Nerea Agud Lombarte

DISEÑO, DESARROLLO Y EVALUACIÓN DE UNA APLICACIÓN ANDROID PARA LA ESTIMULACIÓN COGNITIVA DE AFECTADOS DE ALZHEIMER

TRABAJO DE FINAL DE GRADO

Dirigido por Dr. Pere Millán Marco

Doble Grado en Ingeniería Informática y en Biotecnología

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2023

Resum.

Aquest Treball de Final de Grau se centra en el disseny i el desenvolupament d'una aplicació Android destinada a optimitzar la gestió de les sessions terapèutiques per a afectats d'Alzheimer. Formant part del programa d'Aprenentatge Servei de la URV, aquesta iniciativa s'emprèn amb un enfocament d'aportació social, col·laborant directament amb l'Associació de Familiars i Afectats d'Alzheimer de les Terres de l'Ebre (AFATE).

La finalitat de l'aplicació és oferir als terapeutes una eina eficient i intuïtiva que faciliti la creació i la gestió d'exercicis personalitzats, a més de permetre un seguiment detallat del progrés de cada pacient. Així, s'aconsegueix una optimització del temps dedicat a la programació d'activitats i un increment en la qualitat del seguiment i l'atenció que els pacients d'Alzheimer reben.

L'aplicació inclou un accés restringit per als administradors, el que possibilita la gestió a nivell organitzatiu. La plataforma es configura com una solució integral per a AFATE, abastant des de la personalització del tractament, fins al seguiment del pacient i la gestió administrativa.

El procés de desenvolupament s'ha regit pels estàndards de l'enginyeria de programari, incloent-hi fases d'anàlisi de requisits, disseny, implementació i avaluació. L'ús d'Android Studio i Kotlin per al desenvolupament, i de Firebase per a l'emmagatzematge i la gestió de dades, ha permès la creació d'una aplicació sòlida i de fàcil manteniment.

Aquest projecte exemplifica com l'enginyeria informàtica pot generar solucions tecnològiques innovadores que beneficien els professionals sanitaris i milloren la qualitat de vida de les persones afectades per l'Alzheimer. D'aquesta manera, es fa palès el compromís de la disciplina amb la generació d'impacte social positiu.

Resumen.

Este Trabajo de Final de Grado se centra en el diseño y desarrollo de una aplicación Android destinada a optimizar la gestión de las sesiones terapéuticas para afectados de Alzheimer. Formando parte del programa de Aprendizaje Servicio de la URV, esta iniciativa se emprende con un enfoque de aporte social, colaborando directamente con la Asociación de Familiares y Afectados de Alzheimer de las Tierras del Ebro (AFATE).

La finalidad de la aplicación es ofrecer a los terapeutas una herramienta eficiente e intuitiva que facilite la creación y gestión de ejercicios personalizados, además de permitir un seguimiento detallado del progreso de cada paciente. Así, se consigue una optimización del tiempo dedicado a la programación de actividades y un incremento en la calidad del seguimiento y la atención que los pacientes de Alzheimer reciben.

La aplicación incluye un acceso restringido para los administradores, lo que posibilita la gestión a nivel organizativo. La plataforma se configura como una solución integral para AFATE, abarcando desde la personalización del tratamiento hasta el seguimiento del paciente y la gestión administrativa.

El proceso de desarrollo se ha regido por los estándares de la ingeniería de software, incluyendo fases de análisis de requisitos, diseño, implementación y evaluación. El uso de Android Studio y Kotlin para el desarrollo, y de Firebase para el almacenamiento y gestión de datos, ha permitido la creación de una aplicación sólida y de fácil mantenimiento.

Este proyecto ejemplifica cómo la ingeniería informática puede generar soluciones tecnológicas innovadoras que benefician a los profesionales sanitarios y mejoran la calidad de vida de las personas afectadas por el Alzheimer. De esta forma, se pone de manifiesto el compromiso de la disciplina con la generación de impacto social positivo.

Abstract.

This Final Degree Project focuses on the design and development of an Android application aimed at optimizing the management of therapeutic sessions for Alzheimer's sufferers. As part of the URV's Service Learning program, this initiative is undertaken with a focus on social contribution, collaborating directly with the Association of Relatives and People Affected by Alzheimer's Disease of the Ebro Lands (AFATE).

The aim of the application is to offer therapists an efficient and intuitive tool that facilitates the creation and management of personalized exercises, as well as allowing detailed monitoring of each patient's progress. In this way, the time dedicated to programming activities is optimized and the quality of the monitoring and care that Alzheimer's patients receive is increased.

The application includes restricted access for administrators, which enables management at an organizational level. The platform is configured as a comprehensive solution for AFATE, ranging from personalization of treatment to patient monitoring and administrative management.

The development process has been governed by software engineering standards, including phases of requirements analysis, design, implementation and evaluation. The use of Android Studio and Kotlin for development, and Firebase for data storage and management, has allowed the creation of a robust and easy-to-maintain application.

This project exemplifies how computer engineering can generate innovative technological solutions that benefit healthcare professionals and improve the quality of life of people affected by Alzheimer's disease. In this way, it demonstrates the discipline's commitment to generating positive social impact.

Índice

Índice de tablas

Índice de figuras

1 Introducción

Este Trabajo de Final de Grado (**TFG**[1]) es parte del Programa de Aprendizaje y Servicio (**APS**[2]), que se basa en una metodología que combina el aprendizaje de conocimientos, habilidades y valores con el servicio a la comunidad. Esta integración permite llevar a cabo un proyecto educativo que busca brindar un servicio directo a la sociedad con el propósito de transformarla y mejorarla.

En este caso la propuesta desarrollada se ha llevado a cabo junto a la Associació de Familiars d'Alzheimer Terres de l'Ebre (**AFATE**[3]). Esta asociación nace en 1993 debido a la falta de información y de recursos que había en la comarca del Baix Ebre para enfermos y familiares de Alzheimer. Se constituyó como Asociación en el año 2004 y desde entonces ha continuado con sus actividades y ampliando los "Talleres de Estimulación cognitiva" en los pueblos de la comarca.

En la actualidad, la intervención se basa primordialmente en hojas impresas de ejercicios (Anexo 1), lo que representa un gasto constante en materiales y esfuerzo. Si bien se utilizan aplicaciones de pago ampliamente difundidas como como Gradior (https://www.gradior.es/) y NeuronUP (https://www.neuronup.com/) en tabletas Android, estas presentan desafíos. No solo suponen un coste elevado, sino que, a menudo, sus contenidos no se adaptan a las realidades culturales y contextuales de los pacientes, generando actividades de temáticas distantes y poco familiares para los usuarios. También se encuentra una barrera en términos lingüísticos, ya que muchos residentes mayores de Terres de l'Ebre hablan catalán, y estas aplicaciones funcionan en castellano. Estas herramientas también restringen la continuidad de la terapia al limitar la práctica a las sesiones de terapia.

A raíz de estas limitaciones surge la necesidad de crear una aplicación propia, donde los terapeutas puedan configurar los ejercicios que quieren para su grupo de pacientes y que consideren que van a ayudarles de la mejor forma posible. Como las tabletas utilizadas son Android, se decide desarrollar la aplicación para Android.

Figura 1. Logotipo AFATE

[1] TFG = Trabajo de Final de Grado

[2] APS = Aprenentatge Servei

[3] AFATE = Associació de Familiars d'Alzheimer Terres de l'Ebre

1.1 Objetivos formativos

Desde el punto de vista formativo, este proyecto busca alcanzar los siguientes objetivos:

- **Aplicación de conocimientos teóricos**: Utilizar los conocimientos teóricos y técnicos adquiridos durante el programa de estudios de Ingeniería Informática para desarrollar una aplicación práctica y funcional. Esto implica el uso de lenguajes de programación, diseño de interfaces de usuario, gestión de bases de datos y aplicación de buenas prácticas de codificación.
- **Desarrollo de habilidades técnicas**: Profundizar en el manejo de Android Studio y Kotlin, así como en el uso de Firebase para la gestión de datos. Esto implica también el aprendizaje y aplicación de técnicas y estrategias de desarrollo de software eficiente y sostenible.
- **Gestión de proyectos**: Aprender a manejar las etapas del desarrollo de un proyecto de software, desde la recopilación y análisis de requisitos, hasta el diseño, implementación y pruebas. Esto también incluye la capacidad de planificar y gestionar el tiempo y los recursos de manera efectiva.
- **Trabajo colaborativo y comunicación**: Colaborar estrechamente con los terapeutas y el personal de AFATE para entender sus necesidades y requerimientos, y traducirlos en funciones y características de la aplicación. Esto implica el desarrollo de habilidades de comunicación y empatía, así como la capacidad de trabajar eficazmente en un equipo multidisciplinario.
- **Impacto social y ética**: Comprender y aplicar los principios éticos en el desarrollo de software, especialmente en términos de privacidad y seguridad de los datos de los pacientes. Además, apreciar el potencial que la ingeniería informática tiene para generar un impacto social positivo, y aspirar a contribuir a ese impacto en futuros proyectos.
- **Aprendizaje autónomo y resolución de problemas**: Desarrollar la capacidad de aprender de manera autónoma nuevas herramientas, técnicas y lenguajes de programación, así como la habilidad para resolver problemas técnicos y desafíos que puedan surgir durante el desarrollo del proyecto.

2 Descripción

La aplicación propuesta en este Trabajo de Final de Grado es una herramienta de apoyo, diseñada para optimizar la labor terapéutica en la asociación AFATE, orientada a la mejora de la calidad de vida de los pacientes afectados por esta enfermedad. Su principal finalidad es facilitar la creación y gestión de sesiones de actividades personalizadas para cada paciente, de modo que los terapeutas puedan brindar una atención específica y adaptada a sus necesidades y vivencias.

La aplicación, desarrollada en catalán, castellano e inglés, incorpora una variedad de ejercicios predefinidos de estimulación cognitiva. Estos ejercicios, como la sopa de letras o el emparejamiento de imágenes y palabras, pueden ser personalizados por los terapeutas de acuerdo a las particularidades de cada grupo o paciente. Esta flexibilidad en la configuración de los juegos permite la creación de actividades novedosas y menos repetitivas, además de poder adaptarlos a las experiencias de vida del paciente, potenciando así su implicación y motivación.

Una vez creadas las actividades, estas se agrupan en sesiones de trabajo que se asignan a cada paciente. Los usuarios, a través de sus respectivos perfiles, pueden acceder a sus sesiones y realizar los ejercicios en el tiempo que se les ha establecido.

Además de su enfoque en la personalización, otro aspecto relevante de la aplicación es su capacidad para recopilar y almacenar información sobre el rendimiento del paciente. La aplicación registra las respuestas de los usuarios, los errores cometidos y el tiempo requerido para completar cada ejercicio, proporcionando así a los terapeutas un seguimiento detallado del progreso del paciente y de cómo la enfermedad está afectando a sus capacidades cognitivas. Esta información permite a los terapeutas ajustar el nivel de dificultad de los ejercicios y personalizar aún más el tratamiento.

La aplicación presenta tres entornos funcionales diferentes en función del tipo de usuario: el entorno del terapeuta, donde se crean y gestionan los juegos, las sesiones de desarrollo y la consulta de los historiales de los pacientes; el entorno del paciente, donde se accede a las sesiones actuales y se realizan los ejercicios; y el entorno del administrador, que se encarga de la gestión de usuarios y entidades.

Por tanto, esta aplicación no solo aporta una herramienta de apoyo valiosa para los terapeutas, sino que también ofrece a los pacientes un espacio interactivo y adaptado a sus necesidades, contribuyendo a mejorar su experiencia y su compromiso con el tratamiento.

2.1 Ciclo de vida del software

Para desarrollar cualquier software es necesario seguir un ciclo de vida para poder ajustar plazos, recursos, calidad y mantener un control. En este caso se escogió utilizar un ciclo de vida en cascada, donde en cada fase se va obteniendo un resultado (documento o código) que es la base para la fase siguiente (*Figura 2*). Estas serán las fases que se seguirán también en esta documentación.

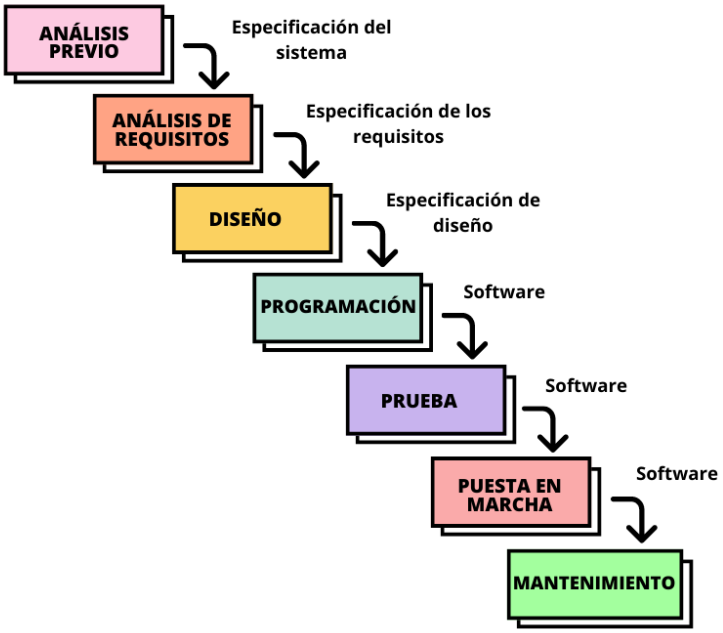


Figura 2. Ciclo de vida del software

Respecto a la metodología utilizada, al ser este un proyecto más pequeño, tener un periodo de desarrollo corto y necesitar flexibilidad a la hora de implementar las modificaciones que considere la organización, se escogió utilizar la metodología *Agile*. Esta metodología utiliza S*prints* cortos, que son ciclos de desarrollo iterativo e incremental en los que se trabaja con un conjunto de funcionalidades durante un período de tiempo determinado y corto (unas 2 semanas aproximadamente). Esto permite que puedan ser entregadas y revisarse con el cliente, fomentando la colaboración y retroalimentación constante para poder adaptarse a posibles cambios o variaciones.

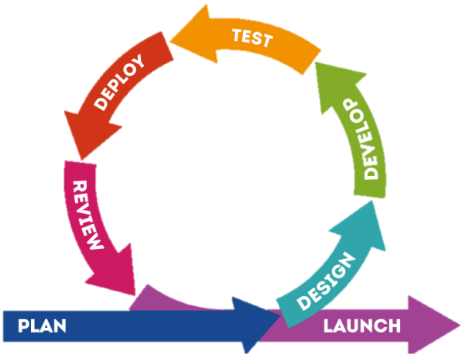


Figura 3. Ciclo de la metodologia *Agile*

2.2 Análisis de la competencia de software

En el entorno actual, existen varias aplicaciones y plataformas que buscan abordar los desafíos asociados con el manejo y tratamiento de los pacientes con Alzheimer. A continuación, se analizan dos de los competidores más notables: Gradior y NeuronUP.

2.2.1 Gradior

Gradior es una suite de software de rehabilitación cognitiva que está diseñada para ser utilizada en centros de día, residencias y otros entornos de atención médica. Proporciona una variedad de ejercicios que pueden ser personalizados para cada usuario en función de su nivel de habilidad y capacidades. Gradior permite a los profesionales de la salud rastrear y monitorizar el progreso del usuario, lo que puede ser útil para adaptar las intervenciones y estrategias de tratamiento a las necesidades individuales de cada paciente. Sin embargo, no cuenta con una aplicación móvil dedicada para facilitar el acceso fuera de los entornos de atención, ni una opción de auto-registro para los pacientes.

2.2.2 NeuronUP

NeuronUP es una plataforma web que ofrece una extensa biblioteca de actividades y ejercicios destinados a la rehabilitación y estimulación cognitiva. La plataforma ofrece una gran variedad de actividades que se adaptan a diferentes niveles de dificultad, lo que permite un enfoque personalizado del tratamiento. A través de NeuronUP, los profesionales pueden rastrear el progreso de los usuarios, lo que puede informar de futuras estrategias de tratamiento y terapia. Aunque NeuronUP ofrece una gran cantidad de contenido y es accesible a través de la web, la ausencia de una aplicación móvil dedicada podría limitar el acceso de los usuarios. Además, no ofrece una opción de auto-registro para los pacientes.

2.2.3 Ventajas y mejoras de esta aplicación

La aplicación propuesta en este TFG ofrece ventajas y mejoras significativas en comparación con las soluciones existentes. En primer lugar, estará disponible en catalán, lo que la hará más accesible y funcional para los usuarios en regiones donde se habla este idioma, llenando un vacío importante en el mercado actual.

Además, se permitirá a los terapeutas generar ejercicios personalizados basados en las vivencias y la personalidad de los pacientes. Por ejemplo, pueden crear sopas de letras con palabras cercanas a los usuarios, como los nombres de los pueblos de la zona. Esta capacidad de personalización a nivel de vida del paciente va más allá de la mera adaptación al nivel de habilidad, proporcionando una experiencia más personalizada y significativa.

Finalmente, la aplicación contará con una versión móvil que facilitará su uso fuera de los entornos de atención médica, además de la opción de auto-registro para los pacientes, incrementando así la accesibilidad y la autonomía de los usuarios.

2.3 Legislación del proyecto

Dada la naturaleza de la aplicación desarrollada, destinada a la gestión de datos sensibles de pacientes afectados por Alzheimer y trabajadores, se han tenido en cuenta diversas regulaciones y legislaciones en su diseño e implementación.

Ley Orgánica de Protección de Datos Personales y Garantía de los Derechos Digitales (LOPDGDD[4]): Este reglamento español, en línea con el Reglamento General de Protección de Datos (RGPD[5]) de la Unión Europea, establece obligaciones estrictas en cuanto al manejo, almacenamiento y transferencia de datos personales. Dado que la

4 LOPDGDD = Ley Orgánica de Protección de Datos Personales y Garantía de los Derechos Digitales

5 RGDP: Reglamento General de Protección de Datos

aplicación maneja datos personales sensibles, se han tomado todas las medidas necesarias para garantizar el cumplimiento de estas normativas [1] [2].

Ley General de Derechos de las Personas con Discapacidad y de su Inclusión Social: Debido a que la aplicación está destinada a personas con Alzheimer, se han tenido en cuenta las disposiciones de esta Ley en el diseño de la aplicación, para asegurar la accesibilidad y la usabilidad para los usuarios con discapacidades[3].

Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico (LSSICE[6]): Se han considerado las disposiciones de esta Ley en términos de la prestación de servicios de la sociedad de la información, especialmente en relación con la obligación de proporcionar información clara y transparente a los usuarios[4].

Ley de Propiedad Intelectual: Se ha tenido en cuenta esta Ley en relación con el uso de cualquier contenido con derechos de autor en la aplicación[5].

6 LSSICE = Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico

3 Análisis de los requisitos

En este apartado se presenta la documentación de los requisitos funcionales (guiones de usuarios expresados y documentados como casos de uso) y análisis de los requisitos no funcionales.

3.1 Guiones

3.1.1 Guion del paciente

Los pacientes pueden registrarse en la plataforma para participar en las sesiones de terapia. Una vez registrados, tienen la posibilidad de unirse a un grupo de terapia mediante un código proporcionado por su terapeuta. Los pacientes pueden ver las sesiones de ejercicios que les han sido asignadas por su terapeuta y así poder realizar las actividades. Adicionalmente, tienen la opción de editar información personal en su perfil.

3.1.2 Guion del terapeuta

Los terapeutas tienen la capacidad de administrar las sesiones de terapia y gestionar a sus pacientes. Pueden ver los grupos que tienen asignados y los pacientes que forman parte de cada grupo. Los terapeutas también tienen la capacidad de crear nuevos ejercicios de terapia y, a partir de estos, crear nuevas sesiones de terapia. Estas sesiones pueden ser asignadas a varios o a un paciente en particular. Los terapeutas pueden modificar las sesiones, controlando así sus actividades. Además, pueden unirse a nuevos grupos mediante un código o retirarse de los grupos cuando sea necesario. También pueden ver el historial de sus pacientes para llevar un seguimiento de su evolución y adaptar las sesiones a sus necesidades. Adicionalmente, también tienen la opción de editar su propio perfil.

3.1.3 Guion del administrador

El administrador tiene el control total del sistema. Este es el encargado de registrar a los terapeutas en la plataforma, proporcionándoles una contraseña temporal que posteriormente pueden cambiar. Tiene la capacidad de desactivar las cuentas de los terapeutas y pacientes cuando sea necesario, garantizando así la seguridad y el correcto funcionamiento del sistema. El administrador también tiene la responsabilidad de gestionar los grupos de terapia y crear los nuevos que sean necesarios. Además, puede revisar las actividades y las sesiones creadas por los terapeutas para asegurar que se ajusten a las normas y estándares de la plataforma y, si es necesario, eliminarlas. Por último, el administrador puede acceder a la información de terapeutas y pacientes para tener una visión general y control de la plataforma.

3.1.4 Otras funciones

Además de las funciones específicas para cada tipo de usuario, hay otras características generales que mejoran la experiencia de uso de la aplicación para todos los usuarios. Por ejemplo, los usuarios tienen la opción de cambiar su contraseña si lo consideran necesario para mantener la seguridad de su cuenta. La aplicación también permite a los usuarios mantener su perfil actualizado con información relevante.

3.2 Diagrama de clases

El diagrama de clases es una representación visual del diseño y la estructura de las clases que componen el proyecto. Proporciona una visión general de cómo se relacionan las clases entre sí, los atributos que contienen y el tipo de las relaciones (asociaciones, agregaciones y composiciones). Este diagrama puede observarse en la *Figura 4*.

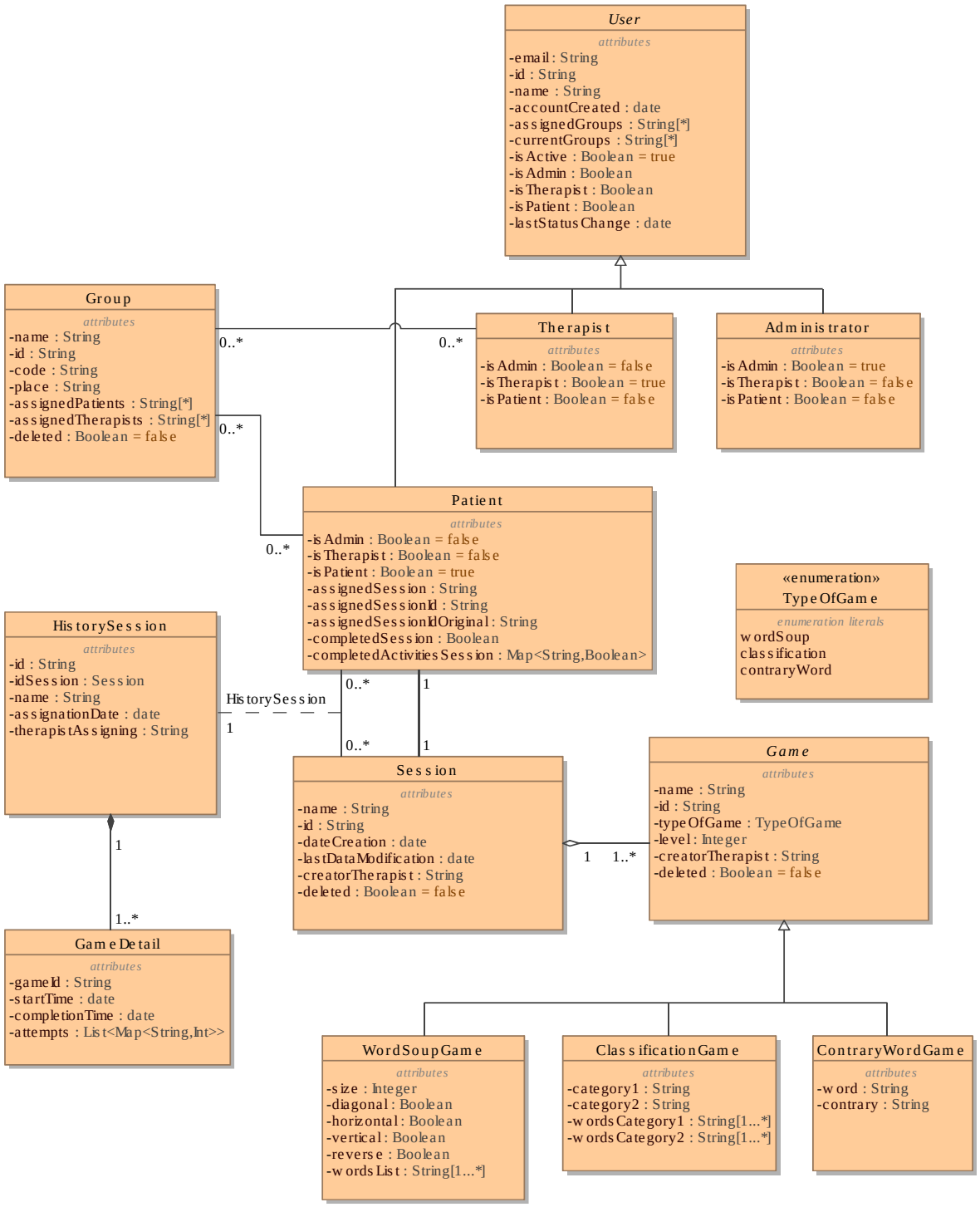


Figura 4. Diagrama de clases

3.2.1 Diseño de clases

Clase ‘Game’ (Abstracta) y sus subclases:

- La clase ‘Game’ actúa como una clase base abstracta para diferentes tipos de juegos en el sistema. Esta clase contiene atributos comunes que son relevantes para todos los tipos de juegos, como *name*, *id*, *level*, *typeOfGame*, entre otros. Nunca se instancia directamente un objeto de tipo ‘Game’ ya que solo sirve como estructura generalizada.
- Las subclases (‘ClassificationGame’, ‘WordSoupGame’, ‘ContraryWordGame’) heredan de ‘Game’ y representan tipos específicos de juegos, cada una con sus propios atributos únicos.
- El campo *typeOfGame* actúa como un discriminador para determinar el tipo específico de juego. Los valores que este campo puede tener se han definido en la clase de enumeración ‘TypeOfGame’.

Clase ‘Group’:

- Representa grupos en el sistema. Los grupos tienen un conjunto de pacientes y terapeutas asignados, representados por las listas *assignedPatients* y *assignedTherapists*.

Clase ‘User’ (Abstracta) y sus subclases: ‘Patient’ y ‘Therapist’:

- Las clases ‘Administrator’, ‘Therapist’ y ‘Patient’ representan a usuario con diferentes permisos y funcionalidades en la aplicación. Estos son los administradores, terapeutas y pacientes, respectivamente. Las tres clases tienen atributos como *id*, *name*, *email*, entre otros.
- Las clases ‘Patient’ y ‘Therapist’ también mantienen información sobre los grupos a los que han sido asignados y a los que están asignados actualmente (*assignedGroups* y *currentGroups*).
- La clase ‘Patient’ también tiene atributos para gestionar las sesiones asignadas y el desarrollo de los ejercicios en la sesión como *assignedSession*, *completedSession* o el atributo *assignedSessionId* que se utiliza para almacenar en su historial los resultados de la realización de la sesión asignada.

Clase ‘Session’:

- Representa sesiones que contienen una serie de actividades o juegos. Cada sesión tiene un conjunto de actividades, representado por la lista *activities*.

Clase ‘HistorySession’ y ‘GameDetail’:

- ‘HistorySession’ representa la asignación de una sesión a un paciente y contiene información sobre cuándo se asignó la sesión y qué terapeuta la asignó.
- ‘GameDetail’ representa el detalle de cómo un paciente lleva a cabo un ejercicio específico de la sesión. Captura información como el número de intentos o la fecha y hora de inicio y finalización del ejercicio.

3.2.2 Relaciones y multiplicidades

Para este diagrama se han utilizado diversos tipos de relaciones:

- **Asociación**: Es una relación básica entre clases. Por ejemplo, entre ‘Patient’ y ‘Group’, donde un paciente puede pertenecer a múltiples grupos y un grupo puede tener múltiples pacientes.
- **Agregación**: Se representa con un rombo transparente y muestra una relación "todo-parte" (indica que una clase (el "todo" o "contenedor") contiene o está compuesta por instancias de otra clase (la "parte")). En este diagrama, la relación entre ‘Session’ y ‘Game’ es una agregación. Esto significa que una sesión es un conjunto de juegos, pero los juegos pueden existir independientemente de la sesión.
- **Composición**: Es una forma más fuerte de agregación, representada con un rombo lleno. Indica que la existencia de una clase está estrictamente vinculada a la existencia de otra. Por ejemplo, ‘HistorySession’ tiene una composición con ‘GameDetail’, lo que significa que, si se elimina una ‘HistorySession’, también se deben eliminar sus detalles de ejercicios asociados.

3.3 Requisitos funcionales

Los requisitos funcionales delinean las capacidades y características específicas que el sistema debe ofrecer con el fin de cumplir con las expectativas y necesidades de los usuarios. Estas características son representadas a través de casos de uso, que muestran de manera detallada las acciones y operaciones que cada usuario puede emprender dentro de la aplicación.

Para facilitar la comprensión y el análisis, los casos de uso han sido clasificados y presentados según los roles de usuario: administrador, terapeuta y paciente. Cada diagrama se centra en las actividades donde el respectivo usuario desempeña el papel de actor principal. En ciertas ocasiones, otros actores aparecen como secundarios, reflejando así las interacciones y dinámicas que se dan entre los diferentes roles en el sistema. A continuación, se representan dichos diagramas.

