Permite enviar ese PDF a través de otras aplicaciones, por ejemplo, en Android aplicaciones como correo, WhatsApp, Google Drive, etc.



En este archivo PDF contendrá el nombre de la sesión con el tipo de cubo, el día creado y una tabla diferenciando el tiempo de resolución, la penalización y el scramble.



7.1.16 Eliminación múltiple de tiempos

Si el usuario **mantiene pulsado** sobre un tiempo, este se selecciona y se marca con un fondo rojo. A partir de ahí, se pueden seguir seleccionando varios tiempos. En la parte inferior aparecerá un botón de **eliminar**, acompañado del número de elementos seleccionados, por ejemplo: **Eliminar (3)**



7.1.17 PANTALLA ESTADISTICAS

La pantalla de estadísticas ofrece al usuario una visión detallada y analítica del rendimiento obtenido en cada sesión de resolución de cubos de Rubik, viendo así un seguimiento visual del progreso a lo largo del tiempo.

Selección de tipo de cubo y sesión

En la parte superior de la pantalla se mantiene la barra principal que permite seleccionar el **tipo de cubo** y la **sesión específica** de la que se desean consultar las estadísticas.

Distintos tipos de estadísticas

1. Estadísticas Generales

En este primer bloque se muestran los datos más relevantes de forma resumida, enfocados a proporcionar una visión rápida del rendimiento del usuario en la sesión seleccionada:

- ❖ Mejor tiempo personal (PB – Personal Best)
- ❖ Peor tiempo registrado.
- ❖ Cantidad total de resoluciones.



- ❖ **Número de penalizaciones** (DNF (Did Not Finish) o +2 segundos).
- ❖ **Porcentaje de DNF y +2 sobre el total.**
- ❖ **Tiempo total de uso de la aplicación.**

2. Análisis de Promedios (Average Analysis)

A continuación, se presenta una **tabla de análisis de medias (averages)**. Esta sección ofrece un análisis de las medias de esa sesión:

- ❖ **Ao5** (Media de 5)
- ❖ **Ao12** (Media de 12)
- ❖ **Ao50** (Media de 50)
- ❖ **Ao100** (Media de 10)
- ❖ **AoTotal** (media de todos los tiempos de la sesión)

Para cada uno de estos valores, se muestran tres métricas:

- ❖ **Mejor media**
- ❖ **Peor media**
- ❖ **Media actual**

Estas medias se calculan quitando el mejor y peor tiempo y haciendo la media de los x numeros que queden.

3. Gráfico de evolución de tiempos

En la parte inferior de la pantalla, se incluye una **gráfica interactiva** que representa visualmente la evolución de los tiempos a lo largo del tiempo. Esta gráfica puede visualizarse a tres escalas temporales: anual, mensual y diaria.

Se ofrecen varias opciones de visualización para adaptarse a las preferencias del usuario:

- ❖ **Orientación de la gráfica:** Horizontal o vertical
- ❖ **Mostrar u ocultar valores numéricos** sobre los puntos del gráfico
- ❖ **Hay dos tipos de líneas:**
 - **Línea continua:** Representa todos los tiempos registrados.
 - **Línea discontinua:** Muestra los mejores tiempos alcanzados a lo largo del periodo seleccionado, ayudando a visualizar las mejoras progresivas.

- ❖ **Interacción:** Al tocar cualquier punto de la gráfica, se muestra un pequeño mensaje con la información del tiempo registrado y fecha de registro.

Si **no existen tiempos** registrados en la sesión seleccionada, se muestra un mensaje informativo "*Todavía no has registrado ningún tiempo en esta sesión.*". Si **existen datos**, se genera automáticamente la gráfica correspondiente con la configuración establecida por el usuario.

7.1.18 NAVEGACIÓN

La aplicación ofrece dos formas principales de navegación para facilitar el acceso a sus diversas funciones:

7.1.19 Navegación inferior

En la parte inferior de la pantalla se encuentra una **barra de navegación** que permite cambiar entre las diferentes secciones principales de la aplicación: **Temporizador, Historial y Estadísticas**.

Al pulsar sobre cualquiera de estas opciones, se realiza el cambio de pantalla correspondiente y se actualiza visualmente la barra de navegación: la opción seleccionada se resalta con un color blanco y se distingue del resto mediante un círculo que la rodea, mientras que las opciones no activas permanecen con un color morado oscuro.

Además de la navegación por botones, también se puede deslizar lateralmente (a la izquierda o a la derecha) sobre la pantalla para cambiar de una sección a otra.



7.1.20 Navegación con Drawer

Para acceder a más opciones, la aplicación cuenta con un **Drawer** (menú lateral) que se activa al pulsar el ícono de **Ajustes** ubicado en la esquina superior de la pantalla.

Este menú desplegable proporcionará datos del usuario como su foto de perfil, nombre y correo electrónico y dará acceso a una variedad de opciones y pantallas adicionales. Entre ellas están:

General

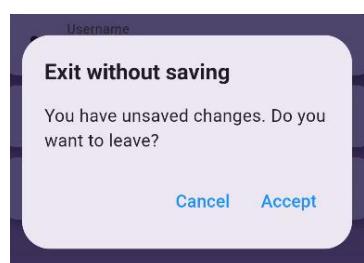
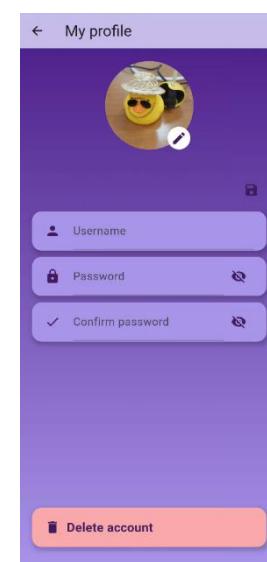
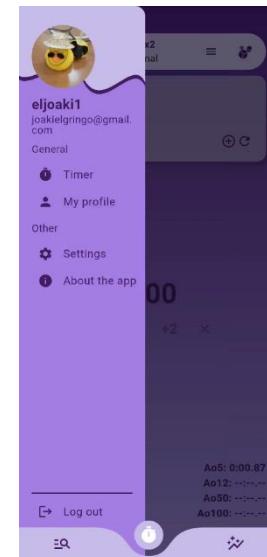
Timer

Redirige al usuario a la pantalla principal del temporizador.

Mi perfil

Lleva al usuario a la pantalla de gestión del perfil. Desde aquí se pueden realizar distintas acciones relacionadas con su cuenta:

- **Actualizar la foto de perfil**
- **Cambiar el nombre de usuario**
- **Modificar la contraseña** (se requiere introducir la contraseña antigua y la nueva dos veces para confirmación)
- **Realizar múltiples cambios a la vez** (nombre, contraseña y/o imagen)

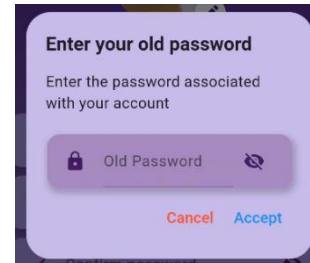


Al modificar cualquier campo, si el usuario intenta salir sin guardar, se mostrará una **alerta de confirmación** indicando que hay cambios sin guardar.

Al guardar los cambios, se solicitará la **contraseña actual** para verificar la identidad del usuario.

Para la selección de imagen:

- En **dispositivos de escritorio** se abrirá el explorador de archivos.



- En **dispositivos móviles** se abrirá la galería del sistema.

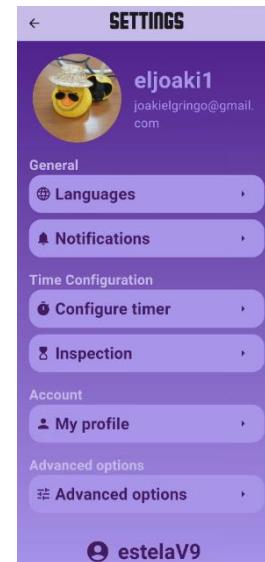
Además, desde esta pantalla se puede **eliminar la cuenta** del usuario. Esta opción se encuentra al final de la pantalla y requiere una confirmación antes de realizarse.

Otras opciones

Ajustes

Permite al usuario modificar diversas configuraciones de la aplicación:

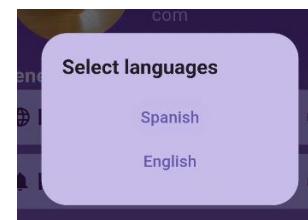
- Cambiar el **idioma de la aplicación**.
- Acceder nuevamente a la pantalla de “Mi perfil”.
- Activar o configurar **notificaciones**.
- Personalizar opciones relacionadas con el **timer** y la **inspección**.
- Acceder a configuraciones **avanzadas**.
- Redirigir nuevamente al perfil de GitHub del propietario.



Se muestra en más detalle las diferentes opciones:

- **Idioma**

Al pulsar esta opción, se mostrará una **alerta** con los dos idiomas disponibles: **español** e **inglés**. Al seleccionar uno de ellos, toda la interfaz de la aplicación se actualizará automáticamente al idioma escogido.

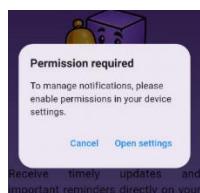


- Notificaciones

En este apartado, el usuario puede gestionar las notificaciones que desea recibir de la aplicación. Estas notificaciones están diseñadas mantener informado al usuario.

Activación de permisos

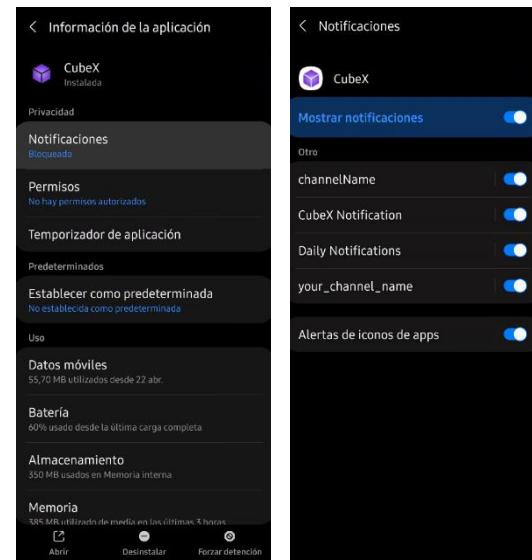
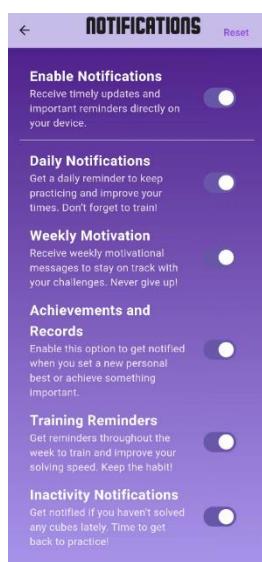
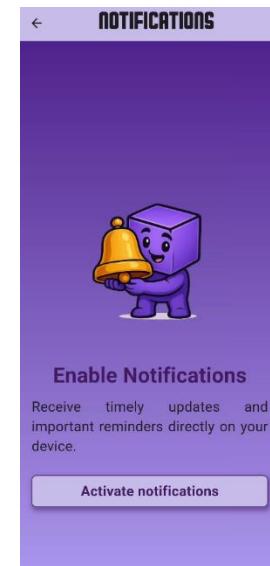
Al iniciar la aplicación por primera vez, algunos dispositivos mostrarán una alerta solicitando permiso para enviar notificaciones. Si el usuario deniega este permiso (o no aparece la alerta automáticamente), las notificaciones permanecerán desactivadas.



