UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
INGENIERÍA DEL SOFTWARE

RejillaApp: una App para la evaluación y entrenamiento de la atención

RejillaApp: Attention assessment and training App

Realizado por
Rubén Molina Lozano

Tutorizado por
José Luis Pastrana Brincones

Departamento
Lenguajes y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO de 2024

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA DEL SOFTWARE

RejillaApp: una App para la evaluación y entrenamiento de la atención

RejillaApp: Attention assessment and training App

Realizado por
Rubén Molina Lozano

Tutorizado por
José Luis Pastrana Brincones

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2024

Fecha defensa: Junio de 2024

Resumen

En la sociedad actual, imperan los estímulos constantes y la sobreexposición a una cantidad de información difícil de procesar para el cerebro humano, así como distracciones constantes que la persona promedio puede, o no, desear en su día a día. Debido a esto y otros factores, las condiciones mentales relacionadas con la falta de atención o la reducción de esta se han visto en auge, modificando y a veces complicando la vida de muchas personas que, a veces sin siquiera ser conscientes, ven su calidad de vida reducida junto con sus habilidades para prestar atención por un periodo de tiempo moderado a algunas cosas.

Este Trabajo de Fin de Grado se centra en el desarrollo de una aplicación accesible e intuitiva para Android, que será utilizada por psicólogos para evaluar la capacidad de atención del usuario que la utiliza. Esta aplicación está específicamente desarrollada para medir diversos tipos de atención y generar una serie de resultados y mediciones que un psicólogo puede usar para determinar si la persona que utiliza la app presenta un problema en su capacidad de atención o no.

La aplicación “RejillaApp” está centrada en lo que se llamará a partir de ahora “rejillas”, que son tablas cuyas casillas están formadas por botones que contendrán, dependiendo de lo que el usuario seleccione: letras, símbolos, números, colores o imágenes. El usuario deberá pulsar estos botones en la pantalla de la aplicación, con un orden concreto seleccionado, y esquivando una serie de distracciones que pondrán a prueba la capacidad de atención que tiene para terminar en un tiempo determinado la prueba. Tras esto, la aplicación generará una serie de datos y gráficas y dará al usuario las opciones pertinentes para enseñar estos resultados a una persona que los pueda evaluar, o hacerlo él mismo.

Para desarrollar esta aplicación se ha usado Android Studio, el lenguaje Java, SQLite y otras tecnologías usadas comúnmente en Android.

Palabras clave:

Android, Móvil, Aplicación, Java, TDAH, Atención, Psicología

Abstract

In today's society, constant stimuli and overexposure to an amount of information that is difficult for the human brain to process, as well as constant distractions that the average person may or may not want in their daily lives, are the norm. Due to this and other factors, mental conditions related to inattention or reduced attention span have been on the rise, modifying and sometimes complicating the lives of many people who, sometimes without even being aware, see their quality of life reduced along with their abilities to pay attention for a moderate period of time to some things.

This Final Year Dissertation focuses on the development of an accessible and intuitive application for Android, which will be used by psychologists to assess the attention span of the user who uses it. This application is specifically developed to measure various types of attention and generate a series of results and measurements that a psychologist can use to determine whether the person using the app presents a problem in their attention capacity or not.

The application “RejillaApp” is focused on what will be called from now on “grids”, which are tables whose boxes are formed by buttons that will contain, depending on what the user selects: letters, symbols, numbers, colors or images. The user will have to press these buttons on the application screen in a specific order that would be selected and by avoiding a series of distractions presented in the application, it will test the user's attention span in order to finish the test within the given time. After this, the application will generate a series of data and graphs then will give the user the appropriate options to show these results to a person who can evaluate them, or do it himself.

Android Studio, Java language, SQLite and other technologies commonly used in Android have been used to develop this application.

Keywords:

Android, Mobile, Application, Java, ADHD, Attention, Psychology

Índice

1

Introducción

1.1 Motivación

La atención es algo muy complejo y difícil de explicar, donde intervienen varios sistemas del cerebro humano. La podríamos definir a grandes rasgos como un filtro. El cerebro no es capaz de percibir totalmente y asimilar toda la información con la que se le bombardea cada segundo, por eso el cerebro intenta centrar la conciencia en algo de forma selectiva y descartar la información no deseada para procesarla y dar una respuesta adecuada. Este filtro es lo que se conoce como “atención”, y es necesario para cualquier tipo de acción (percibir, ver y escuchar). Sin atención, el aprendizaje o el recuerdo no pueden tener lugar o se ven muy reducidos. Los científicos le han prestado cierto interés por su repercusión en nuestra vida cotidiana y algunos lo consideran un tercer sistema neurofisiológico, junto a los dos más importantes del sistema nervioso.

Siendo algo tan abstracto y complejo, medir la atención de forma numérica es algo muy difícil. Este trabajo de Fin de Grado pretende dar solución a este problema, proponiendo

una forma rápida, sencilla y accesible para medir la capacidad de atención de un usuario con una herramienta tan cotidiana como un teléfono móvil Android.

Midiendo la capacidad de atención de una persona, y aportando una serie de análisis y gráficas, podemos ayudar tanto al profesional de la salud encargado de determinar qué tipo de tratamientos deben aplicarse en cada caso, como a la persona que presenta dificultades en su capacidad de atención, a agilizar el proceso por el cual se recibe esta ayuda o terapia destinada a mejorar la calidad de vida del afectado.

Midiendo la capacidad de atención de una persona, podemos determinar qué tipo de terapia puede necesitar si presenta problemas en su vida diaria relacionados con su capacidad de atención que quiere tratar con un psicólogo.

1.2 Objetivos

En este Trabajo de Fin de Grado, se propone el desarrollo de una aplicación Android en lenguaje Java para evaluar los distintos tipos de atención humana. Esta aplicación pretende ofrecer una manera eficiente, práctica y rápida para la evaluación de la capacidad de atención de los usuarios, centrándose en partes fundamentales del desarrollo de software, como el análisis de requisitos, el diseño de interfaces que sigan el concepto de usabilidad y la exposición de los casos de uso.

La metodología seleccionada para llevar a cabo este proyecto es ágil e iterativa, dividiendo el proceso en cuatro iteraciones. En cada iteración se implementarán los diversos requisitos que la aplicación debe cumplir para poder evaluar efectivamente la atención del usuario y dar un servicio lo más completo posible. Para comenzar el desarrollo se analizan las funcionalidades requeridas y se elige la iteración correcta en la que efectuarlo.

En la primera iteración, se crea la interfaz básica de la aplicación y su funcionalidad principal, la rejilla, así como la posibilidad de creación y compleción de la rejilla de letras,

teniendo así listo un primer vistazo bastante completo a la aplicación que permita transmitir su funcionamiento y la estructura de la interfaz al cliente.

En la segunda iteración, tras la confirmación del cliente, se procede a la implementación de la mayoría de las rejillas restantes, así como algunas funcionalidades extra que se requerían y las interfaces asociadas a las mismas. En esta iteración se crea además la base de datos y la recogida de resultados del usuario.

La tercera iteración se basa en completar las funcionalidades restantes, perfilar algunas partes de la interfaz, y dotar de funcionalidad al resto de menús que no estaban aún implementados. En esta iteración se completa además la implementación de las rejillas restantes.

Finalmente, la cuarta y última iteración se concentrará en la implementación de la capacidad de subir los datos a un servidor externo y terminar de pulir la aplicación en su conjunto.

Como resultado final del proyecto, se entregará un archivo comprimido que contenga todos los componentes de la aplicación desarrollada en Java para Android, el código, y los recursos utilizados.

1.3 Estructura de la memoria

En este apartado se habla de la estructura de este documento, explicando de que se habla en cada sección.

1. **Introducción:** En este apartado se presenta el contexto y la motivación detrás del desarrollo de la aplicación. Se explica el problema que se busca resolver y la necesidad que la aplicación pretende cubrir. Además, se detallan los objetivos del proyecto y se ofrece una visión general del contenido del documento, describiendo brevemente lo que se abordará en cada sección.

2. **Fundamentos psicológicos:** En esta sección se explican las bases psicológicas en las que se basa la decisión de crear una aplicación para la evaluación y entrenamiento de la atención.

3. **Análisis de caso de uso:** Esta sección aborda las decisiones de diseño tomadas durante la planificación del proyecto. Se explica la interfaz de la aplicación y las decisiones sobre la experiencia de usuario.

4. **Tecnologías:** Aquí se describen las tecnologías utilizadas en el desarrollo de la aplicación. Se detalla el lenguaje de programación, entorno de desarrollo, base de dato y se hace una explicación resumida de cómo funcionan estas tecnologías. Se explica por qué se eligieron y cómo contribuyen al objetivo del proyecto.

5. **Desarrollo:** Este apartado cubre el proceso de desarrollo de la aplicación. Se describen todos los requisitos de la aplicación y se detalla cómo se implementan. Estos requisitos se organizan en iteraciones, y en este apartado se detalla el trabajo realizado en cada una de ellas.

6. **Conclusiones y líneas futuras:** En el apartado de conclusiones y se resumen los resultados del proyecto. En las líneas futuras, se proponen posibles mejoras y extensiones de la aplicación que podrían llevarse a cabo en trabajos posteriores. Se identifican nuevas funcionalidades, optimizaciones y cualquier otra dirección en la que podría evolucionar el proyecto.

   Apéndice A. Manual de instalación: En esta sección se proporciona una guía detallada sobre cómo instalar y configurar la aplicación, así como las restricciones que el dispositivo debe cumplir para poder utilizarla. Se incluyen instrucciones paso a paso para la instalación en Android y para la configuración inicial de la aplicación.

Apéndice B. Manual de usuario: Aquí se ofrece una guía completa para los usuarios de la aplicación. Se describe cómo utilizar todas las funcionalidades de la aplicación, acompañadas de capturas de pantalla y ejemplos prácticos. Se explican las principales características de la aplicación y cómo realizar tareas específicas, asegurándose de que el usuario pueda sacar el máximo provecho de la herramienta.

2

Fundamentos psicológicos

2.1 Introducción

La atención es un estado neurocognitivo fundamental que juega un papel crucial en la preparación del cerebro para la percepción y la acción. Esta función surge de una intrincada red de conexiones corticales y subcorticales, predominantemente en el hemisferio derecho, y sirve para enfocar selectivamente la conciencia, filtrar información sensorial, resolver la competencia entre estímulos y activar las áreas cerebrales adecuadas para las respuestas. El estudio de la atención abarca múltiples sistemas cerebrales y tiene una relevancia clínica significativa debido a su implicación en diversas condiciones neurológicas y psiquiátricas.

La atención abarca una amplia gama de estructuras cerebrales que conectan sistemas muy complejos, mostrando la naturaleza multifacética de la atención.

La atención está regulada por tres sistemas interrelacionados:

- **Alerta o excitación**
- **Atención posterior**
- **Atención anterior**

Aunque la atención funciona bilateralmente, el hemisferio derecho juega un papel dominante, ya que es crucial para mantener la alerta y regular el sistema de excitación.

2.2 Los Nueve Tipos de Atención

La atención juega un papel crucial en nuestra capacidad para procesar y responder a los estímulos de nuestro entorno. El estudio de la atención ha revelado su complejidad, descomponiéndola en varios tipos, cada uno con características y funciones distintas. Según el documento estudiado, podemos identificar nueve tipos de atención, explicadas a continuación.

1. **Vigilancia o Alerta:**

   La vigilancia, también conocida como alerta o excitación, se refiere al nivel de conciencia y preparación para responder a estímulos. Va desde el sueño profundo hasta la hiperalerta. Este tipo de atención es crucial para mantener la conciencia y la preparación para reaccionar a cambios en el entorno. El nivel de vigilancia se determina mediante registros neuroeléctricos y pruebas clínicas neurológicas. Niveles altos de vigilancia son esenciales para tareas que requieren monitoreo continuo y respuestas rápidas, como el control del tráfico aéreo y la vigilancia.

2. **Amplitud de Atención:**

   La amplitud de atención se refiere a la capacidad de atender a múltiples estímulos simultáneamente. Este tipo de atención está estrechamente ligado a la memoria de trabajo y se mide a menudo a través de tareas que requieren la

repetición inmediata de una serie de estímulos, como golpes rítmicos, dígitos o secuencias espaciales. La amplitud de atención es crucial para actividades que implican retener y manipular información durante periodos cortos, como la aritmética mental y seguir instrucciones complejas.

3. **Atención Selectiva o Focal:**

   La atención selectiva o focal implica concentrarse en un estímulo específico mientras se ignoran otros. Este tipo de atención es esencial para procesar eficazmente la información relevante y filtrar las distracciones. La atención selectiva se explora típicamente utilizando tareas de búsqueda visual y tareas de cancelación. Es un componente crítico de actividades cotidianas, como leer, donde se debe concentrar en el texto ignorando los estímulos visuales circundantes.

4. **Atención de desplazamiento:**

   También conocida como atención desplazada, se refiere a la capacidad de cambiar el enfoque entre diferentes estímulos o ubicaciones. Este tipo de atención es necesario para tareas que requieren monitoreo de múltiples fuentes de información y adaptación a cambios. El sistema de atención posterior, particularmente la corteza parietal posterior, juega un papel significativo en el control del cambio de atención. Esta habilidad es vital para actividades como conducir, donde se debe cambiar continuamente la atención entre la carretera, los espejos y el tablero.

5. **Atención Serial:**

   La atención serial, o procesamiento serial, implica el procesamiento paso a paso de información. Es necesario para tareas que requieren un examen sistemático de los estímulos, como buscar un elemento específico en una lista o resolver rompecabezas. La atención serial se prueba a menudo a través de tareas de cancelación, donde los individuos deben identificar y cancelar estímulos

específicos de una serie de distractores. Este tipo de atención es fundamental para actividades que implican análisis detallado y resolución de problemas.

6. **Atención Dividida o Dual:**

   La atención dividida o dual, también conocida como atención compartida, se refiere a la capacidad de procesar múltiples tareas simultáneamente. Este tipo de atención es crítico para la multitarea, como conducir mientras se conversa o escribir mientras se escuchan instrucciones. Una atención dividida eficiente mejora la productividad y la eficiencia en diversas actividades profesionales y cotidianas.

7. **Atención de preparación:**

   La atención preparatoria implica la disposición para llevar a cabo una operación cognitiva, movilizando los esquemas de respuesta más apropiados para la tarea en cuestión. Este tipo de atención es esencial para anticipar y planificar acciones, asegurando que las áreas cerebrales responsables de los procesos cognitivos requeridos estén activadas. La atención preparatoria es crucial para tareas que requieren planificación estratégica y ejecución, como deportes y resolución de problemas complejos.

8. **Atención Sostenida:**

   La atención sostenida, también conocida como concentración o vigilancia, se refiere a la capacidad de mantener el enfoque en una tarea o evento durante un período prolongado. Este tipo de atención es vital para actividades que requieren monitoreo continuo y reacciones rápidas, como el monitoreo de seguridad y tareas académicas. Los déficits de atención sostenida se observan a menudo en trastornos de atención con hiperactividad. La prueba de rendimiento continuo (CPT) es un método común para evaluar la atención sostenida.

9. **Inhibición de Respuestas Automáticas:**

   La inhibición de respuestas automáticas implica suprimir reacciones instintivas o habituales a estímulos. Este tipo de atención es necesario para controlar comportamientos impulsivos y ejecutar acciones deliberadas. Se evalúa a menudo utilizando pruebas que requieren que los individuos inhiban su respuesta automática de leer una palabra y en su lugar nombren el color de la tinta. La inhibición efectiva de respuestas automáticas es crucial para la autorregulación y el comportamiento disciplinado.

2.3 Desarrollando una Aplicación para Evaluar la Atención

En la era digital actual, nuestra capacidad para mantener y regular la atención es continuamente puesta a prueba. Comprender y mejorar las capacidades atencionales puede mejorar significativamente la productividad, el aprendizaje y la salud cognitiva en general. Desarrollar una aplicación que pruebe varios tipos de atención puede ofrecer una solución innovadora y atractiva. Al integrar principios científicos y características interactivas, esta aplicación puede evaluar de manera integral y ayudar a mejorar nuestras habilidades atencionales.

A continuación, se detallan las funcionalidades de la aplicación que evalúan cada tipo de atención.

2.3.1 Medición de la Alerta

Una de las funcionalidades se centra en evaluar la alerta, también conocida como excitación. Se presentan a los usuarios estímulos visuales aleatorios utilizando la aleatorización de rejillas. Los usuarios necesitarán responder lo más rápido posible, con la aplicación registrando el tiempo de reacción que han requerido para hacer clic en los botones, proporcionando así información sobre el estado de alerta del usuario. Esta función evalúa la preparación general y la capacidad de respuesta, esencial para actividades que requieren reacciones rápidas y monitoreo continuo.

2.3.2 Evaluación de la Amplitud de Atención

Existe otra funcionalidad que prueba la amplitud de atención del usuario. Esta involucra la presentación de secuencias de números, letras o formas que los usuarios deben recordar y reproducir en el orden correcto. Para ello se utilizan las rejillas numérica y de abecedario. La longitud de las secuencias aumentaría según el rendimiento, desafiando la memoria de trabajo y la amplitud de atención del usuario. Esta tarea es crucial para evaluar la capacidad de manejar y manipular información durante periodos cortos.

2.3.3 Evaluación de la Atención Selectiva

Esta funcionalidad evalúa la atención selectiva a través de una tarea específica. Los usuarios seleccionan su color, letra, imagen o símbolo preferido mientras una línea se mueve frente a ellos. Para esto se usa la distracción de línea de la aplicación, midiendo así la capacidad del usuario para filtrar información irrelevante y concentrarse en estímulos específicos, una habilidad esencial para tareas que requieren atención focalizada en medio de distracciones.

2.3.4 Prueba de Atención de Desplazamiento

Los usuarios necesitarán alternar entre diferentes tareas. La aplicación rastreará la velocidad y precisión de estos cambios, evaluando la capacidad del usuario para adaptarse rápidamente a tareas y prioridades cambiantes. Esta habilidad es vital para la multitarea y la gestión de entornos dinámicos, y se evaluará con la función de emparejar, que muestra al usuario un botón a pulsar en el que este se tendrá que centrar antes de pulsarlo en la rejilla.

2.3.5 Examen de la Atención Serial

La atención serial se medirá mediante una tarea de búsqueda visual. Los usuarios tendrán que encontrar un objeto objetivo específico entre numerosos distractores, y la aplicación medirá el tiempo necesario para localizar el objetivo. Esta tarea evalúa la capacidad del usuario para escanear sistemáticamente e identificar estímulos relevantes, reflejando escenarios del mundo real que requieren búsqueda enfocada. Esto se hace en todas y cada una de las rejillas existentes en la aplicación, y es el foco

principal de la misma, ya que en cada rejilla el usuario debe hacer precisamente esto, filtrar botones incorrectos para encontrar los correctos.

2.3.6 Evaluación de la Atención Dividida

Esta funcionalidad evaluará la atención dividida, requiriendo que los usuarios realicen dos tareas simultáneamente, como identificar un número en medio de múltiples distracciones mientras se calcula progresivamente el siguiente número a pulsar. Este paradigma de tarea dual evalúa la capacidad del usuario para procesar y gestionar múltiples flujos de información de manera concurrente, una habilidad crucial para manejar actividades complejas de manera efectiva. Esto se pone a prueba en las rejillas numéricas, y también en la de abecedario, donde tendrá que evaluar que letra es la siguiente en la secuencia.

2.3.7 Evaluación de la Atención Preparatoria.

Los usuarios necesitarán responder rápidamente a ciertos estímulos mientras se abstienen de responder a otros. Esta tarea evalúa la disposición del usuario para actuar y su capacidad para suprimir respuestas automáticas, esencial para la planificación y ejecución estratégica. Esto es evaluado con la función emparejar con límite de tiempo, que hará al usuario tener que esperar por el elemento a pulsar mientras se abstiene de pulsar los que aún no se muestran como correctos en la casilla de emparejar.

2.3.8 Medición de la Atención Sostenida

La función de límite de tiempo de la rejilla probará la atención sostenida a través de una prueba de rendimiento continuo (CPT). Los usuarios necesitan responder a los objetivos específicos que aparecen esporádicamente entre no objetivos durante un período prolongado. Esta tarea mide la capacidad del usuario para mantener el enfoque y la vigilancia, crucial para actividades que requieren concentración prolongada.

2.3.9 Evaluación de la Inhibición de Respuestas Automáticas

Finalmente, esta característica desafía la capacidad del usuario para inhibir respuestas automáticas. Similar a la prueba Stroop, esta tarea involucra estímulos conflictivos que requieren que los usuarios supriman reacciones instintivas y sigan instrucciones específicas, como elegir el color que se les ha indicado antes de que cambie a otro. Esta habilidad es vital para la autorregulación y el comportamiento disciplinado.

2.4 Conclusión

Comprender los nueve tipos de atención proporciona valiosos conocimientos sobre las complejidades del procesamiento cognitivo. Cada tipo de atención juega un papel único en cómo interactuamos con nuestro entorno, gestionamos tareas y regulamos el comportamiento. Al explorar estas diversas formas de atención, podemos comprender mejor su importancia en la vida diaria y las posibles implicaciones para la salud y el rendimiento cognitivo. Este conocimiento es esencial para desarrollar intervenciones y estrategias dirigidas a mejorar la atención y mitigar los déficits atencionales.

Los pacientes con lesiones cerebrales traumáticas, procesos neuroinfecciosos, demencias subcorticales y epilepsia a menudo exhiben déficits significativos de atención. Trastornos del desarrollo como el TDAH y la dislexia subrayan aún más la importancia de la atención en el funcionamiento diario. Las condiciones psiquiátricas, incluyendo la esquizofrenia y la depresión, también demuestran marcados deterioros en la atención. Estos trastornos destacan el amplio impacto de los déficits de atención en la salud cognitiva y emocional, haciendo que el estudio de la atención sea crucial para desarrollar intervenciones y terapias efectivas.

Desarrollar una aplicación que integre estas diversas pruebas de atención proporciona una herramienta integral para evaluar y mejorar las capacidades atencionales. Al utilizar tareas científicamente fundamentadas en un formato interactivo y fácil de usar, la aplicación RejillaApp puede ofrecer valiosas perspectivas sobre las fortalezas cognitivas y áreas de mejora, además de servir como una poderosa herramienta para el crecimiento personal, propósitos educativos y entornos profesionales, avanzando en nuestra comprensión y dominio de la atención.[1]

3

Análisis del caso de uso

3.1 Introducción

En esta sección se desglosan todos los aspectos a tener en cuenta de cara a la toma de decisiones para la aplicación. Para ello se presentan distintos análisis o problemáticas que surgen cuando se tratan de identificar a los usuarios objetivo de la aplicación, trazando el perfil de usuario final y observando las necesidades de este, con el objetivo de ofrecer una experiencia de usuario simple, intuitiva y personalizada.

De esta manera, se introducen distintas guías y estándares que se utilizan en la ingeniería de software en los que el proyecto se apoyará para la toma de decisiones para aspectos concretos de la aplicación, como pueden ser:

- **La interfaz:** Aspirando a desarrollar una interfaz que tenga en consideración al usuario.

- **La temática de la aplicación:** Usando los colores apropiados u otros elementos visuales necesarios para dar coherencia y empaque a la aplicación.

- **La adaptabilidad:** Aplicando soluciones inclusivas a las necesidades específicas del usuario.

3.2 Perfil del usuario objetivo

Las barreras y desigualdades en el uso y aprendizaje de las tecnologías de comunicación móvil se vinculan a varios factores: nivel educativo, experiencia laboral previa y condiciones socioeconómicas, junto con inseguridades y temores por la falta de alfabetización digital. Además, las dificultades originadas a partir de los problemas físicos y cognitivos propios de la edad, como la pérdida de visión y la motricidad fina, se agravan por la falta de dispositivos y aplicaciones móviles diseñados de acuerdo con las características específicas de este grupo de edad.

Teniendo en cuenta todos estos factores, la aplicación se enfoca a un público amplio, pero entendiendo los límites de los dispositivos móviles actuales, siendo que una persona que no es capaz de utilizar correctamente sus manos por el deterioro causado por la edad no podrá pulsar los botones de la aplicación de forma precisa, influyendo esto en su experiencia y evaluación. Una persona mayor puede tener una capacidad de atención excepcional, pero para evaluarla habrá que usar métodos adaptados a sus condiciones físicas, y la naturaleza de un teléfono móvil táctil no permite mucho margen de maniobra en estos casos. Por todo ello, la aplicación tiene como público objetivo cualquier persona que sea capaz de utilizarla en plenas facultades y sin que su condición física interfiera con la evaluación que la aplicación realiza de su capacidad de atención.

Es importante considerar que los problemas de atención son más comunes entre las personas jóvenes, tanto niños como adolescentes. Por lo tanto, sería lógico diseñar la aplicación para ser lo más usable posible para este grupo de edad. Sin embargo, este grupo también coincide con las personas que tienen mayores habilidades tecnológicas, lo que les permitirá manejar la interfaz con mayor facilidad. Además, estas personas no suelen tener problemas físicos graves que les impidan usar un móvil. Por esta razón, aunque sea un público más reducido, es preferible centrarse en la accesibilidad y asegurarse de que la aplicación sea intuitiva para personas de edades más avanzadas, quienes generalmente tienen menos conocimientos y destrezas en el uso de aplicaciones móviles, dando por hecho que el público joven será capaz de manejarla sin problemas, gracias a la naturaleza del propio diseño de la aplicación.[2]

Respecto a la elección de la plataforma, hay que considerar que la aplicación pretende ser usada por el mayor número de personas posible, y ser accesible independientemente del dispositivo del que se disponga. Y cuando se habla de accesibilidad y expansión en el mercado, no hay una plataforma más evidentemente idónea que Android.

Android ha conseguido una posición dominante en el mercado móvil gracias a una combinación de factores clave. En primer lugar, su sistema operativo de código abierto ha permitido una adopción masiva por parte de fabricantes de dispositivos móviles de todo el mundo. Esto ha resultado en muchos dispositivos que ejecutan Android, desde teléfonos inteligentes de gama baja hasta dispositivos insignia de alta gama, lo que ha facilitado que muchos usuarios accedan a la plataforma.

Además, Android ofrece una experiencia altamente personalizable para los usuarios, permitiendo ajustes en la interfaz de usuario, la instalación de aplicaciones de terceros y una integración fluida con servicios de Google. Esta flexibilidad ha atraído a una amplia base de usuarios que buscan dispositivos que se adapten a sus preferencias y necesidades específicas.

La expansión global de Android también ha contribuido a su dominio en el mercado móvil. Google ha trabajado para hacer que Android sea accesible en una variedad de idiomas y para adaptarse a diversas culturas, lo que ha facilitado su adopción en mercados emergentes y en países con diferentes niveles de desarrollo tecnológico, como se puede apreciar en la Figura 1. [3]

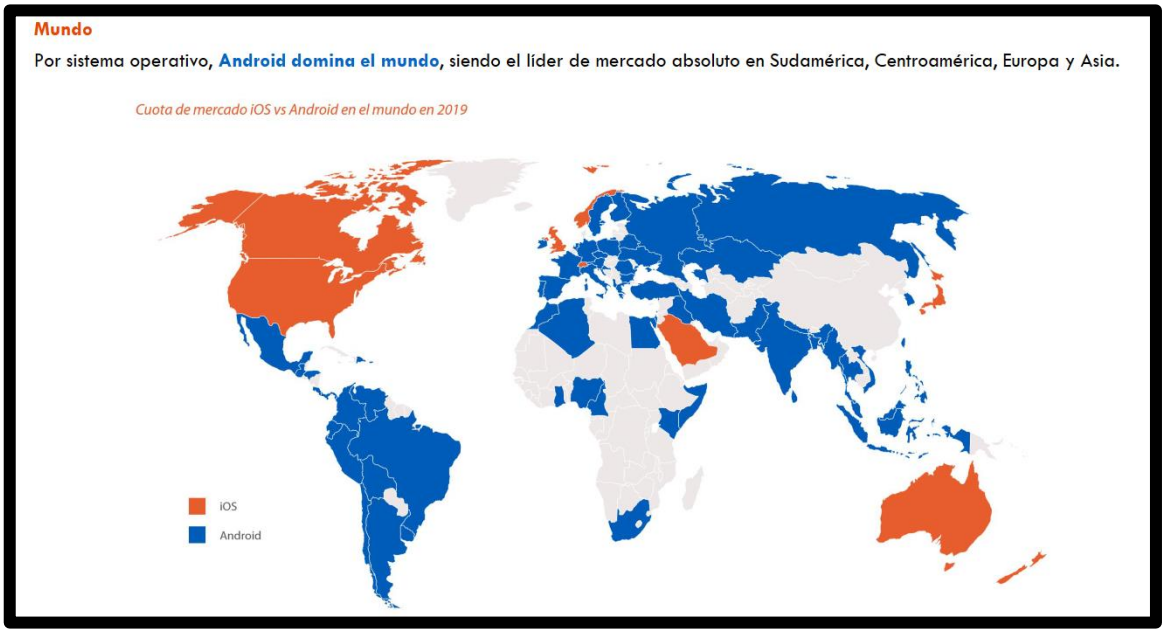


Figura 1. Uso de dispositivos Android en el mundo

Además, Android se ha beneficiado del respaldo de Google y de su ecosistema de aplicaciones y servicios, que incluyen Gmail, Google Maps, YouTube, y el Google Play Store, entre otros. Estos servicios integrados han surgido de manera natural como una respuesta a la necesidad del usuario promedio de usar su teléfono móvil como centro multimedia, para realizar una serie de actividades variadas muy distintas entre sí. En la Figura 2, [3], se muestra un desglose de las actividades que los usuarios tienden a hacer más en sus dispositivos móviles en el mundo y España en concreto:

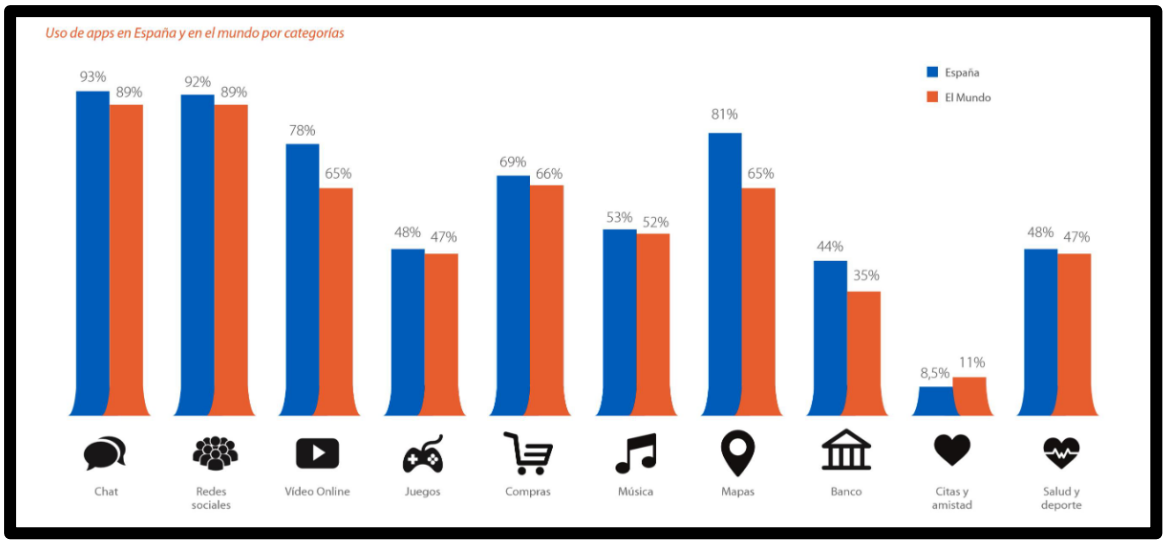


Figura 2. Uso de aplicaciones en España y en el mundo por categorías

Otro factor para tener en cuenta es la transmisión de conocimientos de una aplicación a otra. Un usuario de aplicaciones móviles está acostumbrado a un tipo de interfaz concreta, y el diseño de la aplicación se basa en la idea de mantener la consistencia en el diseño y la distribución de la interfaz de usuario.

Esto significa que, al cambiar de una aplicación a otra, el usuario no debería sentirse perdido o tener que aprender completamente una nueva forma de interactuar. En cambio, se busca que la nueva aplicación siga patrones familiares y convenciones de diseño similares a las que el usuario ya está acostumbrado. Con esta idea en mente se ha diseñado la aplicación para seguir algunos de estos diseños comunes. En la Figura 3 se hace una comparación de las pantallas de inicio de sesión de RejillaApp e Instagram.

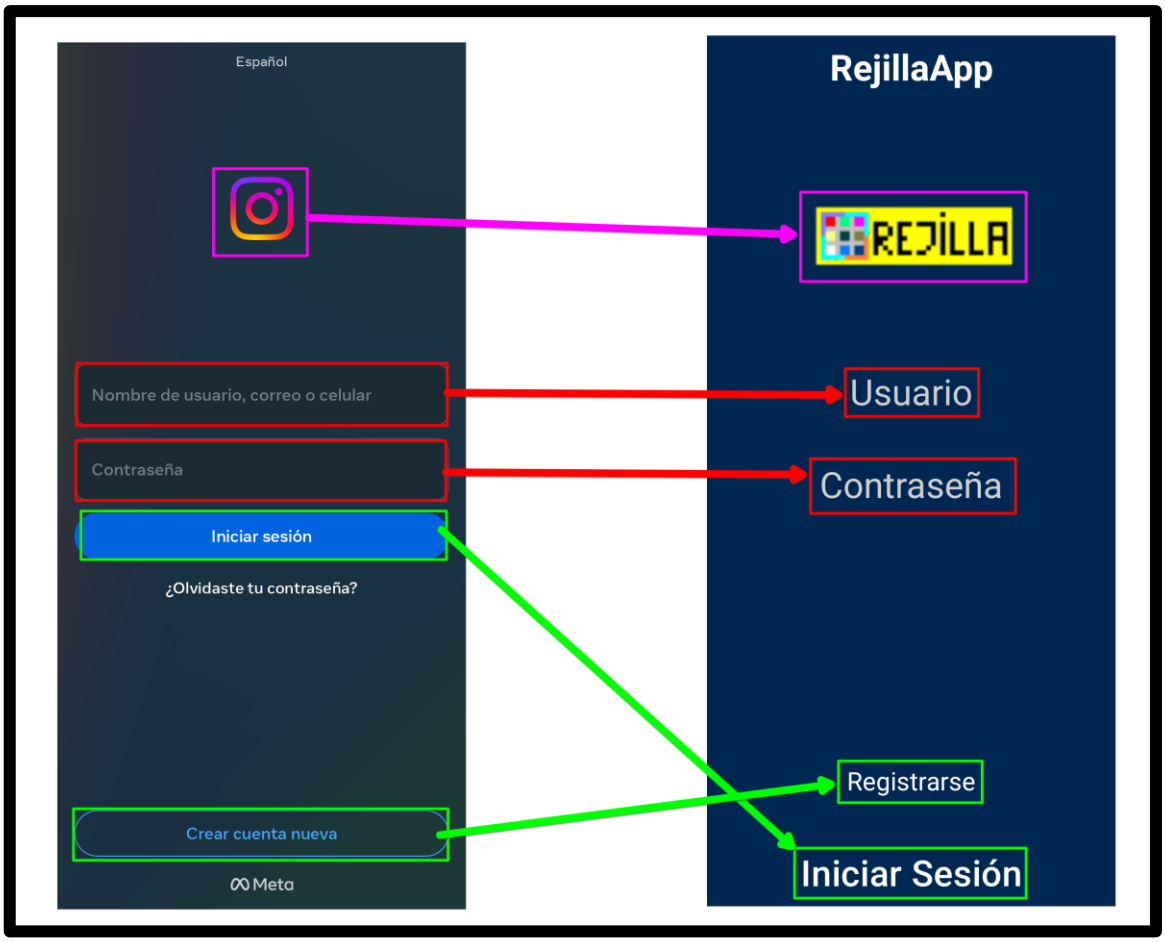


Figura 3. Comparativa con la interfaz de Instagram

Como se puede observar, la estructura de la pantalla de inicio de sesión deja ver como el diseño de las aplicaciones Android tiende a seguir un esquema común que facilite al usuario reconocer los menús e interacciones.

Podemos ver señalado en rosa, como el primer elemento y el más reconocible es el logo de la aplicación, dejando ver al usuario la marca de esta. A continuación, en rojo, se muestran los campos de texto para introducir credenciales, comúnmente usuario y contraseña. Y, por último, en verde, se puede ver como se dan las opciones principales y secundarias de la pantalla. Siendo la mayor y más importante el inicio de sesión, y una menor, pero también presente, el registro.

3.3 Análisis sobre el efecto del tema en el usuario

En el diseño de aplicaciones y dispositivos móviles, un "tema" se refiere a un conjunto de opciones de personalización que determinan el aspecto visual de la interfaz de usuario. Estos temas pueden incluir configuraciones relacionadas con el color, la tipografía, los estilos de los elementos y otros aspectos visuales que afectan la apariencia general de la aplicación o dispositivo.

Una de las características más comunes de un tema es la combinación de colores utilizada para el fondo y el texto. Tradicionalmente, se ha optado por un fondo claro, como blanco o gris claro, con texto oscuro, generalmente negro. Esta combinación proporciona un alto contraste y facilita la legibilidad del texto, lo que resulta en una experiencia de usuario cómoda y fácil de leer, especialmente en condiciones de iluminación estándar.

Sin embargo, con la creciente popularidad de los modos oscuros o "dark mode", la tendencia está cambiando. En estos modos, el fondo tiende a ser oscuro, a menudo negro o gris oscuro, mientras que el texto se muestra en colores claros, como blanco o gris claro. Esto reduce la fatiga visual en entornos con poca luz y puede ayudar a preservar la vida útil de la batería en dispositivos con pantallas OLED, ya que los píxeles oscuros consumen menos energía que los píxeles claros.

La elección entre un tema claro y uno oscuro a menudo se basa en las preferencias del usuario y en el contexto de uso de la aplicación o dispositivo. Algunos usuarios pueden preferir temas oscuros por razones estéticas o de comodidad visual, mientras que otros pueden encontrar que los temas claros son más adecuados para ciertas situaciones, como la lectura durante el día. En las Figuras x y x se ven las pantallas de configuración donde se encuentran las opciones de tema claro y oscuro para dispositivos móviles.

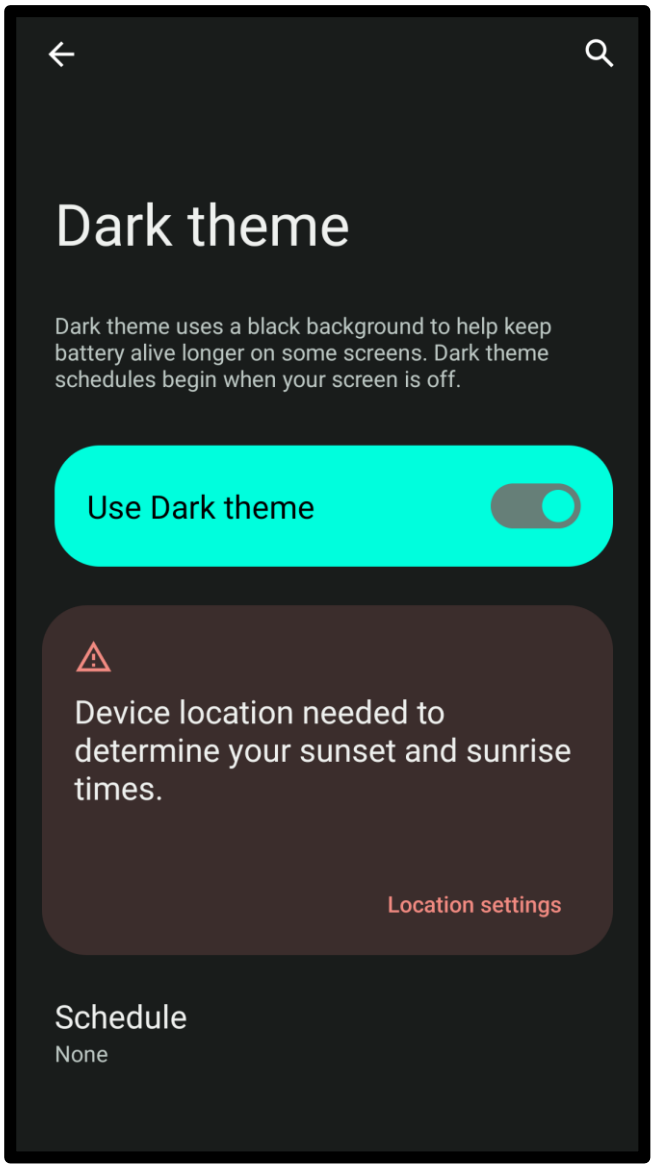


Figura 4. Configuración de tema oscuro de Android

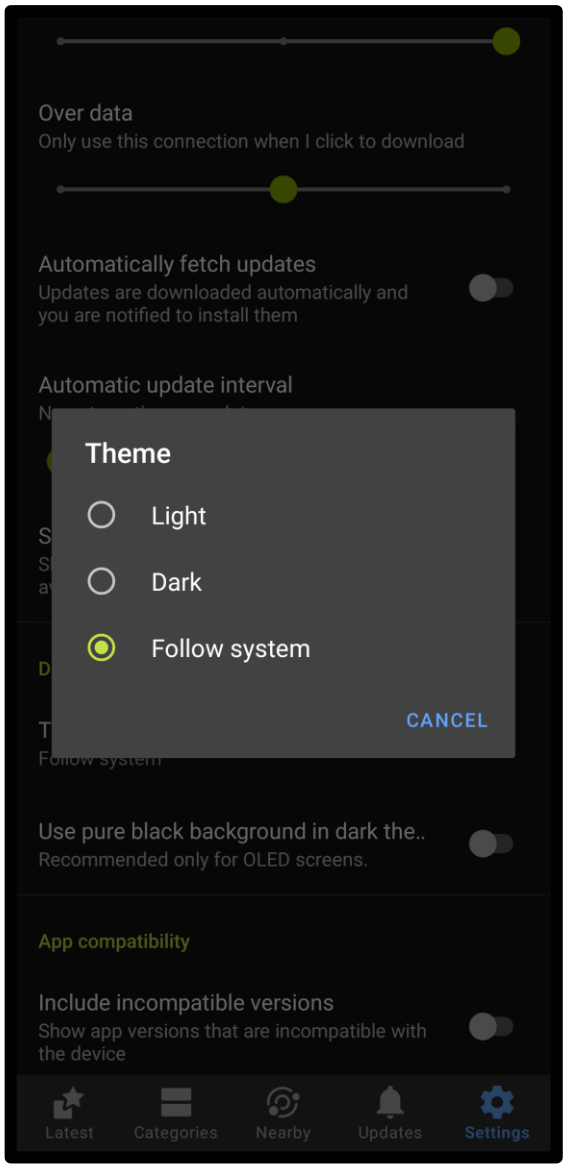


Figura 5. Configuración de tema de F-droid

La elección del tema en una aplicación o dispositivo móvil puede, sin embargo, tener varios beneficios que van más allá de la simple preferencia estética. Uno de los beneficios clave es su impacto en la facilidad de lectura y en la comodidad visual del usuario. Por ejemplo, un tema con fondo oscuro y texto claro puede resultar beneficioso en situaciones de baja luminosidad, como leer en la cama antes de dormir o en entornos con poca luz. La alta legibilidad del texto sobre un fondo oscuro puede reducir la fatiga visual y ayudar a prevenir el cansancio ocular, lo que facilita la lectura prolongada sin causar molestias.

Por otro lado, un tema con fondo claro y texto oscuro puede ser más adecuado en entornos bien iluminados, como al aire libre durante el día o en interiores bien iluminados. El alto contraste entre el texto oscuro y el fondo claro facilita la lectura rápida y la identificación de contenido importante, lo que puede aumentar la eficiencia y la productividad del usuario. Este tipo de temas puede también beneficiar a usuarios con poca visibilidad general.

Esto se traduce en la reafirmación del uso de un tema claro, ya que un tema oscuro puede provocar dificultades a la hora de concentrarse en un solo objetivo, especialmente si el usuario ya tiene problemas de atención en un primer momento. En conclusión, se ha optado por utilizar un tema claro, en contraposición al uso de un tema oscuro, ya que a pesar de que este último sea bastante popular en ámbitos más cercanos al usuario avanzado o más acostumbrado a la tecnología, existe una gran cantidad de usuarios, entre ellos mayormente personas de avanzada edad, para los que un tono oscuro puede dificultar bastante la visión. Utilizando un tema claro se consigue facilitar el uso y entendimiento de la aplicación a la mayor parte de usuarios posible.

3.4 Colores y marca

La elección del color para una aplicación, especialmente cuando se trata del tema principal de la misma, es un aspecto crucial del diseño que puede tener un impacto significativo en la personalidad de la aplicación y en la forma en que es percibida por los usuarios. Los colores pueden evocar emociones y asociaciones específicas. Por ejemplo, un tono oscuro y azulado puede transmitir una sensación de profesionalidad, confianza y seriedad, que son atributos deseables para una institución educativa. Esto puede influir en la forma en que los usuarios perciben la aplicación, ayudando a establecer una impresión positiva desde el principio.

En consecuencia, se ha decidido usar el color principal de la Universidad de Málaga (**#002751** ■) para el diseño de la aplicación. Este color será entendido como el color

representante de la marca de la aplicación RejillaApp, ya que es el color asociado con la Universidad de Málaga, de esta forma se entenderá de manera obvia el origen de la aplicación. Esto puede ayudar a reforzar la marca de la universidad, ya que el color elegido es reconocible y fácilmente asociable, además de ser una forma fácil y rápida de mostrar la relación con la aplicación.

Además de su impacto emocional, es importante considerar la legibilidad y la accesibilidad al elegir colores para una aplicación. Asegurarse de que el color principal seleccionado tenga suficiente contraste con el texto y otros elementos visuales garantizará que la aplicación sea fácil de usar para todos los usuarios, incluidos aquellos con discapacidades visuales o de atención. Lo más apropiado será usar un texto claro (preferiblemente blanco) cuando se necesite y se tenga un fondo con un color más oscuro como el elegido.

El color elegido se usará de forma repetida en el diseño de la aplicación con el objetivo de mantener la coherencia visual en todo el diseño. Esto significa que todos los elementos de la interfaz gráfica de la aplicación, como botones, fondos, texto y otros elementos, estarán alineados visualmente. La coherencia visual facilita la comprensión de la interfaz por parte de los usuarios y crea una experiencia más armoniosa y agradable.

Además, reducir la cantidad de colores utilizados en el diseño, simplifica la apariencia general de la aplicación. Una paleta de colores limitada hace que la interfaz sea más clara y fácil de entender para los usuarios. Esto es especialmente importante en esta aplicación, ya que la atención del usuario podría estar limitada, y la claridad es fundamental para la usabilidad incluso si el usuario no tiene problemas de atención. Esta decisión ayuda también a mantener la imagen de marca y facilita el desarrollo del diseño.

Este color principal influirá en el diseño y la gama de colores de:

- Botones
- Líneas separadoras
- Texto destacado
- Color de fondo de los títulos de cada actividad Android
- Y otros apartados

3.5 Principios de usabilidad en la ingeniería del software

El estándar internacional para la evaluación de la calidad del software ISO/IEC 9126 [4] define usabilidad como un conjunto de atributos relacionados con el esfuerzo necesario para su uso, y en la valoración individual de tal uso, por un establecido o implicado conjunto de usuarios. Estos atributos son:

- **Aprendizaje:** Atributos del software que se relacionan al esfuerzo de los usuarios para reconocer el concepto lógico y sus aplicaciones.

- **Comprensión:** Atributos del software que se relacionan al esfuerzo de los usuarios para reconocer el concepto lógico y sus aplicaciones.

- **Operatividad:** Atributos del software que se relacionan con el esfuerzo de los usuarios para la operación y control del software.

Siendo que esta es una aplicación dedicada potencialmente a personas con problemas de atención, se debe considerar además que el usuario puede llegar a presentar estos problemas que la aplicación intenta evaluar, y por tanto se debe facilitar al máximo el uso por parte de un usuario con problemas de atención.

La aplicación debe guiar al usuario a través de cada paso de manera clara y comprensible, usando un diseño limpio y simple que reduzca la carga cognitiva. Los elementos visuales y textuales deben ser cuidadosamente seleccionados para mantener la atención del usuario y evitar la sobrecarga de información.

La experiencia de usuario debe ser fluida y consistente, con transiciones suaves y tiempos de respuesta rápidos para mantener el interés del usuario y evitar la frustración. Las instrucciones y mensajes dentro de la aplicación deben ser concisos y directos, empleando un lenguaje claro y sin tecnicismos que complican la comprensión, pero informando bien de lo que sucede en la aplicación, usando mensajes flotantes de Android.

Es importante además considerar la personalización de la experiencia, permitiendo al usuario seleccionar que funcionalidades quiere aplicar al crear la prueba de evaluación de la atención.

4

Tecnologías

4.1 Consideraciones previas a la elección de tecnologías

El proyecto se plantea desde un primer momento como una aplicación para evaluar la atención del usuario de distintas formas. Con este objetivo en mente, la elección de Android tiene sentido, siendo esta una plataforma bastante accesible y extremadamente popular. Se estima actualmente el número de usuarios de Android en más de tres mil millones, siendo el sistema operativo móvil más usado, y siendo incluso el sistema operativo más usado a nivel general. [5] [6]

Android es la plataforma ideal para llegar a la mayor cantidad de usuarios posible debido además a su naturaleza de código abierto, que ha permitido una amplia adopción por parte de fabricantes de dispositivos móviles. Esto ha resultado en una diversidad de dispositivos que van desde modelos de gama baja hasta dispositivos insignia de alta gama. Además, la flexibilidad y personalización que ofrece Android, junto con su integración con servicios de Google, atraen a una amplia base de usuarios que buscan adaptar sus dispositivos a sus necesidades específicas. La expansión global de Android,

con soporte en múltiples idiomas y adaptación a diversas culturas, ha contribuido aún más a su dominio en el mercado móvil, haciéndolo la plataforma ideal para la materialización de este proyecto.