3.3.1 Diagrama de casos de uso del administrador

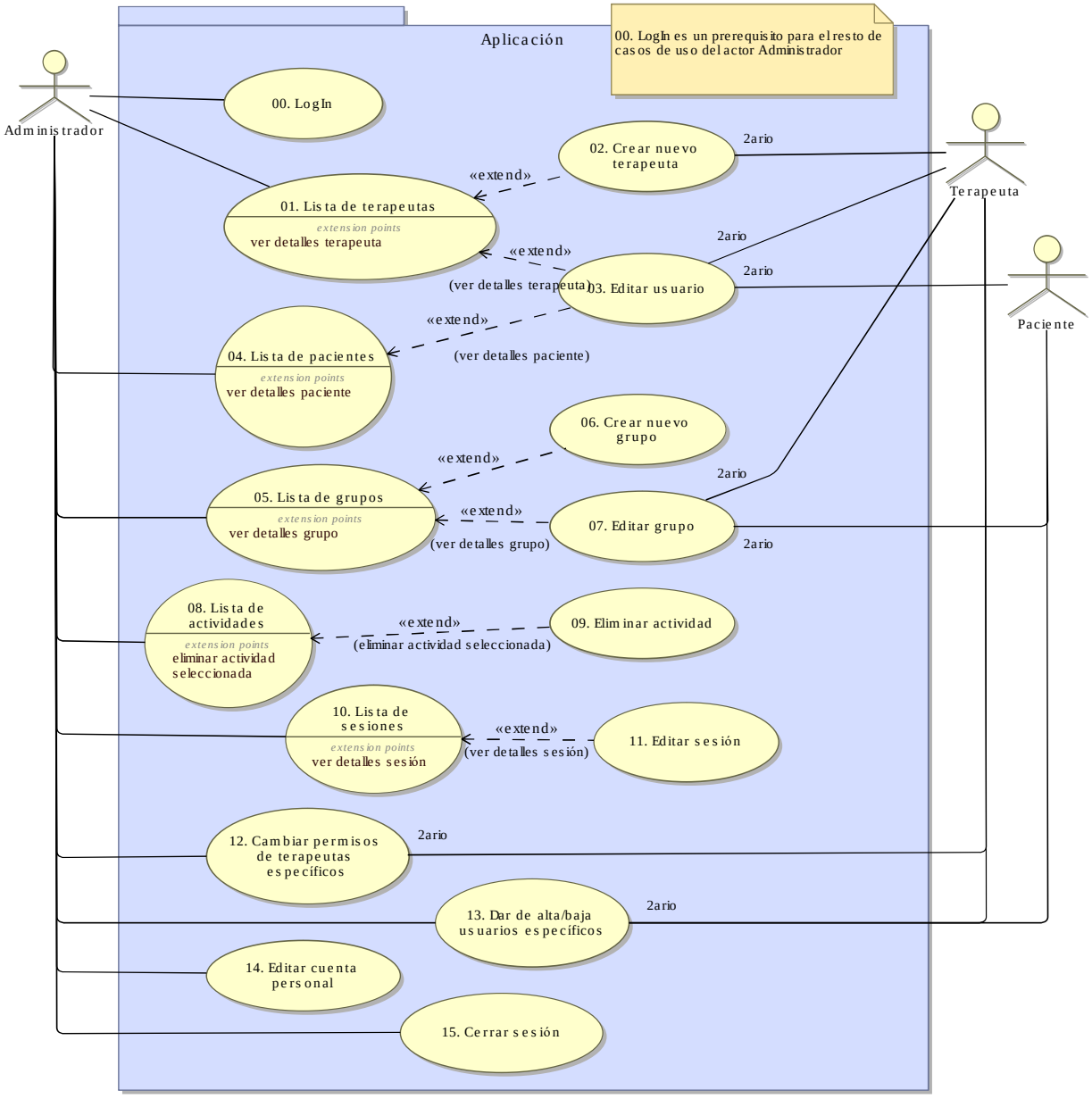


Figura 5. Diagrama de casos de uso del usuario “Administrador”

3.3.2 Diagrama de casos de uso del terapeuta

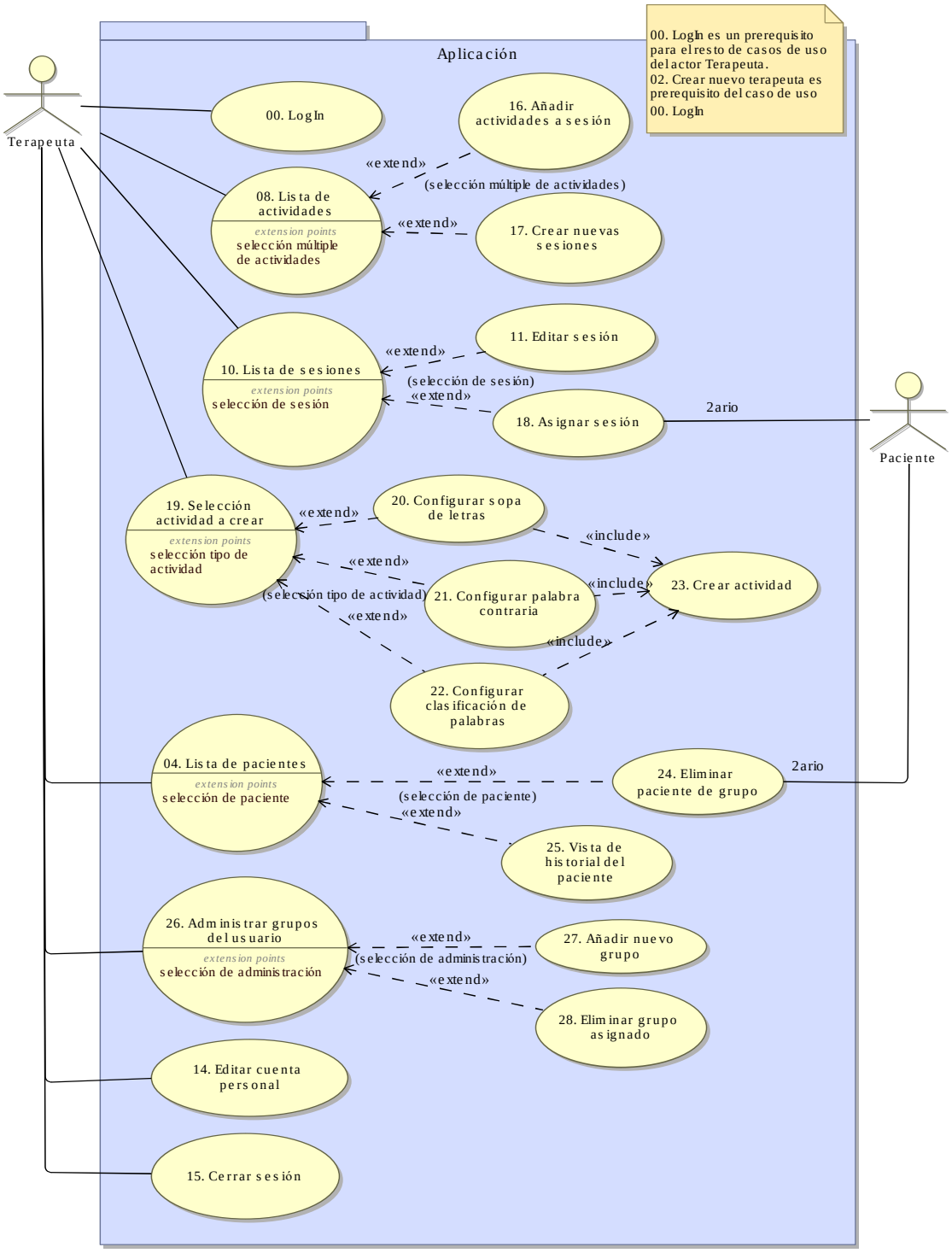


Figura 6. Diagrama de casos de uso del usuario “Terapeuta”

3.3.3 Diagrama de casos de uso del paciente

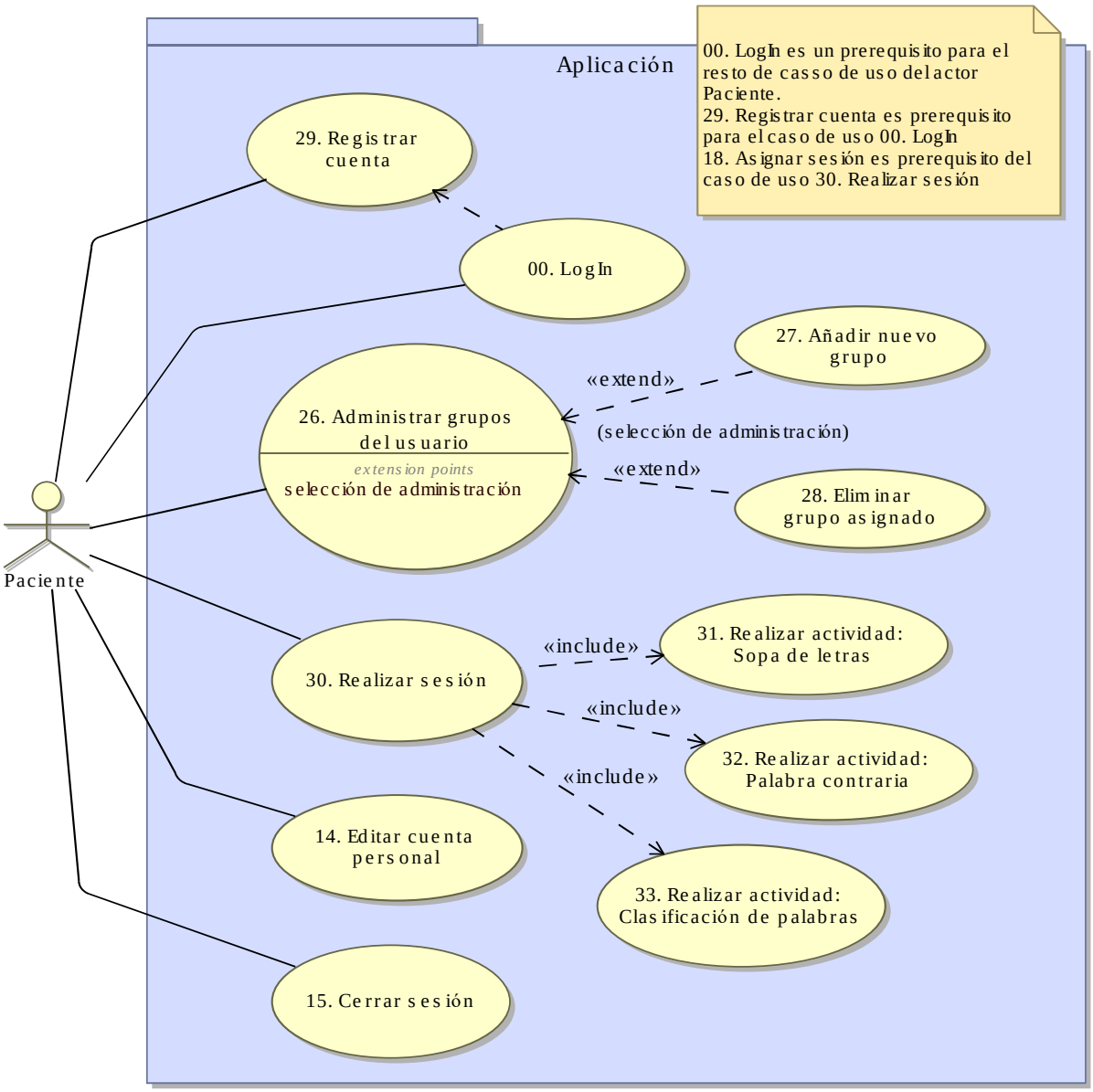


Figura 7. Diagrama de casos de uso del usuario “Paciente”

3.3.4 Especificación textual de los casos de uso

Se utiliza la definición de "usuario" para mencionar a todos los actores del caso de uso específico.

Título	00. Login
Actor	Administrador, Terapeuta, Paciente
Prerrequisitos	El usuario tiene que estar identificado. El Terapeuta debe tener la cuenta creada por el Administrador. El Paciente debe haber registrado la cuenta.
Descripción	El usuario debe introducir el email asociado a su cuenta y la contraseña para poder acceder al resto de funcionalidades de la aplicación.

Tabla 1. Requisito 00. Login

Título	01. Lista de terapeutas
Actor	Administrador
Prerrequisitos	El administrador debe estar identificado
Descripción	Se visualiza un listado con todos los terapeutas registrados en la aplicación. Cada terapeuta está representado por una tarjeta con un icono, el email y el nombre del terapeuta. Los terapeutas dados de baja tienen la tarjeta de color gris y el resto de color blanco. Al seleccionar un terapeuta se muestra una ventana flotante que permite ver toda la información detallada del terapeuta y acceder a editar su cuenta. Existe un botón que permite acceder a la creación de nuevos terapeutas.

Tabla 2. Requisito 01. Lista de terapeutas

Título	02. Crear nuevo terapeuta
Actor	Administrador, Terapeuta (secundario)
Prerrequisitos	El usuario debe estar identificado
Descripción	Se visualizan varios campos que el Administrador debe rellenar con los datos del terapeuta (nombre, email y contraseña) para poder crear la cuenta.

Tabla 3. Requisito 02. Crear nuevo terapeuta

Título	03. Editar usuario
Actor	Administrador, Terapeuta (secundario), Paciente (secundario)
Prerrequisitos	El usuario debe estar identificado
Descripción	Se visualiza los detalles de un determinado usuario. Permite eliminar al usuario de determinados grupos o cambiar el estado de su cuenta (dar de alta o dar de baja).

Tabla 4. Requisito 03. Editar usuario

Título	04. Lista de pacientes
Actor	Administrador, Terapeuta
Prerrequisitos	El usuario debe estar identificado
Descripción	Se visualiza un listado con todos los pacientes registrados en la aplicación. Cada paciente está representado por una tarjeta con un icono, el email, el nombre del paciente y los grupos a los que está asignado actualmente. Los pacientes dados de baja tienen la tarjeta de color gris y el resto de color blanco. Al seleccionar un paciente se muestra una ventana flotante que permite ver toda su información detallada y acceder a editar su cuenta. El terapeuta solo puede ver en el listado de pacientes, aquellos pacientes que formen parte de grupos que él mismo tiene asignados y cuya cuenta esté activa. El administrador puede ver a todos los pacientes.

Tabla 5. Requisito 04. Lista de pacientes

Título	05. Lista de grupos
Actor	Administrador
Prerrequisitos	El administrador debe estar identificado
Descripción	Se visualiza un listado con todos los grupos creados en la aplicación. Cada grupo está representado por una tarjeta con un icono, el nombre del grupo, el lugar y el código de acceso del grupo. Los grupos eliminados tienen la tarjeta de color gris y el resto de color blanco. Al seleccionar un grupo se muestra una ventana flotante que permite ver toda la información detallada del grupo y acceder a editar sus datos. Se muestra la opción de crear un nuevo grupo.

Tabla 6. Requisito 05. Lista de grupos

Título	06. Crear nuevo grupo
Actor	Administrador
Prerrequisitos	El administrador debe estar identificado
Descripción	Introduciendo un nombre de grupo y un lugar, permite crear un nuevo grupo. El código de acceso se genera de forma aleatoria por la aplicación.

Tabla 7. Requisito 06. Crear nuevo grupo

Título	07. Editar grupo
Actor	Administrador, Terapeuta (secundario), Paciente (secundario)
Prerrequisitos	El usuario debe estar identificado.

Descripción	Permite visualizar toda la información detallada del grupo: nombre, lugar, código, terapeutas asignados y pacientes asignados. También permite generar un nuevo código de grupo, cambiar el lugar del grupo o eliminar el grupo.

Tabla 8. Requisito 07. Editar grupo

Título	08. Lista de actividades
Actor	Administrador, Terapeuta
Prerrequisitos	El usuario debe estar identificado
Descripción	Se visualiza un listado con todas las actividades creadas en la aplicación. Cada actividad está representada por una tarjeta con un icono, el nombre de la actividad, el tipo de actividad y el terapeuta creador. Las actividades eliminadas tienen la tarjeta de color gris y el resto de color blanco. Al seleccionar una actividad, se muestra una ventana flotante que permite ver toda la información detallada y poder eliminar la actividad.

Tabla 9. Requisito 08. Lista de actividades

Título	09. Eliminar actividad
Actor	Administrador
Prerrequisitos	El administrador debe estar identificado
Descripción	Permite al administrador eliminar una actividad. La eliminación de una actividad hace que esta deje de ser visible para los terapeutas.

Tabla 10. Requisito 09. Eliminar actividad

Título	10. Lista de sesiones
Actor	Administrador, Terapeuta
Prerrequisitos	El usuario debe estar identificado
Descripción	Se visualiza un listado con todas las sesiones creadas en la aplicación. Cada sesión está representada por una tarjeta con un icono, el nombre de la sesión y las actividades que incluye. Las sesiones eliminadas tienen la tarjeta de color gris y el resto de color blanco. Al seleccionar una sesión se muestra una ventana flotante que permite ver toda la información detallada y poder editar la sesión.

Tabla 11. Requisito 10. Lista de sesiones

Título	11. Editar sesión
Actor	Administrador, Terapeuta
Prerrequisitos	El usuario debe estar identificado.
Descripción	Permite visualizar todos los datos detallados de la sesión seleccionada y realizar cambios en ella. Se pueden eliminar actividades específicas asignadas a dicha sesión, o bien eliminar la sesión. Al eliminar la sesión esta deja de ser visible para los terapeutas.

Tabla 12. Requisito 11. Editar sesión

Título	12. Cambiar permisos de terapeutas específicos
Actor	Administrador, Terapeuta (secundario)
Prerrequisitos	El usuario debe estar identificado
Descripción	Introduciendo un correo específico, se pueden dar o retirar permisos de administrador.

Tabla 13. Requisito 12. Cambiar permisos de terapeutas específicos

Título	13. Dar de alta/baja usuarios específicos
Actor	Administrador, Terapeuta (secundario), Paciente (secundario)
Prerrequisitos	El usuario debe estar identificado
Descripción	Permite al administrador cambiar el estado de los usuarios, ya sea dar de alta (activar) o dar de baja (desactivar) a usuarios específicos introduciendo su correo. Esto afecta a la visibilidad y accesibilidad del usuario en la plataforma.

Tabla 14. Requisito 13. Dar de alta/baja usuarios específicos

Título	14. Editar cuenta personal
Actor	Administrador, Terapeuta, Paciente
Prerrequisitos	El usuario debe estar identificado
Descripción	Permite al usuario editar la información de su perfil, incluyendo nombre, contraseña, o dar de baja su cuenta. Los cambios realizados son almacenados y reflejados en su cuenta inmediatamente.

Tabla 15. Requisito 14. Editar cuenta personal

Título	15. Cerrar sesión
Actor	Administrador, Terapeuta, Paciente

Prerrequisitos	El usuario debe estar identificado
Descripción	El usuario selecciona la opción para cerrar sesión. Una vez cerrada, el usuario deberá volver a ingresar sus credenciales para acceder a la plataforma.

Tabla 16. Requisito 15. Cerrar sesión

Título	16. Añadir actividades a sesión
Actor	Terapeuta
Prerrequisitos	El terapeuta debe estar identificado. La sesión debe estar creada.
Descripción	El terapeuta puede añadir actividades específicas seleccionadas a una sesión ya creada. Para ello se muestra el listado de sesiones actuales creadas.

Tabla 17. Requisito 16. Añadir actividades a sesión

Título	17. Crear nuevas sesiones
Actor	Terapeuta
Prerrequisitos	El terapeuta debe estar identificado
Descripción	El terapeuta introduce detalles de una nueva sesión, incluyendo nombre y actividades incluidas. Una vez creada, la sesión está disponible para ser asignada a pacientes.

Tabla 18. Requisito 17. Crear nuevas sesiones

Título	18. Asignar sesión
Actor	Terapeuta, Paciente (secundario)
Prerrequisitos	El terapeuta debe estar identificado.
Descripción	El terapeuta asigna una sesión específica a un paciente. Esto permite que el paciente pueda acceder a las actividades dentro de esa sesión.

Tabla 19. Requisito 18. Asignar sesión

Título	19. Seleccionar actividad a crear
Actor	Terapeuta
Prerrequisitos	El terapeuta debe estar identificado
Descripción	Se visualizan los tipos de actividades disponibles y el terapeuta puede escoger el tipo de actividad que desea crear.

Tabla 20. Requisito 19. Seleccionar actividad a crear

Título	20. Configurar sopa de letras
Actor	Terapeuta
Prerrequisitos	El terapeuta debe estar identificado
Descripción	El terapeuta define parámetros para la sopa de letras, como el nombre de la actividad, el tamaño del tablero, palabras incluidas, orientaciones permitidas de las palabras y nivel de dificultad.

Tabla 21. Requisito 20. Configurar sopa de letras

Título	21. Configurar palabra contraria
Actor	Terapeuta
Prerrequisitos	El terapeuta debe estar identificado
Descripción	El terapeuta introduce los parámetros para configurar el juego de palabra contraria: el nivel de dificultad y una palabra y su contraria para que el paciente la identifique.

Tabla 22. Requisito 21. Configurar palabra contraria

Título	22. Configurar clasificación de palabras
Actor	Terapeuta
Prerrequisitos	El terapeuta debe estar identificado
Descripción	El terapeuta define parámetros para la actividad de clasificación de palabras, como el nombre de la actividad y el nivel de dificultad. Además, establece categorías y palabras correspondientes para que el paciente las clasifique adecuadamente.

Tabla 23. Requisito 22. Configurar clasificación de palabras

Título	23. Crear actividad
Actor	Terapeuta
Prerrequisitos	El terapeuta debe estar identificado
Descripción	Tras configurar detalles de la actividad, el terapeuta guarda la actividad para ser utilizada en futuras sesiones.

Tabla 24. Requisito 23. Crear actividad

Título	24. Eliminar paciente del grupo
Actor	Terapeuta, Paciente (secundario)
Prerrequisitos	El terapeuta debe estar identificado

Descripción	Permite al terapeuta eliminar a un paciente específico de un grupo determinado. Este paciente ya no tendrá acceso a las sesiones asociadas a ese grupo. El terapeuta solo puede eliminar pacientes de grupos en los que él mismo forme parte.

Tabla 25. Requisito 24. Eliminar paciente del grupo

Título	25. Vista del historial del paciente
Actor	Terapeuta
Prerrequisitos	El terapeuta debe estar identificado
Descripción	El terapeuta puede visualizar el progreso y sesiones y actividades realizadas por un paciente específico. Puede observar el tiempo de duración de cada actividad, si fueron completadas y el número de fallos en el transcurso. Esto ofrece datos sobre el avance de la enfermedad y áreas de mejora.

Tabla 26. Requisito 25. Vista del historial del paciente

Título	26. Administrar grupos del usuario
Actor	Terapeuta, Paciente
Prerrequisitos	El terapeuta debe estar identificado
Descripción	Permite al usuario visualizar, añadir o eliminar grupos a los cuales pertenece.

Tabla 27. Requisito 26. Administrar grupos del usuario

Título	27. Añadir grupo nuevo
Actor	Terapeuta, Paciente
Prerrequisitos	El terapeuta debe estar identificado
Descripción	Tanto terapeutas como pacientes pueden ingresar a un nuevo grupo introduciendo el código de acceso correspondiente.

Tabla 28. Requisito 27. Añadir grupo nuevo

Título	28. Eliminar grupo asignado
Actor	Terapeuta, Paciente
Prerrequisitos	El terapeuta debe estar identificado
Descripción	Permite al usuario salir de un grupo específico, eliminándolo de su lista de grupos asignados actualmente.

Tabla 29. Requisito 28. Eliminar grupo asignado

Título	29. Registrar cuenta
Actor	Paciente
Prerrequisitos	Ninguno
Descripción	El paciente introduce información personal (nombre, email y contraseña) para crear una cuenta en la plataforma. Una vez tenga una cuenta creada puede iniciar sesión.

Tabla 30. Requisito 29. Registrar cuenta

Título	30. Realizar sesión
Actor	Paciente
Prerrequisitos	El paciente debe estar identificado. El paciente debe tener una sesión asignada.
Descripción	El paciente puede ver las actividades asignadas a la sesión. Puede iniciar la realización de las actividades y observar cuáles son las que ya ha completado.

Tabla 31. Requisito 30. Realizar sesión

Título	31. Realizar actividad: Sopa de letras
Actor	Paciente
Prerrequisitos	El paciente debe estar identificado
Descripción	El paciente interactúa con el juego de sopa de letras, buscando y seleccionando palabras en el tablero hasta completar la actividad.

Tabla 32. Requisito 31. Realizar actividad: Sopa de letras

Título	32. Realizar actividad: Palabra contraria
Actor	Paciente
Prerrequisitos	El paciente debe estar identificado
Descripción	Al paciente se le muestra una palabra y debe introducir la palabra contraria para completar la actividad.

Tabla 33. Requisito 32. Realizar actividad: Palabra contraria

Título	33. Realizar actividad: Clasificación de palabras
Actor	Paciente
Prerrequisitos	El paciente debe estar identificado
Descripción	El paciente clasifica palabras marcando si pertenecen a una categoría u otra. Finaliza la actividad cuando todas las palabras están bien asignadas.

Tabla 34. Requisito 33. Realizar actividad: Clasificación de palabras

3.4 Requisitos no funcionales

3.4.1 Requisitos de memoria

La aplicación debe ser eficiente en términos de uso de memoria para asegurar un rendimiento óptimo. Se debe minimizar la cantidad de memoria necesaria para que la aplicación funcione correctamente y para que pueda ser utilizada en dispositivos con diversas capacidades de memoria. La aplicación debe ser capaz de gestionar de manera eficaz la memoria durante su ejecución para evitar fugas de memoria y asegurar la liberación adecuada de recursos.

Google Play establece un límite de tamaño para las aplicaciones de 150 MB para el archivo APK[7]. Sin embargo, es deseable mantener el tamaño del APK lo más pequeño posible para facilitar descargas rápidas y reducir el uso de memoria en el dispositivo del usuario.

3.4.2 Requisitos de rendimiento

El rendimiento de la aplicación es una consideración crítica. La aplicación debe responder de manera rápida y eficiente a las acciones del usuario. Se deben optimizar los tiempos de carga y las operaciones de procesamiento para garantizar una experiencia de usuario suave y sin demoras. La aplicación debe mantener un rendimiento constante. En principio no se esperan un gran número de usuarios concurrentes o grandes volúmenes de datos, pero debe estar preparado para mejorar la escalabilidad en el futuro.

3.4.3 Requisitos de usabilidad

La aplicación debe ser fácil de usar y entender para todos los usuarios, independientemente de su nivel de conocimientos técnicos. Debe tener una interfaz de usuario intuitiva y amigable. Además, las funciones y opciones deben estar claramente etiquetadas y accesibles para facilitar la navegación y el uso de la aplicación. Es aconsejable proporcionar asistencia en línea y documentación de ayuda para guiar a los usuarios a través del proceso de uso de las diferentes funcionalidades de la aplicación.

3.4.4 Requisitos de fiabilidad

Además de cumplir con las regulaciones de protección de datos de la Unión Europea (GDPR)[8], la aplicación también debe asegurar la integridad y la confidencialidad de los datos de los usuarios mediante el uso de medidas de seguridad robustas, como el cifrado. La aplicación debe ser capaz de recuperarse de posibles fallos, sin pérdida de datos y debe garantizar una alta disponibilidad para no interrumpir las actividades de los usuarios.

3.4.5 Requisitos de portabilidad

La aplicación debe ser compatible con diferentes versiones y dispositivos Android y especialmente optimizada para tabletas, que es el dispositivo con el que se realizan las sesiones en AFATE. Por lo tanto, debe ser escalable y adaptable a diferentes tamaños y resoluciones de pantalla, para proporcionar una experiencia de usuario coherente y de

[7] APK = Android Package Kit

[8] El Reglamento General de Protección de Datos es el conjunto de regulaciones de la ley de la Unión Europea que tratan la protección de datos y la privacidad

calidad en todos los dispositivos. También debe ser fácil de instalar y desinstalar, sin dejar datos residuales en el dispositivo del usuario.

4 Diseño

En este apartado se presenta el diseño y arquitectura de la aplicación, diseño de la interfaz gráfica y de la base de datos utilizada.

4.1 Estructura del Software: Arquitectura Model-View-ViewModel (MVVM)

La arquitectura Model-View-ViewModel (MVVM)[9] [6] es el diseño estructural del software escogido para el desarrollo de esta aplicación. Esta arquitectura es particularmente popular en el desarrollo de aplicaciones Android, ya que proporciona un enfoque claro y ordenado para organizar el código de la aplicación y facilita el mantenimiento y la escalabilidad del proyecto.

La arquitectura MVVM divide la estructura del software en tres componentes principales:

- **Model**: Representa los datos y la lógica de negocio de la aplicación. En este caso, sería la información de los usuarios, terapeutas, ejercicios y grupos, entre otros, así como las operaciones relacionadas con estos datos.
- **View**: Corresponde a la interfaz de usuario y muestra la información al usuario. En términos de esta aplicación, serían todas las pantallas y elementos de la interfaz de usuario con los que los usuarios interactúan.
- **ViewModel**: Actúa como un enlace entre el Modelo y la Vista. Recibe las acciones del usuario desde la Vista, procesa los datos del Modelo y luego actualiza la Vista.

Utilizar la arquitectura MVVM facilita la separación de la lógica de la interfaz de usuario de la lógica de negocio y la representación de los datos, lo que a su vez facilita la realización de pruebas y el mantenimiento de la aplicación. Además, debido a la naturaleza desacoplada de los componentes, cualquier cambio en una sección de la arquitectura tiene un impacto mínimo o nulo en las otras secciones, lo que aumenta la flexibilidad y escalabilidad del desarrollo.

4.2 Tecnologías y herramientas utilizadas

4.2.1 Kotlin para el desarrollo Android

Para el desarrollo de la aplicación Android se ha seleccionado Kotlin, un lenguaje de programación moderno, conciso y seguro. Su interoperabilidad con Java y su adopción oficial por Google para el desarrollo Android hacen que sea una elección sólida. Con características como la inferencia de tipos, la seguridad de nulidad y la programación funcional, Kotlin mejora la productividad del desarrollador y la calidad del código. También dispone de documentación detallada muy útil para el desarrollo del proyecto[7], [8].

Además, concuerda con los requisitos que la asociación necesitaba, ya que querían una aplicación para utilizar en las tabletas que disponen actualmente, que son Android.

9 Arquitectura MVVM: Model-View-ViewModel

4.2.2 Servicios de Firebase

Firebase, una plataforma de desarrollo de aplicaciones de Google, se ha utilizado para gestionar los datos y las actividades de la aplicación. Firebase proporciona una variedad de servicios como autenticación, base de datos en tiempo real, almacenamiento de archivos, análisis, notificaciones y mucho más[9]. Esto permite desarrollar características rápidamente sin la necesidad de configurar y mantener servidores. La integración de Firebase con Android y su escalabilidad hacen que sea una elección adecuada para esta aplicación.

Todas sus funcionalidades son gratuitas hasta un número de solicitudes determinado, a partir del cual se empieza a realizar el pago. Teniendo en cuenta que el número de usuarios y datos no va a ser muy elevado en este momento, se presenta como una opción económica y sencilla de implementar.

Sin embargo, es importante considerar el impacto de tener una alta dependencia de esta plataforma. Algunos riesgos potenciales son:

- **Riesgo de plataforma única**: Al depender demasiado de Firebase, se coloca una gran cantidad de confianza en una sola plataforma. Si Firebase experimenta algún problema, ya sea una interrupción del servicio o un fallo en un área específica, la aplicación puede verse significativamente afectada.
- **Flexibilidad limitada**: Firebase ofrece una gran cantidad de funciones, pero esto puede limitar la capacidad para personalizar características específicas según las necesidades de la aplicación. En ocasiones, las soluciones ofrecidas por Firebase pueden no ser la opción óptima para determinados requerimientos de la aplicación.
- **Cambios inesperados en las políticas o costes del servicio**: Firebase, al ser un producto de Google, está sujeto a cambios en las políticas de la empresa. Esto puede incluir cambios en las políticas de privacidad, que pueden impactar en cómo se manejan los datos de los usuarios. Además, si Google decide cambiar el modelo de precios de Firebase, esto puede impactar en los costes de mantener la aplicación.
- **Dependencia de proveedores (*Vendor lock-in*):** Esta es una consideración importante para cualquier servicio de terceros. La dependencia excesiva de Firebase puede hacer que sea difícil, costoso y que requiera mucho tiempo migrar a otra plataforma o servicio si alguna vez se hace necesario.

4.2.2.1 Firebase Authentication

Firebase Authentication proporciona un sistema completo de autenticación de usuarios que permite el registro e inicio de sesión de los usuarios de una forma segura y eficiente. Soporta diversos métodos de autenticación, incluyendo correo electrónico y contraseña, proveedores de terceros como Google y Facebook, y autenticación telefónica. Firebase Authentication se integra estrechamente con otros servicios de Firebase y automáticamente maneja muchas tareas que, de otro modo, requerirían mucho esfuerzo para implementar, como la gestión de tokens de usuario, la persistencia de sesiones, y la recuperación de contraseñas. Para este proyecto, esta funcionalidad resulta crucial para garantizar que solo los usuarios autorizados accedan a la información relevante.

4.2.2.2 Firestore Database

Realtime Database es la opción original de Firebase para el almacenamiento de datos. Es una base de datos NoSQL que almacena sus datos como un objeto JSON. Realtime Database es excelente para escenarios en tiempo real debido a su capacidad de propagación

de datos en tiempo real. Sin embargo, las opciones de consulta son limitadas en comparación con Firestore, y los datos deben ser denormalizados y duplicados para permitir diferentes vistas de los mismos datos.

Cloud Firestore es la opción más reciente de Firebase para el almacenamiento de datos y es más potente y escalable que Realtime Database. Firestore ofrece estructuras de datos más complejas y jerárquicas (*collections*) y ofrece una gran cantidad de opciones de consulta. Con Firestore, se puede realizar consultas complejas y compuestas directamente desde el cliente sin necesidad de descargar todo el conjunto de datos. Además, Firestore también soporta transacciones ACID[10] a nivel de documento.