Cuando el usuario accede a esta sección y pulsa el botón para activar las notificaciones, se mostrará una **alerta de permisos requeridos**, con una opción para **redirigir a los ajustes del sistema**.

Allí deberá seleccionar la aplicación manualmente y **habilitar el permiso de notificaciones**.

Hasta que no se concedan estos permisos, no será posible activar las notificaciones desde la app.

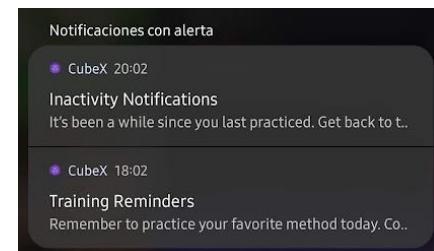
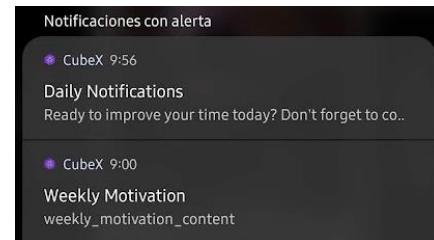


Una vez habilitadas correctamente, la aplicación enviará una **notificación de prueba con mensaje de éxito** y mostrará un mensaje dentro de la interfaz confirmando que las notificaciones han sido activadas satisfactoriamente.

Tipos de notificación disponibles

Una vez activadas, el usuario puede seleccionar individualmente qué tipo de notificaciones desea recibir:

- ❖ **Notificaciones diarias:** Se envían todos los días a las **9:00 a.m.**
- ❖ **Motivación semanal:** Mensaje motivacional enviado cada **lunes a las 9:00 a.m.**
- ❖ **Notificación de nuevo récord.**
- ❖ **Recordatorios para entrenar:** Se envían a las **6:00 p.m.**
- ❖ **Notificaciones de inactividad:** Se envían a las **8:00 p.m.**



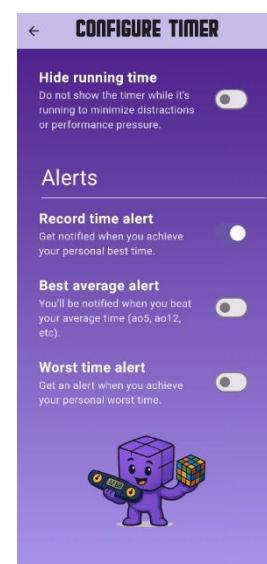
Botón de reinicio rápido

En la parte superior derecha de esta pantalla hay un **botón de reset** que restablece a una configuración por defecto, **activando únicamente las notificaciones diarias, notificaciones de nuevo récord y las notificaciones de inactividad**

- *Configuración del cronómetro*

En esta sección, el usuario puede personalizar el comportamiento del temporizador:

- ❖ **Mostrar/Ocultar el tiempo de resolución:**
Permite desactivar la visualización del tiempo durante la resolución, con el objetivo de **minimizar distracciones**.
- ❖ **Alertas de rendimiento:**
Notificar cuando ha batido su record personal, peor tiempo y mejor media, el cual no se ha implementado correctamente.



- ***Configuración de la inspección***

Permite gestionar los ajustes de la inspección previa a la resolución:

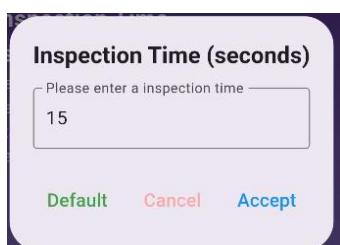
- **Habilitar/Deshabilitar la inspección:**

Si está **deshabilitada**, no se mostrará el tiempo de inspección antes de iniciar la resolución.

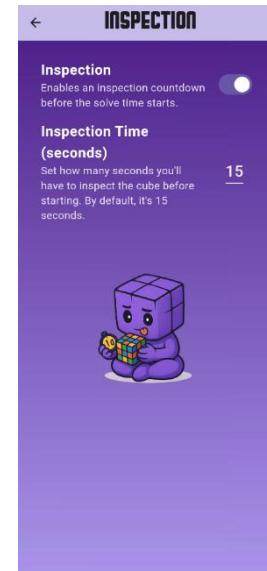
- **Personalización de segundos:**

Si la inspección está **habilitada**, se habilita un campo para definir el tiempo deseado para la inspección (por defecto 15 segundos).

Este campo incluye las siguientes **validaciones**:



- No puede estar vacío.
- Solo se permiten **valores numéricos**.
- El valor debe ser **menor o igual a 59** segundos.



Si la inspección está desactivada, esta opción se encuentra bloqueada.

- ***Opciones avanzadas***

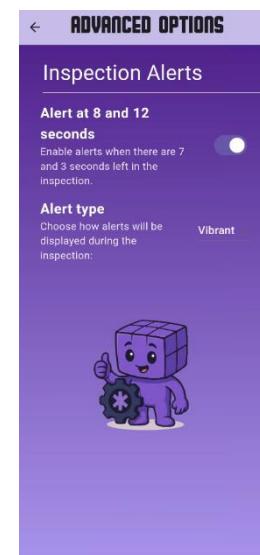
En esta sección se encuentran ajustes específicos para las **alertas durante la inspección**:

- **Activar/Desactivar alertas de inspección:**

Permite activar notificaciones al alcanzar los **8 y 12 segundos** de la cuenta regresiva de inspección.

- **Tipo de alerta (disponible solo si está activado):**

- **Vibración**
- **Sonido** (emitido según el idioma actual de la aplicación)
- **Ambos** (vibración + sonido)

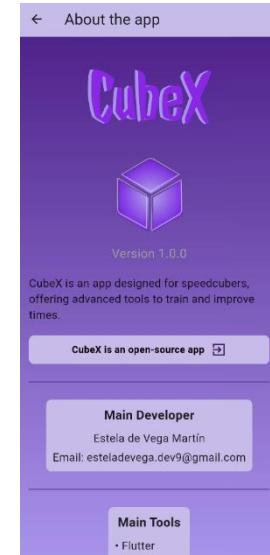


Si la opción de alertas está desactivada, no será posible seleccionar el tipo de alerta.

💡 Sobre la aplicación

Cuando el usuario pulsa esta opción, le redirigirá a la pantalla con información general sobre la aplicación. Desde esta pantalla:

- Se puede pulsar el ícono  para ver el código fuente del proyecto
- O ir abajo de la pantalla y pulsar el nombre de usuario de GitHub  **estelaV9** donde se le redirigirá al perfil de GitHub del desarrollador.



💡 Cerrar sesión

Finaliza la sesión del usuario y redirige a la pantalla de inicio de sesión.

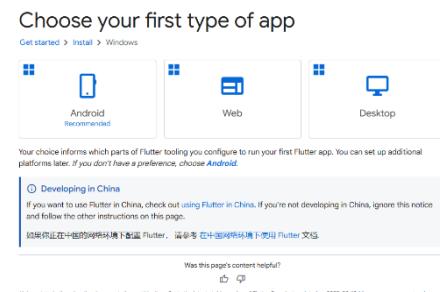
7.2 Manual de instalación

7.2.1 Instalar herramientas básicas

- Descargar e instalar **Git** desde <https://git-scm.com/downloads>
- Descargar e instalar un IDE como **IntelliJ IDEA** o **Visual Studio Code**
Ejemplo: <https://www.jetbrains.com/idea/download/?section=windows>

7.2.2 Instalacion y configuración de Flutter

1. Descargar [Flutter](#) según el sistema operativo que tenga el usuario, en este caso Windows.
2. Nos redirigira a la siguiente pagina donde seleccionaremos **Android**.



3. Posteriormente, nos lleva a esta [pantalla](#), donde descargaremos flutter.

Install the Flutter SDK

To install the Flutter SDK, you can use the VS Code Flutter exten:

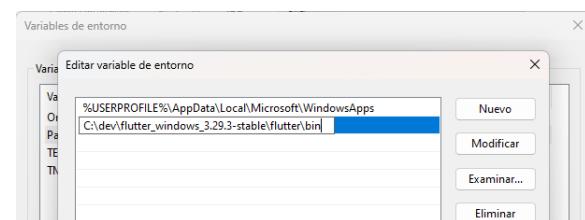
[Use VS Code to install](#) [Download and install](#)

Download then install Flutter

To install Flutter, download the Flutter SDK bundle from its archiv extract the SDK.

1. Download the following installation bundle to get the latest

[flutter_windows_3.29.3-stable.zip](#)



Instalacion

1. Extraemos Flutter en una carpeta, por ejemplo, C:\dev\flutter
2. Agregamos "C:\dev\flutter\bin" al PATH del usuario y del sistema
3. Verificamos la instalación en terminal con: **flutter**. Si se instala estaría bien implementado.

```
C:\Users\valentia.vegar>flutter
Building flutter tool...
Running pub upgrade...
Resolving dependencies... (1.0s)
Downloading packages... (10.0s)
Got dependencies.
Manage your Flutter app development.

Common commands:
  flutter create <output directory>
    Create a new Flutter project in the specified directory.

  flutter run [options]
    Run your Flutter application on an attached device or in an emulator.

Usage: flutter <command> [arguments]

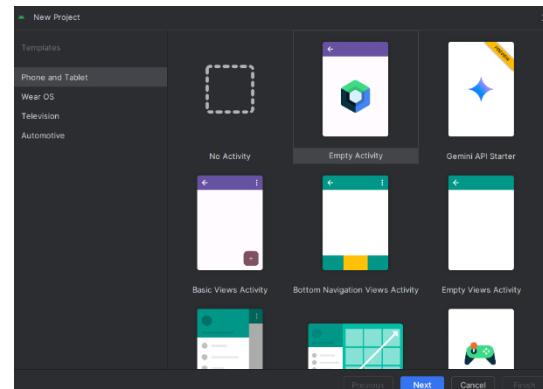
Global options:
  -h, --help          Print this usage information.
  -v, --verbose       Noisy Logging, including all shell commands executed.
                      If used with "--help", shows hidden options. If used with "flutter doctor", shows additional
                      diagnostic information. (Use "-vv" to force verbose logging in those cases.)
  -d, --device-id     Target device id or name (prefixes allowed).
  --version          Returns the version of this tool.

  --enable-analytics  Enable telemetry reporting each time a flutter or dart command runs.
  --disable-analytics Disable telemetry reporting each time a flutter or dart command runs, until it is
                        re-enabled.
  --suppress-analytics Suppress analytics reporting for the current CLI invocation.
```

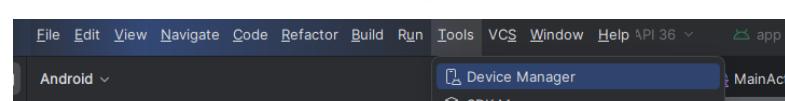
7.2.3 Configurar Android Studio

Ahora vamos a configurar el entorno de Android, para ello deberemos tener instalado Android Studio. Antes de ello podemos ver los requisitos previos para instalarlo.