4.2 Android

Android está basado en Linux junto con otros softwares de código abierto, es un sistema operativo móvil que ha revolucionado la industria de la tecnología móvil desde su debut en 2007. Está diseñado para una amplia variedad de dispositivos móviles, como teléfonos inteligentes, tabletas, relojes inteligentes y televisores, Android es una plataforma versátil y adaptable tanto para los usuarios como para los desarrolladores de software.[7]

Android ha logrado establecerse en diversos ámbitos de la vida moderna, convirtiéndose en una opción recurrente para las tareas del día a día, desde la comunicación personal hasta la productividad laboral y el entretenimiento. Su versatilidad y accesibilidad lo han posicionado como una opción preferida por una amplia gama de usuarios en todo el mundo, desde el usuario doméstico hasta el profesional o más avanzado con conocimientos técnicos, Android ha encontrado lugar en una gran variedad de contextos. En la Figura 6, se ve una representación visual de como Android encuentra su lugar, aportando funcionalidades en distintos ámbitos cotidianos.

Figura 6. Android en distintos ámbitos de la vida cotidiana

En los hogares, proporciona acceso a redes sociales, aplicaciones de mensajería, juegos y contenido multimedia, atendiendo las necesidades de entretenimiento y conectividad de personas de todas las edades. En entornos laborales, se involucra desde la gestión de correos electrónicos hasta la colaboración en documentos y presentaciones, ofreciendo herramientas productivas que satisfacen las necesidades de profesionales y empresarios. En el ámbito educativo también ha destacado como una herramienta de aprendizaje versátil, brindando acceso a aplicaciones educativas y recursos en línea que mejoran la experiencia de aprendizaje.

Comparativamente, frente a otros sistemas operativos móviles como iOS y Windows Phone, Android destaca por su mayor grado de personalización, su amplia variedad de dispositivos disponibles en el mercado y su ecosistema de aplicaciones robusto. Mientras que iOS tiende a atraer a usuarios que valoran la simplicidad y la integración con otros productos de Apple, Android es preferido por aquellos usuarios que valoran las opciones de personalización y una amplia gama de precios y características de hardware. Aunque Windows Phone ofrecía una experiencia única con su integración con servicios de Microsoft y su interfaz basada en azulejos, Android ha logrado superar a estos competidores en términos de disponibilidad de aplicaciones, variedad de dispositivos y cuota de mercado. [8]

En el siguiente punto se hablará de cómo, además de las características que hacen de Android la plataforma ideal en términos de accesibilidad y usabilidad para usuarios, es también una plataforma extremadamente accesible para desarrolladores, contando con una herramienta realmente potente para desarrollar aplicaciones para esta plataforma, Android Studio.

4.2.1 Android Studio

Android Studio es la herramienta oficial de desarrollo integrada (IDE, en inglés "*Integrated development environment"*) para la creación de aplicaciones Android, desarrollada y mantenida por Google. Lanzado en 2013, Android Studio está basado en

IntelliJ IDEA y proporciona un entorno robusto y completo para el desarrollo de aplicaciones móviles.

Entre sus características destacadas, Android Studio ofrece un editor de código avanzado con capacidades de autocompletado, análisis de código y navegación rápida. Su sistema de construcción está basado en Gradle, lo que permite una configuración flexible y escalable de los proyectos. Además, incluye un emulador de Android de alto rendimiento que facilita la prueba de aplicaciones en diversas configuraciones de hardware y versiones del sistema operativo. [9]

Android Studio también proporciona un conjunto de herramientas de depuración y pruebas, como el *Layout Editor*, [10], que permite diseñar interfaces de usuario visualmente, y el *Profiler*, [11], que ayuda a monitorear el rendimiento de las aplicaciones en términos de CPU, memoria y uso de red. Estas herramientas son esenciales para optimizar la eficiencia y la experiencia del usuario final. En la Figura 7, se encuentra un ejemplo de la interfaz de Androd Studio donde se observan varias de las herramientas anteriormente mencionadas.

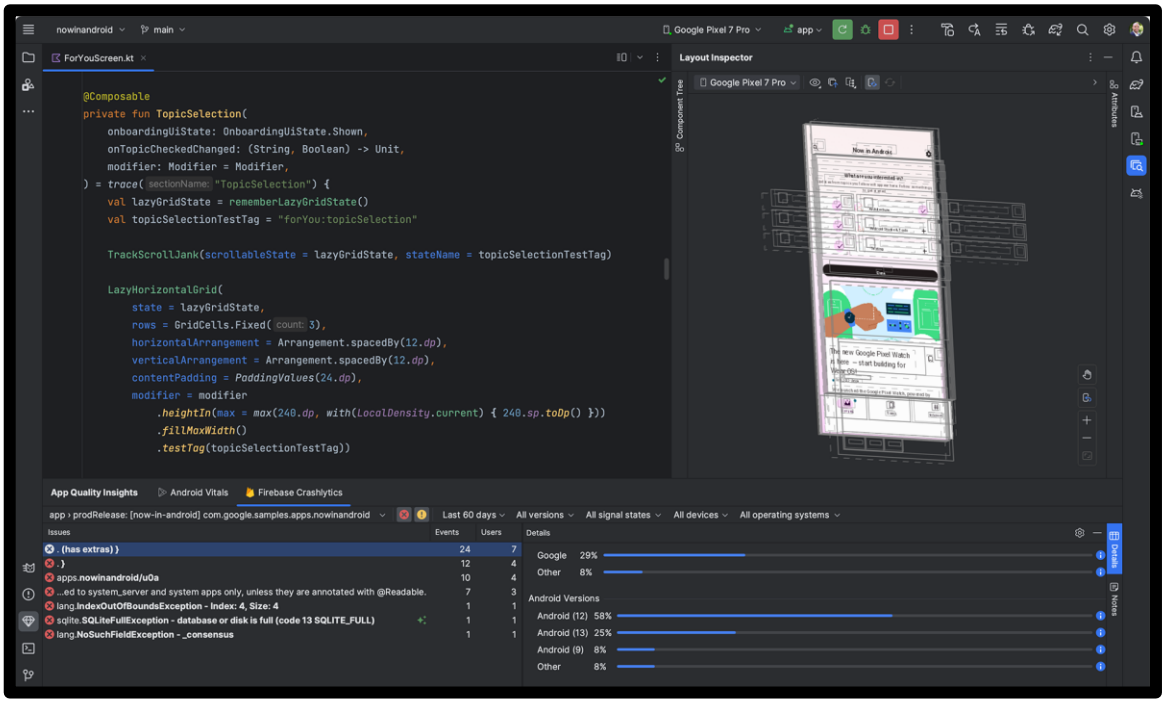


Figura 7. Interfaz de Android Studio

Una de las mayores ventajas de Android Studio es su integración con los servicios de Google Cloud Platform, [12], una plataforma que reúne distintos tipos de servicios en la nube con capacidades para facilitar el desarrollo de aplicaciones móviles, incluyendo funcionalidades muy útiles como: bases de datos en tiempo real, autenticación de usuarios y análisis de uso. Esto permite a los desarrolladores añadir funcionalidades avanzadas a sus aplicaciones con relativa facilidad.

Comparado con otras herramientas de desarrollo, Android Studio se destaca por su soporte directo de Google, actualizaciones frecuentes y una comunidad de desarrolladores activa que contribuye con plugins y extensiones. Si bien existen otras herramientas como Eclipse con ADT, [13], ("*Android Development Tools"*); un plugin para el IDE Eclipse que permite construir aplicaciones Android, y Visual Studio con Xamarin, [14]; una plataforma que permite realizar un desarrollo dual, donde el código de la aplicación final funciona tanto en iOs como Android y también permite el desarrollo desde MAC o Windows, en definitiva, Android Studio se considera el estándar de la industria para el desarrollo de aplicaciones Android debido a su enfoque completo y especializado.

4.2.2 Java y Kotlin como opciones dentro de Android Studio

Dos de los lenguajes más utilizados hoy en día son Java y Kotlin. Ambos tienen características únicas y ventajas que los hacen atractivos para diferentes tipos de desarrolladores y proyectos. En este análisis vamos a dar datos comparativos entre Java y Kotlin, evaluar sus características dentro de Android Studio y determinar por qué Java sigue siendo una opción viable y popular a pesar del creciente interés en Kotlin. [15]

Java, [16], es un lenguaje de programación de propósito general, orientado a objetos, desarrollado por Sun Microsystems (ahora Oracle) y lanzado en 1995. Desde sus inicios, ha sido el principal lenguaje para el desarrollo de aplicaciones Android. Sus principales características incluyen madurez y estabilidad, ya que cuenta con más de dos décadas de desarrollo y mejoras constantes, lo que le ha permitido establecerse como un

lenguaje robusto y confiable. Dispone de un amplio ecosistema de bibliotecas y herramientas que facilitan el desarrollo de aplicaciones complejas. Además, gracias a la máquina virtual Java (JVM), [17], el código Java es altamente portable entre diferentes plataformas. Java también tiene una de las comunidades de desarrolladores más grandes y activas, con soporte invaluable mediante foros, documentación y recursos educativos. En la Figura 8, se observa la popularidad de Java en una de las páginas más populares para desarrolladores en el mundo.

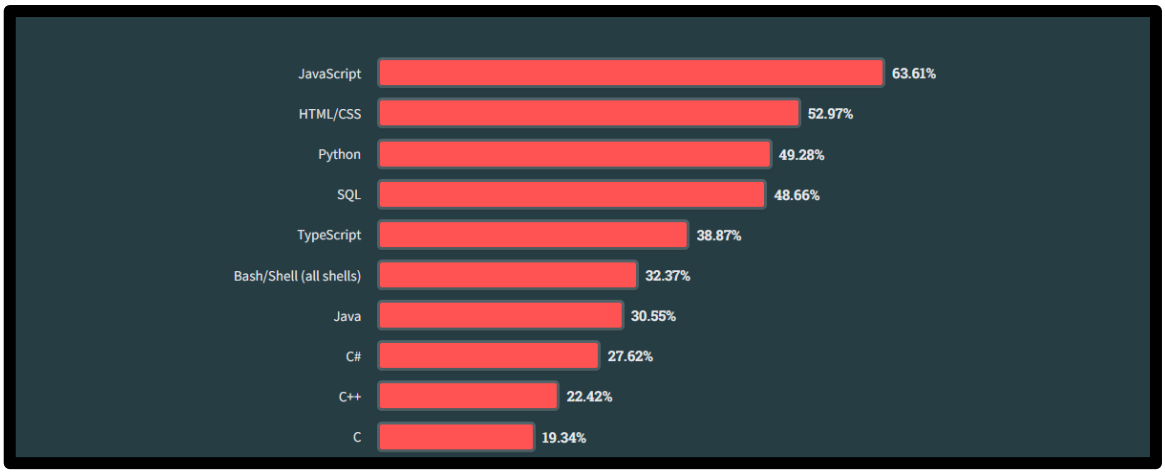


Figura 8. Java en el top 10 tecnologías más usadas en 2023 en la página StackOverflow. [18]

Por otro lado, Kotlin, [19], es un lenguaje de programación moderno, estático, desarrollado por JetBrains y lanzado en 2016. Ha sido diseñado para interoperar completamente con Java, lo que ha facilitado su adopción en el desarrollo de Android. Algunas de sus características destacadas son su sintaxis concisa y moderna, que reduce la cantidad de código necesario para realizar tareas comunes, mejorando la legibilidad y disminuyendo el riesgo de errores. Además, permite utilizar bibliotecas de Java sin necesidad de migrar el código existente, facilitando la integración gradual en proyectos actuales. Kotlin también introduce características como la seguridad de nullabilidad que ayuda a evitar errores comunes relacionados con los null pointers, [20], y desde 2017, Google ha promovido Kotlin como un lenguaje oficial para el desarrollo de Android, impulsando su adopción y crecimiento.

En términos de facilidad de aprendizaje, Java tiene una curva de aprendizaje más pronunciada, debido a su sintaxis detallada y la necesidad de entender conceptos tradicionales de programación orientada a objetos. En contraste, Kotlin ofrece una sintaxis más intuitiva y menos complicada, lo que facilita el aprendizaje y la transición desde otros lenguajes. Aunque la sintaxis de Java, conocida por su nivel de detalle, pueda resultar en código más extenso, ofrece una estructura clara y tradicional. Si hablamos de compatibilidad y ecosistema, Java tiene una ventaja significativa debido a su madurez y a un ecosistema bien establecido con numerosas bibliotecas y recursos. Es importante considerar, además, la comunidad y el soporte, aspectos cruciales en la experiencia del desarrollo de software. Java cuenta con una de las comunidades más grandes y activas, ofreciendo numerosos recursos y soporte. Kotlin aunque es más joven también ha formado una comunidad entusiasta y en crecimiento, respaldada por JetBrains, [21], y Google. Esta comunidad activa contribuye con materiales educativos y bibliotecas específicas para Kotlin, facilitando su aprendizaje y uso en proyectos de Android.

En conclusión, ambos lenguajes tienen un papel relevante en el desarrollo de software, y la capacidad de interoperar permite aprovechar las fortalezas de ambos según convenga. Java sigue siendo una opción viable y robusta para muchos proyectos, mientras que Kotlin ofrece una alternativa moderna y eficiente, Java sigue siendo una opción preferida por muchas razones. La larga trayectoria de Java lo convierte en una opción segura y estable para proyectos a gran escala. La basta cantidad de herramientas disponibles para Java facilita el desarrollo de aplicaciones complejas, y su compatibilidad con sistemas existentes permite que las organizaciones continúen utilizando sus proyectos en Java sin problemas. Además, la enorme comunidad de desarrolladores y la abundante documentación de Java ofrecen un soporte continuo y recursos valiosos para el desarrollador.

4.3 Gradle

Gradle, [22], es una herramienta de automatización de compilación de código abierto, diseñada para ser flexible y eficiente. Los scripts de compilación de Gradle se escriben en Groovy, [23], o Kotlin DSL (Domain Specific Language). Su flexibilidad permite utilizar diversos lenguajes, no solo Java, y cuenta con un robusto sistema de gestión de dependencias. Además, Gradle es altamente personalizable y rápido, ya que reutiliza las salidas de ejecuciones anteriores y procesa solo las entradas que han cambiado, optimizando la velocidad de compilación a través de la paralelización de tareas.

En el caso de Android, Gradle es el sistema de compilación oficial, soportado directamente por Android Studio. Esto significa que Android Studio delega todas las tareas de construcción a Gradle, asegurando una compilación precisa y eficiente, tanto si se ejecuta desde el entorno de desarrollo como desde la línea de comandos o un servidor de integración continua, [24]. Los proyectos Android que usan Java se benefician de la capacidad de Gradle para manejar dependencias, variantes de compilación, empaquetado y distribución de archivos JAR, [25], WAR, [26], y EAR, [27].

En un proyecto Android típico que utiliza Java, Gradle gestiona varios aspectos esenciales. Por ejemplo, el archivo build.gradle, [28], a nivel superior del proyecto define configuraciones comunes para todos los subproyectos o módulos, incluyendo repositorios y dependencias necesarias para la construcción. Este archivo se encuentra en el directorio raíz del proyecto.

Por otro lado, cada módulo del proyecto, como la aplicación principal, tiene su propio archivo build.gradle que especifica detalles particulares como el ID de la aplicación, la versión del SDK de compilación, las dependencias y las configuraciones específicas de compilación y variantes de producto. En la Figura 9, se presenta lo anteriormente mencionado en forma de diagrama.

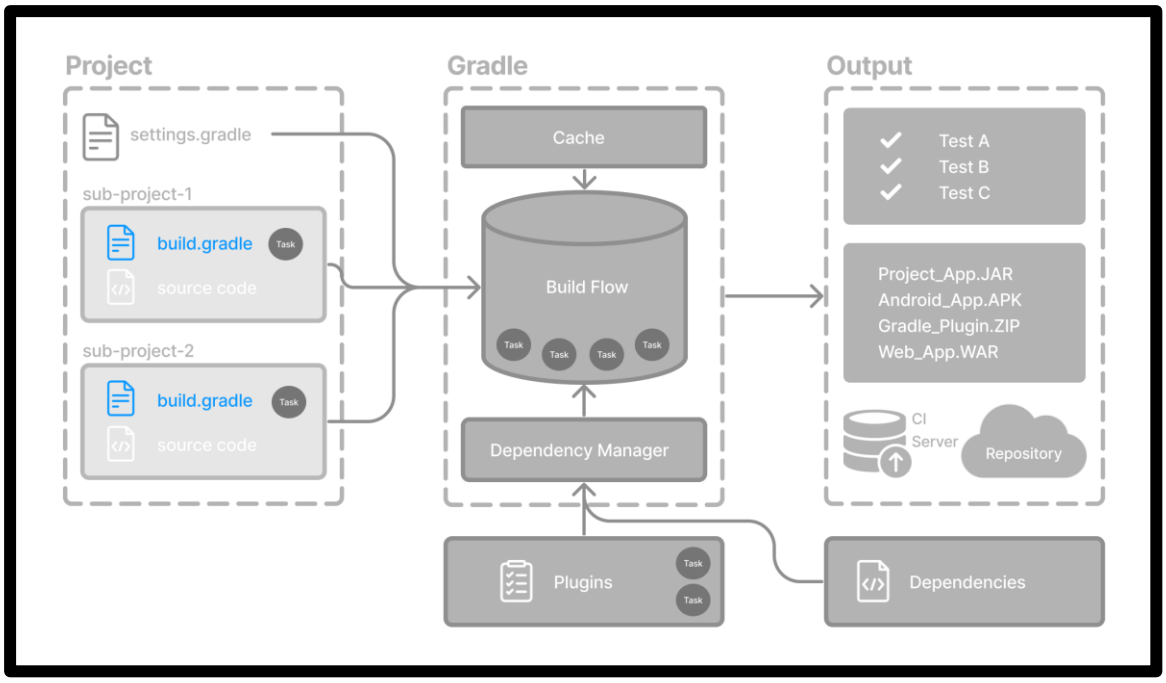


Figura 9. Diagrama de uso de fichero build.gradle

Otro aspecto crucial del uso de Gradle en proyectos Android es la configuración del entorno de compilación. La personalización se logra a través de diferentes archivos, como gradle.properties, [29], donde se pueden ajustar parámetros de memoria para la JVM, o configuraciones de compilación incremental que optimizan el tiempo de construcción.

El sistema de gestión de dependencias, [30], de Gradle es otro punto fuerte. Permite declarar y manejar dependencias locales y remotas de manera eficiente, asegurando que todas las bibliotecas necesarias estén disponibles para la compilación, como se puede observar en el diagrama de la Figura 10. Esto incluye bibliotecas de soporte de Android, servicios de Google Play, y cualquier otra dependencia necesaria para el proyecto.

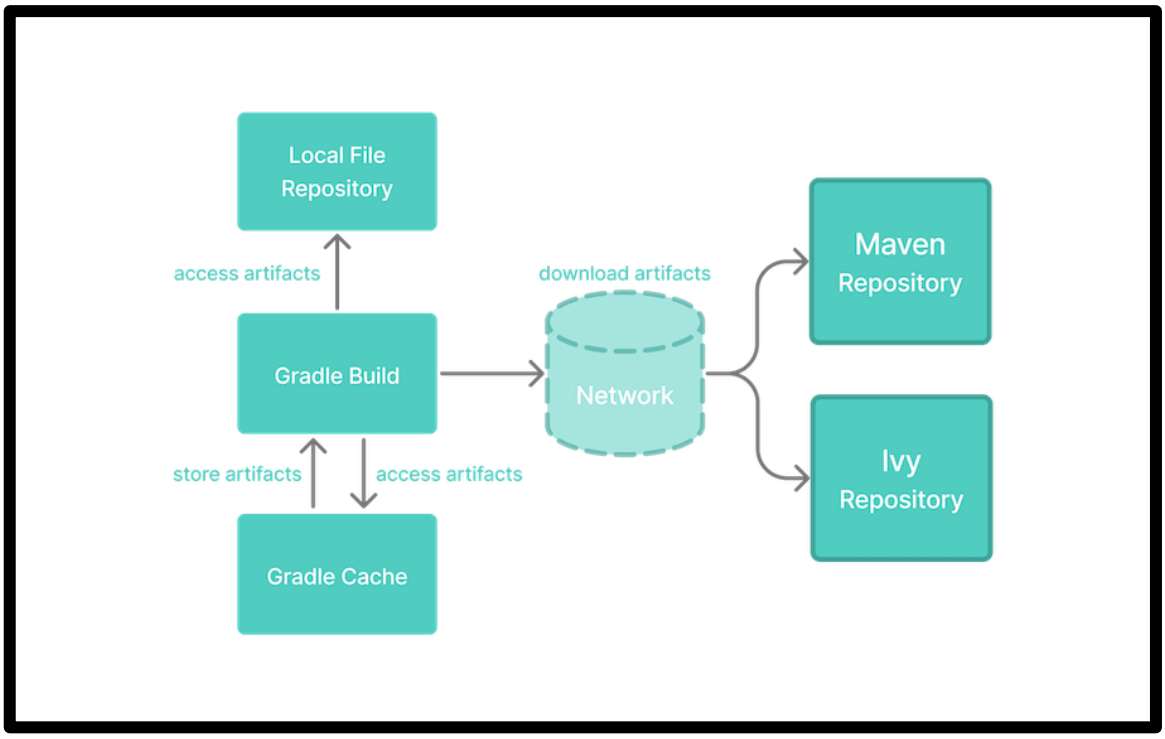


Figura 10. Manejo de dependecias en Gradle

A pesar de su robustez, trabajar con Gradle puede presentar desafíos, como errores de sincronización de dependencias o problemas de memoria. Estos problemas pueden solucionarse ajustando configuraciones en los archivos build.gradle y gradle.properties, y utilizando herramientas de Android Studio para analizar y optimizar el proceso de construcción.

4.4 SQLite

SQLite, [31], es una base de datos ligera y fácil de usar que no necesita instalación ni configuración. Usa un motor de base de datos SQL transaccional, de código abierto y ligero, que se integra directamente en las aplicaciones, eliminando la necesidad de configuraciones adicionales de servidor. Se utiliza ampliamente en aplicaciones móviles, incluyendo Android, para almacenar datos de manera eficiente. No necesita un servidor separado, así que cualquier aplicación en Android puede acceder directamente a la base de datos almacenada en un solo archivo. Esto simplifica el manejo de la base de datos dentro de la aplicación y facilita la transferencia de datos entre dispositivos.

Además, SQLite es muy pequeño, ocupando su código menos de 1MB, lo que es perfecto para aplicaciones móviles que necesitan ser eficientes en el uso de espacio y recursos. Permite almacenar cualquier tipo de dato en cualquier columna, lo que le da mucha flexibilidad a la aplicación. Utiliza solo el espacio en disco necesario para los datos que se guardan, haciendo que funcione más rápido. Su código es claro y fácil de entender, lo que facilita el desarrollo y mantenimiento de la base de datos.

Para usar SQLite en una aplicación Android, se puede aprovechar el soporte nativo que ofrece Android Studio. Primero, se crea o abre una base de datos usando la clase “SQLiteOpenHelper”, [32]. Esta clase ayuda a manejar la creación, actualización y manejo de la base de datos.

En el método “onCreate”, [33], de “SQLiteOpenHelper”, se definen las tablas de la base de datos usando sentencias SQL. Para insertar, actualizar o eliminar datos, se utilizan métodos como “insert”, “update” y “delete” del objeto “SQLiteDatabase”, el cual se pasa por parámetros al método. Las consultas se pueden hacer usando el método “query” o escribiendo directamente sentencias SQL. Para que sea más fácil de ilustrar disponemos de un ejemplo en la Figura 11xdonde se encuentran listadas diferentes clases y métodos.

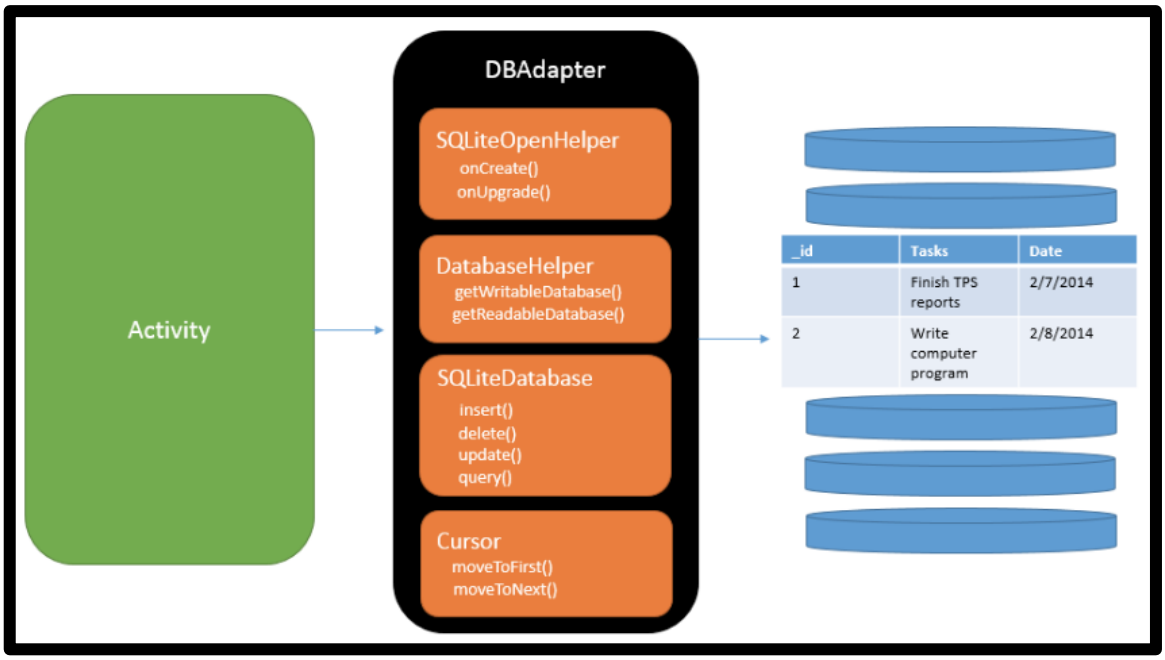


Figura 11. Ejemplo de uso de las clases de SQLite

Con respecto a otras tecnologías de bases de datos para aplicaciones móviles, SQLite tiene muchas ventajas. Al no necesitar un servidor, reduce la complejidad y el costo de mantenimiento. Su pequeño tamaño hace que sea ideal para dispositivos móviles con espacio limitado. Además, la flexibilidad en el manejo de tipos de datos y el uso eficiente del espacio en disco hacen que las aplicaciones funcionen más rápido y sean más fáciles de desarrollar. La capacidad de trabajar con una base de datos en un solo archivo facilita la copia y transferencia de datos. Y por todas esas ventajas ha sido la base de datos escogida este proyecto.

5

Desarrollo de la aplicación Android

5.1 Consideraciones previas

Tras un análisis de las funcionalidades de la aplicación que se quiere adaptar y consultas con el cliente, se han llegado a establecer en orden diversos requisitos a implementar en cada iteración, teniendo como objetivo principal la presentación de versiones frecuente al cliente para evaluar el estado del proyecto en cada momento tras la implementación de requisitos.

Para la especificación de los requisitos se usa una plantilla personalizada adaptada al proyecto, basada en la plantilla de Team Asana [34]. En esta plantilla se especifican secciones para la especificación de requisitos de Software:

- Una introducción a modo de descripción rápida del requisito, mencionando cómo se usará.
- Alcance del producto para explicar los objetivos comerciales del mismo.
- Valor del producto para definir que aportará y porque es importante.
- Público objetivo para decir a quién va dedicado el producto.
- Uso previsto para definir para que se usará el producto.
- Descripción general con las características y funciones del Software.

A partir de estas secciones se han elegido y remodelado las más relevantes para el proyecto, obteniendo una plantilla que se adapta de forma ideal al mismo.

La plantilla consiste en el siguiente esquema:

Nombre	Identificador del requisito
Introducción	Breve introducción con el propósito del requisito
Objetivo	Descripción del problema a solucionar por el requisito
Uso previsto	Definición de las situaciones en las que se usará la solución dada por el requisito
Planteamiento de diseño	Breve planteamiento del diseño técnico de la solución
Diseño de la solución	Descripción detallada del diseño de la solución

5.2 Consideraciones sobre el *refactoring* y la modificación de la solución a lo largo del desarrollo

Con el análisis de algunos requisitos y sus soluciones a lo largo del desarrollo, se puede plantear la reestructuración del código o la modificación de la solución para adaptar esta al resto del proyecto durante el desarrollo del mismo.

Es también común en el desarrollo software el uso de *refactoring*, una técnica de la ingeniería de software que consiste en reestructurar el código fuente, alterando su estructura interna sin cambiar su comportamiento externo. Esto será usado repetidamente durante el desarrollo para optimizar código, hacerlo más legible y permitir una mayor escalabilidad durante el desarrollo o tras este.

El *refactoring* se utiliza también en ingeniería de software para permitir la introducción a futuro de nuevas funcionalidades, y no implica como tal la solución de errores o bugs, sin embargo, a lo largo del desarrollo del proyecto, dado el modelo de iteraciones elegido, se presentarán soluciones a errores y mejoras al código que pueden no entrar en la definición de *refactoring*. [35]

5.3 Diagrama de casos de uso

Un diagrama de casos de uso es una herramienta visual utilizada en el desarrollo software para representar las interacciones entre los usuarios (actores) y las funcionalidades (casos de uso) que el sistema proporciona.

En el diagrama existen tres elementos:

- **Los actores:** Entidades externas que interactúan con el sistema, como usuarios, otros sistemas, o dispositivos. Se representan con una figura humana hecha de palitos

- **Los casos de uso:** Descripciones de las funciones que el sistema ofrece a los actores y se representan con óvalos. Cada óvalo describe una acción o proceso que un actor puede realizar con el sistema.

- **Las relaciones entre actores y casos de uso:** Muestran cómo los usuarios interactúan con el sistema. Estas interacciones se representan con líneas que conectan actores y casos de uso, indicando que el actor participa en esa funcionalidad. Existen dos relaciones especiales:

    - **<<include>>:** Indica que un caso de uso siempre incluye el comportamiento de otro caso de uso.

    - **<<extend>>:** Se utilizan para mostrar comportamientos compartidos o extendidos entre los casos de uso. Muestra que un caso de uso agrega funcionalidad opcional a otro caso de uso.

El diagrama de casos de uso sirve para varias funciones importantes en el desarrollo de software. Facilita la comprensión de los requisitos funcionales al proporcionar una vista clara y concisa de lo que el sistema debe hacer desde la perspectiva del usuario. Ayuda a identificar y definir los límites del sistema, especificando qué interacciones son internas y cuáles son externas. Además, permite detectar posibles mejoras y optimizaciones en el diseño del sistema al visualizar cómo se interrelacionan las diferentes funciones.

El diagrama de casos de uso del sistema se muestra en la Figura 12.

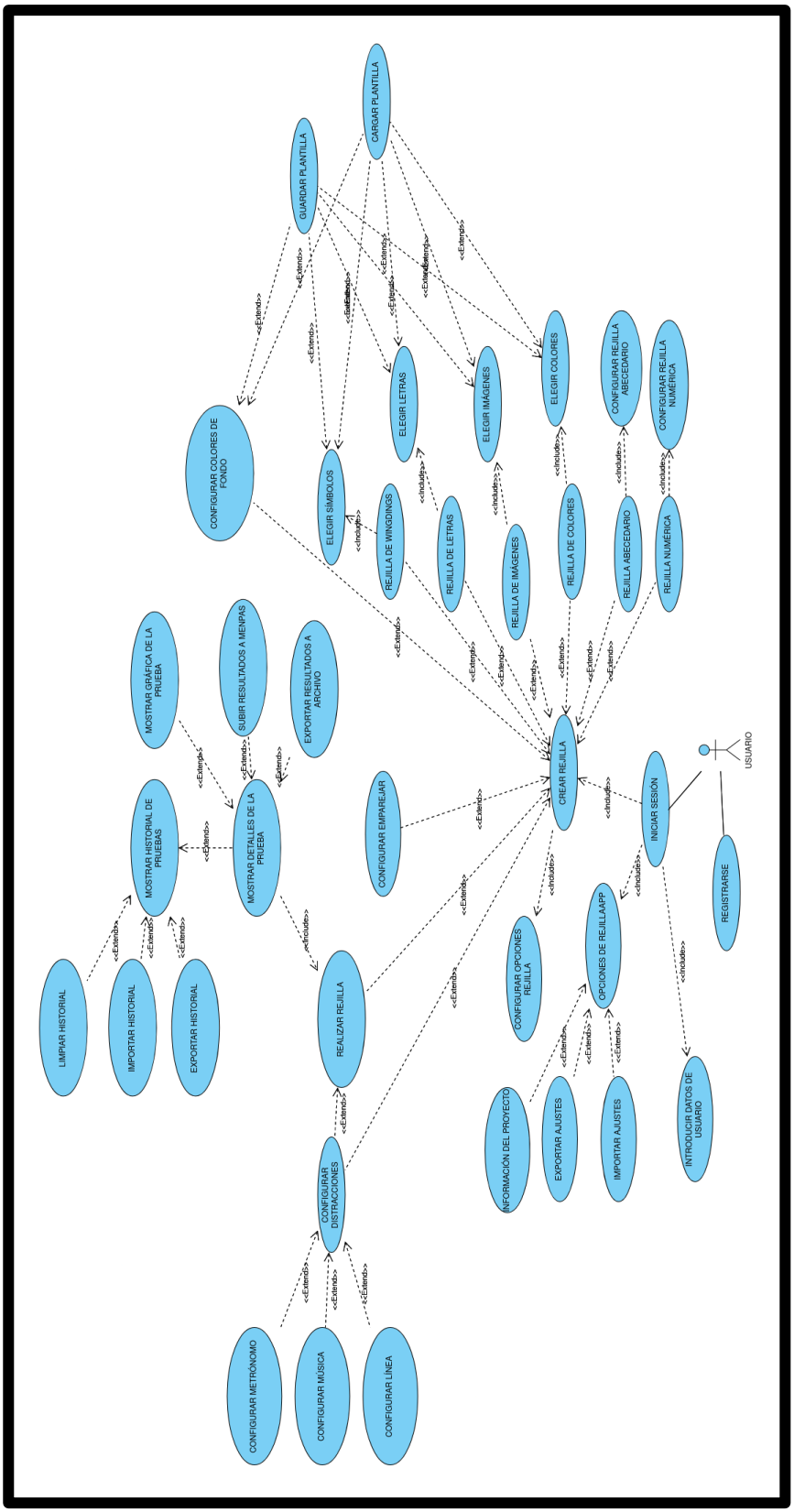


Figura 12. Diagrama de casos de uso RejillaApp

A partir de este diagrama de casos de uso se realizarán las siguientes iteraciones:

- **Primera iteración:** Dedicada a la interfaz y las rejillas

- **Segunda iteración:** Rejillas restantes, algunas funcionalidades extra, la base de datos y la recogida de resultados del usuario

- **Tercera iteración:** Funcionalidades y rejillas restantes

- **Cuarta iteración:** Funcionalidades en línea y ajustes

5.4 Backlog

Requisitos funcionales:

Estos describen las funciones y características específicas que el sistema debe realizar. Se enfocan en el comportamiento del sistema bajo ciertas condiciones y en la interacción entre el sistema y sus usuarios. En esencia, los requisitos funcionales responden a la pregunta "¿qué debe hacer el sistema?".

Los requisitos funcionales del proyecto son:

- Pantalla de inicio
- Menú de creación de rejilla
- Seleccionar filas y columnas
- Seleccionar tipo de rejilla
- Pantalla de creación rejilla numérica
- Pantalla de creación rejilla colores
- Selección de colores ColorPicker
- Pantalla de creación rejilla de elementos
- Pantalla de creación rejilla abecedario
- Seleccionar cantidad de botones rejilla
- Creación de rejillas: Estructura general

- Pantalla elegir elementos
- Creación de rejillas: Rejilla letras

- Tarea preliminar
- Distracción línea
- Distracción metrónomo
- Datos de usuario
- Gráfica de tiempo de botones
- Historial de pruebas
- Pantalla de creación rejilla imágenes
- Creación de rejillas: Rejilla numérica
- Creación de rejillas: Rejilla colores
- Creación de rejillas: Rejilla imágenes
- Registro de aciertos
- Registro de fallos
- Reproductor de música
- Implementación de base de datos SQLite para las pruebas
- Guardar datos recogidos de la rejilla en la base de datos
- Pantalla de creación rejilla wingdings
- Creación de rejillas: Rejilla wingdings
- Creación de rejillas: Rejilla anecedario
- Pantalla de configuración de distracciones
- Pantalla de selección de colores de fondo
- Exportar resultados de la prueba
- Límite de tiempo rejilla
- Emparejar
- Emparejar con tiempo
- Exportar ajustes
- Importar ajustes

- Importar plantillas
- Exportar plantillas

- Subir resultados
- Login
- Aleatorizar
- Conexión con WS
- Almacenamiento en MenPas

Requisitos no funcionales:

Estos describen los criterios que pueden usarse para juzgar el funcionamiento de un sistema, en lugar de comportamientos específicos. Se centran en la calidad del sistema y en cómo debe funcionar, en lugar de qué debe hacer. Los requisitos no funcionales responden a la pregunta "¿cómo debe funcionar el sistema?".

Los requisitos no funcionales del proyecto son:

- Compatible con dispositivos Android
- Interfaz accesible y fácil de usar para todo tipo de personas
- Navegación entre pantallas intuitiva para todo tipo de personas
- Instalación fácil para todo tipo de personas
- Tema de la aplicación
- Icono de la aplicación
- La aplicación debe cargar la pantalla de inicio en menos de 2 segundos
- Las rejillas deben generarse en menos de 2 segundos
- Los datos del usuario deben guardarse correctamente

5.5 Diagrama de secuencia

Un diagrama de secuencia es una herramienta muy práctica para modelar la interacción entre objetos en un sistema a lo largo del tiempo. Se utiliza para visualizar cómo un

conjunto de objetos en una aplicación se comunica y colabora para llevar a cabo una actividad.

Los diagramas de secuencia son útiles para detallar la implementación de escenarios específicos, proporcionando una visión clara de los objetos y clases involucrados y de los mensajes intercambiados a partir de las acciones del usuario o el software.
El diagrama de secuencia muestra a un nivel de detalle mucho mayor que otros diagramas la interacción entre elementos y la línea a seguir en una acción.
Cada objeto se representa con una línea de vida vertical discontinua, y los mensajes que se envían entre ellos son flechas horizontales. Este enfoque facilita la comprensión de cómo se desarrolla el flujo de control en el sistema y cómo los diferentes componentes colaboran entre sí.

Para crear un diagrama de secuencia, se puede partir de la descripción de un caso de uso, identificando los objetos necesarios para implementar el escenario descrito. Si la descripción del caso de uso está modelada como una secuencia de pasos, se puede seguir estos pasos para ver los objetos implicados. Este proceso ayuda a resumir y especificar los detalles de implementación del escenario, clarificando qué objetos interactúan en cada paso y cómo lo hacen.

En esta sección se mostrará el diagrama de secuencia en la Figura 13 para crear y completar una rejilla en la RejillaApp.

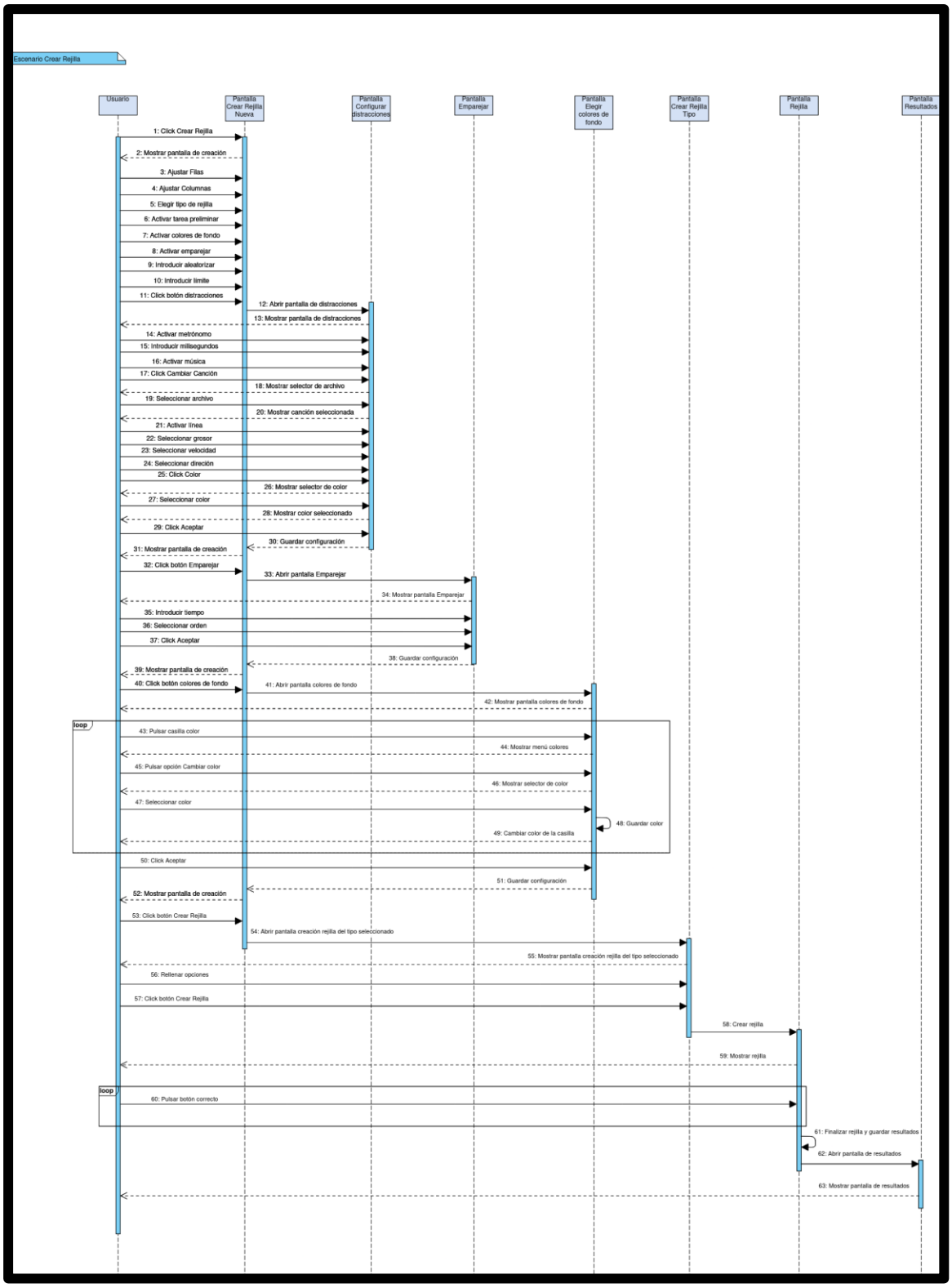


Figura 13. Diagrama de secuencia para crear rejilla y ver resultados

5.6 Diagrama de clases

Un diagrama de clases es una herramienta fundamental en el análisis y diseño de sistemas de software que permite representar la estructura estática de un sistema. El diagrama de clases ilustra las clases y objetos del sistema junto con sus relaciones estructurales y de herencia. Cada clase en el diagrama describe un objeto que comparte una estructura y comportamiento similares, y lo definen los atributos y los métodos que tiene.

Visualmente, una clase se representa como un rectángulo dividido en secciones horizontales, a menudo separadas por una línea.
Estas secciones son:

- **Sección superior:** Muestra el nombre de la clase y propiedades.
- **Sección intermedia:** Muestra los atributos.
- **Sección inferior:** Muestra los métodos.

Cada atributo y método está precedido por un indicador de visibilidad, el cual se especifica en el cuadro de diálogo que define a cada elemento.

El principal objetivo de un diagrama de clases es desglosar la estructura del sistema, proporcionando una forma visualmente sencilla y estructurada del funcionamiento interno del sistema. Esto resulta especialmente útil en la fase de análisis, para identificar y organizar los requisitos del sistema.

De acuerdo con la estructura del proyecto, basada en el uso de clases *Activity* de Android, se ha realizado un diagrama de clases con estas y las otras clases presentes, así como sus relaciones, mostrando que actividad enlaza con otra y que clases se usan en cada una.

A continuación, en las figuras de la 14 a la 17 se presenta el diagrama de clases del sistema completo.