Para esta aplicación se ha escogido Cloud Firestore por varias razones. En primer lugar, Firestore proporciona una estructura de datos más rica y flexible, lo que permite un modelado de datos más intuitivo para la aplicación. En segundo lugar, las consultas potentes y compuestas de Firestore permiten trabajar con los datos de la aplicación de manera más eficiente y efectiva. En tercer lugar, Firestore es más escalable que Realtime Database, lo que garantiza que la base de datos podrá manejar la carga a medida que la aplicación crece y atrae a más usuarios. Por último, Firestore ofrece una mayor escalabilidad y rendimiento, lo que es beneficioso para una aplicación que pretende atender a un gran número de usuarios.

4.3 Fundamentos del desarrollo Android

En esta sección, se describen algunos de los conceptos y componentes esenciales de Android que han sido ampliamente utilizados en el desarrollo de este proyecto[10].

4.3.1 Adapters y ViewHolders en Android

El desarrollo de aplicaciones móviles en Android a menudo implica la presentación de listas o conjuntos de datos, como una lista de contactos, mensajes o elementos de un carrito de compras. Para lograr esto, Android proporciona componentes específicos como *RecyclerView* o *ListView*. Sin embargo, para personalizar cómo se muestran estos datos y optimizar el rendimiento, se necesita usar dos conceptos fundamentales: Adapters y ViewHolders.

4.3.1.1 Adapters (Adaptadores):

Los *adapters* en Android actúan como un puente entre una fuente de datos y un componente de visualización, como *ListView* o *RecyclerView*. Permiten que los datos se presenten y modifiquen de una manera optimizada. El adaptador suele tener métodos como ***onCreateViewHolder*** (para crear una nueva vista de un elemento) y ***onBindViewHolder*** (para asignar datos a una vista). En estos métodos es donde se especifica cómo se transforman los datos en una representación visual. Por ejemplo, para generar una lista de usuarios, el adaptador tomará un usuario de la lista, y usando el método *onBindViewHolder*, llenará una vista con la información de ese contacto (nombre, número, imagen, etc.).

10 ACID: acrónimo que describe cuatro propiedades de una transacción en la base de datos (Atomicity, Consistency, Isolation, Durability)

4.3.1.2 ViewHolders (Portadores de vista):

Un ViewHolder representa una estructura, una plantilla, que contiene las vistas que se usan para mostrar un elemento en un RecyclerView o en un ListView. La idea detrás de un ViewHolder es hacer que el proceso de mostrar datos sea más eficiente.

4.3.1.3 Uso conjunto de Adapters y ViewHolders

El uso de adaptadores y ViewHolders juntos permite a las aplicaciones manejar grandes conjuntos de datos de manera eficiente. Por ejemplo, en una lista larga de elementos, no todos los elementos se muestran en la pantalla al mismo tiempo. Al desplazarse por la lista, los elementos que salen de la pantalla pueden reutilizar las vistas de los elementos que van a entrar, evitando la creación innecesaria de nuevas vistas y optimizando el rendimiento. Este concepto es conocido como "reciclaje de vistas" y es la razón principal por la que RecyclerView lleva ese nombre.

4.3.2 Gestión de errores con Log

Log es una clase en Android que permite escribir mensajes de registro, facilitando la detección y el diagnóstico de problemas y comportamientos inesperados en el código. Se utilizan diferentes niveles de log, como **Log.d()** para depuración, **Log.e()** para errores, entre otros.

4.3.3 Métodos de presentación de mensajes

- **Toast:** Son mensajes de tamaño reducido y transitorios que aparecen brevemente en la parte inferior de la pantalla. Debido a su naturaleza, en nuestra aplicación se han utilizado para operaciones simples, como confirmar que una acción ha sido ejecutada, o para informar al usuario sobre algún proceso exitoso que no requiere una interacción adicional.
- **Dialogs**: Estas ventanas modales aparecen en primer plano y capturan la atención del usuario. En el proyecto, se han utilizado en situaciones que requieren una decisión del usuario o para presentar información que debe ser confirmada o reconocida antes de que el usuario pueda continuar.
- **AlertDialogs**: Es una versión especializada de *Dialogs*, proporcionando estructuras para presentar mensajes junto con botones de acción (como "Aceptar", "Cancelar", “Editar”), listas de selección y más. Por ello son ideales en esta aplicación para situaciones que requieren una respuesta del usuario, como confirmar la eliminación de un elemento, seleccionar una opción de una lista de usuarios o introducir datos en ciertos formularios.

4.4 Control de versiones: GitHub

Para mantener un control efectivo de las versiones del código y el desarrollo de la aplicación, se utiliza Git junto con GitHub. Git es un sistema de control de versiones distribuido que permite rastrear cambios en archivos, ayuda a mantener una historia de modificaciones y facilita la revisión de cambios.

Por su parte, GitHub proporciona una plataforma basada en la nube para alojar repositorios Git. GitHub añade a Git funcionalidades adicionales como una interfaz gráfica de usuario intuitiva y documentación, entre otras. En este proyecto, se ha creado un repositorio en GitHub donde se almacena todo el código de la aplicación (https://github.com/NereaAgud/MentActiva). Esto permite tener un historial completo de todos los cambios realizados, además de servir como respaldo del proyecto.

4.5 Estructura de directorios y ficheros del proyecto

La estructura de los directorios y ficheros en el proyecto se ha diseñado cuidadosamente para garantizar una navegación lógica y coherente. Una organización estructurada es esencial no solo para el desarrollo actual, sino también para facilitar las futuras extensiones y el mantenimiento del código. En la *Figura 8* se puede observar dicha estructura utilizada.

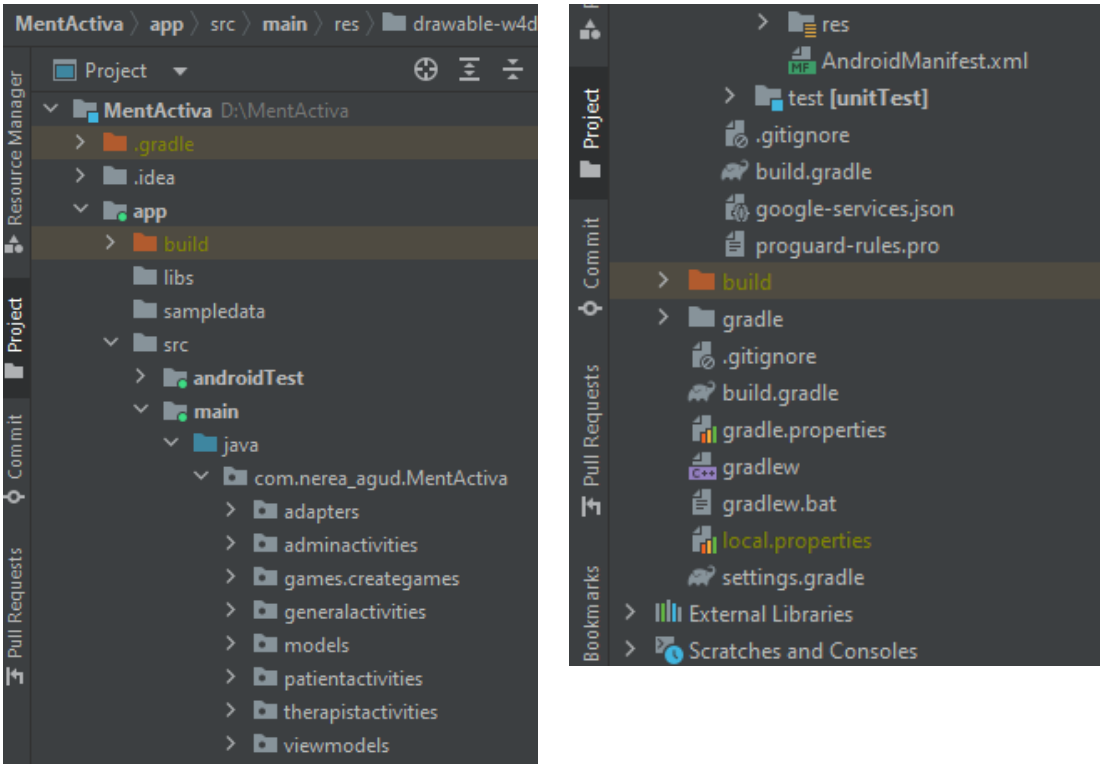


Figura 8. Estructura general de directorios y ficheros del proyecto

Existen varios directorios principales con funcionalidades específicas:

Directorio 'gradle': Este directorio contiene todos los scripts y configuraciones de Gradle necesarios para compilar y construir la aplicación. A través del archivo *build.gradle*, se gestionan las dependencias, versiones y otras configuraciones específicas del proyecto.

Directorio 'app': Este es el directorio central de la aplicación, que contiene:

- **Directorio 'src'**: Almacena el código fuente principal del proyecto.
- **Directorio 'build'**: Este subdirectorio guarda los archivos resultantes de la compilación, como las clases bytecode, recursos procesados y la APK generada.
- **'build.gradle'** (a nivel de app): Se encarga de las configuraciones y dependencias específicas de la aplicación.
- **'google-services.json'**: Este archivo es esencial cuando se integra Firebase en una aplicación Android. Proporciona configuraciones específicas para conectar la app con los servicios de Firebase, incluyendo claves únicas, detalles del proyecto y configuraciones de servicio.

Directorio 'src': Dentro de este directorio reside el núcleo del código fuente de la aplicación. Para lograr una organización eficiente, se ha adoptado una estructura modular basada en paquetes, cada uno de los cuales agrupa funcionalidades afines. Los paquetes principales son:

- **adapters**: Contiene adaptadores de diversas clases, como 'GamesAdapter' y 'SessionsAdapter'.
- **games.creategames**: Engloba las clases destinadas a la creación y desarrollo de juegos.
- **models**: Contiene las clases modelo esenciales como 'Game', 'Group', 'Patient', 'Session' y 'Therapist'.
- **viewmodels**: Aquí se ubican todas las clases viewModel que se utilizan en el resto de clases activity.
- **generalactivities, adminactivities, patientactivities** y **therapistactivities**: Estos paquetes consolidan las principales funcionalidades de la aplicación, categorizadas según el perfil de usuario, o funcionalidades que son compartidas entre varios roles.

Directorio principal 'res': En el directorio de recursos, 'res', se almacenan todos los componentes visuales y elementos de diseño que la aplicación utiliza. Dentro de los subdirectorios denominados **'drawable'**, se han colocado imágenes, iconos y otros recursos gráficos. Por otro lado, en el subdirectorio **'layout'**, se han ubicado los archivos XML responsables de definir la estructura y apariencia de las diferentes interfaces de usuario. Estos archivos XML se han nombrado de manera coherente con las actividades a las que están asociados, asegurando así una referencia clara entre la lógica y la presentación. Por ejemplo, para la actividad *AddGroupActivity.kt* tenemos un correspondiente *activity_add_group.xml*.

Para mantener una estética consistente en toda la aplicación, se ha definido un tema específico y se ha creado el archivo **styles.xml**, donde se especifican los estilos generales, como colores, tamaños de texto y otros elementos visuales. Esta organización permite que los cambios de diseño se realicen de manera global y uniforme.

```
    <resources>
        <!-- Estilos para los botones -->
        <style name="PrimaryButton"
parent="Widget.Material3.Button.Icon">
            <item name="android:layout_width">wrap_content</item>
            <item name="android:layout_height">wrap_content</item>
            <item name="android:paddingLeft">70dp</item>
            <item name="android:paddingTop">20dp</item>
            <item name="android:paddingRight">70dp</item>
            <item name="android:paddingBottom">20dp</item>
            <item name="android:textSize">25sp</item>
        </style>
        <style name="SecondaryButton"
parent="Widget.Material3.Button.ElevatedButton">
            <item name="android:layout_width">wrap_content</item>
            <item name="android:layout_height">wrap_content</item>
            <item name="android:paddingLeft">16dp</item>
            <item name="android:paddingTop">8dp</item>
            <item name="android:paddingRight">16dp</item>
            <item name="android:paddingBottom">8dp</item>
```

```
        <item name="android:textSize">20sp</item>
    </style>
```

Código 1. Código extraído de “styles.xml” donde se definen dos estilos de botones

Con respecto a la localización, se ha adoptado un enfoque que permite la traducción de la aplicación a diferentes idiomas. Todas las cadenas de texto se han centralizado en archivos **strings.xml**, facilitando así su gestión y traducción. Actualmente, la aplicación está preparada para ofrecer su contenido en catalán, español e inglés.

El archivo *AndroidManifest.xml* es esencial en cualquier proyecto Android. En él se declaran los permisos que requiere la aplicación, las actividades, los servicios, los proveedores de contenido y los receptores de transmisión. Además, es donde se especifican características clave, como la actividad de inicio y los requisitos de hardware y software del dispositivo.

5 Interfaz de usuario y patrones de diseño

La aplicación ha sido diseñada para ser intuitiva y para seguir un conjunto coherente de patrones de diseño, tanto en la arquitectura del código como en la interfaz de usuario.

5.1 Diseño de la interfaz gráfica

Se ha buscado ofrecer una experiencia de usuario fluida y consistente a lo largo de todas las actividades y ventanas de la aplicación. La organización y estructuración del código refleja esta intención, permitiendo un fácil seguimiento y mantenimiento.

5.1.1 Patrones de diseño

- **Modularización de funciones**: Como se observa en actividades como 'AddGroupActivity' (*Código 2*), se ha seguido un patrón en el cual las funciones se encapsulan para realizar tareas específicas. Esto se ve reflejado en cómo se maneja la inicialización y configuración de vistas. Cada método, como setupViews(), se encarga de un aspecto específico de la interfaz de usuario, ya sea la asignación de eventos a botones o la configuración inicial de componentes. Este enfoque en la modularización no sólo ofrece un código limpio y legible, sino que también facilita la reutilización y mantenimiento del código.

```
class AddGroupActivity : AppCompatActivity() {
    private lateinit var db: FirebaseFirestore

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_add_group)
        initializeFirebase()
        setupViews()
    }
```

Código 2. Código 'AddGroupActivity'.

- **Principio de responsabilidad única**: Siguiendo este principio, cada función tiene un propósito claro y definido. Esto se ve en cómo las funciones en onCreate() invocan otras funciones que se encargan de tareas específicas. Otro ejemplo es en la actividad 'EditSessionActivity' (*Código 3*). Se puede observar que las funciones tienen un objetivo único para mejorar la reusabilidad del código.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_edit_session)
    initViews()                  //inicialización vistas interfaz
                                 //usuario
    getSessionDataFromIntent()   //recurar los datos de la sesión
                                 //desde el intent de la actividad
    fetchSessionFromFirestore()  //recuperar los datos de una
                                 //sesión desde Firestore
}
```

Código 3. Código 'EditSessionActivity'.

5.1.2 Componentes Material Design 3

El diseño de la aplicación sigue las directrices del **Material Design 3** [11], proporcionando una estética moderna y una interfaz de usuario coherente y agradable. Estos componentes ofrecen componentes y patrones visuales que mejoran la experiencia del usuario:

- **Botones**: Utilizados para acciones primarias y secundarias dentro de la aplicación. Así como para alguna acción concreta.

Se han utilizado diferentes estilos de botones del paquete de Material Design 3 (*Código 4*). Para los botones principales se ha utilizado el botón estándar "Icon" (*Figura 9*). Para botones llamados secundarios, que se utilizan para cancelar una operación o hacer una deselección de *checkboxes*, se ha utilizado el botón de tipo "ElevatedButton" (*Figura 10*) y para las sesiones de pacientes se ha utilizado el estilo "OutlinedButton" (*Figura 11*). Este último muestra el botón sin relleno y cambia de color al ser seleccionado, para ser reactivo con el paciente y facilitarles el uso de la aplicación. Además, se cambia su color de fondo una vez la sesión ha sido completada.

```
<style name="PrimaryButton" parent="Widget.Material3.Button.Icon">

<style name="SecondaryButton"
parent="Widget.Material3.Button.ElevatedButton">

<style name="SessionButton"
parent="Widget.Material3.Button.OutlinedButton">
```

Código 4. Código XML que define las cabeceras de estilo de botones utilizados en el proyecto (*styles.xml*).

Figura 10. Visualización del botón "Icon"

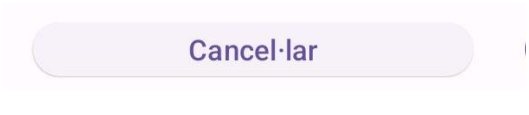


Figura 9. Visualización del botón "ElevatedButton"

Figura 11. Visualización de los 3 estados del botón de tipo "OutlinedButton"

- **Tarjetas**: Ayudan a presentar la información de manera organizada y estructurada. Para ello se ha utilizado la extensión "</androidx.cardview.widget.CardView>"

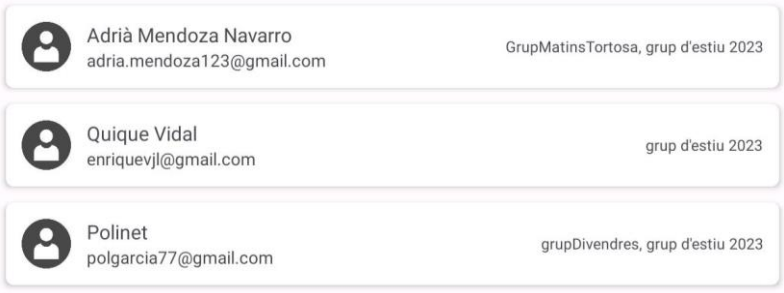


Figura 12. Visualización de "cards"

- **Textos de entrada:** Ofrecen funcionalidades adicionales a los inputText por defecto de Android Studio. Por ejemplo, al seleccionar el cuadro de texto descripción de qué debes escribir, se localiza en la parte superior, permitiendo que el usuario sepa en todo momento qué debería escribir allí. También permite añadir iconos laterales con funcionalidades. Por ejemplo, en la imagen de la *Figura 13* en "Contraseña", el icono del ojo permite visualizar la contraseña, que al escribir se pone como puntos para mantener la privacidad. En el *Código 5* se puede observar el código necesario para la implementación del textInput del campo de la contraseña.

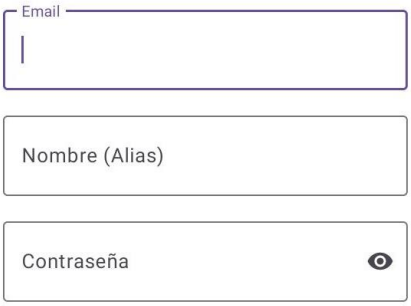


Figura 13. Visualización de inputEditTexts de Material Design

```
    <com.google.android.material.textfield.TextInputLayout
                    android:id="@+id/passwordTextField"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="@string/hint_password"
                    android:layout_gravity="center"
                    android:layout_marginBottom="16dp"
                    android:layout_marginLeft="100dp"
                    android:layout_marginRight="100dp"
                    android:minHeight="70dp"
                    app:endIconMode="password_toggle">