Cuando lo tengamos instalado, crearemos un proyecto para configurarlo, con el nombre que queramos.



Crearemos un dispositivo virtual en: Tools > Device Manager. Nos aparecerán los distintos dispositivos que tenemos, si sale vacío creamos un nuevo servicio.

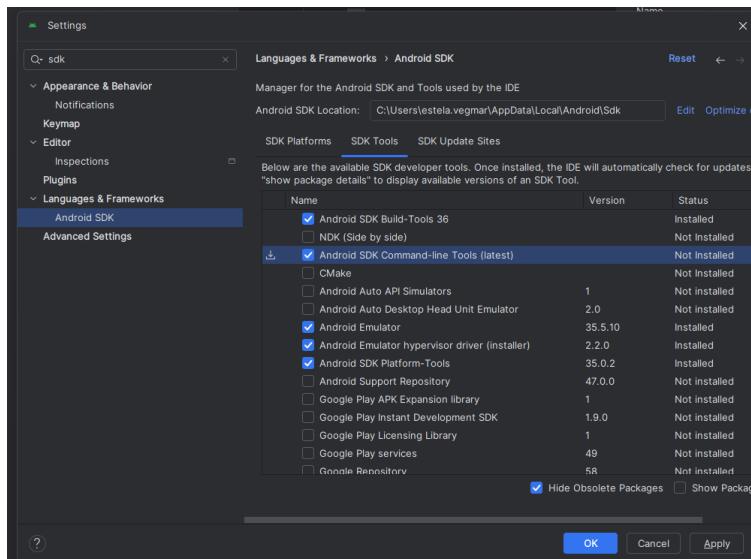


Instalar Android SDK Command-line Tools (latest) para poder emular en Android si es que no lo tenemos instalado

```
To disable animations in this tool, use
'flutter config --no-cli-animations'.

Android sdkmanager not found. Update to the latest Android SDK and ensure that the cmdline-tools are installed to
resolve this
```

Nos vamos Settings > Android SDK > SDK Tools y lo instalamos.



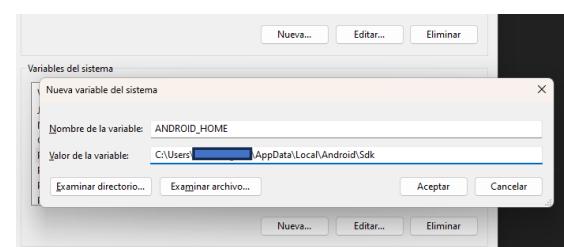
Configuramos las variables de entorno:

Creamos una nueva variable de entorno:

- Nombre de la variable de entorno: ANDROID_HOME
- Valor: C:\Users\<TU_USUARIO>\AppData\Local\Android\Sdk

Y añadimos al Path estados dos rutas:

```
%ANDROID_HOME%\cmdline-tools\latest\bin
%ANDROID_HOME%\platform-tools
```



Aceptamos y reiniciamos la terminal.

Verificamos en una nueva terminal con: echo %ANDROID_HOME%

```
C:\Users\estela.vegmar>echo %ANDROID_HOME%
C:\Users\estela.vegmar\AppData\Local\Android\Sdk
```

Podemos hacer una prueba con sdkmanager --list, si eso funciona podemos hacer el siguiente paso.

```
C:\Users\estela.vegmar>sdkmanager --list
[=====] 100% Computing updates...
Installed packages:
Path                                | Version | Description
----- | ----- | -----
| --- | ---
```

Una vez que ANDROID_HOME esté correctamente configurado, Flutter podrá usar sdkmanager, ejecutar y aceptamos las licencias.

flutter doctor --android-licenses

```
August 15, 2011
-----
Accept? (y/N): y
```

Ya estaría configurado flutter correctamente, lo podemos observar haciendo un flutter doctor:

```
C:\Users\estela.vegmar>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[!] Flutter (Channel stable, 3.29.3, on Microsoft Windows [Version 10.0.22631.4899], locale es-ES)
[!] Windows Version (11 Pro 64-bit, 23H2, 2009)
[!] Android toolchain - develop for Android devices (Android SDK version 36.0.0)
[!] Chrome - develop for the web
[!] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.11.5)
[!] Android Studio (version 2024.1)
[!] IntelliJ IDEA Community Edition (version 2025.1)
[!] VS Code, 64-bit edition (version 1.97.2)
[!] Connected device (2 available)
[!] Network resources

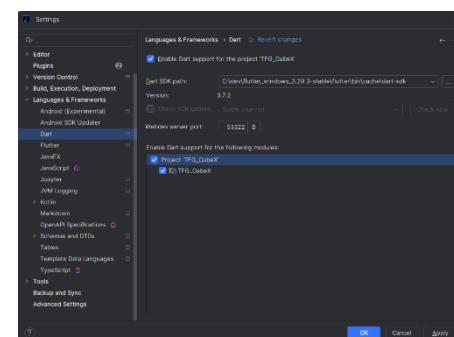
• No issues found!
```

7.2.4 Configuración en IntelliJ

Activar Dart en: File > Settings > Languages & Frameworks > Dart y marcamos la opción enable dart support for the project

Ruta del SDK: C:\dev\flutter\bin\cache\dart-sdk y elegimos el modulo del proyecto

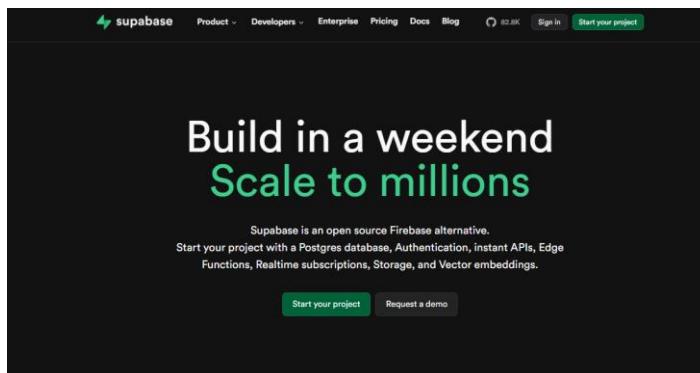
Instalar los plugins de Dart y Flutter desde Plugins



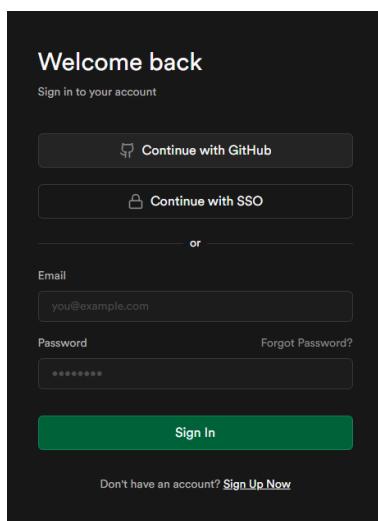
7.2.5 Configurar Supabase

7.2.5.1 Crear Proyecto

Una vez configurado el entorno de flutter, nos crearemos un nuevo proyecto.



Nos creamos una cuenta o nos logeamos



Creamos una nueva organización y un nuevo proyecto (ej: CubeX)

Create a new organization

This is your organization within Supabase.
For example, you can use the name of your company or department.

Name	CubeX
Type	Personal
Plan	Free - \$0/month

Pricing 

The Plan applies to your new organization.

[Cancel](#) You can rename your organization later [Create organization](#)

Create a new project

Your project will have its own dedicated instance and full Postgres database. An API will be set up so you can easily interact with your new database.

Organization	CubeX Free
Project name	CubeX
Database Password	***** Copy
Region	West Europe (London)

SECURITY OPTIONS 

ADVANCED CONFIGURATION 

[Cancel](#) [Create new project](#)



IES RIBERA DE CASTILLA



Obtenemos Project URL y API Key desde settings > API

The screenshot shows the 'API Settings' section of the Supabase dashboard. Under 'Project URL', the URL is listed as <https://jsrszptbmsbrmayfaet.supabase.co>. Under 'Project API Keys', there are two keys: 'anon' (public) and 'service_role' (secret). The 'service_role' key is highlighted with a red border. A note below states: 'This key has the ability to bypass Row Level Security. Never share it publicly. If leaked, generate a new key & secret immediately.' A button 'Read the announcement' is visible at the bottom right.

Volvemos al proyecto y crearemos un archivo .env con las siguientes variables de la url del proyecto y la clave.

The screenshot shows a code editor with a file named '.env'. It contains the following content:

```
SUPABASE_URL=https://jsrszptbmsbrmayfaet.supabase.co
SUPABASE_ANON_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVCJ9.eyJ
```

7.2.5.2 Configurar base de datos:

- Ejecutar SQL en SQL Editor y nos debería poner "Success. No rows returned" para que se haya ejecutado bien.