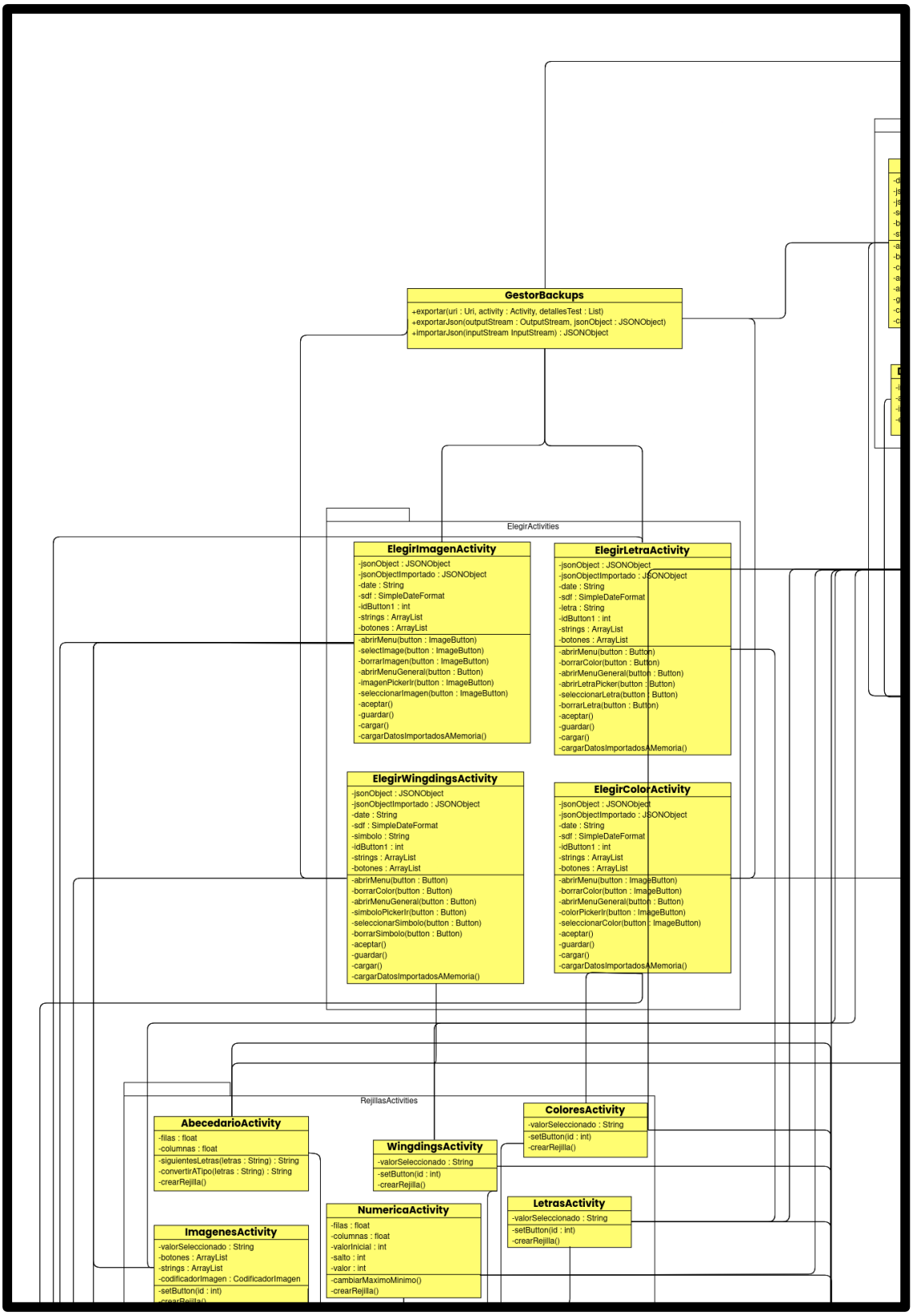


Figura 14. Diagrama de clases RejillaApp (esquina superior izquierda)

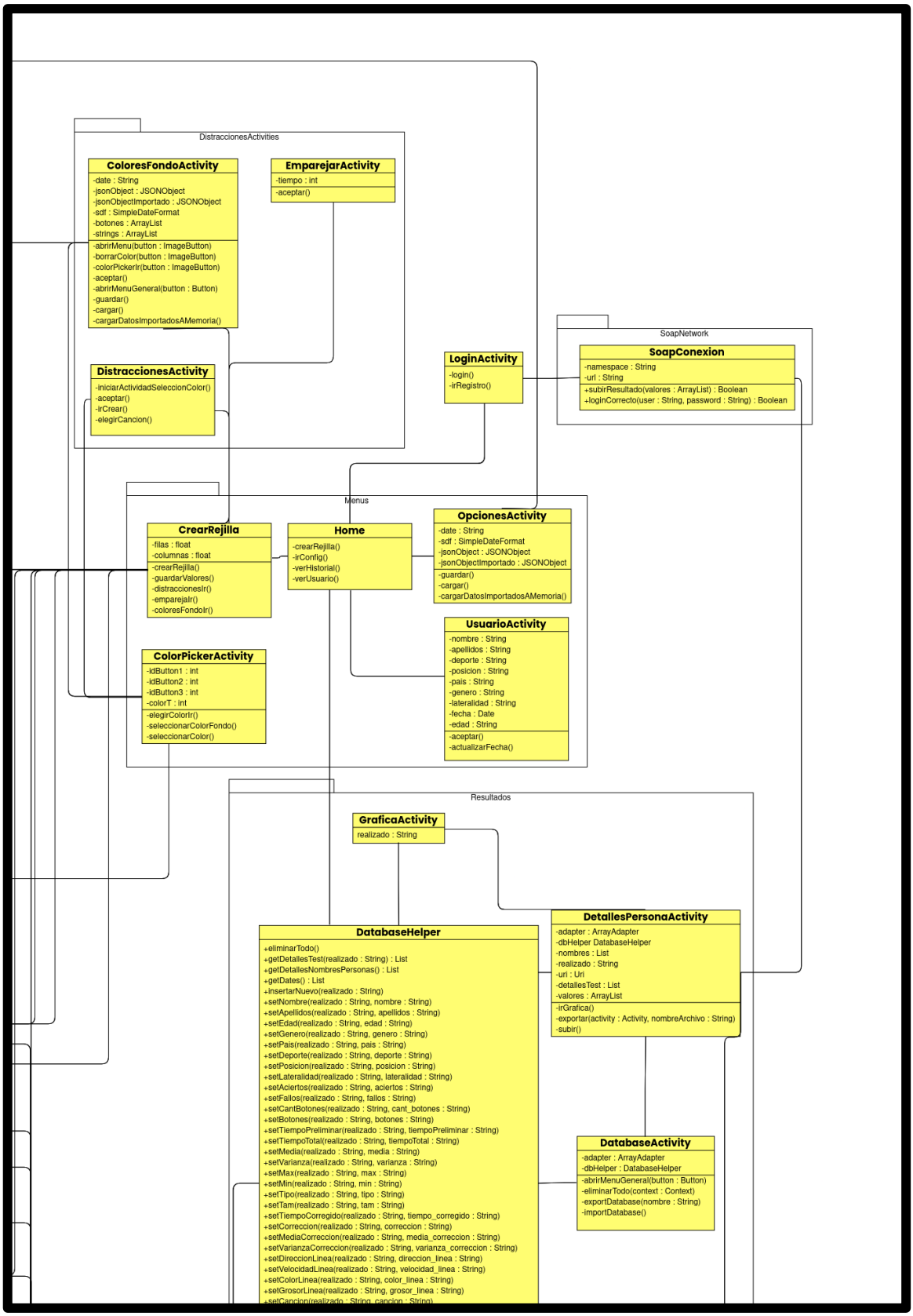


Figura 15. Diagrama de clases RejillaApp (esquina superior derecha)

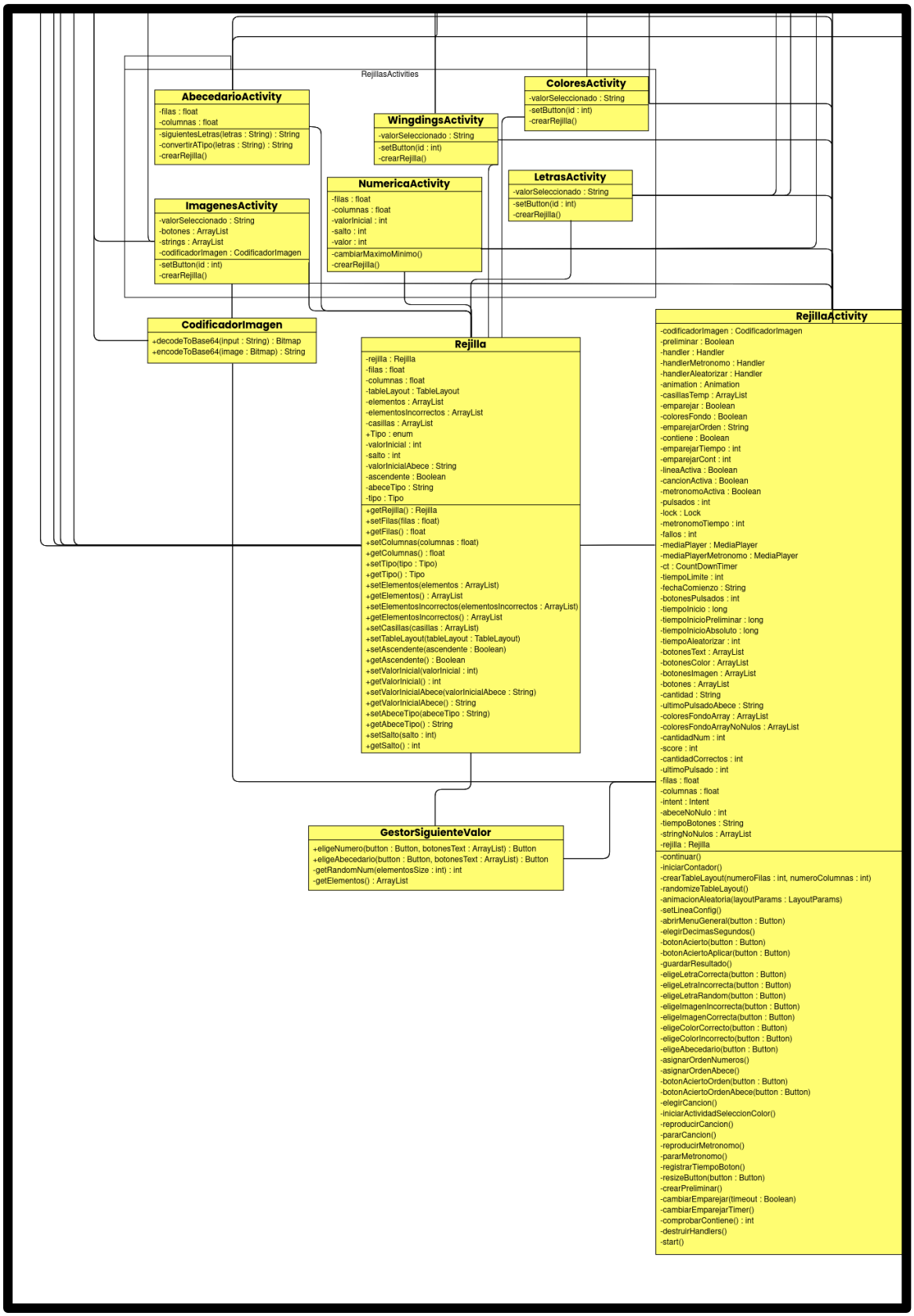


Figura 16. Diagrama de clases RejillaApp (esquina inferior izquierda)

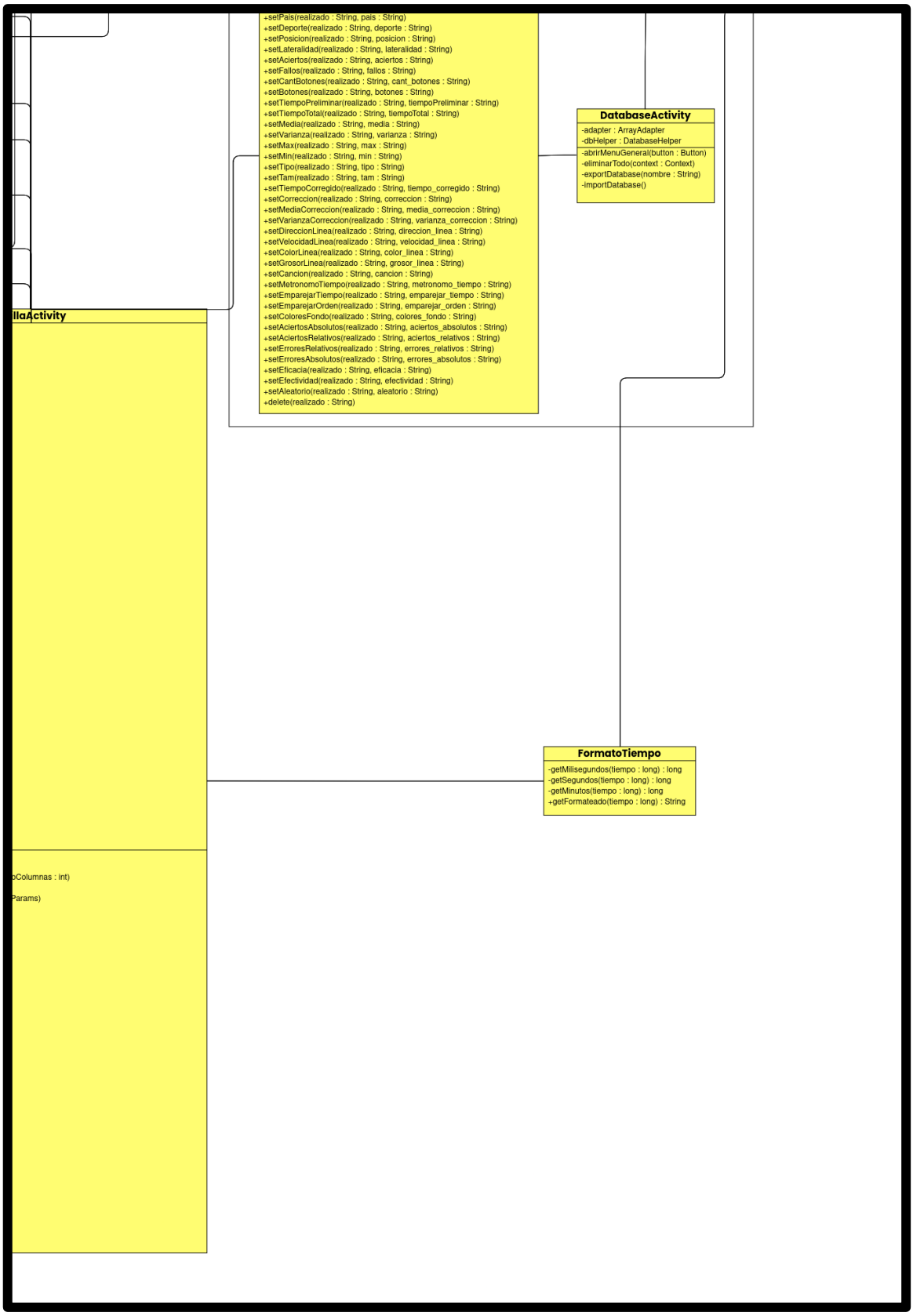


Figura 17. Diagrama de clases RejillaApp (esquina inferior derecha)

5.7 Primera iteración

Tras lo detallado anteriormente se procede a especificar el trabajo realizado en la primera iteración del proyecto.

Esta iteración está centrada especialmente en el front-end.

Al inicio del proyecto se planteó la realización de una maqueta o mockup.

En el desarrollo de software, una maqueta o mockup consiste en una representación estática o interactiva de la interfaz de usuario que aporta una visión aproximada con más o menos precisión (dependiendo del propósito de este) de cómo se verá el producto final. Esta maqueta incluye elementos gráficos, esquemas y todo lo necesario para tener una idea lo más precisa posible de la solución que se quiere crear. Esta puede incluir algunos elementos interactivos que redirijan unas partes del producto a otras para simular el funcionamiento final del mismo. [36] [37]

Tras considerar la opción de usar una maqueta al comienzo del desarrollo, se decidió realizar estas utilizando Android Studio, la decisión se tomó debido a diversos factores:

1) La magnitud del proyecto favorece la realización de una maqueta en Android Studio. El proyecto consiste en una seria de actividades Android estructuradas y organizadas de forma que sea lo más simple posible para el usuario, y esto repercute en una mayor simpleza de la interfaz que es fácilmente reproducible en la propia solución final en la aplicación.

2) La rapidez y agilidad requeridas en el modelo de desarrollo utilizado en el proyecto requiere la realización de cambios en poco tiempo atendiendo a lo que el cliente requiera, por esto se ha utilizado la herramienta de construcción de interfaces de Android Studio para realizar las primeras maquetas, siendo fácilmente manipulables y extensibles si se requiere. En caso de que el cliente acepte la maqueta, se permite empezar a trabajar en la implementación directamente utilizando la maqueta realizada.

3) Las capacidades de Android Studio para la realización de una primera interfaz de prueba son más que suficientes para poder tener preparada una maqueta muy precisa, adaptable, y funcional, dejando así ver al cliente con detalle lo que podría ser el producto final. Realizar la maqueta en la propia herramienta de desarrollo es la mejor forma de dar al cliente una impresión lo más cercana posible al producto final en un periodo de tiempo muy corto, ya que desarrollar esta maqueta es un proceso rápido y ágil con la herramienta de Android Studio, y utilizando la misma interfaz que se usará en el producto final, corriendo además en el dispositivo donde correrá el producto final, un teléfono Android.

Por todo lo explicado anteriormente, se decidió dedicar la primera iteración a la realización de la interfaz (front-end) y partes limitadas del back-end, fijando como objetivo el dejar al cliente ver la aplicación en funcionamiento, con funcionalidades limitadas, pero con una interfaz que, en caso de ser aceptada, sería ya definitiva.
Esta primera iteración, los requisitos que cubre, y los detalles de su desarrollo se detallan en las siguientes tablas:

Nombre	Iteración 1: 001 – Pantalla de inicio
Introducción	Desarrollo de la interfaz y funcionalidades básicas de la pantalla de inicio de la aplicación
Objetivo	Este requisito dotará la aplicación de una página de portada con las opciones iniciales de esta
Uso previsto	El requisito se usará cada vez que el usuario abra la aplicación y tras iniciar

	sesión, ya que esta página será la portada. También se usará para acceder al menú de opciones de importación y exportación, usuario e historial
Planteamiento de diseño	Se usa el tema principal de la aplicación, con los colores de la Universidad de Málaga, y con el icono de la aplicación en la parte superior y los botones en la parte inferior
Diseño de la solución	El diseño de la interfaz pretende ofrecer una forma vistosa y característica de presentar las opciones principales de la aplicación, siendo usado además como portada visual de la misma. Se ha optado por utilizar un fondo de color, usando como color el principal del tema de la aplicación, azul oscuro. Sobre este fondo se presentan los botones con letras blancas e iconos representativos de la funcionalidad que realizan, a excepción del botón de configuración, que por motivos estéticos y prácticos se presenta como un único icono de una llave inglesa en la esquina inferior derecha de la pantalla. A esto se añade el icono de la aplicación en la parte superior, y el nombre de la misma en la barra superior en letra blanca.

	Cada uno de los botones enlaza a la actividad correspondiente a través de clases de tipo *Intent* y el método *StartActivity* **Botones**: • **Crear Rejilla:** Enlaza a la página de crear rejilla • **Historial:** Enlaza a la lista de pruebas realizadas • **Usuario:** Enlaza al formulario de información personal del usuario • **Configuración:** Enlaza a la página de configuraciones

Nombre	Iteración 1: 002 – Tema de la aplicación
Introducción	Decisión de tema de la aplicación, colores principales y tamaños de letra y botones
Objetivo	Estandarizar todas las pantallas de la aplicación para facilitar el entendimiento de esta y mejorar la estética
Uso previsto	Se usará este estándar en todas las pantallas de la aplicación y en los menús
Planteamiento de diseño	Tras un estudio del funcionamiento de las interfaces de usuario en Android y las prácticas más extendidas en la industria,

	teniendo en cuenta además la caracterización de la Universidad de Málaga, se decide usar un tema azul oscuro en los botones, la página de inicio y la barra superior, acompañado de fondos blancos y letras negras en el fondo o blancas en los botones
Diseño de la solución	Se utiliza el tema claro de Android, usando botones y letras con fondo oscuro o claro dependiendo de la posición. Los formularios son sobrios y simples para facilitar el enfoque en la información que hay que introducir. Los colores usados, los tamaños y las posiciones vienen detallados a continuación: **Colores**: • **Color principal:** Azul oscuro (Universidad de Málaga **#002751** ■) • **Color secundario:** Blanco • **Color de letra:** Blanco en portada y botones. Negro en formularios y menús con fondo blanco

	Tamaños: • **Estándar letra botones:** 34dp sin escalado (la configuración de tamaño de letra del dispositivo no influirá en el tamaño de letra de la aplicación) • **Estándar formularios:** 24dp sin escalado (la configuración de tamaño de letra del dispositivo no influirá en el tamaño de letra de la aplicación) • **Menús spinner:** 24dp sin escalado (la configuración de tamaño de letra del dispositivo no influirá en el tamaño de letra de la aplicación) • **Letra formulario Usuario:** 20dp sin escalado (la configuración de tamaño de letra del dispositivo no influirá en el tamaño de letra de la aplicación) • **Botón principal:** 0x77dp sin escalado (la configuración de tamaño de letra del dispositivo no influirá en el tamaño de letra de la aplicación)

	• **Botones secundarios:** 345x54dp sin escalado (la configuración de tamaño de letra del dispositivo no influirá en el tamaño de letra de la aplicación) • **Barra superior:** 0x56dp sin escalado (la configuración de tamaño de letra del dispositivo no influirá en el tamaño de letra de la aplicación) **Posiciones:** • **Botón principal de la pantalla:** Parte inferior de la pantalla y extendido hacia los laterales • **Botones secundarios:** Inmediatamente encima del botón principal, con una longitud de 345dp y a una distancia del botón principal de 8dp. Cada botón se sitúa a 8dp del inmediatamente inferior a él • **Formularios:** Se sitúa el título del formulario a la izquierda y el editable a la derecha con una separación. A excepción del caso

	de no requerir título como en el formulario de Usuario (donde se resuelve todo con *hints*) Toda la aplicación ha sido diseñada dentro de lo posible teniendo en cuenta estas normas de estilo para dotarla de coherencia visual

Nombre	Iteración 1: 003 – Icono de la aplicación
Introducción	Diseño e implementación del icono de la aplicación
Objetivo	Crear un icono para la aplicación que vaya acorde al diseño de la misma y a las necesidades del cliente
Uso previsto	Icono de la aplicación que se mostrará en el menú de aplicaciones de Android y

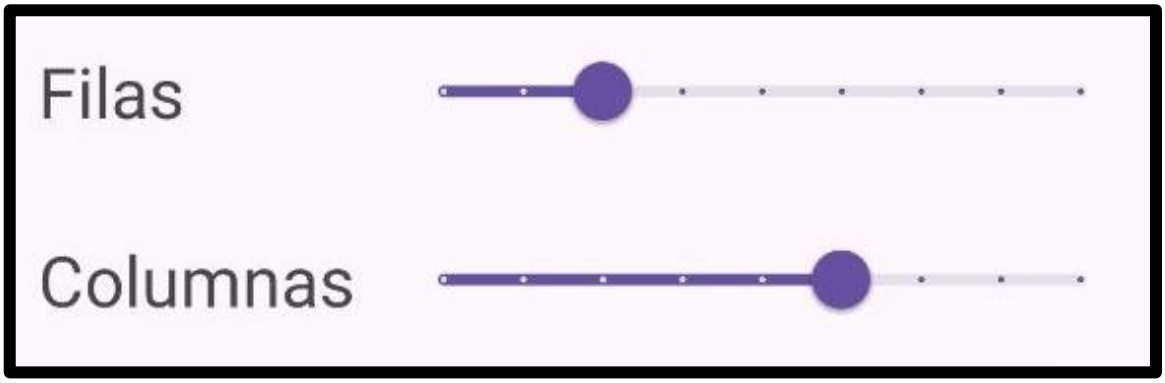
	al iniciar la aplicación en la portada y la pantalla de inicio de sesión
Planteamiento de diseño	Se ofreció un diseño principal minimalista al cliente, y tras algunos reajustes basados en un icono aportado por el cliente, se llegó a un término medio entre un icono que encaje con el diseño de la aplicación y que a la vez cumpla los requisitos estéticos del cliente
Diseño de la solución	Los requisitos del icono final deben ser dos: • Que encaje con el tema principal de la aplicación y la Universidad de Málaga • Que se parezca al icono aportado por el cliente Tras ofrecer varias versiones, el cliente decidió elegir el icono actual por cumplir correctamente con los requisitos y ser estéticamente apropiado El diseño del icono aportado usaba amarillo como fondo, con letras pixeladas en negro y una sección superior con varios pixeles de colores a modo de representación de una rejilla de colores de la aplicación.

	A este diseño se le hizo un rediseño cogiendo los conceptos clave y simplificándolo para encajar en la estética general de un dispositivo móvil, con tendencia a la simplicidad y los bordes redondeados. • Se mantuvo el fondo amarillo para las letras, añadiendo un fondo azul oscuro alrededor para el redondeado del icono • Se conservó la letra pixelada • Se conservó el concepto de pixeles de colores que representen la rejilla • Se conservaron algunos colores usados Se creó un cuadrado con los pixeles de colores dentro en fondo gris, y el nombre de la aplicación al lado derecho, rodeado por amarillo y fondo azul

Nombre	Iteración 1: 004 - Menú de creación de rejilla
Introducción	Desarrollo de la interfaz y funcionalidades básicas del menú de creación de rejillas
Objetivo	Este requisito dará la habilidad de crear rejillas con diversos parámetros al usuario. En la primera iteración se implementa el selector de filas y

	columnas y el botón de crear, así como la capacidad de elegir la rejilla, de las cuales solo se implementa la de letras.
Uso previsto	El requisito se usará cada vez que el usuario entre a la aplicación con el fin de crear una rejilla nueva para probar su capacidad de atención
Planteamiento de diseño	La interfaz planteada es simple, intuitiva y vistosa. Se presentan *drop lists, switchers* y *spinners* para seleccionar las opciones, y botones para detallar las mismas o crear la rejilla
Diseño de la solución	Para ofrecer una interfaz simple e intuitiva a la vez que, con diversas opciones, se agrupan en un *drop list* la selección de tipos de rejilla, y el resto de las opciones se organizan en *spinners* para filas y columnas, *switchers* para las opciones activables, y botones de acceso a menús para configuraciones extra, así como *EditText* para los valores numéricos. El tamaño de los botones y las letras es adaptable a la pantalla y se ha usado un diseño tipo “lista” para opciones y botones, presentando una bajo la otra de forma ordenada y estructurada. Esta y las otras pantallas de la aplicación presentan una barra superior con el

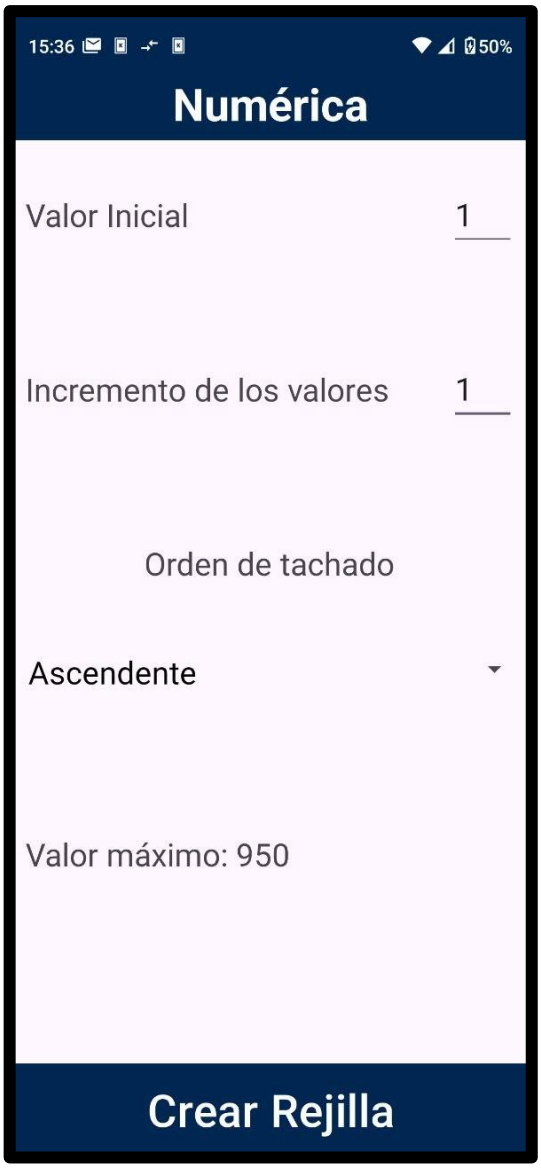
	tema de la aplicación y el nombre de la pantalla actual. Este tema de color también se usa en los botones. Los valores de cada elemento del menú se guardan y cargan de la configuración de la aplicación con un *SharedPreferences*



Nombre	Iteración 1: 005 - Seleccionar filas y columnas
Introducción	Selectores de filas y columnas en la pantalla de creación de rejilla
Objetivo	Ofrecer una forma simple y precisa de elegir la cantidad de filas y columnas de la rejilla
Uso previsto	Cada vez que se quiera crear una rejilla, a excepción de si se dejan los parámetros por defecto
Planteamiento de diseño	Se usan *sliders* con valores máximos y mínimos acoplados a un texto con el

	nombre del parámetro (Filas o Columnas)
Diseño de la solución	Se utilizan dos *sliders* con rango de 2 a 10, siendo el valor por defecto el 7, que se usará si no se selecciona otro. Estos *sliders* van acoplados a dos *TextView,* Filas o Columnas dependiendo del *slider.* Al seleccionar un valor, si se mantiene el dedo en el *slider,* se puede ver el valor exacto que se está seleccionando en una pequeña burbuja que aparece en la parte superior del selector

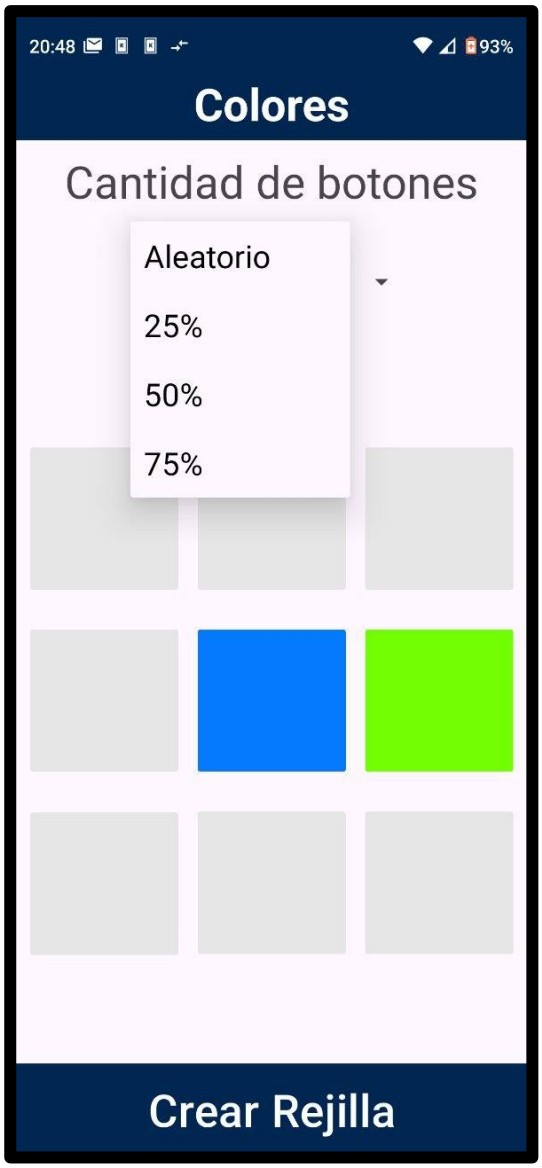
Nombre	Iteración 1: 006 - Seleccionar tipo de rejilla
Introducción	*Dropdown list* con los tipos de rejilla de la aplicación
Objetivo	Dar la lista de opciones al crear una rejilla para que el usuario seleccione el tipo de la misma
Uso previsto	Al crear una rejilla, en la pantalla de creación, se usa esta lista para seleccionar el tipo de la rejilla
Planteamiento de diseño	Se utiliza una *dropdown list* de tipo *spinner* con 6 opciones, la opción seleccionada se utilizará para elegir la pantalla de creación siguiente
Diseño de la solución	Se recoge el valor seleccionado en el *dropwdown list* y al pulsar el botón de crear rejilla, se convierte este valor a texto y se compara con un *switch* con los distintos tipos de rejilla que hay. Dependiendo de cuál sea, se iniciará una nueva actividad con las configuraciones particulares del tipo de rejilla seleccionado



Nombre	Iteración 1: 007 – Pantalla de creación rejilla numérica (front-end)
Introducción	Interfaz de creación de la rejilla numérica con sus opciones
Objetivo	Ofrecer una forma sencilla y rápida de introducir los valores necesarios para crear la rejilla numérica
Uso previsto	Se desplegará esta pantalla cada vez que se seleccione la rejilla numérica en el

	menú de creación de rejilla y se pulse el botón de crear rejilla
Planteamiento de diseño	Se ofrecen las opciones: • **Valor inicial** • **Incremento de los valores** • **Orden de tachado**
Diseño de la solución	La pantalla de creación se estructura en tres partes, cada una con una de las siguientes opciones: • **Valor inicial:** Un *EditText* que solo acepta valores numéricos superiores o iguales a 0. Se usará como primer valor de la rejilla numérica. Se utiliza para comprobar si se puede crear la rejilla antes de hacerlo. • **Incremento de los valores:** Un *EditText* que acepta valores numéricos superiores a 0 • **Orden de tachado:** *Spinner* que da la opción de decidir si el siguiente número será mayor o menor en base al incremento • **Valor máximo/Valor mínimo:** Se muestra en un texto cual es el

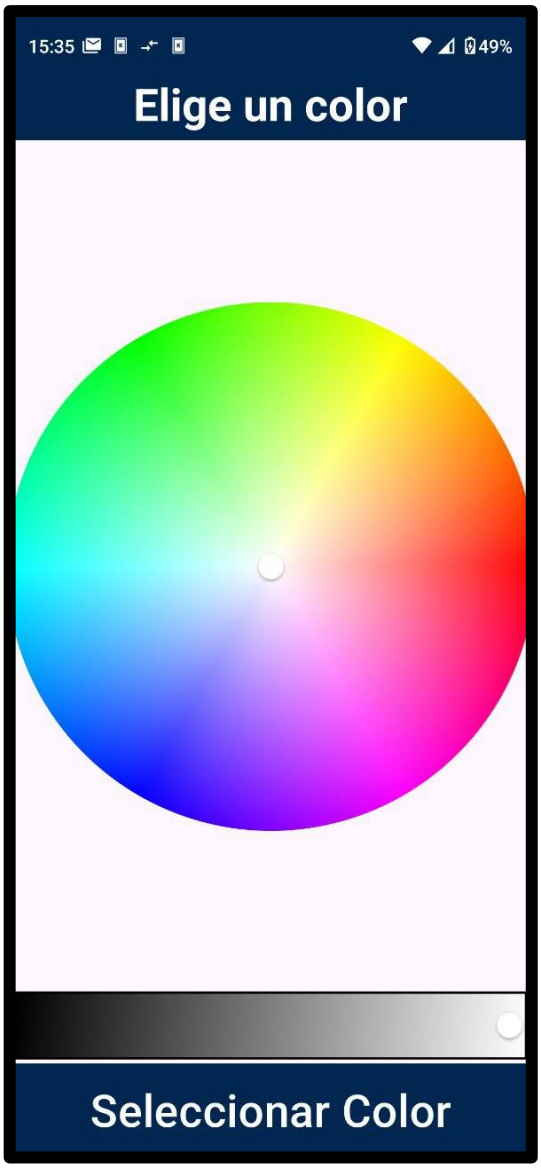
	valor máximo (si el orden es ascendente) o mínimo (si el orden es descendente) en base a los valores introducidos. También se notifica si hay algún error o si se superan los límites. Este formulario está oculto hasta que se introduzca un valor inicial e incremento Si alguna condición no se cumple se notifica al usuario y no se crea la rejilla hasta que lo solucione. En la parte inferior de la pantalla está el botón de creación de rejilla Si todo está en orden, tras pulsar el botón de creación de rejilla, se muestra la rejilla creada y empieza el test



Nombre	Iteración 1: 008 – Pantalla de creación rejilla colores (front-end)
Introducción	Interfaz de creación de la rejilla colores con sus opciones
Objetivo	Ofrecer una forma sencilla y rápida de introducir los valores necesarios para crear la rejilla colores
Uso previsto	Se desplegará esta pantalla cada vez que se seleccione la rejilla colores en el menú

	de creación de rejilla y se pulse el botón de crear rejilla
Planteamiento de diseño	Se ofrecen nueve botones cuadrados a rellenar con el color elegido y un selector para la cantidad de botones correctos
Diseño de la solución	La pantalla consta de nueve botones cuadrados y una *dropdown list spinner* para seleccionar la cantidad de botones correctos que se añadirán a la rejilla, a seleccionar de los disponibles elegidos por el usuario. Se ofrecen las siguientes opciones: • **Cantidad de botones:** *TextView* seguido de una *dropdown list spinner* con cuatro opciones: o Aleatorio o 25% o 50% o 75% El valor por defecto es Aleatorio. Esta cantidad representa cuantas de las casillas de la rejilla serán correctas y serán las que el usuario deba pulsar. • **Cuadrados de colores:** Nueve botones cuadrados. Al pulsar cada uno de estos botones se abrirá la pantalla de selección de colores disponibles, y cuando se

	seleccione uno este cuadrado pasará a ser del color seleccionado En la parte inferior de la pantalla está el botón de creación de rejilla Si todo está en orden, tras pulsar el botón de creación de rejilla, se muestra la rejilla creada y empieza el test



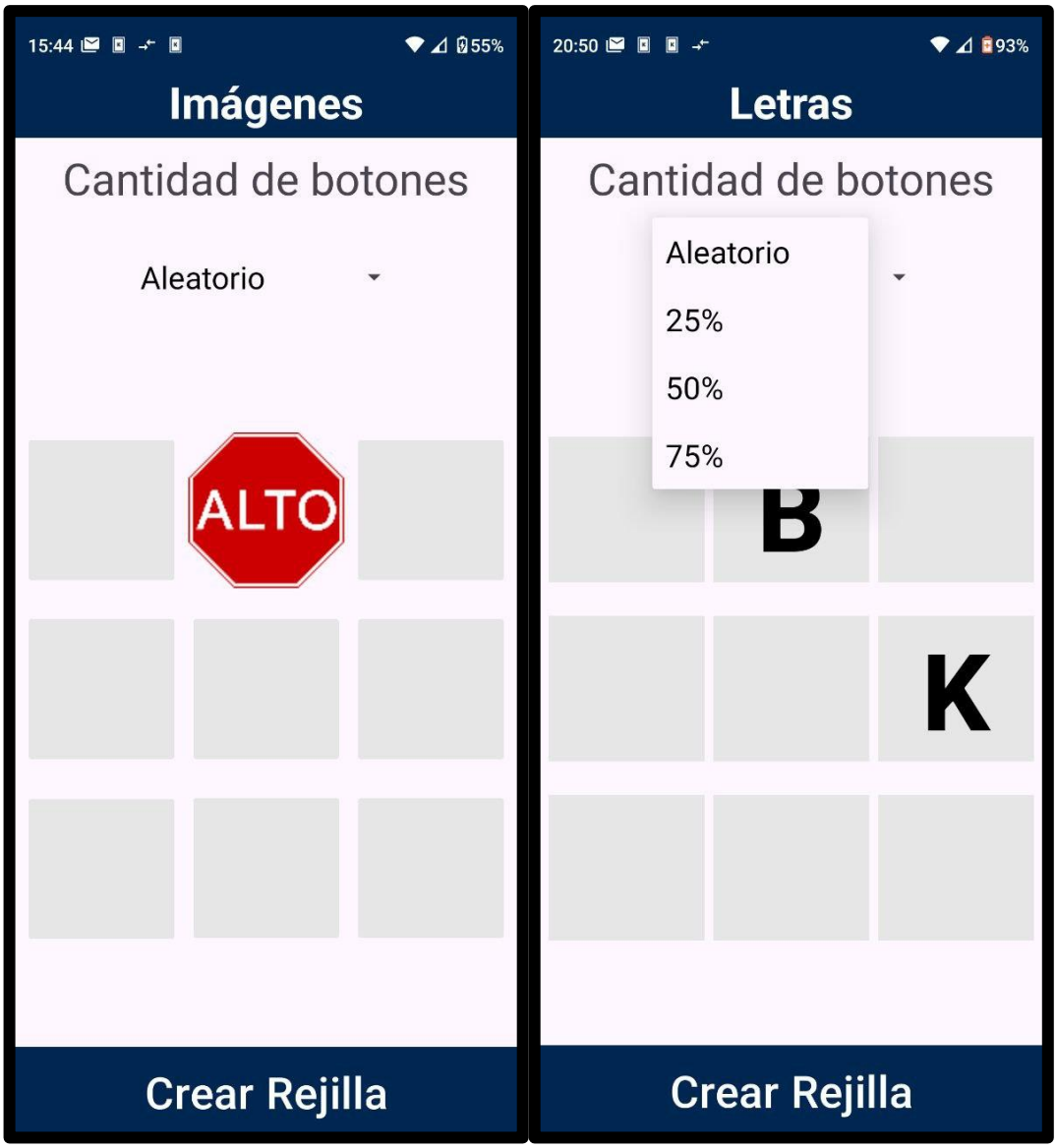
Nombre	Iteración 1: 009 – Selección de colores *ColorPicker* (front-end)
Introducción	Pantalla de selección de color diseñada para permitir al usuario elegir un color para diversas funcionalidades de la aplicación
Objetivo	El propósito de este requisito es proporcionar al usuario una interfaz intuitiva y fácil de usar para seleccionar un color

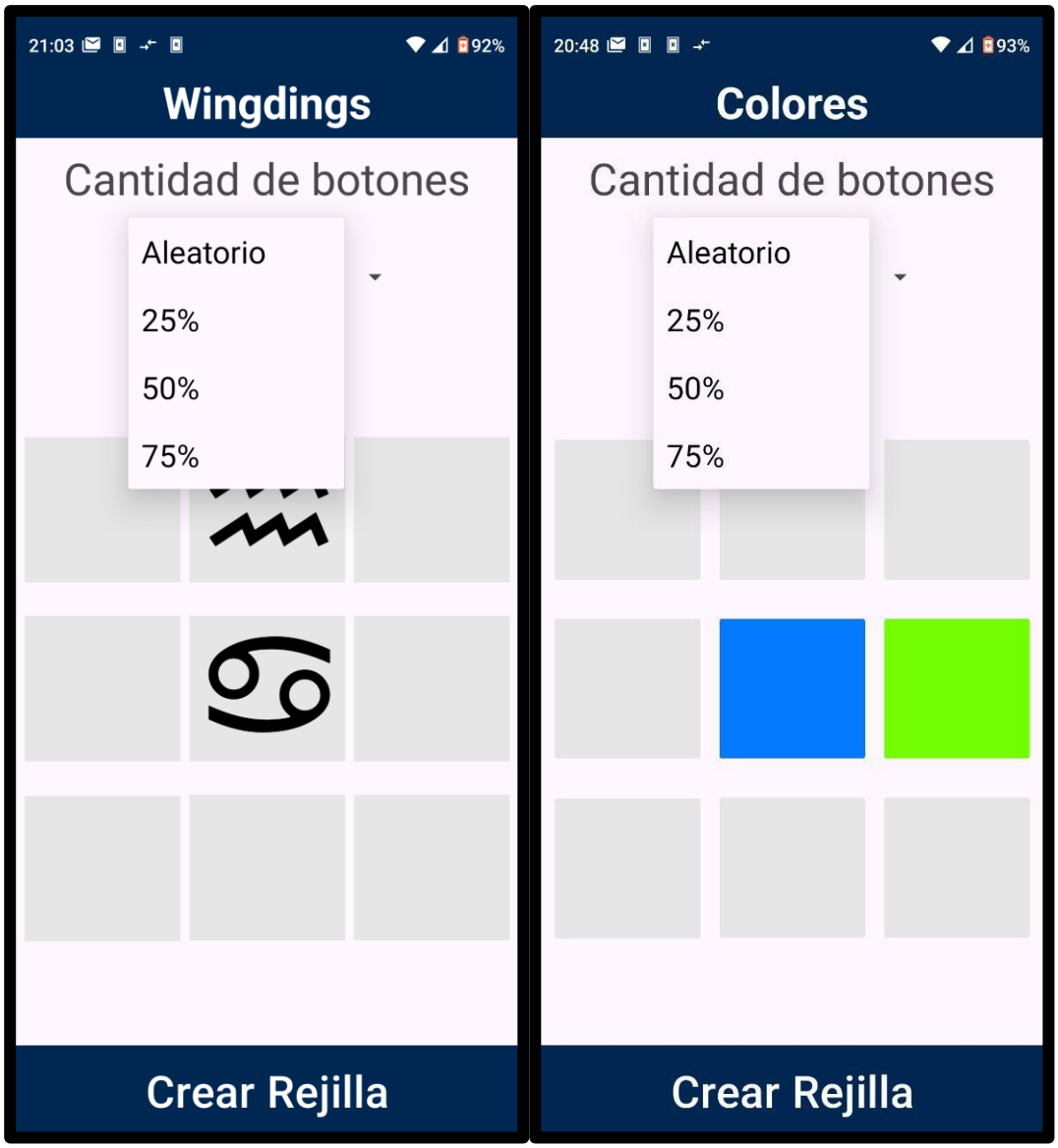
Uso previsto	Este requisito será utilizado cuando el usuario pulse el botón de cambiar color en los siguientes menús: • **Elegir color:** Para elegir los colores de la rejilla de colores • **Colores de fondo:** Para elegir los colores de fondo de las casillas de la rejilla si se activa la función de colores de fondo • **Elegir color de línea de distracción:** Para elegir el color de la línea de distracción desde el menú rápido en la rejilla o desde el menú de configuración de distracciones
Planteamiento de diseño	La pantalla consta de un círculo de colores y un panel inferior para elegir el brillo del color elegido en el círculo
Diseño de la solución	La pantalla de selección de colores de fondo consta de un círculo central que contiene los colores en formato RGB. Pulsando este círculo y desplazando el cursor selector, se va cambiando el color del panel inferior. Justo debajo del círculo, hay un rectángulo que sirve como control deslizante para ajustar el

	brillo del color, permitiendo al usuario variar desde un tono oscuro hasta un tono claro. Además, se incluirá un botón "Aceptar" para confirmar la selección del color y cerrar la pantalla. Para este requisito se ha utilizado el proyecto Github: https://github.com/skydoves/ColorPickerView

Nombre	Iteración 1: 010 – Selección de colores *ColorPicker* (back-end)
Introducción	Funcionamiento de la pantalla de selección de colores *ColorPicker*
Objetivo	Este requisito consiste en la implementación de la funcionalidad de esta pantalla
Uso previsto	Este requisito será utilizado cuando el usuario haga algo en esta pantalla
Planteamiento de diseño	Se gestionan los accesos a esta pantalla dependiendo del menú desde el cual se acceda, para establecer el color requerido por la pantalla anterior y devolver el color seleccionado si es necesario
Diseño de la solución	Pulsando sobre el círculo y desplazando el cursor selector, se va recogiendo el color con el método *setColorListener* presente en el paquete del proyecto. Tras esto se actualiza una variable global que contiene el color.

	Uniendo el selector de brillo al selector de color con el método *attachBrightnessSlider* se reflejará el color seleccionado en la barra inferior de la pantalla, que también sirve de selector de brillo para ajustar el color. Dependiendo de lo que se requiera, al pulsar el botón seleccionar color esta pantalla devolverá como parámetros el color seleccionado y el id del botón que se pulsó para seleccionar un color Para este requisito se ha utilizado el proyecto Github: https://github.com/skydoves/ColorPickerView

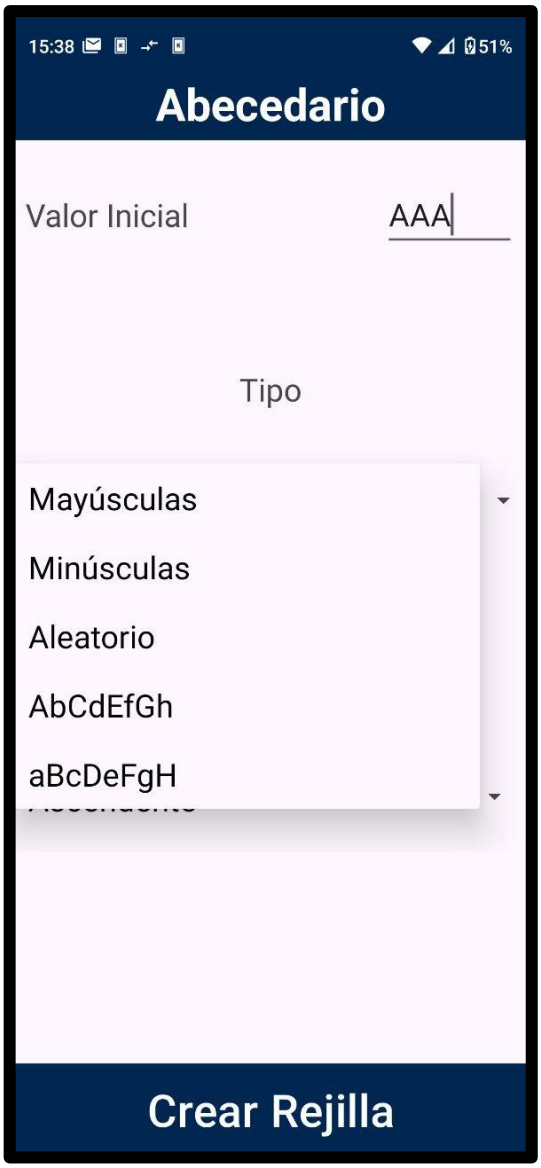

15:44
55%
Imágenes
Cantidad de botones
Aleatorio
ALTO
Crear Rejilla
20:50
93%
Letras
Cantidad de botones
Aleatorio
25%
50%
75%
B
K
Crear Rejilla



Nombre	Iteración 1: 011 – Pantalla de creación rejilla de elementos (front-end)
Introducción	Interfaz de creación de las rejillas de elementos: rejilla de letras, rejilla de colores, rejilla de imágenes, rejilla de símbolos. Estas son las rejillas que requieren para su creación de la selección de los elementos correctos e incorrectos. Esta

	pantalla permite esa selección y otras opciones
Objetivo	Ofrecer una forma sencilla y rápida de introducir los valores necesarios para crear las rejillas de elementos
Uso previsto	Se desplegará esta pantalla cada vez que se seleccione una de las rejillas de elementos en el menú de creación de rejilla y se pulse el botón de crear rejilla
Planteamiento de diseño	Se ofrecen nueve botones cuadrados a rellenar con el elemento elegido y un selector para la cantidad de botones correctos
Diseño de la solución	La pantalla consta de nueve botones cuadrados y una *dropdown list spinner* para seleccionar la cantidad de botones correctos que se añadirán a la rejilla, a seleccionar de los disponibles elegidos por el usuario. Se ofrecen las siguientes opciones: • **Cantidad de botones:** *TextView* seguido de una *dropdown list spinner* con cuatro opciones: o Aleatorio o 25% o 50% o 75% El valor por defecto es Aleatorio. Esta cantidad representa cuantas de las casillas de la rejilla serán

	correctas y serán las que el usuario deba pulsar. • **Cuadrados grises con elementos:** Nueve botones cuadrados. Al pulsar cada uno de estos botones se abrirá la pantalla de selección de elementos correspondiente, y cuando se seleccione una, este cuadrado pasará a tener dentro el elemento seleccionado. Si no hay ninguno seleccionado estará vacío En la parte inferior de la pantalla está el botón de creación de rejilla Si todo está en orden, tras pulsar el botón de creación de rejilla, se muestra la rejilla creada y empieza el test



Nombre	Iteración 1: 012 – Pantalla de creación rejilla abecedario (front-end)
Introducción	Interfaz de creación de la rejilla abecedario con sus opciones
Objetivo	Ofrecer una forma sencilla y rápida de introducir los valores necesarios para crear la rejilla abecedario
Uso previsto	Se desplegará esta pantalla cada vez que se seleccione la rejilla abecedario en el

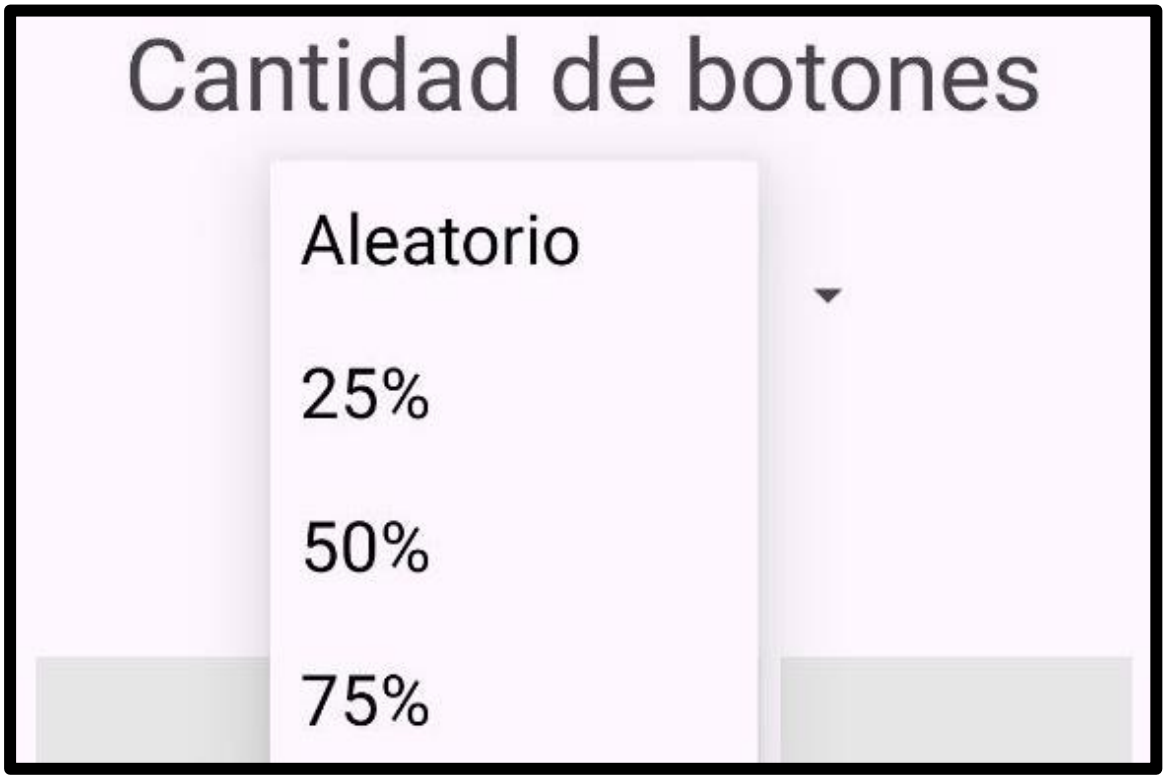
	menú de creación de rejilla y se pulse el botón de crear rejilla
Planteamiento de diseño	Se ofrecen las opciones: • Valor inicial • Tipo de letras • Orden de tachado
Diseño de la solución	La pantalla de creación se estructura en tres partes, cada una con una de las siguientes opciones: • **Valor inicial:** Un *EditText* que solo acepta tres letras, las cuales convierte a mayúscula al introducirlas con un filtro. Se usará como primer valor de la rejilla abecedario. • **Tipo de letras:** *TextView* seguido de una *dropdown list spinner* con Cinco opciones: o Mayúsculas o Minúsculas o Aleatorio o AbCdEfGh o aBcDeFgH Estas determinan el formato de las tres letras introducidas. El valor por defecto es Mayúsculas.