<com.google.android.material.textfield.TextInputEditText
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textPassword"
            android:minHeight="70dp" />
    </com.google.android.material.textfield.TextInputLayout>
```

Código 5. Código XML que define una interfaz de usuario utilizando el componente 'TextInputLayout' y 'TextInputEditText' proporcionados por Material Design de Google.

En este código, <com.google.android.material.textfield.TextInputLayout> es el contenedor principal que envuelve el campo de entrada de texto. Proporciona las funcionalidades adicionales como mostrar una etiqueta flotante (*hint*), gestionar errores y mostrar un ícono al final del campo de texto. Los campos siguientes gestionan el tamaño y lo localizan en pantalla.

Y <com.google.android.material.textfield.TextInputEditText> es el campo de entrada de texto real dentro del 'TextInputLayout' que fija el ancho y la altura del campo de entrada, el tipo de dato de entrada (en este caso contraseña, lo que oculta automáticamente los caracteres de la contraseña) y la altura mínima.

6 Desarrollo de implementación

La fase de implementación es crucial en el desarrollo de software. Es aquí donde los diseños y análisis previos toman forma y se materializan en una aplicación funcional. Esta sección tiene como objetivo exponer las decisiones técnicas y los procedimientos llevados a cabo durante la implementación de la aplicación.

6.1 Configuración del entorno

Antes de comenzar con la implementación, es esencial tener un entorno adecuado. Para el desarrollo de la aplicación, se seleccionó Android Studio como IDE principal, debido a su integración natural con Kotlin y las herramientas específicas que ofrece para el desarrollo Android. Específicamente, se utilizaron estos ajustes de configuración para el proyecto:

- **Plugins utilizados:**
  - **com.android.application**: Plugin básico para aplicaciones Android.
  - **org.jetbrains.kotlin.android**: Plugin para soporte Kotlin en Android.
  - **com.google.gms.google-services**: Plugin para integrar los servicios de Google.
  - **kotlin-android**: Plugin para utilizar Kotlin específicamente para Android.
- **Configuración de Android:**
  - Espacio de nombres: **com.nerea_agud.MentActiva**.
  - SDKs: Se utiliza el **compileSdk** versión 34, con un **minSdk** de 24 y **targetSdk** de 34.
  - Configuración predeterminada:
    - ID de aplicación: "com.nerea_agud.MentActiva".
    - Versión de código: 1
    - Nombre de versión: "1.0".
  - View Binding ha sido habilitado para facilitar la construcción de la UI de la aplicación.
  - Opciones de compilación: Se establece la compatibilidad con la versión 1.8 de Java.
  - Opciones de Kotlin: Se utiliza el **jvmTarget** en la versión '1.8'.
- **Gradle**: El proyecto utiliza la distribución Gradle 7.5, tal y como se indica en el archivo ‘gradle-wrapper.properties’.
- **Bibliotecas utilizadas**:
  - **Android y Jetpack**:
    - ‘androidx.core:core-ktx:1.10.1’: Extensiones Kotlin para la librería core de Android.
    - ‘androidx.appcompat:appcompat:1.6.1’: Biblioteca para compatibilidad y componentes modernos de la interfaz de usuario.
    - ‘com.google.android.material:material:1.9.0’: Componentes de Material Design para Android.

- ‘androidx.constraintlayout:constraintlayout:2.1.4’: Biblioteca para diseño de interfaces con ConstraintLayout.
- ‘androidx.lifecycle:lifecycle-viewmodel-ktx:2.6.1’ y ‘androidx.lifecycle:lifecycle-runtime-ktx:2.6.1’: Bibliotecas para el manejo del ciclo de vida y ViewModels usando Kotlin.
- ‘androidx.activity:activity-ktx:1.7.2’: Extensiones Kotlin para la actividad.
- ‘androidx.recyclerview:recyclerview:1.3.0’: Biblioteca para visualizar listas y colecciones en la UI.

- **Otros**:
  - ‘com.firebaseui:firebase-ui-firestore:7.2.0’: UI de Firebase para Firestore.
  - ‘junit:junit:4.13.2’: Biblioteca para realizar pruebas unitarias.
  - ‘androidx.test.ext:junit:1.1.5’ y ‘androidx.test.espresso:espresso-core:3.5.1’: Bibliotecas para realizar pruebas en Android

6.2 Integración con Firebase

Firebase es esencial para la aplicación, ya que proporciona una variedad de servicios backend. Al inicio se realizó la configuración básica, añadiendo el archivo ‘google-services.json’ al proyecto y asegurando que las dependencias necesarias estuvieran presentes en el archivo ‘build.gradle’.

La implementación se llevó a cabo siguiendo el manual oficial y las recomendaciones de Firebase [9]. Se creó el proyecto (‘appnerea-89f15’) en Firebase Console y se añadió la aplicación Android especificando su package name ‘com.nerea_agud.MentActiva’. Una vez configurado, se descargó el archivo ‘google-services-json’ y se colocó en el directorio ‘:app’ del proyecto. Posteriormente, se añadieron los plugins y dependencias necesarias en el archivo ‘build.gradle’. Se configuró el fichero ‘AndroidManifest.xml’ y, a medida que se fueron creando actividades, se fueron añadiendo aquí para su correcta ejecución.

A continuación, se muestran las especificaciones en detalle utilizadas para realizar esta integración con Firebase:

- **Incorporación del archivo ‘google-services.json’**: Es el archivo clave para conectar la aplicación con Firebase. En este caso, el ID del proyecto Firebase es ‘appnerea-89f15’, con un número de proyecto 336875137668.
- **Dependencias en ‘build.gradle’**:
  - Se ha añadido el Bill of Materials (BOM) de Firebase, lo que simplifica la adición de dependencias relacionadas con Firebase, ya que no es necesario especificar las versiones: ‘com.google.firebase:firebase-bom:32.0.0’.
  - Se han integrado módulos de Firebase como Analytics (com.google.firebase:firebase-analytics-ktx), Auth (com.google.firebase:firebase-auth-ktx), Database (com.google.firebase:firebase-database-ktx:20.2.2), Firestore (com.google.firebase:firebase-firestore-ktx) y Functions (com.google.firebase:firebase-functions-ktx:20.3.1) entre otros.

- **Manifiesto de Android** (**‘AndroidManifest.xml’**): La aplicación ha sido configurada con un icono personalizado (‘mentactiva_logo’), un tema específico (Theme.MentActiva), y se ha establecido que la actividad de inicio (IniActivity) actúa como el punto de entrada a la aplicación. Además, todas las actividades se han establecido en orientación vertical para el aprovechamiento óptimo de la pantalla.

6.3 Base de datos: Estructura y Modelado

La estructura de la base de datos es crucial para la eficiencia y escalabilidad de la aplicación. Firebase Firestore ha sido la elección para el almacenamiento de datos, como se ha mencionado previamente.

Al utilizar Firestore Database, la información se organiza en colecciones (*collections*) y documentos (*documents*). En este caso, se ha decidido utilizar como identificadores de los documentos parámetros únicos de los objetos en lugar de un UID autogenerado. En el caso de los documentos de usuarios se utiliza el correo, y para las sesiones, juegos y grupos, el nombre. Este enfoque aporta ventajas y desventajas:

Ventajas:

- **Búsqueda Rápida**: Se puede acceder o verificar la existencia de una sesión específica directamente usando el ID del documento (que sería el nombre de la sesión) sin tener que hacer una consulta.
- **Consistencia**: Garantiza que no habrá dos sesiones con el mismo nombre, ya que los ID de los documentos en una colección deben ser únicos.

Desventajas:

- **Flexibilidad limitada**: Si alguna vez se necesita cambiar el nombre de una sesión, se tendría que crear un nuevo documento con el nuevo nombre y eliminar el antiguo, lo cual no es óptimo.
- **Limitaciones de caracteres**: Firestore tiene algunas restricciones en cuanto a los caracteres permitidos en los IDs de documentos, lo que podría ser un problema si el nombre de la sesión tiene caracteres especiales.

La elección de este diseño responde a varias razones pragmáticas:

Consistencia del historial: Dado que sesiones, juegos y grupos se vinculan al historial de un paciente, es primordial que estos elementos se mantengan estáticos y coherentes, para asegurar la integridad de la información histórica.

Claridad y evitación de ambigüedades: Se busca que tanto los juegos como las sesiones tengan identidades únicas. Al evitar nombres duplicados, se elimina cualquier posible confusión o ambigüedad para los pacientes, terapeutas y administradores.

Eficiencia y unicidad: La prioridad es doble: lograr búsquedas eficientes y asegurar la unicidad de cada nombre. Al emplear el nombre de la sesión como ID, se cumple. Es mucho más ágil y eficaz verificar la existencia de un nombre directamente a nivel de documentos que acceder a cada documento individualmente para comparar un campo específico, como sería el caso del nombre. Esta eficacia se vuelve aún más crucial en el contexto de nuestra aplicación, donde se anticipa que la cantidad de sesiones y juegos experimentará un crecimiento sustancial.

Por lo tanto, aunque cada diseño tiene sus *trade-offs*[11], este enfoque se alinea con los objetivos y necesidades actuales de la aplicación, priorizando la integridad, claridad y eficiencia en el manejo de datos.

Por estos mismos motivos, actualmente eliminar totalmente y de forma definitiva no se realiza en esta base de datos. Sino que todo objeto tiene un campo booleano “deleted” que si está activado deja de ser visible para el resto de usuarios, pero permanece en la base de datos. Sin embargo, el administrador sí puede seguir viendo esta información por si se necesitase en algún momento. En el caso de los usuarios, por ahora no pueden eliminar su cuenta, solo darla de baja.

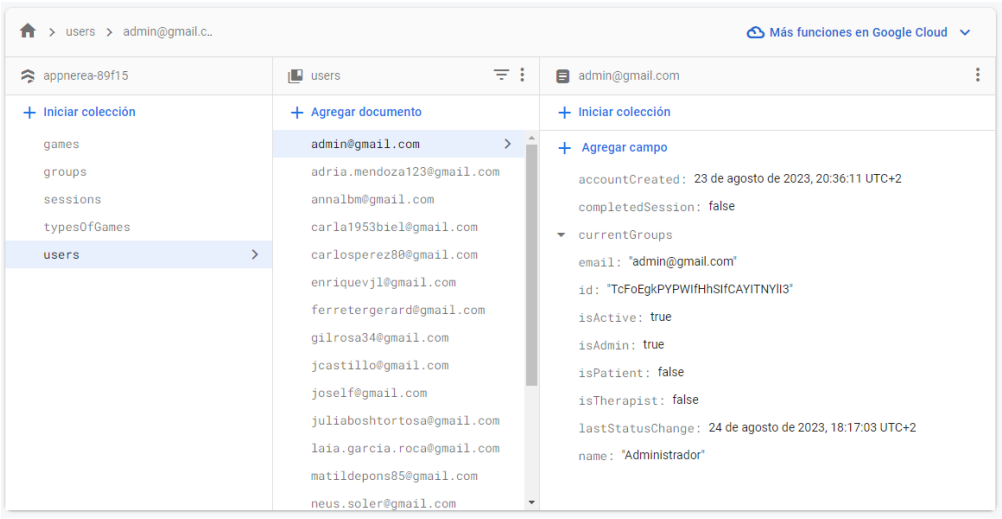


Figura 14. Estructura de los datos en Firebase Firestorage

6.3.1 Colección: users

Dentro de la colección "users", además de definir los campos estándar relacionados con la información personal y de acceso de los usuarios, se han incorporado campos diseñados específicamente para monitorear y administrar el progreso de las sesiones de los usuarios-pacientes. Estos campos detallados capturan aspectos esenciales como: el identificador único de la sesión que ha sido asignada al paciente, el estado actual de la sesión (si ha sido completada o aún está en curso) y una lista de los juegos dentro de esa sesión que ya han sido realizados por el usuario. A medida que el paciente avanza y cumple con las actividades, estos campos se actualizan en tiempo real.

Se diseñó también una colección denominada “sessionHistory” que registra documentos correspondientes a cada sesión asignada a un paciente (*Figura 15*). Cada documento de esta colección detalla los ejercicios que componen la sesión, entre otros datos relevantes. Un aspecto fundamental de estos documentos es que registran los intentos del usuario en cada ejercicio. Con cada intento, se anota el instante de ejecución y el número de errores cometidos. Como ejemplo, en la *Figura 16* se ilustran los intentos efectuados en el juego "frutas y colores", que pertenece a la categoría de clasificación de palabras. Al registrar tanto el número de fallos como el momento exacto de realización, permite deducir el tiempo

11 ***Trade-off***: Término utilizado para describir una situación en la cual se pierde una cualidad o aspecto a cambio de ganar otro. Implica una decisión que debe ser tomada entre varias opciones, donde mejorar un aspecto puede resultar en la disminución de otro.

total invertido en el ejercicio y el nivel con el que se ha desarrollado. Esta estructuración y almacenamiento de la información beneficia a los terapeutas, al darles una visión clara del paciente durante las sesiones, facilitando un seguimiento más preciso y efectivo.

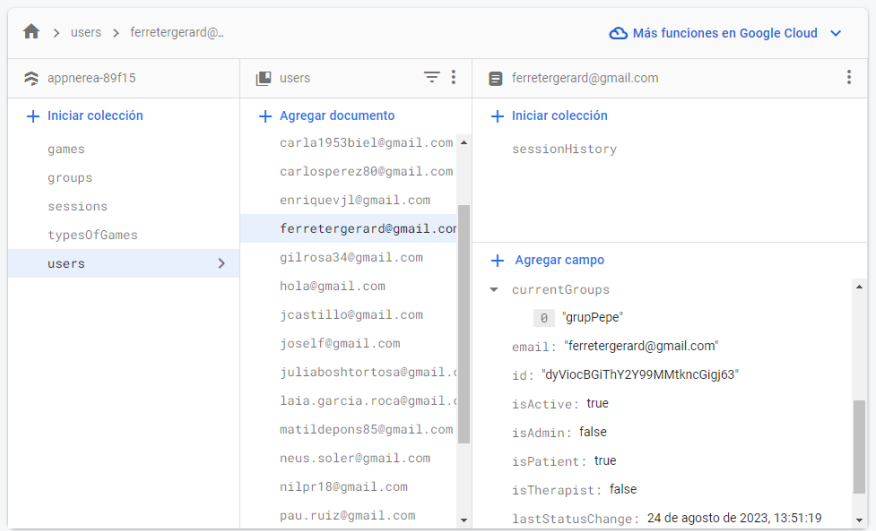
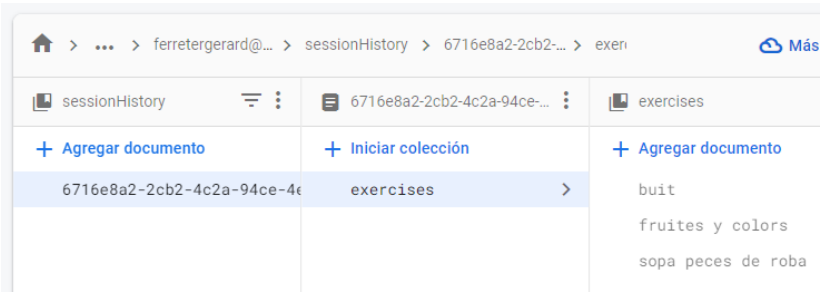


Figura 16. Esquema de la organización de datos de usuarios en Firebase Firestore



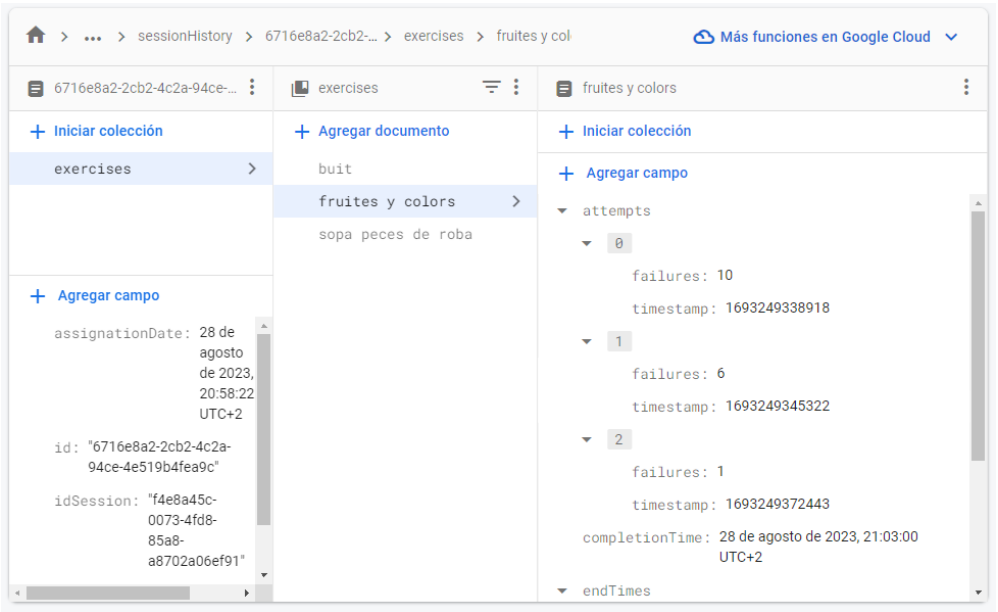


Figura 15. Esquema de la organización del historial de sesiones de los usuarios en Firebase Firestore.

6.3.2 Colección: groups

La colección "groups" se caracteriza por almacenar información general de cada grupo, tal como se ilustra en la *Figura 17*. Un elemento crucial en esta colección es el campo ‘código’, un identificador alfanumérico único compuesto por 8 caracteres, que sirve como contraseña para poder añadirse al grupo. Aunque es posible renovar este código, esta acción está restringida exclusivamente al administrador. Esta decisión garantiza la seguridad y la integridad de cada grupo.

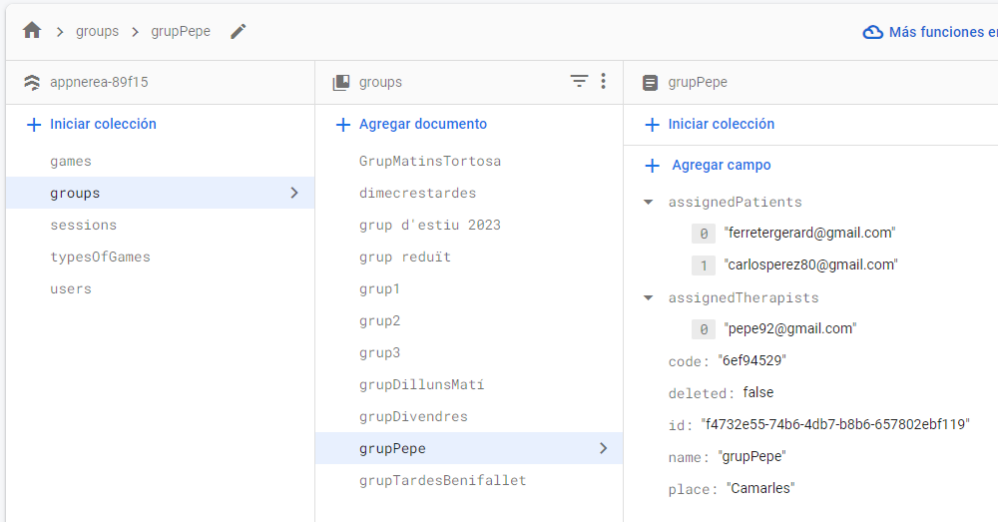


Figura 17. Esquema de la organización de datos de grupos en Firebase Firestore

6.3.3 Colección: games