The screenshot shows the 'SQL Editor' tab in the Supabase dashboard. The code being run is:

```
-- SE USA COMILLAS YA QUE ES UNA PALABRA RESERVADA
CREATE TABLE "user" (
    idUser SERIAL PRIMARY KEY,
    userUUID UUID UNIQUE NOT NULL,
    username TEXT NOT NULL UNIQUE, -- EL NOMBRE DEL USUARIO SERA UNICO Y DE
    mail TEXT NOT NULL UNIQUE, -- EL MAIL SERA UNICO
    passwordHash TEXT NOT NULL,
    creationDate TIMESTAMP NOT NULL,
    imageUrl TEXT NOT NULL /* nota: por defecto tendra una imagen predeterminada */
);

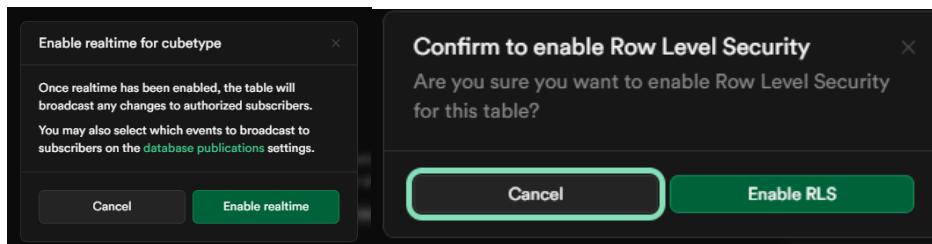
CREATE TABLE cubeType (
    idCubeType SERIAL PRIMARY KEY,
    cubeName TEXT NOT NULL, -- EL NOMBRE DEL CUBO SERA UNICO
    -- CADA USUARIO TENDRA SUS TIPOS DE CUBOS
    idUser INTEGER NOT NULL REFERENCES "user"(idUser) ON DELETE CASCADE,
    -- ASSEGURAR QUE UN USUARIO NO TENGA MAS DE UN TIPO DE CUBO CON EL MISMO NOMBRE
    UNIQUE (idUser, cubeName)
);

CREATE TABLE sessionTime (
    idSession serial PRIMARY KEY,
    idUser INTEGER NOT NULL REFERENCES "user"(idUser),
    sessionName TEXT NOT NULL,
    creationDate TIMESTAMP NOT NULL,
    idCubeType INTEGER NOT NULL REFERENCES cubeType(idCubeType),
    -- ASSEGURAR QUE UN USUARIO NO PUEDA TENER DOS SESIONES CON EL MISMO NOMBRE
    UNIQUE (idUser, sessionName, idCubeType)
);
```

The results section shows the message: 'Success. No rows returned'

- Habilitar RLS y el tiempo real en cada tabla

The screenshot shows the 'Realtime' settings in the Supabase dashboard. The 'Role' dropdown is set to 'postgres'. A button 'RLS disabled' is shown in a yellow box. Other options like 'Realtime off' are also visible.



- Añadir políticas (Policies) en Authentication > Policies con condiciones USING para SELECT, INSERT, DELETE, UPDATE, ALL en cada tabla.

Para el all aunque es fallo de seguridad, los Deep links en Windows son complejos asi que es la única forma que ejecute en Windows, es una de las ampliaciones investigar mas este tema

CubeType

(EXISTS (SELECT 1 FROM "user" WHERE ((user".iduser = cubetype.iduser)
AND ("user".useruuid = auth.uid()))))

DELETE	INSERT	SELECT	ALL
<pre>Policy Name: Delete cubetype for authenticated user Table: on clause Delete cubetype for authenticated user public.cubetype Policy Behavior: as clause Permissive: Policy Command: for clause SELECT [] INSERT [x] UPDATE [] DELETE [] ALL [] Target Roles: to clause authenticated: x void options_scope('on clause'); 1 create policy "Delete cubetype for authenticated user" 2 on "public","cubetype" 3 as PERMISSIVE 4 for DELETE 5 to authenticated 6 using (7 (EXISTS (SELECT 1 8 FROM "user" 9 WHERE ((user".iduser = cubetype.iduser) AND ("user".useruuid = auth.uid()))) 10));</pre>	<pre>Create a new Row Level Security policy Policy Name: Insert cubetype for user authenticated Table: on clause Insert cubetype for user authenticated public.cubetype Policy Behavior: as clause Permissive: Policy Command: for clause SELECT [] INSERT [x] UPDATE [] DELETE [] ALL [] Target Roles: to clause authenticated: x USE OPTIONSCOPE('on clause'); 1 create policy "Insert cubetype for user authenticated" 2 on "public","cubetype" 3 as PERMISSIVE 4 for INSERT 5 to authenticated 6 using (7 (EXISTS (SELECT 1 8 FROM "user" 9 WHERE ((user".iduser = cubetype.iduser) AND ("user".useruuid = auth.uid()))) 10));</pre>	<pre>Create a new Row Level Security policy Policy Name: Enable realtime for cubetype Table: on clause Select cubetype for user authenticated public.cubetype Policy Behavior: as clause Permissive: Policy Command: for clause SELECT [] INSERT [x] UPDATE [] DELETE [] ALL [] Target Roles: to clause authenticated: x USE OPTIONSCOPE('on clause'); 1 create policy "Enable realtime for cubetype" 2 on "public","cubetype" 3 as PERMISSIVE 4 for ALL 5 using (6 (SELECT 1 7 FROM "user" 8 WHERE ((user".iduser = cubetype.iduser) AND ("user".useruuid = auth.uid()))) 9);</pre>	<pre>Create a new Row Level Security policy Policy Name: Enable all actions for cubetypes Table: on clause Enable all actions for cubetypes public.cubetype Policy Behavior: as clause Permissive: Policy Command: for clause SELECT [] INSERT [x] UPDATE [] DELETE [] ALL [] Target Roles: to clause Delete all (public) roles if none selected 1 use option scope('on clause'); 2 create policy "Enable all actions for cubetypes" 3 on "public","cubetype" 4 as PERMISSIVE 5 for ALL 6 using (7 (SELECT 1 8 FROM "user" 9 WHERE ((user".iduser = cubetype.iduser) AND ("user".useruuid = auth.uid()))) 10);</pre>



SessionTime

(EXISTS (SELECT 1 FROM "user" WHERE ((user".iduser = sessiontime.iduser) AND ("user".useruuid = auth.uid()))))

DELETE	INSERT	SELECT	ALL
<pre> Policy Name: Delete session to user authenticated Table: on clause Policy Behavior: as clause Permissive Policy Command: for clause SELECT INSERT UPDATE DELETE ALL Target Roles: no clause authenticated > authenticated > 1 USE OPTIONS ABOVE TO EDIT 2 create policy "Delete session to user authenticated" 3 on "public"."sessiontime" 4 as PERMISSIVE 5 to public 6 using { 7 SELECT 8 INSERT 9 UPDATE 10 DELETE 11 ALL 12 iduser = user.iduser AND user.useruuid = auth.uid(); 13 } 14 </pre>	<pre> Policy Name: Insert sessiontime for authenticated user Table: on clause Policy Behavior: as clause Permissive Policy Command: for clause SELECT INSERT UPDATE DELETE ALL Target Roles: no clause authenticated > 1 USE OPTIONS ABOVE TO EDIT 2 alter policy "Insert sessiontime for authenticated user" 3 on "public"."sessiontime" 4 as PERMISSIVE 5 to authenticated 6 using { 7 SELECT 8 INSERT 9 UPDATE 10 DELETE 11 ALL 12 iduser = user.iduser AND user.useruuid = auth.uid(); 13 } 14 </pre>	<pre> Policy Name: Select time to user authenticated Table: on clause Policy Behavior: as clause Permissive Policy Command: for clause SELECT INSERT UPDATE DELETE ALL Target Roles: no clause authenticated > 1 USE OPTIONS ABOVE TO EDIT 2 alter policy "Select time to user authenticated" 3 on "public"."sessiontime" 4 as PERMISSIVE 5 to authenticated 6 using { 7 SELECT 8 INSERT 9 UPDATE 10 DELETE 11 ALL 12 iduser = user.iduser AND user.useruuid = auth.uid(); 13 } 14 </pre>	<pre> Policy Name: Delete all rows for sessions Table: on clause Policy Behavior: as clause Permissive Policy Command: for clause SELECT INSERT UPDATE DELETE ALL Target Roles: no clause Defaults to all (public) roles if none selected 1 USE OPTIONS ABOVE TO EDIT 2 create policy "Delete all rows for sessions" 3 on "public"."sessiontime" 4 as PERMISSIVE 5 to public 6 using { 7 SELECT 8 INSERT 9 UPDATE 10 DELETE 11 ALL 12 iduser = user.iduser AND user.useruuid = auth.uid(); 13 } 14 </pre>

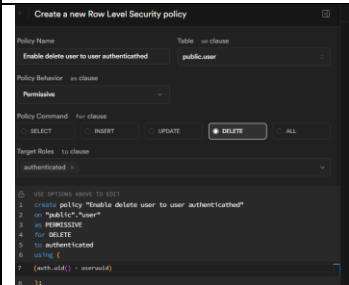
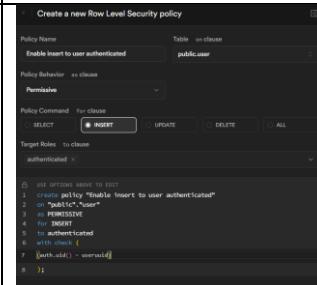
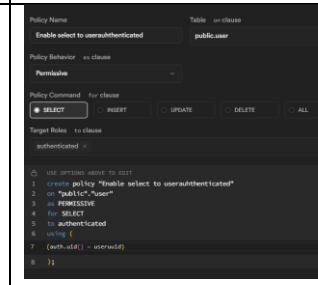
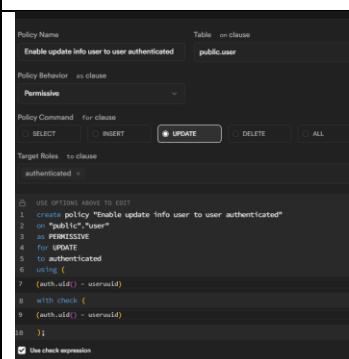
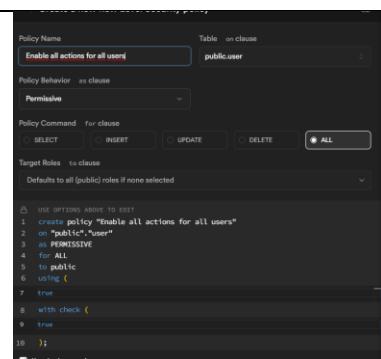
TimeTraining

(EXISTS (SELECT 1 FROM sessiontime, "user" WHERE ((sessiontime.idsession = timetraining.idsession) AND (sessiontime.iduser = "user".iduser) AND ("user".useruuid = auth.uid()))))