	• **Orden de tachado:** *Spinner* que da la opción de decidir si las siguientes tres letras serán posteriores o anteriores alfabéticamente a las anteriores En la parte inferior de la pantalla está el botón de creación de rejilla Si todo está en orden, tras pulsar el botón de creación de rejilla, se muestra la rejilla creada y empieza el test

Nombre	Iteración 1: 013 – Pantalla de creación rejilla de elementos (back-end)
Introducción	Funcionalidad del back-end de la pantalla de creación de rejillas de elementos, con los enlaces correspondientes al pulsar cada botón
Objetivo	Dotar de funcionalidad la pantalla de creación de rejilla de elementos y cargar los elementos seleccionados en esta para crear la rejilla
Uso previsto	Se usará para crear las rejillas de elementos en base a los parámetros introducidos. Se accederá a esta pantalla tras pulsar crear rejilla en el menú de creación y si se seleccionó una rejilla de letras, imágenes, colores o símbolos

Planteamiento de diseño	Se usan botones vacíos que se rellenan dependiendo de los elementos seleccionados. El diseño consiste en botones que, al ser pulsados, crean un *Intent* en el que se decide el elemento a fijar en el botón pulsado, pasando a este el id del botón en cuestión
Diseño de la solución	Cada botón cuadrado pulsado envía su id a una pantalla de selección del elemento que se elija, y al volver a cargar la página de creación de rejilla de elementos una vez elegido el elemento a seleccionar, se pondrá este en el cuadrado elegido. Para esto se utiliza *SharedPreferences,* donde se guarda cada uno de los nueve elementos elegidos en el formato correspondiente, que se cargan al elegir uno nuevo. Si se acaba de entrar en el menú para crear una rejilla nueva, se vacían todos los cuadrados a la espera de que el usuario elija. Esto se hace poniendo a nulo el valor de cada uno en el *SharedPreferences.* Cuando se pulsa un cuadrado, este lleva al método *setButton,* que configura una

	actividad nueva con el id del botón pulsado y abre la pantalla de selección de elementos. Cuando se pulsa el botón de creación de rejilla se recoge el contenido de los cuadrados, y si alguno de ellos contiene un elemento, se procede a crear la rejilla, creando un nuevo *Intent* al que se le añade el valor de la cantidad de botones seleccionada en el *spinner* de cantidad de botones. La lista de elementos correctos e incorrectos se almacenan por separado en la clase única Rejilla



Nombre	Iteración 1: 014 – Seleccionar cantidad de botones rejilla
Introducción	Menú de selección de la cantidad de botones correctos en una rejilla en porcentaje
Objetivo	Seleccionar la cantidad de botones correctos que aparecerán
Uso previsto	Se usa en el menú de creación de rejillas de las rejillas de imágenes, colores, letras y símbolos
Planteamiento de diseño	Consiste en un *dropdown list* con cuatro elementos
Diseño de la solución	Se utiliza un *dropdown list con* cuatro elementos que utiliza un *spinner*. Al seleccionar el elemento y pulsar el botón de creación de rejilla se recoge este dato y se usa en la creación de la rejilla

Nombre	Iteración 1: 015 – Creación de rejillas: Estructura general
Introducción	Creación de la estructura general para creación de rejillas y compleción de la tabla que será usada para mostrar las casillas de la rejilla
Objetivo	Establecer las bases del código que se usará como cimiento sobre el que construir las funciones que se usen para crear cada tipo de rejilla
Uso previsto	Se usa cada vez que se entra a una rejilla

Planteamiento de diseño	Al iniciar la rejilla se hacen una serie de comprobaciones para activar o no las distracciones y otras funciones. Tras esto, se construye la rejilla correspondiente basándose principalmente en una función que itera sobre las casillas de la rejilla y configura cada una correctamente según corresponda
Diseño de la solución	• **Comprobaciones iniciales:** Se comprueban diferentes elementos utilizando *SharedPreferences* y cargando desde la configuración de la aplicación los valores. Estos valores son los siguientes: ○ **Cantidad de botones correctos:** Se comprueba si se ha seleccionado una cantidad de botones correctos en las rejillas correspondientes con esta opción. Para esto se carga el valor del *intent* con el correspondiente nombre de la variable. Tras cargarlo se multiplica por las filas y columnas para obtener la cantidad de casillas correctas, ya que

	este número es una proporción ○ **Tarea preliminar:** Se comprueba si se ha activado la opción de tarea preliminar para crear esta antes de la rejilla. Para esto se carga el valor con un *SharedPreferences* ○ **Colores de fondo:** Si se activó la opción, se comprobará y se creará una lista con todos los colores a usar, que se pondrán en el fondo de cada casilla aleatoriamente. Para esto se carga el valor con un *SharedPreferences* ○ **Emparejar:** En caso de haber activado esta opción, se crea una lista con todos los elementos correctos de la rejilla en el orden especificado por la configuración de emparejar. Estos valores

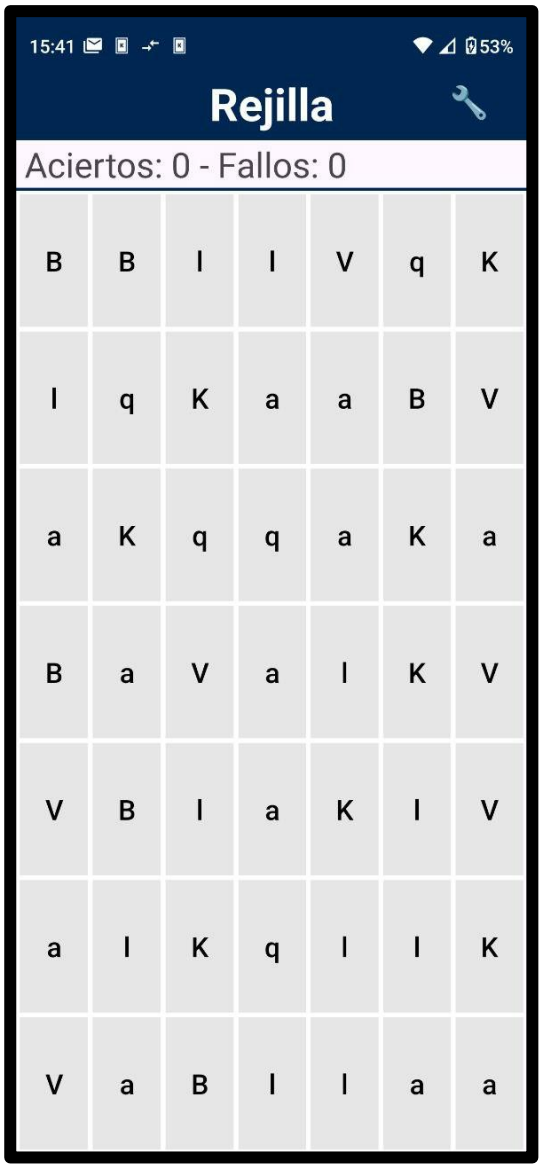
	se cargan con un *SharedPreferences* ○ **Emparejar con tiempo:** Si se ha seleccionado un tiempo para la función emparejar, este se usará, activando el correspondiente *handler* y comenzando el *timer* ○ **Distracción de línea:** En caso de estar activada, se cargan sus configuraciones con un *SharedPreferences* y se activa la línea en pantalla ○ **Distracción de metrónomo:** En caso de estar activado, se carga el tiempo con un *SharedPreferences* y se activa el metrónomo ○ **Distracción de canción:** En caso de estar activada y haber seleccionado una canción, esta se reproducirá. Para

	comprobar si se activó y cargar la canción se lee el *SharedPreferences* ○ **Tiempo límite:** Si se seleccionó un tiempo límite para la rejilla en la pantalla de creación, se empezará el *timer* y se mostrará en pantalla. Este tiempo se obtiene de la configuración con un *SharedPreferences* ○ **Aleatorizar:** Si se seleccionó un tiempo de aleatorización para la rejilla en la pantalla de creación, se empezará el *timer* y se aleatorizarán las casillas de la rejilla con un intervalo dependiente del tiempo seleccionado. Este tiempo se obtiene de la configuración con un *SharedPreferences* • **Crear la rejilla:** Para crear la rejilla se utiliza una función llamada *crearTableLayout*, que es el eje

	principal de todo el sistema que maneja la creación de las rejillas de todos los tipos. Esta función es llamada tras todas las comprobaciones mencionadas anteriormente, y consiste en un bucle *for* que comprueba una serie de parámetros y rellena una a una las casillas de un *TableLayout* con los botones correspondientes a la rejilla que se quiere crear, dotándolos del aspecto y la funcionalidad correcta, tanto si son correctos, en cuyo caso se recoge el acierto, como si son incorrectos, recogiéndose el fallo. En concreto, dentro de este bucle se realizan las siguientes acciones: ○ En caso de ser una rejilla de elementos, es decir: imágenes, letras, wingdings o colores, se cargan los elementos en un array de forma aleatoria para desordenar la rejilla ○ En caso de ser una rejilla incremental, es decir:

	numérica o de abecedario, esto no se hace, ya que su creación no lo necesita ○ Se crea el botón y se configura con el aspecto apropiado. Se configura el tamaño, la posición, el color de fondo... ○ En caso de tener que usar colores de fondo, en esta sección se establece el color para cada casilla ○ Se comprueba el tipo de rejilla y se llama a la función correspondiente para cada una que dotará al botón del contenido y la acción apropiada ○ Se añaden los botones a filas que se añadirán al *TableLayout* ○ Se crean diversos *ArrayList* para ciertos tipos de rejilla que los necesitan en caso

	de ser creadas rejillas de este tipo ○ Al acabar esto, se añade a la vista del *TableLayout* las filas que contienen cada uno de los botones Para almacenar diversos datos de la rejilla a crear y acceder a estos se ha usado el patrón *Singleton*, teniendo un único objeto de la clase *Rejilla* en toda la aplicación, que contiene los elementos de la misma, y que se actualizan cuando esta se crea. La configuración de botones depende de cada rejilla, así como el funcionamiento de la función emparejar. El funcionamiento concreto de cada tipo de rejilla se especifica en otros requisitos



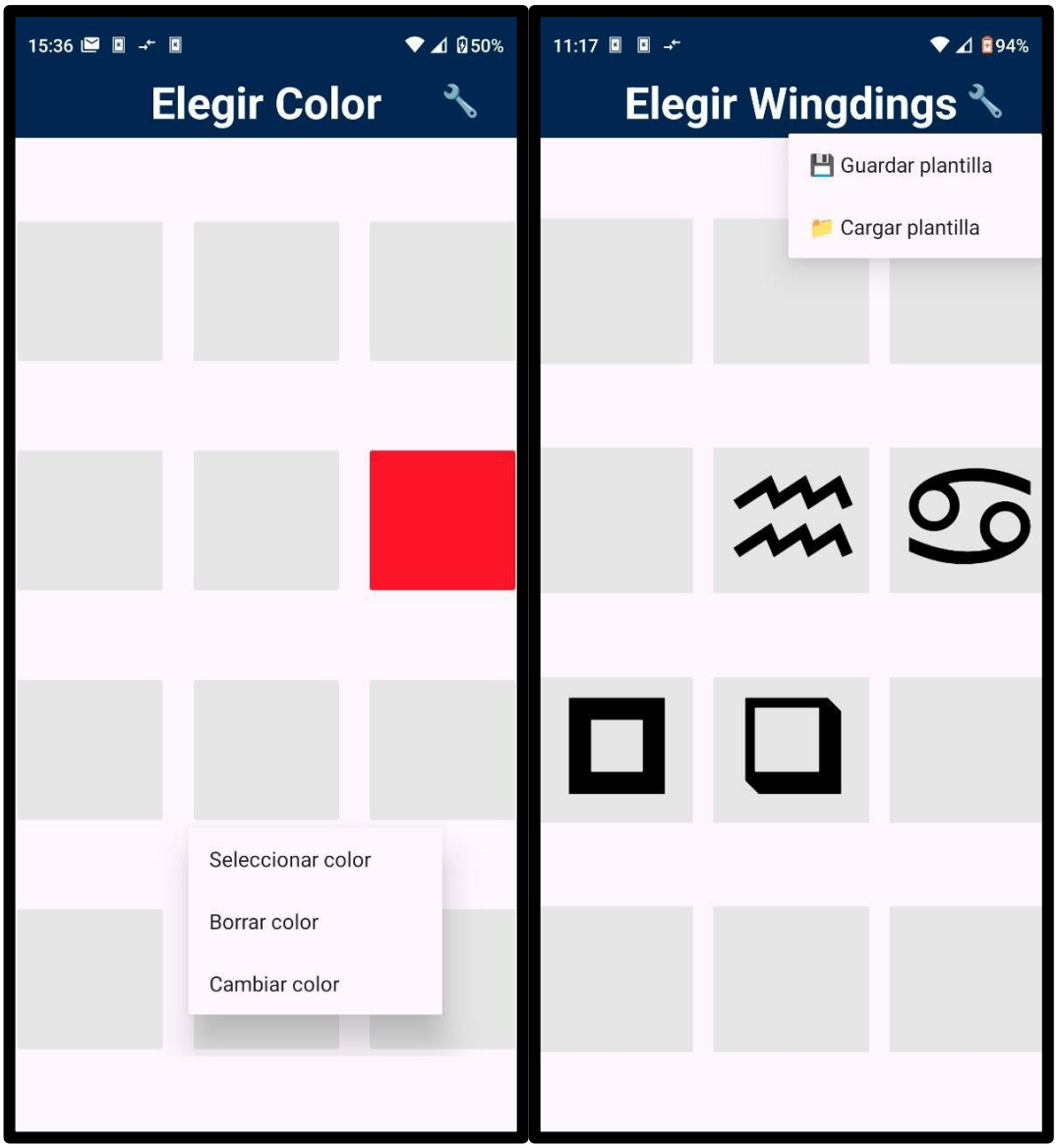
Nombre	Iteración 1: 016 – Creación de rejilla: Rejilla letras
Introducción	Creación de la rejilla de letras
Objetivo	Servir de punto de apoyo para la creación de la rejilla de letras usando el código de creación de rejillas general
Uso previsto	Este requisito se usará cuando se cree una rejilla de letras
Planteamiento de diseño	Se utilizan tres elementos principales:

	• **Selector de letra correcta** • **Selector de letra incorrecta** • **Selector de letra aleatoria** Dependiendo de la situación, durante la creación de rejilla se seleccionará la letra apropiada para cada casilla utilizando estos tres selectores
Diseño de la solución	Durante la creación de una rejilla, se comprobará el tipo de la misma. En caso de ser rejilla de letras, se usarán dos funciones: • ***eligeLetraCorrecta*****:** Se utiliza en caso de que, al iterar sobre las casillas de la rejilla, se considere en el bucle *for*, que se debe establecer en la casilla actual una letra correcta, es decir, la que el usuario debe pulsar. Esta función recibe el botón actual, itera sobre la lista de letras correctas, selecciona una aleatoriamente, y establece el texto del botón con esa letra. Tras esto se dota de funcionalidad al botón, usando un *setOnClickListener* que enlaza el

	botón a la función *botonAcierto*, que se encargará de aumentar el número de aciertos y manejar el resto de parámetros • ***eligeLetraIncorrecta*:** Se utiliza en caso de que, al iterar sobre las casillas de la rejilla, se considere en el bucle *for*, que se debe establecer en la casilla actual una letra incorrecta, es decir, la que el usuario debe ignorar. Esta función recibe el botón actual, itera sobre la lista de letras incorrectas (es decir, todas las seleccionadas en la plantilla excepto las que además de estar en la plantilla están en la lista de correctas elegidas), selecciona una aleatoriamente, y establece el texto del botón con esa letra. En caso de que el número de letras incorrectas seleccionadas sea igual o menor al de correctas, si por ejemplo el usuario ha seleccionado solo una correcta y solo ha puesto esa en la plantilla de incorrectas también, se usará una letra aleatoria de la "A" a la

	"Z" usando el método *eligeLetraRandom*. Tras esto se dota de funcionalidad al botón, usando un *setOnClickListener* que aumenta el número de fallos Tras configurar el botón con la letra adecuada, la creación de rejilla continúa hasta completarla con todos los botones necesarios


15:44
55%
Elegir Imagen
ALTO
15:36
50%
Elegir Letras
a



Nombre	Iteración 1: 017 – Pantalla elegir elemento (rejillas de imágenes, letras, colores y símbolos)
Introducción	Este requisito comprende la pantalla de selección de elementos correctos e incorrectos, conocida como plantilla, de la cual se recogen los elementos correctos que el usuario selecciona

Objetivo	Agrupar en una pantalla todos los elementos que el usuario ha configurado para usar en la creación de estas rejillas
Uso previsto	Se despliega esta vista cuando el usuario pulsa uno de los nueve cuadrados en la pantalla de creación de rejillas de elementos seleccionables. Este requisito se usa para seleccionar los elementos a usar en la creación de la rejilla
Planteamiento de diseño	Consiste en doce casillas a la que se pueden establecer distintos elementos dependiendo del tipo de rejilla que sea. Cuando se ponen todos los elementos deseados, se puede seleccionar el que se desee para aplicar a los botones correctos de la rejilla antes de crearla
Diseño de la solución	Cada uno de los nueve cuadrados disponibles dispone de una acción concreta al pulsarlos, que abre un menú con tres opciones: • **Seleccionar:** Elige el elemento y lo pone en el botón que se pulsó en la pantalla anterior, usando el id que se pasó a esta actividad desde la anterior para reconocer este botón. Tras eso se vuelve a la actividad anterior

	• **Borrar:** Limpia el contenido del cuadrado pulsado dependiendo del tipo de rejilla que sea
	
	• **Cambiar:** Cambia el contenido del cuadrado pulsado. Para esto se dispone de varias formas de hacerlo dependiendo de la rejilla:
	  ○ **Rejilla de colores:** Se abre el selector de colores integrado en la aplicación
	
	  ○ **Rejilla de imágenes:** Se abre el selector de archivos de Android creando una nueva actividad y esperando su resultado
	
	  ○ **Rejilla de letras:** Se abre un campo de texto para escribir. Se acepta únicamente un elemento, de tipo carácter alfabético
	
	  ○ **Rejilla de wingdings:** Se abre un campo de texto para escribir. Se acepta únicamente un elemento,

	de tipo carácter alfanumérico o símbolos. Se utiliza la fuente wingdings en este campo y en el cuadrado de selección Esta pantalla dispone además de un botón en la esquina superior derecha que contiene un menú con dos opciones: • **Guardar plantilla:** Guarda un archivo *JSON* con la información de la plantilla. Para esto se usa la clase *GestorBackups* • **Cargar plantilla:** Carga un archivo *JSON* con la información de la plantilla. Para esto se usa la clase *GestorBackups* Al seleccionar el valor deseado se vuelve a la actividad anterior y se establece el elemento elegido

5.8 Segunda iteración

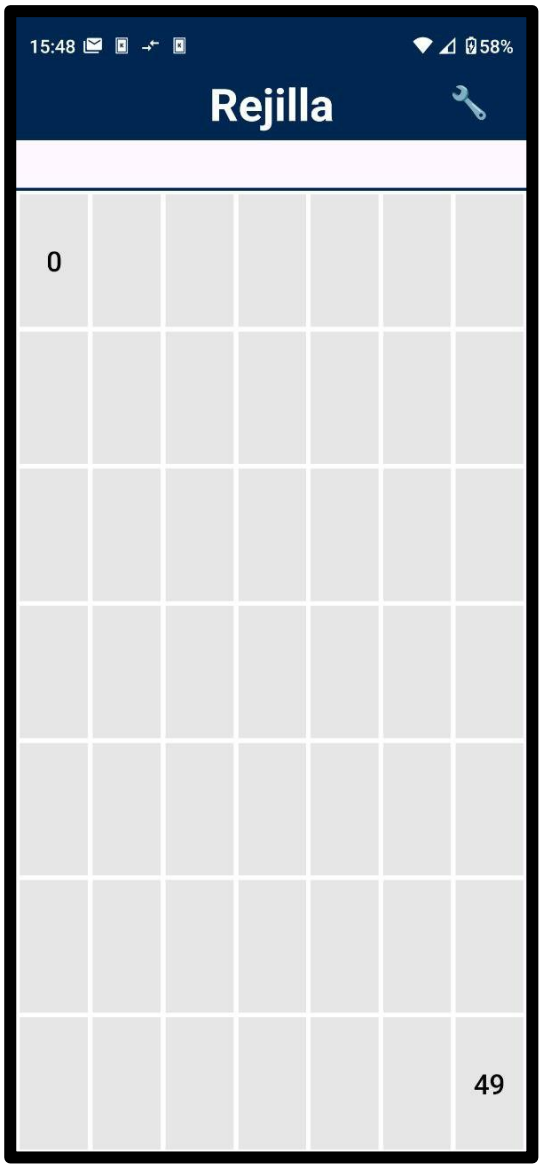
En esta segunda iteración se ha enfocado el trabajo a completar las funcionalidades restantes y ajustar las existentes en base al *feedback* obtenido del cliente en la primera iteración.

Se ha dedicado una cantidad considerable de tiempo de esta iteración a completar la creación de algunas de las rejillas disponibles en la aplicación, y se han hecho ajustes menores a la interfaz, el tema y las características básicas de los menús de creación de rejillas.

El objetivo principal de esta iteración es la compleción del mayor número posible de requisitos en un tiempo moderado, que permita hacer una nueva muestra al cliente lo antes posible y evaluar si la evolución de la aplicación va acorde a sus expectativas, enfocando desde ese momento todos los esfuerzos a formalizar y completar el resto de las funcionalidades en la tercera iteración.

A todo esto, se añade la creación y gestión de la base de datos SQLite y la recogida de datos de usuario para una primera muestra de cómo funcionará el sistema en este apartado.

A continuación, se presentan las tablas de requisitos de la segunda iteración:



Nombre	Iteración 2: 018 – Tarea preliminar
Introducción	Creación de la tarea preliminar y funcionalidad de la misma
Objetivo	Iniciar una tarea preliminar antes de la rejilla en caso de estar activada esta función para recoger el tiempo que el usuario tarda en completarla y usar esto como medida de correción

Uso previsto	Cuando se inicia la rejilla del tipo seleccionado, antes se hace una comprobación para verificar si se activó la tarea preliminar. En caso afirmativo, esta se crea y se muestra
Planteamiento de diseño	La tarea preliminar tiene dos botones, uno en cada extremo de la rejilla, y al completarla se guarda el tiempo tardado
Diseño de la solución	Al iniciar la rejilla se bloquea la creación de esta hasta que se complete la tarea preliminar. Esta tarea preliminar consiste en una rejilla del tamaño seleccionado con solo dos botones: uno en la esquina superior izquierda con el número 0, y otro en la esquina inferior derecha con el número resultado de multiplicar filas y columnas, es decir, este número es el número de casillas. Al acabar la tarea se guarda el tiempo, inicia la rejilla y se notifica al usuario. El tiempo guardado se usará de corrección en los valores corregidos calculados de la rejilla

Nombre	Iteración 2: 019 – Distracción línea
Introducción	Funcionalidad de la distracción de línea en la rejilla
Objetivo	Mostrar una línea configurable en movimiento que atraviesa la pantalla en varias direcciones para distraer al usuario
Uso previsto	Se usará en la rejilla cuando el usuario la active para distraerlo mientras completa la rejilla

Planteamiento de diseño	Se recogen sus parámetros y se configura e inicia su animación en la actividad de rejilla en caso de estar activada. Tras esto se empieza a mover por pantalla y es configurable desde el menú rápido de distracciones de la pantalla de rejillas
Diseño de la solución	Se cargan la configuración de la línea y un *Boolean* que indica si está activada o no desde un *SharedPreferences*. Tras esto se inicia en caso de estar activada, creando un objeto *Animation*, y aplicándolo a un elemento que permanece invisible hasta que se active la línea en la pantalla de la rejilla. Este *Animation* es configurado para moverse en horizontal, vertical o aleatorio dependiendo de la configuración elegida, y una vez configurado se inicia en la vista en la que se creó la línea, haciendo que aparezca la línea y se mueva por encima de la rejilla mientras la completamos

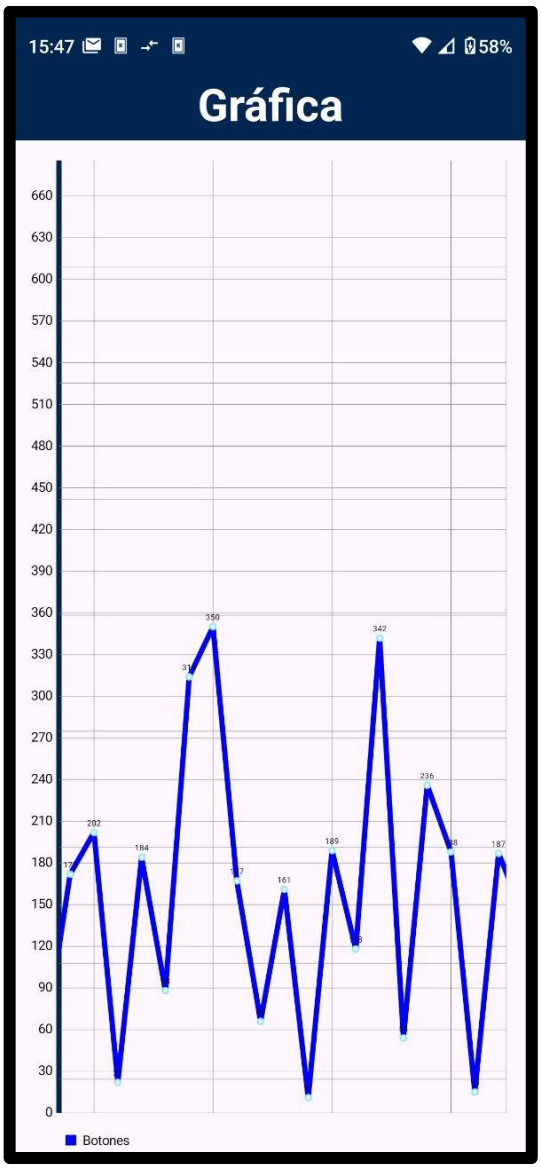
Nombre	Iteración 2: 020 – Distracción metrónomo
Introducción	Metrónomo configurable en la rejilla que hará un sonido con un intervalo de milisegundos configurable
Objetivo	Este requisito dota la aplicación de un metrónomo configurable que, una vez

	activo, hace sonidos durante la realización de una rejilla para distraer al usuario
Uso previsto	Se usa cuando se activa la distracción de metrónomo durante la rejilla
Planteamiento de diseño	Se comprueba al iniciar la rejilla si se ha activado esta función, y en caso afirmativo, se inicia el metrónomo, que hará un sonido con el intervalo guardado en la configuración
Diseño de la solución	Al iniciar el metrónomo se crean dos objetos: un *MediaPlayer* y un *Handler*. El *Handler* se configura para ser activado en el intervalo de milisegundos configurado. Al llegar a este tiempo, se activa el *MediaPlayer*, se reproduce el sonido del metrónomo, y se reinicia para reproducirlo de nuevo tras el tiempo establecido. Al destruir la actividad el metrónomo se detiene. Los milisegundos y el *Boolean* que indica si el metrónomo está activo o no se almacenan y leen de la configuración de la aplicación con un *SharedPreferences* y un *Editor*

Nombre	Iteración 2: 021 – Datos de usuario (front-end)
Introducción	Formulario de datos del usuario de la aplicación
Objetivo	Recoger los datos pedidos del usuario para añadirlos a la información de cada rejilla completada

Uso previsto	El usuario deberá entrar a rellenar los datos en caso de que quiera que estos estén presentes en los resultados de las pruebas que haga en la aplicación
Planteamiento de diseño	Se utiliza un formulario lineal vertical con cada una de las casillas de texto a rellenar por el usuario y algunos *dropdown list* para elegir distintas opciones
Diseño de la solución	Se dan las siguientes opciones: • **Nombre:** EditText • **Apellidos:** EditText • **Edad:** EditText numérico • **Género:** EditText • **País:** EditText • **Fecha:** Se actualiza autimáticamente a la fecha actual • **Deportista (Deporte):** EditText • **Deportista (Posición):** EditText • **Deportista (Lateralidad):** Spinner con dos opciones: Derecha e Izquierda

	El nombre de cada opción (véase el dato que se debe introducir en cada casilla), viene dado en un *Hint* dentro de la propia casilla del *EditText*. El botón Aceptar se sitúa en la parte inferior de la pantalla



Nombre	Iteración 2: 022 – Gráfica de tiempo de botones (front-end)
Introducción	Interfaz de la gráfica de tiempo de los botones pulsados en una prueba
Objetivo	Mostrar una gráfica que refleje de forma clara el tiempo tardado en pulsar cada botón
Uso previsto	Se puede acceder pulsando el botón de ver gráfica en los detalles de una rejilla una vez se acaba la prueba
Planteamiento de diseño	La pantalla presenta la gráfica en color azul oscuro
Diseño de la solución	La gráfica presenta con puntos unidos por una línea azul oscura el tiempo tardado en pulsar cada botón. La escala es ajustada dependiendo del tiempo máximo tardado en pulsar un botón en toda la prueba. Con gestos, usando dos dedos, se puede ampliar o reducir la gráfica horizontal o verticalmente. Se ha usado para este requisito el proyecto Github: https://github.com/PhilJay/MPAndroidChart

Nombre	Iteración 2: 023 – Historial de pruebas (front-end)
Introducción	Lista de las pruebas realizadas en la aplicación
Objetivo	Muestra una lista con todas las pruebas realizadas para que el usuario tenga acceso a los resultados de pruebas pasadas

Uso previsto	El usuario entra a esta pantalla a través de la opción del menú de inicio. Aquí puede consultar pruebas anteriores. También ofrece la opción de guardar, cargar, y eliminar las pruebas
Planteamiento de diseño	Es una lista simple con todas las entradas de la base de datos. Hay un botón de opciones en la esquina superior derecha
Diseño de la solución	Se utiliza un *ListView* que contiene la fecha de la prueba realizada. En caso de haber introducido un nombre, también se mostrará al lado de la fecha. En la esquina superior derecha hay un botón con el icono de una llave inglesa que abre un pequeño menú con las siguientes opciones: • **Limpiar historial:** Elimina todo de la base de datos y limpia la pantalla de historial • **Guardar historial:** Abre una pantalla de selección de localización para guardar el archivo de la base de datos en el dispositivo • **Cargar historial:** Abre una pantalla de selección de archivos para cargar desde el dispositivo la

	copia de seguridad hecha de la base de datos Es posible interactuar con los elementos de la lista para dirigirse a la pantalla de detalles de la prueba pulsada

Nombre	Iteración 2: 024 – Historial de pruebas (back-end primera iteración)
Introducción	Funcionalidades básicas de la pantalla de historial
Objetivo	Proveer la pantalla de historial de funciones básicas para mostrar las pruebas realizadas y redirigir a ellas
Uso previsto	Se usará para cargar las pruebas y la pantalla de detalles de las mismas
Planteamiento de diseño	Al iniciar la actividad se cargan todas las fechas de las pruebas, y en caso de tenerlo, el nombre del usuario, y se muestran en cada línea
Diseño de la solución	En la primera iteración se cargan de la base de datos usando el método *getDates* de *DatabaseHelper* todas las entradas de la base de datos, y se muestran línea a línea. También se cargan los nombres con el método *getNombresPersonas* de *DatabaseHelper,* y en caso de existir, se

	añade el nombre al lado de la fecha en la entrada correspondiente. Al pulsar una prueba se carga la pantalla de detalles de pruebas, a la que se le pasa la fecha exacta de realización de la misma para que cargue la información correspondiente a partir de esta. En siguientes iteraciones se añadirán todos los parámetros que faltan

Nombre	Iteración 2: 025 – Pantalla de creación rejilla numérica (back-end)
Introducción	Funcionalidad del back-end de la rejilla numérica
Objetivo	Dotar de funcionalidad la pantalla de creación de rejilla numérica usando sus opciones para crear la rejilla
Uso previsto	Se usará para crear la rejilla numérica en base a los parámetros introducidos
Planteamiento de diseño	Se introduce un valor inicial que se usará para calcular si, sumando la cantidad de casillas y en base a los saltos, entra dentro de los límites posibles, teniendo en cuenta el orden de incremento
Diseño de la solución	El valor inicial y el valor de salto se introducen en dos *EditText* que solo aceptan valores numéricos.

	El orden de tachado se introduce en un *Spinner* con dos valores: ascendente y descendente Cuando se introduce un valor inicial y un valor de salto, se hace un cálculo para comprobar, en base al tipo de incremento, si son valores válidos. Esta información se muestra luego en un *TextView* que indica el valor máximo o mínimo requerido, y si se cumple o no. Cuando se pulsa el botón de creación de rejilla se comprueba por último si los valores son correctos, y en caso de no serlos, se notifica al usuario. Si los valores son correctos se guardan en configuración con un *Editor* de un *SharedPreferences* y se pasa a crear la rejilla

Nombre	Iteración 2: 026 – Pantalla de creación rejilla colores (back-end)
Introducción	Funcionalidad del back-end de la rejilla de colores
Objetivo	Dotar de funcionalidad la pantalla de creación de rejilla de colores usando sus opciones para crear la rejilla
Uso previsto	Se usará para crear la rejilla de colores en base a los parámetros introducidos

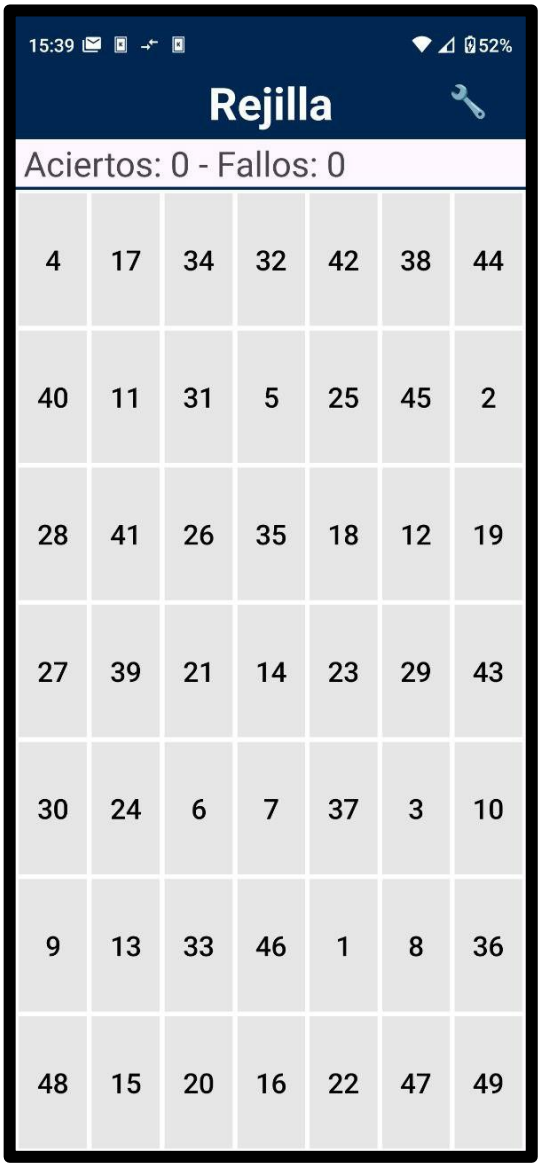
Planteamiento de diseño	Se usan botones vacíos que se rellenan dependiendo de los colores seleccionadas. El diseño consiste en botones que, al ser pulsados, crean un *Intent* en el que se decide el color a fijar en el botón pulsado, pasando a este el id del botón en cuestión
Diseño de la solución	Cada botón cuadrado pulsado envía su id a una pantalla de selección del color que se elija, y al volver a cargar la página de creación de rejilla de colores una vez elegida el color a seleccionar, se pondrá ese color como color de fondo en el cuadrado elegido. Para esto se utiliza *SharedPreferences,* donde se guarda cada uno de los nueve colores en formato *Integer*, que se cargan al elegir uno nuevo. Si se acaba de entrar en el menú para crear una rejilla nueva, se vacían todos los cuadrados a la espera de que el usuario elija los colores. Esto se hace poniendo a nulo el valor de cada uno de los colores en el *SharedPreferences.* Cuando se pulsa un cuadrado, este lleva al método *setButton,* que configura una

	actividad nueva con el id del botón pulsado y abre la pantalla de selección de colores. Cuando se pulsa el botón de creación de rejilla se recoge el contenido de los cuadrados, y si alguno de ellos contiene un color de fondo, se procede a crear la rejilla, creando un nuevo *Intent* al que se le añade el valor de la cantidad de botones seleccionada en el *spinner* de cantidad de botones

Nombre	Iteración 2: 027 – Pantalla de creación rejilla imágenes (back-end)
Introducción	Funcionalidad del back-end de la rejilla de imágenes
Objetivo	Dotar de funcionalidad la pantalla de creación de rejilla de imágenes usando sus opciones para crear la rejilla
Uso previsto	Se usará para crear la rejilla de imágenes en base a los parámetros introducidos
Planteamiento de diseño	Se usan botones vacíos que se rellenan dependiendo de las imágenes seleccionadas. El diseño consiste en botones que, al ser pulsados, crean un *Intent* en el que se decide la imgen a fijar en el botón pulsado, pasando a este el id del botón en cuestión

Diseño de la solución	Cada botón cuadrado pulsado envía su id a una pantalla de selección de la imagen que se elija, y al volver a cargar la página de creación de rejilla de imágenes una vez elegida la imagen a seleccionar, se pondrá esa imagen en el cuadrado elegido. Para esto se utiliza *SharedPreferences,* donde se guarda cada una de las nueve imágenes elegidas codificadas en base 64, que se decodifican en base 64 y se cargan al elegir una nueva. Si se acaba de entrar en el menú para crear una rejilla nueva, se vacían todos los cuadrados a la espera de que el usuario elija las imágenes. Esto se hace poniendo a nulo el valor de cada una de las letras en el *SharedPreferences.* Cuando se pulsa un cuadrado, este lleva al método *setButton,* que configura una actividad nueva con el id del botón pulsado y abre la pantalla de selección de imágenes. Cuando se pulsa el botón de creación de rejilla se recoge el contenido de los cuadrados, y si alguno de ellos contiene una imagen de fondo, se procede a crear la rejilla, creando un nuevo *Intent* al que

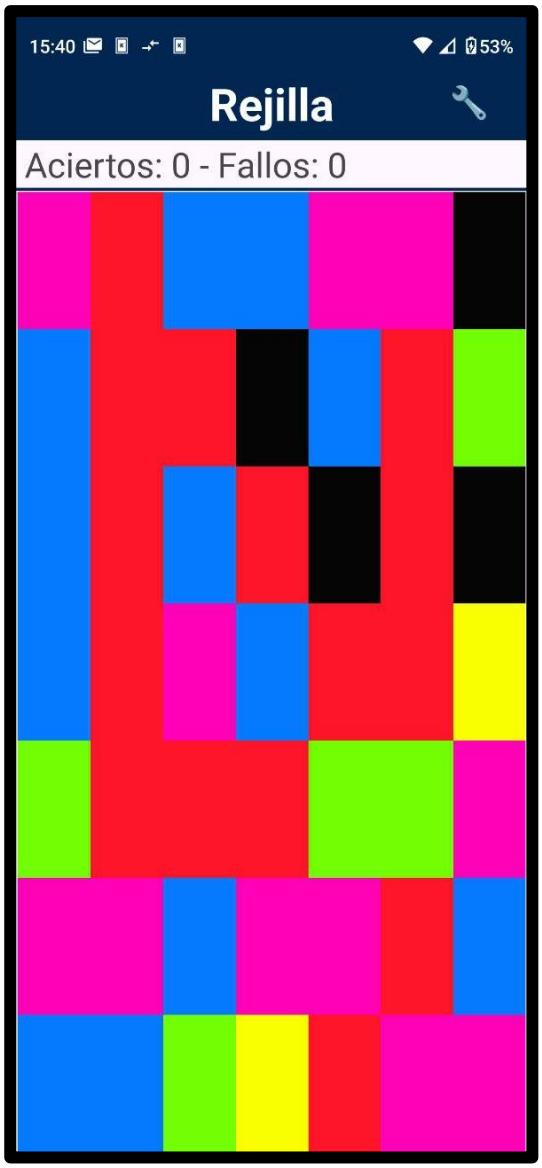
	se le añade el valor de la cantidad de botones seleccionada en el *spinner* de cantidad de botones



Nombre	Iteración 2: 028 – Creación de rejilla: Rejilla numérica
Introducción	Creación de la rejilla numérica

Objetivo	Servir de punto de apoyo para la creación de la rejilla numérica usando el código de creación de rejillas general
Uso previsto	Este requisito se usará cuando se cree una rejilla numérica
Planteamiento de diseño	Se generan números aleatorios comprendidos entre el valor inicial y el último valor, y tras fijar todos los números de la rejilla, se comprueba si el número pulsado es el que corresponde al siguiente que había que pulsar en el orden establecido
Diseño de la solución	Durante la creación de una rejilla, se comprobará el tipo de la misma. En caso de ser rejilla numérica, se usará la clase *GestorSiguienteValor* para elegir un número aleatorio que poner en la casilla actual acorde a los límites pertinentes y teniendo en cuenta que no haya repeticiones. Esta función recibe el botón actual y devuelve el mismo botón, pero con el número correcto establecido. Tras esto se dota de funcionalidad al botón, comprobando si se está usando una rejilla numérica y en ese caso llamando a la función *asignarOrdenNumeros,* que gestiona la asignación de orden a las casillas, usando

	un *setOnClickListener* que enlaza el botón a la función *botonAciertoOrden*, que se encargará de comprobar que el botón pulsado es el inmediatamente siguiente al anterior, y en caso contrario, aumenta el número de fallos. Tras configurar el botón con la letra adecuada, la creación de rejilla continúa hasta completarla con todos los botones necesarios

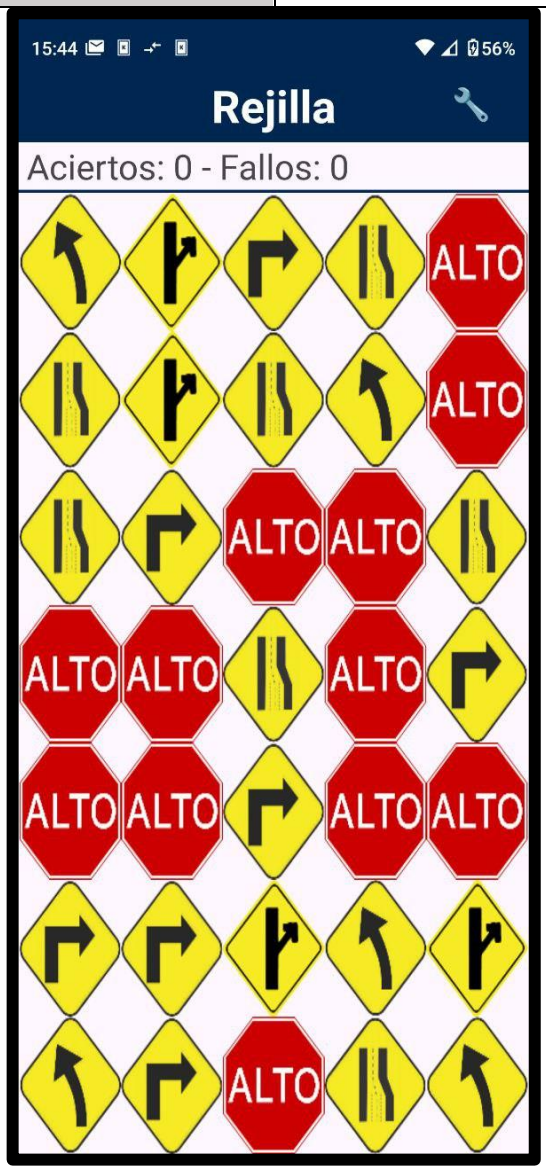


Nombre	Iteración 2: 029 – Creación de rejilla: Rejilla colores
Introducción	Creación de la rejilla de colores
Objetivo	Servir de punto de apoyo para la creación de la rejilla de colores usando el código de creación de rejillas general
Uso previsto	Este requisito se usará cuando se cree una rejilla de colores
Planteamiento de diseño	Se utilizan dos elementos principales:

	• **Selector de color correcto** • **Selector de color incorrecto** Dependiendo de la situación, durante la creación de rejilla se seleccionará el color apropiado para cada casilla utilizando estos dos selectores
Diseño de la solución	Durante la creación de una rejilla, se comprobará el tipo de la misma. En caso de ser rejilla de colores, se usarán dos funciones: • *eligeColorCorrecto*: Se utiliza en caso de que, al iterar sobre las casillas de la rejilla, se considere en el bucle *for*, que se debe establecer en la casilla actual un color correcto, es decir, el que el usuario debe pulsar. Esta función recibe el botón actual, itera sobre la lista de colores correctos, selecciona uno aleatoriamente, y establece el color de fondo del botón con este. Para establecer el color se recoge del *SharedPreferences* el *Integer* con el código del color. Tras esto se dota de funcionalidad al botón, usando un

	setOnClickListener que enlaza el botón a la función *botonAcierto*, que se encargará de aumentar el número de aciertos y manejar el resto de parámetros • *eligeColorIncorrecto*: Se utiliza en caso de que, al iterar sobre las casillas de la rejilla, se considere en el bucle *for*, que se debe establecer en la casilla actual un color correcto, es decir, el que el usuario debe ignorar. Esta función recibe el botón actual, itera sobre la lista de colores incorrectos (es decir, todos los seleccionados en la plantilla excepto los que además de estar en la plantilla están en la lista de correctos elegidos), selecciona uno aleatoriamente, y establece el color de fondo del botón con este. En caso de que el número de colores incorrectos seleccionados sea igual o menor al de correctos, se usará el color por defecto, gris.

	Tras esto se dota de funcionalidad al botón, usando un *setOnClickListener* que aumenta el número de fallos Tras configurar el botón con la letra adecuada, la creación de rejilla continúa hasta completarla con todos los botones necesarios



Nombre	Iteración 2: 030 – Creación de rejilla: Rejilla imágenes
Introducción	Creación de la rejilla de imágenes
Objetivo	Servir de punto de apoyo para la creación de la rejilla de imágenes usando el código de creación de rejillas general
Uso previsto	Este requisito se usará cuando se cree una rejilla de imágenes
Planteamiento de diseño	Se utilizan tres elementos principales: • **Selector de imagen correcta** • **Selector de imagen incorrecta** Dependiendo de la situación, durante la creación de rejilla se seleccionará la imagen apropiada para cada casilla utilizando estos dos selectores
Diseño de la solución	Durante la creación de una rejilla, se comprobará el tipo de la misma. En caso de ser rejilla de imágenes, se usarán dos funciones: • ***eligeimagenCorrecta:*** Se utiliza en caso de que, al iterar sobre las casillas de la rejilla, se considere en el bucle *for*, que se debe establecer en la casilla actual una imagen correcta, es decir, la que el usuario debe pulsar.