La colección "games" es el repositorio central de todos los juegos diseñados y creados. Algunos atributos son universales a todos los juegos, y otros campos son específicos según el 'typeOfGame'. Esta estructura dual asegura que los juegos se puedan personalizar eficazmente según las necesidades de los usuarios. Para visualizar esta estructura en detalle, la *Figura 18* desglosa los campos de un juego específico, como la sopa de letras.

Una decisión de diseño crucial fue la de consolidar todos los juegos en una única colección, sin subdividirlos según su tipo. Esta elección se tomó debido principalmente a motivos de escalabilidad. En la práctica, centralizar la información facilita la implementación de mejoras y adaptaciones a medida que el sistema evoluciona. Si se incorporan nuevos juegos o categorías en el futuro, no será necesario reestructurar toda la base de datos ni crear nuevas colecciones. Además, este diseño unificado agiliza las consultas, permitiendo una búsqueda y filtrado más eficientes, ya que se evitan consultas dispersas en múltiples colecciones.

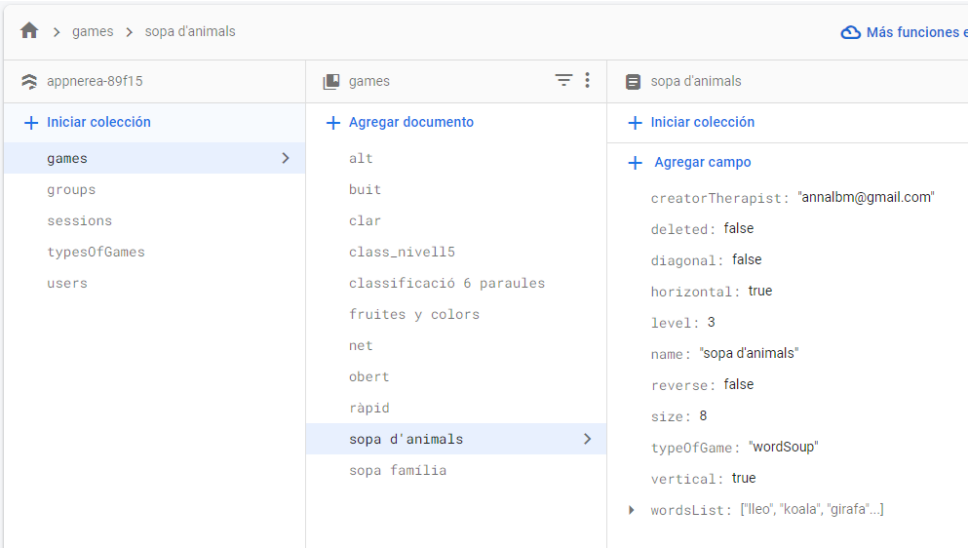


Figura 18. Esquema de la organización de datos de juegos en Firebase Firestore

6.3.4 Colección: sessions

Las sesiones comprenden sus características propias y un listado de referencias a las actividades o juegos que contienen. Específicamente, se conserva el nombre identificador de cada actividad. Al momento de cargar una sesión, este nombre actúa como una referencia directa para acceder al documento del juego o actividad correspondiente.

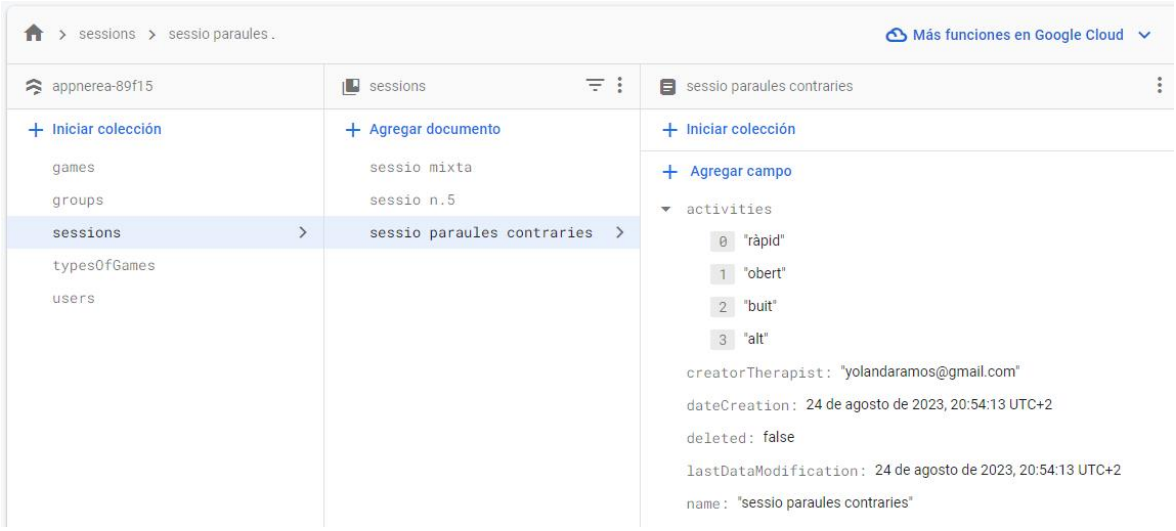


Figura 19. Esquema de la organización de datos de sesiones en Firebase Firestore

6.3.5 Colección: typesOfGames

La colección "typesOfGames" funciona como una lista de definición de los diferentes tipos de juegos disponibles. Además de detallar la nomenclatura de los juegos ya existentes, esta colección ha sido diseñada con una visión a largo plazo, permitiendo añadir nuevos tipos de juegos en el futuro. Esta estructura facilita la extensibilidad y adaptabilidad del sistema a medida que se introduzcan nuevas modalidades de juegos.

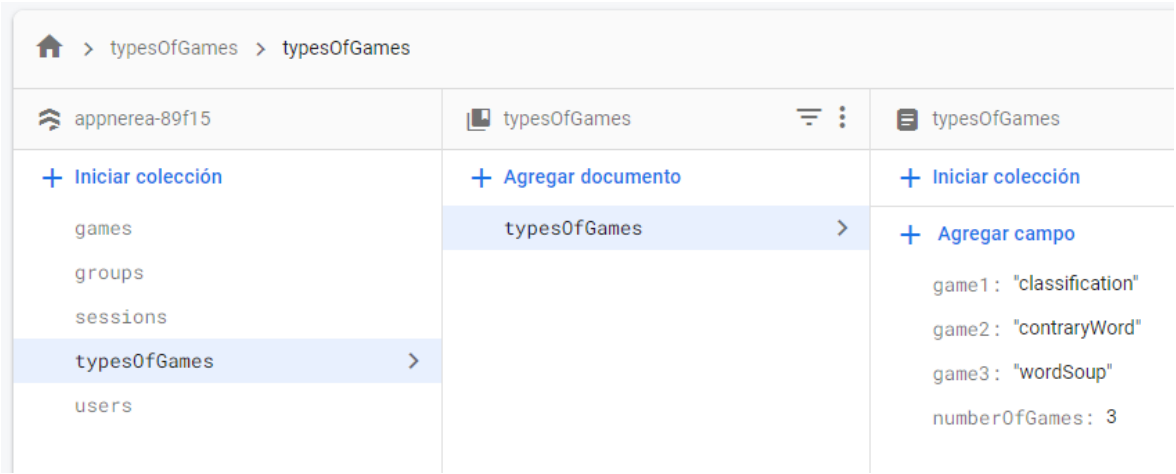


Figura 20. Esquema de la organización de tipos de juegos en Firebase Firestore

6.4 Implementación de actividades y ventanas con capturas

La interfaz de usuario es un componente vital en cualquier aplicación, ya que determina cómo los usuarios interactúan y experimentan el software. Las decisiones de diseño y funcionalidad se reflejan en cada ventana y actividad, y debe primar la fluidez y la lógica de navegación. A lo largo de este apartado, se desglosan las actividades clave, ofreciendo una visión detallada de su propósito, funcionalidad y diseño. Esta descripción se complementa con capturas de pantalla para una visualización directa de cómo se presenta cada actividad al usuario final.

6.4.1 Package: generalactivities

Este paquete contiene actividades que son comunes y no se limitan a un tipo específico de usuario o a un rol concreto.

A continuación, se detallan las principales funciones de cada actividad y se proporciona una descripción basada en las vistas de usuario definidas en los archivos XML correspondientes:

1. **IniActivity:**
    - **Función:** Esta actividad actúa como la pantalla de bienvenida o inicio de la aplicación. Proporciona una primera impresión al usuario y guía las acciones iniciales, el registro o el inicio de sesión.
    - **Vista:** En la *Figura 21* se muestra una captura del diseño “activity_ini.xml”. Se puede observar el nombre y logotipo que he diseñado para la aplicación, junto con el nombre de la asociación. Además, aparecen dos botones, que permiten tanto iniciar sesión como registrar una nueva cuenta.

Figura 21. Captura del diseño ‘activity_ini.xml’

2. **SignUpActivity:**
    - **Función:** Esta actividad permite al usuario introducir sus datos y crear una nueva cuenta en la aplicación. Creando una cuenta de esta forma solo se obtendrán permisos de paciente.
    - **Vista:** En la vista se especifican de forma clara los datos que hay que introducir para que sea más sencillo el proceso. Además, al utilizar los cuadros de introducción de texto de *Material Design* permite que la contraseña tenga seguridad en visibilidad. También permite volver a la pantalla de inicio de sesión, para generar fluidez en la aplicación.

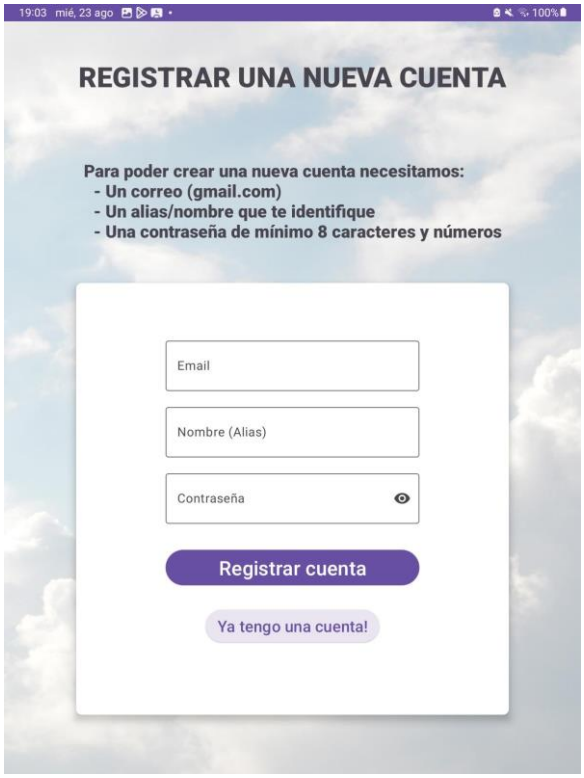


Figura 22. Captura del diseño ‘activity_signup.xml’

3. **LoginActivity:**
    - **Función:** Esta actividad facilita el inicio de sesión de los usuarios registrados, solicitando sus credenciales y validándolas contra la base de datos.
    - **Vista:** Se han generado diferentes mensajes de error para que sea más sencillo para el paciente comprender qué está ocurriendo.

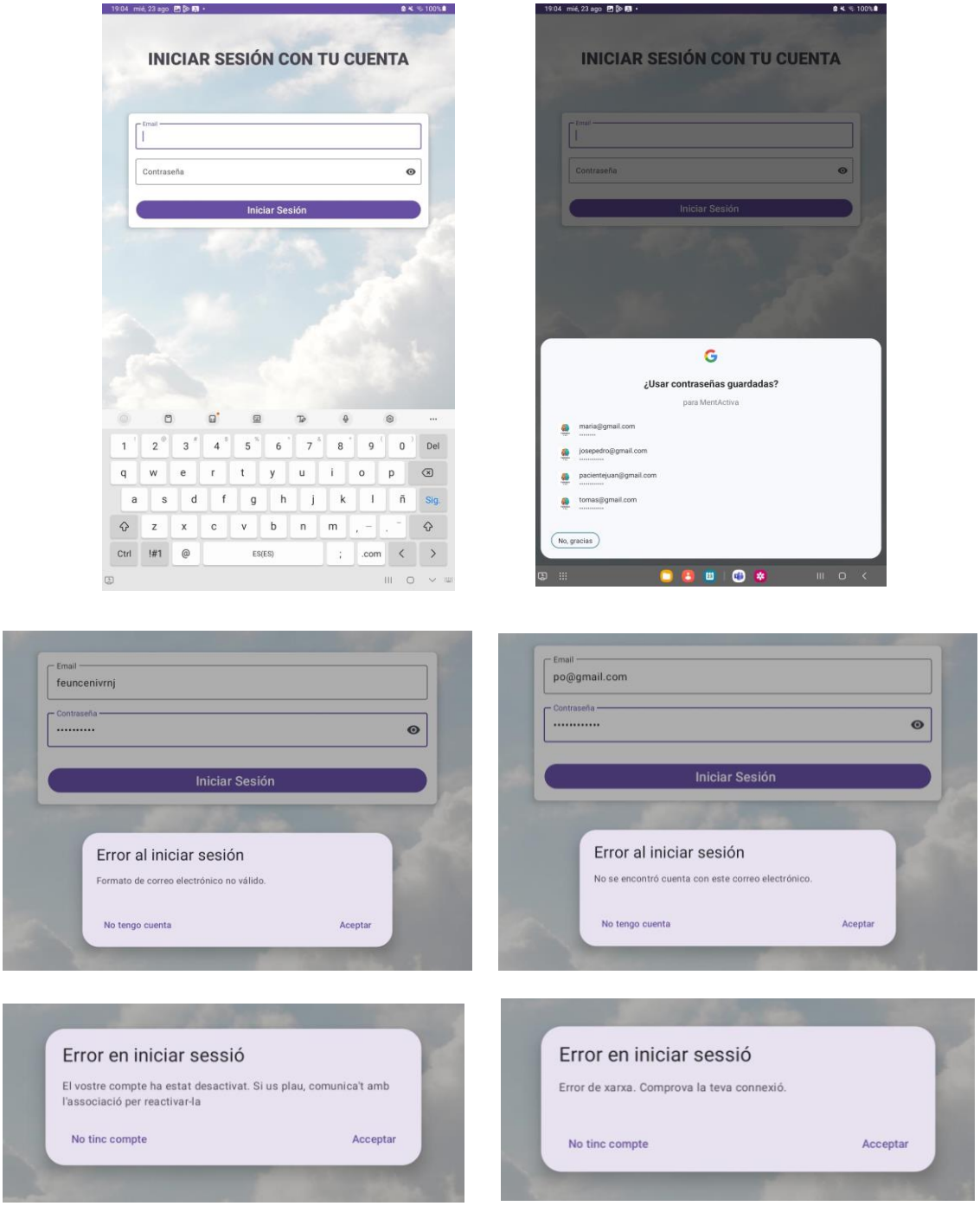


Figura 23. Captura del diseño ‘activity_login.xml’ y mensajes de gestión de errores que utiliza

4. **EditProfileActivity:**
    - **Función:** Permite a los usuarios modificar datos de su perfil incluyendo el nombre, la contraseña y tienen también la opción de dar de baja su cuenta.
    - **Vista:** Para realizar cualquier operación, por seguridad se pide una confirmación final. Para cambiar la contraseña, al ser un dato sensible y de vital importancia, se pide que se introduzca el correo actual, la contraseña actual y la nueva contraseña dos veces. Todos los cambios se realizan al momento. Al dar una cuenta de baja, se cierra la aplicación al usuario directamente y se le restringe el acceso, no puede volver a acceder con su cuenta hasta que el administrador le dé de alta.

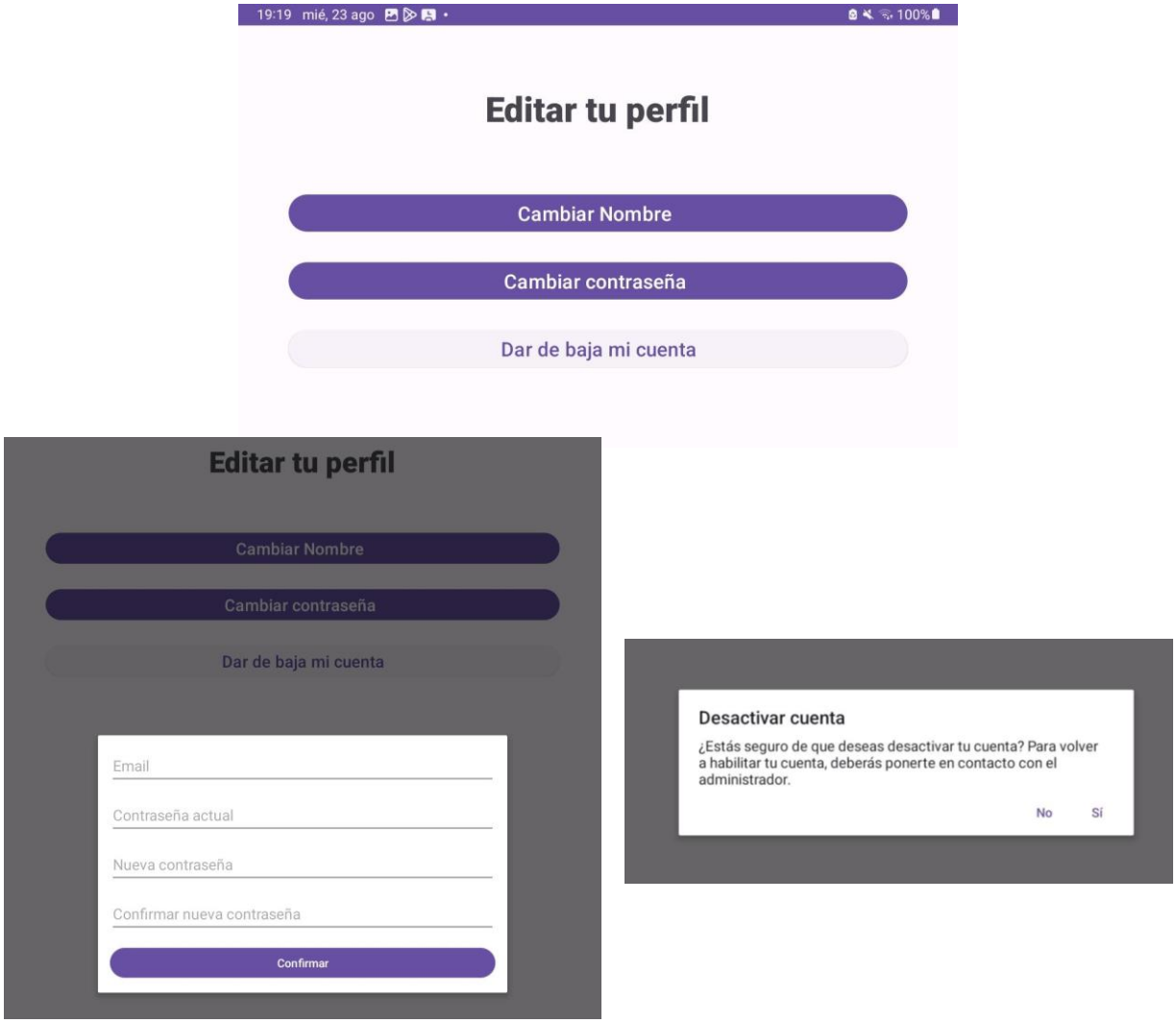
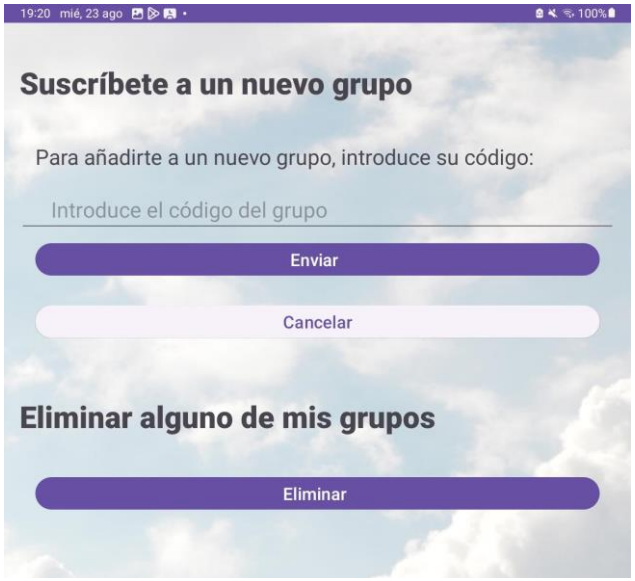


Figura 24. Captura del diseño 'activity_editprofile.xml' y ejemplos de las ventanas interactivas que utiliza con el usuario

5. **AddGroupActivity:**
    - **Función:** Sirve para la gestión personal de grupos asignados. Permite suscribirse a nuevos grupos, introduciendo el código que el personal de la asociación proporciona previamente; o permite también abandonar uno o varios grupos que ya tenga el usuario asignados.
    - **Vista:** Permite realizar ambas actividades de forma clara y una eliminación múltiple de grupos actuales.



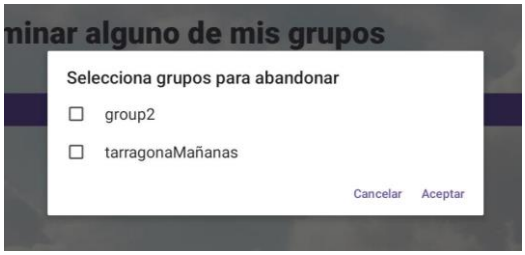


Figura 25. Captura del diseño ‘activity_ addgroup.xml’ y la ventana interactiva de selección de grupos que utiliza con el usuario

6. **SelectionActivity:**
    - **Función:** Esta actividad sirve como centro de selección, donde los usuarios que tienen permisos de administrador y de terapeuta al mismo tiempo, pueden elegir entre qué tipo de papel quieren desarrollar en este momento dentro de la aplicación.
    - **Vista:** Es una vista clara que va a dirigir al usuario a la "homeActivity" correspondiente.

Figura 26. Captura del diseño 'activity_selection.xml'

6.4.2 Package: adminactivties

Este paquete alberga las actividades específicas para los administradores de la aplicación, facilitando una serie de funcionalidades que permiten la gestión, supervisión y control de diferentes aspectos de la plataforma. A continuación, se proporciona un desglose de cada actividad, con una descripción de su propósito principal y una visualización basada en las vistas de usuario definidas en sus archivos XML correspondientes:

1. **HomeAdminActivity:**
    - **Función:** Presenta una interfaz inicial para los administradores, proporcionando acceso directo a herramientas y resúmenes de gestión clave.
    - **Vista:** Los botones se han agrupado en 3 bloques para mejorar la claridad de las funciones. El primer bloque comprende la administración de entidades, el segundo funciones concretas de administrador y las dos finales son funciones personales.

Figura 27. Captura del diseño ‘activity_home_admin.xml’

2. **ShowGamesActivity:**
    - **Función**: Proporciona una lista detallada de juegos, ofreciendo al administrador opciones de supervisión y gestión.
    - **Vista**: Aparece una lista de las actividades/juegos, mostrando el nombre identificador de la actividad, el tipo de actividad que es y el terapeuta que lo ha creado. Si se pulsa sobre uno de ellos, se abre una ventana emergente con toda la información detallada y la opción de poder eliminar la actividad. Las actividades eliminadas se marcan en gris y el resto en blanco.

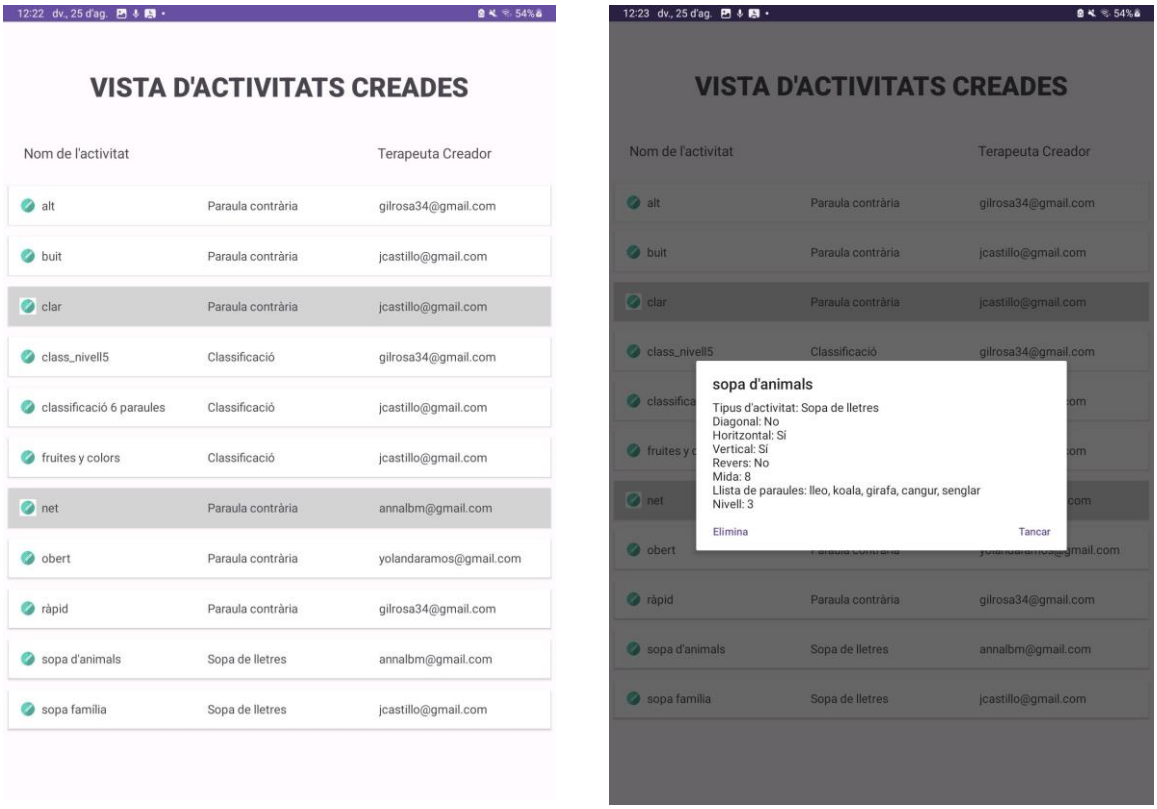


Figura 28. Captura del diseño ‘activity_showgames_admin.xml’ y ventana emergente de datos de la actividad con opciones de “Eliminar” y “Cerrar”

3. **ShowSessionsActivity:**
    - **Función:** Presenta un listado de las sesiones creadas y permite observar de forma rápida los juegos que incluye. Además, se puede ver la información más detallada seleccionando una de las sesiones.
    - **Vista:** Muestra todas las sesiones con un icono, el nombre de la sesión y las actividades que incluye. La ventana emergente de datos permite editar la sesión y acceder a la vista de "EditSessionActivity".

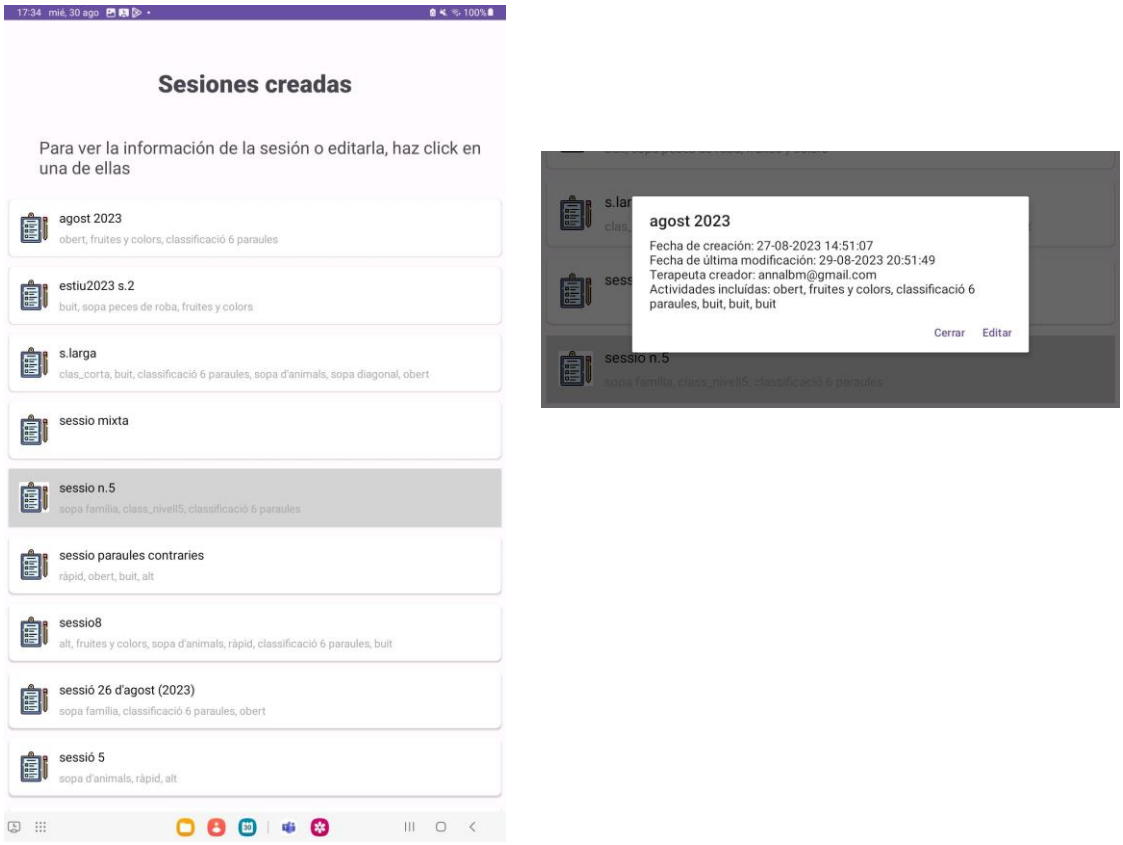


Figura 29. Captura del diseño 'activity_showsessions.xml' y ventana emergente de datos de la sesión con opciones de "Editar" y "Cerrar"

4. **EditSessionActivity:**
    - **Función:** Permite ver la información completa de la sesión. Tiene la funcionalidad de eliminar la sesión y eliminar actividades determinadas que forman parte de la sesión.
    - **Vista:** Para ambas acciones se requiere un mensaje de confirmación final. Las actividades pueden eliminarse de forma múltiple.

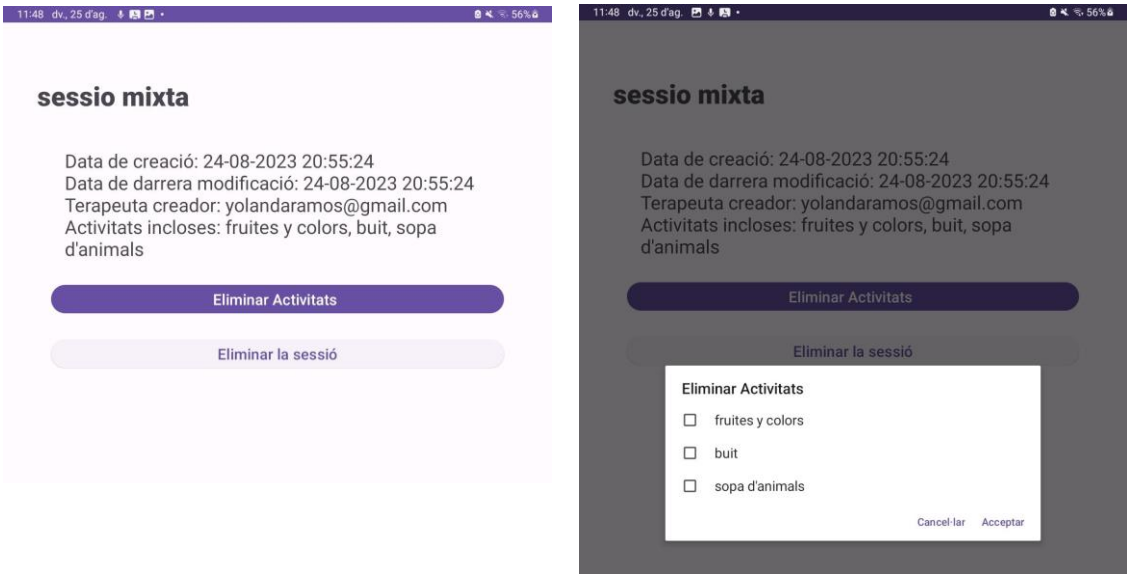


Figura 30. Captura del diseño ‘activity_ editSession.xml’ y la ventana interactiva de selección de actividades que utiliza con el usuario

5. **ShowGroupsActivity:**
    - **Función:** Muestra un listado de todos los grupos creados y permite observar la información de forma rápida. Se muestran de color gris los grupos eliminados y que no están disponibles para nuevos accesos. El resto de grupos de color blanco.
    - **Vista:** La vista de cada grupo muestra un icono, el nombre del grupo y el lugar donde se desarrollan las sesiones de dicho grupo. La selección de un grupo muestra la ventana flotante con los datos de forma detallada. También aparece la opción de acceder a "Editar" el grupo.

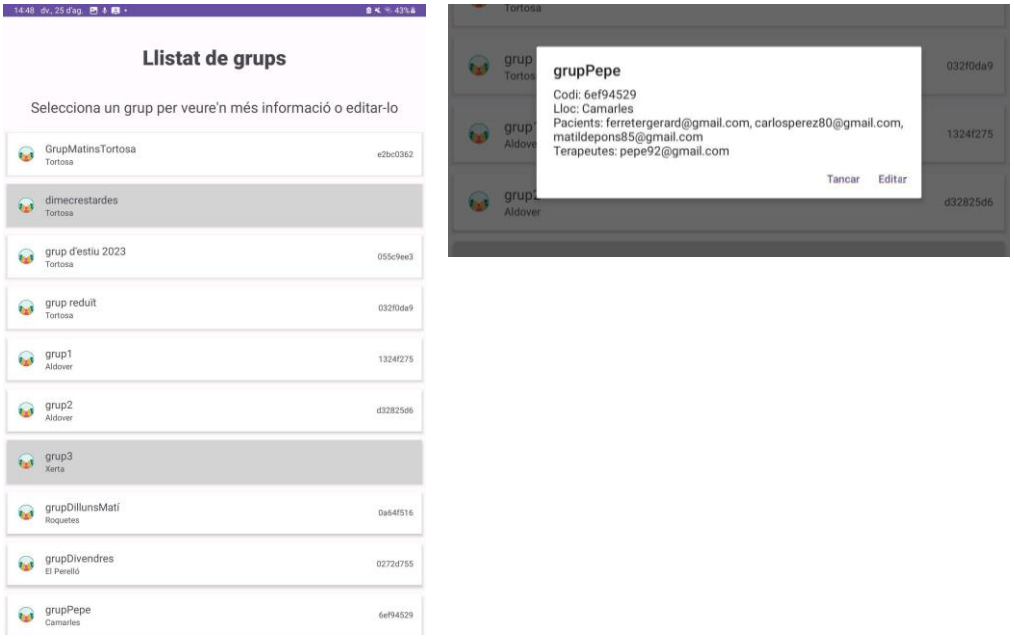


Figura 31. Captura del diseño 'activity_showgroups.xml' y ventana emergente de datos de la sesión con opciones de "Editar" y "Cerrar"

6. **CreateGroupActivity:**
    - **Función:** Posibilita la creación de nuevos grupos, definiendo el nombre del grupo y el lugar donde se ubicará el grupo. A partir de aquí, la aplicación genera un código aleatorio alfanumérico que será el código para que los usuarios puedan acceder al grupo.
    - **Vista:** Si algún campo no está completo, no se permite crear el grupo. Además, se comprueba que el nombre del grupo no esté ya registrado.

Figura 32. Captura del diseño 'activity_creategroup.xml'

7. **EditGroupActivity:**
    - **Función:** Ofrece al administrador la posibilidad de editar y modificar detalles de grupos ya existentes. Se pueden eliminar pacientes o terapeutas del grupo, cambiar la ubicación del grupo, o se puede escoger generar un nuevo código, dejando el código previo sin uso.
    - **Vista:** Todas las acciones utilizan mensajes de confirmación para asegurar que no se realizan cambios erróneos. Para eliminar un terapeuta o un paciente basta con hacer clic sobre el nombre del usuario y aparece la ventana que se observa a continuación. Si se genera un nuevo código o se cambia el lugar, la información se actualiza al momento.

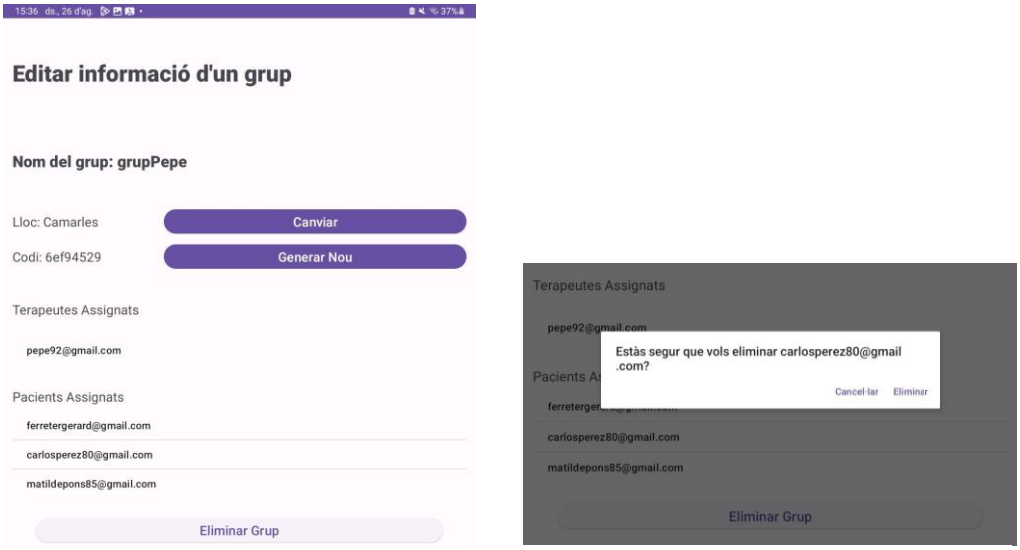


Figura 33. Captura del diseño 'activity_editgroup.xml'

8. **ShowPatientsActivity:**
    - **Función:** Muestra un listado de todos los pacientes que se han registrado en la aplicación. Muestra los pacientes cuyas cuentas han sido dadas de baja de color gris, el resto de color blanco. Permite ver de forma rápida el nombre, el correo y los grupos que tiene actualmente asignados.
    - **Vista:** La selección de un paciente muestra una ventana flotante con toda la información de dicho paciente. También se muestra la opción de "Editar" la cuenta.

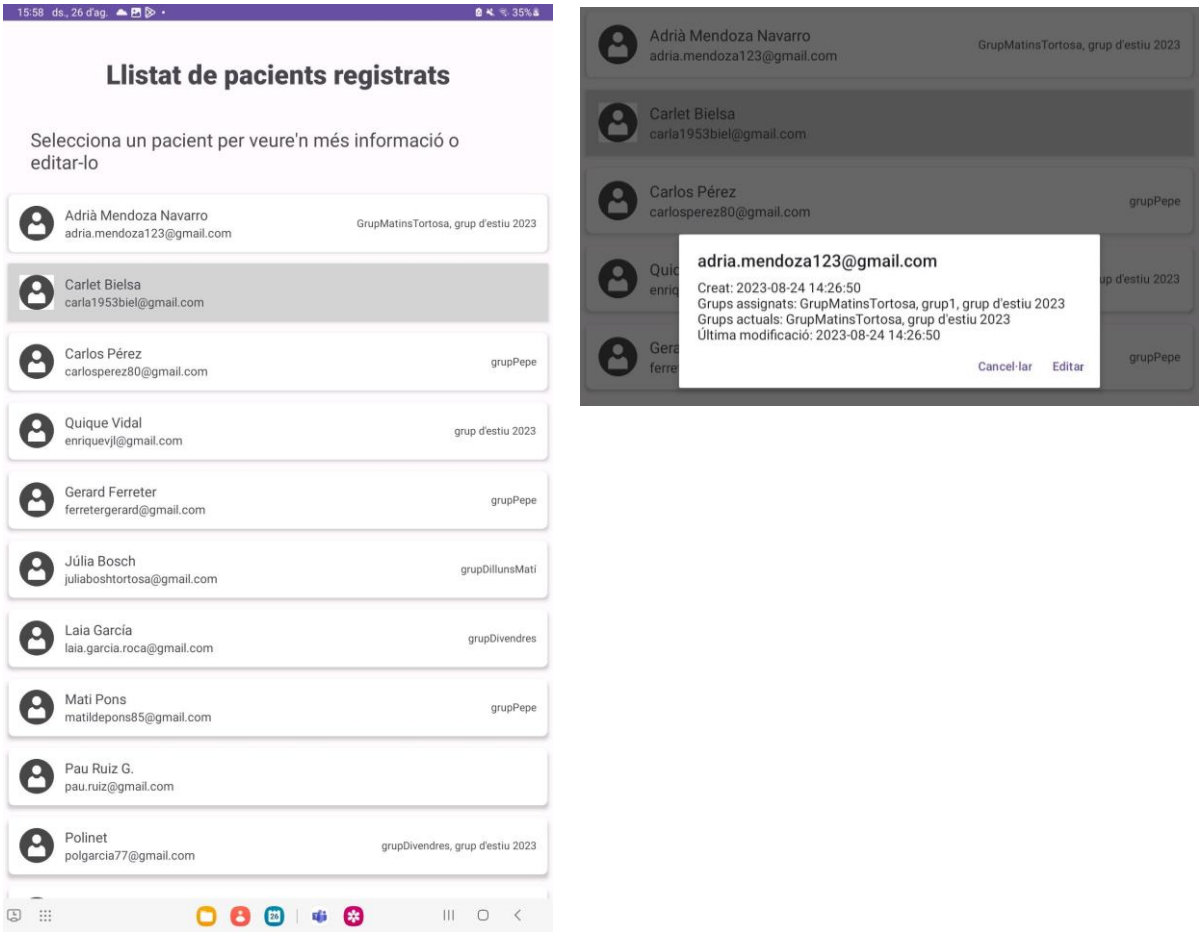


Figura 34. Captura del diseño 'activity_showpatients.xml'

9. **ShowTherapistActivity:**
    - **Función:** Muestra un listado de todas las cuentas de terapeutas que se han creado en la aplicación. Muestra los terapeutas cuyas cuentas han sido dadas de baja de color gris, el resto de color blanco. Permite ver de forma rápida el correo y los grupos que tiene actualmente asignados.
    - **Vista:** La selección de un terapeuta muestra una ventana flotante con toda la información de dicho usuario. También se muestra la opción de "Editar" la cuenta, que llevaría a la misma ventana que en el caso de editar una cuenta de paciente.