DELETE	INSERT	SELECT	UPDATE	ALL
<pre> Create a new Row Level Security policy Policy Name: Enable delete times to user authenticated Table: on clause Policy Behavior: as clause Permissive Policy Command: for clause SELECT INSERT UPDATE DELETE ALL Target Roles: no clause Defaults to all (public) roles if none selected 1 USE OPTIONS ABOVE TO EDIT 2 alter policy "Enable delete times to user authenticated" 3 on "public"."timetraining" 4 as PERMISSIVE 5 to public 6 using { 7 SELECT 8 INSERT 9 UPDATE 10 DELETE 11 ALL 12 idsession = sessiontime.idsession AND sessiontime.iduser = user.iduser AND user.useruuid = auth.uid(); 13 } 14 </pre>	<pre> Policy Name: Enable insert times to user authenticated Table: on clause Policy Behavior: as clause Permissive Policy Command: for clause SELECT INSERT UPDATE DELETE ALL Target Roles: no clause authenticated > 1 USE OPTIONS ABOVE TO EDIT 2 alter policy "Enable insert times to user authenticated" 3 on "public"."timetraining" 4 as PERMISSIVE 5 to authenticated 6 using { 7 SELECT 8 INSERT 9 UPDATE 10 DELETE 11 ALL 12 idsession = sessiontime.idsession AND sessiontime.iduser = user.iduser AND user.useruuid = auth.uid(); 13 } 14 </pre>	<pre> Policy Name: Enable select time to user authenticated Table: on clause Policy Behavior: as clause Permissive Policy Command: for clause SELECT INSERT UPDATE DELETE ALL Target Roles: no clause authenticated > 1 USE OPTIONS ABOVE TO EDIT 2 alter policy "Enable select time to user authenticated" 3 on "public"."timetraining" 4 as PERMISSIVE 5 to authenticated 6 using { 7 SELECT 8 INSERT 9 UPDATE 10 DELETE 11 ALL 12 idsession = sessiontime.idsession AND sessiontime.iduser = user.iduser AND user.useruuid = auth.uid(); 13 } 14 </pre>	<pre> Create a new Row Level Security policy Policy Name: Enable update to user authenticated Table: on clause Policy Behavior: as clause Permissive Policy Command: for clause SELECT INSERT UPDATE DELETE ALL Target Roles: no clause Defaults to all (public) roles if none selected 1 USE OPTIONS ABOVE TO EDIT 2 alter policy "Enable update to user authenticated" 3 on "public"."timetraining" 4 as PERMISSIVE 5 to public 6 using { 7 SELECT 8 INSERT 9 UPDATE 10 DELETE 11 ALL 12 idsession = sessiontime.idsession AND sessiontime.iduser = user.iduser AND user.useruuid = auth.uid(); 13 } 14 </pre>	<pre> Policy Name: Enable all actions to timeTraining Table: on clause Policy Behavior: as clause Permissive Policy Command: for clause SELECT INSERT UPDATE DELETE ALL Target Roles: no clause Defaults to all (public) roles if none selected 1 USE OPTIONS ABOVE TO EDIT 2 create policy "Enable all actions to timeTraining" 3 on "public"."timetraining" 4 as PERMISSIVE 5 for ALL 6 to public 7 using { 8 true 9 with check { 10 true 11 } 12 } 13 </pre>

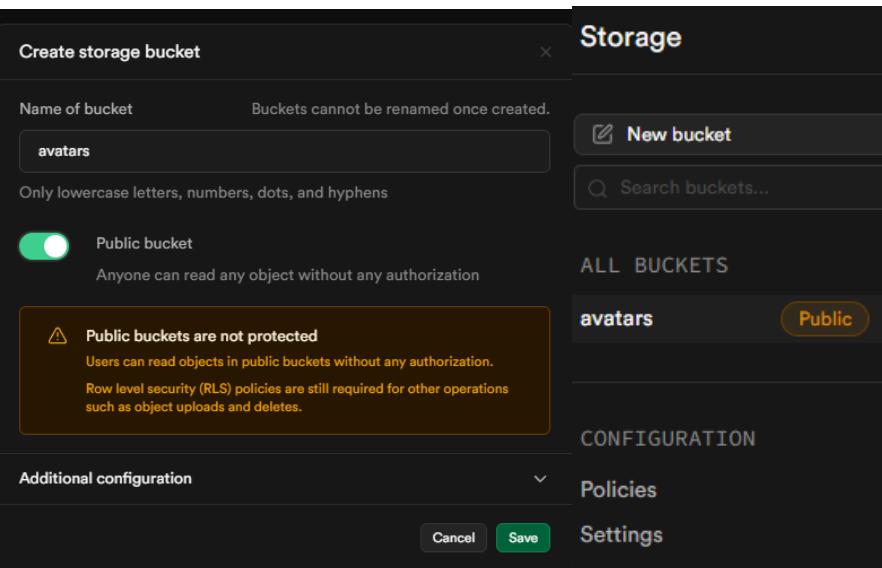
User

(auth.uid() = useruuid)

DELETE	INSERT	SELECT
		
UPDATE	ALL	
		

7.2.5.3 Configurar Storage:

Creamos un bucket público llamado "avatars" para poder subir las fotos.



Create storage bucket

Storage

Name of bucket Buckets cannot be renamed once created.
avatars

Only lowercase letters, numbers, dots, and hyphens

Public bucket Anyone can read any object without any authorization

Public buckets are not protected
Users can read objects in public buckets without any authorization.
Row level security (RLS) policies are still required for other operations such as object uploads and deletes.

ALL BUCKETS

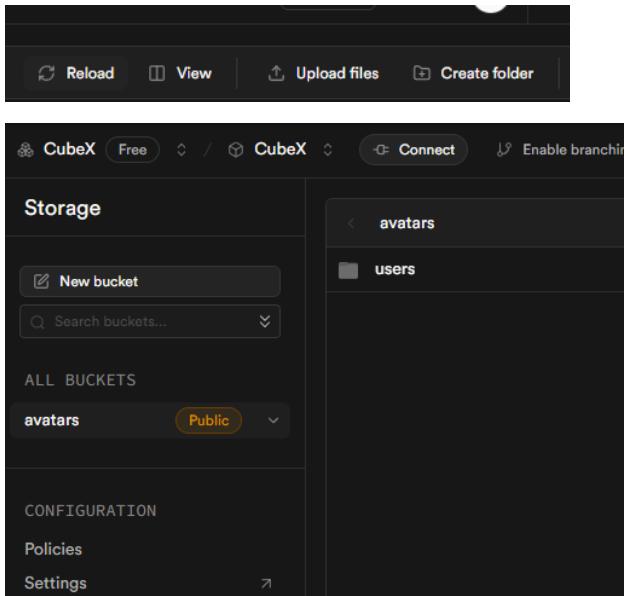
avatars **Public**

CONFIGURATION

Additional configuration ▾ **Policies** **Settings**

Cancel **Save**

- Agregamos la carpeta "users" dentro del bucket

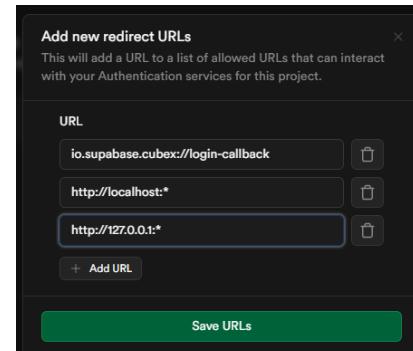


The screenshot shows the CubeX storage interface. On the left, there's a sidebar with 'Storage' and 'ALL BUCKETS' sections. Under 'ALL BUCKETS', there's a bucket named 'avatars' which is set to 'Public'. On the right, the main area shows a list of files and folders. Inside the 'avatars' bucket, there is a folder named 'users'.

7.2.5.4 Configurar URLs de redirección:

Por ultimo, configuraremos las urls de redireccion, ubicadas en authentication. En redirect urls añadimos lo siguiente:

- io.supabase.cubex://login-callback
- http://localhost:*
- http://127.0.0.1:*



The screenshot shows a modal dialog titled 'Add new redirect URLs'. It contains a list of URLs: 'io.supabase.cubex://login-callback', 'http://localhost:*', and 'http://127.0.0.1:*'. There is a green 'Save URLs' button at the bottom.

7.2.6 Clonar y ejecutar el proyecto

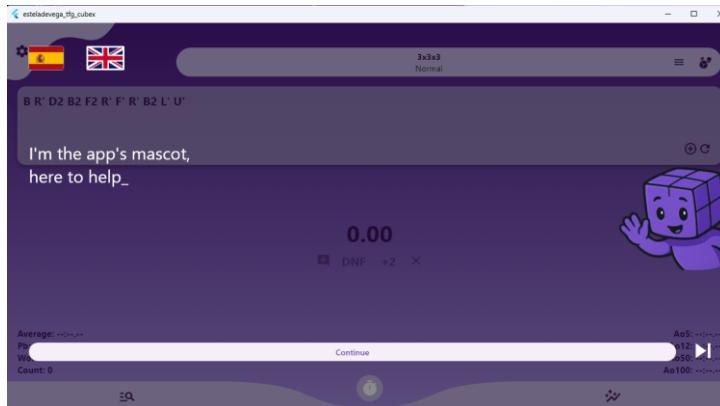
Clonar repositorio: https://github.com/estelaV9/TFG_CubeX.git

Abrir en IntelliJ, instalar dependencias y ejecutar:

- flutter doctor (para asegurar que esta correctamente)
- flutter pub get
- flutter run



IES RIBERA DE CASTILLA



8 Conclusiones y posibles ampliaciones

8.1 Conclusiones

Despues de desarrollar este proyecto durante medio año, dedicando cada hora disponible, puedo decir que me siento muy orgullosa del trabajo realizado. Desde el planteamiento inicial, el **objetivo principal fue desarrollar una aplicación funcional** que ofreciera una experiencia completa a los speedcubers, y **considero que este objetivo se ha cumplido en gran medida**. Aunque no todas las funcionalidades previstas pudieron implementarse por limitaciones de tiempo, el resultado final es plenamente funcional y satisfactorio.

He conseguido crear una **aplicación funcional y atractiva para speedcubers**, tanto principiantes como avanzados, o incluso para cualquier persona interesada en aprender, buscando siempre ofrecer una experiencia de usuario personalizada. He logrado implementar un temporizador con penalizaciones y una lógica estable que gestiona la relación entre tipos de cubos y sesiones. También incluí *opciones de personalización*, como aumentar el tiempo de inspección, ocultar el tiempo durante la resolución o añadir alertas de diferente tipo en la inspección, entre otras.

Además, integré un sistema de notificaciones y utilicé Supabase como backend, permitiendo que los datos se mantengan sincronizados entre plataformas. A nivel visual, considero que la aplicación ha acabado siendo atractiva para el usuario.

Una parte especialmente gratificante fue dar un toque cercano y personal mediante la creación de la mascota de la app, **Cubix**, quien guía al usuario a través de un tutorial inicial, e incluso ha servido como inspiración para diseñar merchandising.

Durante el proceso me **enfrenté a múltiples retos**, como la integración de una API externa, el sistema de notificaciones, el diseño de formas onduladas personalizadas (modificando píxel por píxel) o la implementación de funcionalidades para las que apenas existían referencias, como el sistema del tutorial guiado. Sin embargo, **he disfrutado y aprendido con cada línea de código programada**.

En cuanto al aprendizaje técnico, **he consolidado mis conocimientos en Dart** y he aprendido a trabajar con herramientas avanzadas como **CustomPainter**, **SmartDialog** o la integración de **Supabase**. Además, he afianzado el uso del control de versiones y he adquirido nuevas habilidades tanto prácticas como técnicas.