	Esta función recibe el botón actual, itera sobre la lista de imágenes correctas, selecciona una aleatoriamente, y establece el fondo del botón con la imagen actual usando un *BitmapDrawable*. Tras esto se dota de funcionalidad al botón, usando un *setOnClickListener* que enlaza el botón a la función *botonAcierto*, que se encargará de aumentar el número de aciertos y manejar el resto de los parámetros • ***eligeimagenIncorrecta:*** Se utiliza en caso de que, al iterar sobre las casillas de la rejilla, se considere en el bucle *for*, que se debe establecer en la casilla actual una imagen incorrecta, es decir, la que el usuario debe ignorar. Esta función recibe el botón actual, itera sobre la lista de imágenes incorrectas, selecciona una aleatoriamente, y establece el fondo del botón con la imagen seleccionada. En caso de que el número de imágenes incorrectas

	seleccionadas sea igual o menor al de correctas, se vaciarán los botones erróneos, mostrándose como cuadrados grises. Tras esto se dota de funcionalidad al botón, usando un *setOnClickListener* que aumenta el número de fallos Tras configurar el botón con la imagen adecuada, la creación de rejilla continúa hasta completarla con todos los botones necesarios

Nombre	Iteración 2: 031 – Registro de aciertos
Introducción	Se cuentan y muestran los aciertos en la rejilla mientras se hace
Objetivo	Mostrar al usuario sus aciertos, de esta forma es consciente de que todo está funcionando correctamente y que sus aciertos están siendo registrados

Uso previsto	Se muestra mientras el usuario está completando una rejilla, y se actualiza al pulsar un botón correcto
Planteamiento de diseño	Se utiliza un contador que incrementa al pulsar una casilla correcta. Tras esto se actualiza el valor
Diseño de la solución	La información está situada en la parte superior de la rejilla, debajo de la barra superior. Se presenta junto a la cantidad de fallos, y se actualiza cada vez que se pulsa un botón correcto, cargando la cantidad de aciertos hasta el momento sumado al último acierto

Nombre	Iteración 2: 032 – Registro de fallos
Introducción	Se cuentan y muestran los fallos en la rejilla mientras se hace
Objetivo	Mostrar al usuario sus fallos, de esta forma es consciente de que todo está funcionando correctamente y que sus fallos están siendo registrados
Uso previsto	Se muestra mientras el usuario está completando una rejilla, y se actualiza al pulsar un botón erróneo

Planteamiento de diseño	Se utiliza un contador que incrementa al pulsar una casilla incorrecta. Tras esto se actualiza el valor
Diseño de la solución	La información está situada en la parte superior de la rejilla, debajo de la barra superior. Se presenta junto a la cantidad de aciertos, y se actualiza cada vez que se pulsa un botón incorrecto, cargando la cantidad de fallos hasta el momento sumado al último error

Nombre	Iteración 2: 033 – Reproductor de música (back-end)
Introducción	Se maneja el reproductor de música de la aplicación desde la pantalla de rejilla y se ajusta la respuesta a la interacción con la interfaz rápida del mismo
Objetivo	Dar funcionalidad al menú rápido del reproductor de música y reproducir la canción en la rejilla cuando se requiere
Uso previsto	Se usa cuando el usuario quiere reproducir, parar y cambiar la canción
Planteamiento de diseño	Se utiliza un menú flotante para controlar el reproductor, y este funciona utilizando una clase disponible en Android y leyendo la canción seleccionada de la configuración de la aplicación
Diseño de la solución	Al iniciar una rejilla se comprueba si se activó la opción de música en la creación de la rejilla. Esta información se lee desde

	la configuración de la aplicación con un *SharedPreferences*. En caso afirmativo, y si se ha seleccionado una canción se inicia la reproducción de la canción y se ofrece al usuario la posibilidad de pararla, cambiarla o reproducirla de nuevo desde el menú rápido. El reproductor usa un objeto de la clase *MediaPlayer*, que se gestiona en la clase *RejillaActivity*. Al finalizar la rejilla se detiene la música

Nombre	Iteración 2: 034 – Reproductor de música (front-end rápido)
Introducción	Interfaz rápida de configuración de música en la rejilla
Objetivo	Tener una forma rápida de hacer ajustes simples a la música durante el trascurso de la rejilla
Uso previsto	Se usará mientras se está haciendo una rejilla para hacer ajustes rápidos a la música si se necesita
Planteamiento de diseño	Es un menú simple de tres opciones que se despliega pulsando el botón de configuración en la esquina superior derecha
Diseño de la solución	En el trascurso de la rejilla hay un botón con el icono de una llave inglesa en la parte superior derecha de la rejilla. Al pulsarlo aparece un pequeño menú flotante que muestra tres opciones: • **Reproducir:** Reproduce la canción • **Parar:** Para la canción • **Cambiar canción:** Abre un menú de selección de archivos de audio del dispositivo para seleccionar la canción Este menú está diseñado para poder controlar la música desde la rejilla de

	forma rápida sin tener que entrar a la configuración de música del menú principal. Para el funcionamiento del menú se ha usado un *popup floating menu* de Android asociado al botón con icono

Nombre	Iteración 2: 035 – Implementación de base de datos SQLite para las pruebas
Introducción	Implementación de las clases y métodos necesarios para desplegar la base de datos en la aplicación
Objetivo	Tener un sistema de almacenamiento persistente de los datos obtenidos en las rejillas que hace el usuario
Uso previsto	Almacenar los datos de una rejilla completada una vez esta finaliza y recoger estos datos para mostrarlos en la página de historial y detalles
Planteamiento de diseño	Se crea una clase para controlar los accesos a la base de datos, tanto de escritura como lectura
Diseño de la solución	Se crea una base de datos *SQLite* y una clase *DatabaseHelper* que extiende a *SQLiteOpenHelper* y se encarga de gestionar los accesos a la base de datos. La base de datos se llama “tests.db” y se estructura de forma simple en una serie de *Strings* que contienen la información

	del usuario y sus resultados en la rejilla. Estos datos no requerirán ser usados en formato número una vez se almacenan, por lo que incluso los numéricos están en formato *String* para poder ser mostrados más fácilmente y no necesitar una conversión previa. La base de datos solo presenta una tabla, "tests", que recoge toda la información. Se utiliza principalmente la fecha de realización de la prueba para acceder a su entrada en la base de datos. De esta forma se aprecia de forma más intuitiva cuál es la prueba seleccionada que si se usa el id aleatorio creado. Los datos recogidos y sus tipos son: • **id INTEGER PRIMARY KEY AUTOINCREMENT:** id de la entrada en la tabla • **realizado TEXT:** Fecha de realización de la prueba, contiene minutos, segundos y milisegundos • **nombre TEXT:** Nombre del usuario recogido del formulario rellenado por este. Cabe mencionar que este nombre es

	distinto al del usuario de MenPas con el que se inicia sesión en la aplicación para usarla • **apellidos TEXT:** Apellidos del usuario recogido del formulario rellenado por este • **edad TEXT:** Edad del usuario recogido del formulario rellenado por este • **genero TEXT:** Género del usuario recogido del formulario rellenado por este • **pais TEXT:** País introducido por el usuario recogido del formulario rellenado por este • **deporte TEXT:** Deporte practicado por el usuario recogido del formulario rellenado por este • **posicion TEXT:** Posición del usuario en el deporte practicado recogido del formulario rellenado por este

	• **lateralidad TEXT:** Lateralidad del usuario recogido del formulario rellenado por este, puede ser izquierda o derecha • **aciertos TEXT:** Cantidad de aciertos en la rejilla. Se recoge al acabar la prueba • **fallos TEXT:** Cantidad de fallos en la rejilla. Se recoge al acabar la prueba • **cant_botones TEXT:** Cantidad de botones de la rejilla. Se calcula multiplicando filas y columnas. Se recoge al acabar la prueba • **botones TEXT:** Lista de todos los botones pulsados (correctos e incorrectos) y el tiempo tardado en pulsarlos. Se rellena la lista conforme se pulsan botones en la rejilla y se recoge todo al acabar la prueba • **tiempoPreliminar TEXT:** Tiempo tardado en la prueba preliminar. Se recoge al acabar la prueba

	preliminar y se almacena al acabar la rejilla • **tiempoTotal TEXT:** Tiempo total empleado en completar la rejilla. Se recoge al acabar la prueba • **media TEXT:** Media del tiempo tardado en pulsar cada botón en la prueba. Se calcula al acabar la prueba y se recoge tras ello • **varianza TEXT:** Varianza del tiempo tardado en pulsar cada botón en la prueba. Se calcula al acabar la prueba y se recoge tras ello • **max TEXT:** Valor máximo del tiempo tardado en pulsar cada botón en la prueba. Se calcula al acabar la prueba y se recoge tras ello • **min TEXT:** Valor mínimo del tiempo tardado en pulsar cada botón en la prueba. Se calcula al acabar la prueba y se recoge tras ello

	• **tipo TEXT:** Tipo de la rejilla. Se recoge al acabar la prueba • **tam TEXT:** Filas x columnas. Se recoge al acabar la prueba • **tiempo_corregido TEXT:** Tiempo total empleado en completar la rejilla corregido usando el tiempo total de la prueba preliminar. Se calcula y recoge al acabar la prueba • **correccion TEXT:** Corrección aplicada en la prueba, es decir, el tiempo total tardado en completar la prueba preliminar. Se utiliza para calcular el resto de variables corregidas • **media_correccion TEXT:** Media del tiempo tardado en pulsar cada botón en la prueba corregida usando el tiempo total de la prueba preliminar. Se calcula y recoge al acabar la prueba • **varianza_correccion TEXT:** Varianza del tiempo tardado en pulsar cada botón en la prueba corregida usando el tiempo total

	de la prueba preliminar. Se calcula y recoge al acabar la prueba • **direccion_linea TEXT:** Dirección de la distracción de línea utilizada en la prueba. Se recoge al acabar la prueba • **velocidad_linea TEXT:** Velocidad de la distracción de línea utilizada en la prueba. Se recoge al acabar la prueba • **color_linea TEXT:** Color de la distracción de línea utilizada en la prueba. Se recoge al acabar la prueba • **grosor_linea TEXT:** Grosor de la distracción de línea utilizada en la prueba. Se recoge al acabar la prueba • **cancion TEXT:** Nombre de la canción usada en la distracción de música utilizada en la prueba. Se recoge al acabar la prueba • **metronomo_tiempo TEXT:** Cantidad de milisegundos cada la cual suena la distracción de

	metrónomo utilizado en la prueba. Se recoge al acabar la prueba • **emparejar_tiempo TEXT:** Cantidad de tiempo que pasa antes de que cambie el indicador de "siguiente botón a pulsar" en la rejilla cuando se activa la funcionalidad de emparejar con tiempo. Se recoge al acabar la prueba • **emparejar_orden TEXT:** Orden utilizado para emparejar en la prueba • **colores_fondo TEXT:** Colores de fondo utilizados en la prueba • **aciertos_absolutos TEXT:** Porcentaje de aciertos absolutos en la prueba. Se calcula y recoge al acabar la prueba. La fórmula usada es: *%Aciertos Absolutos = aciertos / Cantidad Total de Botones* • **aciertos_relativos TEXT:** Porcentaje de aciertos relativos

	en la prueba. Se calcula y recoge al acabar la prueba. La fórmula usada es: *%Aciertos Relativos* *= aciertos / (aciertos + errores)* • **errores_relativos TEXT:** Porcentaje de errores relativos en la prueba. Se calcula y recoge al acabar la prueba. La fórmula usada es: %Errores Relativos = errores / (aciertos + errores) • **errores_absolutos TEXT:** Porcentaje de errores absolutos en la prueba. Se calcula y recoge al acabar la prueba. La fórmula usada es: *%Errores* Absolutos = errores / Cantidad Total Botones • **eficacia TEXT:** Porcentaje de eficacia en la prueba. Se calcula y recoge al acabar la prueba. La fórmula usada es: %Eficacia = (aciertos – errores) / Cantidad Total Botones

	• **efectividad TEXT:** Porcentaje de efectividad en la prueba. Se calcula y recoge al acabar la prueba. La fórmula usada es: %Efectividad = (aciertos – errores) / (aciertos + errores) • **aleatorio TEXT:** Cantidad de segundos cada cuánto se aleatoriza la rejilla Todos estos valores se actualizan en la base de datos con su respectivo setter en la clase *DatabaseHelper*, introduciendo la fecha de la prueba y el valor a añadir como parámetros

Nombre	Iteración 2: 036 – Guardar datos recogidos de la rejilla en la base de datos (Primera iteración)
Introducción	Este requisito consiste en almacenar algunos datos de la rejilla en la base de datos para usarlos posteriormente en la aplicación
Objetivo	Guardar los resultados y datos de la rejilla realizada para poder acceder a ellos.

	En la primera iteración se almacenan datos básicos para poder mostrar cómo funcionará el sistema cuando esté completo
Uso previsto	Esto se usa para almacenar datos cada vez que se completa una rejilla
Planteamiento de diseño	Se llama a una función que guarda los datos de aciertos y errores de la rejilla en la base de datos al acabar
Diseño de la solución	Cuando se pulsa el último botón correcto de la rejilla, se llama a una función *guardarResultado* que almacena en la base de datos los aciertos y fallos de la rejilla. Para esto se llama a *DatabaseHelper* y se usa un *setter* para establecer los valores de acierto y fallo en la base de datos. Se almacenan únicamente estos datos por ahora para poder mostrar al cliente cómo funciona el sistema y hacer una primera toma de contacto con la base de datos y poder verificar que todo funciona correctamente. En iteraciones futuras se almacenarán el resto de datos

5.9 Tercera iteración

Para la tercera iteración del desarrollo se consolidan todas las funcionalidades restantes del proyecto, además de pulir y mejorar aspectos específicos de la interfaz de usuario. Este período representa una etapa crucial en el proceso de desarrollo, ya que se abordan aquellos aspectos que quedaron pendientes en las iteraciones anteriores y se trabaja en perfeccionar la experiencia del usuario.

Una de las principales metas durante esta iteración es completar la implementación de las funcionalidades que aún no se han integrado completamente en la aplicación. Esto implica la incorporación de características clave que fueron planificadas inicialmente pero que aún no se han desarrollado, así como resolver cualquier problema o bug que haya surgido durante las iteraciones anteriores.

Además de completar las funcionalidades pendientes, se dedica tiempo a perfilar y mejorar la interfaz de usuario y añadir ciertos menús y configuraciones que serán necesarias para acceder de forma cómoda a todas las funcionalidades de la aplicación.

Un punto clave a destacar en esta iteración es la compleción de todos los valores extraídos de la realización de una rejilla, que pasan a almacenarse en la base de datos junto a los parámetros que ya se implementaron en la segunda iteración.

Además, durante esta etapa se completa la implementación de las rejillas restantes. El total de rejillas implementadas ascenderá a seis: Rejilla de letras, colores, imágenes, abecedario, wingdings y numérica. Tras completar estas rejillas, se dispone finalmente de una demostración completa del funcionamiento de la aplicación con sus características más importantes de cara al cliente.

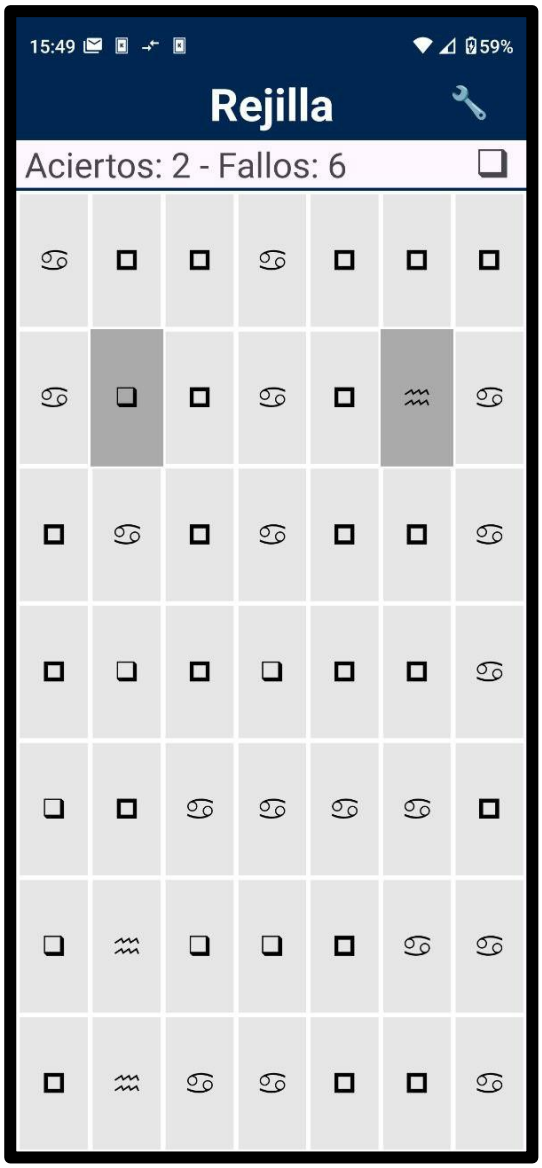
Las siguientes tablas detallan la implementación y definición de cada requisito de esta iteración:

Nombre	Iteración 3: 037 – Pantalla de creación rejilla wingdings (back-end)
Introducción	Funcionalidad de la pantalla de creación de rejilla wingdings
Objetivo	Establecer los elementos apropiados para la creación de la rejilla wingdings
Uso previsto	Se usará este requisito cada vez que se quiera crear una rejilla wingdings, tras pulsar el botón de crear rejilla
Planteamiento de diseño	El diseño de este requisito es idéntico al de la rejilla de elementos de tipo letras, con la excepción del uso de una fuente distinta, la fuente wingdings
Diseño de la solución	El diseño es el mismo que el de la rejilla letras a excepción de dos cambios: • Se permite el uso de caracteres numéricos y símbolos • El campo de texto editable que se abre para poder escribir el símbolo a usar y los cuadrados donde se ponen estos símbolos estás configurados para usar la fuente wingdings

Nombre	Iteración 3: 038 – Pantalla de creación rejilla abecedario (back-end)
Introducción	Funcionalidad del back-end de la rejilla abecedario

Objetivo	Dotar de funcionalidad la pantalla de creación de rejilla abecedario usando sus opciones para crear la rejilla
Uso previsto	Se usará para crear la rejilla abecedario en base a los parámetros introducidos
Planteamiento de diseño	Consiste en un campo de texto donde se introducen tres letras, que serán el valor inicial, un selector de tipo, y un selector del orden a seguir en el abecedario
Diseño de la solución	El valor inicial se introduce en un *EditText* que solo acepta como entrada tres letras. El tipo se selecciona en un *dropdown list* de tipo *spinner* que contiene cinco opciones: • **Mayúsculas:** Todas las letras serán mayúsculas • **Minúsculas:** Todas las letras serán minúsculas • **Aleatorio:** Se usarán aleatoriamente uno de los otros cuatro tipos para cada casilla • **AbCdEfGh:** Se alterna entre letra mayúscula y minúscula • **ABcDeFgH:** Se alterna entre letra minúscula y mayúscula

	El orden de tachado se introduce en un *spinner* con dos valores: ascendente y descendente Cuando se pulsa el botón de creación de rejilla se generan todas las cadenas de tres letras que se usarán, y se ordenan según se seleccionó en el orden de tachado, y se edita cada letra para adecuarse al tipo elegido

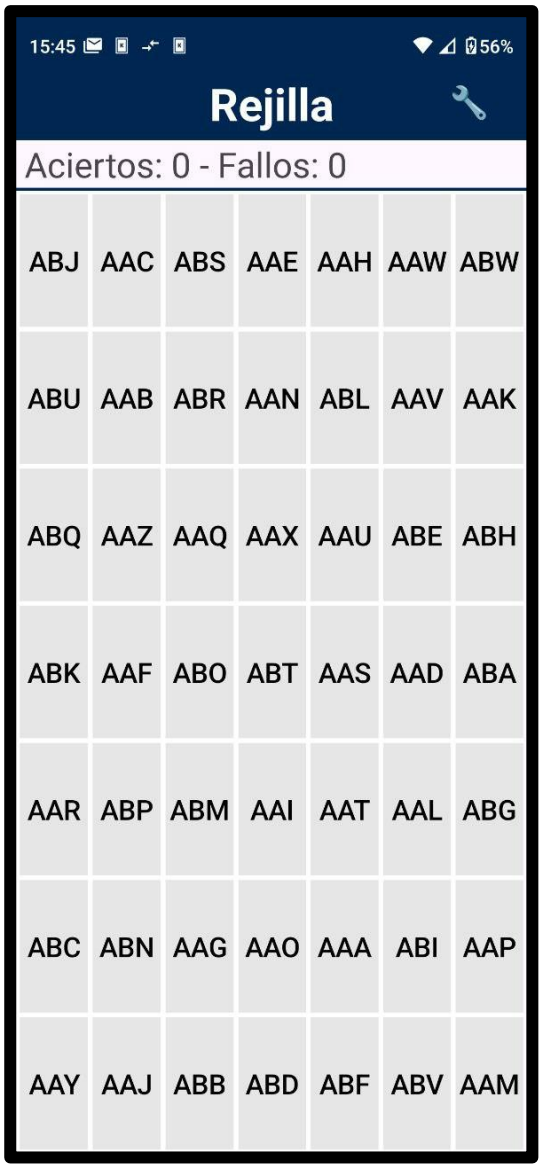


Nombre	Iteración 3: 039 – Creación de rejilla: Rejilla wingdings
Introducción	Creación de la rejilla wingdings
Objetivo	Servir de punto de apoyo para la creación de la rejilla wingdings usando el código de creación de rejillas general
Uso previsto	Este requisito se usará cuando se cree una rejilla wingdings
Planteamiento de diseño	Idéntico al de la rejilla de letras
Diseño de la solución	Idéntico al de la rejilla de letras.

	Se utilizan las mismas funciones y componentes


15:46
56%
Rejilla
Aciertos: 0 - Fallos: 0
aAv aBf aAm aBj aBp aBt aAk
aAz aAh aAf aAd aBk aAn aBd
aAl aBn aBq aBh aAo aAg aAw
aBv aBc aBe aBw aBo aAb aBb
aAu aBg aBl aAe aAq aAa aAi
aBa aBr aAr aAx aBu aAp aAc
aAy aBs aAs aBi aBm aAj aAt
15:45
56%
Rejilla
Aciertos: 0 - Fallos: 0
AbH AaT AaP AaG AaF AbO AbT
AbD AbF AbL AbQ AbI AaH AbR
AbA AaB AaJ AaQ AaC AbU AaI
AaK AbV AbE AaS AaO AaZ AaM
AaL AaA AaY AbP AaX AbN AbJ
AaE AaW AaV AbC AbW AaU AbM
AaN AbB AbS AaR AaD AbK AbG

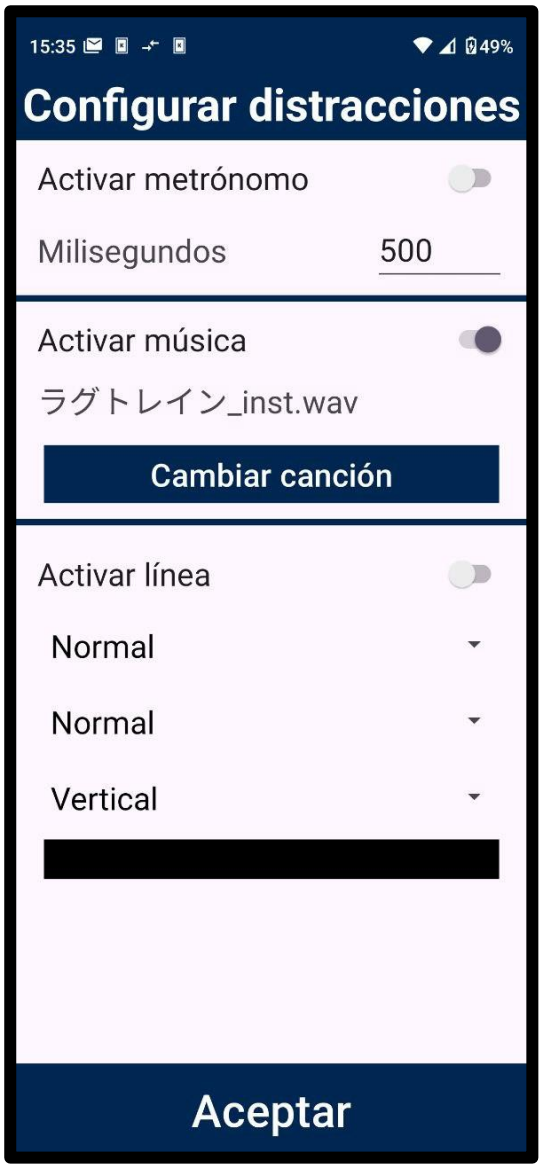

15:45
56%
Rejilla
Aciertos: 0 - Fallos: 0
aap aay abf aas aao abv aah
abw aae aaq aaj abl abg abp
aab aaz aaw aba aav aam aac
abo aau aat abi aad abr abs
abj abt aai abc abk abd aax
abq abn aaa abh abu aak abm
aaf aal aag abb aar abe aan
15:45
56%
Rejilla
Aciertos: 0 - Fallos: 0
ABF AaT aAa aAv ABU AAH ABE
AaJ ABC aad aAf aBp AAU AbN
ABB AbR AAC aBw ABA aBg abd
ABK AbV AAW AaS aAy AAI aBj
AAK abt AaG aAe aBm abi AaO
abq AbH AaL aan AaB AAP aaz
aar AbS aAm aax AAQ AbL ABO



Nombre	Iteración 3: 040 – Creación de rejilla: Rejilla abecedario
Introducción	Creación de la rejilla abecedario
Objetivo	Servir de punto de apoyo para la creación de la rejilla abecedario usando el código de creación de rejillas general
Uso previsto	Este requisito se usará cuando se cree una rejilla abecedario

Planteamiento de diseño	Se recogen los valores generados en la pantalla de creación y se colocan en cada casilla de la rejilla de forma desordenada
Diseño de la solución	Durante la creación de una rejilla, se comprobará el tipo de la misma. En caso de ser rejilla numérica, se usará la clase *GestorSiguienteValor* para elegir una cadena de letras aleatoria que poner en la casilla actual, cogiéndola de la lista generada en la creación, y teniendo en cuenta que no haya repeticiones. Esta función recibe el botón actual y devuelve el mismo botón, pero con la cadena de letras correcta establecida. Tras esto se dota de funcionalidad al botón, comprobando si se está usando una rejilla abecedario y en ese caso llamando a la función *asignarOrdenAbece,* que gestiona la asignación de orden a las casillas, usando un *setOnClickListener* que enlaza el botón a la función *botonAciertoOrdenAbece*, que se encargará de comprobar que el botón pulsado es el inmediatamente siguiente al anterior, utilizando como referencia la lista obtenida al crear la rejilla, y en caso contrario, aumenta el número de fallos.

	Tras configurar el botón con las letras adecuadas, la creación de rejilla continúa hasta completarla con todos los botones necesarios



Nombre	Iteración 3: 041 – Pantalla de configuración de distracciones (front-end)
Introducción	Pantalla de configuración de las distracciones disponibles en la aplicación
Objetivo	Configurar las distracciones antes de empezar una rejilla y activarlas o desactivarlas para que empiecen al iniciar la rejilla
Uso previsto	Se accede a esta pantalla desde el menú de creación de rejilla para configurar las distracciones antes de empezar si se requiere
Planteamiento de diseño	La pantalla está dividida en tres secciones: • Metrónomo • Música • Línea Cada sección está separada de la otra por una línea, y ofrecen las opciones de configuración de la distracción
Diseño de la solución	La pantalla contiene tres secciones diferenciadas y separadas por una línea divisora color azul oscuro. Estas secciones corresponden a cada una de las distracciones disponibles, y son: • **Metrónomo:** Configuración de la distracción de metrónomo

	○ **Activar metrónomo:** *Switch* que activa o desactiva el metrónomo ○ **Milisegundos:** *EditText* para introducir los milisegundos de latencia entre sonido y sonido del metrónomo. Este campo solo admite números como entrada • **Música:** Configuración de la distracción de música ○ **Activar música:** *Switch* que activa o desactiva la música ○ **Canción:** Se muestra el nombre de la canción actual ○ **Cambiar canción:** Botón que al pulsarlo muestra una pantalla de selección de archivo de audio para elegir la canción deseada del dispositivo

	• **Línea:** Configuración de la distracción de línea
	  ○ **Activar línea:** *Switch* que activa o desactiva la línea
	  ○ **Grosor:** *Spinner* que sirve para cambiar el grosor de la línea, tiene tres opciones:
	  ▪ Fina
	  ▪ Normal
	  ▪ Gruesa
	  ○ **Velocidad:** *Spinner* que sirve para cambiar la velocidad de movimiento de la línea, tiene tres opciones:
	  ▪ Rápido
	  ▪ Normal
	  ▪ Lento
	  ○ **Dirección:** *Spinner* que sirve para cambiar la dirección de movimiento de la línea, tiene tres opciones:
	  ▪ Vertical
	  ▪ Aleatorio
	  ▪ Horizontal

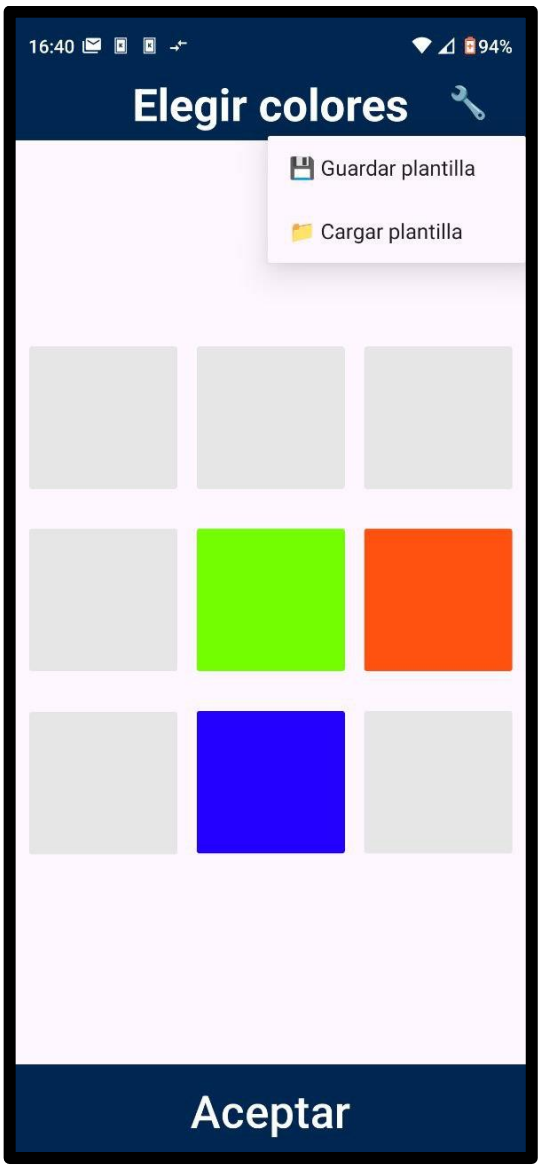
	○ **Color:** *TextView* que sirve a modo de panel para mostrar el color de la línea. Al pulsarlo abre el selector de color para seleccionar un nuevo color En la parte inferior de la pantalla hay un botón para aceptar los cambios realizados, que guarda estos en la configuración de la aplicación

Nombre	Iteración 3: 042 – Pantalla de configuración de distracciones (back-end)
Introducción	Funcionalidad de cada opción presente en la pantalla de configuración de distracciones
Objetivo	Dotar de funcionalidad cada Switch, Spinner y botón de la pantalla de configuración de distracciones
Uso previsto	Cuando se pulse el botón de aceptar, se aplican todos los cambios realizados en esta pantalla. Se usa también para abrir el menú de selección de colores y otras funciones en esta pantalla
Planteamiento de diseño	Se cargan los valores guardados al iniciar la pantalla para reflejar en esta la

	configuración actual. Tras pulsar el botón de aceptar se guardan en la configuración de la aplicación todos los parámetros modificados
Diseño de la solución	Al iniciar la actividad se cargan con un *SharedPreferences* todos los valores almacenados en la configuración general de la aplicación referentes a las distracciones. Si una opción aparece activada en la configuración, se refleja este cambio en la pantalla visualmente, actualizando el respectivo menú. Al pulsar aceptar, se recoge el estado de todas las opciones en la pantalla y se guardan en la configuración de la aplicación las opciones y los valores correspondientes usando un Editor. Se dota también de funcionalidad los siguientes menús: • **Milisegundos:** Se utiliza un filtro para admitir solo números • **Cambiar canción:** Al pulsar este botón se crea una actividad nueva con el selector de archivos

	de audio del dispositivo, y se espera a recibir un resultado de la actividad que contenga la localización del archivo seleccionado con *startActivityForResult*. Al acabar se recarga la pantalla considerando la nueva canción seleccionada, que sustituye la anterior en el *TextView* que muestra la canción actual. • **Grosor:** Sus opciones se cargan al iniciar la actividad. La opción guardada se carga como la opción por defecto • **Velocidad:** *S*us opciones se cargan al iniciar la actividad. La opción guardada se carga como la opción por defecto • **Dirección:** *S*us opciones se cargan al iniciar la actividad. La opción guardada se carga como la opción por defecto • **Cambiar color:** Al pulsar este *TextView* se crea una actividad nueva con el selector de color, y

	se espera a recibir un resultado de la actividad que contenga el color con *startActivityForResult*. Al acabar se recarga la pantalla considerando el nuevo color seleccionado, que sustituye al anterior en el *TextView*.



Nombre	Iteración 3: 043 – Pantalla de selección de colores de fondo (front-end)
Introducción	Interfaz de selección de colores que se usarán de fondo en las casillas de la rejilla cuando se active la opción de colores de fondo
Objetivo	Ofrecer una forma sencilla y rápida de seleccionar los colores a usar para la funcionalidad de colores de fondo
Uso previsto	Se accede a esta pantalla cuando se pulsa el botón de colores de fondo en el menú de creación de nueva rejilla, y se usa para seleccionar los colores de fondo a usar
Planteamiento de diseño	Se ofrecen nueve botones cuadrados a rellenar con el color elegido y un selector para la cantidad de botones correctos
Diseño de la solución	La pantalla consta de nueve botones cuadrados. Al pulsar cada uno de estos botones se abrirá un pequeño menú flotante con dos opciones: • **Borrar color:** Borra el color en la casilla • **Cambiar color:** Abre el selector de colores, y cuando se seleccione uno este cuadrado pasará a ser del color seleccionado

	Existe un botón con el icono de una llave inglesa en la parte superior derecha de la pantalla que, tras ser pulsado, muestra dos opciones: • **Guardar plantilla:** Abre el menú de selección de archivos para elegir la carpeta del dispositivo donde se guardará el archivo con la plantilla • **Cargar plantilla:** Abre el menú de selección de archivos para elegir el archivo del dispositivo con la plantilla guardada En la parte inferior de la pantalla está el botón de aceptar, que guarda los cambios

Nombre	Iteración 3: 044 – Pantalla de selección de colores de fondo (back-end)
Introducción	Maneja el funcionamiento de la pantalla de selección de colores de fondo
Objetivo	Dotar la pantalla de selección de colores de fondo de funcionalidad
Uso previsto	Se usa cada vez que se abre la pantalla de selección de colores de fondo, cargando los colores seleccionados anteriormente o haciendo que los botones tengan funcionalidad

Planteamiento de diseño	Al iniciar la actividad se cargan desde la configuración de la aplicación todos los colores guardados anteriormente en esta pantalla. Cada botón despliega, al ser pulsado, un menú flotante que, dependiendo de la opción seleccionada, abre un menú de selección de color o elimina el color de la casilla
Diseño de la solución	Al iniciar la actividad se utiliza un *SharedPreferences* para cargar todos los colores de fondo guardados anteriormente, y se pone cada uno de los nueve cuadrados con el color correspondiente. Si no había color guardado, el cuadrado permanece en el color por defecto, gris claro. Se configura un *PopupMenu* para los nueve cuadrados. Este menú contiene dos opciones al ser desplegado: • **Borrar color:** Borra el color en la casilla estableciéndolo al color por defecto, gris. También actualiza la configuración de la aplicación para reflejar el cambio

	• **Cambiar color:** Abre el selector de colores creando una nueva actividad a la que envía el id del botón correspondiente. Desde la actividad abierta se seleccionará el color deseado y se actualizará el botón en base a su id al nuevo color, y cuando se seleccione uno se volverá a esta pantalla y este cuadrado pasará a ser del color seleccionado

Nombre	Iteración 3: 045 – Gráfica de tiempo de botones (back-end)
Introducción	Este requisito consiste en la recogida de datos de la prueba de rejilla realizada para reflejar el tiempo tardado en pulsar cada botón en la interfaz de la gráfica
Objetivo	Rellenar la gráfica con datos obtenidos de la prueba realizada por el usuario para mostrarle los resultados en forma de gráfica
Uso previsto	Se accede a esta gráfica desde la pantalla de detalles de una prueba realizada, a la que se accede desde el historial de pruebas
Planteamiento de diseño	Al iniciar esta actividad se crea una nueva gráfica, se ajustan los parámetros para su visualización y se cargan desde la base de datos los valores de tiempo de los botones pulsados. Tras cargar estos valores, se reflejan en la gráfica.

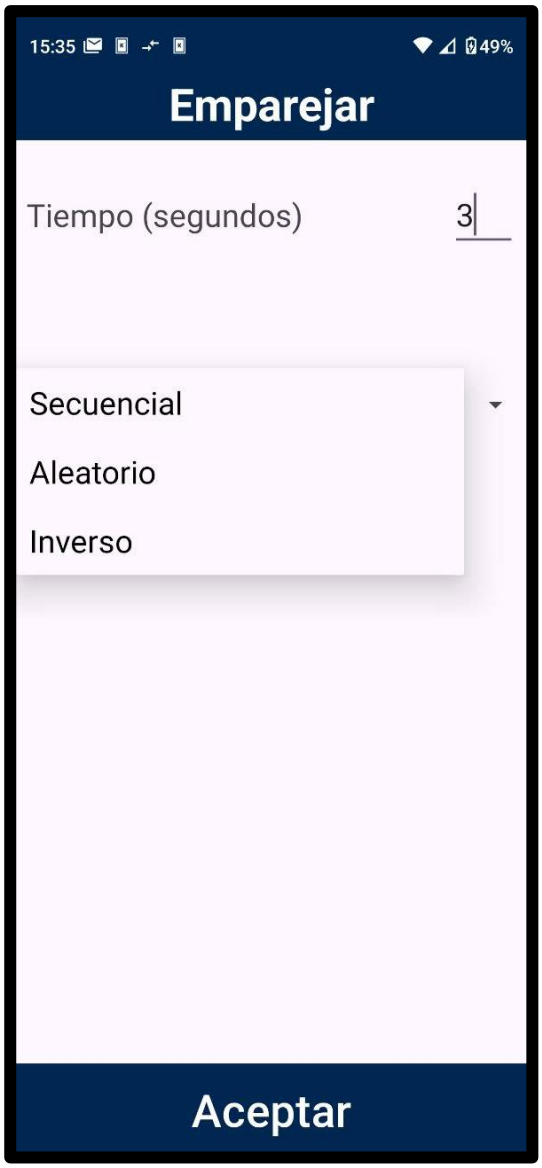
Diseño de la solución	Al iniciar la actividad se crea un objeto de tipo *LineChart* y se configuran los siguientes parámetros: • **Description:** Descripción de la gráfica "Tiempo por botón" • **XAxis:** Parámetros del eje X que refleja el número del botón • **YAxis:** Parámetros del eje Y que refleja el tiempo tardado en el botón • **Color:** Color de la línea. Azul oscuro por defecto • **LineWidth:** Grosor de la línea que une los puntos de la gráfica • **Data:** Lista de datos para la gráfica. Contiene el valor de tiempo de todos los botones Tras configurar todo, se escoge el tiempo máximo como valor máximo del eje Y de la gráfica y se cargan los datos Se ha usado para este requisito el proyecto Github: https://github.com/PhilJay/MPAndroidChart

Nombre	Iteración 3: 046 – Exportar resultados de la prueba
Introducción	Exportar resultados de la prueba a un archivo de texto en el dispositivo
Objetivo	Ofrecer al usuario una forma local, sin conexión, y fácil de almacenar los datos obtenidos en la prueba en un archivo de texto para poder ser enviado o consultado
Uso previsto	Para usar este requisito se dispone de un botón de exportar en la pantalla de vista de detalles de una prueba
Planteamiento de diseño	Al pulsar el botón de exportar se despliega la pantalla de selección de carpeta del dispositivo en la que se guardará el archivo.
Diseño de la solución	Se despliega una nueva actividad de selección de carpeta y, con un *onActivityResult* se espera la respuesta de la actividad que contiene la *url* de localización de la carpeta. Esta *url* se usa para guardar ahí el archivo. Para esto se hace uso de una clase *GestorBackups* que utiliza un *OutputStream* que, tras recibir la *url* de localización de la carpeta, escribe el archivo en la misma, con un nombre predeterminado con la fecha de la prueba. Este nombre puede ser

	cambiado por el usuario antes de guardar el archivo

Nombre	Iteración 3: 047 – Límite de tiempo rejilla
Introducción	Se dispone de una cuenta atrás con el tiempo límite de la prueba en la pantalla de realización de la rejilla
Objetivo	Mostrar al usuario el tiempo restante del que dispone para completar la rejilla que está haciendo
Uso previsto	Este requisito aparece cuando el usuario selecciona un tiempo límite para la rejilla en la pantalla de creación de rejilla, y es visible en la pantalla de la rejilla cuando esta se inicia
Planteamiento de diseño	Se usa un contador y se muestra la información en la barra superior de la aplicación en la pantalla de rejilla
Diseño de la solución	Al iniciar la rejilla, se comprueba si se ha establecido un tiempo límite, y si es el caso, se crea e inicia un contador con este tiempo límite usando la clase *CountDownTimer*. Tras esto, se configura

	el contador para recoger y actualizar la información del tiempo restante. Para mostrar la información en pantalla, se recogen los milisegundos del contador y se formatean usando la clase *FormatoTiempo,* que devuelve el tiempo formateado en minutos:segundos:milisegundos. El límite de tiempo se establece en el menú de creación de rejilla en segundos



Nombre	Iteración 3: 048 – Emparejar (front-end)
Introducción	Este requisito comprende la pantalla de configuración de la función de emparejar botones y la muestra del siguiente botón a pulsar en la pantalla de rejilla
Objetivo	Dar al usuario las opciones de configuración de la función de emparejar, para establecer el tiempo y orden. Así

	como mostrar en la pantalla de rejilla el siguiente botón a pulsar
Uso previsto	Se accede a esta pantalla desde el menú de creación de rejilla, pulsando el botón de emparejar. Se ve el resultado una vez se activa la función de emparejar y se inicia la rejilla. Este requisito solo es usado cuando la rejilla creada no es de tipo numérica o alfabética, ya que el orden de pulsación de estas ya está intrínsicamente definido por la configuración de estas
Planteamiento de diseño	La pantalla de configuración es simple, contiene únicamente dos opciones: tiempo y orden. Para mostrar el botón a pulsar se utiliza un pequeño cuadrado al lado de los errores y aciertos en la pantalla de rejilla
Diseño de la solución	La pantalla de configuración está dividida en dos: • **Tiempo (segundos):** *EditText* en el que se introduce el tiempo en segundos cada cuanto cambiará el botón a emparejar • **Orden:** *Spinner* para seleccionar el orden de iteración de los botones. Tiene tres opciones:

	○ Secuencial ○ Aleatorio ○ Inverso Al pulsar el botón de aceptar se guardan los cambios. La segunda parte del requisito consiste en la visualización del botón a emparejar en la rejilla. Este se sitúa en la esquina superior derecha de la rejilla y cambia al pulsar el botón correcto si no se ha fijado un tiempo en la configuración de la función emparejar, o cambia en caso contrario con un tiempo de espera seleccionado por el usuario

Nombre	Iteración 3: 049 – Emparejar (back-end)
Introducción	Este requisito comprende el funcionamiento de la pantalla de configuración de la función de emparejar botones y la muestra del siguiente botón a pulsar en la pantalla de rejilla
Objetivo	Establecer la funcionalidad de la pantalla de configuración de la función emparejar y manejar el funcionamiento de esta función durante la realización de la rejilla
Uso previsto	Se utiliza cuando se accede a la pantalla de configuración de la función emparejar

	o desde la rejilla para cambiar el botón según corresponda. Este requisito solo es usado cuando la rejilla creada no es de tipo numérica o alfabética, ya que el orden de pulsación de estas ya está intrínsicamente definido por la configuración de las mismas
Planteamiento de diseño	La pantalla de configuración consiste en un diseño simple que recoge los parámetros introducidos y los actualiza en la configuración de la aplicación. Para el botón emparejar en la rejilla, se compara el botón siguiente con el pulsado, y si es correcto, se cambia al siguiente en el orden seleccionado

Diseño de la solución	La pantalla de configuración tiene las siguientes opciones:
	• **Tiempo (segundos):** *EditText* en el que se introduce el tiempo en segundos. Está limitado a solo números
	
	• **Orden:** *Spinner* para seleccionar el orden de iteración de los botones. Tiene tres opciones:
	  ○ Secuencial
	  ○ Aleatorio
	  ○ Inverso

	Al pulsar el botón aceptar, se utiliza un *SharedPreferences* para guardar estos parámetros. En la pantalla de rejilla, el botón siguiente a pulsar estará mostrándose en pantalla hasta que se pulse el siguiente, y cuando esto pase, se comparará el botón pulsado recogiendo los valores contenidos en este, y comparándolo adecuadamente con el botón que había que pulsar. Al iniciar la rejilla, se comprueba si esta es de tipo numérico o alfabético, y en caso de serlo, la funcionalidad no se activa y se notifica al usuario, permitiendo seguir con la rejilla de forma normal. Para recoger los valores a pulsar, se itera sobre los valores elegidos en la creación de la rejilla como valores correctos, y a esos valores no nulos (teniendo como nulos aquellos cuadrados que no se han rellenado en la creación) se los considera correctos no nulos, y se almacenan en un *ArrayList* en el que se itera en el orden configurado por el usuario para mostrar el siguiente botón correcto a pulsar.

	La forma de procesar el valor de esta casilla y de comparar esta con el botón pulsado dependen del tipo de rejilla. Para las rejillas de letras y símbolos se hace una comprobación simple del carácter contenido en la casilla pulsada. Para la rejilla de color se compara el valor *Integer* del color Para la rejilla imagen se comparan dos Bitmap creados a partir de los *BitmapDrawable* que se obtienen de decodificar en base 64 las imágenes de los botones. Esta decodificación hace uso de la clase *CodificadorImagen.* La aplicación dispone de una lista con todos los botones a pulsar con repetición, es decir, contiene una lista de todas las casillas correctas de la rejilla, y cuando se pulsa una correcta, se elimina esta casilla de la lista. Esta lista es a la que se accede para verificar el siguiente botón a pulsar, por lo que, cuando se pulse este botón, no se volverá a mostrar. Por ejemplo, en una rejilla de color, si hay que pulsar colores amarillo, rojo y azul, en el momento en el que no se disponga de más colores azul en la rejilla,

	este dejará de aparecer para ser emparejado, así que solo se mostrarán los amarillos y rojos hasta que se agoten todos y se complete la rejilla

Nombre	Iteración 3: 050 – Emparejar con tiempo (back-end)
Introducción	Este requisito consiste en el control de la función de emparejar con tiempo, y explica como funciona esta en la pantalla de rejilla
Objetivo	Establecer la funcionalidad de la función emparejar con tiempo límite entre botones y manejar el funcionamiento de esta función durante la realización de la rejilla
Uso previsto	Se utiliza desde la rejilla en caso de haber establecido un tiempo en la configuración de emparejar para cambiar el botón según corresponda. Este requisito solo es usado cuando la rejilla creada no es de tipo numérica o alfabética, ya que el orden de pulsación de estas ya está intrínsicamente definido por la configuración de las mismas
Planteamiento de diseño	El botón emparejar en la rejilla se compara con el pulsado, y si es correcto, se cambia al siguiente en el orden seleccionado. Si es incorrecto o se acaba

	el tiempo para pulsarlo, el botón cambia al siguiente en el orden establecido
Diseño de la solución	En la pantalla de rejilla, el botón siguiente a pulsar estará mostrándose en pantalla hasta que se acabe el tiempo configurado. En caso de no haber tiempo configurado, se iniciará la función emparejar normal. Si hay un tiempo fijado, se iniciará el contador en segundo plano y se comenzará a manejar el tiempo de pulsación y los cambios de botón siguiente en paralelo. Al iniciar la rejilla, se comprueba si esta es de tipo numérico o alfabético, y en caso de serlo, la funcionalidad no se activa y se notifica al usuario, permitiendo seguir con la rejilla de forma normal. Para recoger los valores a pulsar, se itera sobre los valores elegidos en la creación de la rejilla como valores correctos, y a esos valores no nulos (teniendo como nulos aquellos cuadrados que no se han rellenado en la creación) se los considera correctos no nulos, y se almacenan en un *ArrayList* en el que se itera en el orden configurado por el usuario para mostrar el siguiente botón correcto a pulsar.

	La forma de procesar el valor de esta casilla y de comparar esta con el botón pulsado dependen del tipo de rejilla. Para las rejillas de letras y símbolos se hace una comprobación simple del carácter contenido en la casilla pulsada. Para la rejilla de color se compara el valor *Integer* del color Para la rejilla imagen se comparan dos *Bitmap* creados a partir de los *BitmapDrawable* que se obtienen de decodificar en base 64 las imágenes de los botones. Esta decodificación hace uso de la clase *CodificadorImagen* Para gestionar el tiempo en segundo plano se utiliza un objeto de la clase *Handler* con un *delay* que depende del tiempo en segundos establecido. Cuando se alcanza el tiempo límite, se llama a una función *CambiarEmparejarTimer*, que es usada para cambiar al siguiente botón y continuar la gestión del tiempo. Para asegurar la integridad de los datos accedidos en esta función se configura esta como *synchronized*, y se bloquea su

	acceso en el hilo que accede en cada momento para que no se pueda modificar hasta que se complete la tarea actual

Nombre	Iteración 3: 051 – Exportar ajustes
Introducción	Se dispone de un botón para guardar los ajustes actuales de la aplicación en un archivo en el dispositivo
Objetivo	Guardar todas las plantillas y configuraciones realizadas a la aplicación en un archivo en el dispositivo
Uso previsto	Se accede a este requisito desde la pantalla de configuración general de la aplicación, y se usa para hacer una copia de seguridad de las opciones configuradas en la aplicación
Planteamiento de diseño	Para exportar a un archivo las configuraciones, se lee de la configuración de la aplicación toda la información relevante, incluyendo plantillas, ajustes a distracciones y colores de fondo, entre otros
Diseño de la solución	Al pulsar el botón de guardar se crea un *JSONObject*. Tras esto, se utiliza un *SharedPreferences* y se añade al *JSONObject* cada una de las configuraciones relevantes en la aplicación.

	Al final se abre una actividad de selección de carpeta con *startActivityForResult* y se establece la carpeta recibida con un *onActivityResult* como la carpeta en la que se guarda en archivo *JSON* utilizando el método *exportarJson* de la clase *GestorBackups*, que utiliza un *OutputStream* para escribir este *JSON* a archivo. Los valores guardados son los siguientes: • **Datos de usuario** • **Plantilla selección de colores de la rejilla de color** • **Plantilla selección de símbolos de la rejilla de wingdings** • **Plantilla selección de imágenes de la rejilla de imágenes** • **Plantilla selección de letras de la rejilla de letras** • **Configuraciones distracciones** • **Configuraciones emparejar** • **Plantilla colores de fondo**

	• **Usuario MenPas** • **Contraseña MenPas**

Nombre	Iteración 3: 052 – Importar ajustes
Introducción	Se dispone de un botón para cargar los ajustes actuales de la aplicación desde un archivo en el dispositivo
Objetivo	Cargar todas las plantillas y configuraciones realizadas a la aplicación desde un archivo exportado previamente en el dispositivo desde la aplicación
Uso previsto	Se accede a este requisito desde la pantalla de configuración general de la aplicación, y se usa para cargar una copia de seguridad de las opciones configuradas en la aplicación
Planteamiento de diseño	Para importar desde un archivo las configuraciones, se despliega una actividad de selección de archivo y se lee la información del archivo para cargarla en la configuración de la aplicación
Diseño de la solución	Al pulsar el botón de abrir se crea un *JSONObject*. Tras esto, se abre una actividad de selección de archivo con *startActivityForResult* y se establece el archivo recibido con un *onActivityResult* como el archivo importado, que se usará

	para configurar la variable *JSON* creada anteriormente. Utilizando el método *importarJSON* de la clase *GestorBackups*, que utiliza un *BufferedReader* para leer el archivo recibido y cargarlo a un *JSON* que devuelve por parámetro. Tras esto, se utiliza un *SharedPreferences* y se cargan los valores del *JSONObject* uno a uno en base a los valores de configuración de la aplicación, usando un *Editor* para fijar al valor del *JSON* el valor de configuración correspondiente de la aplicación. Los valores cargados son los siguientes: • **Datos de usuario** • **Plantilla selección de colores de la rejilla de color** • **Plantilla selección de símbolos de la rejilla de wingdings** • **Plantilla selección de imágenes de la rejilla de imágenes** • **Plantilla selección de letras de la rejilla de letras**

	• **Configuraciones distracciones** • **Configuraciones emparejar** • **Plantilla colores de fondo** • **Usuario MenPas** • **Contraseña MenPas** Si el archivo cargado es incorrecto se informa con una notificación al usuario

Nombre	Iteración 3: 053 – Importar plantillas (todas las rejillas)
Introducción	Se dispone de la opción para importar plantilla en la pantalla de creación de rejillas de imágenes, colores, letras y wingdings
Objetivo	Dar al usuario la opción de guardar las configuraciones realizadas en cada uno de los tipos de rejilla que requieren la introducción de valores separados para cada botón correcto
Uso previsto	Se usa cuando se quiere importar desde un archivo la configuración de la plantilla de la pantalla actual
Planteamiento de diseño	Se utilizan archivos *JSON* para guardar la información, recogiendo uno a uno el

	parámetro de cada casilla de la plantilla actual
Diseño de la solución	Al pulsar el botón de cargar plantilla se crea un *JSONObject*. Tras esto, se abre una actividad de selección de archivo con *startActivityForResult* y se establece el archivo recibido con un *onActivityResult* como el archivo importado, que se usará para configurar la variable *JSON* creada anteriormente. Utilizando el método *importarJSON* de la clase *GestorBackups*, que utiliza un *BufferedReader* para leer el archivo recibido y cargarlo a un *JSON* que devuelve por parámetro. Tras esto, se utiliza un *SharedPreferences* y se cargan los valores del *JSONObject* uno a uno y se establece el parámetro de cada casilla (imagen, letra, símbolo o color). Tras esto, usando un *Editor,* se guarda en la configuración de la aplicación cada parámetro cargado. Si el archivo cargado es incorrecto se informa con una notificación al usuario

Nombre	Iteración 3: 054 – Exportar plantillas (todas las rejillas)

Introducción	Se dispone de la opción para exportar plantilla en la pantalla de creación de rejillas de imágenes, colores, letras y wingdings
Objetivo	Dar al usuario la opción de cargar las configuraciones realizadas en cada uno de los tipos de rejilla que requieren la introducción de valores separados para cada botón correcto
Uso previsto	Se usa cuando se quiere exportar a un archivo la configuración de la plantilla de la pantalla actual
Planteamiento de diseño	Se utilizan archivos *JSON* para guardar la información en un archivo, recogiendo uno a uno los parámetros de la plantilla actual y escribiéndolos al *JSON* para exportarlo posteriormente a un archivo
Diseño de la solución	Al pulsar el botón de guardar plantilla se crea un *JSONObject*. Tras esto, se utiliza un *SharedPreferences* y se añade al *JSONObject* cada una de las casillas en la plantilla. Al final se abre una actividad de selección de carpeta con *startActivityForResult* y se establece la carpeta recibida con un *onActivityResult* como la carpeta en la que se guarda en archivo *JSON* utilizando el método *exportarJson* de la clase *GestorBackups*,

	que utiliza un *OutputStream* para escribir este *JSON* a archivo.