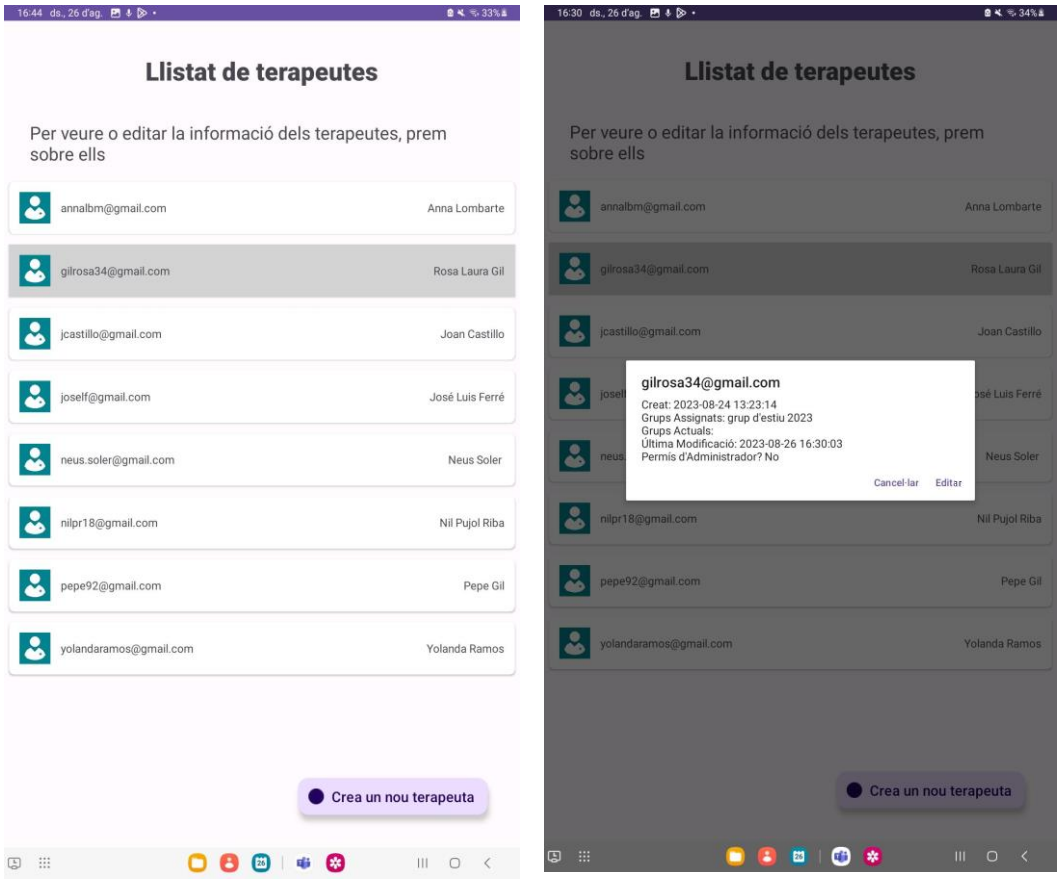


Figura 35. Captura del diseño 'activity_showptherapists.xml'

10. **EditUserActivity:**
    - **Función:** Facilita la edición y modificación de la información de los usuarios, permitiendo al administrador realizar cambios en los detalles de las cuentas. Permite ver toda la información del usuario escogido (terapeuta o paciente) y se puede eliminar de algún grupo que tenga asignado o dar su cuenta de baja.
    - **Vista:** Si el usuario está dado de alta, aparece el botón de dar de baja; si no, el de dar de alta. Para ello se ha utilizado el campo "Visibility" de los botones dependiendo de la condición de si está activo o no. La eliminación de grupos puede ser múltiple. Estos cambios se reflejan en la aplicación de forma instantánea.

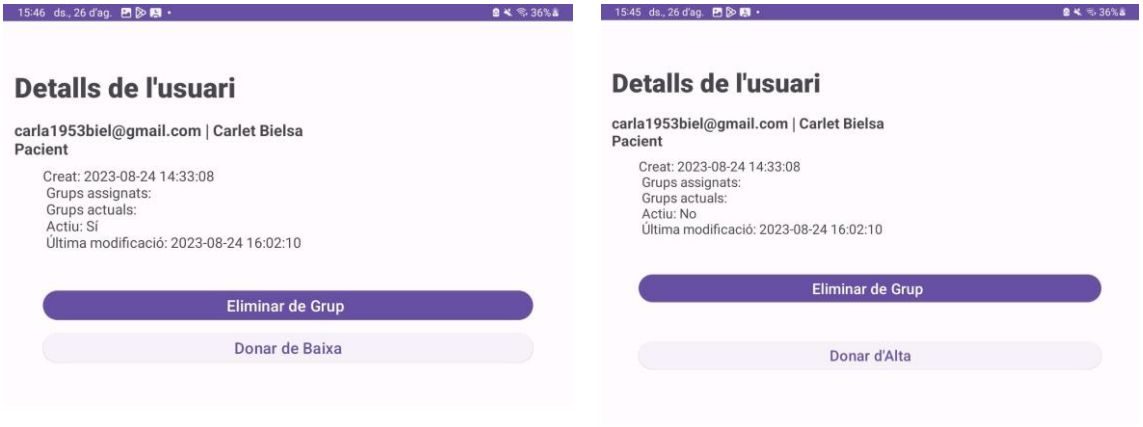


Figura 36. Captura del diseño 'activity_edituser.xml'

11. **AdminPermissionActivity:**

- **Función:** Proporciona al administrador la capacidad de gestionar los permisos de usuarios terapeutas, pudiendo asignar o revocar el permiso de administrador. Esta función se ha encapsulado aparte, porque es una función de mucha importancia. Por lo tanto, es preferible que se encuentre separada para evitar confusiones y errores.
- **Vista:** Una vez se introduce el correo y se pulsa el botón, la aplicación responde con mensajes (*toast*) para dar *feedback* al usuario, como se puede observar por pantalla.

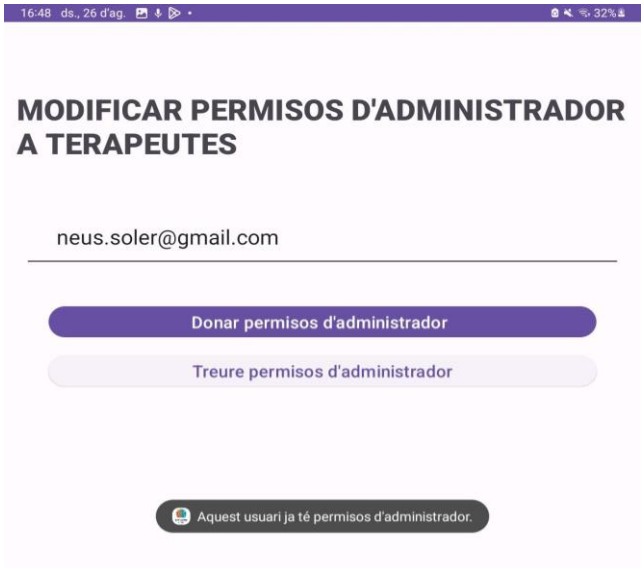


Figura 37. Captura del diseño 'activity_adminpermission.xml'

12. **ChangeAccountsStateActivity:**

- **Función:** Ofrece herramientas para modificar el estado de las cuentas de usuario, permitiendo activaciones o desactivaciones según sea necesario. Esta función ya se puede realizar desde editar la cuenta de un usuario, pero se ha añadido por si podía ser más clara para el administrador.
- **Vista:** Esta vista también muestra los dos botones de activar y desactivar, pero si se intenta realizar una acción como activar a una cuenta activa, la aplicación también responde al usuario. La gestión se realiza mediante *toast*.

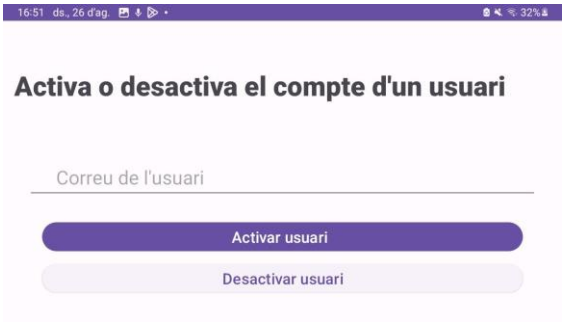


Figura 38. Captura del diseño 'activity_changeaccountsstate.xml'

6.4.3 Package: therapistactivities

Este paquete engloba las actividades diseñadas específicamente para los terapeutas de la aplicación, proporcionando herramientas fundamentales para la gestión, supervisión y seguimiento de pacientes, actividades y sesiones. A continuación, se detalla cada actividad, su función principal y una representación visual basada en las vistas de usuario:

1. **HomeTherapistActivity:**
    - **Función:** Presenta una interfaz inicial para los terapeutas, ofreciendo una visión general de sus funciones y opciones.
    - **Vista:** El botón de "Cerrar sesión" se ha puesto en otro color para poder diferenciarlo más fácilmente.

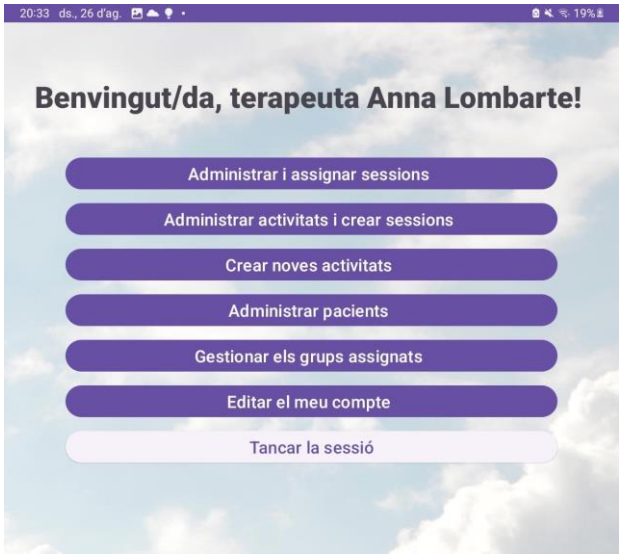


Figura 39. Captura del diseño 'activity_hometherapist.xml'

2. **CreateNewGameActivity:**
    - **Función:** Permite a los terapeutas crear nuevos juegos personalizados para sus pacientes. En esta pantalla pueden escoger qué tipo de juego quieren.
    - **Vista:** Por ahora solo hay 3 juegos, por lo que se ha decidido hacer de esta forma. Si hubiese muchos juegos, podrían añadirse iconos para poder diferenciar a primera vista de qué botón se trata cada juego.

Figura 40. Captura del diseño 'activity_createnewgame.xml'

3. **SessionHistoryActivity:**
    - **Función:** Ofrece un registro histórico de las sesiones llevadas a cabo por un paciente y sus intentos y fallos en cada ejercicio, facilitando el seguimiento y revisión de progresos y resultados. Se muestra el tiempo de inicio del ejercicio (si lo ha iniciado) y el tiempo de finalización (si lo ha completado). Esto permite saber la duración del ejercicio. Además, se almacenan por cada intento que realiza cuántos fallos ha realizado y en qué momento.
    - **Vista:** Al inicio se presenta toda la información del paciente. A continuación, cada *card* violeta es una sesión. Muestra el nombre de la sesión y la fecha de asignación. Además, incluye diferentes actividades como *cards* blancas que recogen los datos de realización de la actividad. También se ha añadido un botón "Detalles de la actividad" que permite ver los parámetros de ese ejercicio, para así poder comparar con los datos que ha obtenido el paciente y realizar un análisis más rápido.

Figura 41. Captura del diseño 'activity_session_history.xml'

4. **ShowActivitiesAndCreateSessionsActivity:**
    - **Función:** Presenta un listado de actividades disponibles y permite la creación de sesiones basadas en estas. Comparando esta vista con la del administrador, en esta se añade también el nivel, para facilitar la creación de sesiones de los terapeutas, y unos *checkboxs* que les permiten realizar la selección para añadirlo a sesiones. Estas dos nuevas funcionalidades permiten, o bien añadir las actividades seleccionadas a una sesión existente, o bien crear una nueva sesión con esas actividades. Los terapeutas no pueden ver actividades eliminadas.
    - **Vista:** Para añadir las actividades a sesiones ya creadas, se muestra una lista y pueden realizar selección múltiple. Si esta actividad ya formaba parte de la sesión, no se vuelve a añadir de nuevo. La opción de creación de nueva sesión solicita un nombre, que se comprueba que no exista previamente en la base de datos. El botón "Cancelar" deselecciona todos los *checkboxs*.

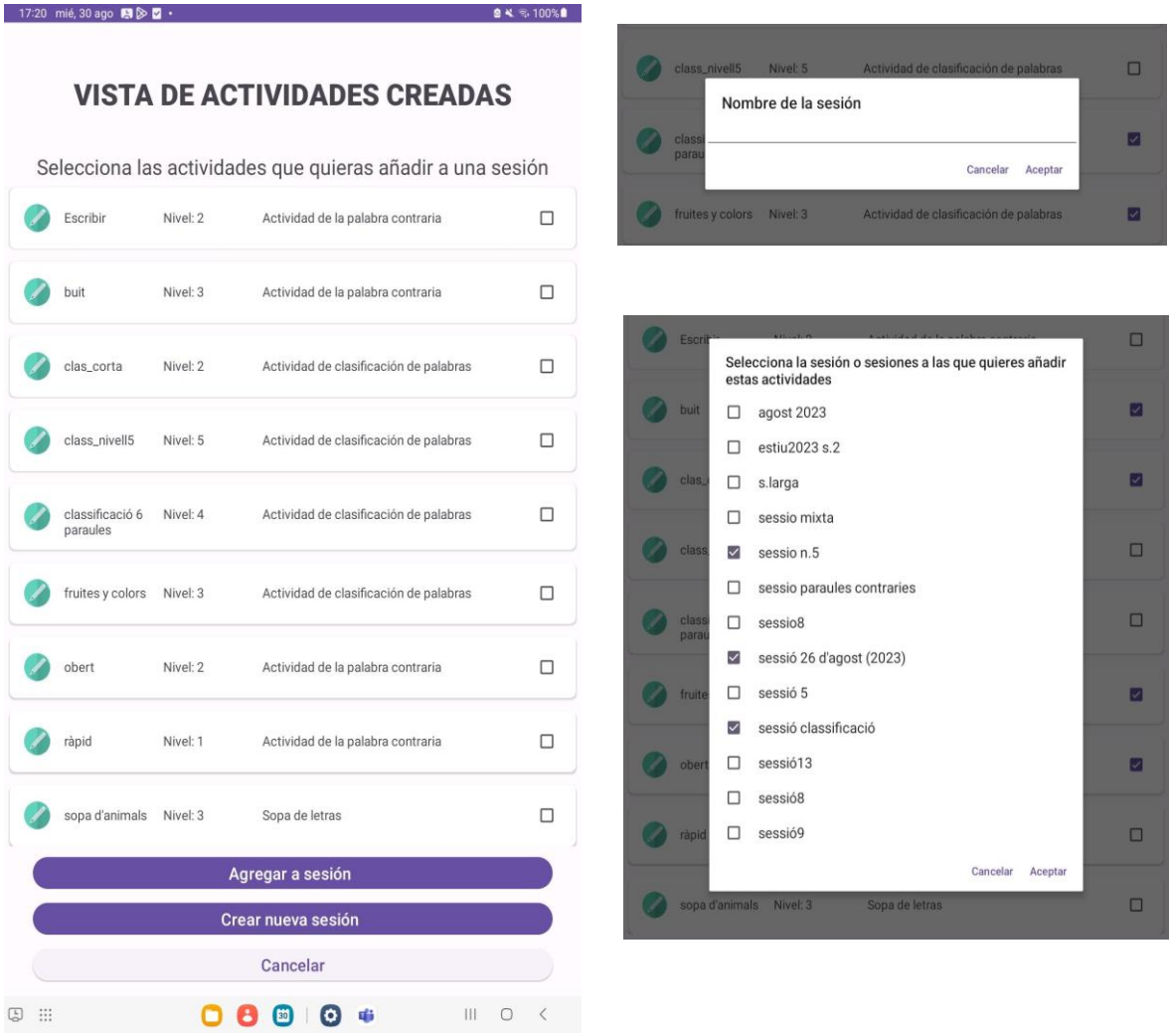


Figura 42. Captura del diseño 'activity_show_activities.xml'

5. **ShowAndAssignSessionsActivity:**
    - **Función:** Proporciona una lista de sesiones y facilita la asignación de estas a pacientes específicos. Es similar a la vista de administradores; sin embargo, aquí la ventana de detalles de la sesión permite también “Asignar”. Esta función permite asignar las sesiones a los pacientes de sus grupos. Los terapeutas no pueden ver sesiones eliminadas.
    - **Vista:** Para asignar una sesión, se muestran solo pacientes que compartan grupo con el usuario terapeuta. Además, se muestra la sesión asignada previamente y si está completada. La asignación puede ser múltiple.

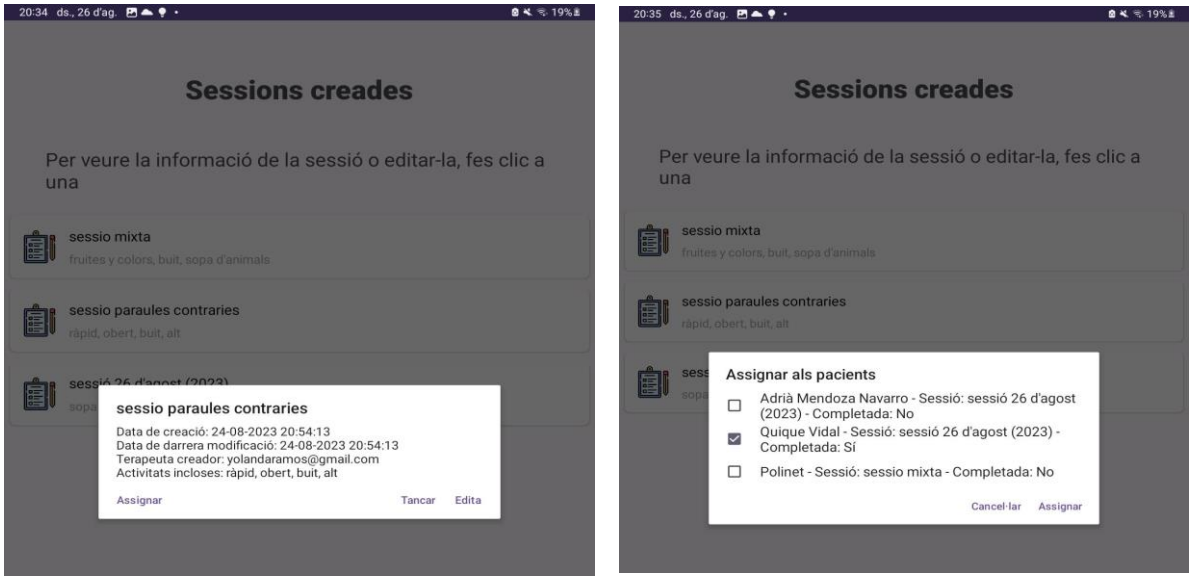


Figura 43. Captura del diseño ‘activity_sessions_assign.xml’

6. **ShowGroupPatientsActivity:**
    - **Función:** Muestra el listado de pacientes que pertenecen a los grupos a los cuales está asignado también el terapeuta, facilitando su gestión y seguimiento. Esta vista difiere con la de administrador en la ventana de información detallada. Esta tiene la función de eliminar al paciente de un grupo (puede eliminarlo de los grupos en los que compartan paciente-terapeuta) y también puede acceder a su historial. Sin embargo, no puede editar su cuenta.
    - **Vista:** La eliminación de grupo muestra el listado de los grupos actuales del paciente seleccionado, en los cuales el terapeuta también está asignado, y permite eliminarlos de forma múltiple. Si se accede al historial, se abre la actividad “SessionHistoryActivity”.

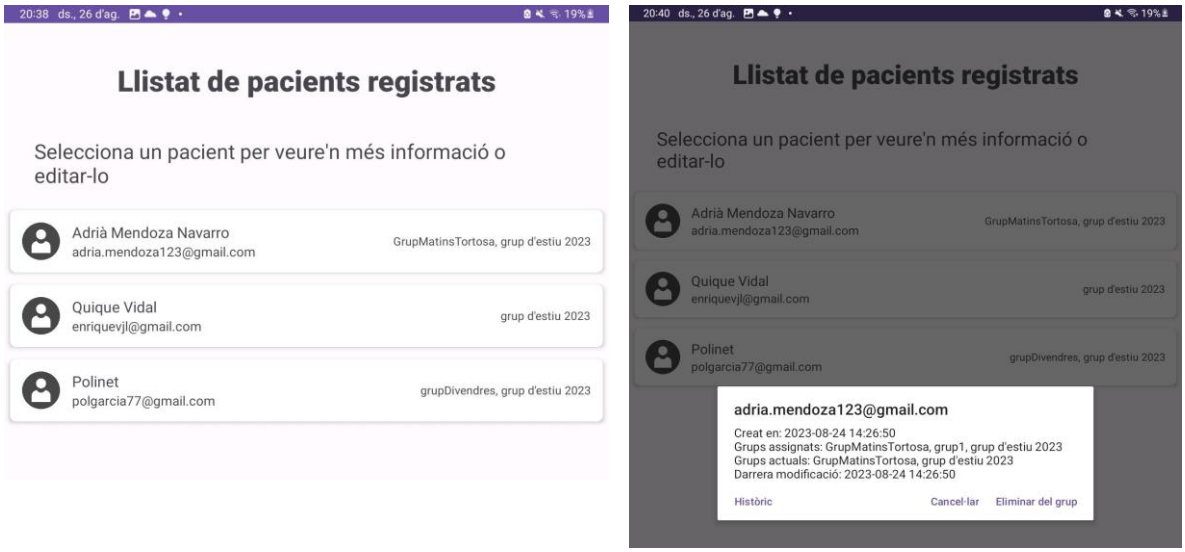


Figura 44. Captura del diseño ‘activity_show_groups.xml’ para terapeutas

6.4.4 Package: patientactivities

Este paquete se centra en las actividades diseñadas específicamente para los pacientes, proporcionando herramientas e interfaces para que desarrollen su experiencia y participación en las sesiones y actividades. Los diseños de estas interfaces se han intentado realizar sencillos y claros, debido al tipo de usuario que los va a utilizar.

1. **HomePatientActivity:**
    - **Función:** Proporciona una interfaz inicial para los pacientes, mostrando su información, si tiene o no grupos asignados, y/o sesión asignada pendiente para realizar.
    - **Vista:** Para cada caso se ha diseñado una vista diferente que guie al usuario de la mejor forma posible y que sea visual.

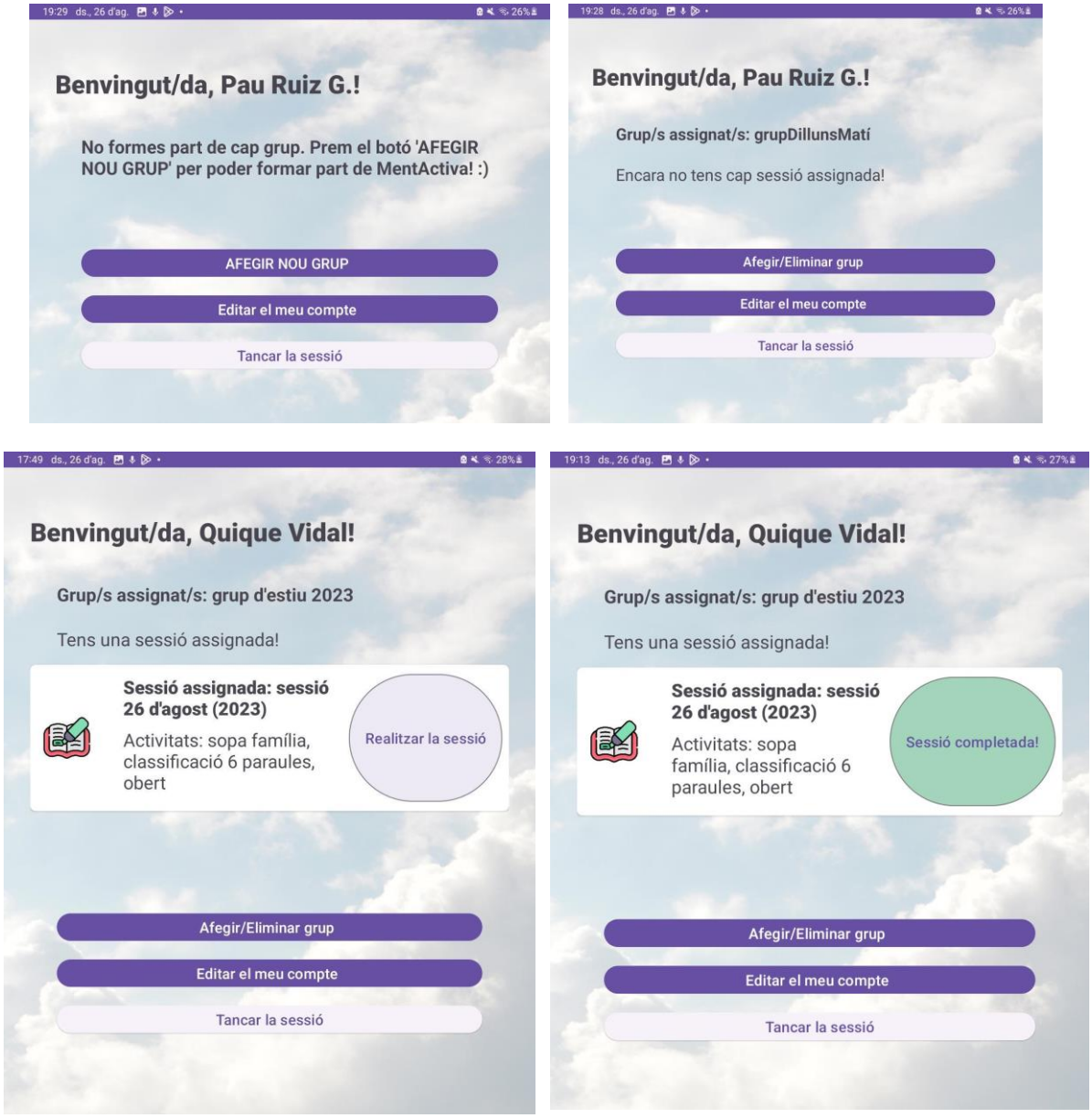


Figura 45. Captura del diseño ‘activity_home_patient.xml’

2. **DoSessionActivity:**
    - **Función:** En caso de que tengan una sesión asignada, pueden acceder a esta pantalla. Aquí se muestran las actividades que incluye la sesión. A medida que realizan las actividades, estas se van marcando automáticamente como completadas (check morado) y no permite volver a acceder.
    - **Vista:** Una vez todas las actividades están marcadas como completadas, se muestra un mensaje de felicitaciones. Si realizan clic sobre alguna actividad, se les mostrará un mensaje de que no se puede realizar porque ya está completada.

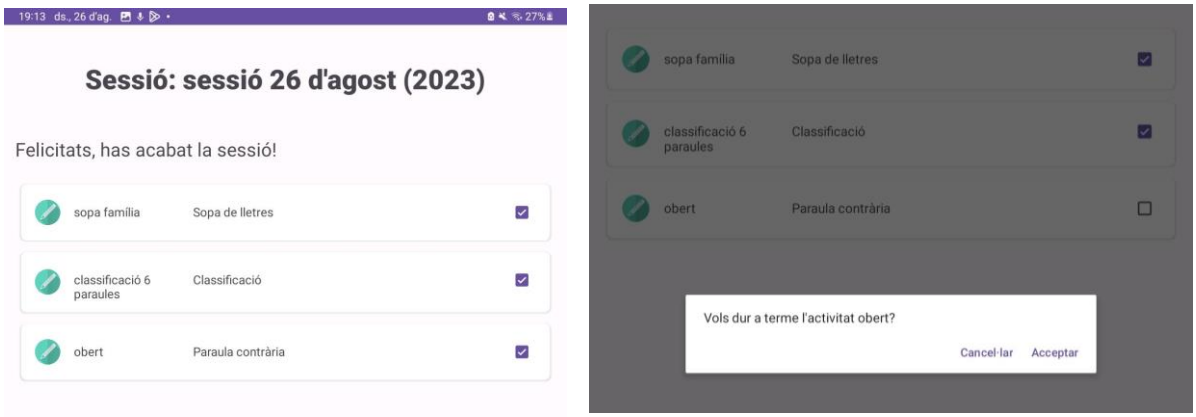


Figura 46. Captura del diseño ‘activity_do_session.xml’

6.4.5 Package: games.createGames

Este paquete está destinado a la creación de nuevas versiones de los juegos, y el código de actividad de los juegos en sí mismos, ofreciendo interfaces específicas para diferentes tipos de actividades.

En estas actividades son los usuarios terapeutas los que introducen los datos para crear la actividad. Por lo tanto, se han tenido que añadir restricciones. Se comprueba el tipo de dato que se puede introducir, valores que pueden dar errores, o nombres de juegos que ya están creados y, por lo tanto, no puede crearse uno nuevo con esas características.

1. **CreateClassificationActivity:**
    - **Función:** Permite la creación de juegos basados en la clasificación de elementos.
    - **Vista:** Se muestra un texto sobre qué se debe introducir en cada lugar y se dan indicaciones. No se permite la creación hasta que todos los parámetros están completados y los datos son correctos. Se van mostrando mensajes de error concretos con los cambios necesarios para poder crear el juego.

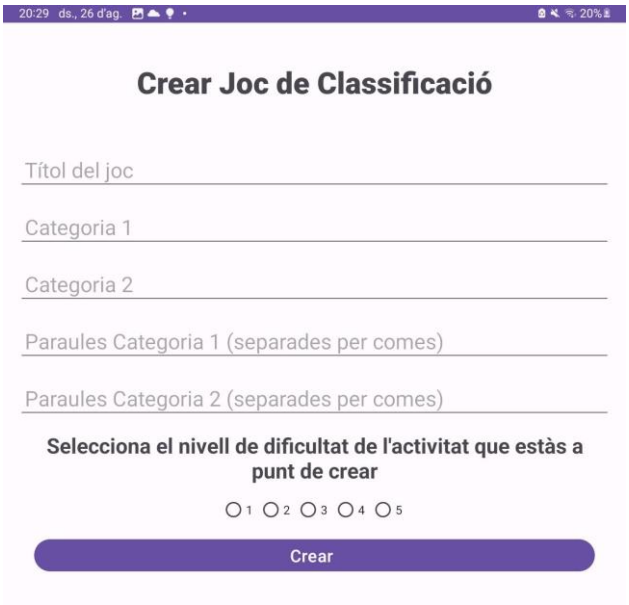


Figura 47. Captura del diseño ‘activity_create_classification.xml’

2. **CreateContraryWordActivity:**
    - **Función:** Facilita la creación de juegos que trabajan con palabras opuestas o contrarias.
    - **Vista:** En este caso se utiliza la palabra como identificador del juego.

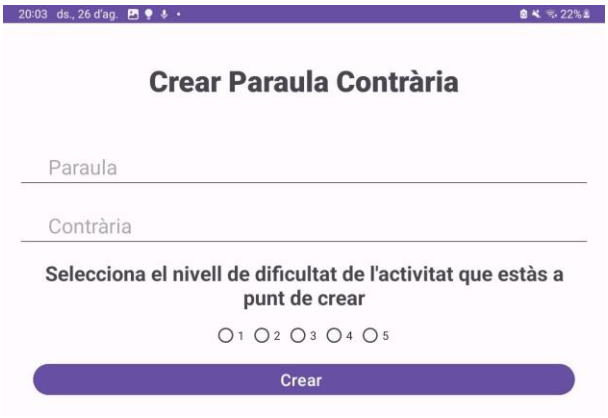


Figura 48. Captura del diseño ‘activity_create_contrary.xml’

3. **CreateWordSoupActivity:**
    - **Función:** Ofrece una herramienta para crear sopas de letras personalizadas.
    - **Vista:** En este caso, además de los datos escritos, se deben seleccionar las orientaciones en que la sopa se va a crear; se debe seleccionar como mínimo una orientación (diagonal, vertical o horizontal) para que se pueda crear.

Figura 49. Captura del diseño ‘activity_create_word_soup.xml’

Subpaquetes de juegos:

Este paquete contiene el código para la ejecución de juegos diseñados a partir de actividades especificadas por terapeutas. Estos juegos son generados dinámicamente, basados en los parámetros almacenados en la base de datos. Al finalizar un juego, el sistema brinda una retroalimentación positiva a través de un mensaje de felicitación y redirige al usuario a la pantalla 'activity_do_session.xml', donde la actividad que acaba de completar se marca como finalizada.

- **Package wordsoupgame: WordSoupActivity / WordSoupViewModel**
    - **Función:** Este paquete está diseñado para manejar y presentar el juego de sopa de letras. Se encarga de generar el ejercicio de forma dinámica para los pacientes, en función de los parámetros especificados por el terapeuta: tamaño del tablero, palabras a buscar y orientaciones permitidas.
    - **Vista:** Cuando se pulsa una letra, esta se marca de color azul. Si se vuelve a pulsar o hacer clic en el botón cancelar, se desmarca. Para validar una palabra se debe pulsa el botón verde de "Enviar". Si la palabra es correcta, se queda marcada de color verde; si la palabra es incorrecta, se queda marcada en rojo durante un periodo muy corto de tiempo y luego vuelve a su estado normal. Cada vez que se acierta una palabra, el contador aumenta.
    - ***Anotación detallada de la funcionalidad:** Para garantizar que las palabras se distribuyan adecuadamente en el tablero, se ha ideado un algoritmo eficiente que combina aleatoriedad y lógica. La mecánica es la siguiente:
        1. Se determina aleatoriamente una coordenada inicial (startRow y startCol) para la palabra.
        2. Se decide si la palabra se coloca de forma regular o invertida, dependiendo si el juego permite palabras en reverso.
        3. Posteriormente, se verifica si es posible ubicar la palabra en alguna de las orientaciones permitidas: horizontal, vertical o diagonal.
        4. Si la palabra puede ser colocada en alguna de estas orientaciones sin conflictos, se ubica. Si no, se verifica la siguiente orientación. Si ninguna orientación es viable, el proceso se repite generando nuevas coordenadas iniciales.