Possibles ampliaciones

A futuro, se plantean varias ampliaciones que enriquecerían notablemente la experiencia del usuario:

- **Implementación de un chat en tiempo real:** permitiría conectar a los usuarios dentro de la misma app, favoreciendo la creación de comunidad mediante publicaciones, tutoriales, grupos de resolución y mensajería directa, eliminando la necesidad de utilizar plataformas externas. No se pudo implementar por la falta de tiempo.
- **Modo versus local:** una funcionalidad que permita a dos personas utilizar la misma app en un único dispositivo para competir en tiempo real, compartiendo pantalla y timers independientes. Esto fomentaría la interacción cara a cara sin necesidad de varios dispositivos, lo que sería ideal para encuentros presenciales o competiciones informales.
- **Asistente basado en inteligencia artificial:** entrenado con información del mundo del speedcubing para ayudar a usuarios principiantes o analizar los tiempos del usuario y ofrecer consejos personalizados de mejora.
- **Sección de algoritmos:** incluir algoritmos para F2L, OLL y PLL organizados y clasificados para facilitar el aprendizaje y consulta rápida dentro de la aplicación.
- **Mejoras en accesibilidad y personalización:** permitir cambiar el tamaño de fuente y personalizar los colores de la aplicación.
- **Ranking global de usuarios:** mostrar las mejores marcas de los usuarios según el tipo de cubo, fomentando la competitividad.
- **Soporte completo para cubos grandes:** añadir scrambles programados para 4x4x4, 5x5x5, 6x6x6 y 7x7x7, que no se pudo implementar por falta de tiempo.
- **Investigación sobre patentes y distribución:** como parte del desarrollo futuro, ya que seguiré trabajando en mejorar el proyecto, se investigará la posibilidad de patentar el software, la marca y la identidad visual de *CubiX*. Esto permitiría proteger legalmente el proyecto y, en caso de ser viable, proceder a su publicación en plataformas como **Google Play Store** y **Apple App Store**, ampliando así su alcance a una audiencia global.

Estas ampliaciones tienen como objetivo transformar la aplicación en una herramienta completa e integral para los entusiastas del cubo de Rubik, combinando práctica, aprendizaje y comunidad en un solo espacio.

9 Bibliografía

9.1 Documentación oficial de flutter

- **Flutter Material Components**
 - <https://api.flutter.dev/flutter/material/Tooltip-class.html>
 - <https://api.flutter.dev/flutter/material/AlertDialog-class.html>
 - <https://api.flutter.dev/flutter/material/showDialog.html>
 - <https://api.flutter.dev/flutter/material/AnimatedIcons-class.html>
 - <https://api.flutter.dev/flutter/material ListTile-class.html>
 - <https://api.flutter.dev/flutter/material/Divider-class.html>
- **Flutter Widgets y Layouts**
 - <https://api.flutter.dev/flutter/widgets/WidgetsBindingObserver-class.html>
 - <https://api.flutter.dev/flutter/widgets/ListView-class.html>
 - <https://api.flutter.dev/flutter/widgets/GridView-class.html>
 - <https://api.flutter.dev/flutter/scheduler/Ticker-class.html>
- **Accesibilidad e internacionalización**
 - <https://docs.flutter.dev/ui/accessibility-and-internationalization/internationalization>
- **Flutter Drawer**
 - <https://es.stackoverflow.com/questions/249793/como-cambiar-el-ancho-de-un-drawer-en-flutter>
 - <https://stackoverflow.com/questions/57748170/flutter-how-to-open-drawer-programmatically>
 - <https://docs.flutter.dev/cookbook/design/drawer>

9.2 Herramientas y paquetes externos (pub.dev)

- **Navegación y diálogos**
 - https://xdd666t.github.io/flutter_use/web/index.html?dialogType=AttachDialogScalePoint#/smartDialog
 - https://pub.dev/packages/flutter_smart_dialog/example
 - https://pub.dev/packages/curved_navigation_bar/install
- **Personalización UI**
 - https://pub.dev/packages/animated_text_kit
 - https://pub.dev/packages/popup_menu/example
 - <https://pub.dev/packages/popover>
 - https://pub.dev/packages/dropdown_button2
 - https://pub.dev/packages/custom_pop_up_menu
 - https://pub.dev/packages/stroke_text

- **Compartir y exportar contenido**
 - https://pub.dev/packages/share_plus/example
 - <https://pub.dev/packages/printing/install>
 - <https://pub.dev/packages/pdf>
 - <https://www.youtube.com/watch?v=3BgoL-I85-M>
 - <https://www.youtube.com/watch?v=6bYG-JwnoO4>
 -
- **Notificaciones, audio y vibración**
 - <https://pub.dev/packages/audioplayers/example>
 - <https://pub.dev/packages/vibration/example>

9.3 Seguridad y cifrado

<https://baguskto.medium.com/encrypting-passwords-in-a-secure-banking-app-using-flutter-11d8e74e4fe8>

9.4 Supabase

- **Uso e integración**
 - <https://supabase.com/docs/reference/dart/auth-onauthstatechange>
 - https://pub.dev/packages/supabase_flutter
 - https://pub.dev/packages/flutter_dotenv/install
- **Tutoriales**
 - <https://www.youtube.com/watch?v=vkXRRONI2U0>
 - <https://www.youtube.com/watch?v=F2j6Q-4nLEE&list=PL5S4mPUpp4OtkMf5LNDLXdTcAp1niHjoL>

9.5 Recursos prácticos (StackOverflow, Medium, etc.)

- **UI personalizada y lógica**
 - <https://minikin.medium.com/popover-for-flutter-90fb85b5630e>
 - <https://stassop.medium.com/animated-flutter-tooltip-c5fdcc4c613d>
 - <https://medium.com/@bhoomiharkhani22/custom-clipper-wave-curve-part-1-b49853a75876>
 - <https://medium.com/@omartinma/f%C3%A1cil-barra-de-b%C3%A1squeda-searchbar-en-flutter-58d22d6071d5>
- **Manejo de formularios y datos**
 - <https://stackoverflow.com/questions/54102043/how-to-do-a-database-table-update-with-sqlite-in-flutter>
 - <https://stackoverflow.com/questions/54972928/how-to-expand-a-textfield-in-flutter-looks-like-a-text-area>

- **Acciones del sistema**
 - https://www.youtube.com/watch?v=9Wic899udBQ&ab_channel=WidgetWisdom
 - <https://stackoverflow.com/questions/52568958/flutter-increase-height-and-width-of-switch>
 - <https://stackoverflow.com/questions/58977815/flutter-setstate-on-showdialog>
 - <https://stackoverflow.com/questions/43877288/how-to-hide-android-statusbar-in-flutter>
 - https://localhorse.net/article/como-usar-url_launcher-en-flutter
- **Manejo de formularios y datos**
 - <https://stackoverflow.com/questions/54102043/how-to-do-a-database-table-update-with-sqflite-in-flutter>
 - <https://stackoverflow.com/questions/54972928/how-to-expand-a-textfield-in-flutter-looks-like-a-text-area>
 - <https://es.stackoverflow.com/questions/323949/textfield-de-mi-pantalla-login-es-tapado-cuando-se-habilita-el-teclado>
- **Widget Icon**
 - <https://www.ricardogottheil.com/descubriendo-el-widget-icon-de-flutter/>
- **Problemas en paquetes**
 - https://github.com/fluttercommunity/plus_plugins/issues/3299
 - https://github.com/flutter-stripe/flutter_stripe/issues/1951
- **Selector de imágenes (imagen_picker)**
 - <https://medium.com/@hdgch1106/flutter-selecciona-im%C3%A1genes-desde-la-galer%C3%ADa-o-c%C3%A1mara-image-picker-17a78389c113>
- **Waves y CustomPainter**
 - https://www.youtube.com/results?search_query=custompainter+waves+flutter

9.6 YouTube - Tutoriales y soluciones

- **Interfaces y navegación**
 - <https://www.youtube.com/watch?v=FYFqUMHxu3Q>
 - <https://www.youtube.com/watch?v=4FLn4qYAfIA>
- **Funciones avanzadas**
 - <https://www.youtube.com/watch?v=aE8I66qr9Qw>
 - <https://www.youtube.com/watch?v=uKz8tWbMuUw>
 - <https://www.youtube.com/watch?v=EAW2aHbt4Fw>
 - https://www.youtube.com/watch?v=X_hQijCqaKA

- <https://www.youtube.com/watch?v=CN8ZAdvFKwA>
- <https://www.youtube.com/watch?v=DfgGP0OvhpE>
- **Custom Snackbar:**
https://www.youtube.com/watch?v=E2DXPr1SbrU&ab_channel=CodewithBenji
- **Guardar y mostrar imágenes desde SQLite (con ruta de imagen)**
https://www.youtube.com/watch?v=-ZmA1VJ-0Z4&ab_channel=ProgramingYT
- **Animated Floating Action Button**
<https://www.youtube.com/watch?app=desktop&v=B9qJcONJb-E>
- **Speed Dial Menu Animations**
<https://www.youtube.com/watch?v=YHNCYfqGrBY>
- **Ocultar el teclado automáticamente**
https://www.youtube.com/watch?v=ZHMNJ_XXVYc
- **Uso de PageView para navegación o sliders**
<https://www.youtube.com/watch?v=PrHBhyAf3lo>

9.7 Widget avanzados (*Popover, Tooltip, Search, Charts*)

- **Tooltip y Popover**
 - <https://nilenpatelinc.com/post/flutter-ui-82-fun-with-super-tooltip-in-flutter/>
 - https://github.com/stassop/flutter_animated_tooltip/blob/main/lib/animate_tooltip.dart
 - <https://stassop.medium.com/animated-flutter-tooltip-c5fdcc4c613d>
 - <https://www.geeksforgeeks.org/flutter-popover-button/>
 - <https://pub.dev/packages/popover/example>

- **Charts:**

https://pub.dev/packages/syncfusion_flutter_charts/example

- **SearchDelegate**

<https://medium.com/@sharansukesh2000/implementing-flutter-searchdelegate-a-step-by-step-guide-b8c4550627d1>

9.8 Herramientas externas

- **Generación de voz para archivos MP3**
 - <https://speechgen.io/es/>
- **Lógica del scramble del Square-1**
 - <https://www.jaapsch.net/scramblesq1.htm>

9.9 *Recursos personales y clases*

- Ejercicios y exámenes

- Tema 2: generación de Interfaces de Usuario, ejercicio 17 – cambiar modo.
- Ejercicio examen – NavigationBar + documentación.

- Ayuda mediante Inteligencia Artificial

- Resolver overflow del Snackbar.
- Añadir hover en el ListTile.
- Conversión a PDF.
- Generación de imágenes para Cubix.
- Ayuda para el entendimiento inicial de Supabase.