5.10 Cuarta iteración

La cuarta iteración está enfocada en corrección de errores e implementar las funcionalidades de red, así como completar con algunas funcionalidades extra más pequeñas que faltaban por añadir.

Entre las funcionalidades más relevantes se encuentra la pantalla de inicio de sesión y su respectiva funcionalidad. En esta pantalla, el usuario tendrá que iniciar sesión con sus credenciales de MenPas para poder acceder al resto de la aplicación. Esta adición es un paso importante a la hora de ofrecer una experiencia de usuario acorde a los requisitos de la aplicación, siendo ahora la pantalla de inicio de sesión la que se mostrará justo al iniciar la aplicación, y siendo un paso fundamental el iniciar sesión para poder acceder a la pantalla principal de RejillaApp.

Además del inicio de sesión, se añade la posibilidad de subir resultados de una prueba a la plataforma MenPas utilizando un botón dedicado en la pantalla de detalles de una prueba.

Por último, se añade una función importante en la aplicación que da al usuario una nueva opción a la hora de realizar una rejilla: la aleatorización.
Con esta nueva función, el usuario puede aleatorizar en un tiempo determinado las casillas de una rejilla para poner a prueba su atención.

A continuación, se detallan los requisitos de esta última iteración del desarrollo:

Nombre	Iteración 4: 055 – Subir resultados

Introducción	Se dispone de un botón para subir los resultados a MenPas en la pantalla de detalles de la prueba
Objetivo	Subir los resultados de la rejilla a la página web de MenPas
Uso previsto	Este requisito se usa cuando se pulsa el botón de subir resultados desde la página de detalles de una prueba
Planteamiento de diseño	Se utiliza *ksoap* para manejar la conexión con la web. Al pulsar el botón de subir resultados se almacenan los resultados en una lista, que se pasa como parámetro a la clase responsable de hacer la subida a la web
Diseño de la solución	Se dispone de una clase *SoapConexion*, que gestiona las conexiones *Soap* con la página web. Al subir un resultado se llama a la función *subirResultados*, pasando como parámetro la lista de resultados. Esta función configura la conexión *Soap* y hace la llamada correspondiente para subir los datos a la web

Nombre	Iteración 4: 056 – Login (front-end)
Introducción	Se dispone de una pantalla de inicio de sesión al iniciar la aplicación. Para entrar a usar la aplicación el usuario tiene que iniciar sesión en esta correctamente
Objetivo	Ofrecer al usuario una forma visual e intuitiva de iniciar sesión, bloqueando el acceso a la aplicación si este no inicia sesión

Uso previsto	Se usa cada vez que se entra a la aplicación. En caso de entrar por primera vez, se usará para introducir los parámetros e iniciar sesión. A partir de lo cual se quedarán guardados las credenciales para la próxima vez
Planteamiento de diseño	El diseño consiste en una pantalla con fondo del color del tema principal de la aplicación (azul oscuro), el icono de la aplicación, campos de texto para el usuario y la contraseña, un botón de registro y un botón para iniciar sesión
Diseño de la solución	Los campos de texto usan *hints* para señalar cuál es para el usuario y cuál para la contraseña. El icono de la aplicación se muestra en la parte superior. El fondo es azul oscuro, como el tema principal de la aplicación, y todos los botones y campos tienen fondo transparente, es decir, solo se ven las letras blancas, dando un aspecto moderno y visualmente atractivo. El botón de inicio de sesión se sitúa en la parte inferior, y sobre él se encuentra en botón de registro algo más pequeño, que enlaza a la web de MenPas

	Si las credenciales son incorrectas se notifica al usuario y no se inicia sesión

Nombre	Iteración 4: 057 – Login (back-end)
Introducción	Funcionalidad de la pantalla de login, llamadas al servicio web, y confirmación de inicio de sesión
Objetivo	Comprobar que las credenciales son correctas y dotar de funcionalidad las opciones presentes en la pantalla
Uso previsto	Se comprueba si el usuario y contraseña introducidos son correctos cada vez que se pulsa el botón de iniciar sesión. Esta pantalla se muestra al iniciar la aplicación.
Planteamiento de diseño	Se utiliza *ksoap* para manejar la conexión con la web. Al pulsar el botón de iniciar sesión se hace una llamada a la web con los parámetros introducidos para verificar si es correcto
Diseño de la solución	Se dispone de una clase *SoapConexion*, que gestiona las conexiones *Soap* con la página web. Al iniciar sesión primero se almacenan los valores de usuario y contraseña en la configuración de la aplicación con un

	Editor de un *SharedPreferences*, y tras esto se invoca una función llamada *loginCorrecto*, que recoge el usuario y contraseña introducidos y llama con un *SoapObject* al servicio web, pasando como parámetros el usuario y la contraseña. El servicio devuelve un *Boolean* con la confirmación de si las credenciales son correctas o no. Si son correctas, se inicia sesión, dando paso a la pantalla de inicio de la aplicación. Si no lo son, se notifica al usuario

Nombre	Iteración 4: 058 – Aleatorizar
Introducción	Este requisito alade una función de aleatorización a la rejilla en la aplicación
Objetivo	Cambiar de sitio aleatoriamente todas las casillas de la rejilla con un intervalo de tiempo seleccionado mientras el usuario está completándola
Uso previsto	Se usa en la pantalla de rejilla tras haber activado la función introduciendo un tiempo en la opción de aleatorizar en la pantalla de creación de rejilla
Planteamiento de diseño	Al crear una rejilla se recoge el valor de tiempo de aleatorizar, y si es superior a cero se empieza a aleatorizar la rejilla

	con el tiempo dado utilizando un contador
Diseño de la solución	Si la comprobación de aleatorizar al crear la rejilla es afirmativa, se crea un *handler* con el tiempo dado que ejecuta la función aleatorizar con ese intervalo. La función que aleatoriza la rejilla utiliza elementos de tipo *Button*, *TableRow* y *TableLayout* para extraer y reconstruir los elementos de la rejilla conservando sus propiedades. Consiste en los siguientes pasos: • **Recoger todas las filas de la rejilla** • **Extraer los botones de cada fila** • **Aleatorizar la lista de botones** • **Limpiar los parámetros de las filas y la tabla** • **Insertar los botones en filas de nuevo** • **Insertar las filas en la tabla** Esto permite que se conserven los botones pulsados y sus funcionalidades sin perder nada.

	Tras aleatorizar, se espera el tiempo seleccionado y se vuelve a ejecutar la aleatorización. Este bucle se repite indefinidamente hasta completar la rejilla o salir

Nombre	Iteración 4: 059 – Conexión con WS
Introducción	Gestión de conexiones con el Web Service de MenPas
Objetivo	Establecer conexiones con el WS para realizar diversos intercambios de información
Uso previsto	Este requisito se usa para conectar con el WS al subir los resultados y almacenarlos en el servicio o para comprobar si el *login* es correcto
Planteamiento de diseño	Se utiliza *ksoap* para manejar la conexión con el servicio web. Se dispone de dos métodos que gestionan todo esto utilizando *requests* al servicio
Diseño de la solución	Se dispone de una clase *SoapConexion*, que gestiona las conexiones *Soap* con el servicio web. Al subir un resultado se llama a la función *subirResultados*, pasando como parámetro la lista de resultados. Al hacer *login* se llama a la función loginCorrecto, pasando como parámetro el usuario y la contraseña. Estas funciones configuran la conexión *Soap* y hacen la llamada correspondiente para subir los datos a la web.

	Se utilizan los parámetros: • **namespace:** http://tempuri.org/ • **url:** http://150.214.108.138/menpas/ServiceApp.asmx • **método:** LogUser o AddRejillaValores, dependiendo de lo que se desee hacer • **AccionSoap:** http://tempuri.org/[método] Y los datos que se envían son: • **ArrayList<String> valores:** En caso de querer subir los valores de la rejilla al servicio • **String user, String password:** Para comprobar el *login* Primero se crea un *SoapObject request,* al que luego se añaden las propiedadescon las variables utilizando *addProperty*. Se configura el *request* para utilizar el *namespace* y métodos adecuados, y se establecen los mecanismos para recibir la respuesta del servidor, que indicará si se ha realizado correctamente la conexión o no, almacenando en un *Object* el resultado del servidor, que será de tipo *Boolean*. Por último, se configura un *SoapSerializationEnvelope* con el *request* y parámetros correctos, y se hace un

	getResponse a este para realizar la conexión y obtener el resultado del servidor.

Nombre	Iteración 4: 060 – Almacenamiento en MenPas
Introducción	Almacenamiento de los datos de la rejilla en el servicio web de MenPas
Objetivo	Subir los resultados de la rejilla a la página web de MenPas para su almacenamiento de forma correcta
Uso previsto	Este requisito se usa cuando se pulsa el botón de subir resultados desde la página de detalles de una prueba
Planteamiento de diseño	Se utiliza *ksoap* para manejar la conexión con la web. Se dispone de un método para configurar la conexión y gestionar el envío de datos
Diseño de la solución	Se dispone de un método subirResultado en la clase *SoapConexion*, que gestiona las conexiones *Soap* con la página web para almacenar los resultados en el servicio web. El método recibe como parámetro un *ArrayList<String>* formado por treinta valores obtenidos de la rejilla y que se subirán al servicio web.

	Se utilizará el método del servicio web *AddRejillaValores* que recibe una lista de *String* y devuelve un *Boolean*. Los valores dentro del *ArrayList* y sus respectivas posiciones son: • **valores[0] … Nombre_Usuario** • **valores[1] … Tipo_Rejilla** • **valores[2] … Aciertos** • **valores[3] … Fallos** • **valores[4] … Filas** • **valores[5] … Columnas** • **valores[6] … Tiempo** • **valores[7] … Media** • **valores[8] … Varianza** • **valores[9] … T_Corregido** • **valores[10] … Correccion** • **valores[11] … TTpreliminar**

	• **valores[12] … MediaCorreccion** • **valores[13] … VarCorreccion** • **valores[14] … TareaPreliminar** • **valores[15] … Colores_Fondo** • **valores[16] … Emparejar** • **valores[17] … Aleatorio** • **valores[18] … Limite** • **valores[19] … incremento** • **valores[20] … ordenTachado** • **valores[21] … pAciertosABS** • **valores[22] … pAciertosREL** • **valores[23] … pErroresREL** • **valores[24] … pErroresABS** • **valores[25] … pEficacia** • **valores[26] … pEfectividad**

	• **valores[27] … Tipo_Distraccion** • **valores[28] … Idioma** • **valores[29] … Fecha** Al recibir este *ArrayList*, el método configura la conexión como se indica en el requisito 59 y se realiza el almacenamiento. Se obtendrá como resultado del servicio web *True* si se ha almacenado correctamente o *False* si ha habido algún error

6

Conclusiones y líneas futuras

La conclusión de este trabajo se estructura en torno a varios aspectos fundamentales que reflejan el desarrollo y los resultados alcanzados con la aplicación RejillaApp.

En primer lugar, se destaca el trabajo anterior en el que se basa el proyecto, una aplicación de escritorio para Windows que fue realizada como proyecto de fin de grado hace unos años. El objetivo principal de este proyecto era ofrecer una versión renovada y adaptada a móviles de esta, añadiendo diversas funcionalidades extra que son de utilidad actualmente. A destacar, la implementación de un sistema de usuarios con *login* y la posibilidad de subir los resultados a la web entre otros. Así mismo, se han eliminado funciones obsoletas o que no encontrarían utilidad en dispositivos móviles, como puede ser la impresión en papel de los resultados, considerando más práctica la subida de estos a la web.

Tras toda la reestructuración de interfaz, funcionalidades y requisitos, lo cual llevó un trabajo bastante notable, se ha conseguido obtener una interfaz realmente amigable con el usuario de móviles y sin sacrificar ninguna funcionalidad importante en el proceso, obteniendo una aplicación bastante competente que no presentaría ningún problema de ser utilizada a nivel público por usuarios o psicólogos.
Para el desarrollo de una interfaz que resulte amigable y fácil de usar, se ha recurrido con frecuencia a conocimientos adquiridos en la asignatura de Interfaces de Usuario, que, si bien no trata explícitamente a fondo las interfaces de usuario para dispositivos móviles, ofrece nociones básicas de interfaces de usuario en general.

En cuanto al desarrollo del código, ha sido posible en gran parte gracias a conocimientos adquiridos en Fundamentos de la Programación, y especialmente, Programación Orientada a Objetos. Durante la carrera, se utiliza Java para programar en diversas asignaturas, sin embargo, la asignatura por excelencia para aprender este lenguaje ha sido Programación Orientada a Objetos sin duda.

Para el desarrollo y uso de la base de datos, como no podía ser menos, se ha recurrido a conocimientos de la asignatura de Bases de Datos, que sin importar el lenguaje que se use, la tecnología, o la plataforma, siempre es útil para aprender a gestionar datos permanentes.

Las dificultades encontradas durante su desarrollo, por otro lado, no han sido pocas, y requirieron soluciones creativas y técnicas para poder completar la aplicación apropiadamente.

Para empezar, se han tenido que exprimir bastantes de las funcionalidades de Android, siempre teniendo en cuenta mantener una compatibilidad lo más alta posible con dispositivos antiguos, lo que ha presentado una serie de retos.
La asignatura de Software para Sistemas Empotrados y Dispositivos Móviles ha sido extremadamente útil para superar fácilmente la primera barrera de entrada a la

programación en dispositivos móviles Android utilizando Android Studio, sin embargo, en esta asignatura no se profundiza en ello, por tanto, de ahí en adelante se ha dedicado bastante tiempo a estudiar las tecnologías Android y el funcionamiento general de Android Studio y las actividades de Android.
Como solución a estas dificultades, la comunidad en línea de desarrolladores de Android y Java ha demostrado ser un recurso de apoyo extremadamente valioso.

Otra de las dificultades más notables encontradas ha sido la gestión del tiempo, dado que, aunque no lo parezca a simple vista debido a la filosofía de diseño de interfaz de Android, la aplicación cuenta con muchas funcionalidades y opciones, algunas de ellas no precisamente fáciles de implementar, a destacar la gestión de animaciones o aleatorización en segundo plano durante la realización de una rejilla. Por todo ello, para terminar en la fecha acordada, se ha tenido que dedicar muchísimo tiempo a la implementación de estas funcionalidades a la vez que se aprendía Android casi desde cero, lo cual ha sido sin duda un reto que, por si fuera poco, va sumado a la propia redacción de esta memoria, que acabó siendo más extensa de lo planeado. El motivo para esta extensión es, como se ha mencionado, las muchas funcionalidades de la aplicación, que han resultado en una cantidad enorme de requisitos plasmados en este documento, y un manual también bastante extenso, como se puede comprobar.

Al final, y pese a estas dificultades, se ha obtenido una aplicación con el potencial de tener una gran utilidad en el campo de la evaluación de la atención. Sus características permiten una evaluación precisa y accesible, ofreciendo una herramienta valiosa tanto para profesionales en psicología como para el usuario promedio. La facilidad de uso y la precisión de los datos recolectados consolidan su relevancia y aplicabilidad práctica.

En términos de mejoras futuras, se identifican varias áreas de desarrollo potencial. La integración de técnicas de inteligencia artificial podría mejorar la personalización y precisión de las evaluaciones. Sería interesante plantear un sistema de recogida de resultados de las rejillas que utilice algún tipo de algoritmo que aprenda de los patrones

del usuario y evalúe si está progresando adecuadamente o no, o con que distracción se distrae más.
Además, la expansión de la base de datos y la inclusión de un mayor número de parámetros podrían proporcionar datos más robustos para realizar los análisis.

Otro proyecto interesante podría ser realizar una aplicación con más componentes en línea, como por ejemplo una tabla de resultados mundiales que permita calcular si el usuario está en la media, por encima, o por debajo, o incluso filtrar según ciertos parámetros. Esto requeriría un trabajo exhaustivo de adaptación de la aplicación a un entorno completamente online y una gestión de servidores más compleja que lo planteado en este proyecto.

En conclusión, el proyecto de desarrollo de una aplicación para evaluar la atención se basa en un sólido trabajo previo, ha superado diversas dificultades técnicas y metodológicas, y tiene el potencial de ser una herramienta útil y aplicable en múltiples contextos. Puede ser útil para ayudar a muchas personas con problemas de atención o similares, y puede servir a profesionales para tratar estos casos con la mayor precisión posible.

Apéndice A

Manual de Instalación

En este apéndice se muestra como instalar la aplicación paso a paso en el dispositivo del usuario. Para instalar la aplicación en un dispositivo Android, hay que cumplir una serie de requisitos hardware y software que se especificaran en esta sección.

A1. Requerimientos previos

RejillaApp es una aplicación desarrollada para Android, en concreto se ha tenido como objetivo principal la compatibilidad con versiones Android superiores a Android 5.1.1 inclusive.

RejillaApp debería funcionar correctamente en cualquier dispositivo Android que cumpla las siguientes cualidades:

- **Android 5.1.1 o superior**
- **Pantalla con ratio alargado (con ratio de 20:9 o más alargado)**
- **Internet para poder iniciar sesión**
- **Recomendado el tema claro**

Cabe mencionar que el ecosistema Android es muy diverso, dando pie a una cantidad enorme de variaciones del sistema que dependen de la marca del fabricante del dispositivo. Algunos fabricantes pueden aplicar capas de personalización al sistema que hagan que la aplicación funcione de formas imprevisibles, especialmente dependiendo de la aplicación del tema oscuro de cada capa de personalización, por ello se recomienda usar un tema claro.

La aplicación ha sido probada en dispositivos Android One y stock, que son versiones de Android lo más cercanas posible al proyecto AOSP (Android Open Source Project), donde se ha comprobado una compatibilidad mayor y un mejor funcionamiento general de la aplicación.

A continuación se detalla como instalar la aplicación a partir de un archivo APK que se encuentra previamente descargado en un dispositivo con Android 13. Los pasos pueden variar en otras versiones, especialmente en lo referente a permisos de instalación, pero los pasos son, a excepción de algunos pasos que no serán necesarios en ciertas versiones, los especificados en el siguiente apartado.

A2. Instalación

Para instalar la aplicación habrá que seguir en orden los siguientes pasos.

Para instalar la aplicación lo primero será localizar el archivo APK en el dispositivo. Para ello se usará el gestor de archivos de Android o el que el usuario tenga en su dispositivo.

Figura 1.A: RejillaApp.apk en el explorador de archivos “Material Files”

Tras localizar el archivo, tan solo habrá que pulsar sobre él, y comúnnmente, el explorador de archivos preguntará al usuario que desea hacer con la aplicación.

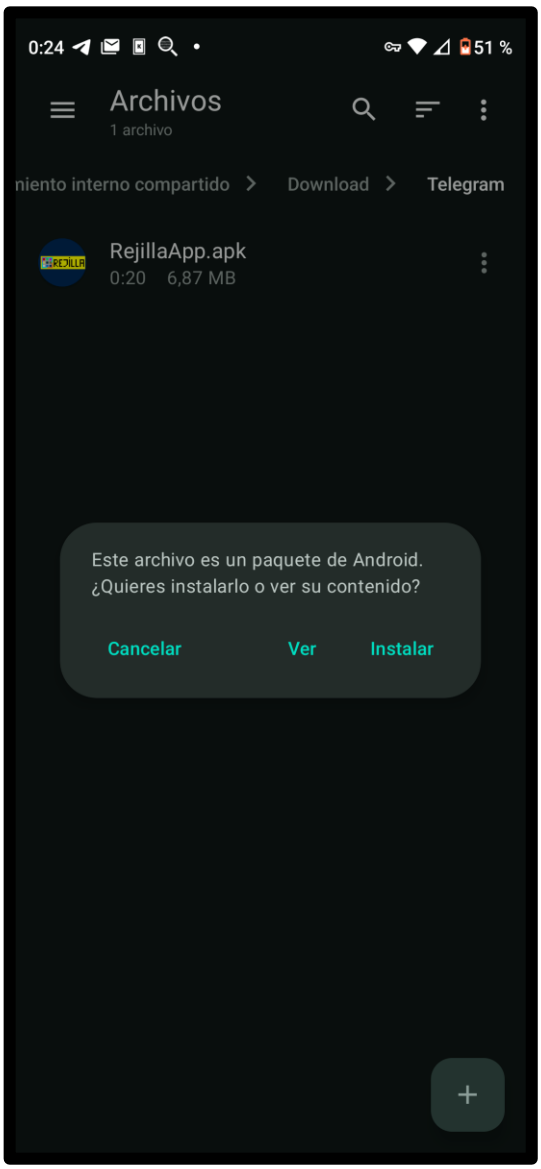


Figura 2.A: Opciones al abrir el archivo RejillaApp.apk

En este diálogo habrá que pulsar “Instalar”, a lo cual, en caso de no haberlo hecho aún, Android nos dará la advertencia de que se va a instalar una aplicación desde el explorador de archivos, para lo cual se requiere el permiso del usuario. Tras esto preguntará al usuario qué desea hacer.

Figura 3.A: Android pregunta al usuario si quiere permitir la instalación

Cuando Android pregunte si se desean cambiar los ajustes para permitir la instalación de aplicaciones desconocidas, habrá que pulsar “Ajustes” y permitir al explorador de archivos la instalación.

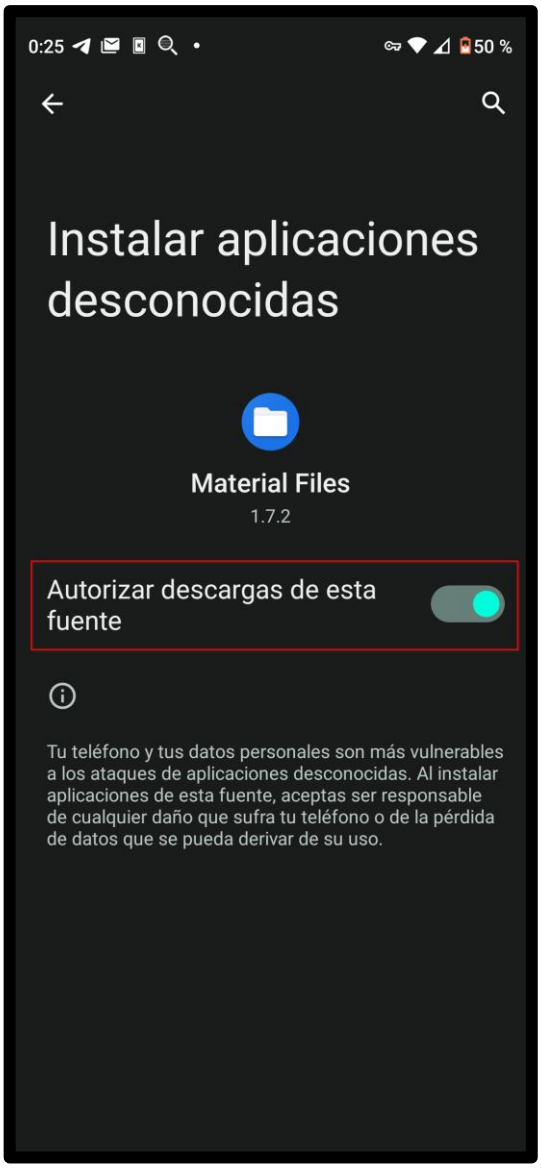


Figura 4.A: Pantalla de ajustes de instalación de aplicaciones desconocidas. En rojo marcada la opción que hay que activar

En la pantalla de ajustes habrá que marcar la opción de “Autorizar descargas de esta fuente”, o su equivalente en cualquier otro sistema Android.

Tras eso, se dará la opción de instalar directamente la aplicación.

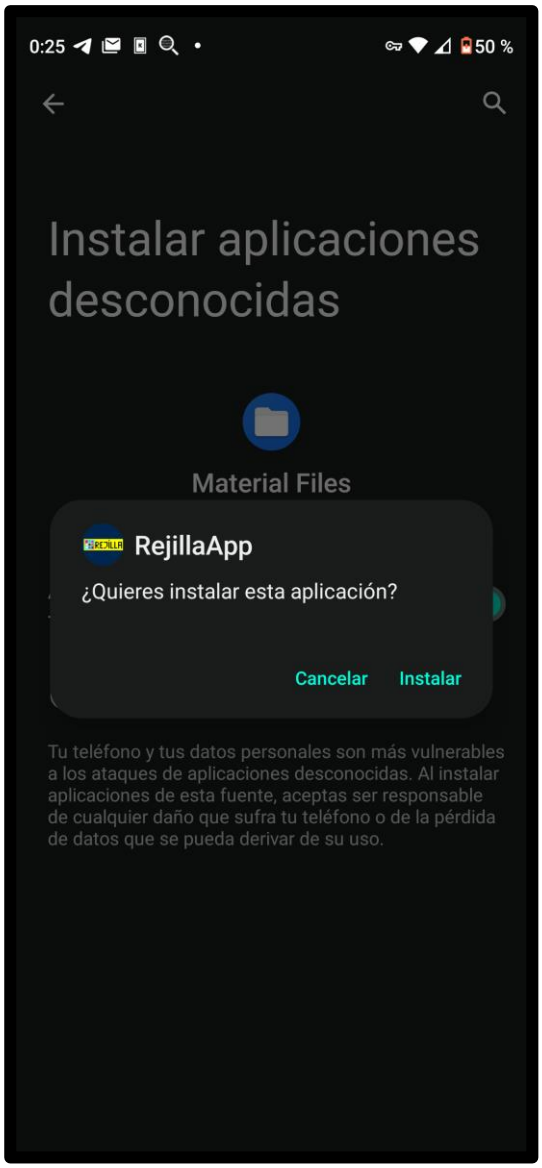


Figura 5.A: Pantalla de ajustes de instalación de aplicaciones desconocidas. En rojo marcada la opción que hay que activar

En el diálogo flotante que se muestra en la figura habrá que pulsar “Instalar” de nuevo, ahora sí, instalando la aplicación en el dispositivo.

En algunas versiones de Android, tras activar esta opción, habrá que ir hacia atrás y volver a pulsar el archivo de la aplicación para que muestre efectivamente el menú de instalación funcional.

Tras esto la aplicación aparecerá en el menú de aplicaciones del dispositivo y será totalmente funcional. El usuario solo debe pulsar sobre el icono y esta se abrirá automáticamente.

Apéndice B

Manual de usuario

En este manual de usuario se ofrecen explicaciones de cada una de las funcionalidades de la aplicación, proporcionando al usuario una guía para utilizarlas de forma correcta. En este apéndice se explican de forma lineal todas las características del sistema en el orden que se considera el apropiado para un uso correcto de la misma.

B1. Abrir la aplicación

Al instalar la aplicación aparecerá el icono de la aplicación en el menú de aplicaciones de Android.

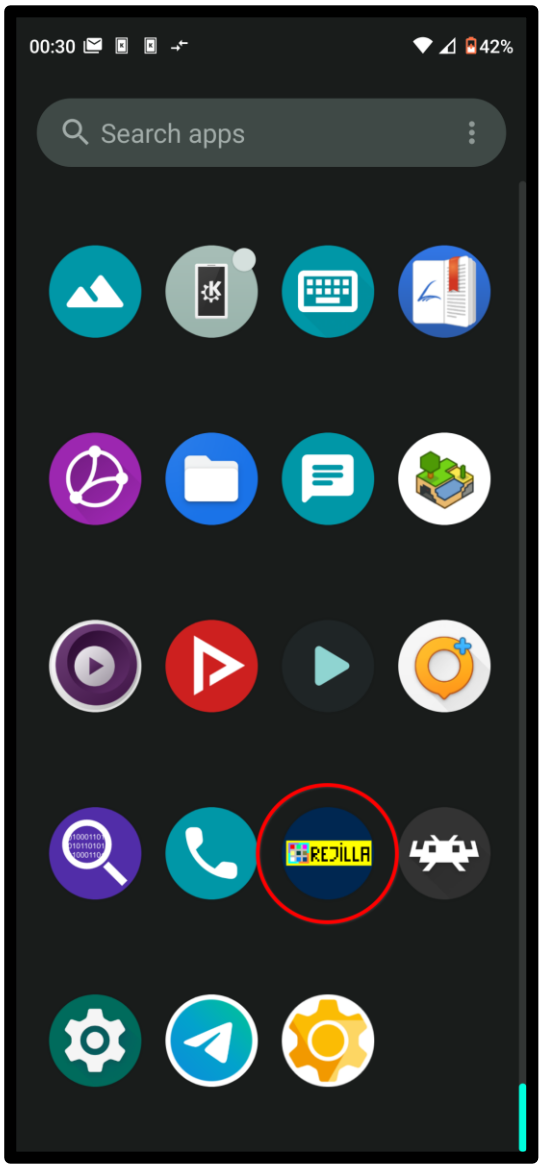


Figura 1.B: RejillaApp en el menú de aplicaciones de Android

Al pulsar sobre el icono de la RejillaApp se abrirá la aplicación. Lo primero que aparece es la pantalla de inicio de sesión.

B2. Inicio de sesión

Al iniciar la aplicación es necesario iniciar sesión con la cuenta de MenPas. Por lo tanto, la primera actividad que aparece al iniciar la aplicación es la pantalla de inicio de sesión, que permite introducir las credenciales e iniciar sesión para continuar usando la aplicación.

Figura 2.B: Pantalla de inicio de sesión

Es necesario tener una cuenta registrada en MenPas para iniciar sesión en la aplicación.

Se dispone de una opción de registro. Al pulsar este botón se abrirá en el navegador de internet la página web de registro de MenPas.

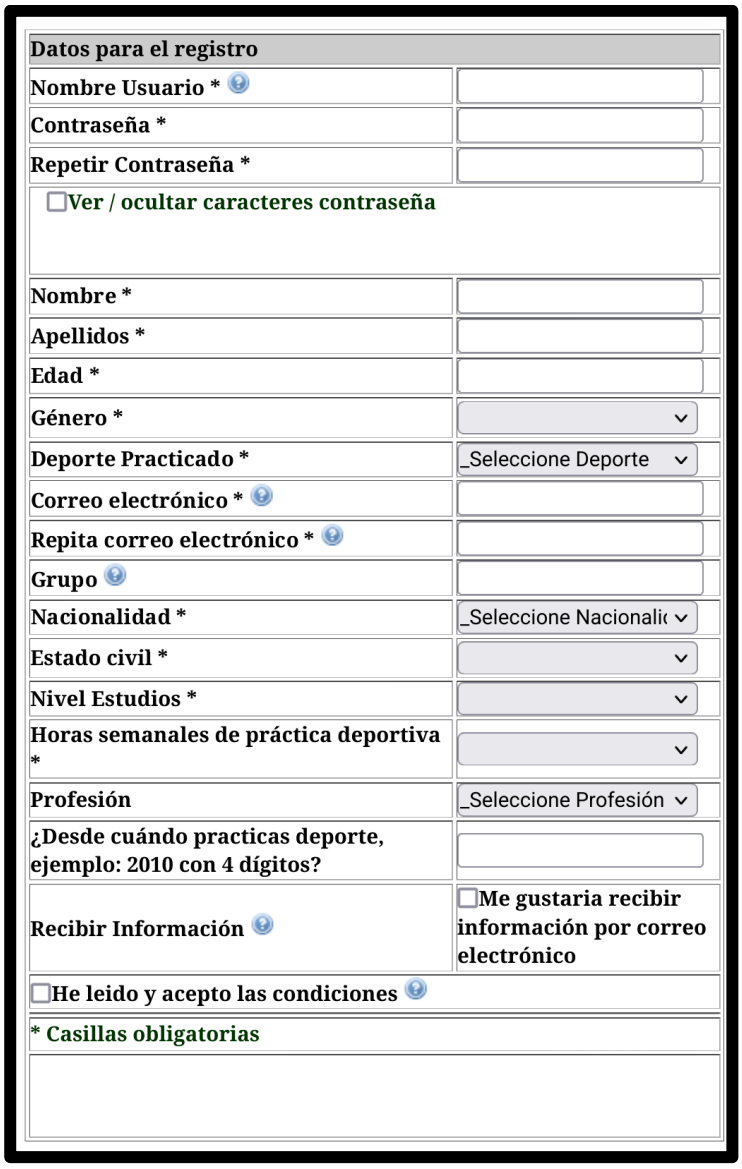


Figura 3.B: Formulario de registro en MenPas

Tras registrarse, el usuario puede iniciar sesión con las credenciales de MenPas en la aplicación, lo que abrirá la pantalla de inicio de la aplicación.

B3. Pantalla de inicio

En la pantalla de inicio se encuentran varios botones.

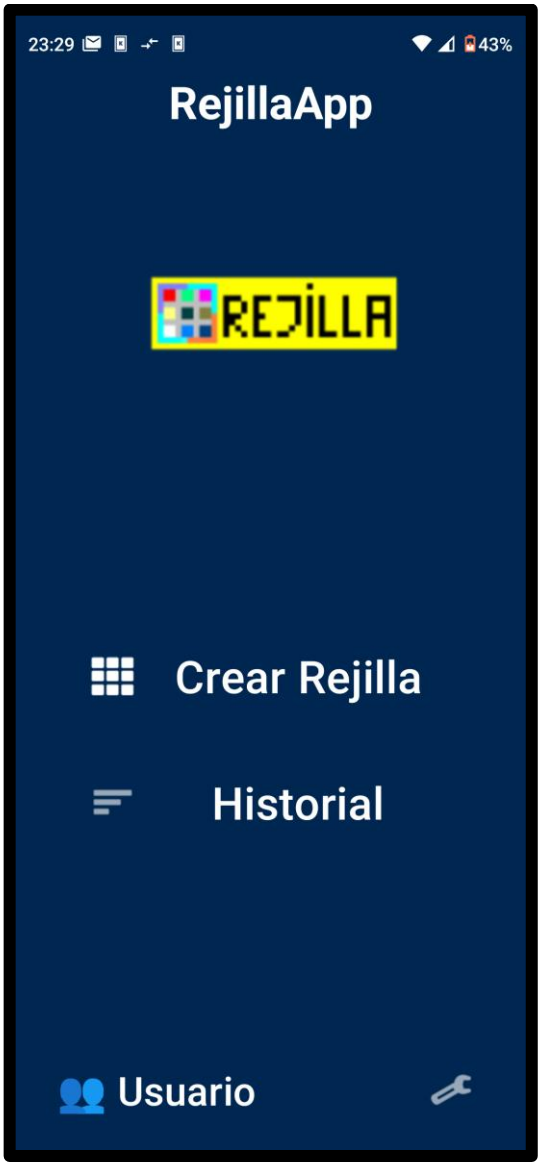


Figura 4.B: Pantalla de inicio de la aplicación

- **Crear Rejilla:** Abre la pantalla de creación de rejillas.

- **Historial:** Lista de resultados de pruebas realizadas.

- **Usuario:** Formulario de datos del usuario.

- **Opciones:** Incono de llave inglesa en la esquina inferior derecha. Abre la pantalla de opciones.

B4. Formulario de datos de usuario

En esta pantalla el usuario puede introducir sus datos para que aparezcan reflejados en los resultados de las rejillas que complete.

La introducción de estos datos es optativa, y aparecen vacíos en las pruebas en caso de no estar cumplimentados.

Figura 5.B: Formulario de información de usuario

- **Nombre:** Nombre del usuario. Aparecerá en el título de la prueba en la pantalla de historial junto a la fecha de realización.

- **Apellidos:** Apellidos del usuario.

- **Edad:** Edad del usuario.

- **Género:** Género del usuario.

- **País:** País del usuario.

- **Fecha:** Fecha actual.

- **Deporte:** Deporte practicado por el usuario.

- **Posición:** Posición que el usuario tiene en este deporte.

- **Lateralidad:** Selector de la lateralidad del usuario. Tiene dos opciones: Izquierda y Derecha.

Tras pulsar el botón de Aceptar, estos datos se guardan a partir de ese momento y se usan en todas las pruebas, pudiendo ser cambiados en cualquier momento desde esta misma pantalla.

B5. Creación de rejillas

Al pulsar el botón Crear Rejilla en la pantalla de inicio se abre el menú de creación de rejillas.

Las rejillas son el elemento principal de la aplicación RejillasApp, y consisten en una tabla de entre cuatro y cien casillas configurable por el usuario, y donde cada casilla es un elemento que el usuario debe identificar y pulsar en un orden concreto o con unos requisitos específicos.

El tiempo tardado en pulsar cada casilla y otros parámetros se usarán para generar un informe que pueda ser evaluado por un psicólogo para determinar la capacidad de atención del usuario.

Esta aplicación también puede ser usada por el usuario para entrenar su atención o comprobar él mismo si esta está mejorando con el paso del tiempo.

Para crear una rejilla que el usuario pueda usar, se deben configurar antes sus características y opciones. En esta pantalla se pueden configurar diversos parámetros para crear una rejilla.

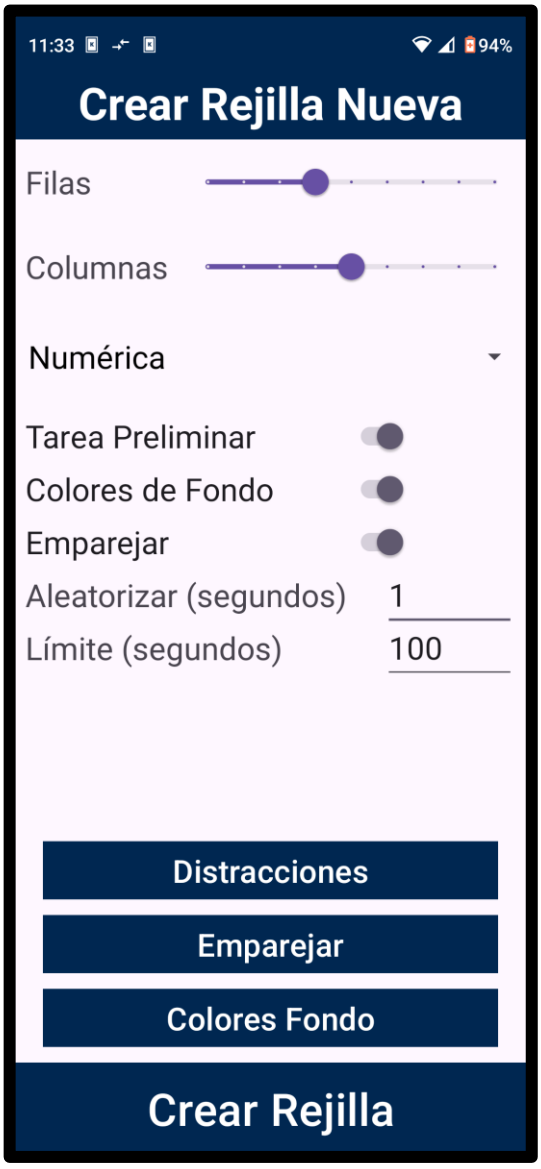


Figura 6.B: Pantalla de creación de rejilla

- **Filas:** Selector de la cantidad de filas de la rejilla.

- **Columnas:** Selector de la cantidad de columnas de la rejilla.

- **Tipo de rejilla:** Lista de tipos de rejilla donde se puede seleccionar una entre las seis disponibles:
    - **Numérica:** Rejilla formada por números que habrá que pulsar en orden numérico incremental o inverso.
    - **Colores:** Rejilla formada por colores elegidos por el usuario, donde una serie de colores serán los correctos a pulsar por este.
    - **Letras:** Rejilla formada por letras elegidas por el usuario, donde una serie de letras serán las correctas a pulsar por este.
    - **Imágenes:** Rejilla formada por imágenes elegidas por el usuario, donde una serie de imágenes serán las correctas a pulsar por este.
    - **Abecedario:** Rejilla formada por series de tres letras, mayúsculas, minúsculas o variadas, que se deberán pulsar en orden incremental o inverso alfabéticamente.
    - **Wingdings:** Rejilla formada por símbolos de la fuente wingdings elegidos por el usuario, donde una serie de símbolos serán los correctos a pulsar por este.

- **Tarea preliminar:** La tarea preliminar consiste en una rejilla vacía que solo contiene dos botones: uno en la esquina superior izquierda y otro en la esquina inferior derecha.

  El usuario tendrá que pulsar ambos botones en el menor tiempo posible, y ese tiempo se usará como medida de corrección en las medidas de la prueba realizada.

  Al acabar esta tarea inicia automáticamente la rejilla.

- **Colores de fondo:** Colores que aparecerán en el fondo de una casilla que no sea de colores ni de imágenes.

  El botón de selección de colores de fondo permite al usuario acceder a la pantalla para elegir que colores se usarán aleatoriamente en las casillas de la rejilla.

- **Emparejar:** La función emparejar se utiliza para indicar al usuario que botón debe pulsar en la rejilla.
  Esta opción se desactiva para las rejillas Abecedario y Numérica.
  El botón de configuración de emparejar permite al usuario acceder a la pantalla para elegir las opciones de configuración de esta función.

- **Aleatorizar:** La función aleatorizar se usa para cambiar aleatoriamente de posición todas las casillas de una rejilla de forma cíclica con un intervalo de tiempo indicado en el campo de texto de esta opción.
  Si no se introduce ningún número o el número es 0, esta función quedará desactivada automáticamente.

- **Límite:** El usuario puede introducir un límite de tiempo para la rejilla en segundos en este campo. Al alcanzar este límite la rejilla finalizará y se guardarán los resultados.
  Si no se introduce ningún número o el número es 0, esta función quedará desactivada automáticamente.

- **Distracciones:** Pantalla de configuración de las distracciones. Estas son:
  - **Metrónomo:** Se reproduce el sonido de un metrónomo con un intervalo de tiempo dado.
  - **Música:** Se reproduce una canción de fondo.
  - **Línea:** Aparece una línea en movimiento en pantalla.

  El funcionamiento de las distracciones será detallado en el siguiente punto.

Al pulsar el botón Crear Rejilla todos los valores y configuraciones introducidos se guardarán y se abrirá la pantalla de configuración de la rejilla seleccionada, que depende del tipo de la misma.

B6. Distracciones

Las distracciones serán usadas por la aplicación para poner impedimentos a la atención del usuario y lograr así ponerla aprueba.

Las distracciones disponibles en la aplicación se pueden activar, desactivar y configurar en la pantalla de configuración de distracciones o en el menú rápido disponible en las rejillas.

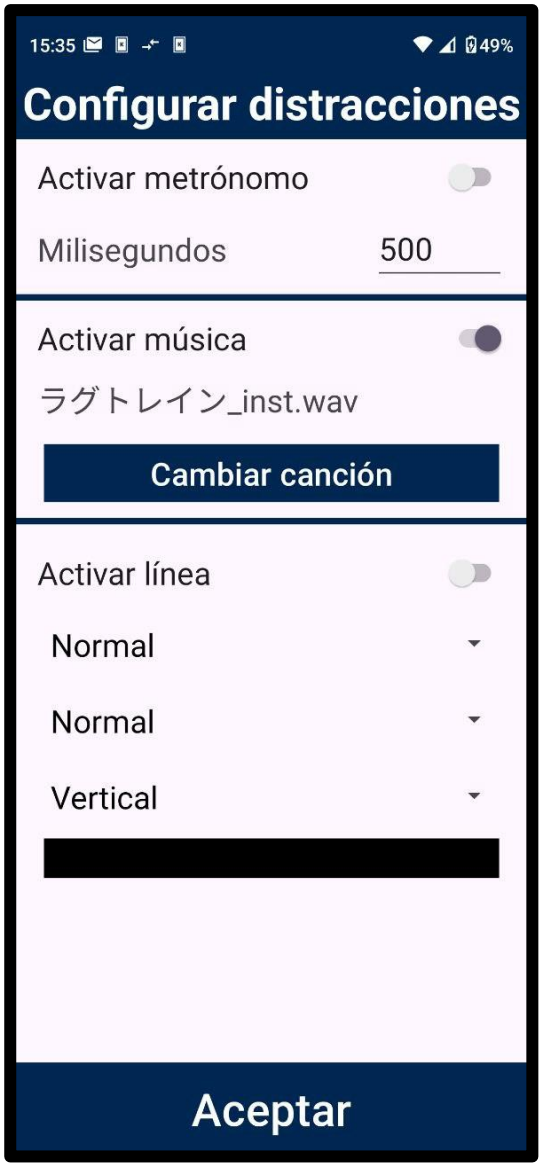


Figura 7.B: Pantalla de configuración de distracciones

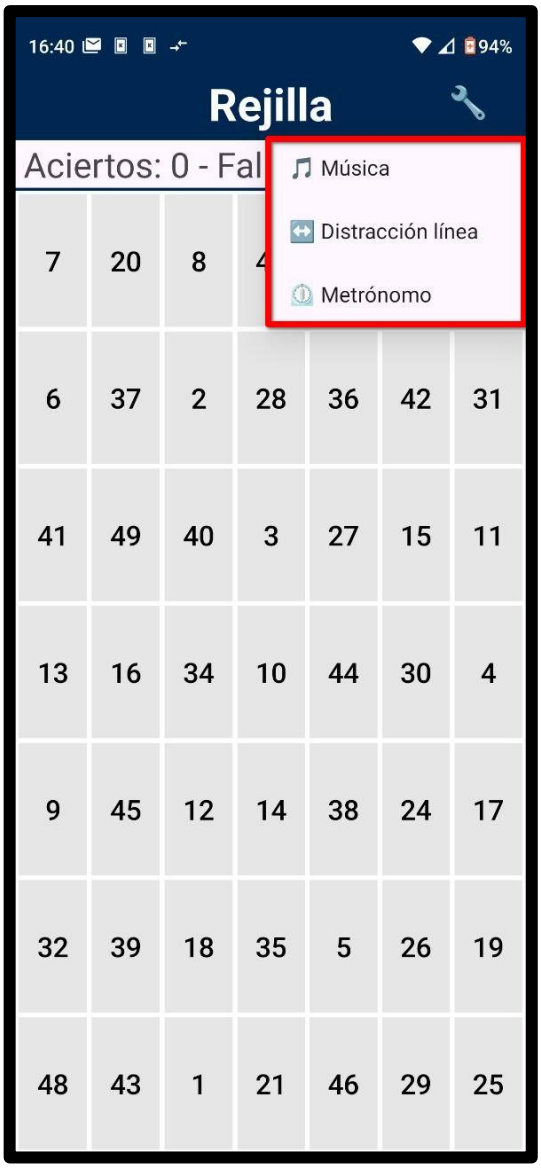


Figura 8.B: Menú de configuración rápida de distracciones en la pantalla de rejilla resaltado en rojo en la esquina superior derecha

Las distracciones disponibles en la aplicación son:

- **Metrónomo:** Un metrónomo configurable. Se configura introduciendo el tiempo en milisegundos en esta pantalla.

- **Música:** Se puede seleccionar una canción que sonará de fondo durante la realización de la rejilla. Para seleccionar la canción hay que pulsar el botón

Cambiar canción, lo que abrirá un menú de selección de archivos para elegir la canción deseada.

El nombre de la canción seleccionada aparecerá en la sección de música en esta pantalla justo encima del botón de cambiar canción.

- **Línea:** Una línea configurable que se moverá por toda la pantalla durante una rejilla. En esta pantalla se pueden seleccionar los siguientes parámetros:
    - **Grosor de la línea:** Con su respectiva lista desplegable.
    - **Velocidad de movimiento de la línea:** Con su respectiva lista desplegable.
    - **Dirección de movimiento de la línea:** Con su respectiva lista desplegable.
    - **Color de la línea:** Pulsando la barra de color. Se abrirá el selector de colores de la aplicación.

  La línea recorrerá la pantalla de forma indefinida hasta finalizar la rejilla.