```
for (word in wordsToFind) {
    var placed = false
    while (!placed) {
        val startRow = Random.nextInt(size)
        val startCol = Random.nextInt(size)
        val isReverse = gameData.reverse && Random.nextBoolean()
        val wordToPlace = if (isReverse) word.reversed() else word

        if (gameData.horizontal || gameData.vertical ||
gameData.diagonal) {
            val orientation = listOf(gameData.horizontal,
gameData.vertical, gameData.diagonal).filter { it }.random()

            if (orientation == gameData.horizontal &&
canPlaceHorizontal(wordToPlace, startRow, startCol, emptyGrid)) {
                placeHorizontal(wordToPlace, startRow, startCol,
```

```
emptyGrid)
                placed = true
            } else if (orientation == gameData.vertical &&
canPlaceVertical(wordToPlace, startRow, startCol, emptyGrid)) {
                placeVertical(wordToPlace, startRow, startCol, emptyGrid)
                placed = true
            } else if (orientation == gameData.diagonal &&
canPlaceDiagonal(wordToPlace, startRow, startCol, emptyGrid)) {
                placeDiagonal(wordToPlace, startRow, startCol, emptyGrid)
                placed = true
            }
        }
    }
}
```

Código 6. Este fragmento pertenece a 'WordSoupViewModel' y es responsable de posicionar las palabras en la sopa de letras.

Figura 50. Captura del diseño 'activity_word_soup.xml'. En este caso la sopa tiene palabras en horizontal, vertical y reversas.

- **Package classification game: ClassificationActivity / ClassificationViewModel**
    - **Función:** Implementa el juego de clasificación de palabras para los usuarios, a partir de unas categorías dadas y unas palabras que pertenecen a una u otra. El juego finaliza cuando todas las palabras se han clasificado correctamente.
    - **Vista:** Se muestra una palabra y debe clasificarse en una de las dos categorías seleccionando el *checkbox* correspondiente. Se han utilizado líneas divisorias para mejorar la claridad de la pantalla. Como errores se contabilizan huecos en blanco y palabras mal clasificadas.

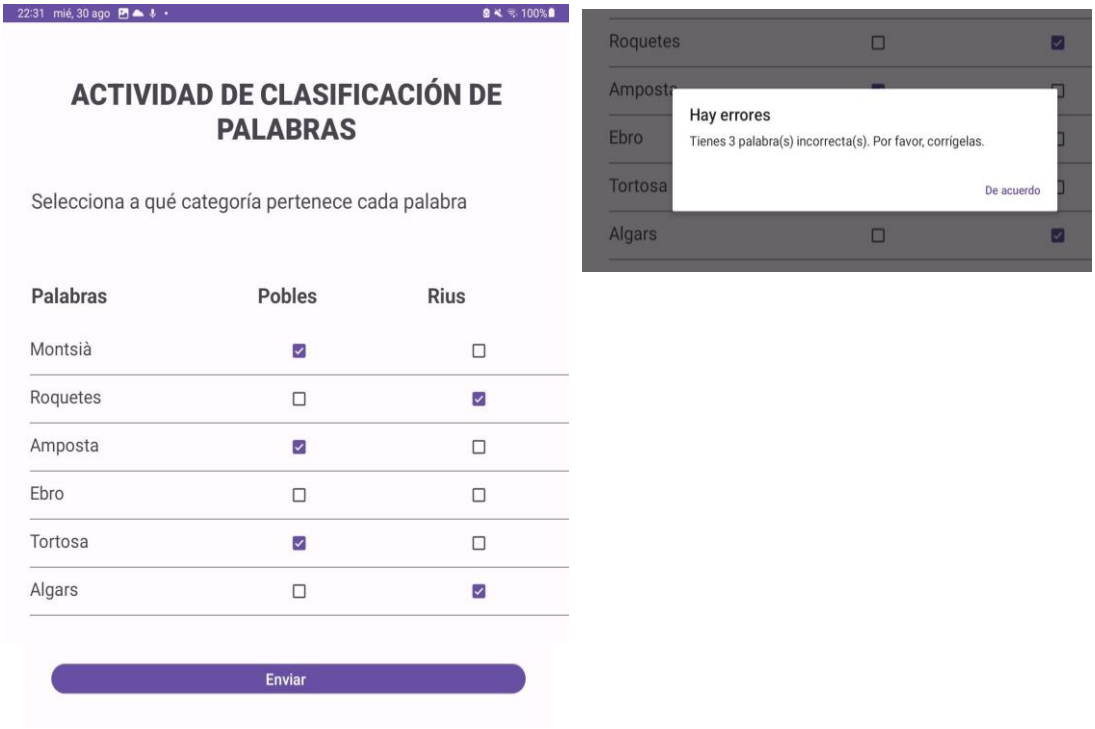


Figura 51. Captura del diseño ‘activity_classification.xml’.

- **Package contrarywordgame: ContraryWordActivity / ContraryWordViewModel**
    - **Función:** Implementa el juego de palabras contrarias para los usuarios. Se muestra una palabra por pantalla y el usuario debe escribir la contraria.
    - **Vista:** Se muestra la palabra en la parte superior y con tamaño más grande, y debajo el cuadro de texto donde se debe escribir la palabra contraria. La comprobación se ha realizado para que posibles espacios en blanco o mayúsculas o minúsculas no influyan en la comprobación y la palabra se dé por válida. Para realizar un intento, debe pulsar el botón "Enviar". Si se pulsa el botón "Cancelar", se borran las letras introducidas.

Figura 52. Captura del diseño 'activity_contrary_word xml'.

6.5 Gestión de datos

La gestión óptima de datos en cualquier aplicación moderna es fundamental para asegurar una funcionalidad ininterrumpida. En el caso de esta aplicación, como se ha mencionado previamente se ha utilizado Firebase, específicamente Firestore, para llevar a cabo la gestión de datos en tiempo real. Todo el proceso de gestión de datos se lleva a cabo a través de APIs proporcionadas por Firebase, asegurando operaciones eficientes y escalables.

6.5.1 Lectura de datos

Cuando un usuario accede a la aplicación, se realizan llamadas asíncronas a Firestore para recuperar datos relevantes para ese usuario. Estas llamadas utilizan las referencias de documentos o colecciones específicas y para obtener los datos.

Que las llamadas sean asíncronas significa que, cuando se hace una llamada, la aplicación no se bloquea ni espera a que se complete; por el contrario, la ejecución del código

continua. Esto es especialmente útil en aplicaciones móviles, donde bloquear el hilo principal podría causar que la aplicación no responda. Para la gestión de las llamadas asíncronas existen los métodos '.addOnSuccessListener' y '.addOnFailureListener', que son "callbacks" (o funciones de devolución de llamada) que proporcionan una manera de definir lo que quieres que ocurra cuando la llamada asíncrona se complete con éxito o falle, respectivamente.

A continuación, se muestran unos ejemplos de cómo se ha realizado la lectura de datos en este proyecto.

El *Código 7* muestra la función **fetchExercisesAndAttemps(…)** de **SessionHistoryActivity**. Esta función tiene el propósito de recuperar los ejercicios y sus intentos asociados a una sesión específica para un usuario determinado y mostrarlos en un RecyclerView. Para ello se consulta la base de datos, se acede a la subcolección 'sessionHistory' del usuario, luego se selecciona un documento específico (la sesión determinada por sessionId), y finalmente apuntamos a la subcolección 'exercises' de esa sesión.

- **get()**: Es un método asíncrono que intenta recuperar todos los documentos (en este caso, ejercicios) dentro de la subcolección 'exercises'.
- **.addOnSuccessListener { querySnapshot -> ... }**: Esta es una función de *callback* que se invoca si la consulta a Firestore tuvo éxito. El parámetro **querySnapshot** contiene los resultados de la consulta y tiene una propiedad llamada 'documents' que es una lista de todos los documentos recuperados: **val exercises = querySnapshot.documents**.
- **.addOnFailureListener { e -> ... }**: Esta es otra función de *callback* que se invoca si hubo un error en la solicitud a Firestore. El parámetro e contiene detalles sobre el error. En este caso, simplemente estamos registrando el error usando Log.e().

```
// Se genera unas instancia de Firebase.firestore
private val db = Firebase.firestore

private     fun     fetchExercisesAndAttempts(sessionId:     String,
recyclerView: RecyclerView) {
    userId?.let {
        db.collection("users").document(it)
            .collection("sessionHistory").document(sessionId)
            .collection("exercises")
            .get()
            .addOnSuccessListener { querySnapshot ->
                val exercises = querySnapshot.documents
                recyclerView.layoutManager                          =
LinearLayoutManager(this@SessionHistoryActivity)
                recyclerView.adapter = ExercisesAdapter(exercises)
            }
            .addOnFailureListener { e ->
                Log.e(ContentValues.TAG,       "Error       fetching
exercises", e)
            }
    }
}
```

Código 7. Función fetchExercisesAndAttemps() de 'SessionHistoryActivity'

En el *Código 8* del LoginViewModel se intenta autenticar al usuario y luego recuperar su información. La autenticación se hace primero y, si es exitosa, se procede a obtener los datos del usuario de Firestore. En este proceso, se demuestra cómo se pueden recuperar campos específicos de un documento.

```
// Se generan unas instancias de Firebase
private val auth: FirebaseAuth = Firebase.auth
private val db = Firebase.firestore

// En la función se intenta iniciar sesión usando FirebaseAuth
auth.signInWithEmailAndPassword(email.value.toString(),
password.value.toString())
    .addOnCompleteListener { task ->
        if (task.isSuccessful) {

            // Una vez que la autenticación es exitosa, se recupera
            //la información del usuario de Firestore
            db.collection("users")
                .document(email.value ?: "")
                .get()
                .addOnSuccessListener { document ->

                    // Se recuperan los campos específicos del
                    //documento del usuario
                    val isActive = document.getBoolean("isActive")
                        ?: false
                    ...
                }
                ...
        }
    }
```

Código 8. Código extraído de 'LoginViewModel'

6.5.2 Escritura y actualización de datos

Cuando un usuario realiza una acción que necesita registrar datos nuevos o modificar datos existentes, la aplicación realiza llamadas para escribir o actualizar documentos en Firestore en tiempo real. Dado que Firestore es una base de datos NoSQL, podemos añadir o modificar campos de documentos sin alterar la estructura general del documento.

El *Código 9* de EditProfileActivity muestra cómo se pueden actualizar diferentes campos en la base de datos. En este caso se muestra cómo se editan valores propios de los usuarios.

- **updatePassword():** Es un método específico diseñado para cambiar la contraseña de un usuario. Al ser un dato sensible y que requiere un manejo especial por cuestiones de seguridad, esta función se encarga directamente de este proceso. Para su uso, solo se necesita proporcionar la nueva contraseña que se desea establecer.
- **update():** Para la mayoría de las otras actualizaciones, se utiliza el método update(). Este método es versátil y permite modificar uno o múltiples campos en un documento de Firestore. Si solo se desea actualizar un campo específico, simplemente se proporciona el nombre del campo y el nuevo valor que se quiere asignar (en este ejemplo para actualizar el nombre). Si la intención es modificar varios campos simultáneamente, se utiliza una estructura hashMapOf<String,Any>. Esta estructura permite definir pares de clave-valor, donde la clave es el nombre del campo y el valor es la nueva información que se desea asignar. Gracias a esta técnica, en una sola llamada a Firestore, es posible actualizar múltiples atributos del documento (en este caso se actualiza el estado de la cuenta a inactivo, se eliminan sus grupos actualmente asignados y se modifica el tiempo del último cambio).

```
// Actualizar la contraseña del usuario
user.updatePassword(newPassword)...

// Mostrar diálogo para cambiar el nombre del usuario
private fun changeName(newName: String) {
    ...
    // Actualizar el campo 'name' del documento del usuario
    docRef.update("name", newName)...
}

// Deshabilitar la cuenta del usuario actualizando el campo
//'isActive' a false
private fun disableAccount() {
    ...
    // Actualizar varios campos en el documento del usuario
    val updates = hashMapOf<String, Any>(
        "isActive" to false,
        "currentGroups" to emptyList<String>(),
        "lastStatusChange" to FieldValue.serverTimestamp()
    )
    docRef.update(updates)...
}
```

Código 9. Código extraído de 'EditProfileActivity'

6.6 Seguridad y autenticación

Garantizar la seguridad de los datos y la privacidad de los usuarios es primordial. Para ello se han utilizado herramientas y se han tenido en cuenta diversos aspectos:

- **Autenticación**: Se ha utilizado Firebase Authentication para gestionar la autenticación de usuarios. Esto proporciona múltiples métodos de autenticación, desde correo electrónico y contraseña hasta proveedores de terceros como Google y Facebook. Cuando un usuario intenta iniciar sesión, Firebase verifica sus credenciales y, si son válidas, proporciona un token de autenticación único que se utiliza para identificar al usuario en todas las futuras interacciones con la aplicación hasta que cierre sesión. Por ahora, solo se ha implementado el acceso con cuenta de Google (Gmail).
- **Encriptación**: Firebase utiliza la encriptación en tránsito y en reposo. Esto significa que los datos se cifran automáticamente cuando se envían entre la aplicación y Firestore (protocolo TLS[12]) y también cuando se almacenan en la base de datos (con protocolos estándares de la industria como AES256 o AES128). Este nivel de seguridad garantiza que, incluso si los datos se interceptan de alguna manera, no serán legibles sin la clave de descifrado adecuada.
- **Gestión de Roles y Accesos**: La aplicación implementa un sistema robusto de roles para asegurar que cada usuario solo acceda a la información que le corresponde y a las funciones permitidas. Se definen tres roles principales: ‘isAdmin’, ‘isTherapist’, e ‘isPatient’. Un usuario designado como ‘isPatient’ únicamente puede acceder a la información de su propia sesión asignada, y este acceso se habilita al recibir y utilizar un código de grupo. Por otro lado, un terapeuta (isTherapist) puede ser elevado al rol de administrador (isAdmin) solo si otro administrador le otorga dicho permiso. Es importante destacar que existe un administrador maestro, único en su clase, que posee exclusivamente permisos de ‘isAdmin’. Este administrador maestro es inmutable: ningún otro usuario, incluso otros administradores, puede modificar sus permisos o dar de baja su cuenta. Esta salvaguarda garantiza una constante integridad y control sobre el sistema. Adicionalmente, para añadir una capa extra de seguridad, las cuentas de los terapeutas son creadas únicamente por administradores, asegurando así que sólo personal confiable tenga acceso a la información detallada y sensible de los pacientes. En cuanto al control de accesos, un administrador tiene la capacidad de revocar el acceso a la aplicación, tanto a terapeutas como a pacientes, en cualquier momento, garantizando que, ante cualquier inconveniente, estos usuarios dejen de tener acceso total a la aplicación de manera inmediata. También los pacientes pueden ser eliminados de grupos por el administrador, o por el terapeuta en caso de que se trate de sus propios grupos asignados.

[12] TLS = Transport Layer Security

7 Evaluación

La evaluación es un componente esencial en el ciclo de vida del software, permitiendo identificar fortalezas y áreas de mejora. En esta sección, se describen las métricas y criterios utilizados para evaluar la aplicación, así como los resultados obtenidos y el *feedback* recopilado.

Para garantizar que una aplicación cumple con los estándares de calidad y puede satisfacer las necesidades de los usuarios, existen las siguientes métricas y criterios:

- **Usabilidad**: Se evaluó la facilidad de uso, la intuición del diseño y la satisfacción general del usuario.
- **Rendimiento**: Se midió la rapidez de respuesta de la aplicación, tiempos de carga y la fluidez de las transiciones entre actividades.
- **Fiabilidad**: Se evaluó la capacidad de la aplicación para funcionar sin fallos en diversas condiciones y dispositivos.
- **Seguridad**: Se analizaron aspectos como la eficacia de la autenticación y la protección de datos.
- **Compatibilidad**: Se verificó el rendimiento de la aplicación en diferentes dispositivos y versiones de Android.

Teniendo en cuenta que esta es una primera versión no definitiva de la aplicación, esta evaluación se centró en los tests funcionales.

7.1 Tests funcionales

En este apartado se analiza y se pone a prueba que los requisitos funcionales de la aplicación se garantizan, incluso en las condiciones más adversas.

Estos tests se han realizado teniendo en cuenta que no hay ningún problema o anomalía en el servidor, es decir, que hay conexión a internet y se puede acceder a los datos de Firebase. En caso de que no funcione Firebase o la conexión a este, la aplicación no tendría datos con los que trabajar.

7.1.1 Secuencia de test para administradores

PRUEBA	RESULTADO ✓/✗
Iniciar la aplicación	- Se inicia correctamente. ✓
Iniciar sesión con una cuenta de administrador.	- Se abre la sesión dando la bienvenida al administrador. La aplicación ofrece diferentes opciones que se podrán realizar. ✓
Administración de terapeutas	
Acceder al listado de terapeutas.	- Se accede al listado correctamente. Se muestra el correo y el nombre. Los terapeutas dados de baja se muestran mediante un sombreado gris. ✓
Vista de la información del terapeuta.	- Al pulsar encima de cada terapeuta se abre la vista de información con: correo, fecha de creación, grupos asignados, grupos actuales, última modificación de estado de la cuenta y si tiene permiso de administrador. ✓

Eliminar terapeuta de un grupo.	- La aplicación permite seleccionar el/los grupo/s que se desean eliminar para cada terapeuta. ✓ - Se actualizan tanto el campo de grupos del terapeuta como los terapeutas asignados a dicho grupo. ✓
Dar de alta a un terapeuta.	- Funciona correctamente. Aparece un mensaje para confirmar la activación de la cuenta. La *card* del terapeuta se muestra de color blanco. ✓ - El parámetro de último cambio de estado de cuenta se actualiza. ✓
Dar de baja a un terapeuta.	- Funciona correctamente. Aparece un mensaje para confirmar la desactivación de la cuenta. La *card* del terapeuta se muestra de color gris. ✓ - El parámetro de último cambio de estado de cuenta se actualiza. ✓ - Se elimina al terapeuta de todos los grupos en los que estaba asignado. ✓ - Se eliminan los grupos que el terapeuta tenía activos en ese momento. ✓
Diferenciación de terapeutas activos, y terapeutas dados de baja.	- Las *cards* de los terapeutas activos se muestran sin sombreado. Las *cards* de los terapeutas dados de baja se muestran con un sombreado gris. ✓
Dar de baja a un terapeuta que ya se había dado de baja anteriormente.	- En este caso aparece un nuevo botón que sustituye el anterior "Dar de Baja" que nos permite dar de alta otra vez al terapeuta. ✓ - Cuando se da de baja al terapeuta, se eliminan sus grupos actuales. ✓
Crear nuevo terapeuta.	- Se abre una ventana donde, después de introducir Nombre, Email y Contraseña, permite crear el nuevo terapeuta. ✓
Crear nuevo terapeuta sin haber rellenado todos los campos, con un correo sin dominio @gmail.com, o con una contraseña inferior a 8 caracteres.	- La cuenta del nuevo terapeuta no se crea. ✓
Administración de pacientes	
Acceder al listado de pacientes.	- Se accede al listado correctamente. Se muestra el nombre, el correo y los grupos actuales. ✓
Vista de la información del paciente.	- Al pulsar encima de cada paciente podemos conocer su correo, la fecha de creación, grupos asignados, grupos actuales y la última modificación de la configuración de este. Además, podemos ver los grupos actuales a la derecha del usuario sin pulsar encima de cada paciente. ✓
Eliminar paciente de un grupo.	- La aplicación permite seleccionar el/los grupo/s que se desean eliminar para cada paciente. ✓

Dar de baja a un paciente.	- Funciona correctamente. Aparece un mensaje para confirmar la desactivación de la cuenta. La *card* del paciente se muestra de color gris. ✓ - Se elimina al paciente de todos los grupos en los que estaba asignado. ✓ - Se eliminan los grupos que el paciente tenía asignados en ese momento. ✓
Diferenciación de pacientes activos, y pacientes dados de baja.	- Las *cards* de los pacientes activos se muestran sin sombreado. Las *cards* de los pacientes dados de baja se muestran con un sombreado gris. ✓
Dar de baja a un paciente que ya se había dado de baja anteriormente.	- En este caso aparece un nuevo botón que sustituye el anterior "Dar de Baja", que nos permite dar de alta otra vez al paciente. ✓
Eliminar paciente dado de baja de un grupo.	- Cuando se ha dado de baja al paciente, se le ha eliminado de los grupos actuales. ✓
Administración de grupos	
Acceder al listado de grupos.	- Se accede al listado correctamente. Se muestra el nombre, la ubicación y el código del grupo. Los grupos dados de baja se muestran bajo un sombreado gris. ✓
Seleccionar un grupo.	- El grupo se selecciona correctamente y, en caso de no estar desactivado, nos muestra la información y nos permite editarlo. ✓
Editar información de un grupo.	- Permite editar el "Lugar" y el "Código". - Permite eliminar terapeutas y pacientes asignados correctamente, actualizando los datos de los grupos actuales de los usuarios. ✓
Eliminar grupo.	- El grupo se elimina correctamente. En el listado de grupos se muestra la *card* bajo un sombreado gris. Los grupos eliminados no se muestran a los terapeutas. ✓
Creación de un nuevo grupo.	- Siempre que los campos "Nombre del Grupo" y "Lugar del Grupo" estén rellenados, se crea el grupo correctamente. ✓
Administración de actividades	
Acceder al listado de actividades.	- Se accede al listado correctamente. Se muestra el nombre, el tipo de actividad, y el correo del terapeuta creador. Las actividades dadas de baja se muestran con un sombreado gris. ✓
Seleccionar una actividad.	- La actividad se selecciona correctamente y nos muestra su información. En caso de que esta no esté eliminada, nos muestra una opción de "Eliminar". ✓

Eliminar actividad.	- La actividad se elimina correctamente. ✓ - En el listado de actividades se muestra la *card* bajo un sombreado gris. ✓ - Las actividades eliminadas no se muestran para los terapeutas y, por lo tanto, no pueden ser añadidas a más sesiones. ✓ - Al eliminar una actividad esta no se elimina de todas las sesiones de las que forma parte. ✗
Administración de sesiones	
Acceder al listado de sesiones.	- Se accede al listado correctamente. Se muestra el nombre y las actividades que contiene. Las sesiones dadas de baja se muestran con un sombreado gris. ✓
Seleccionar una sesión.	- La actividad se selecciona correctamente y nos muestra la información. En caso de que esta no esté eliminada, nos muestra una opción de "Editar". ✓
Eliminar actividades de una sesión.	- La aplicación nos permite eliminar las diferentes actividades que deseemos. ✓ - Si eliminamos todas las actividades de una sesión, esta se mantiene activa, pero sin actividades. ✗
Eliminar una sesión.	- La sesión se elimina correctamente. ✓ - En el listado de sesiones se muestra la *card* eliminada con un sombreado gris. ✓ - Las sesiones se eliminan correctamente y no son visibles para los terapeutas. ✓ - Los pacientes a los que se les ha asignado previamente la sesión antes de eliminarla, siguen teniéndola. ✗
Cambiar permisos terapeuta/administrador	
Modificación de los permisos.	- Siempre y cuando el correo del terapeuta se encuentre en la base de datos, se le pueden dar o quitar los permisos correspondientes. ✓ - Si el correo no corresponde a un usuario con permisos de terapeuta, no se le pueden dar permisos de administrador. ✓ - Si el correo no corresponde a un usuario con permisos de administrador, no se le pueden quitar los permisos de administrador. ✓ - El administrador master no puede retirarse el permiso a sí mismo, ni tampoco lo puede hacer ningún otro usuario. ✓
Dar de alta/baja cuentas específicas	
Activar cuenta de un usuario.	- Siempre y cuando la cuenta del usuario se encuentre en la base de datos y se encuentre desactivada, se activa. ✓ - En caso de que la cuenta ya esté activada, aparece un mensaje conforme el usuario ya está activo. ✓

Desactivar cuenta de usuario.	- Siempre y cuando la cuenta del usuario se encuentre en la base de datos y se encuentre activa, se desactiva. ✓ - En caso de que la cuenta ya esté desactivada, aparece un mensaje conforme el usuario ya está inactivo. ✓
Editar la cuenta	
Cambiar de nombre	- El nombre de la cuenta se actualiza correctamente. ✓
Cambiar de contraseña.	- En caso de introducir correctamente el email y la contraseña actual, si las dos contraseñas nuevas coinciden, esta se puede actualizar correctamente. ✓ - En caso de que el email no coincida con el de la cuenta, no se puede actualizar la contraseña. Aparece un mensaje de error. ✓ - En caso de que la contraseña actual introducida no coincida con la real, no se puede actualizar la contraseña. Aparece un mensaje de error. ✓ - En caso de que la nueva contraseña propuesta por el usuario no coincida en los dos campos, no se puede actualizar la contraseña correctamente. Aparece un mensaje de error. ✓
Dar de baja la cuenta.	- La actividad funciona correctamente. Aparece un mensaje de confirmación y se cierra la aplicación. ✓ - La cuenta del administrador master no puede ser dada de baja. ✓
Cerrar sesión	
Cerrar sesión de la cuenta.	- La sesión de la cuenta se cierra correctamente. ✓

Tabla 35. Secuencia de test para administradores especificando la prueba y el resultado obtenido

Los errores identificados en estos tests no se consideran fallos en sí mismos, sino aspectos a considerar para futuras etapas del desarrollo de la aplicación. En el diseño actual, al eliminar una actividad, esta no se retira de las sesiones existentes en las que ya esté incluida, sino que simplemente se desactiva para prevenir su incorporación en nuevas sesiones. Esta decisión se tomó con el objetivo de mantener una coherencia en las sesiones asignadas a un paciente y las expectativas de resultados.

Por otro lado, al eliminar una sesión, esta no impacta en las que ya fueron asignadas a los pacientes. En lugar de eso, simplemente deja de ser visible y no se permite su asignación a nuevos pacientes. Esta decisión se adoptó para garantizar que, si un terapeuta ya había planificado trabajar con cierta sesión, pueda continuar sin interrupciones, minimizando así la posibilidad de confusiones o problemas inesperados.

No obstante, es importante destacar que cualquier modificación realizada en actividades o sesiones sí afectará a las sesiones previamente asignadas. Esto se debe a que estas sesiones se recargan directamente desde la base de datos, por lo que los cambios se reflejan en tiempo real.

7.1.2 Secuencia de test para terapeutas

PRUEBA	RESULTADO ✓/✗
Iniciar la aplicación	- Correctamente iniciada. ✓
Iniciar sesión con una cuenta de terapeuta.	- Se inicia la sesión si la cuenta existe y la contraseña es correcta. ✓
Opción de escoger rol (si posee permisos de administrador)	- Se muestra la pantalla para seleccionar el rol. ✓ - En rol escogido corresponde con la próxima pantalla de inicio. ✓
Gestión y asignación de sesiones	
Acceder al listado de sesiones.	- Se visualiza correctamente, con el nombre de la sesión y los juegos incluidos. ✓
Conocer la información de la sesión.	- Al seleccionar una sesión, se despliegan sus detalles: fecha de creación, última modificación, correo del terapeuta creador y actividades incluidas. Además, se muestran las opciones “Asignar”, “Cerrar” y “Editar”. ✓
Editar una sesión. - Eliminar actividades - Eliminar sesión	- Muestra los detalles de la sesión correctamente. ✓ - La eliminación de una o varias actividades se hace correctamente. ✓ - La eliminación de la sesión hace que deje de ser visible para los terapeutas. ✓
Asignar una sesión. - Asignar una sesión a paciente - Asignar a paciente sin sesión - Reasignar sesión a paciente	- Al pulsar en la opción “Asignar” en una sesión, se muestra una lista con los diferentes pacientes correspondientes al terapeuta, junto con la sesión que tienen asignada actualmente y si está o no completada. ✓ - En caso de asignar una sesión a un paciente sin ninguna sesión asignada, aparece un mensaje de confirmación y se actualiza la sesión de este. Se marca como no finalizada. ✓ - En caso de asignar una sesión a un paciente con alguna sesión asignada (completada o sin completar), aparece un mensaje de confirmación y se actualiza la sesión de este, cambiando la antigua por la nueva. ✓
Gestión de actividades y creación de sesiones	
Vista de actividades creadas.	- El listado con las actividades creadas se muestra correctamente con: nombre asignado a la actividad, nivel y tipo de juego. - A la derecha aparece un *checkbox* que permite la selección múltiple. ✓ - El botón cancelar deselecciona todos los *checkboxs*. ✓

Añadir actividades a una sesión nueva.	- En el caso de no seleccionar ninguna actividad y pulsar sobre “Agregar a sesión”, aparece un mensaje de aviso “Selecciona al menos una actividad”. ✓ - En el caso de seleccionar una o varias actividades, se abre una casilla para rellenar con el nombre de la sesión. ✓ - Aparece un mensaje de confirmación conforme el terapeuta quiere crear la nueva sesión. ✓ - La sesión se crea correctamente. ✓
Añadir actividades a sesión existente.	- En el caso de no seleccionar ninguna sesión y pulsar sobre “Agregar a sesión” aparece un mensaje de aviso “Selecciona al menos una actividad”. ✓ - Aparece un listado con todas las sesiones actuales y permite añadir las actividades a múltiples sesiones. ✓ - Si la actividad ya estaba presente en la sesión existente, no se duplica, se sobreescribre. ✓
Crear nuevas actividades	
Acceso a la creación de actividades/juegos.	- El usuario puede elegir entre los tres tipos de actividad propuestos: Sopa de letras, Palabra contraria y Clasificación. ✓
Creación de una Sopa de letras.	- Cumple los requisitos para la creación de una sopa de letras. ✓ - Si algún parámetro está vacío o con datos incorrectos, muestra un mensaje de error. - Se valida el número máximo de filas permitido. ✓ - Almacena las palabras en una lista, eliminando las comas y los posibles espacios indeseados. ✓
Creación de un juego de Palabra contraria.	- Muestra un mensaje de error y no se permite la creación del juego, en caso de no completar todos los campos. ✓ - Se almacenan los datos correctamente, eliminando espacios. ✓
Creación de un juego de Clasificación.	- Muestra un mensaje de error y no se permite la creación del juego, en caso de no completar todos los campos. ✓ - Almacena las palabras en una lista, eliminando las comas y posibles espacios indeseados. ✓
Administrar pacientes	
Información del paciente.	- Se muestran solo los pacientes que pertenezcan a los grupos del terapeuta. ✓ - Se muestra el listado correctamente con: nombre, correo y grupos actuales - La selección de un paciente permite ver su información completa: correo, fecha de creación de la cuenta, grupos asignados, grupos actuales, y última modificación de la cuenta. ✓

Eliminar a un paciente.	- Al pulsar la opción “Eliminar”, se muestra un mensaje de confirmación y permite seleccionar el grupo del que se quiere eliminar al paciente. Solo aparecen los grupos que coinciden con los del terapeuta. ✓ - El paciente se elimina correctamente (campos de sus grupos actuales y del grupo en pacientes asignados). ✓
Historial de un paciente.	- Se muestra si la actividad ha sido iniciada, mostrando el momento de inicio. - Se muestra si la actividad ha sido completada, mostrando el momento de finalización. - Se muestra el número de errores que se han hecho, asociado al tiempo del intento. ✓
Botón “Detalles de actividad” dentro del historial del paciente.	- Se muestra una ventana con información detallada del juego. ✓
Administrar los grupos asignados	
Añadir a un nuevo grupo.	- En caso de introducir un código de grupo válido, se añade correctamente. Se añade el grupo a sus grupos actuales, a grupos asignados y se añade el terapeuta a los terapeutas asignados del grupo ✓ - En caso de introducir un código de grupo no válido, o dejar el campo en blanco, aparece un mensaje de error. ✓ - El botón “Cancelar” borra el texto del campo. ✓
Eliminar alguno de los grupos del usuario.	- Muestra solo los grupos que el terapeuta tiene actualmente asignados. ✓ - La selección de uno o varios grupos, los elimina. Actualiza sus parámetros de grupos asignados y el campo de terapeutas asignados del grupo. ✓
Editar la cuenta	
Cambiar de nombre	- El nombre de la cuenta se actualiza correctamente. ✓
Cambiar de contraseña.	- En caso de introducir correctamente el email y la contraseña actuales, si las dos contraseñas nuevas coinciden, esta se puede actualizar correctamente. Permite iniciar sesión con esta nueva credencial. ✓ - No se verifica que la contraseña nueva sea segura. ✗ - En caso de que cualquier dato no coincida, aparece un mensaje de error. ✓
Dar de baja la cuenta.	- Aparece un mensaje de confirmación y, si se acepta, la cuenta se desactiva en ese momento, cerrando la sesión. ✓
Cerrar sesión	
Cerrar sesión de la cuenta.	- La sesión de la cuenta se cierra correctamente. ✓

Tabla 36. Secuencia de tests para terapeutas especificando la prueba y el resultado obtenido

7.1.3 Secuencia de tests para pacientes

PRUEBA	RESULTADO ✓/✗
Iniciar la aplicación	- Se inicia correctamente ✓ - En caso de no tener ninguna sesión asignada, ofrece la opción de unirse a alguna. ✓ - En caso de tener una sesión asignada, permite realizarla. ✓
Página de inicio de paciente. - El paciente no forma parte de ningún grupo. - El paciente forma parte de uno o más grupos, pero no tiene una sesión asignada. - El paciente forma parte de uno o más grupos y tiene una sesión asignada.	- Muestra un mensaje indicando que no es parte de ningún grupo y la indicación para unirse a uno. ✓ - Se muestran los grupos correctos a los que pertenece, pero se indica que no hay sesiones pendientes. ✓ - La página de inicio muestra la sesión asignada y permite acceder a ella. ✓
Realizar una sesión	
Visualización de la sesión asignada. - Paciente completa una actividad. - Paciente completa toda la sesión.	- Al acceder a la sesión asignada al paciente, se muestran los diferentes ejercicios a realizar que forman parte de esa sesión. Se ve el título y el tipo de juego. ✓ - Al seleccionar un ejercicio, se abre una ventana que el paciente puede aceptar y realizar el ejercicio. ✓ - Al completar un ejercicio, este queda marcado en el resumen de los ejercicios de la sesión y no permite volver a realizarlo. Además, se marca como ejercicio finalizado en la base de datos. ✓ - Una vez la sesión está completada (con todos los ejercicios finalizados), se muestra un mensaje de felicitación y se cambia el campo de sesión finalizada a *true*. ✓
Realizar actividades	
Realizar una sopa de letras. - Interacción del paciente con la sopa de letras.	- El juego se carga correctamente, permite al paciente interactuar y reconocer palabras correctamente. ✓ - Todas las palabras mostradas para buscar se encuentran en la sopa. ✓ - Las palabras se corresponden con las orientaciones especificadas. ✓ - Al seleccionar una palabra incorrecta, se marca en rojo y desaparece al cabo de un instante. ✓ - Al seleccionar una palabra correcta, se marca de color verde de forma permanente. ✓ - Después de validar una palabra, se deseleccionan todas las teclas. ✓ - En ocasiones no se genera una palabra en diagonal porque es muy complicado que no coincida con otra palabra y el programa genera otras orientaciones. ✗

Realizar la actividad de la palabra contraria. - Interacción del paciente con la palabra contraria	- Se presenta una palabra y el paciente introduce su contraria por teclado; el sistema valida la respuesta correctamente (omitiendo espacios e ignorando mayúsculas y minúsculas). ✓ - El botón de cancelar borra el texto escrito. ✓ - El botón de enviar valida la palabra y muestra un mensaje de error o de éxito. ✓
Realizar la actividad de clasificación de palabras - Interacción del paciente con la clasificación de palabras	- Las palabras se presentan y el paciente puede clasificarlas correctamente en las categorías dadas. ✓ - No se pueden marcar dos *checkboxs* al mismo tiempo para una misma palabra. ✓ - Al enviar el ejercicio, se validan correctamente todas las palabras. Se devuelve el número de palabras erróneas (contando las no marcadas también como error). ✓ - Cuando todas son seleccionadas correctamente y se envía, se muestra un mensaje de felicitación. ✓
Almacenamiento del historial	
Generar documento de historial al asignar sesión	- Se crea un nuevo documento de identificador único en el historial del paciente con los ejercicios a realizar. ✓
Inicio de un ejercicio	- Cuando el usuario inicia el ejercicio, se guarda el tiempo de inicio. ✓
Fin de un ejercicio	- Cuando el paciente finaliza el ejercicio, se guarda el tiempo de finalización. ✓
Realización de actividades	- Se anota cada intento del paciente al enviar el resultado del juego. Se anotan el número de fallos junto con el momento en que se ha producido. ✓
Administrar los grupos asignados	
Mismas pruebas y resultados que en el caso de los terapeutas. ✓	
Editar la cuenta	
Mismas pruebas y resultados que en el caso de los terapeutas. ✓	
Cerrar sesión	
Cerrar sesión de la cuenta.	- La sesión de la cuenta se cierra correctamente. ✓

Tabla 37. Secuencia de tests de pacientes especificando la prueba y el resultado obtenido

7.2 Test no funcionales

Una vez se han testeado las funcionalidades de la aplicación, en este apartado se analizan los aspectos relacionados con los requisitos no funcionales. Sin embargo, hay que tener en cuenta que no es un versión final y, por lo tanto, los tests deberán volver a ser realizados en la próxima iteración.

7.2.1 Test de memoria

En los requisitos de memoria se estableció que el tamaño del APK debía ser inferior a 150 MB para poder ser subida a Google Play. Sin embargo, se prefería mantener este tamaño al mínimo. En este caso, se ha logrado mantener el tamaño del APK por debajo de los 20 MB; concretamente, el APK tiene un tamaño de 15,60 MB. Este tamaño es bastante óptimo, considerando el rango de funcionalidades de la aplicación, y deja margen para añadir nuevas actividades en el futuro. Además, asegura que la aplicación puede ser descargada e instalada rápidamente por los usuarios.

Después de la instalación, la aplicación ocupa un total de 19,19 MB en el dispositivo. Este tamaño está desglosado entre el propio APK, los datos generados por la aplicación y la caché. El consumo de datos es de 389 kB y el de caché es de 3,19 MB. Estos valores son razonables y no deberían suponer un problema para la mayoría de los dispositivos modernos, incluso para aquellos con capacidad de almacenamiento más limitado. Así se asegura una buena experiencia de usuario en términos de descarga e instalación y rendimiento en el dispositivo.

7.2.2 Test de rendimiento

El uso de la aplicación es correcto, los tiempos de espera no son elevados y no interfieren con la experiencia de usuario. Actualmente, el número de usuarios y datos testeados no es muy elevada, pero se espera que pueda tener mayor escalabilidad en el futuro.

Sin embargo, para la versión final de la aplicación deberían realizarse tests de rendimientos más específicos para asegurar una buena calidad.

7.2.3 Test de usabilidad

La aplicación ha implementado los *widgets* de *Material Design* y ha seguido las directrices de usabilidad, tal y como se ha visto reflejado en las decisiones tomadas durante el desarrollo. Además, se ha utilizado un fichero “styles.xml” que permite modificar el estilo de los textos de forma rápida y centralizada (por ejemplo, aumentando el tamaño de la letra en toda la aplicación si esto fuera necesario para mejorar la usabilidad).

Por otra parte, las pruebas de usuario arrojarán resultados sobre la facilidad de uso de la aplicación.

7.2.4 Test de fiabilidad

Respecto a la recuperación de fallos, se han simulado diferentes escenarios de fallo, como pérdida de conexión a Internet y cierres inesperados de la aplicación. En todos estos escenarios, la aplicación ha sido capaz de recuperarse sin pérdida de datos, lo que demuestra su robustez.

Por otra parte, la aplicación ha sido probada en diversos dispositivos al mismo tiempo, realizando peticiones continuas para evaluar la disponibilidad. A pesar del aumento de tráfico, la aplicación ha mantenido un funcionamiento continuo y sin interrupciones. Se prevé que el número de usuarios que utilicen al mismo tiempo la aplicación no sea muy

elevado, sin embargo, deberían hacerse test más exhaustivos para un posible escalado en el futuro.

Todos los usuarios de la aplicación utilizan un correo y una contraseña, que aseguran que ningún usuario no registrado pueda acceder a la aplicación. El correo electrónico y la contraseña están asegurados por el *EU General Data Protection Regulation* (GDPR[13]) [12] del servicio de privacidad y seguridad proporcionado por Firebase [13]. Además, los administradores y terapeutas deben ser creados previamente por un administrador, lo que asegura la protección de datos de los pacientes y el resto de los usuarios.

7.2.5 Test de portabilidad

La aplicación ha sido desarrollada utilizando el SDK de Android versión 34 como referencia, que representa la versión más reciente disponible en el momento de desarrollo. Es compatible con dispositivos que ejecutan Android 7.0 (Nougat) o versiones posteriores, hasta la versión Android 14 correspondiente al SDK 34.

Se llevó a cabo una serie de pruebas de instalación en una variedad de dispositivos Android, y en la mayoría de ellos, la aplicación ha demostrado un rendimiento óptimo. Sin embargo, hemos identificado ciertos problemas en dispositivos con pantallas más pequeñas. Específicamente, algunos elementos de diseño no se adaptan adecuadamente a estos tamaños, impidiendo la visualización total de ciertas páginas o secciones.

Por ejemplo, en el dispositivo Huawei P30 Lite, que cuenta con un tamaño de pantalla de 6,15 pulgadas, se ha detectado una incompatibilidad. Al intentar jugar a la sopa de letras, es necesario realizar desplazamientos en la pantalla (*scroll*) para poder visualizar la totalidad de las letras. De igual manera, en actividades como “AddGroupActivity” el último botón queda casi fuera de la pantalla. Por lo que se debería mejorar el diseño para pantallas más pequeñas.

Por ahora, los dispositivos que utiliza la asociación AFATE son unas tabletas con pantallas de gran tamaño, por lo que no deberían presentar problemas. Sin embargo, de cara al futuro se deberían solucionar estos errores para mejorar su portabilidad.

13 GDRP = General Data Protection Regulation

7.3 Pruebas de usuario y *feedback* obtenido

La interacción con usuarios reales es esencial para afinar cualquier aplicación. Para esta primera versión de la aplicación, los terapeutas y psicólogos de AFATE en Tortosa fueron esenciales, ofreciendo retroalimentación directa para optimizar la experiencia del usuario. También ayudaron a generar mejoras tanto para la utilización como terapeuta o administrador, así como la posible facilidad de uso de los pacientes.

Las pruebas con usuarios reales constituyen una herramienta indispensable en el proceso de desarrollo. Permiten obtener una visión más clara sobre cómo los usuarios interactúan con la aplicación en contextos reales y captar opiniones directas respecto a su experiencia de uso. Especialmente, en este tipo de aplicación que responde a las necesidades de una asociación concreta. Al haber desarrollado una primera versión del proyecto, se ha dado prioridad a recibir opiniones de profesionales como terapeutas y psicólogos de AFATE. El *feedback* de este grupo ha sido crucial para identificar áreas de mejora y ajustar la aplicación, garantizando su eficacia para el usuario final: los pacientes.

7.3.1 Objetivos

El propósito principal de estas pruebas es evaluar la intuitividad y accesibilidad de la aplicación. Se pretende detectar fallos de usabilidad, ambigüedades en las interfaces y funcionalidades que puedan resultar confusas para los usuarios. Dado que la aplicación será utilizada por un público diverso, con diferentes edades, posibles problemas de visión y variados niveles de destreza tecnológica, es esencial garantizar su fácil manejo.

7.3.2 Metodología

Se realizó una sesión de presentación en las oficinas de AFATE en Tortosa. En ella pude presentar mi aplicación al director de la asociación y al equipo de terapeutas y psicólogos que la componen. Expliqué cómo se había planteado el desarrollo del proyecto a raíz de las reuniones previas y los requerimientos que se habían alcanzado. Después, pudieron proponer nuevas funcionalidades o modificaciones. A continuación, se realizó una pequeña prueba donde ellos mismo pudieron probar a realizar un uso general que sería el que utilizarían a diario. También evaluaron la usabilidad desde el punto de vista del paciente, conociendo las limitaciones y necesidades de este tipo de usuario.

7.3.3 Resultados y observaciones

Las pruebas arrojaron resultados mayormente positivos, donde se examinaron todos los roles disponibles en la aplicación. La organización por grupos, que permite a los terapeutas visualizar y asignar sesiones únicamente a pacientes de sus grupos asignados, fue un punto muy positivo debido a que, actualmente, en la aplicación comercial de la que disponen se encuentran todos los pacientes registrados en una única vista, generando dificultades y conflictos para los terapeutas al asignar sesiones. También se consideró de vital utilidad el almacenamiento del historial para poder monitorear la evolución y desarrollo de los pacientes a lo largo del tiempo.

No obstante, surgieron áreas de mejora, especialmente en la interacción del terapeuta con la selección de actividades para formar sesiones, la experiencia de pacientes al interactuar con ciertos juegos y la gestión de inicio de sesión de los pacientes. A continuación, se recoge de forma detallada estos aspectos a mejorar en las interfaces:

- Algunos **pacientes tienen dificultades para utilizar tecnologías** y necesitan ayuda para realizar diversas tareas. Por esta razón, el proceso de registro y posterior inicio de sesión con un correo y contraseña únicos podría dificultar

su uso y el correcto desarrollo de las sesiones. Actualmente, la asociación utiliza una aplicación comercial que tiene una cuenta única de administrador/terapeuta. Esta cuenta se tiene iniciada en todas las tabletas y permite ver a los pacientes, y mediante la selección de uno de ellos, se inicia su sesión de forma directa. De esta forma la visualización en el dispositivo es la misma que en el caso de mi aplicación al iniciar sesión un paciente con sus credenciales. Este acceso directo les facilita mucho el trabajo a los terapeutas a la hora de trabajar con grupos más numerosos y sin necesidad de almacenar numerosas cuentas y contraseñas. Además, permite que el paciente pueda realizar su sesión de actividades nada más obtener la tableta. Sin embargo, la opción actual de registrar su propia cuenta debería mantenerse, ya que hay personas que sí serían capaces de hacerlo por sí mismas y esto permitiría el acceso a la aplicación a personas con movilidad reducida que no pueden salir de su domicilio o que tienen dificultades para llegar al lugar donde se realizan las sesiones.

- **Diversificar las opciones de registro** en el caso de los pacientes, de manera que no se limite únicamente al uso de correos Gmail, debido a que muchos pueden no tener este tipo de cuentas y haría más tedioso el trabajo de los terapeutas.
- Respecto a las interfaces que ven los pacientes, añadir una opción de **tamaño de letra** que permita aumentarla, ya que existen pacientes con visión limitada.
- **Crear sesiones se dificulta** debido a que la vista de actividades actualmente tiene agrupados todos los diferentes tipos de juegos y no permite la búsqueda de algún juego determinado. Además, todos los diferentes tipos de juegos no tienen ninguna diferencia visual rápida como podría ser un icono diferencial o la tarjeta de diferente color.
- Se propone realizar **interfaces más interactivas** para captar más la atención del paciente.
- **Mejorar el transcurso de la sesión** para que sea más continuado.
- También se proponen mejoras para el **historial de los pacientes**, como apartados de la interfaz genéricos, con gráficas para poder ver de forma clara y visual la evolución de los pacientes.
- **Generar otra versión del juego de palabra contraria**. La versión actual se mantendría, pero se necesitaría otra más sencilla. Algunos pacientes tienen dificultades para escribir o encontrar las letras en el teclado de una tableta, por lo tanto, se necesitaría eliminar esta escritura y sustituirlo por otras opciones más sencillas y visuales.

7.3.4 Recomendaciones

En base a los resultados y observaciones obtenidos, se proponen las siguientes posibles mejoras en la aplicación actual para favorecer la accesibilidad y usabilidad de la herramienta:

- **Adaptabilidad del texto**: Implementar una función que permita a los usuarios ajustar el tamaño del texto según sus necesidades visuales.
- **Sesión directa para pacientes**: Permitir que los terapeutas, tras iniciar sesión, puedan seleccionar un paciente y emular su inicio de sesión (sin necesidad de introducir sus credenciales), facilitando el proceso de realización de las sesiones en los grupos presenciales de la asociación.

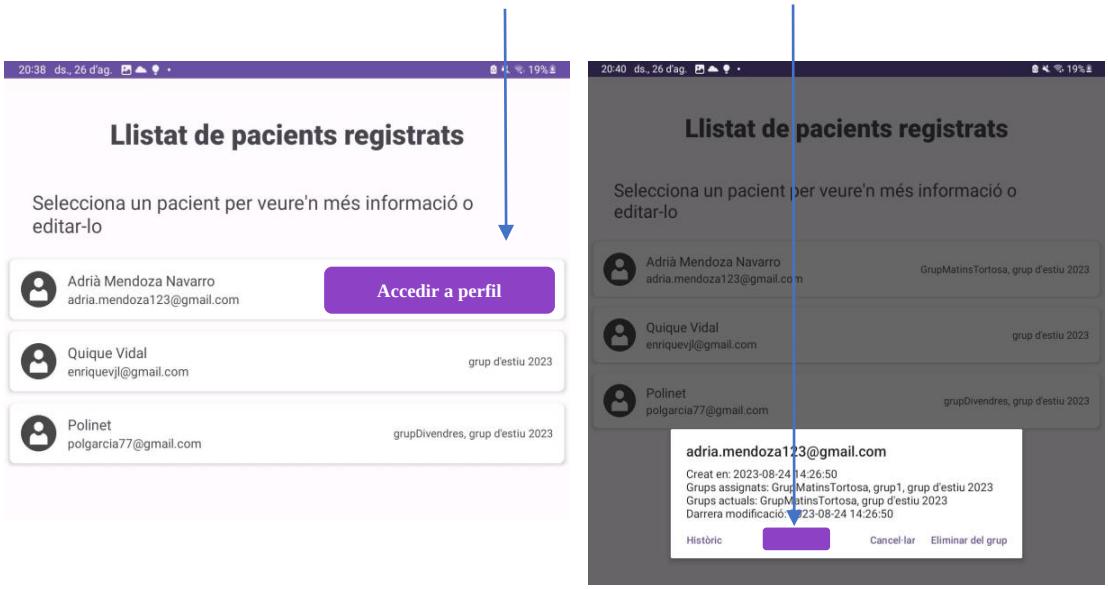


Figura 53. Posibles lugares donde añadir el botón de acceso a la nueva funcionalidad

- **Diversificación de opciones de registro para pacientes**: Es viable considerar registros mediante códigos u otros mecanismos que los propios terapeutas puedan dar al paciente y anotar en la base de datos.
- **Mejorar la organización visual y búsqueda para terapeutas y administradores**: Mejorar la organización visual y búsqueda en la vista de actividades, incorporando distintivos como iconos o colores. Así como posibles buscadores o filtros en todos los tipos de vistas (sesiones, actividades, pacientes, terapeutas).
- **Enriquecimiento visual para los pacientes**: Diversificar las actividades, integrando más elementos visuales como símbolos y dibujos, para hacer la experiencia más lúdica y atractiva. Además, incorporar más mensajes motivacionales y más claros, que fomenten la participación activa del paciente. También se recomienda el uso de mayúsculas en todos los juegos.
- **Realización de la sesión continua**: En lugar de que los pacientes deban entrar a una nueva actividad, permitir que una vez finalizan una actividad, se pase directamente a la siguiente sin tener que volver a la vista de sesión. También establecer unos parámetros de número de fallos máximos o tiempo máximo en el ejercicio, para darlo por finalizado, pero sin completar. De esta forma, pueden seguir con la sesión evitando la frustración.
- **Extensión del historial de los pacientes**: En relación con el historial de pacientes, se sugiere diseñar un sistema de puntos en función del tiempo transcurrido en los ejercicios y el número de fallos. Una línea de tiempo con puntuaciones permitiría monitorear el progreso del paciente y observar las fluctuaciones de nivel a lo largo del tiempo. También se podría añadir una gráfica simple sobre en qué tipos de juegos son los que el paciente obtiene mejores y peores resultados. Esta funcionalidad facilitaría el análisis rápido de los juegos en los cuales el paciente destaca o presenta dificultades.
- **Nueva versión del juego de la palabra contraria**: Para eliminar el uso del teclado en la actividad podrían mostrarse varias opciones directamente y que el usuario deba hacer clic sobre la palabra correcta, es decir, la palabra contraria a la que ve en el ejercicio.

Estas recomendaciones buscan, no solo corregir las áreas identificadas durante las pruebas, sino también mejorar la experiencia general del usuario, haciendo la aplicación más intuitiva, accesible y atractiva para todos los involucrados.

7.3.5 Propuestas de nuevos juegos

Durante la reunión, tuvimos la oportunidad de evaluar los juegos actuales en papel que se llevan a cabo en las sesiones, así como algunas características de otra aplicación comercial relacionada. A partir de esta revisión, se diseñaron diversas propuestas para nuevos juegos que se sugiere integrar en la próxima versión del proyecto. A continuación, se presenta un resumen de estas recomendaciones:

- **Juegos con componentes dinámicas**: Se propone la adición de juegos que contengan elementos dinámicos, como el de las burbujas móviles. Este juego consiste en ordenar las sílabas para formar una palabra, añadiendo la peculiaridad de que cada sílaba se encuentra en una burbuja y estas burbujas pueden estar estáticas (nivel más sencillo) o pueden estar en movimiento (nivel más complejo).
- **Juegos de memoria a corto plazo**: Se podría desarrollar un juego donde se exhiban palabras en la pantalla por un breve período, las cuales desaparecerán para posteriormente mostrar un listado más extenso. Los pacientes deberán identificar las palabras que recuerdan.
- **Juegos con imágenes**: La idea es permitir cargar imágenes con uno o varios elementos gráficos, proporcionando opciones para que los pacientes las identifiquen. Por ejemplo, imágenes de relojes marcando diferentes horas y que digan cuál es cuál o dibujos de animales para clasificar.
- **Juegos predefinidos recurrentes**: Se propone tener juegos estándar, como uno que muestre un dibujo del cuerpo humano con palabras desordenadas. Los pacientes tendrán que arrastrar cada término a la parte correspondiente del cuerpo. En cada repetición, se mostrarían palabras diferentes aleatoriamente.
- **Juegos de orientación**: Se visualizarían flechas en pantalla, algunas apuntando hacia la izquierda y otras hacia la derecha. Los pacientes seleccionarían las que indiquen una dirección previamente especificada.
- **Juegos de cálculos**: Se sugiere la inclusión de ejercicios matemáticos simples, como sumas y restas, indicando visualmente la posición donde el paciente debe efectuar el cálculo y seleccionar la respuesta entre opciones mostradas en pantalla.
- **Juegos de descripción**: Por ejemplo, comenzar una frase con “El panadero hace…” y ofrecer múltiples opciones para que los pacientes elijan la continuación adecuada.
- **Juegos de estimación de tiempos**: Se presentaría una acción, como “Lavarse los dientes”, y se proporcionarían distintas opciones de duración (ej. “5 segundos / Varios minutos / Muchas horas”) para que los pacientes seleccionen la más apropiada.

8 Instalación y mantenimiento

El proceso de instalación, despliegue y mantenimiento es un aspecto esencial en la vida de cualquier aplicación. No solo es necesario instalar y desplegar correctamente la solución, sino también asegurarse de que se mantenga actualizada y funcional con el tiempo. Dado que la intención es que este TFG sea continuado por otro desarrollador, es esencial proporcionar información detallada para garantizar una transición sin problemas y la continuidad del proyecto.

8.1 Preparación y configuración del entorno de desarrollo

Antes de empezar la instalación, es crucial preparar y configurar adecuadamente el entorno de desarrollo. Esto garantiza un proceso de desarrollo fluido y sin complicaciones.

- **Sistema Operativo**: Aunque el desarrollo puede llevarse a cabo en cualquier sistema operativo (Linux, MacOS o Windows), en este proyecto se ha utilizado Windows.
- **Java Development Kit (JDK[14]):** Es necesario tener instalado el JDK en su versión más reciente. Esto es esencial para desarrollar aplicaciones Android.
- **Android Studio**: Es necesario descargar e instalar la última versión de Android Studio, el IDE oficial para desarrollo Android. Durante la instalación, se recomienda instalar todas las actualizaciones y componentes sugeridos.
- **Emulador Android**: A través de Android Studio, es posible configurar un emulador Android. En este proyecto, se utilizó un emulador de una Tablet Samsung Galaxy Tab S6 Lite con One UI versión 5.1.
- **Kotlin**: El lenguaje de programación es Kotlin, versión 1.9.0.
- **Gradle plugins**:
  - **Plugin Android**: 7.4.2
  - **Plugin Kotlin Android**: 1.9.0
  - **Google Services Plugin**: 4.3.15

8.2 Proceso de instalación

Una vez configurado el entorno de desarrollo, se debe instalar el proyecto y sus dependencias. El código fuente completo se encuentra en GitHub, facilitando así el acceso al próximo desarrollador.

8.2.1 Instalación de Firebase

Para la integración de Firebase:

1. **Acceso a Firebase**: Iniciar sesión en la consola de Firebase utilizando una cuenta de Google.
2. **Acceso al proyecto existente**: Dado que el proyecto de Firebase ya existe, se compartirá el acceso al repositorio y el nuevo desarrollador podrá acceder a él con permisos de propietario.
3. **Configuración para Android**: Es esencial que el archivo "google-services.json" esté adecuadamente ubicado en la carpeta "app" del proyecto

[14] JDK = Java Develpment Kit

Android. En caso de requerir cambios, puede ser descargado nuevamente desde la consola de Firebase.

Desde Firebase se puede acceder a los siguientes servicios utilizados actualmente en la aplicación:

- **Authentication**: Esta sección gestiona la autenticación de los usuarios. Todos los usuarios registrados se encuentran aquí, identificados por sus correos electrónicos. Es importante resaltar que las contraseñas están encriptadas y no pueden ser vistas ni modificadas directamente desde este panel. Para garantizar la seguridad del usuario, en caso de necesidad de cambio de contraseña, se puede enviar un correo al usuario correspondiente con un enlace seguro para que la actualice. Para realizar este proceso, se puede hacer clic en "ver más" (⋮) al lado del usuario deseado y seleccionar la opción "reset de contraseña".
- **Firestore Database**: Esta base de datos almacena todos los datos asociados a la aplicación, estructurados en colecciones y documentos. La organización jerárquica permite una gestión y consulta eficiente de la información. Aquí se puede acceder a datos específicos de usuarios, interacciones, configuraciones y cualquier otro elemento que la aplicación necesite almacenar para su funcionamiento óptimo.
- **Firestore Storage**: Sección que gestiona el almacenamiento de archivos como fotografías o vídeos, pero que no se ha utilizado finalmente en esta aplicación.
- **Analytics Dashboard**: En este tablero, se tiene una visión panorámica de la actividad y comportamiento de los usuarios dentro de la aplicación. Se puede visualizar información sobre la cantidad de usuarios activos, el tiempo medio que pasan en la aplicación, las páginas o secciones más visitadas y otros indicadores clave de rendimiento (KPIs). Esta información es vital para entender el uso de la aplicación y para tomar decisiones basadas en datos reales, sobre futuras mejoras o cambios.
- **Analytics Events**: Esta sección recoge eventos específicos que suceden dentro de la aplicación, tales como clics en botones, envíos de formularios, visualizaciones de páginas o interacciones con ciertas funcionalidades. Estos eventos se pueden configurar de acuerdo con las necesidades de la aplicación, para rastrear acciones específicas que son de interés. Actualmente, se registran eventos como accesos al inicio de sesión o registros de nuevas cuentas.

8.2.2 Instalación del proyecto Android

1. **Clonar el Repositorio**: Requiere la instalación previa de Git. A continuación, se debe clonar el repositorio en la máquina local.
2. **Abrir con Android Studio**: Una vez clonado, abrir Android Studio y seleccionar "Open an existing Android Studio project", y elegir el directorio del proyecto.
3. **Sincronización y Dependencias**: Una vez abierto el proyecto, permitir que Android Studio sincronice el proyecto y descargue todas las dependencias necesarias.
4. **Ejecución**: Una vez sincronizado, se puede ejecutar la aplicación en un emulador o en un dispositivo Android conectado.

8.3 Mantenimiento y actualizaciones

El mantenimiento es esencial para garantizar que la aplicación sigue funcionando correctamente con el tiempo, especialmente cuando hay cambios en las dependencias o en los sistemas operativos de los dispositivos.

8.3.1 Mantenimiento de Firebase

Monitorización: La consola de Firebase puede ser utilizada para monitorizar la actividad de la aplicación, errores y problemas potenciales.

Actualizaciones de SDK: Firebase lanza regularmente actualizaciones de su SDK. Es importante mantener el SDK de Firebase en el proyecto actualizado a la última versión para beneficiarse de correcciones de errores y nuevas funcionalidades.

Seguridad: También se debe revisar regularmente las reglas de seguridad de Firestore y la autenticación para garantizar que los datos de los usuarios estén protegidos.

8.3.2 Modificaciones y actualizaciones del código

Comprender a fondo los casos de uso, parámetros de clases y estructura de la base de datos es vital antes de hacer modificaciones. Un cambio erróneo podría comprometer la aplicación. Se recomienda realizar actualizaciones incrementales, subiéndolas a GitHub, para poder revertir cambios si es necesario.

8.4 Presupuesto y costes adicionales

Este presupuesto se ha enfocado en los costes asociados con el desarrollo, operación y hardware del proyecto.

Costes de desarrollo: Al ser parte de un Trabajo de Final de Grado no hay presupuesto destinado al desarrollo del software (sin coste): 0 €. Sin embargo, es importante señalar que, si se hubiera subcontratado a un programador profesional, posiblemente hubiera tardado varias semanas en realizarla y el coste podría rondar los 2.000 €.

Costes de mantenimiento: Firebase, para las características actuales de la aplicación, tiene un coste gratuito (sin coste): 0 €. En el caso de que se quisiera aumentar mucho el número de usuarios o la capacidad de almacenamiento, se deberían revisar las tarifas actuales de Google.

Costes operativos: Electricidad (estimación para el uso del Ordenador y la Tablet): 10 €/mes x 12 meses = 120 €

Costes de hardware: Tablets para los pacientes. Actualmente ya disponen de tablets para hacer algunos ejercicios, pero tal vez se debería aumentar el número si piensan que su principal método de sesiones sea la aplicación. 200 € x 5 tablets = 1.000 €.

Costes de formación: Tiempo empleado en la formación de los terapeutas a utilizar la aplicación (sin coste): 0 €. Aunque no hay un coste económico directo, es esencial tener en cuenta el tiempo y esfuerzo que AFATE tendrá que invertir en formar a sus terapeutas para usar la aplicación. Esto implica organizar sesiones de formación, realizar pruebas, integrar a los pacientes, grupos y actividades, lo que sin duda supone un compromiso y esfuerzo significativo por parte de la asociación.

Contingencia: Se recomienda tener un presupuesto de contingencia para cubrir gastos no previstos o emergentes. Calculando el 10% del costo total (sin contar el desarrollo, ya que no fue una inversión real), serían: 10% x (120 € + 1.000 €) = 112 €.

Total: 120 € (operativos) + 1000 € (hardware) + 112 € (contingencia) = 1.232 €

9 Propuestas de futuro

En este apartado, se discute la evaluación final del proyecto y las posibles propuestas de ampliaciones futuras.

9.1 Evaluación final

La creación de una aplicación específicamente diseñada para satisfacer las necesidades de AFATE en Tortosa representa un paso significativo hacia la digitalización de herramientas terapéuticas en el ámbito de la salud mental. La interacción directa con profesionales y pacientes ha permitido moldear un producto que no sólo funcione técnicamente, sino que también atienda de manera efectiva las expectativas y requerimientos de los usuarios finales. Durante el proceso de evaluación, ha quedado patente la importancia de la usabilidad y la accesibilidad, especialmente al tratar con un público con diversos niveles de destreza tecnológica y necesidades específicas. Si bien la aplicación ha recibido una respuesta en su mayoría positiva, la retroalimentación obtenida ha destacado áreas de mejora, en particular relacionadas con la experiencia del usuario y la intuitividad de la interfaz. Cabe destacar que los juegos de pruebas han mostrado resultados exitosos, y que la aplicación cumple adecuadamente con todas las funcionalidades actuales.

Los resultados y recomendaciones recabados son esenciales para garantizar que la aplicación continúe evolucionando en función de las necesidades reales de sus usuarios.

9.2 Potenciales ampliaciones futuras

Como se mencionó anteriormente, hay un conjunto claro de mejoras y ampliaciones que podrían enriquecer aún más la aplicación:

- **Nuevos juegos y funcionalidades**: Basándonos en las sugerencias directas de profesionales y pacientes, la incorporación de nuevos juegos terapéuticos ofrecerá una diversidad más amplia de herramientas. Estas adiciones buscan no solo variar la oferta terapéutica sino potenciar mejores resultados clínicos.
- **Revisión estética**: Es vital mejorar el diseño actual de la aplicación, integrando elementos visuales más atractivos. Estas modificaciones no son solamente estéticas; buscan mejorar la experiencia del usuario y mantener su interés activo, lo que es esencial, en particular, para los pacientes.
- **Mejoras en la interfaz y mensajes**: El refinamiento de los mensajes de error y éxito es fundamental. Al hacer estos mensajes más comprensibles y visibles, se mejorará la interacción con la aplicación y se reducirán posibles frustraciones y malentendidos.
- **Creación de manuales**: Se sugiere el diseño de un manual de usuario específico para los diferentes tipos de usuarios. Esta herramienta facilitará la comprensión y uso óptimo de la aplicación, maximizando su potencial.
- **Optimización de vistas**: Es necesario mejorar y clarificar las vistas de elementos, añadiendo opciones para ordenarlos o filtrarlos según criterios definidos por el usuario.
- **Mejoras iconográficas**: La revisión y mejora de los iconos utilizados en la aplicación contribuirán a una interpretación más intuitiva y una navegación más fluida.
- **Expansión de métodos de registro**: La inclusión de nuevas formas de registro, como el uso de códigos, puede agilizar el acceso a la aplicación.

- **Notificaciones de errores**: Sería beneficioso que, en caso de error, se envíen notificaciones automáticas a terapeutas o administradores. Utilizando funciones específicas de Firebase, este proceso puede ser automatizado y eficiente.
- **Estadísticas para administradores**: La implementación de un panel de estadísticas permitiría al administrador visualizar datos relevantes, como el número de cuentas creadas, sesiones realizadas o el acceso de terapeutas en un periodo determinado. Mediante el uso de eventos de Firebase, esta funcionalidad puede ser llevada a cabo.
- **Almacenamiento de fotografías**: Se propone la utilización del almacenamiento de Firebase para fotos (*Firebase Storage*), para ofrecer nuevas actividades multimedia a la asociación.
- **Realizar test funcionales con los propios usuarios**: Una vez la aplicación llegue a una versión final, será clave hacer pruebas directamente con los usuarios para la evaluación.
- **Realizar tests no funcionales más exhaustivos**: Antes de que la aplicación sea operativa definitivamente, debe cumplir unos mínimos de calidad que se deben verificar adecuadamente.
- **Mejorar la adaptabilidad de interfaces**: Es fundamental garantizar que la aplicación ofrezca una experiencia de usuario uniforme y óptima en todos los dispositivos. A partir de las pruebas realizadas, se detectaron problemas de adaptabilidad en ciertos dispositivos con pantallas de dimensiones específicas. Para ello, se propone una revisión exhaustiva y mejora de las interfaces para que se adapten de manera eficiente a distintos tamaños y resoluciones de pantalla.

10 Conclusiones

El TFG en Ingeniería Informática, en mi caso, se ha presentado no solo como una oportunidad para demostrar las habilidades y conocimientos adquiridos durante los años de formación, sino también para aprender a desarrollar de forma autónoma un proyecto que combina aspectos técnicos y humanos. La ejecución de este TFG ha aportado de manera invaluable a mi formación como futura ingeniera informática en múltiples dimensiones.

Primero, y ante todo, la interacción directa con profesionales terapeutas y pacientes me ofreció un entendimiento profundo sobre la complejidad de la toma de requisitos. Fue crucial aprender a escuchar activamente, interpretar y traducir las necesidades reales en requisitos funcionales y no funcionales, garantizando así que la aplicación sea efectiva y adecuada.

En cuanto al ámbito técnico, la utilización de herramientas y lenguajes como Kotlin en Android Studio me permitió iniciar y profundizar mi competencia en el desarrollo de aplicaciones móviles. Además, trabajando con bases de datos NoSQL, en particular Firebase Firestorage, tuve la oportunidad de aprender sobre las modernas soluciones de almacenamiento de datos, comprendiendo sus ventajas y cómo aplicarlas en contextos reales.

La implementación de *Material Design 3* no solo me permitió crear una interfaz agradable y funcional, sino que también subrayó la importancia de seguir directrices y patrones de diseño reconocidos para garantizar una experiencia de usuario óptima.

El autoaprendizaje fue una constante. A pesar de la base sólida obtenida durante estos años de estudio en la URV, enfrentar un proyecto real trajo consigo desafíos inesperados que exigieron investigación, persistencia y una continua actitud de aprendizaje. Llegar a equilibrar soluciones prácticas y reales con las expectativas iniciales generadas, fue también todo un reto.

Esta experiencia me ha brindado un entendimiento más profundo de lo que significa ser Ingeniera Informática. Más allá de escribir código o diseñar sistemas, se trata de conectar tecnología con personas, de comprender y resolver problemas y, sobre todo, de crear soluciones que aporten valor al mundo real y a la sociedad. Con la finalización de este TFG, siento que no solo he logrado un gran proyecto, sino que también he fortalecido mi identidad y mis habilidades como profesional en el campo de la Ingeniería Informática.

Finalmente, me gustaría expresar mi más sincero agradecimiento. En primer lugar, a mi tutor, Pere Millán Marco, por guiarme en este proyecto y por su gran predisposición. A la Universidad Rovira i Virgili y a su equipo de profesorado, por ofrecerme una formación integral y por ser el espacio donde he forjado las bases de lo que soy como profesional hoy en día. Y, por supuesto, a mis amigos y familiares que, con su apoyo constante, me han acompañado y animado en este complicado, aunque sumamente interesante, camino.

Referencias

[1] «BOE-A-2018-16673 Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.» https://www.boe.es/buscar/doc.php?id=BOE-A-2018-16673 (accedido 10 de agosto de 2023).

[2] «BOE.es - DOUE-L-2016-80807 Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento general de protección de datos).» https://www.boe.es/buscar/doc.php?id=DOUE-L-2016-80807 (accedido 10 de agosto de 2023).

[3] «BOE-A-2013-12632 Real Decreto Legislativo 1/2013, de 29 de noviembre, por el que se aprueba el Texto Refundido de la Ley General de derechos de las personas con discapacidad y de su inclusión social.» https://boe.es/buscar/act.php?id=BOE-A-2013-12632 (accedido 10 de agosto de 2023).

[4] «BOE-A-2002-13758 Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico.» https://www.boe.es/buscar/act.php?id=BOE-A-2002-13758 (accedido 10 de agosto de 2023).

[5] «BOE-A-1996-8930 Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia.» https://boe.es/buscar/act.php?id=BOE-A-1996-8930 (accedido 10 de agosto de 2023).

[6] «Model-View-ViewModel | Microsoft Learn». https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm (accedido 2 de junio de 2023).

[7] «Aprende el lenguaje de programación Kotlin | Android Developers». https://developer.android.com/kotlin/learn?hl=es-419 (accedido 21 de abril de 2023).

[8] «Kotlin Programming Language». https://kotlinlang.org/ (accedido 10 de junio de 2023).

[9] «Documentación de Firebase». https://firebase.google.com/docs?hl=es-419 (accedido 22 de mayo de 2023).

[10] «Desarrolladores de Android | Android Developers». https://developer.android.com/?hl=es-419 (accedido 12 de junio de 2023).

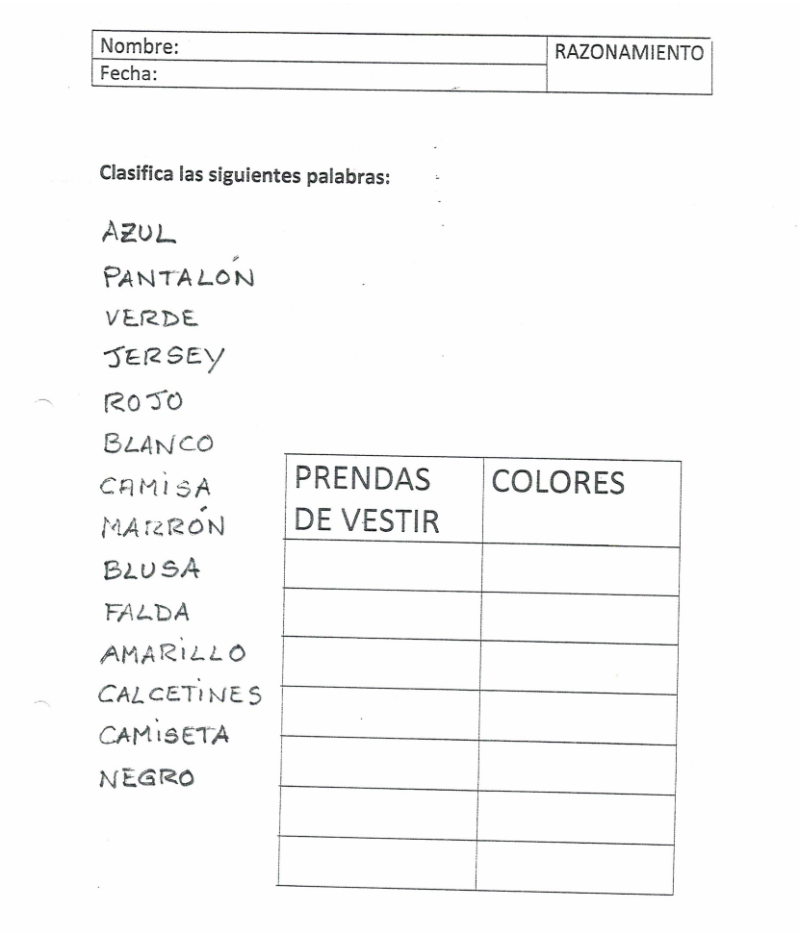
[11] «Material Design». https://m3.material.io/ (accedido 1 de agosto de 2023).

[12] «EUR-Lex - 32016R0679 - EN - EUR-Lex». https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679 (accedido 30 de agosto de 2023).

[13] «Privacidad y seguridad en Firebase». https://firebase.google.com/support/privacy?hl=es-419 (accedido 23 de agosto de 2023).

11 ANEXOS

Anexo 1: Ejemplos de ejercicios a papel que realizan en la asociación AFATE actualmente.



Nombre:	RAZONAMIENTO
Fecha:	

Clasifica las siguientes palabras:

AZUL
PANTALÓN
VERDE
JERSEY
ROJO
BLANCO
CAMISA
MARRÓN
BLUSA
FALDA
AMARILLO
CALCETINES
CAMISETA
NEGRO

PRENDAS DE VESTIR	COLORES

Nombre:	RAZONAMIENTO
Fecha:	

Encuentre la palabra que sobra en cada serie.

→	Pimienta	Orégano	Azúcar	Romero
→	Granito	Rubí	Diamante	Zafiro
→	Lápiz	Plato	Bastón	Aguja
→	Mar	Río	Pantano	Lago
→	Marino	Añil	Turquesa	Rojo
→	Labio	Ceja	Bigote	Pestaña

Nombre:	LENGUAJE
Fecha:	

ESCRIBE LA PALABRA CONTRARIA:

Correcto _____

Imperfecto _____

Barato _____

Delgado _____

Cómico _____

Rubio _____

Delicioso _____

Valiente _____

Moderno _____

Pequeño _____

Ruidoso _____

Generoso _____

Desinteresado _____

Pacífico _____

Nombre:	RAZONAMIENTO
Fecha:	

Instrucciones: escriba para qué sirven los siguientes objetos:

.....

.....

.....

.....

.....

.....

.....

.....