- Documentacion interna del IDE

- La propia documentación incorporada en el IDE al mantener en los distintos widgets.

10 Anexos

10.1 Anexo I – Código fuente de la aplicación

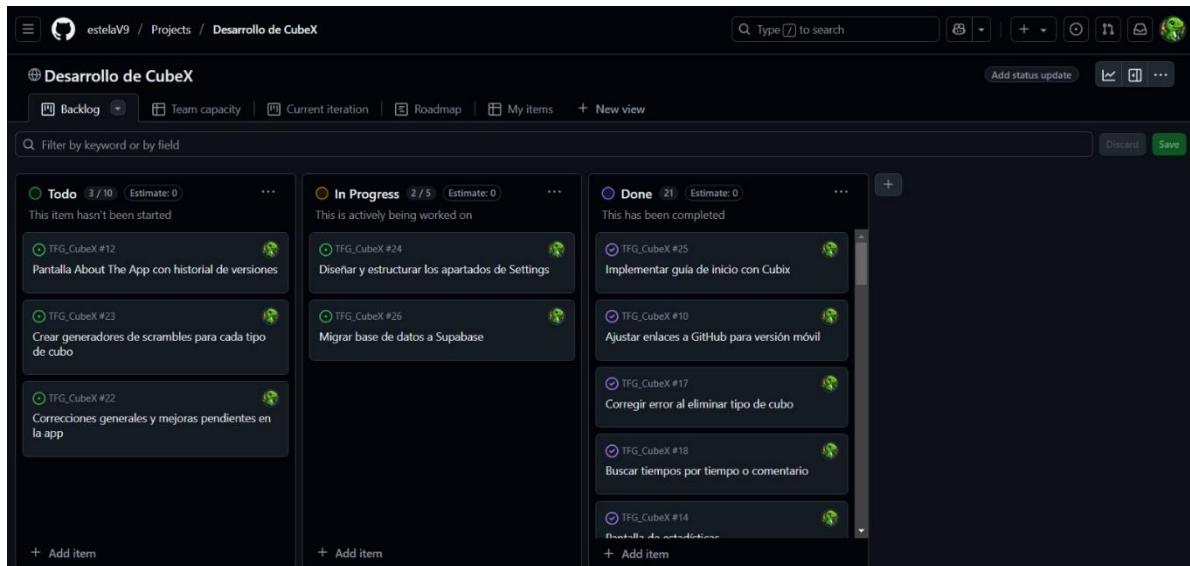
Este anexo contiene los archivos principales del proyecto desarrollado en Flutter y Dart. Se incluye la lógica del temporizador, el sistema de autenticación, el manejo de la base de datos y otras funcionalidades. El código está organizado en diferentes módulos y sigue el patrón de arquitectura MVVM (Modelo-Vista-ViewModel). El código completo se encuentra disponible en el repositorio de GitHub: [\[enlace\]](#).

10.2 Anexo II: Backlog del Proyecto

En este anexo se muestra el [backlog](#) del proyecto, que fue utilizado para gestionar las tareas durante el desarrollo de la aplicación. El flujo de trabajo siguió una estructura de **To Do**, **In Progress** y **Done**, lo que permitió un seguimiento del avance de las funcionalidades.

- **To Do:** Tareas que estaban pendientes de ser realizadas.
- **In Progress:** Tareas en desarrollo o en fase de implementación.
- **Done:** Tareas completadas y listas para ser integradas en la aplicación.

Las tareas fueron gestionadas de manera progresiva, con actualizaciones frecuentes a medida que se completaban y/o se añadian los objetivos. El backlog se mantuvo actualizado a lo largo de todo el proyecto, proporcionando una visión clara del progreso y de las funcionalidades implementadas.



The screenshot shows a digital backlog management tool with three main columns: **Todo**, **In Progress**, and **Done**. Each column contains several tasks, each with a status indicator (green for Todo, orange for In Progress, purple for Done), an estimate value, and a brief description.

Column	Task ID	Description
Todo	TFG_CubeX #12	Pantalla About The App con historial de versiones
	TFG_CubeX #23	Crear generadores de scrambles para cada tipo de cubo
	TFG_CubeX #22	Correcciones generales y mejoras pendientes en la app
In Progress	TFG_CubeX #24	Diseñar y estructurar los apartados de Settings
	TFG_CubeX #26	Migrar base de datos a Supabase
	TFG_CubeX #17	Corregir error al eliminar tipo de cubo
Done	TFG_CubeX #25	Implementar guía de inicio con Cubix
	TFG_CubeX #10	Ajustar enlaces a GitHub para versión móvil
	TFG_CubeX #18	Buscar tiempos por tiempo o comentario
Done	TFG_CubeX #14	Detalles de estadísticas

10.3 Anexo III – Prototipo de diseño (*Figma*)

Este anexo incluye un enlace al prototipo de la aplicación, diseñado en Figma. En dicho prototipo se muestran las pantallas principales de la aplicación junto con sus respectivos componentes visuales. Además, se incluye el diagrama de algunos de estos componentes. Este prototipo sirvió como base para la implementación de la interfaz de usuario.

10.4 Anexo V – Tutorial de bienvenida

Este anexo contiene las capturas del tutorial implementado dentro de la aplicación, pensado para guiar al usuario en su primer uso. También se incluye una descripción de su lógica de funcionamiento y un fragmento del código principal.

Capturas de pantalla del tutorial

A continuación, se muestran algunas capturas del tutorial que se presenta la primera vez que un usuario se registra en la aplicación:

10.4.1.1 Pantalla de bienvenida – Presentación de Cubix

- En esta primera pantalla, el usuario es recibido por **Cubix**, la mascota de la aplicación, quien se presenta y ofrece su ayuda para realizar un recorrido inicial.
- **Selección de idioma del tutorial:** Se da la opción de elegir entre **español o inglés**, con el objetivo de mejorar la experiencia de usuario desde el primer momento, adaptándose al idioma del usuario.
- Una vez seleccionado el idioma, Cubix pregunta al usuario si desea realizar el recorrido guiado. Aquí se presentan la opción de continuar con el tour u omitir el tutorial.

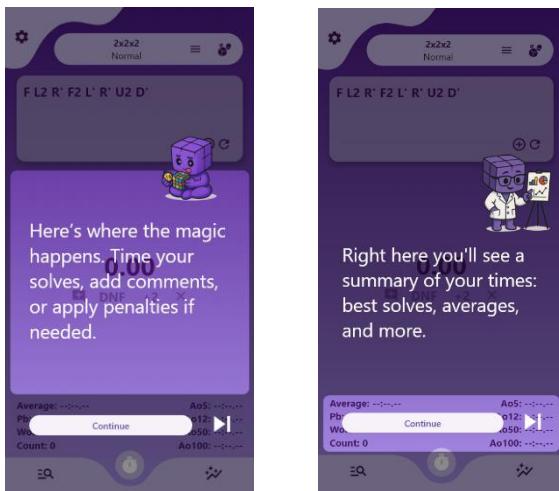


10.4.1.2 Pantalla del cronómetro

En esta pantalla, el tutorial explica los distintos componentes de la vista principal de la aplicación:

- **Barra superior:** El usuario es guiado para identificar la barra principal desde donde puede ver y gestionar tanto el tipo de cubo como la sesión activa.
- **Gestión de tipos de cubo:** Se señala el botón que permite elegir y personalizar diferentes tipos de cubos.
- **Gestión de sesiones:** Se señala el botón destinado a cambiar, eliminar o crear nuevas sesiones para organizar los tiempos de forma más clara según las preferencias del usuario.
- **Scramble:** El tutorial muestra el generador de scrambles adaptados al tipo de cubo seleccionado.
- **Temporizador:** Se enseña al usuario donde usar el cronómetro para registrar sus tiempos. Además, se indican las opciones para añadir comentarios, aplicar penalizaciones o eliminar tiempos.
- **Estadísticas rápidas:** Finalmente, se muestra el área donde aparecen estadísticas clave: mejor y peor tiempo, cantidad total de resoluciones, y promedios del total y de 5, 12, 50 y 100 últimos tiempos.



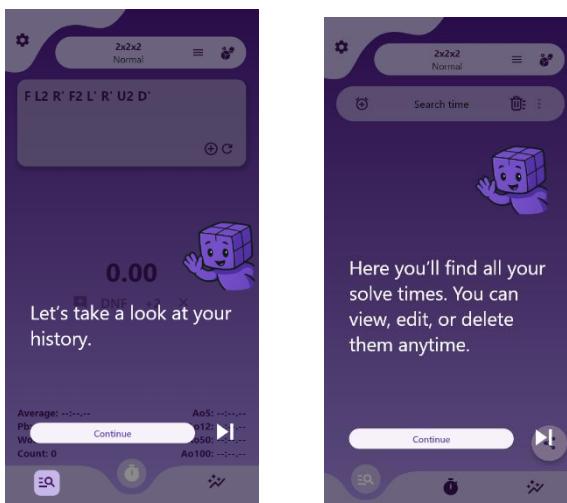


10.4.1.3 Pantalla del historial

Después de explorar la pantalla principal, el tutorial invita al usuario a dirigirse a la sección de historial, guiándolo mediante el botón correspondiente en la barra de navegación inferior.

En esta pantalla, se explican las funcionalidades disponibles:

- El usuario puede visualizar un listado con todos los tiempos registrados en la aplicación.
- Cada tiempo puede ser seleccionado para ver su detalle, editarlo (por ejemplo, agregar o cambiar penalizaciones), o eliminarlo si lo desea.
- Esta sección permite llevar un control detallado de todas las resoluciones realizadas, organizadas según el usuario prefiera.

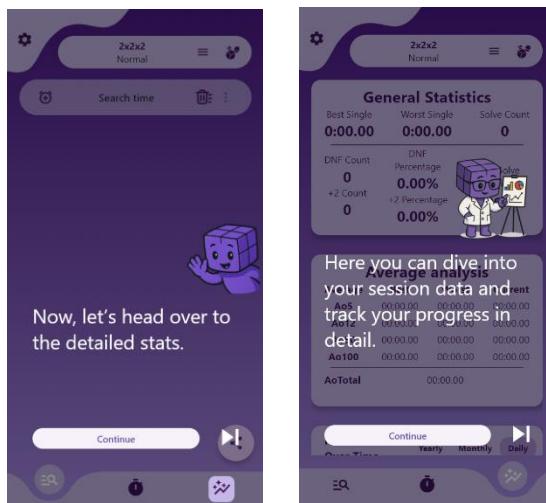


10.4.1.4 Pantalla de las estadísticas

Tras revisar el historial, el tutorial guía al usuario hacia la sección de estadísticas utilizando el botón correspondiente en la barra de navegación inferior.

En esta pantalla se destacan las herramientas disponibles para el análisis del rendimiento:

- Se presentan estadísticas generales como el mejor y peor tiempo, cantidad de tiempos registrados y medias calculadas.
- El usuario puede consultar un análisis detallado de los promedios (media de 5, 12, 50, 100 y total).
- Además, se incluye una gráfica que representa visualmente los tiempos a lo largo del año, mes y día, lo que permite seguir la evolución del rendimiento en diferentes períodos de tiempo.

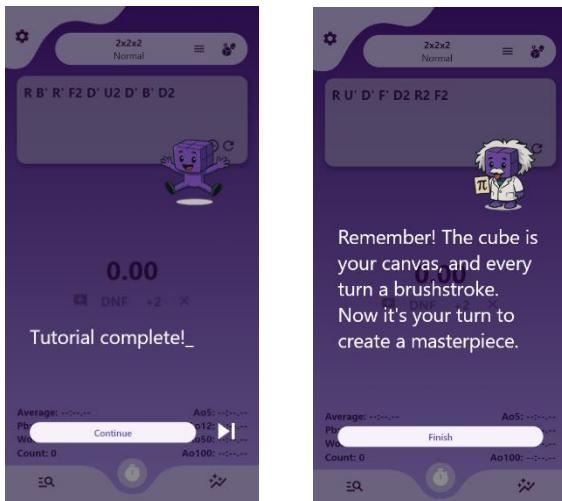


10.4.1.5 Final del tutorial por la aplicación

Después de explorar las principales funcionalidades de la aplicación, el recorrido concluye regresando a la pantalla principal del temporizador.

En esta última etapa:

- Se informa al usuario de que el tutorial ha finalizado correctamente.
- Cubix, la mascota de la aplicación, se despide con una frase motivacional: *“¡Recuerda! El cubo es tu lienzo, y cada giro una pincelada. Ahora es tu turno de crear tu obra.”*
- Finalmente, el usuario puede pulsar el botón de **Finalizar** para empezar a utilizar la aplicación por su cuenta.



10.4.1.6 Breve explicación del funcionamiento del tutorial

El tutorial de bienvenida tiene como objetivo ofrecer al usuario una guía inicial por las secciones principales de la aplicación, facilitando así su primera experiencia de uso. Está diseñado para presentarse **únicamente la primera vez que un usuario accede a la aplicación tras registrarse**, mostrándole de forma visual e interactiva los elementos clave como las partes del temporizador, el menú de navegación, el área de estadísticas y otras funcionalidades importantes.

Antes de comenzar, el usuario puede **elegir el idioma** en el que desea visualizar el tutorial (español o inglés), lo que mejora la accesibilidad y la comprensión para distintos usuarios.

El recorrido está guiado por **Cubix**, la mascota de la aplicación, quien acompaña al usuario con mensajes explicativos durante todo el tutorial, aportando un enfoque más amigable y cercano.

Durante el tutorial, el usuario puede **avanzar las entre pantallas explicativas** del tutorial mediante el botón de "Continuar" o, si lo prefiere, puede **omitirse por completo** utilizando la opción "Saltar". Una vez finalizado o saltado, el tutorial no vuelve a mostrarse.

En cuanto a la implementación técnica, se ha utilizado la librería *flutter_smart_dialog* para gestionar la presentación de los mensajes y las pantallas del tutorial.

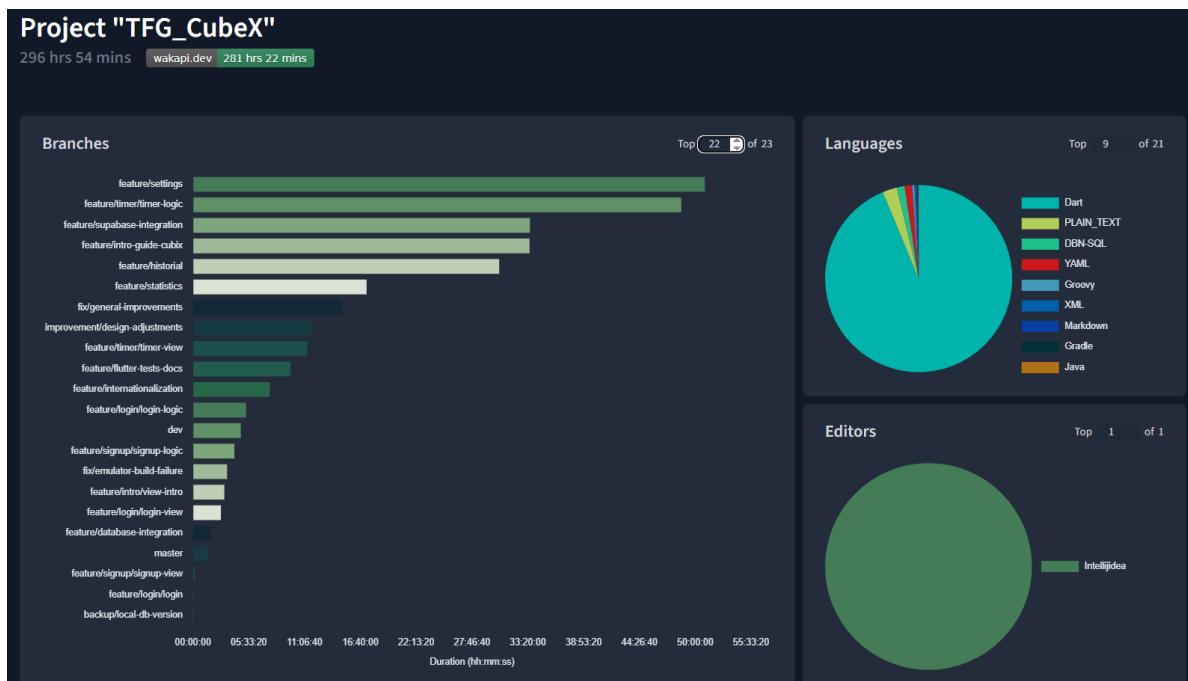
10.5 Registro de tiempo de desarrollo

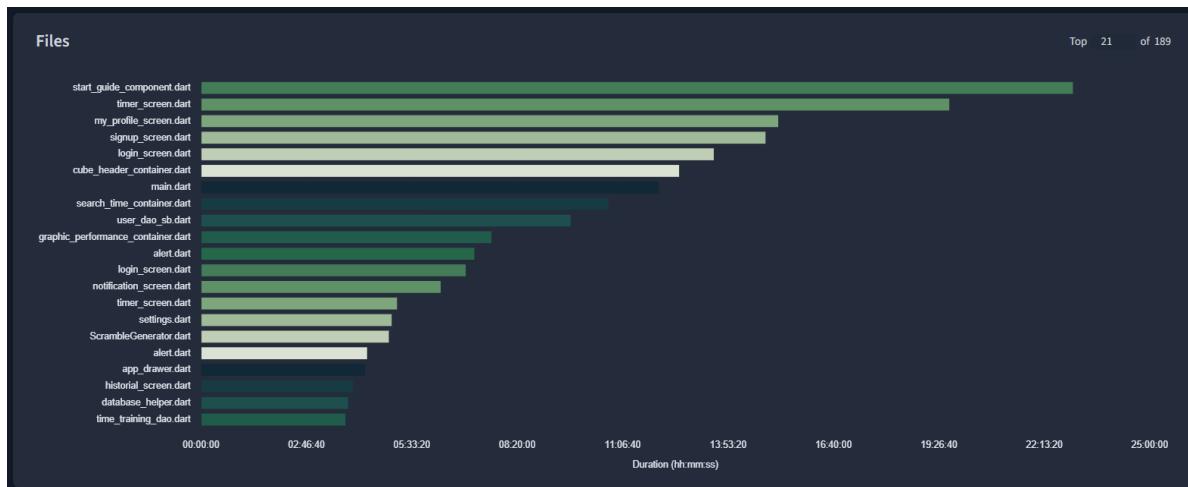
Este anexo contiene una captura de pantalla extraída de **Wakapi**, una herramienta de seguimiento de actividad de desarrollo, utilizada durante el transcurso del proyecto para registrar el tiempo real dedicado a la programación.

En la imagen se muestra un resumen detallado de:

- **Las horas trabajadas distribuidas por ramas del repositorio**, lo que refleja el enfoque y evolución del trabajo en las distintas partes del proyecto.
- **Los archivos en los que más tiempo se ha invertido**, indicando claramente cuáles han requerido mayor esfuerzo o complejidad.
- Una vista general del tiempo acumulado en el proyecto, clasificado también por tecnologías y lenguajes utilizados.

Este registro aporta **evidencia de la dedicación y la organización** a lo largo de todo el proceso de desarrollo.

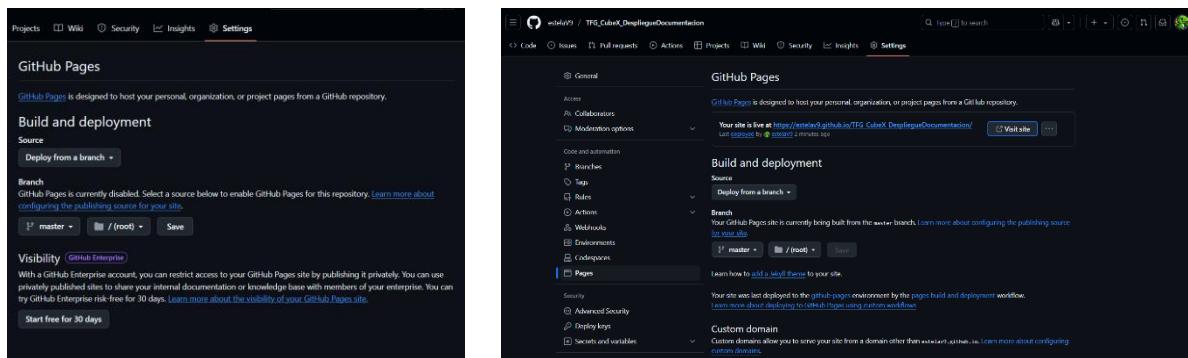




10.6 Página de documentación

Se desplegaron las páginas de documentación generadas automáticamente mediante la herramienta **dart doc**. Esta documentación permite visualizar de manera estructurada los elementos del proyecto, como clases, funciones y estructuras generales.

Para facilitar el acceso, se creó un repositorio que contiene los archivos generados por dart doc. Luego, desde la configuración del repositorio (**Ajustes > Pages**), se seleccionó la rama principal como fuente de publicación. Tras guardar los cambios y recargar la página, el despliegue de la documentación quedó disponible.



La documentación generada se puede consultar desde el siguiente enlace:
https://estelav9.github.io/TFG_CubeX_DespliegueDocumentacion/