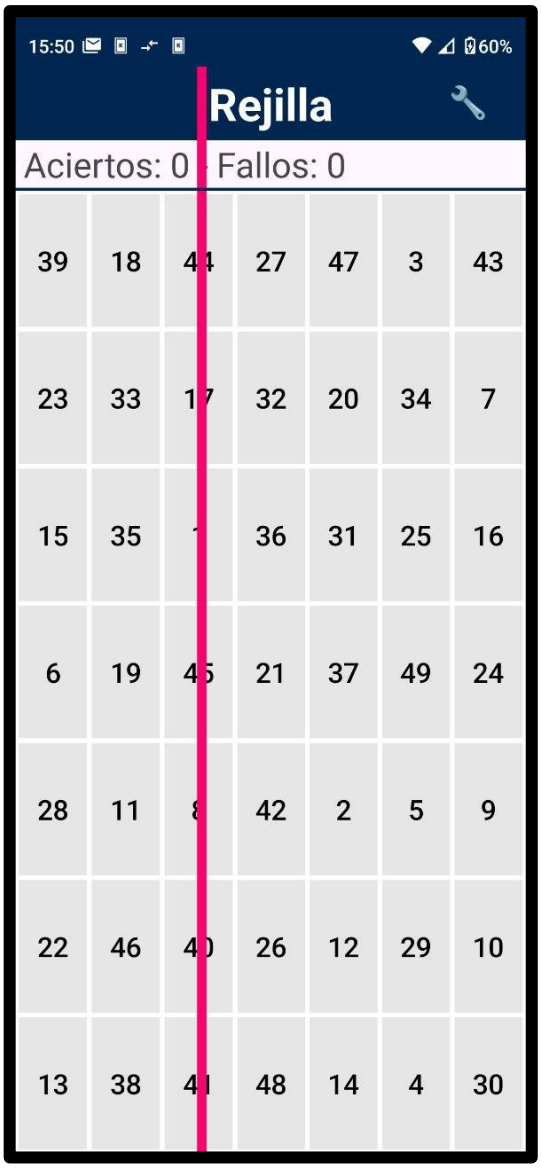


Figura 9.B: Distracción de línea vertical roja en una rejilla numérica

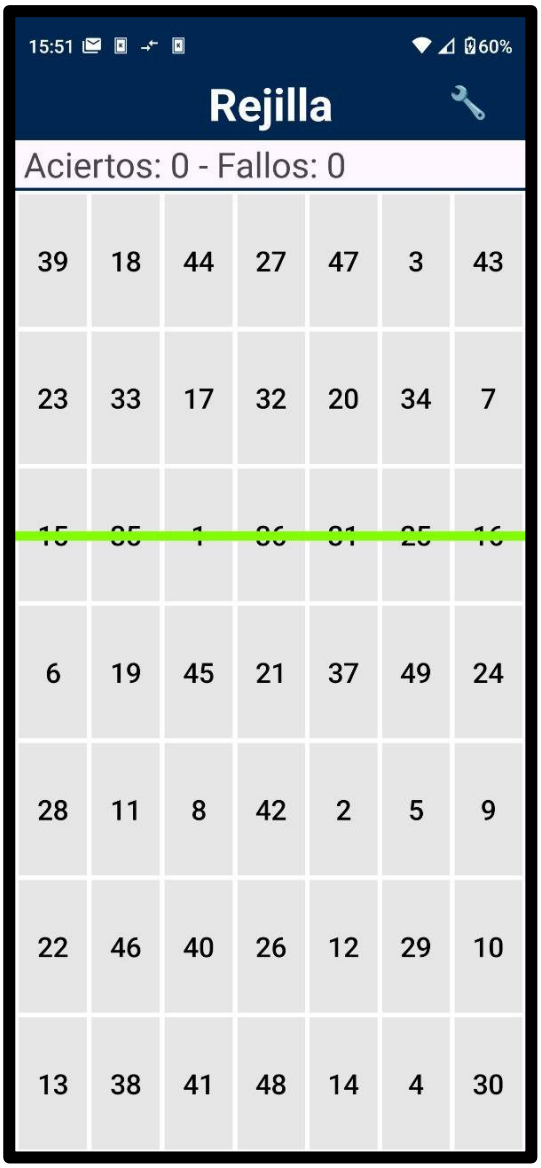


Figura 10.B: Distracción de línea horizontal verde en una rejilla numérica

B7. Selector de colores

En la aplicación se requiere seleccionar colores para utilizar diversas opciones, como pueden ser:

- Elegir colores en la rejilla de colores
- Seleccionar el color de la distracción de línea
- Elegir colores de fondo para las casillas

Para esto se utiliza una pantalla de selección de color, que con un simple click en el color deseado se puede seleccionar este y configurar su brillo para usarlo en la aplicación.

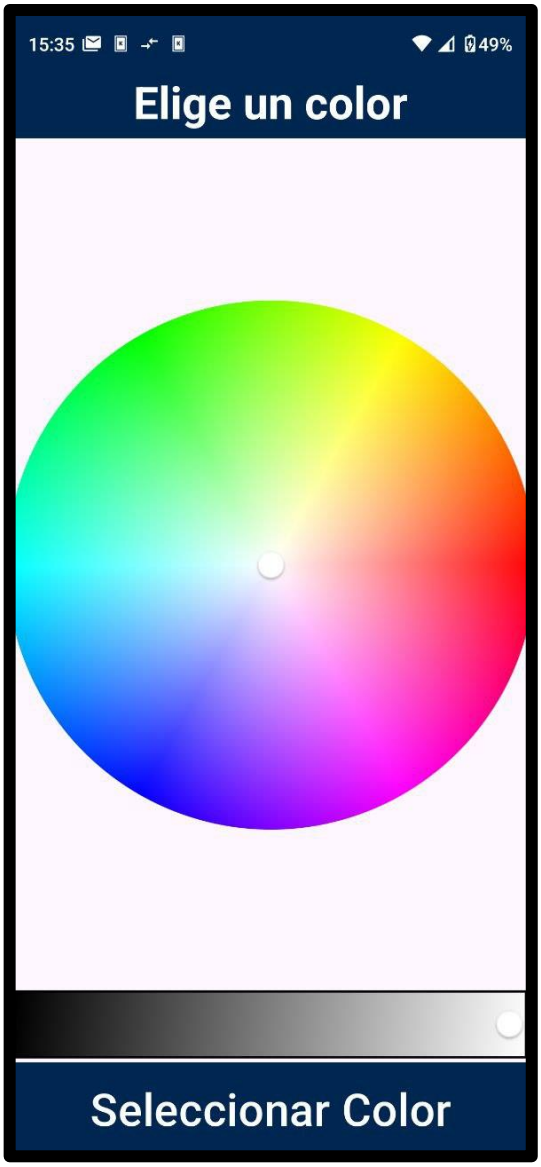


Figura 11.B: Pantalla de selección de colores y brillo

En esta pantalla se pulsará sobre el circulo de colores y tras eso el color pulsado aparecerá en la barra de selección de brillo que se encuentra en la parte inferior de la pantalla.

Tras configurar el color según se desee, se pulsa el botón Seleccionar Color para redirigir a la pantalla anterior habiendo seleccionado el color que el usuario quiera.

B8. Colores de fondo

Se pueden seleccionar colores de fondo a usar en las casillas de las rejillas numérica, de letras, de imágenes y de colores.

Estos colores se seleccionan en la pantalla de selección, que permite hasta nueve, y se usarán de forma aleatoria en las casillas.

Aquí se dispone también de un menú de plantillas, donde se pueden cargar los colores de un archivo o exportar los colores actuales a uno.

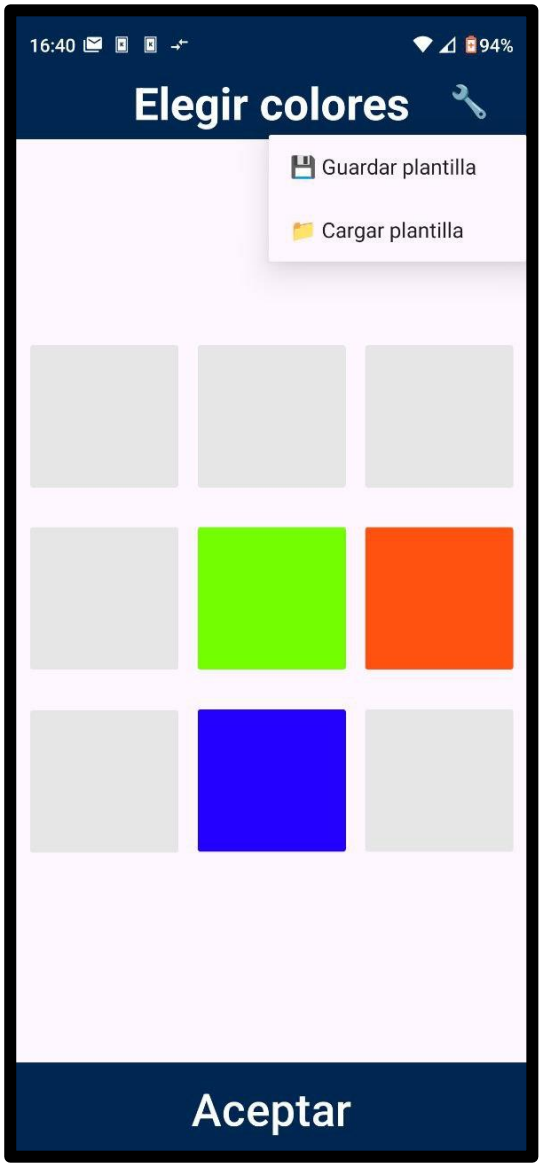


Figura 12.B: Pantalla de selección de colores de fondo y menú de gestión de plantillas

Para seleccionar un color únicamente hay que pulsar una casilla, lo que abrirá un menú flotante con dos opciones:

- **Borrar color:** Borra el color de la casilla pulsada.

- **Cambiar color:** Abre la pantalla de selección de colores para elegir un color que poner en esta casilla.

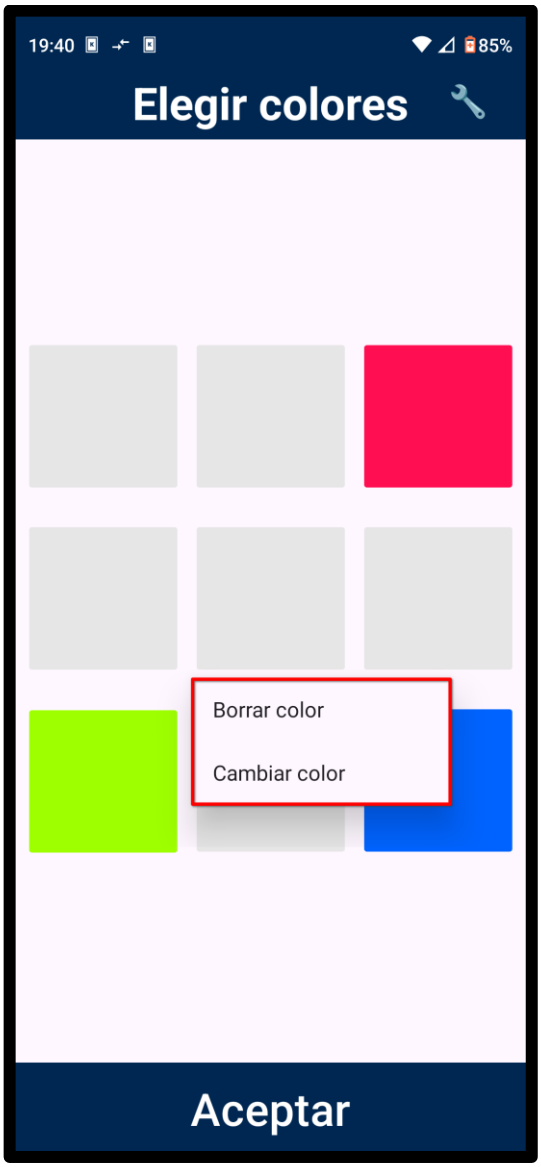


Figura 13.B: Menú flotante de gestión de colores resaltado en rojo en la pantalla de selección de colores de fondo

Tras pulsar el botón Aceptar se aplicarán los cambios y se volverá a la pantalla de creación de rejilla.

Para utilizar estos colores hay que activar la opción de colores de fondo en el menú de creación de rejillas.

B9. Emparejar

La función de emparejar se usa para añadir a la rejilla una dificultad extra: hacer que el botón a pulsar sea uno concreto y no cualquiera válido.
Es decir, en el caso de una rejilla de colores, se puede pulsar cualquiera de los colores que se han seleccionado como correctos durante su creación, pero al activar la función emparejar, el color que se tendrá que pulsar será el que se indique.

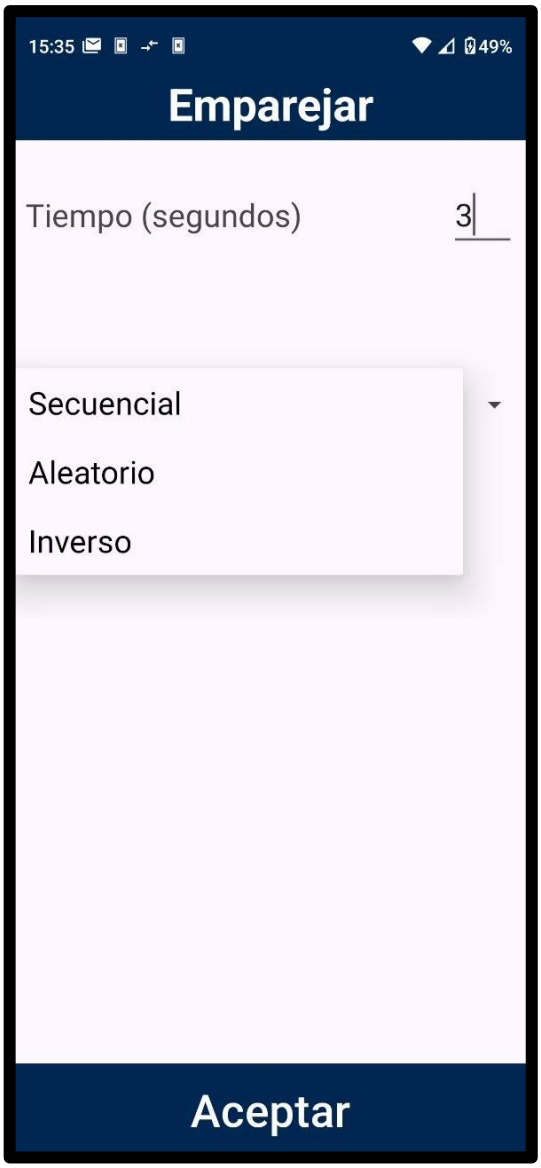


Figura 14.B: Pantalla de configuración de la función emparejar

En la pantalla de configuración hay dos opciones:

- **Tiempo (segundos):** Tiempo en segundos cada cuánto cambiará el botón a pulsar. Si se deja vacío o en 0 el botón solo cambiará cuando se pulse el botón correcto indicado.

- **Orden:** Orden de iteración para los elementos a pulsar. Puede ser:
    - **Secuencial**

- **Aleatorio**
- **Inverso**

Tras configurar todo y pulsar Aceptar, habrá que activar la opción de emparejar desde el menú de creación de rejillas, y entonces aparecerá el botón a pulsar en la rejilla como se indica en las figuras 15.B y 16.B.

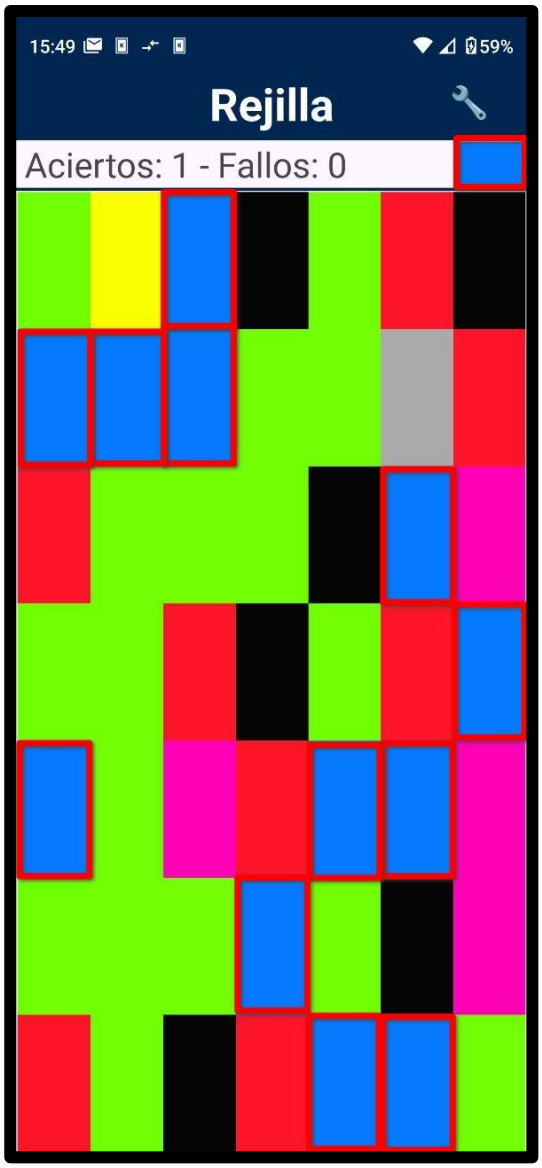


Figura 15.B: Función emparejar en una rejilla de colores. El color a pulsar es el azul

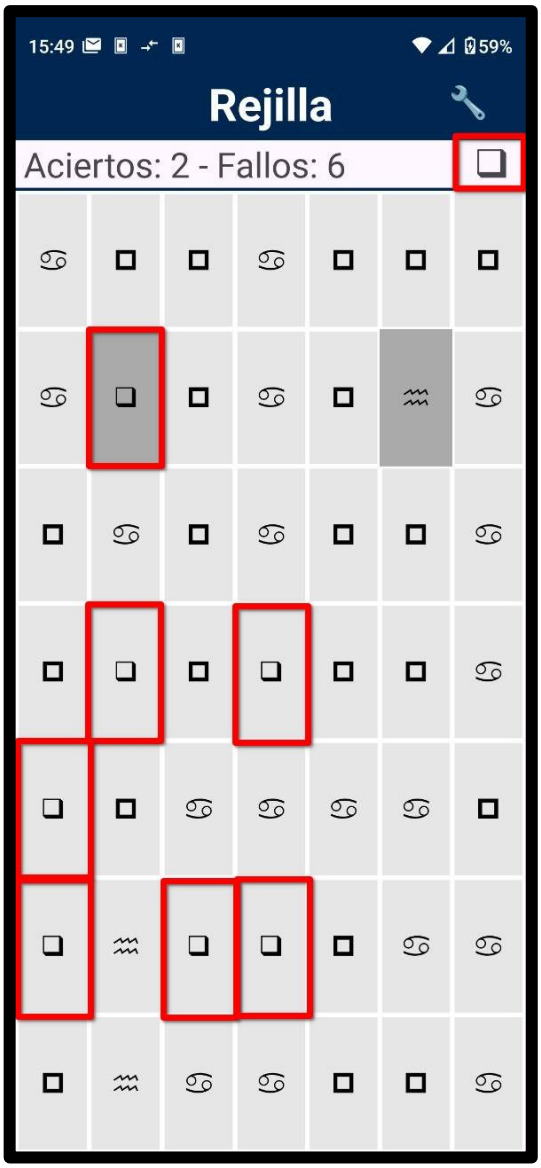


Figura 16.B: Función emparejar en una rejilla de wingdings. El símbolo a pulsar es el cuadrado

B10. Aleatorizar la rejilla

Al establecer un tiempo en la función aleatorizar en el menú de creación de rejillas, se activará la función aleatorizar con el tiempo seleccionado, que cambiará de posición todos los botones de la rejilla con el intervalo de tiempo dado.

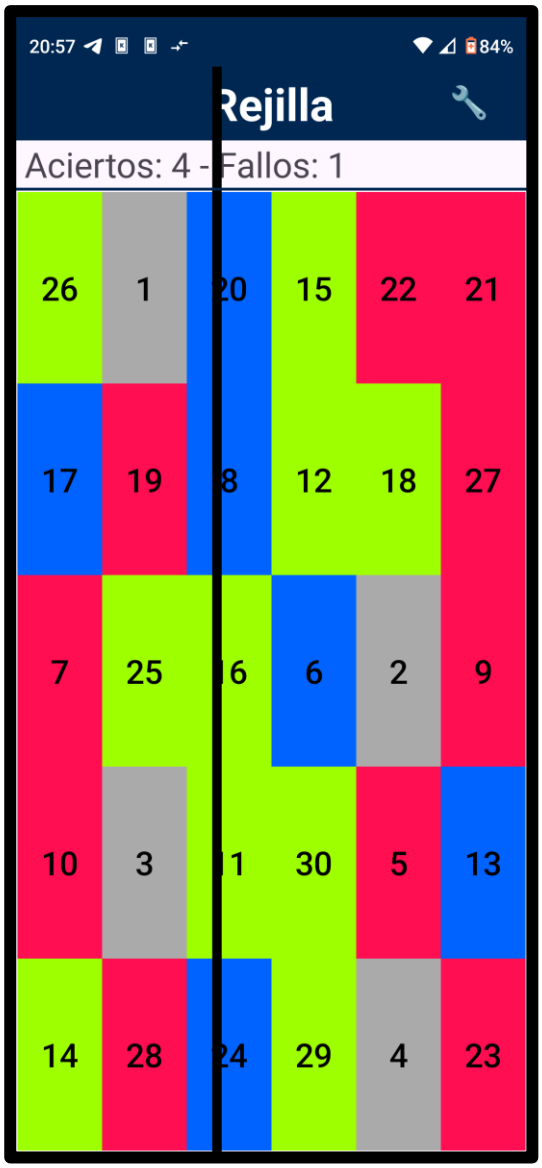


Figura 17.B: Ejemplo primera iteración de aleatorización en una rejilla numérica

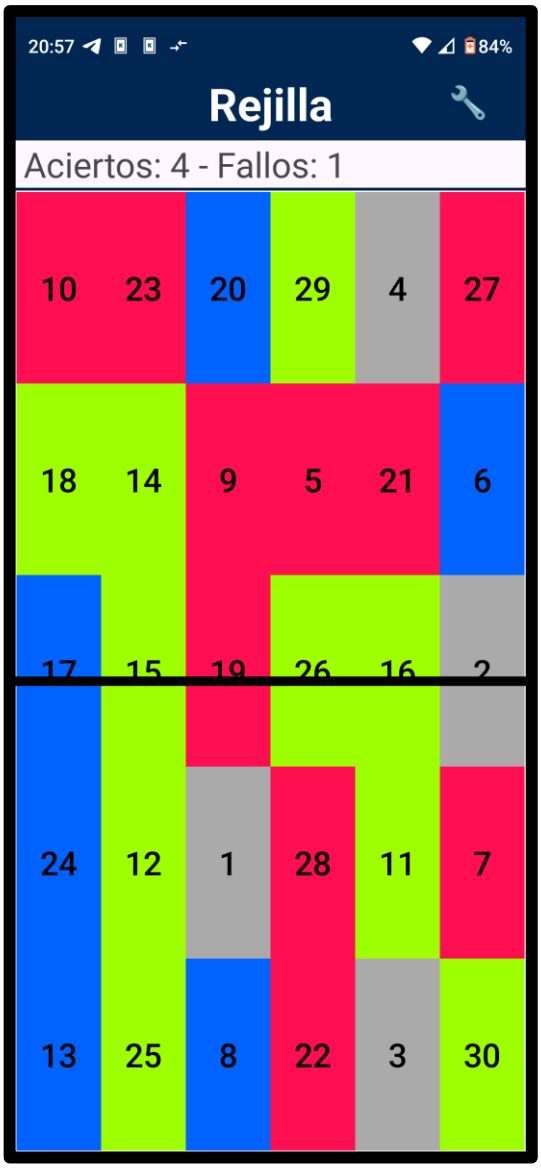


Figura 18.B: Ejemplo segunda iteración de aleatorización en la misma rejilla numérica

Como se puede observar en las figuras 17.B y 18.B, la aleatorización mantiene los colores de fondo de cada botón. También se mantiene su estado, si este ha sido pulsado se mantendrá gris y sin interacción.

B11. Límite de tiempo

La aplicación dispone de una función para limitar el tiempo que el usuario tiene para completar una rejilla utilizando un contador.

Si se ha seleccionado un tiempo límite en la pantalla de creación de rejilla, este tiempo aparecerá en la barra superior de la rejilla en forma de contador, que al llegar a cero finalizará la rejilla y registrará los resultados.

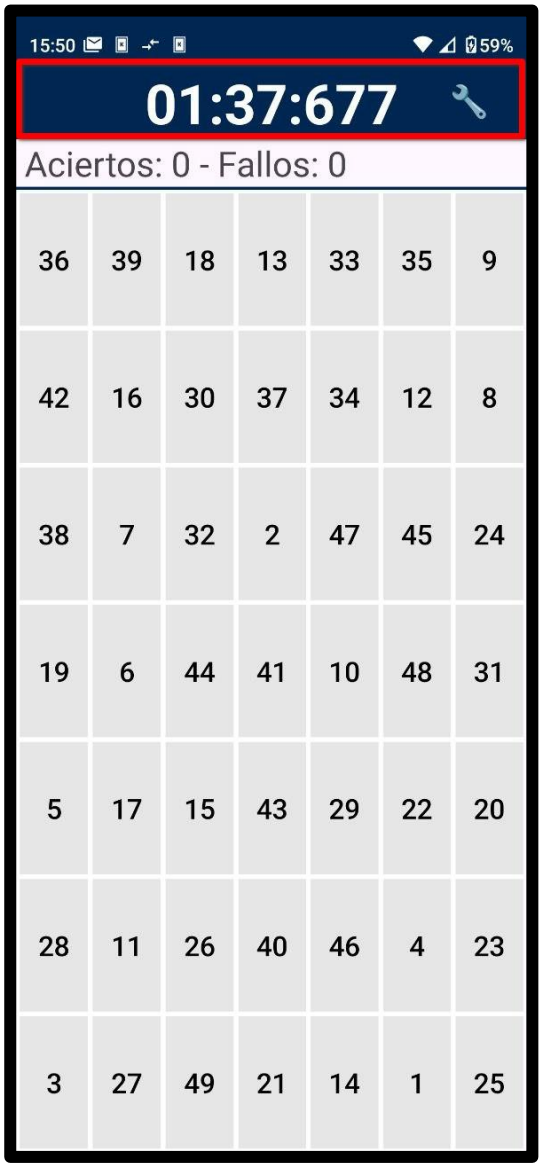


Figura 19.B: Contador de tiempo límite marcado en rojo en una rejilla numérica

B12. Rejilla numérica

La rejilla numérica está formada por números que deberán ser pulsados en el orden especificado en la pantalla de creación de la rejilla.

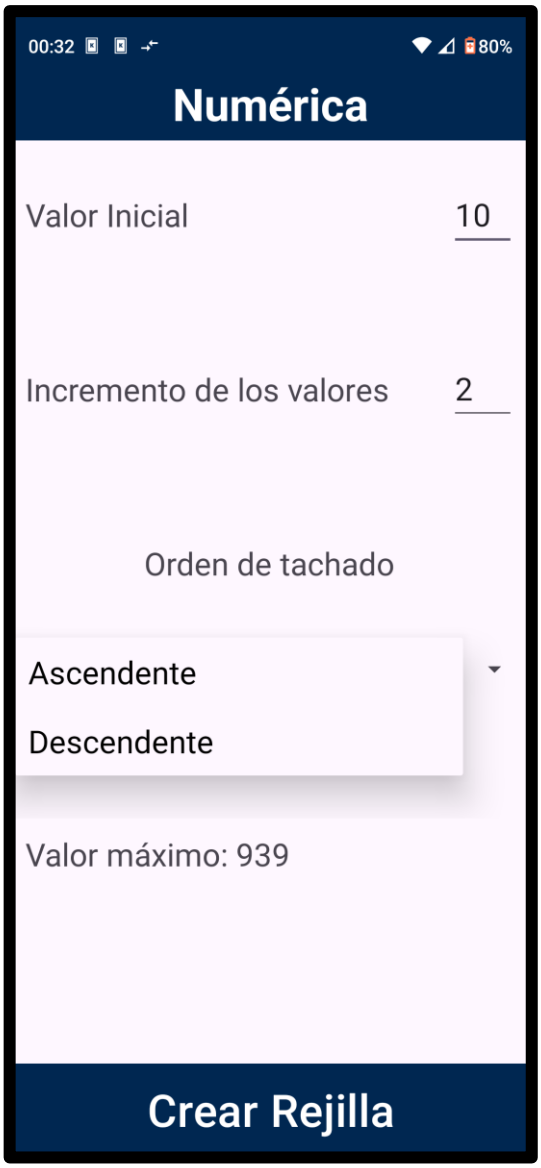


Figura 20.B: Pantalla de creación de rejilla numérica

En la pantalla de creación de este tipo de rejilla existen los siguientes componentes:

- **Valor inicial:** Valor del que parte la numeración y primer valor a pulsar en la rejilla.

- **Incremento de los valores:** Tamaño del salto que se da de un número al siguiente.

- **Orden de tachado:** Orden de pulsación de los números.
    - En caso de ser ascendente, el número siguiente al pulsado será el número pulsado más el número introducido en el incremento.
    - En caso de ser descendente, el número siguiente al pulsado será el pulsado menos el incremento.

- **Indicador valor máximo/mínimo:** En la parte inferior de la pantalla existe un indicador que informa de cuál es el valor máximo (en caso de seleccionar un orden ascendente) o mínimo (en caso de seleccionar un orden descendente). Si no se respeta este valor, se notificará al usuario y no se creará la rejilla.

Al pulsar el botón Crear Rejilla se crea la rejilla y se muestra al usuario la pantalla de la misma, comenzando el proceso de evaluación y entrenamiento de la atención.

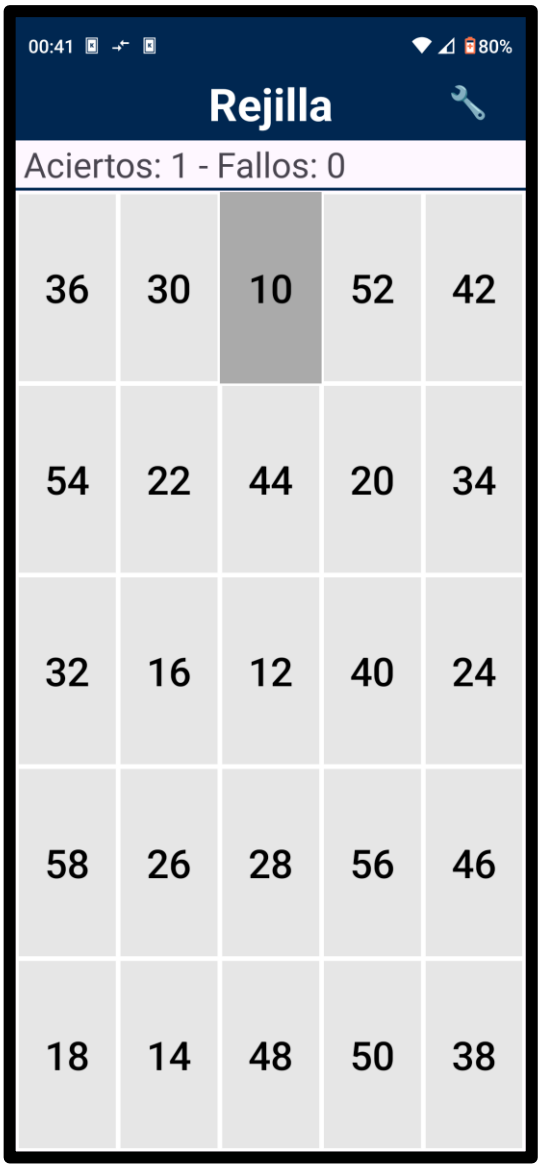


Figura 21.B: Rejilla numérica con valor inicial 10 e incremento de 2 y orden ascendente

El usuario debe pulsar el botón siguiente al pulsado, en el caso de la figura 21.B, esta rejilla es ascendente, así que el siguiente número a pulsar es el 12, ya que el primero pulsado fue el 10.

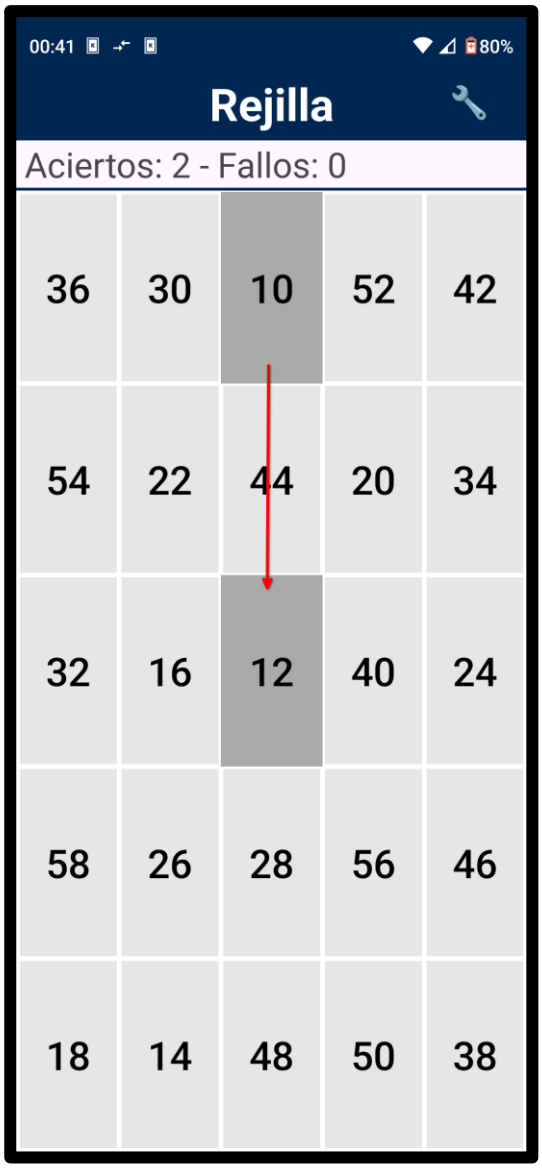


Figura 22.B: Continuación de figura 21.B. El valor pulsado es el 12

Tras eso se seguirán pulsando números con valores de dos en dos: 14, 16, 18, 20... hasta el valor final, en este caso el 58

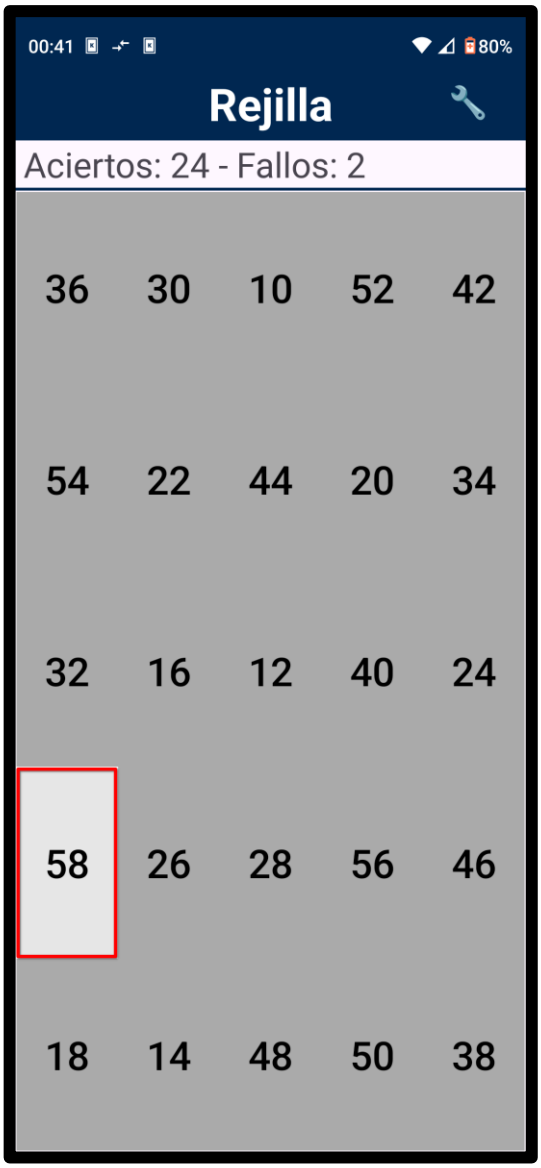


Figura 23.B: Continuación de figura 22.B. Solo falta un número, el 58, marcado en rojo, para completar la rejilla

Tras pulsar el último valor, se abrirá la pantalla de detalles que informa al usuario de los resultados obtenidos.

B13. Rejilla de colores

En la rejilla de colores el usuario debe escoger doce colores, y de estos doce colores, nueve de ellos elegidos por él se utilizarán como colores correctos que deberá pulsar, mientras los no seleccionados serán distracciones que debe evitar. Se pueden

seleccionar los doce y los nueve o seleccionar menos de cada uno, por ejemplo, cuatro y solo dos correctos.

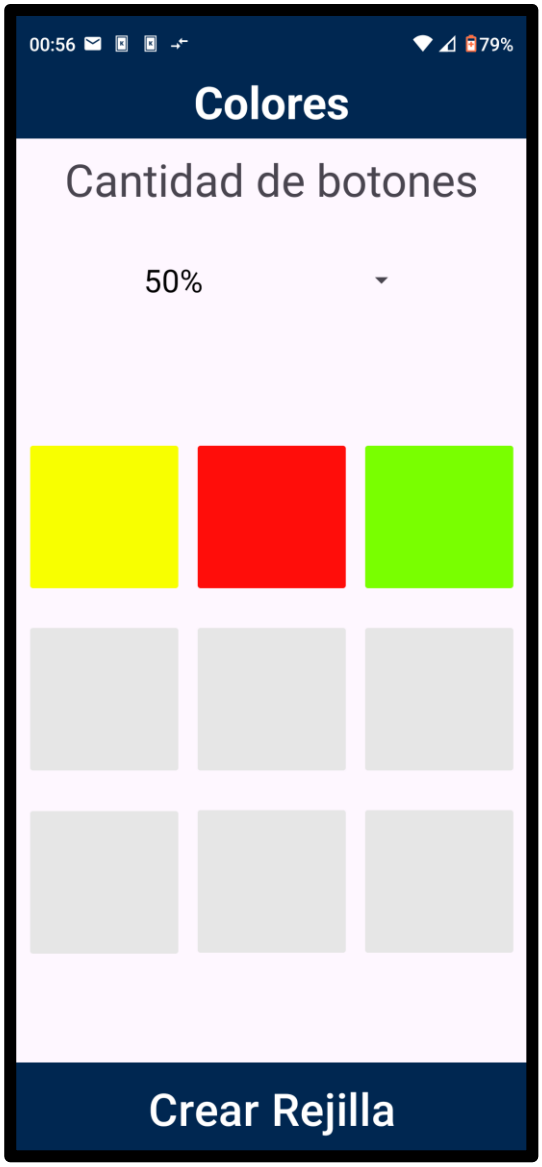


Figura 24.B: Pantalla de selección de colores correctos. Aquí se seleccionan los colores que el usuario deberá pulsar. Los colores seleccionados son: amarillo, rojo y verde

La pantalla de creación de la rejilla de colores consiste en nueve casillas y un selector de cantidad de botones.

Las nueve casillas son botones con los que se puede interactuar, y al pulsar uno se abre la pantalla de selección de colores, desde la cual se puede elegir el color que se usará para esa casilla pulsada.

El selector de cantidad indica cuantas de las casillas generadas en la rejilla serán correctas. Se dispone de las siguientes opciones:

- **Aleatorio (opción por defecto)**
- **25%**
- **50%**
- **75%**

La pantalla de selección de colores se muestra en la siguiente figura:

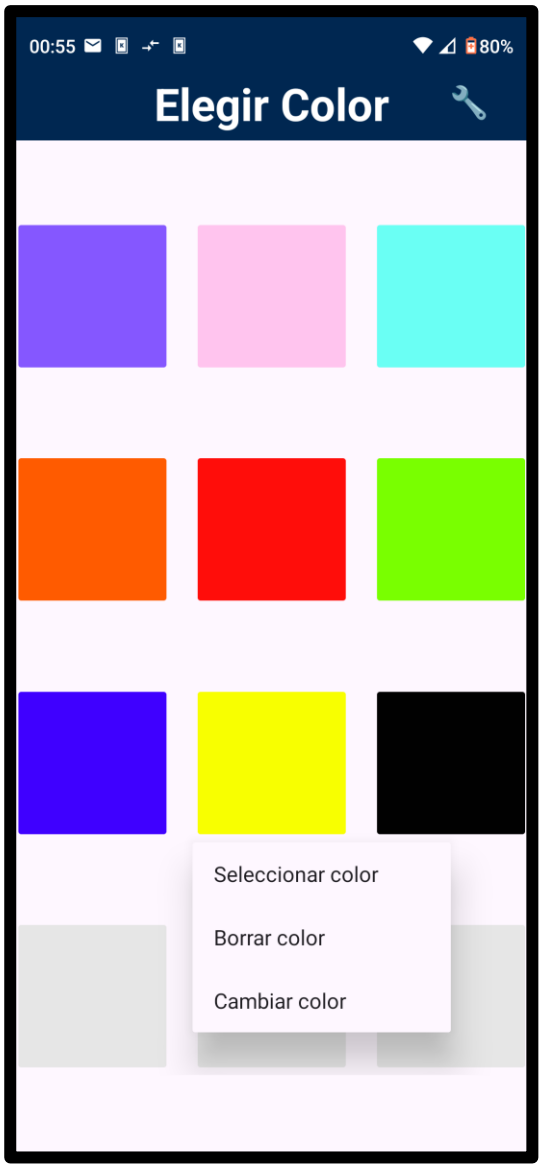


Figura 25.B: Pantalla de selección de colores. Aquí se seleccionan todos los colores que formarán parte de la rejilla

En esta pantalla estarán, divididos en doce casillas, tanto los colores correctos como los incorrectos, en ambos casos seleccionados por el usuario.

Al pulsar una casilla de esta pantalla se abrirá un menú flotante como el que se ve en la figura 25.B, que ofrece las siguientes opciones:

- **Seleccionar color:** El color en esa casilla será usado como color correcto.

- **Borrar color:** Se borra el color de la casilla.

- **Cambiar color:** Se abre el selector de colores de la aplicación para elegir un nuevo color para la casilla.

En esta pantalla, al igual que en las demás pantallas de selección de elementos para la creación de rejillas y al igual que en la pantalla de selección de colores de fondo, se dispone de un icono de una llave inglesa en la esquina superior derecha, que al pulsarlo abre el menú de gestión de plantillas para guardar o cargar las casillas actuales en archivos de plantilla tipo JSON.

Tras seleccionar los colores a usar, se pulsa el botón de Crear Rejilla.
Si no se han seleccionado colores se mostrará al usuario una notificación y no se creará la rejilla.

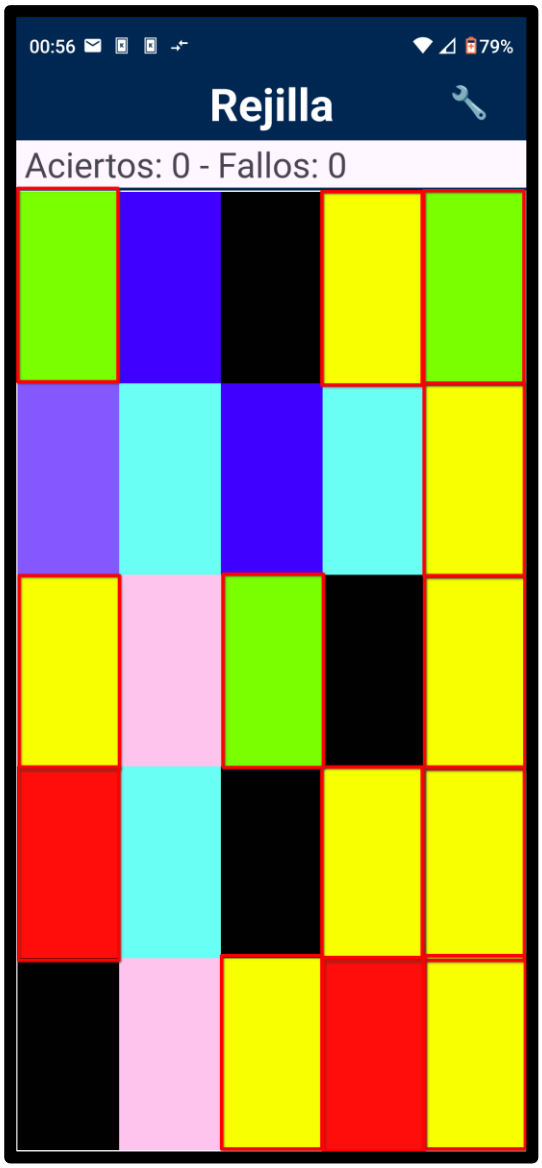


Figura 26.B: Pantalla de rejilla de colores creada. Los colores correctos están marcados en rojo

Como se puede apreciar en la rejilla 26.B, la cantidad de colores correctos (amarillo, rojo y verde, seleccionados en la figura 24.b) es la mitad, como se seleccionó.

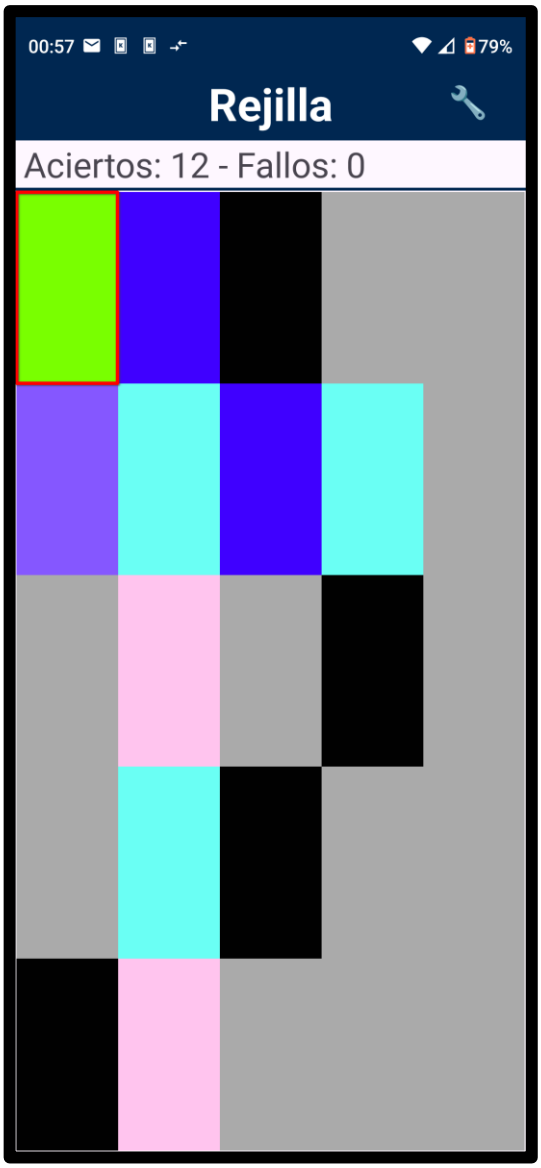


Figura 27.B: Pantalla de rejilla de colores. El único color correcto restante es el verde. Su casilla está marcada en rojo en esta figura

Al acabar esta rejilla, como con el resto de rejillas, se abrirá la pantalla de detalles.

B14. Rejilla de letras

Para crear una rejilla de letras habrá que hacer algo similar a la rejilla de imágenes.

En la pantalla de creación se dispone de nueve casillas que serán las correctas, y que habrá que rellenar con letras que se seleccionan en otra pantalla que consiste en doce casillas con opciones de edición del contenido para establecer la letra de cada una.

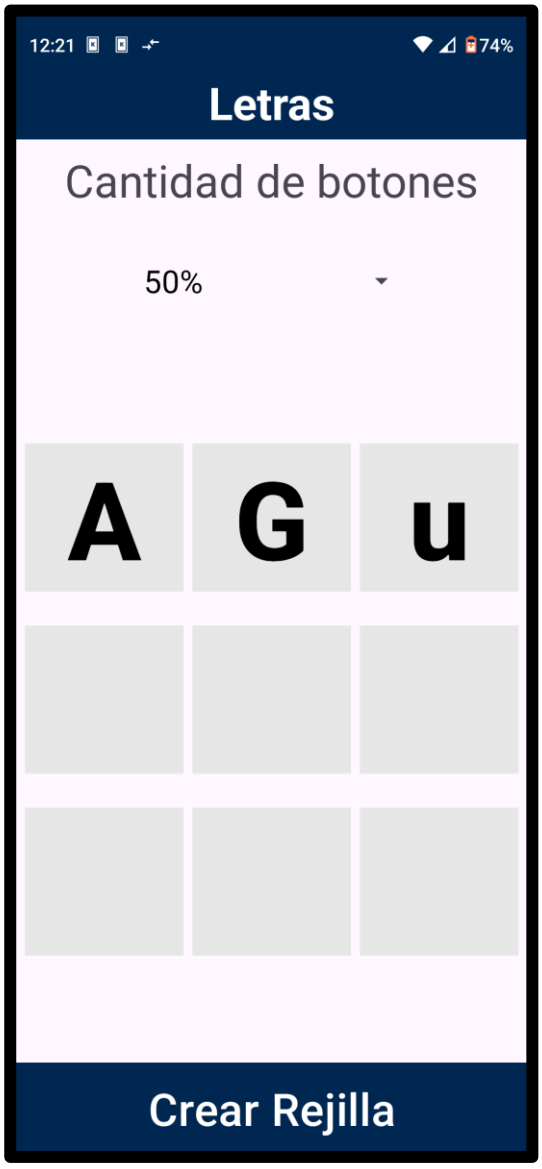


Figura 28.B: Pantalla de creación de rejilla de letras. Aquí se seleccionan las letras correctas

En la pantalla de creación también se dispone del selector de cantidad de letras correctas, pudiendo seleccionar un porcentaje o aleatorio.

Al pulsar una casilla se abre el selector de letras.

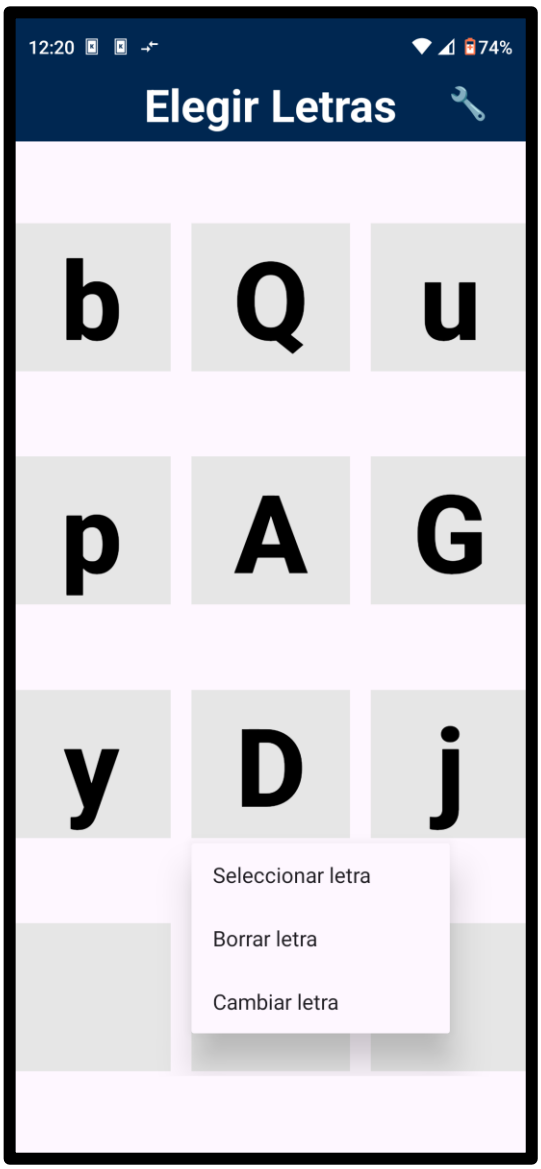


Figura 29.B: Pantalla de selección de letras

En esta pantalla tenemos doce casillas, y en cada una de ellas se puede establecer la letra que se desee utilizando un campo de texto que se abre al pulsar la opción de Cambiar letra.

Figura 30.B: Menú de introducción de texto en la selección de letras

Al igual que en las otras rejillas de elementos, se dispone de opciones para importar y exportar plantillas en la esquina superior derecha.

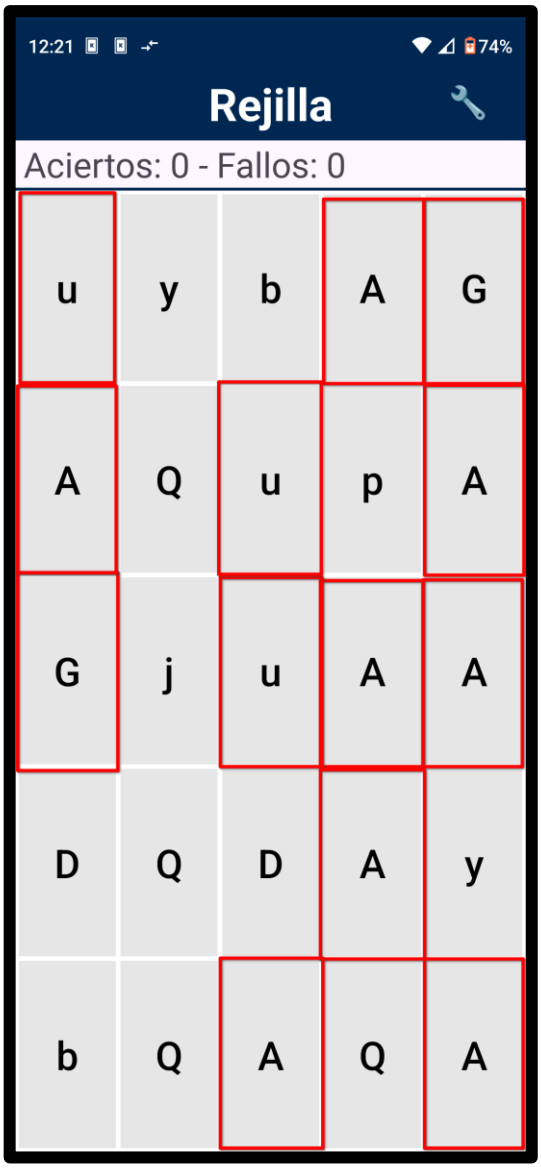


Figura 31.B: Rejilla de letras. Marcadas en rojo se encuentran las casillas correctas seleccionadas en la figura 28.B

Tras crear la rejilla habrá que pulsar de forma desordenada las letras correctas que se seleccionaron en la creación.

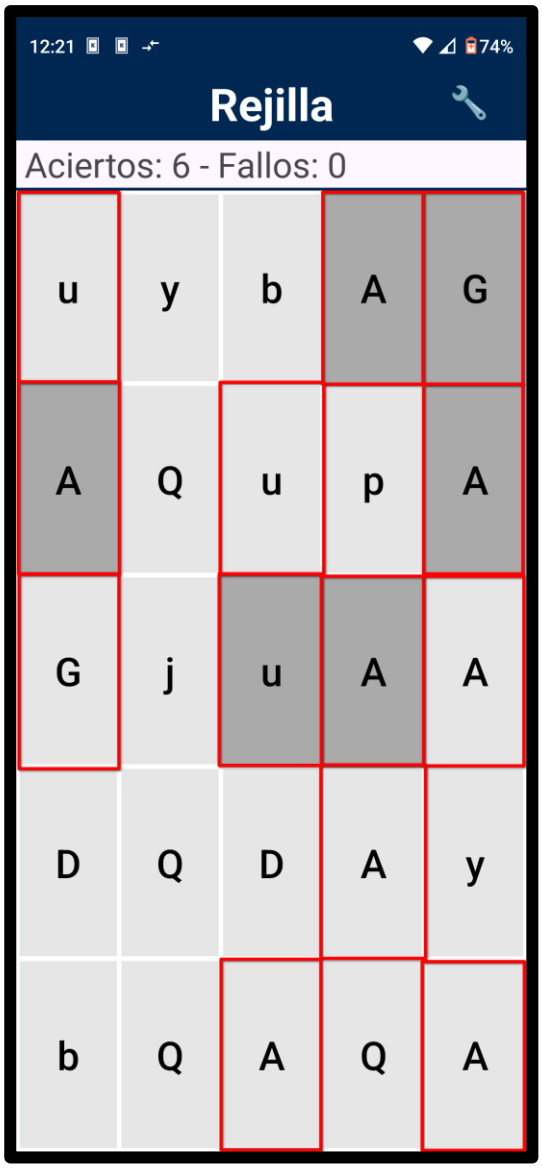


Figura 32.B: Continuación de la rejilla de letras de la figura 31.B con algunas casillas correctas ya pulsadas.

B15. Rejilla de imágenes

En la rejilla de imágenes el usuario puede cargar imágenes de su dispositivo para usarlas como casillas en la rejilla de imágenes.

La creación y el funcionamiento funcionan con el mismo sistema que las rejillas de letras y colores, disponiendo de una pantalla de selección de doce casillas, de las cuales nueve serán correctas.

Figura 33.B: Pantalla de creación de la rejilla de imágenes con la selección de imágenes correctas

De forma análoga a las dos rejillas anteriores, al pulsar una casilla se abrirá la pantalla de selección de imágenes.

Figura 34.B: Pantalla de selección de imágenes

Al pulsar la opción de cambiar imagen se abrirá el selector de archivos de Android para seleccionar la imagen del dispositivo que se quiere usar.

Tras elegir todas las casillas de la rejilla y las que habrá que pulsar, se crea la rejilla de imágenes.

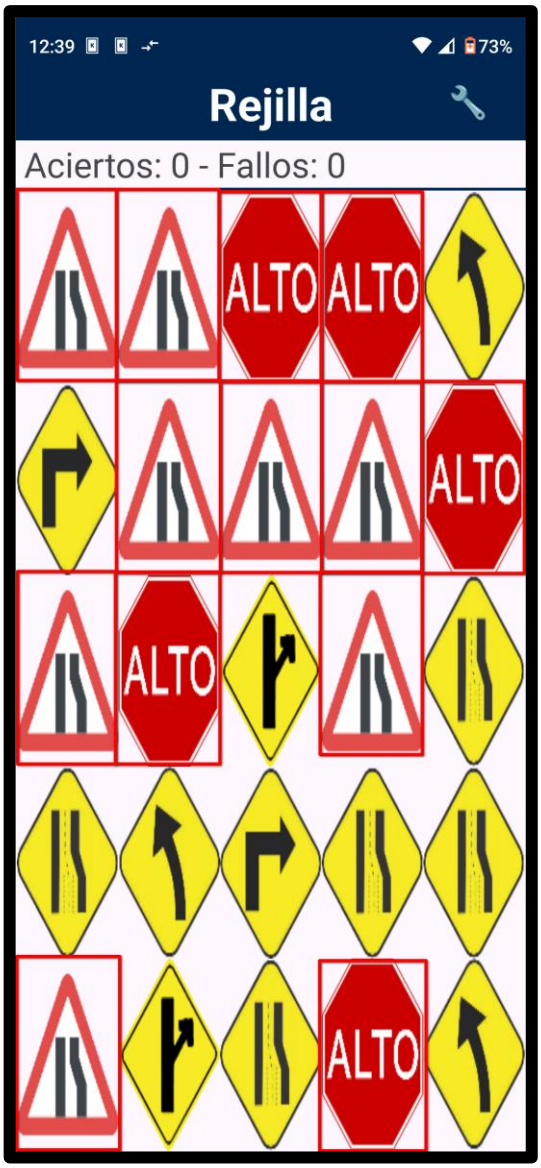


Figura 35.B: Rejilla de imágenes creada, las casillas correctas están marcadas en rojo y son las imágenes seleccionadas en la figura 33.B

Hay que pulsar las imágenes correctas sin un orden específico.

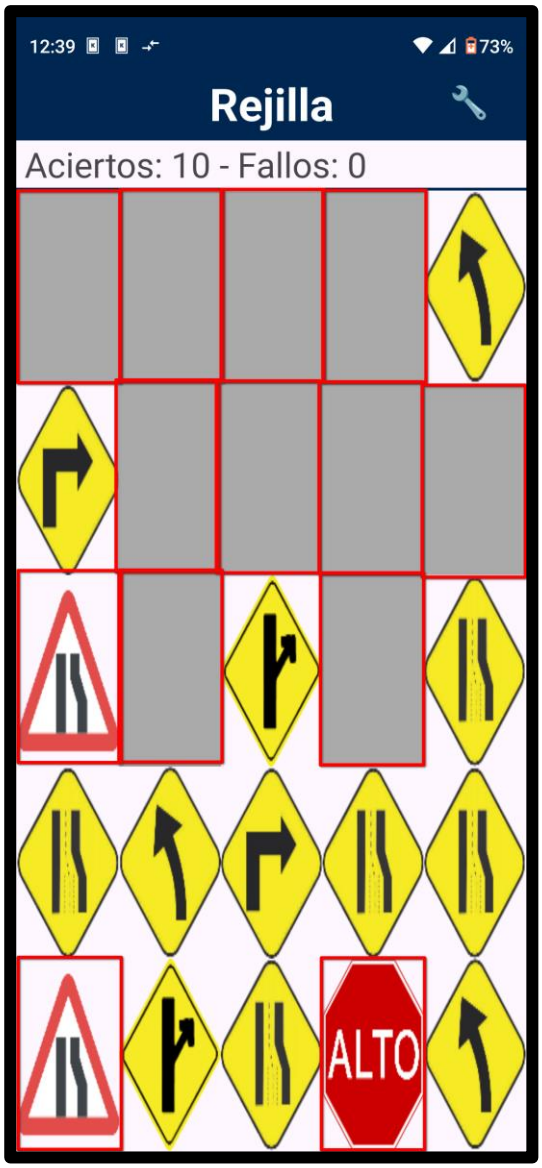


Figura 36.B: Continuación de la rejilla de imágenes creada en la casilla 35.B, las casillas pulsadas aparecen en gris

Al acabar la rejilla se abrirá la pantalla de detalles de los resultados obtenidos.

B16. Rejilla de abecedario

En la rejilla de abecedario el usuario deberá seleccionar un valor inicial que consiste en tres letras que se usarán como primera casilla a pulsar en la rejilla, y a partir de la cual, siguiendo el orden seleccionado, se generarán el resto de casillas en orden alfabético.

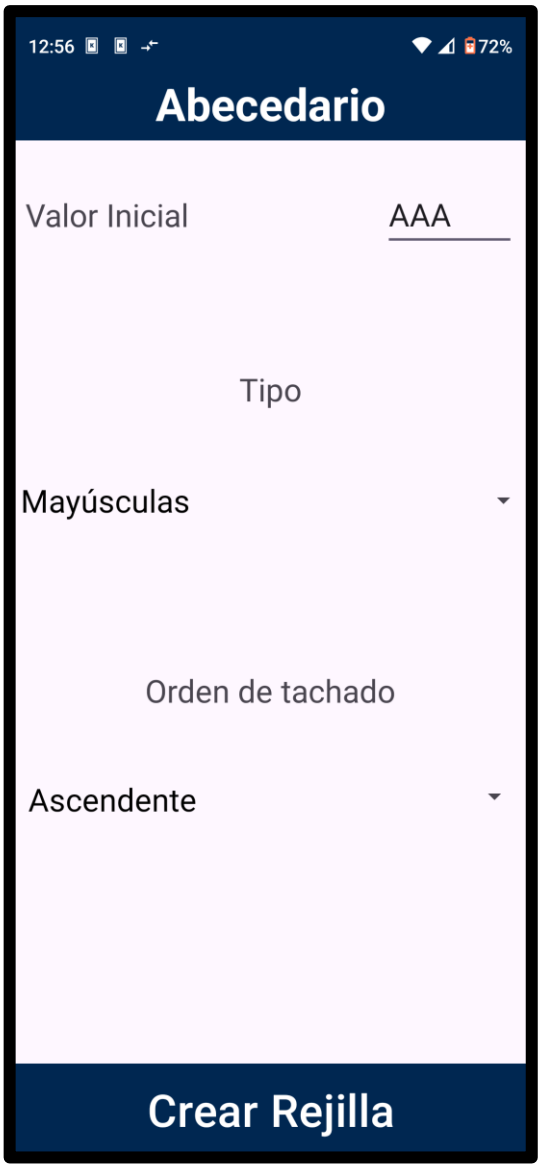


Figura 37.B: Pantalla de creación de la rejilla de abecedario

En la pantalla de creación se dispone de tres opciones:

- **Valor inicial:** Tres letras que se usarán inicialmente como primer valor de la rejilla.

- **Tipo:** Selector de tipo de letras. Se dispone de cinco opciones:
    - **Mayúsculas:** Todas mayúsculas.

- **Minúsculas:** Todas minúsculas.

- **Aleatorio:** Cada casilla sigue una estructura distinta a elegir aleatoriamente de las otras cuatro.

- **AbCdEfGh:** La primera y tercera letra mayúsculas y la segunda minúscula.

- **aBcDeFgH:** La primera y tercera letra minúsculas y la segunda mayúscula.

La opción elegida modificará las tres letras de cada casilla para que sigan la estructura especificada.

- **Orden de tachado:** Orden alfabético de tachado, pudiendo ser ascendente o descendente.

Al pulsar el botón de Crear Rejilla se creará la rejilla de abecedario usando los parámetros introducidos.

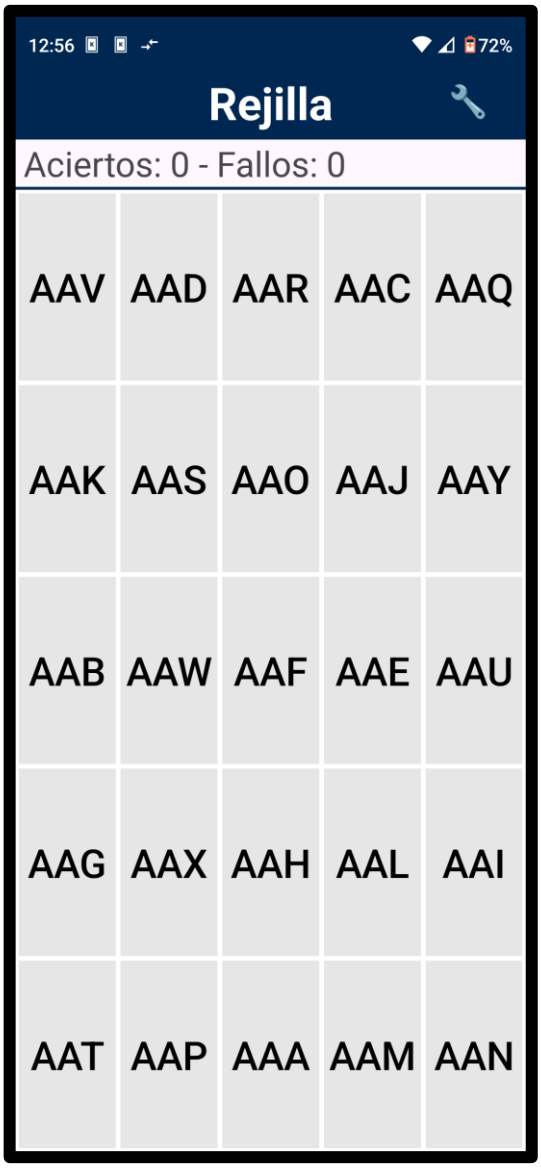


Figura 38.B: Rejilla de abecedario. La primera casilla a pulsar es AAA, y se debe seguir un orden ascendente, como se especificó en la figura 37.B

Se pulsará cada casilla en el orden especificado empezando por el valor inicial seleccionado.

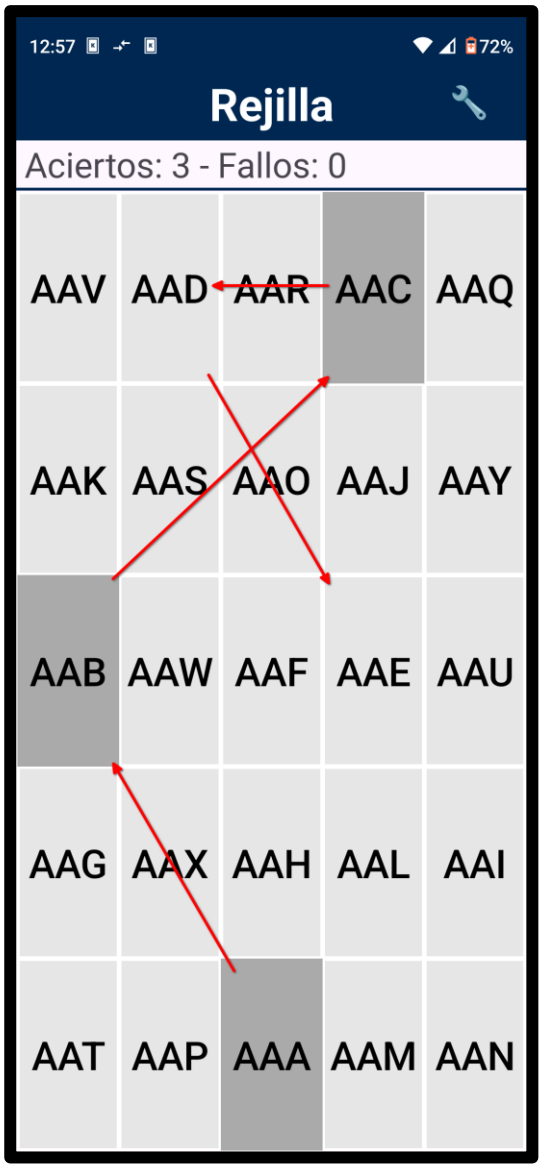


Figura 39.B: Continuación de la rejilla de abecedario de la figura 38.B. Las flechas rojas indican el orden a seguir

En el caso de la rejilla de la figura 38.B, el orden a seguir es ascendente, así que será: AAA, AAB, AAC, AAD, AAE, AAF, AAG... hasta AAY

Al acabar la rejilla se abrirá la pantalla de detalles de la prueba, que muestra los resultados.

B17. Rejilla de wingdings

La rejilla de wingdings es muy similar a la rejilla de letras, con la única diferencia de utilizar la fuente Wingdings para cambiar el aspecto de las letras seleccionadas.

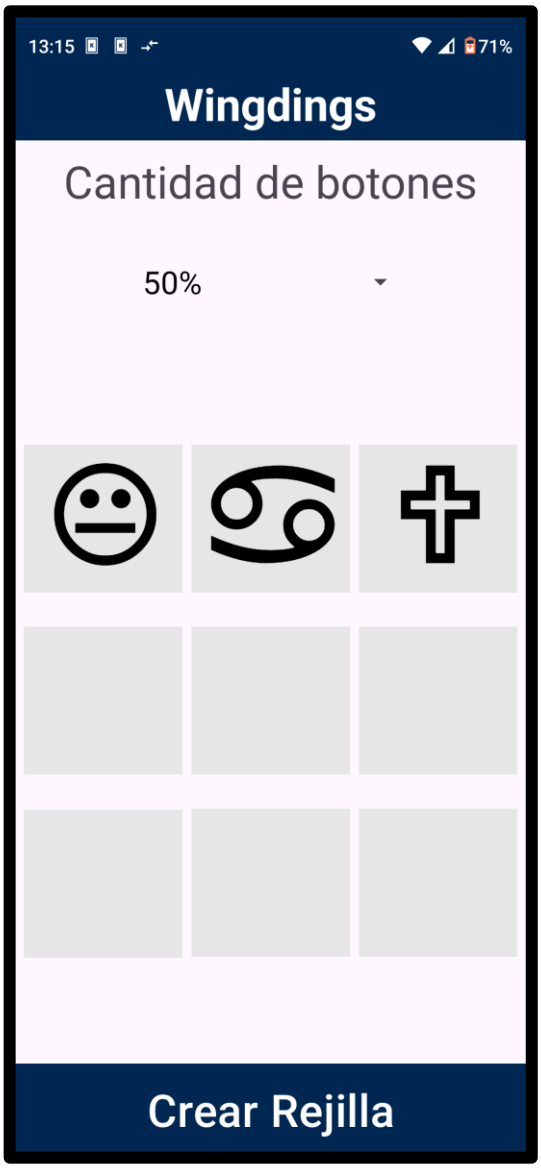


Figura 40.B: Pantalla de creación de la rejilla de wingdings

En la pantalla de creación se seleccionan los nueve (o menos) símbolos correctos a pulsar en la rejilla.

Análogamente con la rejilla de letras, al pulsar una casilla se abre la pantalla de selección de símbolos, con doce casillas.

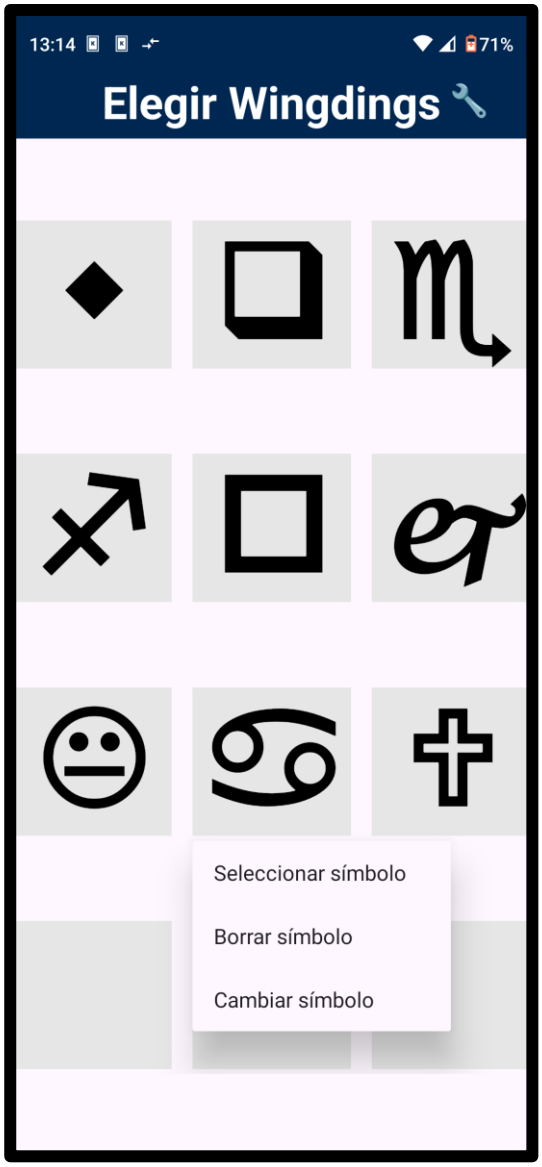


Figura 41.B: Pantalla de selección de símbolos

Al pulsar una casilla se dan las tres opciones disponibles en el resto de rejillas, siendo en este caso similar a la rejilla de letras, si se pulsa Cambiar símbolo, se abrirá un menñu de introducción de texto para introducir el símbolo deseado.

Al crear la rejilla se usarán los símbolos seleccionados y se deberán pulsar los elegidos como correctos.

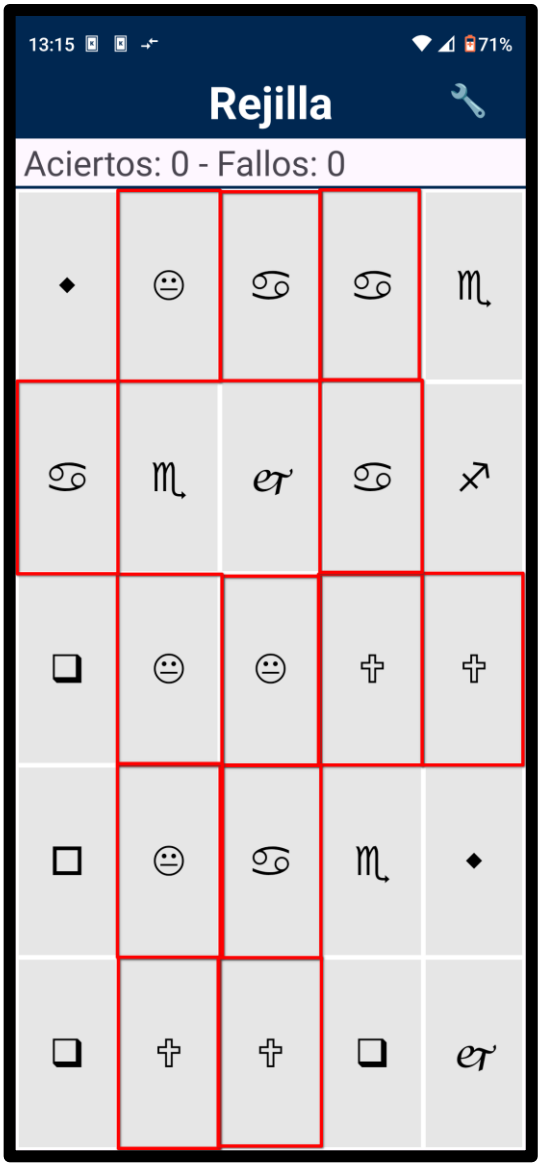


Figura 42.B: Rejilla de wingdings. Marcadas en rojo las casillas que contienen símbolos correctos seleccionados en la figura 40.B

Se seleccionarán los símbolos correctos sin orden específico hasta completar la rejilla.

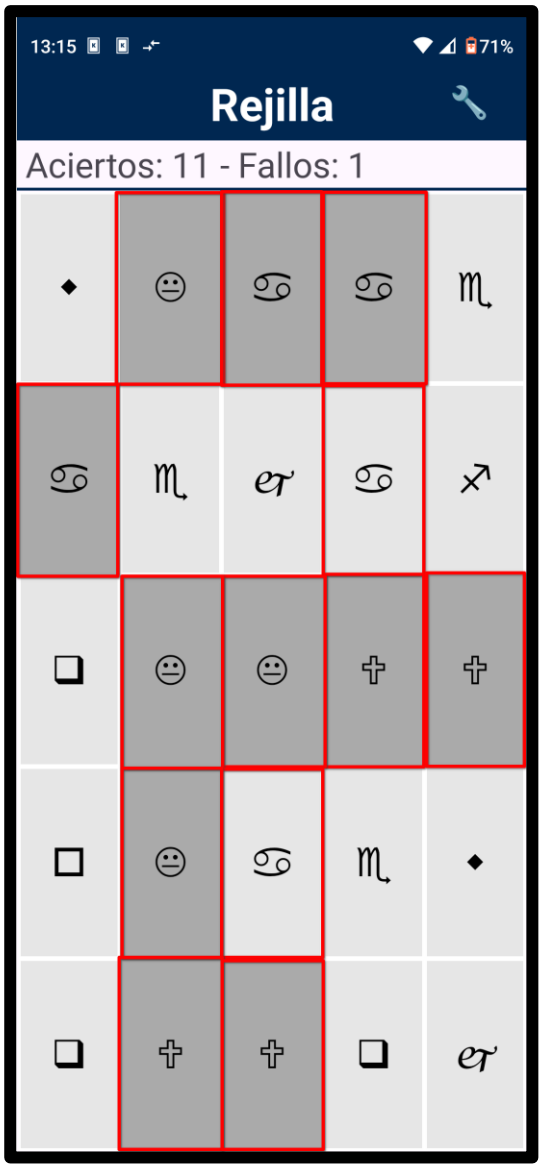


Figura 43.B: Continuación de la rejilla de wingdings de la figura 42.B. En gris las casillas correctas ya pulsadas

Al acabar se mostrarán los resultados.

B18. Pantalla de detalles

Al acabar una rejilla se mostrarán los resultados de la prueba realizada, que incluyen los datos del usuario, tiempo de pulsación de cada botón y mediciones extraídas de diversos parámetros obtenidos durante la realización de la rejilla.

Figura 44.B: Pantalla de detalles de los resultados

Los valores extraídos pueden ser usados para medir la capacidad de atención del usuario.

Los valores mostrados en esta pantalla son:

- **Fecha de realización:** Fecha de realización de la prueba, contiene minutos, segundos y milisegundos

DATOS DE USUARIO:

- **Nombre:** Nombre del usuario recogido del formulario rellenado por este. Es distinto al del usuario de MenPas con el que se inicia sesión en la aplicación para usarla

- **Apellidos:** Apellidos del usuario recogido del formulario rellenado por este

- **Edad:** Edad del usuario recogido del formulario rellenado por este

- **Género:** Género del usuario recogido del formulario rellenado por este

- **País:** País introducido por el usuario recogido del formulario rellenado por este

- **Deporte:** Deporte practicado por el usuario recogido del formulario rellenado por este

- **Posición:** Posición del usuario en el deporte practicado recogido del formulario rellenado por este

- **Lateralidad:** Lateralidad del usuario recogido del formulario rellenado por este

RESULTADOS DE LA REJILLA:

- **Aciertos:** Cantidad de aciertos en la rejilla.

- **Fallos:** Cantidad de fallos en la rejilla

- **Cantidad Botones:** Cantidad de casillas en la rejilla

BOTONES:

- **Botón X:** Lista de todos los botones pulsados (correctos e incorrectos) y el tiempo tardado en pulsarlos

- **Tiempo preliminar:** Tiempo tardado en la prueba preliminar

- **Tiempo total:** Tiempo total empleado en completar la rejilla

- **Media:** Media del tiempo tardado en pulsar cada botón en la prueba

- **Varianza:** Varianza del tiempo tardado en pulsar cada botón en la prueba

- **Máximo:** Valor máximo del tiempo tardado en pulsar cada botón en la prueba

- **Mínimo:** Valor mínimo del tiempo tardado en pulsar cada botón en la prueba

- **Tipo:** Tipo de la rejilla

- **Tamaño rejilla:** Tamaño de la rejilla en formato "Filas X Columnas"

- **Tiempo corregido:** Tiempo total empleado en completar la rejilla corregido usando el tiempo total de la prueba preliminar

- **Corrección:** Corrección aplicada en la prueba, es decir, el tiempo total tardado en completar la prueba preliminar

- **Media corrección:** Media del tiempo tardado en pulsar cada botón en la prueba corregida usando el tiempo total de la prueba preliminar

- **Varianza corrección:** Varianza del tiempo tardado en pulsar cada botón en la prueba corregida usando el tiempo total de la prueba preliminar

DISTRACCIONES:

LÍNEA:

- **Dirección línea:** Dirección de la distracción de línea utilizada en la prueba

- **Velocidad línea:** Velocidad de la distracción de línea utilizada en la prueba

- **Color línea:** Color de la distracción de línea utilizada en la prueba

- **Grosor línea:** Grosor de la distracción de línea utilizada en la prueba

MÚSICA:

- **Canción:** Nombre de la canción usada en la distracción de música utilizada en la prueba

METRÓNOMO:

- **Milisegundos:** Cantidad de milisegundos cada la cual suena la distracción de metrónomo utilizado en la prueba

EMPAREJAR:

- **Emparejar tiempo:** Cantidad de tiempo que pasa antes de que cambie el indicador de "siguiente botón a pulsar" en la rejilla cuando se activa la funcionalidad de emparejar con tiempo

- **Emparejar orden:** Orden utilizado para emparejar en la prueba

COLORES DE FONDO:

- **Colores de fondo:** Indica si se han usado colores de fondo en la prueba

VALORES RELATIVOS Y ABSOLUTOS:

- **Aciertos absolutos:** Porcentaje de aciertos absolutos en la prueba. Se calcula con la fórmula:
  %Aciertos Absolutos = aciertos / Cantidad Total de Botones

- **Aciertos relativos:** Porcentaje de aciertos relativos en la prueba. Se calcula con la fórmula:
  %Aciertos Relativos = aciertos / (aciertos + errores)

- **Errores relativos:** Porcentaje de errores relativos en la prueba. Se calcula con la fórmula:
  %Errores Relativos = errores / (aciertos + errores)

- **Errores absolutos:** Porcentaje de errores absolutos en la prueba. Se calcula con la fórmula:
  %Errores Absolutos = errores / Cantidad Total Botones

- **Eficacia:** Porcentaje de eficacia en la prueba. Se calcula con la fórmula:
  %Eficacia = (aciertos – errores) / Cantidad Total Botones

- **Efectividad:** Porcentaje de efectividad en la prueba. Se calcula con la fórmula:
  %Efectividad = (aciertos – errores) / (aciertos + errores)

- **Aleatorio:** Cantidad de segundos cada cuánto se aleatorizan las casillas de la rejilla

Aparte de los valores obtenidos, en esta pantalla se dispone de tres botones:

- **Ver gráfica:** Abre la pantalla de gráfica de la aplicación, que muestra en una gráfica el tiempo tardado en pulsar cada botón.

- **Exportar:** Permite al usuario guardar los resultados de la prueba en un archivo de texto en el dispositivo.

- **Subir resultados:** Permite al usuario subir a MenPas los resultados de la prueba.

Se puede acceder a esta pantalla de detalles al acabar un prueba o también desde el historial de pruebas en la pantalla de inicio de la aplicación.

B19. Gráfica

En la pantalla de gráfica se muestra la gráfica de tiempo de pulsación de cada botón.

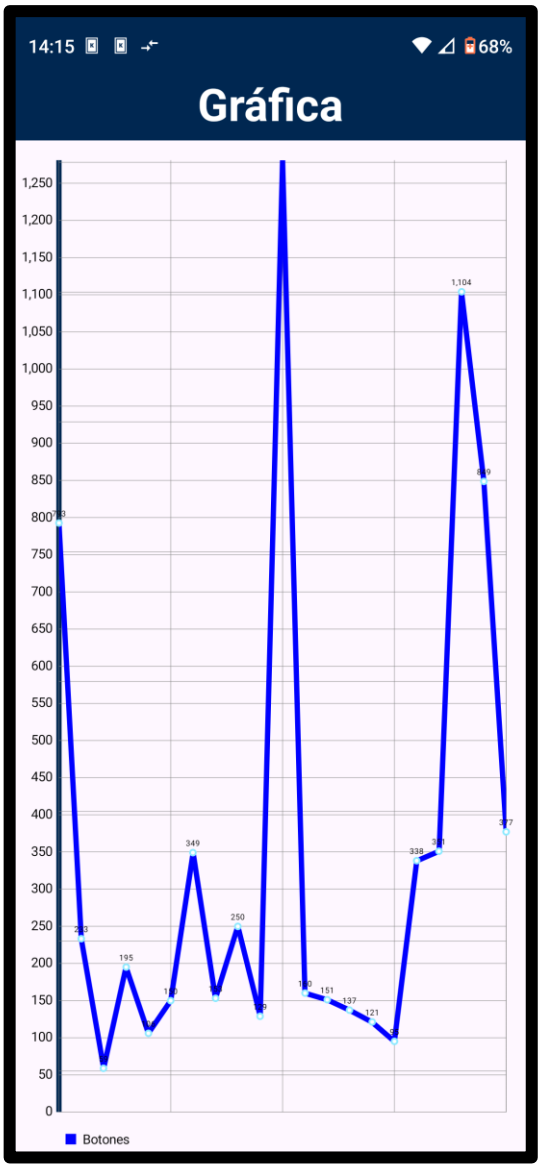


Figura 45.B: Pantalla de gráfica

En esta gráfica se puede hacer zoom horizontal y vertical, y se muestra la evolución del tiempo de pulsación con una línea azul que une los puntos de tiempo de cada botón.

B20. Exportar resultados

Pulsando el botón de exportar en la pantalla de resultados se puede guardar el resultado de la prueba y todos sus parámetros en un archivo de texto.

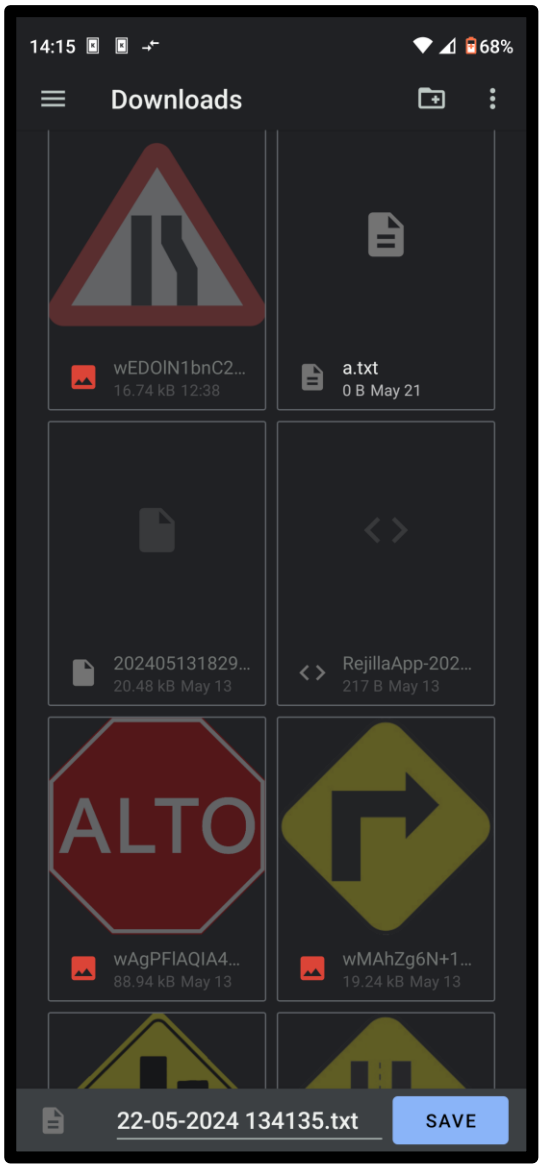


Figura 46.B: Selector de carpeta de Android

Al pulsar el botón de exportar se abrirá el selector de carpeta de Android, que pedirá al usuario la localización donde quiere guardar el archivo y el nombre del mismo.

El nombre se puede modificar, pero por defecto la aplicación usa como nombre del archivo la fecha de realización de la prueba que se quiere exportar.

B21. Subir resultados a MenPas

Para subir los resultados a MenPas únicamente es necesario acceder a la pantalla de detalles de la prueba que se quiere subir, ya sea desde el menú de historial o al acabar la prueba.

Pulsando el botón de Subir resultados, los resultados serán subidos a MenPas, y en caso de ser subidos correctamente se notificará al usuario. En caso de no poder subirlos por algún error, también se notificará al usuario.

B22. Historial de pruebas

En el menú de inicio de la aplicación existe una opción llamada Historial. Al pulsar este botón se abre la pantalla de historial de pruebas realizadas, desde la cual se pueden ver todas las pruebas realizadas ordenadas según su fecha de realización.

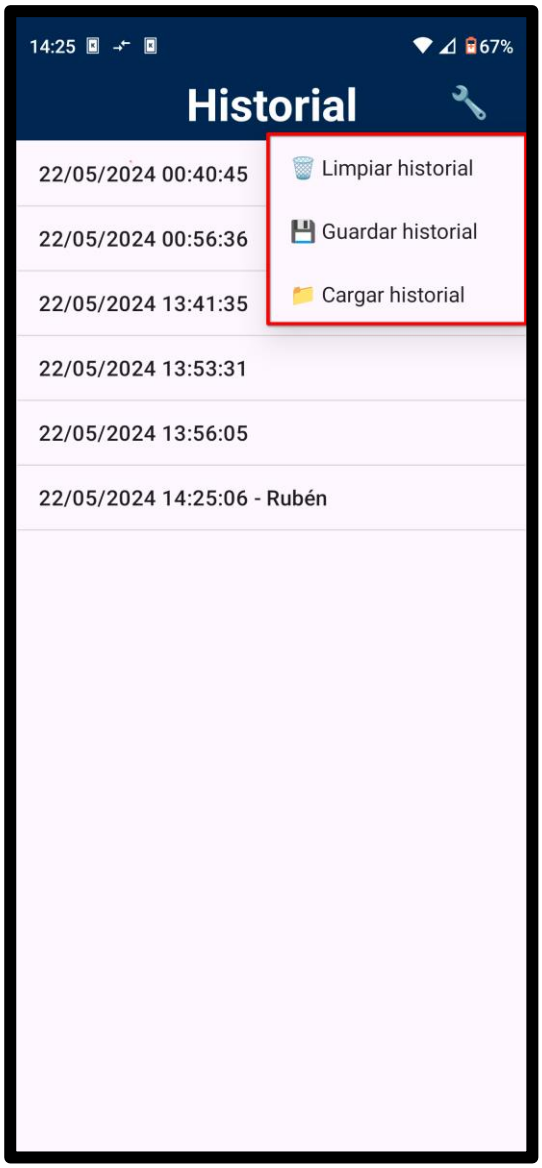


Figura 47.B: Pantalla de historial de pruebas, marcado en rojo el menú de gestión de historial

En esta pantalla aparecerá cada prueba guardada en la aplicación con su fecha de realización, seguida de, en caso de ser introducido, el nombre de la persona que la ha realizado.

Se dispone además de un menú accesible al pulsar el icono de llave inglesa en la esquina superior derecha, que se encarga de hacer diversas gestiones en el historial de pruebas. Las opciones presentes son:

- **Limpiar historial:** Elimina todas las pruebas del historial y limpia la base de datos.

- **Guardar historial:** Guarda una copia de la base de datos a un archivo con localización a seleccionar por el usuario.

- **Cargar historial:** Carga la copia realizada de la base de datos de un archivo del dispositivo.

B23. Pantalla de ajustes

En la pantalla de ajustes se dispone de dos opciones:

- **Abrir:** Permite abrir un archivo de configuración de la aplicación y cargar los parámetros de configuración de este.

- **Guardar:** Permite guardar a un archivo los parámetros de configuración de la aplicación.

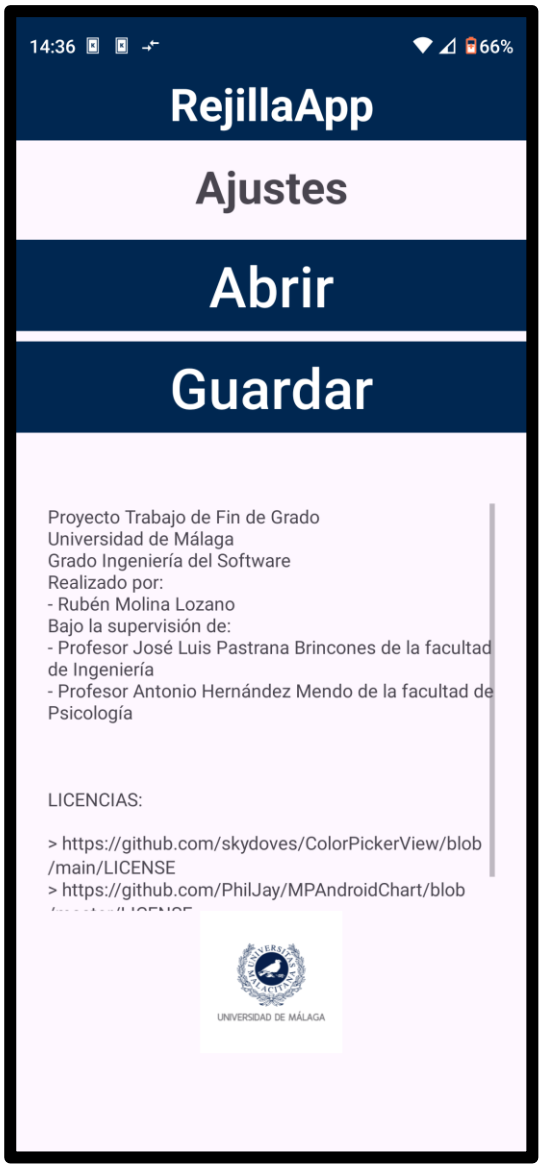


Figura 48.B: Pantalla de ajustes

Los parámetros que se guardarán y cargarán son:

- Todas las plantillas de todas las rejillas.
- Estado y configuraciones de las distracciones.
- Credenciales de MenPas.

- Datos del usuario.

- Colores de fondo guardados.

- Configuraciones de la función emparejar.

Por último, en la pantalla de ajustes se encuentra la información del proyecto y las licencias utilizadas.

Referencias

1. La atención: una compleja función cerebral - A. Estévez-González a, C. García-Sánchez b, C. Junqué
2. TDAH, un problema que empieza en la infancia - Clínica Universidad de Navarra from https://www.cun.es/enfermedades-tratamientos/cuidados-casa/diez-mitos-tdah-deficit-atencion-hiperactividad
3. Ditrendia. (n.d.). *Informe Ditrendia Mobile en España y en el Mundo 2020*.
4. ISO/IEC 9126 – Wikipedia from https://es.wikipedia.org/wiki/ISO/IEC_9126
5. Operating System Market Share Worldwide – statcounter from https://gs.statcounter.com/os-market-share
6. Android Statistics (2024) - Business of Apps from https://www.businessofapps.com/data/android-statistics/
7. *Android mobile app developer tools – Android developers*. (n.d.). Android Developers. Retrieved May 5, 2024, from https://developer.android.com
8. *iOS vs Android: Ventajas y comparativa*. (2023, April 10). Lowi. https://www.lowi.es/blog/ios-vs-android/
9. Wikimedia, C. de los proyectos. (2024, April 29). *Android Studio*. Wikipedia. https://es.wikipedia.org/wiki/Android_Studio
10. Cómo desarrollar una IU con Views. (n.d.). *Android Developers*. Retrieved May 24, 2024, from https://developer.android.com/studio/write/layout-editor?hl=es-419
11. Profile your app performance. (n.d.-b). *Android Developers*. Retrieved May 24, 2024, from https://developer.android.com/studio/profile

12. Contributors to Wikimedia projects. (2024, May 19). *Google Cloud Platform*. Wikipedia from https://en.wikipedia.org/wiki/Google_Cloud_Platform
13. Cloninger, E. (n.d.). *Android development tools for Eclipse*. Eclipse Plugins, Bundles and Products - Eclipse Marketplace | Eclipse Foundation. Retrieved May 24, 2024, from https://marketplace.eclipse.org/content/android-development-tools-eclipse
14. *Desarrollo de aplicaciones de Xamarin con Visual Studio*. (2018, April 23). Visual Studio. https://visualstudio.microsoft.com/es/xamarin/
15. Domínguez, S. (2021, January 29). Kotlin vs Java: Comparativa en profundidad. *OpenWebinars.Net*. https://openwebinars.net/blog/kotlin-vs-java/
16. Wikimedia, C. de los proyectos. (2024, April 8). *Java (lenguaje de programación)*. Wikipedia from https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)
17. Wikimedia, C. de los proyectos. (2023a, April 14). *Máquina virtual Java*. Wikipedia. https://es.wikipedia.org/wiki/M%C3%A1quina_virtual_Java
18. *Stack overflow developer survey 2023*. (n.d.). Stack Overflow. Retrieved May 25, 2024, from https://survey.stackoverflow.co/2023/#technology-most-popular-technologies
19. Wikimedia, C. de los proyectos. (2024, March 4). *Kotlin (lenguaje de programación)*. Wikipedia. https://es.wikipedia.org/wiki/Kotlin_(lenguaje_de_programaci%C3%B3n
20. Wikimedia, C. de los proyectos. (2022, September 15). *Null*. Wikipedia. https://es.wikipedia.org/wiki/Null
21. Wikimedia, C. de los proyectos. (2023b, November 11). *JetBrains*. Wikipedia. https://es.wikipedia.org/wiki/JetBrains
22. Wikimedia, C. de los proyectos. (2024, April 16). *Gradle*. Wikipedia. https://es.wikipedia.org/wiki/Gradle
23. Wikimedia, C. de los proyectos. (2024b, February 10). *Groovy (lenguaje de programación)*. Wikipedia. https://es.wikipedia.org/wiki/Groovy_(lenguaje_de_programaci%C3%B3n

24. Set up continuous integration. (n.d.). *Android Developers*. Retrieved May 25, 2024, from https://developer.android.com/studio/projects/continuous-integration
25. *Gradle documentation*. (n.d.). Gradle DSL Version 8.7. Retrieved May 25, 2024, from https://docs.gradle.org/current/dsl/org.gradle.api.tasks.bundling.Jar.html
26. *Gradle documentation*. (n.d.). Retrieved May 25, 2024, from https://docs.gradle.org/current/userguide/war_plugin.html#sec:war_default_settings
27. *Gradle documentation*. (n.d.). Retrieved May 25, 2024, from https://docs.gradle.org/current/userguide/ear_plugin.html
28. *Gradle documentation*. (n.d.). Retrieved May 25, 2024, from https://docs.gradle.org/current/userguide/build_file_basics.html
29. *Gradle documentation*. (n.d.). Retrieved May 25, 2024, from https://docs.gradle.org/current/userguide/build_environment.html
30. *Gradle documentation*. (n.d.). Retrieved May 25, 2024, from https://docs.gradle.org/current/userguide/core_dependency_management.html
31. *About sqlite*. (n.d.). Retrieved May 25, 2024, from https://www.sqlite.org/about.html
32. *SQLiteOpenHelper*. (n.d.). Android Developers. Retrieved May 25, 2024, from https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper
33. *SQLiteOpenHelper*. (n.d.). Android Developers. Retrieved May 25, 2024, from https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper#onCreate(android.database.sqlite.SQLiteDatabase
34. Cómo redactar un documento de requisitos de software – Asana from https://asana.com/es/resources/software-requirement-document-template
35. Refactorización - Wikipedia from https://es.wikipedia.org/wiki/Refactorizaci%C3%B3n
36. Maqueta – Wikipedia from https://es.wikipedia.org/wiki/Maqueta

37. Definición de Mockups - Imagine Apps from https://www.imagineapps.co/blog-posts-es/que-es-un-mockup#:~:text=Los%20mockups%20son%20representaciones%20est%C3%A1ticas,se%20ver%C3%A1%20el%20producto%20final

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA