Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN INGENIERÍA DE COMPUTADORES

Aplicación móvil para la detección de dificultades en el aprendizaje de las matemáticas

Estudiante: Manuel del Río González
Dirección: Paula María Castro Castro
Francisco Javier Vázquez Araújo

A Coruña, febrero de 2022.

A mis padres, Jose Fernando y Annabel y a mi hermano Fernando.

Agradecimientos

A mi familia, que me ha apoyado a continuar la carrera en los momentos más difíciles de mi vida, y a mis amigos, que han estado ahí siempre que lo he necesitado.

A toda la comunidad universitaria, que continúa a lo largo del tiempo en el deber atemporal de que el conocimiento no se olvide de una generación a la posterior, tan importante en estos tiempos como en el pasado y futuro. En especial a aquellos profesores, tanto en el colegio, el instituto como en la universidad, que han colaborado en convertirme en el individuo que soy hoy en día, así como a los organismos directores de dichas instituciones.

A mi tutora Paula María Castro Castro y a mi tutor Francisco Javier Vázquez Araújo, por proponer esta herramienta y dedicar toda la atención que yo requería en el desarrollo de la misma.

Resumen

En este documento se describirá el proyecto Blue-Pine Testing. La finalidad del proyecto consiste en la ayuda a la detección de dificultades en el proceso de aprendizaje de las matemáticas mediante una aplicación Android para tablets. Se implementa el test TEDI-MATH, teniendo como objetivo de evaluación a los infantes en el periodo entre segundo de Educación Infantil y tercero de Educación Primaria. De esta forma, los logopedas tienen una herramienta que les permite detectar las dificultades de aprendizaje en las matemáticas en edades tempranas. Adicionalmente, se permite la gestión de pacientes en grupos dirigidos por un logopeda. En consecuencia, esta aplicación permite no sólo la evaluación de dificultades en el proceso de aprendizaje de las matemáticas en grupos de estudiantes, sino también la evaluación de los mismos a lo largo del tiempo.

Abstract

This document will describe the Blue-Pine Testing project. The purpose of the project is to help detect difficulties in the learning process of mathematics through an Android application for tablets. The TEDI-MATH test is implemented, with the objective of evaluating infants in the period between the Second Year of Early Childhood Education and the Third Year of Primary Education. In this way, therapists have a tool that allows them to detect learning difficulties in mathematics at an early age. Additionally, the management of patients in groups led by a therapist is allowed. Consequently, this application allows not only the evaluation of difficulties in the mathematics learning process in groups of students, but also their evaluation over time.

Palabras clave:

- Discalculia
- TEDI-MATH
- Android
- Aprendizaje
- Matemáticas

Keywords:

- Dyscalculia
- TEDI-MATH
- Android
- Learning
- Mathematics

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En esta sección del documento se va a reflejar, en primer lugar, la motivación del proyecto. A continuación, se expresarán los objetivos a cumplir con el desarrollo de esta aplicación. Por último, se va a establecer una estructura para esta memoria donde se expondrán los contenidos de la misma.

1.1 Motivación

Las dificultades del aprendizaje son un conjunto de alteraciones en individuos que afectan a las personas que las sufren, especialmente a los niños, mediante alteraciones en diversas capacidades cognitivas: cálculo, razonamiento matemático, lectura-escritura, etc. Estas circunstancias pueden provocar síntomas con los cuales estamos familiarizados debido, sobre todo, a la dislexia y al conocimiento amplio que existe a día de hoy sobre la misma, gracias, en gran parte, a la gran cantidad de información y documentación que se puede encontrar sobre ella.

Todas las dificultades en el aprendizaje contribuyen en mayor o menor medida a un bajo rendimiento académico y a un aumento de la probabilidad de abandono escolar. Entre las distintas dificultades del aprendizaje hay una, en particular, que llama la atención por el alcance que tiene y a su vez, aunque en aumento, a su baja popularidad. Dicho trastorno es la discalculia o Dificultad en el Aprendizaje de las Matemáticas (DAM). De la misma forma que la dislexia afecta a la parte cognitiva que realiza las funciones de lectura-escritura, la discalculia tiene como origen las funciones cognitivas relativas a la realización de operaciones, tanto aritméticas como lógicas, geometría, numeración, conteo, etc.

La principal preocupación asociada a este trastorno, lo que ha acentuado su estudio en los últimos 50 años [1], es que cada vez el mundo está más centrado en la matemática, en gran medida gracias a la evolución de la informática y la digitalización de muchos aspectos de la vida, y eso hace ver en mayor medida los problemas relacionados con esta área del

conocimiento.

Por tanto, hay una necesidad de tener herramientas que ayuden a la detección de las primeras manifestaciones de este trastorno en el aprendizaje de las matemáticas. En particular, los especialistas precisan herramientas que sean capaces de detectar estos primeros síntomas en etapas tempranas del proceso del aprendizaje, para así poder aplicar medidas correctivas a tiempo, antes de que dichas dificultades puedan afectar al proceso de maduración intelectual del individuo adulto.

1.2 Objetivos

El objetivo de este proyecto es el desarrollo de una aplicación móvil en Android con un sistema que permita detectar dificultades en el aprendizaje de las matemáticas. Concretamente esta aplicación se orienta a niños de educación infantil y de primeros cursos de educación primaria. Para ello, se propone como objetivo la implementación del test TEDI-MATH [2]. Este test se basa en la realización de una serie de pruebas por niveles de edad y por indicadores.

Cabe destacar de este y otros tests similares que no son ejercicios de matemáticas, es decir, no es una forma de evaluar el rendimiento escolar o la inteligencia (por ejemplo, a través de su coeficiente intelectual), sino una evaluación en profundidad de las dificultades en el aprendizaje de las matemáticas.

Por último, la aplicación debe permitir la visualización de resultados de las evaluaciones en un estudiante (o paciente, como se denominará en al aplicación) o en un grupo de estudiantes en un momento dado, así como su evolución temporal. Estos diferentes datos estadísticos indicarán el estado en el momento de la evaluación del estudiante y la exportación de dicha información permitirá el análisis y el tratamiento de forma cómoda por parte de los especialistas y profesionales que atienden al paciente o pacientes de estudio.

Por tanto, se destacan los siguientes objetivos:

- Desarrollo de una aplicación de Android que permita realizar evaluaciones de las dificultades en el aprendizaje de las matemáticas.
- Implementación de un módulo que realice el test de evaluación TEDI-MATH [2].
- Implementación de un sistema de informes que permita ver, según métricas y edad, los resultados comparativos con el resto de alumnos de su entorno.
- Implementación de un sistema de grupos que permita un análisis comparativo de los alumnos y sus informes.
- Implementación de una aplicación con capacidad para visualizar la evolución de los estudiantes a lo largo del tiempo.

- Exportación de dichos informes en ficheros.
- Reducir a cero la cantidad de papel usada para anular el impacto ambiental que conlleva el manejo de evaluaciones e informes en papel, así como el seguimiento de casos.

1.3 Estructura de la memoria

La memoria se dividirá en los siguientes capítulos:

- **Capítulo 1: Introducción.** Capítulo actual, en el que se han descrito brevemente la motivación del proyecto, así como sus objetivos y la estructura de la memoria del Trabajo de Fin de Grado (TFG).
- **Capítulo 2: La discalculia.** En este capítulo se describe en qué consiste esta dificultad en el aprendizaje, cómo y cuándo se manifiesta y de qué forma afecta a los individuos. También se explorarán los distintos métodos de diagnóstico del trastorno y los tratamientos recomendados.
- **Capítulo 3: Estado del arte.** Se hace un estudio de las distintas herramientas que ya existen en el mercado y que realizan funciones similares o complementarias.
- **Capítulo 4: Fundamentos tecnológicos.** Se realiza una descripción de los fundamentos de programación en Android-Java de forma nativa.
- **Capítulo 5: Metodología de trabajo.** Se describe la metodología iterativa-incremental para el desarrollo software de proyectos, así como una explicación de como se implementó y una comparativa con otras metodologías.
- **Capítulo 6: Planificación y costes.** Se explora la planificación del proyecto, el resultado final una vez se haya aplicado el seguimiento, así como el cálculo de costes del mismo.
- **Capítulo 7: Análisis de Requisitos.** Se elaboran los requisitos de la aplicación. Se disciernen los distintos perfiles de acceso a la aplicación así como el establecimiento de una jerarquía de requisitos primarios, secundarios y terciarios.
- **Capítulo 8: Diseño.** Se muestran los diseños de los distintos elementos de la aplicación: estructura de base de datos, arquitectura de la aplicación y comunicación entre los distintos elementos de la misma.
- **Capítulo 9: Implementación.** Se exploran las características más relevantes de proceso de implementación, así como los elementos fundamentales.

- **Capítulo 10: Depuración y pruebas.** Se describen las pruebas realizadas, tanto en una emulación como en un dispositivo real.
- **Capítulo 11: Conclusiones finales.** Se realiza un estudio comparativo del cumplimiento de los objetivos en este capítulo, de los requisitos establecidos en la fase de análisis y se muestran distintas funcionalidades que se podrían añadir o modificar sobre el trabajo ya realizado.

Capítulo 2

La Discalculia

La discalculia o DAM es un trastorno que suele aparecer y desarrollarse en la etapa infantil, como otras dificultades en el aprendizaje. En muchas ocasiones, los profesionales la denominan "la dislexia de los números". Sin embargo, no sólo afecta a la lectura/escritura de números, lo cual sí que lo relacionaría directamente con la dislexia, sino que además afecta a las principales capacidades matemáticas del individuo: cálculo, numeración, relaciones entre los números (mayor que, menor que, igual...), operaciones entre ellos (suma, resta, multiplicación, división...), visión espacial, etc. Por tanto, no es correcto definirlo tan sólo como un problema de lectura/escritura. [3]

Con todo, por otros motivos la discalculia sí puede aparecer con mayor probabilidad conjuntamente con otros trastornos, como la dislexia o el Trastorno de déficit de atención con hiperactividad (TDAH), que en individuos no afectados por la discalculia. [4]

Conviene destacar que esta dificultad del aprendizaje no es un indicador de una inteligencia inferior, igual o mayor que la de un individuo sin ella. Sin embargo, si no se tratara adecuadamente, podría llevar a un bajo rendimiento escolar, así como a problemas de autoestima, emocionales, de comunicación social y, retrasos en el aprendizaje que podrían llevar a un abandono escolar prematuro o mayor probabilidad de tener un perfil criminal. [4]

2.1 Origen

En la actualidad, numerosos estudios estiman que la discalculia está presente entre el 3% y el 6% de la población, y que dicho trastorno persiste hasta la edad adulta. Es importante entender que, en la mayoría de los casos, estas dificultades no se solucionan por sí mismas, sino que requieren de una intervención, que podría subsanarlas o al menos limitar mucho el impacto en la vida diaria del individuo, especialmente si se atiendan en etapas tempranas de desarrollo. [4]

En la mayoría de los casos, la discalculia suele tener un origen genético con un alto factor

de herencia. En concreto, en cuanto a las familias de los niños diagnosticados con DAM, el 40% de los padres compartían el trastorno, y entre el 70% y 50% para gemelos monocigóticos y dicigóticos, respectivamente. [5]

Además, se ha podido observar que entre los niños identificados como potencialmente con discalculia, el 81% previo de algún otro trastorno del neurodesarrollo. [4]

En casos menos frecuentes, la discalculia puede ser adquirida por traumatismos craneales, por ejemplo, por accidentes de tráfico. Estos casos son más raros y preocupantes, ya que el tejido neuronal no puede ser regenerado y, por tanto, el individuo perdería las capacidades asociadas permanentemente.

2.2 Afectación

La discalculia afecta a los individuos de las siguientes formas:

- Dificultad para el reconocimiento de números.
- Lentitud para aprender a contar.
- Ausencia de relación entre los símbolos numéricos (3) con su grafía (tres).
- Necesidad de estímulos visuales para contar: contar con los dedos, señalar los objetos mientras los cuenta…
- Dificultad para realizar operaciones aritméticas básicas: sumas, restas, multiplicaciones, divisiones…
- Problemas para relacionar los números con cantidades de objetos sensibles: bolígrafos, árboles, etc.
- Lentitud para asimilar conceptos matemáticos: matrices, ángulos, objetos geométricos como cuadrados o triángulos…
- Bajas calificaciones en asignaturas de matemáticas o relacionadas, como la física, química, dibujo técnico…

2.3 Repercusiones

Si esta dificultad del aprendizaje no se solventa, estas son algunas de las repercusiones a largo plazo en los individuos [6] en sociedad:

- Baja autoestima, al no ser capaces de realizar las tareas como el resto de personas.

- Aislamiento social, por sentir que se es diferente al resto de individuos sin esa dificultad.
- Dificultad en los aspectos que requieran el uso de las capacidades matemáticas: leer un código postal, pagar en efectivo, recordar números de teléfono, medir tiempos, gastos, etc.
- Lentitud para seguir indicaciones espaciales o cumplir normativas: avanzar 800 metros, no ir a más de 80 km/h, etc.
- Dificultad para alcanzar un nivel alto de estudios.
- Depresión.
- Altos niveles de ansiedad.
- Sufrir de otros trastornos psicológicos.
- Mayor probabilidad de sufrir trastornos de conducta.
- Consumir tabaco, alcohol u otras substancias.
- Dificultades en relaciones interpersonales.
- Desmotivación para el estudio.
- Mayor probabilidad de tener conflictos con la ley.
- Obtener una baja cualificación educativa o laboral.
- Bajo nivel de ingresos.
- Desempleo.
- Precariedad laboral.
- Dificultades para emanciparse.

2.4 Diagnóstico

Además del reconocimiento de los síntomas anteriormente mencionados, se han desarrollado una serie de pruebas para la detección de la discalculia. Algunas de ellas son:

- Test TEDI-MATH. Este producto es una batería de tests que consta de 25 pruebas diferentes agrupadas en 6 ámbitos de las matemáticas. Está dirigido a niños entre 4 y 8 años. [2]

- Test CAB-DC. Esta prueba online permite realizar una imagen cognitiva centrada en el riesgo de presencia de la discalculia. Está dirigido a niños mayores de 7 años, jóvenes y adultos. [7]
- NEUREKA-TEST. Este producto evalúa distintas dificultades en el aprendizaje: atención sostenida, lectura, memoria, procesamiento numérico, cálculo y TDAH. Está dirigido a niños entre 5 y 12 años. [8]
- Test BERDE. Evalúa distintas capacidades con las matemáticas, así como la ansiedad del paciente a la hora de enfrentarse a problemas matemáticos. Está dirigido a alumnos de toda la educación primaria. [9]
- Test Smartick. Más breve que los anteriores, también pone en relevancia la velocidad de la resolución de las tareas. Está dirigido a alumnos entre primero y cuarto de primaria. [10]

2.5 Tratamiento

Hay una serie de características que debe tener la enseñanza individual al alumno que padece de discalculia [11]:

- Enseñanza más intensiva, explícita y práctica sobre el sentido numérico.
- Período de tiempo más extenso en el aprendizaje de los conocimientos básicos.
- Proporcionarle experiencias concretas con los números grandes y pequeños.
- Trabajar y repasar constantemente la noción de proporción y cantidad: conceptos como mucho, poco, bastante, más o menos, mayor, menor, etc.
- Hacer hincapié en la asociación del número con la cantidad que representa. Es conveniente utilizar referentes visuales, concretos y manipulables.
- Contar y hacer grupos de objetos, utilizar el ábaco en los cálculos.
- Practicar muchos ejercicios de seriación. Presentar series de números y ordenarlos de mayor a menor y viceversa, completar los que falta, etc.
- Estimular la memoria a corto plazo y entrenar la atención sostenida, a través de ejercicios específicos.
- Practicar diariamente el cálculo mental: primero sumas y restas simples y más adelante ir incluyendo multiplicaciones y divisiones.

- Trabajar la correspondencia entre el lenguaje matemático y las operaciones necesarias para resolver un problema.
- Utilizar recursos informáticos con el objetivo de hacer más atractivas las tareas y facilitar la práctica diaria en el cálculo, las tablas de multiplicar y la resolución de problemas.

Además, hay una serie de factores que protegen a los alumnos de estas y otras dificultades del aprendizaje [6]:

- Aumentar el personal especializado en dificultades del aprendizaje en los centros educativos.
- Aumento de la formación del profesorado en esta materia.
- Metodologías que ayuden a incluir socialmente a los alumnos con dificultades con otros grupos de alumnos.
- Reducción del ratio profesor-alumno.

Capítulo 3

Estado del arte

En este capítulo se va a realizar un breve estudio de las herramientas que existen actualmente que realizan funciones similares o complementarias a la desarrollada con esta aplicación. Se centrará fundamentalmente en aplicaciones digitales que hacen pruebas de diagnóstico de la discalculia y, de forma adicional, otras dificultades en el aprendizaje y, por otra parte, aquellas que sirven también como tratamiento o apoyo.

Además, se ofrecerá un estudio comparativo y, como conclusión, una serie de objetivos que tendrá este proyecto relativos a lo visto en las herramientas descritas.

3.1 Herramientas similares

En este apartado se incluyen aplicaciones digitales que permiten la realización de pruebas de diagnóstico de la discalculia y, de forma adicional, otras dificultades en el aprendizaje.

3.1.1 NeurekaTEST

Esta batería de test fue creada y validada en la Universidad de Barcelona y en la Universidad de Vic-Universidad Centra de Cataluyna, mediante el proyecto NeurekaLAB. Ha sido diseñada para niños de toda la educación primaria.

Esta herramienta detecta señales de alerta sobre la presencia de las siguientes dificultades del aprendizaje en niños: discalculia, dislexia, falta de memoria de trabajo, atención sostenida y TDAH. El test de evaluación de discalculia dura 15 minutos y está diseñado para ordenador.

Una vez que el diagnóstico de esta herramienta está finalizado, NeurekaLAB recomienda el uso de NeurekaNUM para su tratamiento, el cual se describe más adelante en el documento.

Por último, la aplicación de este test tiene un coste monetario, con dos posibles planes: plan de profesional independiente o plan de escuela [8] .

3.1.2 CogniFit CAB-DC

Este test [7], de la compañía CogniFit, permite evaluar el índice de riesgo de la discalculia en el individuo.

Está pensado para niños mayores de 7 años, jóvenes y adultos. Las pruebas tienen una duración total de entre 30 y 40 minutos, y son para múltiples plataformas: ordenador, Android smartphone, Android tablet, IPhone y IPad.

La prueba comienza con un cuestionario inicial que evalúa los síntomas y signos clínicos para su edad y, a continuación, se realizan una serie de ejercicios y tareas en forma de sencillos juegos de ordenador.

Esta compañía dispone de una serie amplia de tests cognitivos, algunos centrados en dificultades del aprendizaje: dislexia, discalculia, TDAH, Parkinson o depresión, entre otros. Algunos de ellos son gratuitos y otros requieren su compra, pero el de la discalculia, en particular, no pertenece a ninguna de esas categorías, sino que está reservado para fines de investigación. Por tanto, para usarlo hay que contactar directamente con la empresa.

3.1.3 Smartick: Test de Discalculia

Esta prueba [10] permite detectar si hay riesgos de discalculia en la persona que lo realiza. Está diseñada para niños de los primeros cuatro cursos de primaria.

Tiene una duración de 15 minutos y evalúa los siguientes campos:

- Comparación y reconocimiento de cantidades: comparación de puntos y subitización (capacidad para reconocer cantidades sin contar).
- Números arábigos y numeración: reconocimiento de números, comparación de números, línea numérica mental, recta numérica, conteo y secuencias numéricas.
- Aritmética: suma, resta y multiplicación.

Este test es completamente gratuito y es posible su uso en tablets y ordenadores.

3.2 Herramientas complementarias

Este apartado se focaliza en las aplicaciones digitales que sirven como herramientas de apoyo al tratamiento de la discalculia.

3.2.1 NeurekaNUM

Dentro del proyecto NeurekaLAB, esta herramienta es la continuación a NeurekaTEST y, por tanto, sirve para realizar las intervenciones necesarias recomendadas por la herramienta

anterior. Además de su uso en aquellos estudiantes diagnosticados con discalculia, también se describe como una herramienta que potencia el aprendizaje en las matemáticas en aquellos niños que no padecen este trastorno.

Está diseñada para niños entre 5 y 8 años, mientras que NeurekaTEST estaba pensada para niños entre 5 y 12.

Utiliza la metodología de aprendizaje basado en videojuegos, de forma que los niños perciben esa sensación familiar de estar jugando con un videojuego cuando manejan esta herramienta, que además hace uso de dibujos infantiles (cohetes, marcianos, etc.). Las sesiones duran entre 10 y 15 minutos al día.

Junto con NeurekaTEST, el precio de NeurekaNUM es de 29€/mes por cada niño. Como con NeurekaTEST, también se pueden consultar precios para profesionales y escuelas [12].

3.2.2 Smartick

Este método online [10] está dedicado a niños entre 4 y 14 años. Está preparado para ser usado durante 15 minutos cada día en sesiones individuales para cada niño. La aplicación está disponible para tablets y ordenadores.

La aplicación trabaja cuatro áreas: cálculo mental, razonamiento, lógica y programación. A través de pequeños ejercicios, esta aplicación es capaz de reforzar las capacidades del niño afectadas por la discalculia.

La aplicación tiene un método de subscripción por pagos y cuesta entre 19,92€/mes y 44€/mes, en función de los meses y los niños. También hacen precios especiales para colegios y centros educativos.

3.3 Conclusiones y objetivos de la herramienta

En la tabla 3.1 se hace un estudio comparativa de las aplicaciones descritas anteriormente. Con esta información, se destacan los siguientes objetivos:

- El usuario de la aplicación no será el paciente, sino el logopeda, que tendrá control completo sobre la aplicación y sus funcionalidades.
- No sólo se mostrarán la puntuaciones resultado de la evaluación, sino también se permitirá que el logopeda pueda realizar sus anotaciones, como si de un caso clínico se tratara.
- No se pondrá el foco en reducir la duración de la prueba, sino de implementar una lo más completa posible, aunque eso conlleve aumentar el tiempo de la prueba. Para ello se hará una implementación del test TEDI-MATH [2] .

	NeurekaTEST	**CogniFit CAB-DC**	**Smartick**
Multiplataforma	No	Sí	Sí
Duración de evaluación	15 minutos	30 a 40 minutos	15 minutos
Usuario de la aplicación	Paciente	Paciente	Paciente
Edad de pacientes	5 a 12 años	Mayores de 7 años	5 a 10 años
Se elaboran informes	Sí	Sí	Sí
Apartado de gestión de usuarios, datos, grupos	No	No	Parcial
Precio	29€/mes	No publicado	19,92 €/mes
Tiene herramienta de refuerzo	Sí	No	Sí
Anotaciones	No	No	No

Cuadro 3.1: Comparativa inicial del estado del arte

- Reducir la edad de paciente evaluado para un diagnóstico más temprano.
- Elaborar un apartado de gestión de usuarios, grupos y datos.
- Elaboración de informes.
- No es fundamental, pero sí recomendable que la aplicación sea multiplataforma.

Capítulo 4

Fundamentos tecnológicos

En este capítulo se va a realizar una breve descripción de las tecnologías usadas a lo largo del proyecto.

4.1 Análisis de Requisitos

Microsoft Word

Microsoft Word [13] es un software de procesamiento de textos que permite la creación de documentos, páginas web o tablas, entre otros, a partir de un documento de texto.

Se ha usado esta herramienta para realizar el documento con los requisitos de la aplicación y para estructurar las tablas de la base de datos de la aplicación, debido a su alta capacidad y herramientas para la modificación y escritura de textos.

4.2 Diseño

Dia

Dia [14] es una herramienta que permite la realización de diagramas estructurados. Se usó por su alta capacidad para la realización de esquemas técnicos en la ingeniería.

En la fase de diseño se ha usado Dia para la realización de los diagramas de entidad-relación de base de datos y para diseñar la arquitectura de la aplicación.

4.3 Entorno de desarrollo

Android Studio

Android Studio [15] es el entorno de desarrollo oficial para la plataforma Android. Esta plataforma ofrece no sólo un entorno de desarrollo para las aplicaciones Android, sino que

también tiene un emulador que permite usar una amplitud de imágenes de Android sobre las que probar la aplicación. Otra de sus ventajas es su agilidad, no sólo para desarrollar código fuente, sino también para realizar ficheros de *layout* (diseño) usados para la creación de pantallas, entre otros.

Por todo esto, es una herramienta esencial para el desarrollo de proyectos en Android.

4.4 Lenguajes de desarrollo

Java

Java [16] es un lenguaje de programación orientado a objetos que fue comercializado en 1995 por Sun Microsystems.

Hay dos lenguajes que se usan principalmente para la programación de aplicaciones Android: Java y Kotlin. Aunque en los últimos años se está popularizando el uso de Kotlin, finalmente se ha escogido Java para el desarrollo de la aplicación, ya que ya se estaba familiarizado con el desarrollo en este lenguaje.

XML

Similar a HTML, XML [17] es un lenguaje de marcado que permite la creación de ficheros con los diseños de pantallas en Android. Más tarde, desde Java, se permite la obtención de los objetos de diseños, como botones o editores de textos, creados en los ficheros.

4.5 Base de datos

Room

Room [18] es una capa de abstracción sobre SQLite que permite acceder a la base de datos de forma ágil y sencilla en Android. De esta forma se realiza la creación de una base de datos, las entidades y los Data Access Objet (DAO) de forma rápida y abstrayéndose de problemas que surgen normalmente al tratar de crear estos elementos de la forma clásica en SQLite, como el mantenimiento de la Base de Datos y la configuración de las comunciaciones con la misma.

4.6 Gestión del proyecto

Microsoft Excel

Microsoft Excel [19] es una herramienta que permite la creación de tablas de datos que se pueden relacionar y con los que se pueden realizar distintas operaciones.

Se ha utilizado Microsoft Excel para el seguimiento del proyecto. De esta forma, cada vez que se realiza una sesión de trabajo, también se crea una entrada en un fichero con el registro de distintos datos, como el tiempo empleado o el tipo de actividad realizada. De esta forma, se pudo seguir de forma clara e intuitiva el estado actual del proyecto mientras se realizaba.

Se usó esta herramienta por familiaridad con ella y su capacidad de realizar tanto tablas como operaciones entre celdas, como el sumatorio de horas dedicadas a una tarea.

4.7 Control de versiones

Git y GitLab

Git [20] es una herramienta que permite realizar el control de versiones de proyectos software, ayudando a gestionar distintas ramas de desarrollo de forma paralela y a crear *tags* (etiquetas) con las distintas versiones u otros puntos importantes del proyecto.

En el caso de este proyecto, como solo existe un desarrollador activo, se usaron dos ramas:

- *master*: rama maestra, donde se realizan las publicaciones de las distintas versiones del programa.
- *develop*: rama de desarrollo, donde se implementan las funcionalidades de la aplicación y se corrigen errores.

Normalmente se establecen ramas paralelas para *hotfix* (la corrección de errores en caliente) y *develop* (desarrollo de nuevas funcionalidades). Sin embargo, se ha reutilizado la rama de *develop* porque en ningún caso se detectó un error mientras se desarrollaban funcionalidades nuevas que necesitase crear una rama paralela.

Git es, desde mi punto de vista, una herramienta fundamental para el desarrollo de proyectos software a nivel profesional, ya que permite todo lo descrito anteriormente y mayor utilidad todavía al trabajar varias personas de forma paralela con su metodología.

Al principio se usó el GitLab administrado por la Falcultade de informática da Coruña (FIC) hasta que se retiró este servicio. Entonces, se pasó a usar la plataforma oficial de GitLab [21] para el desarrollo del proyecto.

4.8 Seguimiento del proyecto

Microsoft Teams

Microsoft Teams es una herramienta que abarca muchas funcionalidades: gestión de grupos de trabajo, compartición de ficheros, chat, llamadas, entre otros.

En el caso de este proyecto, se ha usado Microsoft Teams para establecer el grupo de trabajo, compartir ficheros, chat, planificación y realización de reuniones.

4.9 Conclusiones

Con este conjunto de herramientas se creó un entorno de desarrollo del proyecto organizado, gracias a las interacciones entre las tecnologías anteriores. A destacar lo siguiente:

- Rápida edición e identificación de los requisitos fundamentales de la aplicación, así como la elaboración de documentación de diseño de base de datos con Microsoft Word [13].
- Comunicación rápida y efectiva entre el alumno y los tutores mediante Microsoft Teams [22].
- Alta claridad e identificación de las tareas y fases del proyecto en las que se invirtió altas cantidades de tiempo al seguir el proyecto con Microsoft Excel [19].
- Alta definición en los diagramas creados con Dia [14].
- Capacidad para la elaboración del proyecto completo con todas las herramientas incluidas en Android Studio [15].
- Guardado sistemático y marcado de versiones con Git [20].
- Agilidad de aprendizaje e implementación en la capa de base de datos con Room [18].

Capítulo 5

Metodología de trabajo

Las metodologías de trabajo en el desarrollo de proyectos software tienen como objeto la realización de un producto final de calidad de forma organizada y bajo unos estándares. Estas metodologías han ido variando a lo largo del tiempo, pero siempre manteniendo su importancia a la hora de la realización de los proyectos.

Cabe destacar que, a pesar de la gran variedad de metodologías en el ámbito de desarrollo software, no existe una metodología única para realizar todos los proyectos, sino que unas suelen ser más apropiadas para unos proyectos y otras para otros. Incluso, es común que en el mismo proyecto se usen varias metodologías de trabajo diferentes para las distintas fases.

Sin embargo, trabajar sin metodología tiende a terminar en desarrollos complejos que implican retrasos, dificultad, sobrecostes, errores y, en el peor caso, cancelación por insostenibilidad del proyecto.

5.1 Tipos de metodologías

A continuación se describirán brevemente los dos grandes grupos de metodologías de desarrollo software: las tradicionales y las ágiles.

5.1.1 Metodologías tradicionales

Las metodologías de desarrollo software tradicionales tienen como característica principal la definición inicial de los requisitos en su totalidad y la estructuración completa de las distintas fases del proyecto antes de conseguir el producto final.

La principal ventaja de estas metodologías es su alta claridad de ejecución, una vez se han establecido los requisitos del producto final. Esto es debido a que están especializadas, cada una en su forma, en la ejecución ordenada del desarrollo de proyectos software en su completitud.

Por otra parte, muchas de estas metodologías, si se tiene alta competencia en ellas, son sencillas de seguir, al estar fundamentalmente bien estructuradas y con productos técnicos, definidos y estandarizados, entre fases que sirven como intermediarios entre las mismas.

Su principal debilidad es que la premisa que hace que estas metodologías funcionen falle, es decir, que cambien los requisitos a lo largo del proceso de desarrollo. Esto tiene consecuencias negativas en mayor o menor grado, en función de la metodología que se aplique.

Algunas de las principales metodologías de desarrollo software tradicionales más comunes son:

- **Cascada**: Esta metodología es la más rígida, ya que las etapas del desarrollo se definen ”de arriba a abajo”. Esto quiere decir que la siguiente fase ocurre necesariamente cuando acabe la actual y se hayan desarrollado los productos acordados necesarios para la siguiente fase.
- **En V**: El método en V es similar al método en cascada, salvo que incluye, para cada fase en la que se definan requisitos, diseños u otros documentos técnicos, una complementaria en la que se verifica que el producto final cumple los requisitos, diseños o documentos pactados.
- **Espiral**: En esta metodología existen series de fases (vueltas a la espiral) de cuatro etapas: planificación, análisis de riesgos, desarrollo de prototipo y evaluación del cliente. Cuanto más se avance entre fases, más cerca se está de tener el funcionamiento completo del producto final.
- **Iterativa**: Esta metodología se caracteriza por, en vez de trabajar con un producto final, lo hace con iteraciones que devuelven productos tangibles y que pueden ser evaluados de forma individual. Más adelante se agregan para llevar al producto final.
- **Incremental**: La metodología incremental se caracteriza, de forma similar a la iterativa, en que se realiza en iteraciones definidas. Sin embargo, en vez de trabajar en iteraciones independientes en mayor o menor grado, se trabaja con incrementos de capacidades, funcionalidades, etc.

5.1.2 Metodologías ágiles

En la actualidad, cada vez es más común el uso de metodologías de desarrollo ágiles. Su principal característica, frente a las tradicionales, es su alta capacidad de flexibilidad y agilidad de tareas. Esto es muy conveniente en el peor caso para las metodologías tradicionales, que es cuando los requisitos varían en mitad del desarrollo.

Normalmente, se desarrollan en ciclos cortos de trabajo y con alta comunicación entre todas las partes del equipo. De esta forma, los productos se desarrollan poco a poco y se

realiza una verificación constante de que lo que se está haciendo es lo que desea el cliente, y éste tiene la capacidad de aportar nuevos requisitos, cambiar requisitos anteriores o realizar correcciones.

Algunas de las principales metodologías de desarrollo software ágiles más comunes son:

- **Extreme Programming XP**: Esta metodología se apoya en la comunicación fundamental con el cliente y entre los programadores. El proyecto se planifica con el cliente y el código se diseña en grupo [23] .
- **Scrum**: Similar al iterativo-incremental. Scrum reduce los incrementos en *Sprints* donde, con cierta regularidad, se entregan de forma parcial partes del producto final al cliente [24].
- **Kanban**: En esta metodología se elabora un diagrama con tres columnas de tareas: pendientes, en proceso o terminadas. Dicha tabla está al alcance de todos los miembros del equipo. Las tareas van pasando por los tres estados hasta conseguir el producto final [25].

5.2 Metodología usada en el proyecto

En este proyecto se ha seguido, en su mayor parte, una metodología iterativa-incremental, combinada con reuniones periódicas entre los directores del proyecto y el desarrollador.

A continuación, se explican las fases en las que se divide el proyecto.

5.2.1 Búsqueda de información

Esta primera fase consiste en acudir a diversas fuentes de información, bibliotecas y páginas web, en búsqueda de información acerca de la discalculia y las dificultades del aprendizaje, además de buscar información acerca de otras herramientas existentes en mercado actual que realicen funcionalidades similares o complementarias.

5.2.2 Análisis de requisitos

En esta fase se definen los requisitos que deben cumplir la aplicación. Para cumplir con los requisitos de forma ordenada, se se elabora una jerarquía de tres niveles:

- **Primarios**: Requisitos fundamentales, de necesario cumplimiento para tener un producto final acorde con las expectativas del proyecto.
- **Secundarios**: Requisitos no fundamentales para la elaboración de un producto final suficiente pero con la capacidad de otorgar al proyecto una alta calidad.

- **Terciarios**: Similar a los secundarios, pero con menor prioridad.

En consecuencia, se puede establecer un orden en la fase de implementación sobre qué requisitos se van a codificar en las primeras iteraciones y cuáles más adelante.

5.2.3 Diseño

En esta fase se elaboran los diagramas de los siguientes elementos:

- **Base de datos**: Esquematización mediante el método Entidad-Relación de los conceptos o entidades que interactúan o requieren persistencia en el proyecto.
- **Arquitectura de la aplicación**: Diagrama de las distintas capas del programa que interactúan en el proyecto, así como las responsabilidades de cada una de ellas.
- **Pantallas**: Se elaboran bocetos en los que se establecen las estructuras de las pantallas y la localización de los elementos con los que interactuará el usuario final de la aplicación.

Por último, se revisa que los todos los diseños lleguen a alcanzar, de forma teórica, la capacidad de cumplir todos los requisitos, tanto primarios como secundarios y terciarios.

5.2.4 Fase de implementación en iteraciones

Para elaborar los requisitos primarios, en esta fase se realizan una serie de iteraciones. En cada una de ellas se establecen las siguientes fases:

- **Planificación**: Saber qué se va a implementar en base a los requisitos que tengan mayor prioridad y a los diseños con el objetivo de saber cómo debe ser el resultado final de la iteración.
- **Alcance de la iteración**: Reflexionar de forma breve acerca de cuántos elementos va a ser necesarios implementar para alcanzar los objetivos de la iteración. Además, esta fase incluye también pensar también si lo que se haga en esta iteración se puede relacionar con lo implementado en alguna iteración anterior.
- **Búsqueda de información**: Conocer qué tecnologías, dentro del entorno de desarrollo del proyecto, hay disponibles para implementar lo planificado.
- **Codificación**: En base a toda la información anterior, implementar las clases simples, abstractas, interfaces, bloques de código o *layouts*, que sean necesarios para cumplir con lo que se planificó al principio de la iteración.

- **Pruebas de iteración**: Se elaboran los recursos necesarios para realizar pruebas que verifiquen que lo realizado en la fase de codificación cumple con lo planificado al principio de la iteración.
- **Pruebas de integración**: Una vez que se haya verificado que los productos de la iteración cumplen con lo planificado, se validan en simulaciones que no interfieran y se integren correctamente con los productos de iteraciones anteriores. Sólo se hace si hay una interacción directa entre lo implementado en la iteración con alguna de las anteriores.

De esta forma, se obtienen las siguientes ventajas:

- **Distribución de las pruebas**: Las pruebas que, bajo otras metodologías, se hacen al final del proyecto se realizan a lo largo de las iteraciones.
- **Eficacia al solucionar errores**: Los errores son más fáciles de encontrar, ya que los bloques de código que se prueban son de menor tamaño.
- **Eficiencia al solucionar errores**: Los errores se detectan a las pocas horas de realizar el código que los provoca, lo que evita tener que buscar información de nuevo sobre la codificación de la iteración.

  Al mismo tiempo, al tener los conocimientos usados para codificar el contenido de la iteración más recientes, es más probable que se encuentre antes la solución a los errores.

  Por tanto, se destinam menos recursos a la corrección de errores y se obtienen mejores resultados.
- **Mayor dinamismo**: Se realiza un mayor número de tareas de distintos tipos en las iteraciones, lo que se adapta mejor a la forma de trabajar del desarrollador.
- **Visibilidad del trabajo**: De cara a las reuniones con los tutores en el proyecto, siempre se tiene un producto reciente y con novedades que se pueden observar. De esta forma, los tutores tienen la capacidad de valorar el trabajo del alumno sobre partes del producto que se elaboran de forma mejorable.

Por último, al final de la implementación, se obtiene un producto que ha cumplido con los diseños y con los requisitos primarios del proyecto.

5.2.5 Fase de implementación en incrementos

Una vez que se ha conseguido un producto completo con los requisitos primarios, se avanza a una fase de implementación en incrementos. Dichos incrementos pueden estar motivados por lo siguiente:

- **Añadir funcionalidades**: En base a los requisitos secundarios o terciarios, se seleccionan uno o varios para implementar.
- **Modificar funcionalidades**: Como se están realizando pruebas de la aplicación, se pueden encontrar algunos aspectos de mejora que puedan modificarse, como pueden ser textos o acceso a elementos de pantalla.
- **Corrección de errores**: Se detecta algún error que no había sido detectado en la fase anterior del proyecto, y que ha de ser subsanado con prioridad alta.

De cara a cumplir con los objetivos seleccionados, y de forma similar a la implementación en iteraciones, se realizan incrementos que se estructuran en las siguientes fases:

- **Planificación**: Conociendo el objetivo del incremento, se elabora un boceto para representar los elementos de la pantalla con los que interactuará el usuario final cuando se obtenga el producto final, si es necesario. También se actualizan, si fuera necesario, los diagramas que se hicieron en las fases anteriores.
- **Búsqueda de información**: Conocer qué tecnologías, dentro del entorno de desarrollo del proyecto, hay disponibles para implementar lo planificado.
- **Codificación**: En base a toda la información anterior, implementar las clases simples, abstractas, interfaces, bloques de código o *layouts*, que sean necesarios para cumplir con lo que se planificó al principio del incremento.
- **Pruebas de incremento**: Se elaboran los recursos necesarios para realizar pruebas que verifiquen que lo realizado en la fase de codificación cumple con lo planificado al principio del incremento.
- **Pruebas de integración**: Una vez que se haya verificado que los productos del incremento cumplen con lo planificado, se validan en simulaciones que no interfieran y se integren correctamente con los productos de iteraciones anteriores.

Por último, se etiqueta un número de versión del producto que el usuario final podrá ver en la aplicación al usarla. Al mismo tiempo, en el repositorio de Git [20] se elabora una etiqueta-resumen con la *release* (liberación) asociada a la versión. Dicho *release* tiene los siguientes campos:

- **Nombre**: Número de versión correspondiente al incremento.
- **Contenido**: Objetivos que se cumplieron en la iteración.

5.3 Conclusiones

En este proyecto, los requisitos estaban establecidos desde el principio, por lo que diseñar una metodología de desarrollo software tradicional era lo apropiado, amparándose en la tesis de que están diseñadas para cumplir con unos requisitos estáticos en el tiempo.

Gracias a lo anterior, se elaboran unos documentos de diseño que también permanecen estáticos a lo largo del desarrollo del proyecto. Esto provoca que la implementación en iteraciones de los requisitos primarios de la aplicación sea acorde a dichos diseños y sean sencillos de seguir.

Por ejemplo, en la iteración ”Elaborar pantalla de pacientes” hay que codificar una pantalla que sea acorde con los bocetos realizados al comienzo del proyecto. Aunque dicha pantalla se diseñe en las primeras semanas del proyecto, permanece inmutable hasta que se implementa en el futuro.

Gracias a esa seguridad, el desarrollador tiene una idea clara de cómo va a ser el resultado final de la iteración previamente a codificarla. Dicha idea sirve de guía en el proceso de desarrollo. También reduce el tiempo de búsqueda de información al tiempo necesario para reproducir la idea del diseño en código del proyecto.

Otra de las claves de este proceso iterativo es el orden en que se implementan las iteraciones. En el enfoque práctico, se implementan de forma secuencial aquellas iteraciones que se van a codificar de forma similar.

Por ejemplo, aunque se verá más adelante, las pantallas con el esquema Listado - Información - Creación (LIC) se implementan en iteraciones consecutivas. La capas de Servicio y Acceso a Base de Datos se codifican una vez que ya se han implementado, en una clase prototipo, aquellas llamadas que la Capa de Visión de la Aplicación necesita, sabiendo la totalidad de llamadas que serán necesarias antes de desarrollarlas por completo. De esta forma, los conocimientos adquiridos en una iteración, así como el código que la implementa, se recicla de una iteración a la siguiente, minimizando el tiempo requerido para completarlas.

Las iteraciones son lo suficientemente pequeñas como para parecer abordables en cortos periodos de tiempo.

Por último, gracias a la medición del alcance de la iteración antes de codificarla, se pueden ver de antemano posibles conflictos que se vayan a encontrar en el futuro, tanto en el código de la iteración como en el de iteraciones pasadas. Este proceso se puede criticar por su coste en tiempo, en el que no siempre es necesario invertir, ya que no va a haber en todas errores de integración con otras iteraciones. En primera instancia se puede asumir, al menos, que es una preocupación razonable. Pero, a lo largo del tiempo, se obtiene el efecto secundario de reducir la complejidad del código, ya que se evita la existencia de bloques de códigos duplicados o incluso clases duplicadas. Esto no se observaría si no se echara un vistazo al pasado

del proyecto, y se considera que esta es una ventaja frente a otras metodologías, entre ellas las ágiles.

Capítulo 6

Planificación y costes

En este capítulo se va a visualizar la planificación y estimación de costes para la realización de este proyecto. Al mismo tiempo, se hará una comparación de los datos estimados con los datos del seguimiento al 100% del proyecto.

6.1 Planificación

Esta sección incluye la planificación del desarrollo del proyecto.

6.1.1 Planificación inicial

Antes de iniciar el proyecto, se realizó una estimación de la duración del mismo. Para ello, se estimaron inicialmente las horas que haría falta dedicar para realizar las tareas descritas en el capítulo anterior. Dichas estimaciones están resumidas en la tabla 6.1.

En Android, la capa vista tiene un gran peso dentro de la programación, por lo que se ha estimado esa cantidad de horas mucho más elevada si se compara con el resto.

Fase	Tiempo estimado
Búsqueda de información	10 horas
Análisis de requisitos	4 horas
Diseño	40 horas
Iteraciones de capa vista	160 horas
Iteraciones de capa servicio y acceso a base de datos	40 horas
Incrementos	40 horas
Total	294 horas

Cuadro 6.1: Planificación inicial del proyecto.

Fase	Estimación	Seg. al 100%	Diferencia
Búsqueda de información	10 horas	6,5 horas	-3,5 horas
Análisis de requisitos	4 horas	2 horas	-2 horas
Diseño	40 horas	22,25 horas	-17,75 horas
It. Capa Vista	160 horas	183 horas	+23 horas
It. Capa Servicio y Ac. a BBDD	40 horas	26,25 horas	-13,75 horas
Incrementos	40 horas	17,5 horas	-22,5 horas
Reuniones	0 horas	16,25 horas	+16,25 horas
Manual de Uso	0 horas	13,5 horas	+13,5 horas
Total	294 horas	287,25 horas	-6,75 horas

Cuadro 6.2: Seguimiento al 100%.

6.1.2 Seguimiento al 100%

Al finalizar el proyecto, se realiza una comparativa entre la estimación inicial y el número final de horas que ha llevado realizarlo. Esto fue posible gracias a un seguimiento constante a lo largo de todo el desarrollo del proyecto del número de horas dedicado a cada una de las tareas, lo cual se registró en un fichero de Microsoft Excel [19].

El resumen del seguimiento se puede ver en la figura 6.2.

Se toma la siguiente información como relevante de cara a realizar estimaciones en el futuro:

- **Búsqueda de información**: La tarea se realizó en el 65% del tiempo estimado, ya que el proceso fue más rápido de lo esperado.
- **Análisis de requisitos**: La tarea se realizó en el 50% del tiempo estimado. Esto fue debido a la rápida identificación de los requisitos más fundamentales.
- **Diseño**: La tarea se realizó en el 55,63% del tiempo estimado, ya que se redujo la calidad de los esquemas de las pantallas de la aplicación a bocetos, en vez de realizarse mediante una herramienta profesional, con el objetivo de ahorrar tiempo.
- **Iteraciones de la Vista**: La tarea se realizó en el 114% del tiempo estimado. Esto es debido a que, a pesar de haber proporcionado una estimación alta, en Android gran parte de la carga de trabajo está en el manejo de las capas más cercanas al usuario.
- **Iteraciones de Capa de Servicio y Acceso a Base de Datos**: La tarea se realizó en el 65,63% del tiempo estimado, debido a la correcta planificación de las iteraciones y a la fluidez en la programación con la librería Room [18].
- **Incrementos**: La tarea se realizó en el 43,75% del tiempo estimado, en gran medida porque se redujeron los requisitos necesarios para dar con un producto final de suficiente

calidad como para ser entregado.

- **Reuniones y Manual de Uso**: Estas dos tareas no se tuvieron en cuenta de cara a la estimación, pero sí se hizo un seguimiento de ellas, ya que son parte de la realización del proyecto. En un futuro, ambas se tendrán en cuenta en la estimación inicial.
- **Desviación total**: El proyecto se realizó en 97,75% del tiempo estimado.

6.2 Recursos empleados

En el desarrollo del proyecto se emplearon los siguientes recursos hardware y software.

- Hardware:
  - Ordenador:
    - Compañia: ASUSTek COMPUTER INC.
    - Tipo: Portátil.
    - Modelo del sistema: K55VD
    - Procesador: Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz, 2401 Mhz, 4 procesadores principales, 8 procesadores lógicos.
    - RAM instalada: 12 GB (modificado manualmente).
    - Almacenamiento: 231,77 GB.
    - Sistema Operativo: Microsoft Windows 10 Pro.
    - Tipo de sistema: PC basado en x64.
    - Tarjeta gráfica: NVIDIA GEFORCE 610M 2 GB.
  - Tableta:
    - Compañia: BQ.
    - Modelo: Aquaris M10.
    - Sistema Operativo: Android 6.0.
    - Memoria RAM: 1,9 GB.
- Software:
  - Microsoft Word [13].
  - Dia [14].
  - Android Studio [15].
  - Microsoft Excel [19].
  - GitLab [21].
  - Microsoft Teams [22].

Tarea	Seg. al 100%	Coste por hora	Coste tarea
Búsqueda de información	6,5 horas	15€/hora	97,5 €
Análisis de requisitos	2 horas	15€/hora	30 €
Diseño	22,25 horas	15€/hora	333,75 €
It. Capa Vista	183 horas	15€/hora	2.745 €
It. Capa Servicio y Ac. a BBDD	26,25 horas	15€/hora	393,75 €
Incrementos	17,5 horas	15€/hora	262,5 €
Reuniones	16,25 horas	49€/hora	796,25 €
Manual de Uso	13,5 horas	15€/hora	202,5 €
Total			4.861,25 €

Cuadro 6.3: Coste real de los recursos humanos.

6.3 Coste del proyecto

En este apartado se calculará el coste del proyecto en base al seguimiento del mismo y el coste de los recursos empleados. La duración del proyecto fue de 1 año, 1 mes y 18 días.

Al mismo tiempo se establecerá una comparación con los costes basados en las estimaciones iniciales, donde el desarrollador estimó que la duración del proyecto sería de 1 año.

6.3.1 Coste de recursos humanos

El coste de los recursos humanos se calculará en base al sumatorio de los siguientes elementos:

- **Horas en tareas individuales**: Cantidad de horas realizadas por el estudiante. Cada una de las tareas que no sean de reunión se incluirán en este grupo.
- **Horas en tareas colectivas**: Cantidad de horas realizadas por el conjunto de los miembros del proyecto. Se incluirán las horas dedicadas a reuniones.

Más adelante se multiplican las horas en tareas individuales por el sueldo del desarrollador, que es de 15€/hora. A continuación, se añade a ese cálculo la multiplicación de las horas en tareas colectivas por el sueldo del desarrollador más el sueldo de los directores, de 17€/hora cada uno, dando un total de coste del tiempo en estas tareas de 49€/hora (2 directores y 1 desarrollador). Estos datos están basados en los salarios medios en España de esta profesión [26] .

Este proceso se resume en la tabla 6.3, de la que se obtiene un coste de recursos humanos total de **4.861,25 €**.

En cuanto al coste estimado, hay que tener en cuenta que la estimación no consideró las reuniones, porque lo que sólo habrá un tipo de coste por hora, que es el correspondiente a las

Tarea	Estimación	Coste por hora	Coste tarea
Búsqueda de información	10 horas	15€/hora	150 €
Análisis de requisitos	4 horas	15€/hora	60 €
Diseño	40 horas	15€/hora	600 €
It. Capa Vista	160 horas	15€/hora	2.400 €
It. Capa Servicio y Ac. a BBDD	40 horas	15€/hora	600 €
Incrementos	40 horas	15€/hora	600 €
Total			4.410 €

Cuadro 6.4: Coste estimado de los recursos humanos.

Recurso	Coste	Tiempo de vida	Coste en el proyecto
Ordenador	579,06 €	4 años	164,19 €
Tableta	169,00 €	2 años	95,84 €
Total			260,03 €

Cuadro 6.5: Coste real de los recursos hardware.

horas en tareas individuales. Esto se representa en la tabla 6.4, la cual da un coste estimado de **4.410 €**.

6.3.2 Coste de recursos hardware

Dados los elementos hardware descritos anteriormente, su coste se calculará en base al coste del los productos [27] [28] en relación a la duración del proyecto (en años), 1,1342 años, sobre su tiempo de vida (también en años). Como se ve en la tabla 6.5, el coste real de los recursos hardware es de **260,03 €**.

En contraposición, el coste estimado se calculará con la misma fórmula, pero variando la duración del proyecto a la estimada, 1 año. En la tabla 6.6 se puede ver que la estimación era de **229,27 €**.

6.3.3 Coste de recursos software

Para realizar el cálculo del coste de los recursos software, primero hay que descartar los productos software utilizados durante el desarrollo del proyecto que fueron gratuitos y que,

Recurso	Coste	Tiempo de vida	Coste en el proyecto
Ordenador	579,06 €	4 años	144,77 €
Tableta	169,00 €	2 años	84,50 €
Total			229,27 €

Cuadro 6.6: Coste estimado de los recursos hardware.

por lo tanto, no repercutieron en coste alguno. Dichos productos son:

- Dia [14] .
- Android Studio [15] .

Por tanto, habrá que tener en cuenta el coste asociado a Microsoft 365 Empresa Básico [29], que es un paquete que incluye, entre otros, los siguientes elementos necesarios para realizar el proyecto:

- Microsoft Word [13].
- Microsoft Excel [19].
- Microsoft Teams [22].

Su coste es de 4,20 €/mes. Dado que la duración final del proyecto es de 13 meses y 18 días, se necesitará una suscripción de 14 meses para conocer el coste real imputable al proyecto, para los tres participantes. Por tanto, el coste real de recursos software será de **176,4 €**, tal y como puede comprobarse en el cálculo siguiente:

$$4,20\,€ \times 3 \times 14 = 176,4\,€.$$

Sin embargo, si lo comparamos con el coste estimado, hay que tener en cuenta que la duración estimada del proyecto es de 12 meses, y eso provoca que el coste se reduzca a **151,2 €**.

$$4,20\,€ \times 3 \times 12 = 151,2\,€.$$

6.4 Conclusiones

La planificación de un proyecto cuyo seguimiento presente una desviación mayor del 20% puede considerarse errónea y debería ser revisada. En este caso, al realizar un sólo seguimiento al final del proyecto, no se podía replanificar. Sin embargo, sí puede servir de medida para saber los déficits de la estimación, tanto en su total en el proyecto completo como en el conjunto de sus fases.

Sin embargo, se puede considerar un resultado positivo, por ser menor el tiempo de desarrollo del proyecto real que el estimado. Además, la diferencia es sólo del 2,25%, lo cual es un indicio de que el proyecto se estimó y planificó correctamente a nivel global.

A pesar de ello, existen a nivel de tarea ciertas deficiencias en la estimación. Las más importantes son:

Concepto de coste	Coste estimado	Coste real	Diferencia
Coste recursos humanos	4.410 €	4.861,25 €	+451,25 €
Coste recursos hardware	229,27 €	260,03 €	+30,76
Coste recursos software	151,2 €	176,4 €	+25,2 €
Total	7.142,47 €	7.651,05 €	+507,21 €

Cuadro 6.7: Comparación coste estimado y coste real en el proyecto.

- La ausencia en la estimación del proyecto de las reuniones con los directores.
- La ausencia en la estimación del proyecto de la redacción del Manual de Uso.
- La excesiva estimación en la fase de diseño, con un error del 44,37% entre la realidad y la estimación. Este porcentaje es similar en las fases de Búsqueda de Información y Análisis de Requisitos, pero estas tareas tienen menor repercusión, ya que su duración fue más corta, como se muestra en la tabla 6.2.
- En la fase de Iteraciones de Capa de Servicio y Acceso a Base de Datos hubo una diferencia del 34,37% en una tarea estimada de 40 horas. En este caso, la reducción en el tiempo en comparación con la estimación se produce por la agilidad del desarrollador al implementar las iteraciones. Esto fue gracias a la metodología que se usó.

Al realizar una comparación entre los costes estimados y los costes reales de los recursos, como se ve en la figura 6.7, se puede observar que, el proyecto ha conllevado un sobrecoste de **507,21 €**.

Para entender este sobrecoste, se deben tener en cuenta las siguientes claves:

- El proyecto se alargó 1 mes y 18 días en comparación con la estimación, lo que obliga a ampliar en 2 meses las suscripciones al plan Microsoft 365 Empresa Básico [29].
- La dilatación del proyecto en el tiempo también contribuye al aumento de costes en los productos hardware, ya que se usan durante más tiempo.
- No se estimó correctamente el proyecto, al no tener en cuenta las tareas de reuniones ni de redacción del Manual de Uso, necesario para el manejo de la aplicación. Esto conlleva también un fallo en el cálculo del coste de los recursos humanos, ya que en las reuniones participan tres recursos en vez de uno, lo cual incrementa el coste por hora.

Capítulo 7

Análisis de requisitos

En este capítulo se van a examinar los requisitos de la aplicación, así como el criterio que se usó para jerarquizarlos en niveles.

7.1 Requisitos

Este proyecto se enmarca como una actividad de aprendizaje y servicio [30], es decir, se pretende que el estudiante adquiera las competencias asociadas a la asignatura de Trabajo Fin de Grado al mismo tiempo que proporciona un servicio a la comunidad. En este caso, se ha colaborado con la Asociación Gallega de Asperger (ASPERGA) [31], entre cuyos usuarios es frecuente encontrar trastornos del aprendizaje de las matemáticas. Esta asociación establece las necesidades de la aplicación, tras varias reuniones con los tutores de este trabajo. Teniendo en cuenta estas necesidades, y en consonancia con los objetivos marcados en el anteproyecto, se establecen los siguientes requisitos:

- Implementación del test TEDI-MATH [2].
- Implementación de otros tests.
- Orientación de la aplicación a una atención a niños, en etapas de escolarización comprendidas entre segundo de Educación Infantil y tercero de Educación Primaria.
- En las primeras iteraciones, los logopedas son administradores.
- Aplicación multiplataforma: Android (obligatorio) e iOS (recomendable).
- Gestión de usuarios:
  - Alta y baja entre usuarios.
  - Gestión de contraseñas

  - Garantía de protección de datos.
- Interfaz amigable y divertida para los niños.
- Adaptación a diversidad:
  - Variabilidad en la configuración del tamaño de fuente.
  - Poder cambiar el color de la aplicación.
  - Cada niño pueda tener una imagen personalizable.
  - Mensajes de voz.
- Uso de pictogramas en menús e instrucciones.
- Capacidad de ser usado en móviles y tablets.
- Registro y almacenamiento de datos.
- Análisis de datos:
  - Visualización en el dispositivo.
  - Módulo de gráficas.
  - Exportación de datos a CSV.
  - Generación de informes en PDF.
  - Generación de informes grupales en PDF.
- Módulo de evaluación, donde se pueda evaluar la posibilidad de DAM mediante la realización de la batería de tests.
- Análisis, donde se visualizan los datos y se generan informes.
- Contacto, gestión de reuniones e intercambio de información entre familias y especialistas.

7.2 Roles en la aplicación

Los siguientes roles se definen en los requisitos:

- Administrador: Puede hacer todo.
- Logopeda: Puede realizar evaluaciones, ver y exportar datos de un paciente en concreto o de todos los que evalúa, y exportar informes.
- Familia: Sólo puede ver informes y exportarlos a PDF.
- Niño: no evalúa, visualiza datos, ni genera ni exporta informes.

7.3 Jerarquización de los requisitos

En esta sección se explora la jerarquía que se siguió para definir qué requisitos se implementan en las primeras iteraciones.

7.3.1 Requisitos de primer nivel

Estos requisitos se seleccionan con el objetivo de tener una aplicación funcional en todos los aspectos, pero que pueda tener en un futuro ampliaciones que incrementen su calidad. De esta forma, a la hora de implementar estos requisitos, se hará de forma que los siguientes sean fáciles de implementar.

En primer lugar, se reducen los roles:

- Logopeda: Puede hacer todo. Tiene acceso a la aplicación.
- Familia: Sólo puede ver informes y exportarlos a PDF. Tiene acceso a la aplicación.
- Paciente: no evalúa, visualiza datos, ni genera ni exporta informes. No tiene acceso a la aplicación.

Todos los logopedas podrán tener un grupo cada uno, y podrán ver, si lo desean, sólo los informes de los pacientes del grupo.

Se seleccionan los siguientes requisitos:

- A.17 Implementación del test TEDI-MATH [2].
- Orientación de la aplicación a una atención a niños, en etapas de escolarización comprendidas entre segundo de Educación Infantil y tercero de Educación Primaria [2].
- A.24 En las primeras iteraciones, los logopeda son administradores.
- Aplicación en Android.
- A.25 Gestión de usuarios:
  - Alta y baja entre usuarios.
  - A.23 Gestión de contraseñas.
  - A.27 Garantía de protección de datos.
- Adaptación a diversidad:
  - A.22 Poder cambiar el color de la aplicación.

  - A.21 Cada usuario puede tener una imagen personalizable. De esta forma, si se implementa el acceso de pacientes a la aplicación, esta parte ya estará lista.

- Capacidad de ser usado en tablets.
- Registro y almacenamiento de datos.
- A.20 Análisis de datos:
  - Visualización en el dispositivo.
  - Módulo de gráficas.
  - Generación de informes en PDF.
- A.17 Módulo de evaluación, donde se pueda evaluar la posibilidad de DAM mediante la realización de la batería de tests.
- A.20 Análisis, donde se visualizan los datos y se generan informes.

Con estos requisitos implementados, se espera tener un producto básico con capacidades suficientes.

7.3.2 Requisitos de segundo nivel

Estos requisitos se seleccionan para su implementación una vez que se hayan realizado todos los requisitos primarios. Su objetivo será el de tener una prioridad en aquellos requisitos no fundamentales que dotarían de mayor calidad a la aplicación una vez ya es utilizable.

Se modifican los roles hacia la estructura original.

- Administrador: Puede hacer todo.
- Logopeda: Puede realizar evaluaciones, ver y exportar datos de un paciente en concreto o de todos los que evalúa, que serían los definidos en el primer nivel como los pacientes de su grupo, y exportar informes.

  Por tanto, los logopedas pierden acceso a las funcionalidades de administración y a la visión de pacientes que no pertenecen a su grupo.
- Familia: Sólo puede ver informes y exportarlos a PDF.
- Niño: no evalúa, visualiza datos, ni genera ni exporta informes.

Se seleccionan los siguientes requisitos:

- Atención a la diversidad:

  - Mensajes de voz.
  - Uso de pictogramas en menús e instrucciones.
- Adaptación a móvil.
- Análisis de datos:
  - Exportación de datos a ficheros CSV.
  - Compartición de informes en PDF.
- Contacto. Gestión de reuniones e intercambio de información entre familias y logopedas. Sólo para familias y logopedas.

Con estos requisitos implementados, la aplicación aumentaría su calidad en gran escala.

7.3.3 Requisitos de tercer nivel

Los requisitos de tercer nivel, al igual que los del segundo, no son necesarios para tener una aplicación funcional pero sí la dotarían de la mayor calidad posible, una vez se implementen anteriormente los requisitos de primer y segundo nivel.

Los requisitos de esta categoría son los siguientes:

- Recomendación de implementar otro test de evaluación, además de TEDI-MATH.
- Recomendación de implementar la aplicación en iOS.
- Informes PDF de grupo, donde se muestren en un mismo informe los datos del grupo.

7.4 Conclusiones

De la forma en que se han definido los requisitos, el desarrollador es capaz de elaborar unos diseños que estén acordes a los requisitos primarios.

Al mismo tiempo, la presencia de los requisitos secundarios y terciarios hace que, en la realización de los diseños, se tenga en cuenta que en un futuro se puedan incorporar dichas funcionalidades. Por tanto, los diseños no deben imposibilitar añadir a la aplicación elementos que se conoce a priori que proporcionarían calidad al producto final.

Por último, esta jerarquización de los requisitos también permite la obtención de un producto funcional lo antes posible.

Capítulo 8

Diseño

En este capítulo se van a explorar las decisiones de diseño que se tomaron antes del proceso de implementación, acorde a los requisitos de la aplicación.

8.1 Diseño de la arquitectura de la aplicación

Se ha diseñado una arquitectura de tres capas software, para su implementación en Android-Java. A su alrededor, habrán dos elementos más. Una de ellos es el hardware, el dispositivo Android, y el otro será la base de datos.

Este diseño está representado en la figura 8.1, en la página 42, donde se muestra además cómo se relacionan dichos elementos entre sí.

8.2 Dispositivo tablet

Dado que en los requisitos se estableció como primaria la implementación en tabletas, esta capa hardware se definió de la misma forma.

Las tabletas tienen la ventaja sobre los teléfonos de su mayor tamaño de pantalla, por lo que se podrán añadir elementos auxiliares en las interfaces, así como dar más espacio a los elementos existentes.

La tableta será el elemento hardware donde se realizarán las siguientes funciones:

- **Visualización**: La pantalla será la plataforma donde se verán los datos que la Capa de Visión de la Aplicación ofrezca. Estos elementos serán los siguientes:
  - Imágenes.
  - Textos.
  - Campos de escritura.
  - Otros.

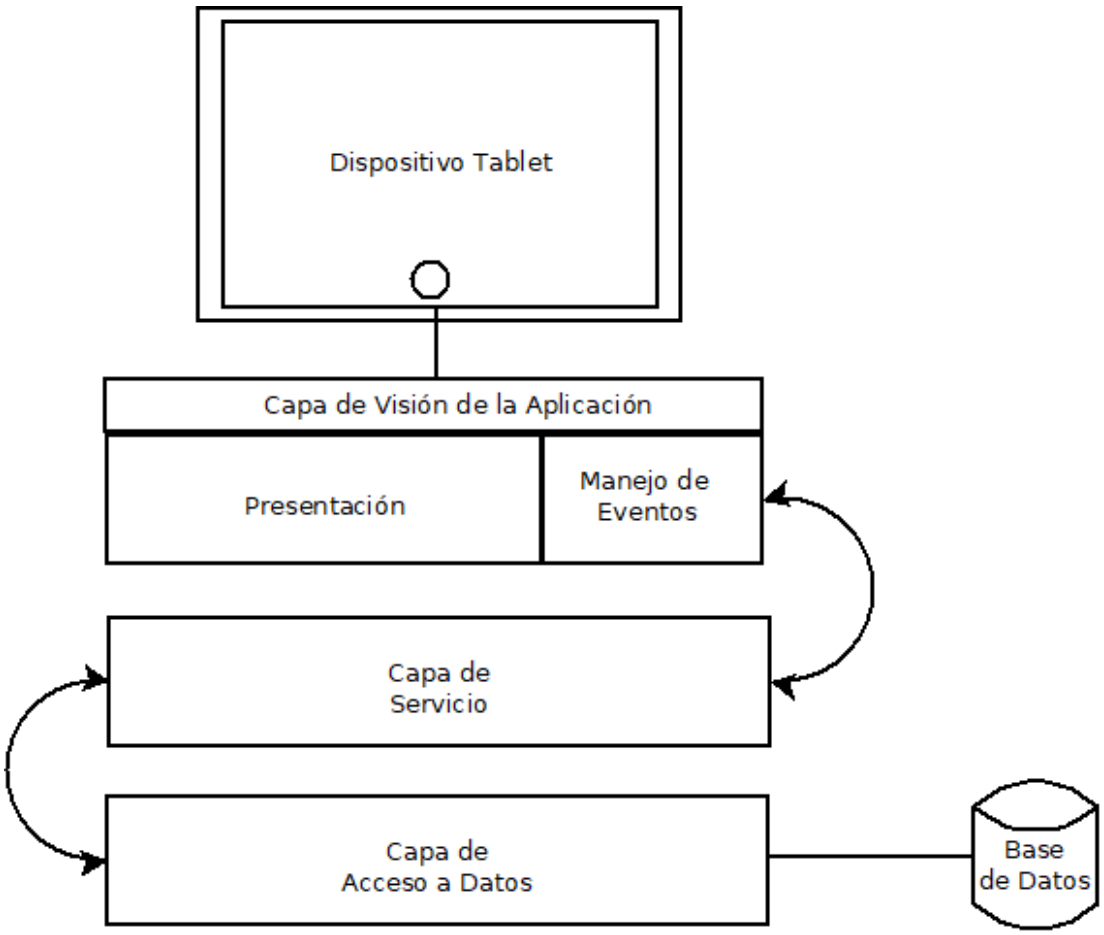


Figura 8.1: Arquitectura de la aplicación.

- **Acceso a funcionalidades**: La pantalla, a través de los elementos que ofrecerá la Capa de Visión de la Aplicación, permitirá una interacción táctil que llevará al usuario final hacia aquellas funcionalidades a las que quiera acceder.

Al mismo tiempo, la tableta no deberá saturarse de recursos ni quedarse congelada por la aplicación.

Por último, al usar la versión de Android 6.0, se tiene una amplia gama de elementos para la realización del proyecto.

8.3 Capas de la Arquitectura Software

A continuación, se describirán los elementos de la arquitectura software de la aplicación.

8.3.1 Capa de Visión de la Aplicación

Esta primera capa del desarrollo software tiene las siguientes responsabilidades:

- **Presentación**. Presentar al usuario final los datos de la aplicación. Esto se realizará con la creación de modelos, que la aplicación usará en esta capa para representar los distintos agentes que interactúan en la aplicación.

  Esta presentación tendrá una interfaz definida, clara y completa, para que el usuario final pueda observar los datos deseados en la aplicación.

  Dicha interfaz puede ser de dos tipos.

  - **Estática**: Hay una cantidad fija de elementos en unas posiciones determinadas. Por ejemplo, si se están viendo los datos personales de un paciente, serán los mismos para todos los pacientes y estarán en las mismas localizaciones.
  - **Dinámica**: La interfaz está compuesta de elementos estáticos pero en un número que puede variar a través del uso de la aplicación. Por ejemplo, a lo largo del uso de la misma se crearán informes, cuya cantidad, al contrario que el número de campos de un paciente, es variable.

    Por tanto, estas interfaces tendrán que implementarse en formas que permitan reusar los mismos elementos en múltiples ocasiones.

- **Manejo de eventos**. Se manejará la recepción de eventos del usuario en la interfaz. Se mostrarán botones para que el usuario pueda interactuar con la aplicación, así como otros elementos seleccionables, como campos de escritura de texto.

Esta capa, además, será la encargada de llamar a la capa de servicio, así como de esperar por su respuesta cuando sea necesario.

También tendrá otras funciones, como la adaptación al tamaño de la pantalla en función del tamaño de la tableta, así como abrir pantallas nuevas que den acceso a funcionalidades.

8.3.2 Capa de Servicio

La segunda capa software de la aplicación actuará siempre y cuando reciba una llamada de la Capa de Visión de la Aplicación.

Esta capa estará representada por una *interface* (interfaz) Java, donde se acumularán las llamadas a implementar mientras se desarrolla la primera capa. Hasta que no se implemente por completo la Capa de Visión de la Aplicación, se creará una clase Java, heredada de la *interface*, que actuará como prototipo, ya que devolverá datos no reales, pero suficientes para probar las llamadas que solicite la primera capa.

Esta capa obtiene los datos necesarios de la Capa de Acceso a Base de Datos, y se comunica con ella a través de las clases representativas de las entidades de la Base de Datos.

Se encargará de las siguientes funciones, a petición de la Capa de Visión de la Aplicación:

- Realizar validaciones de datos, antes de ser insertados, obtenidos, modificados o eliminados de Base de datos.
- Realizar la operación de login.
- Obtiene las opciones de personalización de la interfaz gráfica: colores y avatares de usuario.
- A través de operaciones de selección, ordenamiento, eliminación en memoria o acondicionamiento, proporciona datos en forma de modelos a la capa superior.
- Aplica, en caso de ser necesario, cambios en los datos almacenados en Base de Datos por motivo de actualización, comunicándose con la capa inferior.

Al mismo tiempo, será esta capa, mediante el patrón *Singleton*, la que tenga un objeto único de la Base de Datos, para garantizar una única instancia de la misma. También inicializará una única instancia del servicio y de los elementos de la Capa de Acceso a Base de Datos.

8.3.3 Capa de Acceso a Base de Datos

La tercera capa software de la arquitectura se encargará de obtener, mediante llamadas SQL, los datos ”en crudo” almacenados en Base de Datos, es decir, representaciones directas de las entradas de las tablas. Esta capa otorgará dichos datos a la Capa de Servicio, para que luego el servicio se las dé a la Capa de Visión de la Aplicación en forma de modelos.

También proporcionará acceso a los mecanismos propios de SQL:

- Inserción de datos.
- Eliminación.
- Selección.
- Actualización.

8.4 Base de Datos

La Base de Datos es el elemento en el que se almacenarán los datos de la aplicación. La forma en la que lo hace se describe en el diagrama Entidad-Relación de la figura 8.2, en la página 46.

En ella se pueden distinguir las siguientes entidades:

- **Usuario**. Representa a pacientes, tutores, logopedas y administradores.
- **Grupo**. Representa a los grupos de pacientes.
- **Test**. Test creado para evaluar pacientes. Pueden ser de varios tipos, en función del período escolar para el que se crea el test.

Y las siguientes relaciones:

- **Tutor de**. Un tutor puede tener varios pacientes asociados. Un paciente puede tener hasta 2 tutores asociados.
- **Pertenece**. Un paciente pertenece a un grupo, y en un grupo puede haber más de un paciente.
- **Gestiona**. Un logopeda puede gestionar a un grupo, y un grupo tiene que ser gestionado por un logopeda.
- **Implementa**. Un logopeda puede crear tests de un tipo concreto.
- **Evalúa**. Un paciente puede ser evaluado en varios tests, y asociársele, para cada prueba, un Puntuación Directa (PD) y una anotación. Un test puede ser realizado por varios pacientes.

Al transformar este diagrama entidad-relación en un esquema relacional, se indica que la Base de Datos final debe tener las siguientes tablas:

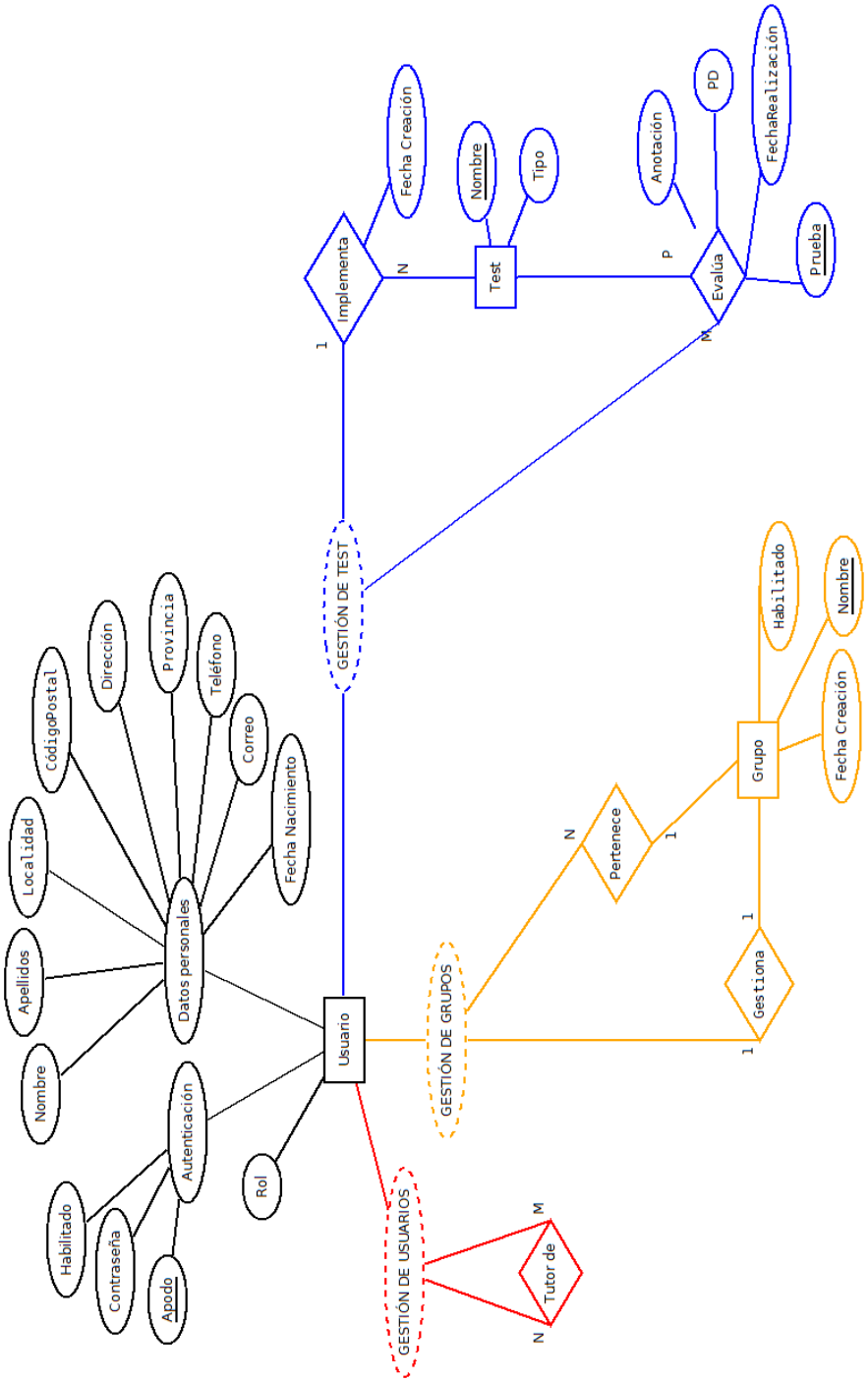


Figura 8.2: Diseño de la Base de Datos.

- **Usuario** (Apodo, Contraseña, Habilitado, Nombre, Apellidos, Localidad, Código Postal, Dirección, Provincia, Correo, Fecha Nacimiento, Color, Avatar, Rol, Grupo).
- **TutorDe** (ApodoUsuarioTutor, ApodoUsuarioTutelado.)
- **Grupo** (NombreGrupo, Habilitado, Fecha Creación, ApodoUsuarioGestor).
- **Test** (NombreTest, ApodoUsuarioCreador, TipoTest, Fecha Creación).
- **Evaluación** (NombreTest, ApodoUsuario, Prueba, PD, Anotación, FechaRealización).

8.5 Diseño de la interfaz gráfica

En este proyecto se diseñan distintos esquemas para mostrar en la Capa de Visión de la Información.

Los esquemas de interfaz son una representación de la forma de disponer al usuario los elementos en la pantalla. Si un esquema es efectivo y se repite a lo largo de la aplicación, el usuario estará familiarizado con las pantallas si conoce su esquema, aunque no las haya visitado todavía.

Este es un listado de las pantallas con las que trabajará el usuario final y sus esquemas concretos:

- Pantalla de login (Esquema de login).
- Pantalla principal (Esquema principal).
- Pantalla de Informes (Esquema LIC).
- Pantalla de Tests (Esquema LIC).
- Pantalla de Grupos (Esquema LIC).
- Pantalla de Pacientes (Esquema LIC).
- Pantalla de Editar Perfil (Esquema de edición).
- Pantalla de Configuración (Esquema LA).
- Pantalla de Administrador (Esquema LA).
- Pantalla de Editar Grupo (Esquema de edición).
- Pantalla de Evaluación (Esquema de evaluación).

A continuación se describen los esquemas diseñados para el proyecto.

8.5.1 Esquema login

Esquema utilizado en la **Pantalla de Login**. Se muestra un amplio espacio sin usar con un color de fondo. En el centro hay un recuadro con los siguientes elementos:

- Logotipo de la aplicación. Se sitúa en la izquierda.
- Nombre de la aplicación. Se ubica debajo del logotipo.
- Campos de textos:
  - Usuario: nombre que usará el usuario final para entrar en la aplicación.
  - Contraseña: clave que el usuario conoce y que usará para entrar en la aplicación.
- Botón de Login: Valida las credenciales y da acceso a la aplicación.

8.5.2 Esquema principal

Este esquema se usará en la **Pantalla Principal**. Su función debe ser la de proporcionar accesibilidad al usuario a la mayor cantidad de funcionalidades de la aplicación, sin saturar de elementos la pantalla. Está compuesto por los siguientes elementos:

- Los siguientes elementos se muestran a la izquierda de la pantalla.
  - Botón Infomes. Da accesibilidad a la **Pantalla de Informes**.
  - Botón Tests. Da accesibilidad a la **Pantalla de Tests**. Sólo para logopedas.
  - Botón Grupos. Da accesibilidad a la **Pantalla de Grupos**. Sólo para logopedas.
  - Botón Pacientes. Da accesibilidad a la **Pantalla de Pacientes**. Sólo para logopedas.
- En el centro de la pantalla se muestra una lista de los últimos 20 informes. De esta forma el usuario siempre tiene un acceso rápido a esos informes, que el desarrollador estima que serán los más usados.
- A la derecha se muestra un botón de mayores dimensiones que el resto, que llevará al módulo de evaluación. Se pedirá al usuario un test y un paciente para la evaluación. Una vez que se han elegido, el usuario irá a la **Pantalla de Evaluación** para realizar el test al paciente. Sólo para logopedas.

También tendrá un menú en la esquina superior derecha, que dará acceso a las funcionalidades de **Editar Perfil**, **Configuración** y **Administrador** (este último sólo para logopedas).

8.5.3 Esquema LIC

El esquema Listado - Información - Creación (LIC) es uno muy sencillo y concreto. El objetivo es que, en las pantallas en las que se implemente, se proporcionen los siguientes elementos, para un objeto genérico:

- **Listado** de objetos existentes en la aplicación. Se sitúa a la izquierda de la pantalla.
- **Botón de Creación**. Este botón tiene la capacidad de crear un objeto nuevo en la aplicación.
- **Espacio de información**. Espacio libre para su uso, tanto para la visualización de un objeto del listado, si éste se pulsa, como para la creación del objeto, si se pulsa el botón de Creación.

Este esquema se usa en las siguientes pantallas:

- **Pantalla de Tests**.
- **Pantalla de Grupos**.
- **Pantalla de Pacientes**.
- **Pantalla de Informes**, sin opción de creación, sólo listado e información. Además se añade la opción en el espacio de información de exportar el informe a PDF mediante un botón.

8.5.4 Esquema de Edición

Este esquema es similar al anterior, en el sentido de que son capaces de mostrar información, sólo que, en este caso, no se muestra opción de creación ni un listado, sino sólo la información concreta de un objeto.

Además, abajo en el centro, habrá un botón Editar que, si se pulsa, convertirá la información de la pantalla en editable y mostrará dos nuevos botones: Cancelar y Guardar.

Este esquema se usa en la **Pantalla de Editar Perfil** y **Editar Grupo**.

8.5.5 Esquema de Evaluación

Este esquema tendrá un formato de formulario y se usará en la **Pantalla de Evaluación**. En primer lugar, se muestran en lo más alto de la pantalla los siguientes campos:

- **Test** que se realiza, tanto su nombre como su tipo.
- **Paciente** al que se le realiza el test.

Para cada subtest de una evaluación, se reciclarán los mismos campos de información del formulario. Toda la información que se mostrará a continuación se extraerá del test oficial [2]. Dichos campos son los siguientes:

- **Prueba**, primera jerarquía dentro del test. Un test tiene pruebas.
- **Subtest**. Para cada prueba, hay subtests que, en su conjunto, forman la prueba.
- **Material**. En caso de ser necesario, se mostrarán los materiales para el subtest.
- **Instrucciones**. Pasos que sirven como guía al logopeda para realizar la prueba.
- **Anotación**. Campo de texto editable donde el usuario logopeda puede introducir y almacenar sus notas acerca de cómo el paciente está realizando la prueba. Por ejemplo, se le vé distraído, falló en un número, u otras informaciones que los logopedas consideren necesarias.

  Comenzará siendo inicializado con los posibles resultados de la prueba, para que el logopeda no tenga que escribir todo desde cero.
- **Criterio de éxito**. Campo de texto no editable con las instrucciones para que el logopeda evalúe la puntuación que recibe el paciente en el subtest.
- **Puntuación**. Campo de texto editable en el que el logopeda, acorde al criterio de éxito, puntuará al paciente en el subtest. No puede tener valores fuera del rango.

Por último, habrá dos botones en la parte inferior de la pantalla:

- **Atrás**. Llevará al subtest anterior y se podrá repetir, en caso de que el logopeda lo considere necesario.
- **Siguiente**. Si se pulsa y se ha puntuado el subtest correctamente, se pasará al siguiente. En la última prueba debe tener el texto **Finalizar y Guardar**.

8.5.6 Esquema LA

El esquema Listado - Acción (LA) tiene dos componentes fundamentales:

- **Listado de botones**, presente a la izquierda de la pantalla
- **Espacio de acción**, que ocupa el resto de la pantalla.

Lo que se busca con este esquema es que, por un lado, los botones de la izquierda den acceso a funcionalidades y, por otra, que, si se pulsa uno de los botones, el espacio de acción se usará para ejecutar la acción que busca ese botón.

Este esquema se usa en las siguientes pantallas:

- **Pantalla de Configuración**. Tendrá el siguiente botón en el listado:
  - **Cambiar contraseña**, que si se pulsa mostrará un formulario para cambiar la contraseña en el espacio de acción.
- **Pantalla de Administración**. Tendrá el siguiente listado de botones:
  - **Gestión Usuarios**, que si se pulsa mostrará lo siguiente en el espacio de acción:
    - Una Lista de los Usuarios (pacientes, tutores y logopedas), cada uno de ellos con información significativa: apodo, rol y nombre y apellidos. Además, para cada usuario, habrá hasta tres botones:
      - **Editar Usuario**, que llevará, si se pulsa, a la pantalla de Editar Perfil (Esquema de Edición), pero en este caso con los datos del usuario seleccionado.
      - **Eliminar Usuario**, que hace que el usuario pierda el acceso a la aplicación y sus datos no aparezcan mientras siga así.
      - **Restablecer Contraseña**, que permitirá que el usuario final pueda cambiar la contraseña a otros usuarios en el caso de que se les haya olvidado. Estará presente en los usuarios con rol de tutor o logopeda.
    - Botón de **Añadir Logopeda**, localizado en la esquina inferior derecha de la aplicación. Si se pulsa, se abrirá un formulario similar al de "Añadir Paciente", parte de la pantalla de pacientes (Esquema LIC). Permitirá añadir logopedas nuevos a la aplicación.
  - **Gestión de Grupos**. En este caso, si se pulsa, se mostrará el siguiente contenido en el espacio de acción:
    - **Listado de Grupos**. Este listado contendrá a todos los grupos de pacientes de la aplicación. Para cada grupo, aparecerá su nombre y su logopeda gestor. Además, habrá dos botones:
      - **Editar grupo**. Esto llevará a la pantalla Editar Grupo (Esquema de Edición), donde se podrán cambiar tanto los pacientes como el logopeda gestor.
      - **Eliminar grupo**. Si se pulsa, eliminará el grupo de la aplicación. Los pacientes quedarán disponibles para ser añadidos a otro grupo y el logopeda gestor podrá gestionar otro grupo.
  - **Gestión de Datos**. Motivado por la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales [32], se añade esta opción. En este apartado, si se pulsa el botón, se muestra lo siguiente en el espacio de acción:

    * **Listado de usuarios eliminados**. Para cada usuario eliminado, se muestra su nombre y apellidos. También se muestran dos botones:
        · **Restablecer Usuario**. Si se pulsa este botón, se recuperará el usuario que se eliminó como si nunca se hubiera hecho.
        · **Eliminar Datos**. Con este botón se eliminarán todos los datos y todo registro de la aplicación en el que haya participado el usuario (grupos, evaluaciones, entre otros).

8.6 Conclusiones

En este capítulo se han podido ver las decisiones de diseño a tres niveles:

- **Hardware**. Uso de tableta Android en Versión 6.0, que interactúa con la primera capa del siguiente nivel.
- **Arquitectura Software de la Aplicación**. Diseño de tres capas que se comunican entre sí y con los niveles inferior y superior.
  - **Capa de Visión de la Aplicación**. Se encarga de lanzar los elementos visibles y con los que puede interactuar el usuario. También captura los eventos del usuario final y acude a la siguiente capa, si es necesario.
  - **Capa de Servicio**. Capa interna, donde se hacen operaciones de varios tipos, pero que no interactúa con ningún otro nivel.
  - **Capa de Acceso a Base de Datos**. Capa encargada de realizar operaciones de selección, inserción, eliminación y modificación de datos en la Base de Datos.
- **Base de Datos**. Elemento encargado, de usa forma definida en los diseños, de hacer que los datos sean persistentes en las ejecuciones de la aplicación.

Este diseño tiene la ventaja de que, tanto los niveles como las capas internas de la arquitectura software, son reemplazables individualmente por otras versiones de las mismas, siempre y cuando mantengan las mismas operaciones declaradas en su *interface*, en el caso de las capas software, mantengan sus estructuras, en el caso de la Base de Datos, y el hardware incluya una tableta de versión mayor o igual a Android 6.0.

En el proceso, se han diseñado 11 pantallas, abstraídas a 6 esquemas idénticos con los que el usuario se familiarizará con el uso. Esto implica que hay una reducción de casi la mitad (45%) de esquemas de visualización e interacción con la pantalla del dispositivo, acelerando el proceso de aprendizaje del usuario con el producto.

Por último, estos diseños cumplen con los requisitos de primer nivel descritos en el capítulo anterior. Esto implica que si se implementan en un producto, en primera instancia, se puede afirmar que el producto tiene la calidad suficiente como para ser una aplicación funcional acorde a lo que se demanda en el proyecto.

Capítulo 9

Implementación

En este capítulo se describe el proceso de la implementación del proyecto.

9.1 Contexto

Antes de comenzar a describir la implementación, se realizaron las tareas descritas en los capítulos anteriores, y todo ello ha proporcionado a esta fase del proyecto un amplio contexto sobre el que trabajar. Todas y cada una de dichas tareas son formas de fortalecer, de un modo u otro, el proceso de implementación.

En este sentido, conviene destacar los siguientes aspectos:

- Proceso de lectura de bibliografía acerca de la discalculia o DAM, con el objetivo de familiarizarse con la temática del proyecto, así como la lectura del test TEDI-MATH [2] que se implementa.
- Estudio del estado del arte, para ver en qué puede destacar esta aplicación donde otras no lo están haciendo.
- Fundamentos tecnológicos, para facilitar el desarrollo del proyecto en todas sus fases.
- Metodología de trabajo, que define claramente cómo funcionará esta fase.
- Planificación y costes, que proporciona una visión final de la calidad de la implementación en función de las desviaciones entre los cálculos estimados y los reales.
- Análisis de requisitos, en el que se priorizaron unas funcionalidades para implementarlas antes que otras.
- Unos diseños claros que sirven de guía para la implementación de la aplicación.

En consecuencia, de cada una de estas fases se obtiene algo provechoso para el proceso de implementación.

9.2 Aspectos destacables de la implementación

A continuación se detallarán algunos de los aspectos importantes de la implementación.

9.2.1 Modelos

A lo largo de las iteraciones se implementaron lo que se conoce como modelos. Un modelo es una clase que contiene información de lo que representa en el mundo. Por ejemplo, el modelo ”Usuario” tiene varios campos: nombre, apellidos o localidad, entre otros.

Estas clases pueden ser convertidas en objetos, en un proceso de inicialización. Estos objetos son los que se manejan dentro de la Capa de Visión de la Aplicación, y es la forma que tiene de comunicarse con la Capa de Servicio.

Esta es la lista de modelos que se implementaron:

- **Usuario**: Puede ser paciente, tutor o logopeda.
- **Grupo**: Grupo de pacientes con un logopeda gestor.
- **TestAbstracto**: Nivel de test TEDI-MATH [2].
- **Test**: Test concreto, con un test abstracto asociado que implementa.
- **Resultado**: Resultado de una evaluación. Tiene un Test, un usuario (el paciente), sus puntuaciones en las pruebas y las anotaciones realizadas por el logopeda que le evaluó.
- **Histórico**: Resultado de una evaluación en una fecha concreta.

9.2.2 Actividades

De acuerdo con la documentación oficial de Android Developers, ”la clase Activity es un componente clave de una app para Android, y la forma en que se inician y se crean las actividades es una parte fundamental del modelo de aplicación de la plataforma. A diferencia de los paradigmas de programación en los que las apps se inician con un método main(), el sistema Android inicia el código en una instancia de Activity invocando métodos de devolución de llamada específicos que corresponden a etapas específicas de su ciclo de vida.” [33]. En lo que respecta a este proyecto, de la definición dada, se destaca que se invocan métodos específicos en ciertos puntos concretos del ciclo de vida de un objeto Actividad.

Uno de los métodos más importantes es el *onCreate()*, de obligada implementación. En la implementación del método en las distintas Actividades de la aplicación, en primer lugar, se obtienen los datos necesarios del servicio y del **intent**, si lo hubiera. Los **intents** son intentos de crear un objeto de esta clase. Más adelante, se inicializan todos los componentes de la parte gráfica de la aplicación, con el método *setContentView()*. Este método acepta como argumento

un fichero XML, donde se definen los elementos de la pantalla así como su posición. Además, se realizan las tareas programáticas que no se realizan en el fichero XML, pero sí están relacionadas con la interfaz gráfica.

Entre estas tareas se destaca la de hacer que los botones, u otros elementos, presentes en la pantalla actual, envíen sus eventos, de ser pulsados, a las actividades, ya que éstas se definen implementando la *interface View.OnClickListener*, es decir, escuchadora de eventos de click. Al inicializar las variables de los elementos que el usuario final puede pulsar, se llama, en cada una de esas variables, al método *setOnClickListener()*, aceptando como argumento la actividad, es decir, el objeto de la clase que implementa la *interface View.OnClickListener*.

Las actividades de la aplicación, aunque no son las únicas que participan en la interfaz gráfica, sí son las únicas que se encargan de las comunicaciones con la Capa de Servicio. De esta forma, se centralizan las llamadas al servicio en menos clases, en vez de tenerlas más esparcidas, lo que ayuda al control del código.

Por último, esta es la lista de las Actividades que se implementaron:

- **EvaluadorActivity**, para el módulo de evaluación.
- **LoginActivity**, para la pantalla de login.
- **MainActivity**, para la pantalla principal.

9.2.3 Fragmentos

Citando de nuevo la documentación oficial [34], ”un Fragment representa un comportamiento o una parte de la interfaz de usuario en una FragmentActivity. Puedes combinar varios fragmentos en una sola actividad para crear una IU multipanel y volver a usar un fragmento en diferentes actividades. Puedes pensar en un fragmento como una sección modular de una actividad que tiene un ciclo de vida propio, que recibe sus propios eventos de entrada y que puedes agregar o quitar mientras la actividad se esté ejecutando (algo así como una ”subactividad” que puedes volver a usar en diferentes actividades)”. De forma similar a las actividades, los fragmentos también tienen un ciclo de vida.

En el proyecto, las actividades crean los fragmentos para sustituir ”trozos” de pantalla por la pantalla que representa el fragmento.

En un caso práctico, desde la pantalla principal, que es una actividad, se inicializa el botón Tests. Al inicializarse, se llama al método *setOnClickListener()* del botón, pasando como parámetro la propia actividad. En la actividad, se implementa el método *onClick()*, que se lanza cuando el usuario pulsa un elemento, y que tiene como argumento el componente de la pantalla que fue pulsado. Con una selección comparativa para saber qué elemento fue pulsado, se detecta que el elemento pulsado fue el botón Tests. Entonces, se ejecuta un código que crea, en

primer lugar, el objeto fragmento de la clase TestsFragment, que hereda de la clase Fragment. Por último, indica el espacio que debe usar ese fragmento, en este caso, toda la pantalla.

Esta metodología se repite para todas los procesos de creación de fragmentos que ocurren en la aplicación.

Por último, al inicializarse los fragmentos, se le pasa por parámetro un objeto que implementa la *interface View.OnClickListener*, es decir, la actividad propia que lo creó. De esta forma, de forma similar a las actividades, los fragmentos tienen un método *onCreateView()*, en el que se seleccionará el fichero XML que tiene la información de los elementos de la pantalla y de su localización.

Más adelante, en el método *onActivityCreated()*, otro método del ciclo de vida de los fragmentos, se inicializan los elementos gráficos, como botones y textos y, para aquellos que puedan ser pulsados, se llama a su método *setOnClickListener()* pasando como parámetro la actividad.

Esta es la lista de clases fragmento en el proyecto:

- **AbstractTFGDiscalculiaFragment**. Esta es una clase abstracta que hereda de Fragment [34], y que reúne características comunes a otros fragmentos para que hereden de ella, como la referencia a la actividad.
- **AdministrarFragment**, para la pantalla de administrador.
- **ConfiguracionFragment**, para la acción de cambiar la contraseña.
- **EditarPerfilFragment**, usada para la pantalla de editar perfil.
- **GruposFragment**, para la representación de la pantalla de grupos.
- **InformesFragment**, para ver la lista de informes.
- **PacientesFragment**, para la lista de pacientes.
- **TestsFragment**, encargada de la visualización de los tests.
- **AñadirLogopedaFragment**, con el formulario para crear el logopeda.
- **AñadirPacienteFragment**, similar al anterior pero con pacientes y sus tutores asociados.
- **CrearGrupoFragment**, otro formulario pero para crear los grupos.
- **CrearTestsFragment**, en el que se insertan los tests en la aplicación en base al nivel deseado.
- **EditarGrupoFragment**, usado para poder editar un grupo después de ser creado.

- **GestionarDatosFragment**, para la característica de administrar el uso de datos personales.
- **GestionarUsuariosFragment**, que implementa la administración de usuarios en la aplicación.
- **GraficaFragment**, donde se ve, además de una gráfica muy visual de los resultados de la evaluación, también se pueden ver todas y cada una de las pruebas realizadas, así como sus puntuaciones y las anotaciones del logopeda.
- **InfoGrupoFragment**, para visualizar la información de un grupo.
- **InfoPacienteFragment**, para ver los datos de un paciente y sus tutores.
- **InfoTestsFragment**, donde se pueden ver no sólo los datos de un test, sino también cuántos pacientes y grupos han realizado la evaluación correspondiente.

9.2.4 Solicitud de permisos

Las aplicaciones Android se ejecutan en una zona del sistema dedicado a cada una de ellas, con un acceso limitado al resto del dispositivo. Para obtener acceso a otras partes del dispositivo, hay que realizar una serie de pasos [35].

El primero de ellos es declarar que la aplicación puede que use, en algún lugar del código, un acceso a un recurso. En el caso de este proyecto, se requieren permisos para exportar informes a PDF y colocarlos en la carpeta de descargas (Downloads). Para ello, hay un fichero XML dentro de la aplicación llamado AndroidManifest.xml, que contiene una gran cantidad de metadatos de la misma.

En el manifiesto, se añaden dos lineas para declarar que la aplicación usa los permisos de lectura y escritura en la memoria externa:

```
1
2  <?xml version="1.0" encoding="utf-8"?>
3  <manifest package="es.udc.tfg2021.bluepinetesting"
4      xmlns:android="http://schemas.android.com/apk/res/android">
5
6      ...
7
8      <uses-permission
       android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
9      <uses-permission
       android:name="android.permission.READ_EXTERNAL_STORAGE" />
10
11     ...
12
```

```
</manifest>
```

Más adelante, en el código que se invoca cuando el usuario quiere exportar el fichero PDF, se añadirá una consulta para saber si a la aplicación se le han concedido los permisos necesarios. En el caso del proyecto, se examina si tiene dichos permisos y, si no los tiene, los solicita:

```

public void exportPDFFromView(){

        if (actividad.checkSelfPermission
        (android.Manifest.permission.WRITE_EXTERNAL_STORAGE) !=
    PackageManager.PERMISSION_GRANTED) {


    Snackbar.make(actividad.findViewById(R.id.actividad_principal),
                    getString(R.string.permiso_denegado),
    BaseTransientBottomBar.LENGTH_LONG).show();
            ActivityCompat
                    .requestPermissions(actividad, new
    String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}, 23);
        } else {

            crearPDF();
        }
    }
```

Al ejecutar este bloque de código *if*, al usuario se le muestra un diálogo desde el sistema operativo que le notifica que esta aplicación le está solicitando permisos, como se muestra en la figura 9.1, en la página 68.

Por último, una vez que el usuario ha otorgado permisos a la aplicación, puede ejecutar de nuevo el código anterior, con la diferencia de que ahora sí irá al bloque *else*, donde se crea el PDF y se almacena en el directorio de descargas.

9.2.5 DAOs

Data Access Objet (DAO) es un patrón cuya función principal es la de aislar los elementos de la Base de Datos, o cualquier otra capa de persistencia, del resto de capas de la aplicación. Son los elementos principales de la Capa de Acceso a la Base de Datos. Estos objetos en el proyecto son *interfaces* definidas con la librería Room [18], cuyos métodos tendrán las anotaciones necesarias para servir de relación con las sentencias de la Base de Datos.

Cada DAO en el proyecto tiene, por lo menos, cuatro elementos que afectan a la Base de Datos:

- **Insert**: Inserción de elementos.
- **Update**: Actualización de los campos de una entrada.
- **Delete**: Eliminación de una única entrada.
- **Queries**: Consultas de los datos.

Por ejemplo, este es el código de la *interface* TestDao:

```
@Dao
public interface TestDao {
    @Insert(onConflict = OnConflictStrategy.ABORT)
    void insertTest(Test test);

    @Delete
    void deleteTest(Test test);

    @Update
    void updateTest(Test test);

    @Query("SELECT * FROM Test")
    Test[] getAllTest();

    @Query("SELECT * FROM Test WHERE nombreTest = :nombreTest")
    Test getTest(String nombreTest);

    @Query("SELECT * FROM Test WHERE apodo = :apodo")
    Test[] getTestByApodo(String apodo);

}
```

Como se mencionaba anteriormente, si se fija la atención primero en el nombre de la *interface*, se observa que tiene la etiqueta ”@Dao” encima de la declaración del elemento. Al mismo tiempo, para cada método, hay una etiqueta, afín a los elementos mencionados anteriormente: ”@Insert”, ”@Delete”, ”@Update” y ”@Query”.

Por último, cabe destacar una opción de la etiqueta ”@Insert”, *(onConflict = OnConflictStrategy.ABORT)*. Esto quiere decir que, al llamar al método de inserción, si ya existe en la tabla Test un elemento con la clave primaria que se quiere insertar, se aborta el proceso, en vez de sobrescribir la entrada.

Esta es la lista de cada DAO que se implementó y su función:

- **EvaluacionDao**, para almacenar los resultados de las evaluaciones.
- **GrupoDao**, donde se interactúa con los Grupos.

- **TestDao**, para tener registro de los tests creados.
- **TutorDeDao**, donde se guarda la relación entre los pacientes y sus tutores.
- **UsuarioDao**, para interactuar con la tabla Usuarios.

9.2.6 Base de Datos

De acuerdo con los diseños del proyecto, se imlementó la Base de Datos con Room [18], una abstracción de SQLite implementada en Android. En ella, se destacan tres elementos fundamentales:

- **Entidades**: Representaciones ya en código de los diagramas definidos en el capítulo anterior (ver figura 8.2, en la página 46). Por ejemplo, la entidad Test:

```
1
2     @Entity
3     public class Test {
4         @NonNull
5         @PrimaryKey
6         public String nombreTest;
7
8         @ColumnInfo(name = "apodo")
9         public String apodo;
10
11        @ColumnInfo(name = "test_abstracto")
12        public String testAbstracto;
13
14        @ColumnInfo(name = "fecha_creacion")
15        public Date fecha_creacion;
16
17        public Test(@NonNull String nombreTest,
18                    String apodo,
19                    String testAbstracto,
20                    Date fecha_creacion) {
21
22            this.nombreTest = nombreTest;
23            this.apodo = apodo;
24            this.testAbstracto = testAbstracto;
25            this.fecha_creacion = fecha_creacion;
26        }
27    }
```

Como los DAO, tienen varias anotaciones: @Entity, para señalar que es una entidad; @NonNull y @PrimaryKey, para señalar que son campos que no pueden ser nulos y son

claves primarias; y @ColumnInfo, con el nombre de la columna de la tabla. También se añadió un constructor, para hacer más sencillo el proceso de inserción.

Existes cinco entidades en el proyecto:

- **Evaluación**, para los registros de evaluaciones.
- **Grupo**, con los datos de los grupos.
- **Test**, para la persistencia de los tests.
- **TutorDe**, que representa la relación N-M entre los pacientes con los tutores, ambos en la tabla Usuarios.
- **Usuario**, con los datos de los usuarios.

- **Relaciones**: Relaciones entre las entidades. Esto sirve para mantener la integridad de los datos y que no se violen las restricciones de clave foránea creadas. Por ejemplo, la relación Test-Usuarios (implementa):

```
1
2     public class UsuarioConTests {
3     @Embedded
4     public Usuario usuario;
5     @Relation(
6             parentColumn = "apodo",
7             entityColumn = "apodo"
8     )
9     public List<Test> tests;
10    }
11
12
```

Como se puede observar, existe la etiqueta @Relation, en la que se indica la relación 1-N entre las entidades Usuario y Test, además de las columnas que son referencias una de la otra.

- **Base de Datos**: La Base de Datos viene representada por una clase abstracta. Con ella se instanciará la Base de Datos en el código y se declararán los DAO. Su código fuente es el siguiente:

```
1
2     @Database(entities = {Usuario.class, TutorDe.class, Test.class,
      Grupo.class, Evaluacion.class}, version = 3, exportSchema =
      false)
3     public abstract class Bluepinedb extends RoomDatabase {
4         public abstract UsuarioDao UsuarioDao();
5         public abstract GrupoDao GrupoDao();
```

```
6           public abstract TestDao TestDao();
7           public abstract TutorDeDao TutorDeDao();
8           public abstract EvaluacionDao EvaluacionDao();
9
10      }
11
12
```

9.2.7 Patrón Singleton

Por último, el patrón *Singleton* sirve para tener instancias únicas de objetos de distintas clases. En este proyecto se usa para mantener una única instancia de los siguientes objetos:

- Servicio.
- DAO.
- Base de datos.

Todo esto se realiza en el siguiente bloque de código de la clase ServicioTestingImp:

```
1
2  public class ServicioTestingImp implements ServicioTesting {
3
4      ...
5
6      //Patrón Singleton
7      private static ServicioTestingImp sServicioTestingImp;
8      private EvaluacionDao mEvaluacionDao;
9      private GrupoDao mGrupoDao;
10     private TestDao mTestDao;
11     private TutorDeDao mTutorDeDao;
12     private UsuarioDao mUsuarioDao;
13
14     private ServicioTestingImp(Context context){
15         Context appContext = context.getApplicationContext();
16         Bluepinedb database = Room.databaseBuilder(appContext,
       Bluepinedb.class, "database")
17                 .allowMainThreadQueries().build();
18         mEvaluacionDao = database.EvaluacionDao();
19         mGrupoDao = database.GrupoDao();
20         mTestDao = database.TestDao();
21         mTutorDeDao = database.TutorDeDao();
22         mUsuarioDao = database.UsuarioDao();
23     }
24
```

```
25      public static ServicioTestingImp get(Context context){
26          if (sServicioTestingImp == null){
27              sServicioTestingImp = new ServicioTestingImp(context);
28          }
29          return sServicioTestingImp;
30      }
31
32      ...
33  }
```

9.3 Implementación en iteraciones

Esta primera parte de la implementación está constituida por iteraciones donde se realizaron tareas concretas y definidas.

Se realizaron las iteraciones en este orden:

1. Pantalla Principal.
2. Pantalla Informes.
3. Pantalla Test.
4. Pantalla Pacientes.
5. Pantalla Grupos.
6. Pantalla Editar Perfil.
7. Separación por roles a los accesos de la Pantalla Principal.
8. Pantalla Administración.
9. Pantalla Gestión de Usuarios.
10. Pantalla Gestión de Grupos.
11. Pantalla Gestión de Datos.
12. Pantalla Login.
13. Módulo de evaluación.
14. Pantalla Informes: integrar resultados de evaluaciones con la pantalla informes existentes, así como el módulo de Gráficas.

15. Implementación de la Base de Datos.
16. Implementación de los DAO.
17. Implementación del Servicio.
18. Elaboración PDF con Informe.

Se priorizan las iteraciones que implementan la Capa de Visión de la Aplicación, para tener siempre un resultado visible. Al mismo tiempo, se diseñaban los métodos que iban a necesitar de la Capa de Servicio.

Más adelante, se investigan las posibles formas de implementar la Capa de Acceso a Base de Datos, así como la Base de Datos. Tras esto, se implementan estos dos elementos.

Por último, se implementa la Capa de Servicio, sustituyendo el prototipo que se tenía desde la implementación de la Capa de Visión de la Aplicación por uno que devolviera datos obtenidos a través de la Capa de Acceso a Base de datos recién implementada. También se dejó para el final la generación del fichero PDF con el informe y la solicitud de permisos, porque no había ninguna ventaja con su realización en etapas más tempranas.

9.4 Implementación en incrementos

Al acabar la fase de implementación por iteraciones, se marca como primera versión de la aplicación la versión 0.1.0. Es inferior a la versión 1.0.0, ya que considera que está en fase de pruebas.

Inicialmente, se esperaba que esta fase sirviera para añadir los requisitos secundarios y terciarios, así como para corregir errores. Sin embargo, se considera que la aplicación ya era de suficiente calidad, por la cantidad de requisitos que se habían implementado, como para ser ya un producto final, una vez que los errores detectados hubieran sido corregidos.

Por tanto, la lista de incrementos y sus versiones correspondientes fueron:

1. v0.1.1. Se invierte el orden en el que se recuperan los informes de Base de Datos.
2. v0.2.0. Se cambia el formato del PDF del informe.
3. v0.2.1. Se corrige el error de congelación de la aplicación por devolver informes de usuarios eliminados.
4. v1.0.0. Ahora es obligatorio rellenar menos campos en los datos de los usuarios. Se corrigen errores.
5. v1.0.1. Se realizan pequeñas modificaciones y se corrigen errores.

6. v1.0.2. Se realizan pequeñas modificaciones y se corrigen errores.
7. v1.0.3. Se corrigen errores.
8. v1.1.0. Se modifica la pantalla de login para adaptarla a tabletas de menores dimensiones. Se corrigen errores.

En la figura 9.2 se muestra un ejemplo de cómo se etiquetaron dichas versiones en GitLab [21].

9.5 Conclusiones

Con esta implementación, podemos hacer un seguimiento de cómo el trabajo realizado cumple con el diseño realizado para la arquitectura de la aplicación, comparando dicho diseño con los objetos y cómo se comunican. En la figura 9.3, se muestra ese proceso:

- El dispositivo tablet será una tableta Android con versión 6.0 o mayor.
- Los Fragmentos y Actividades se comunican con la tableta a través de la presentación de pantallas formadas en ficheros de formato XML.
- Las Actividades realizan la funcionalidad de manejar los eventos de pulsación del usuario final en la pantalla.
- Las Actividades se comunican con el Servicio a través de Modelos. Estos modelos se usan también para la representación de información en Fragmentos y Actividades.
- El Servicio se comunica con los DAO a través de objetos Entidades. Dichas entidades son representaciones de las tablas de Base de Datos.
- Los DAO también se comunican con la Base de Datos mediante objetos Entidades.
- La Base de Datos mantiene la persistencia de los datos en forma de tablas.

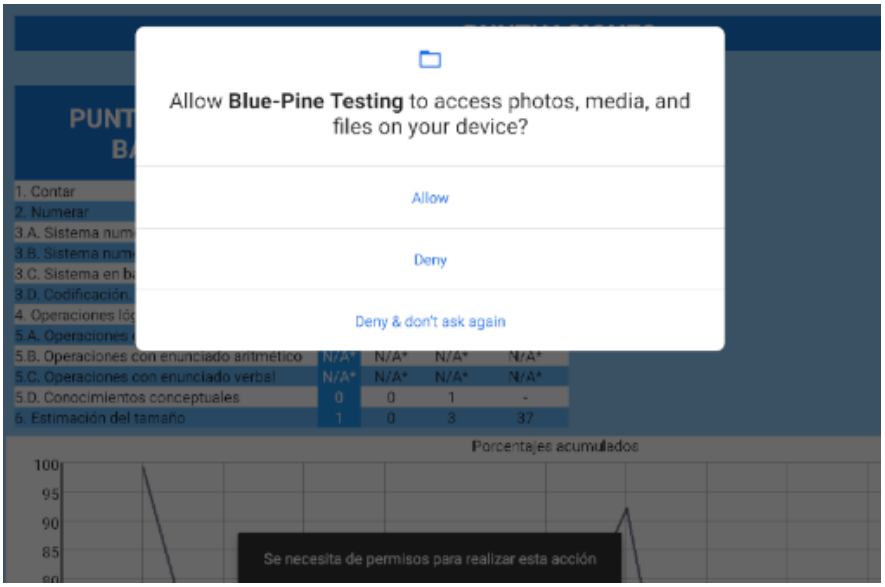


Figura 9.1: Solicitud de permisos.

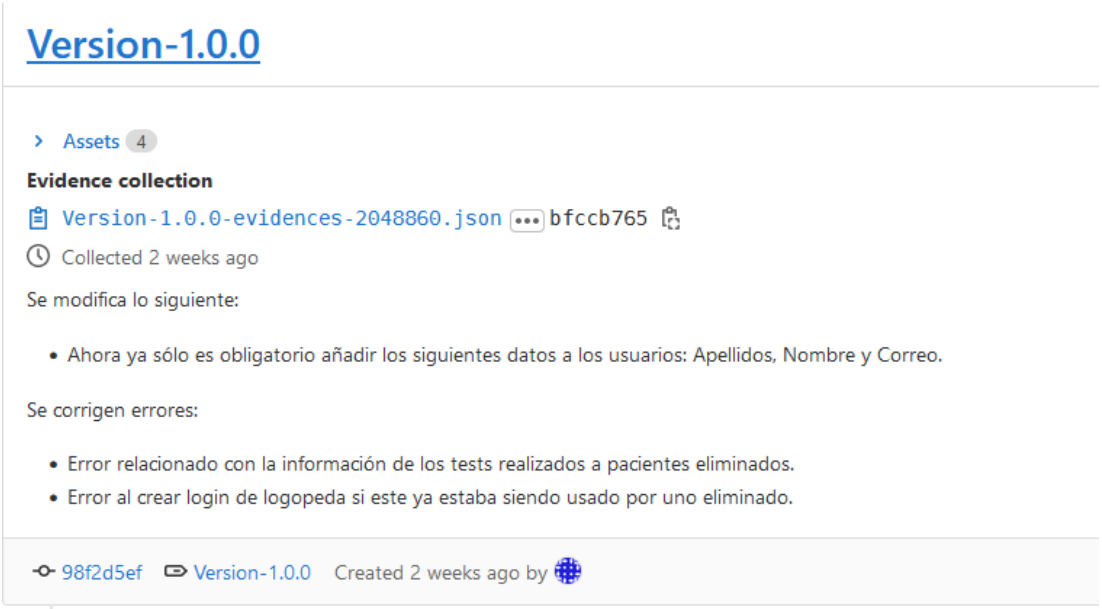


Figura 9.2: Ejemplo de versionado en GitLab.

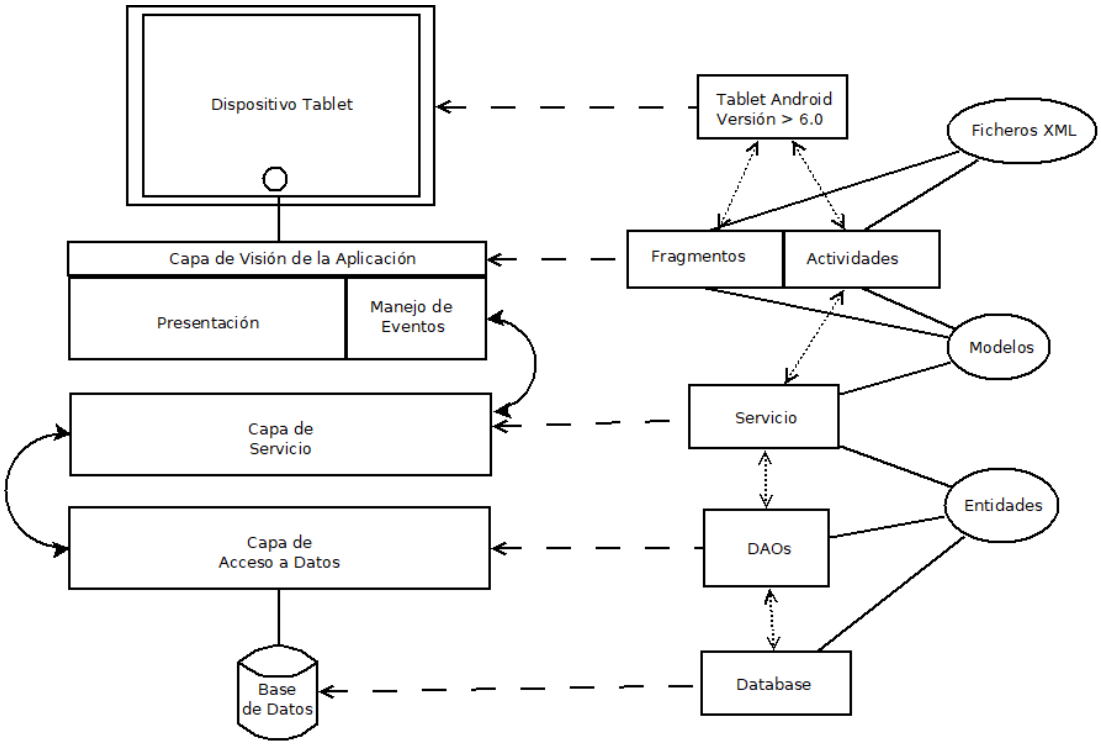


Figura 9.3: Implementación del diseño de la arquitectura de la aplicación.

Capítulo 10

Depuración y pruebas

En este capítulo se comentarán las pruebas realizadas para la depuración del producto desarrollado.

10.1 Pruebas continuas

En cada iteración, se realizaron pruebas de los elementos implementados en la iteración, ya fuera con datos falsos en prototipo o reales. Dichas pruebas se realizaron en el emulador de Android Studio [36], con una imagen de una tableta Nexus 10 con Android en versión 6.0.

Se distinguen tres tipos de pruebas:

- De caja negra, en las que se prueba que el usuario final interactúa correctamente con los elementos de la pantalla. También se evalúa si, para cada acción, la aplicación muestra un mensaje para el usuario confirmándola o comunicándole que no se puede realizar dicha acción y el motivo. De esta forma se prueba que haya una interfaz interactiva.
- De integración, donde se observa si los cambios realizados en la iteración afectan a los productos de las anteriores, si tienen relación entre sí.
- De rendimiento, donde se evalúa que la interfaz responde rápidamente a los eventos del usuario con la pantalla.

En las últimas iteraciones las pruebas de integración tomaron gran relevancia, ya que se implementa la Capa de Servicio, comunicando todos los elementos de la arquitectura.

10.2 Pruebas finales

De cara a la versión final de la aplicación, se prueba el funcionamiento en dispositivos reales. En este sentido, se prueba la aplicación en dispositivos que varían entre Android 6.0 (octubre de 2015) y Android 11 (junio de 2020).

Una de las pruebas de mayor relevancia fue la de rendimiento en una tableta Aquaris M10 con Android 6.0. En general, los resultados fueron positivos, salvo por la función de rellenar el informe en la pantalla. Este es un proceso lento, ya que un informe tiene una gran cantidad de elementos para su inicialización, debido a la completitud del test de evaluación.

Sin embargo, el usuario recibe correctamente un mensaje de ”Cargando... Por favor, espere.”, notificándole que se está ejecutando la aplicación y no está congelada. Dura unos pocos segundos y, al no ser una acción que el usuario haga rápidamente, no se le da mayor importancia.

Las otras pruebas en importancia fueron las de exportar el fichero PDF con un informe a la memoria externa, ya que la forma en la que Android solicita permisos variaba entre las versiones de Android mencionadas anteriormente, provocando fallos según las distintas versiones. Esto sí se consideró un error y fue corregido para todas las versiones de Android para las que está diseñada la aplicación.

Por último, conviene destacar que, en estos momentos, los profesionales de la entidad ASPERGA han iniciado las pruebas del producto final entregado y, una vez validado, se realizarán las primeras pruebas con los usuarios objetivos que la entidad había identificado al establecer las necesidades de la aplicación a desarrollar por el estudiante.

Capítulo 11

Conclusiones finales

En este capítulo se van a explorar las conclusiones finales relativas a la realización del proyecto.

El producto final, al que se denominó **Blue-Pine Testing**, es una aplicación de Android para tablets que permite realizar el test TEDI-MATH [2] de diagnóstico para la discalculia o DAM, a lo largo del proceso educativo que se desarrolla entre la escolarización de segundo de Educación Infantil y tercero de Educación Primaria, es decir, entre 4 y 8 años. Al mismo tiempo, permite realizar un seguimiento del histórico de los pacientes a lo largo del tiempo, observando las puntuaciones y anotaciones realizadas por logopedas en informes de evaluaciones ya realizadas. Cada logopeda, dentro de la aplicación, puede gestionar grupos de pacientes. También se pueden realizar otras tareas, como la gestión de los usuarios de la aplicación, así como la gestión de datos.

En las figuras de este capítulo se puede hacer una comparativa entre el informe generado por la aplicación y el informe original. Se destaca lo siguiente:

- La eliminación del consumo de papel con la aplicación desarrollada.
- La presencia de todos los elementos del informe original.
- El cálculo automático de las puntuaciones. En consecuencia, también la eliminación de la posibilidad de cometer un error humano en esa tarea, así como el ahorro de tiempo en la realización de la misma.
- De la misma forma, el cálculo y dibujo de la gráfica también están automatizados.
- En el apartado de anotaciones, se muestran campos con más información que en el informe del test original.

Paciente: Rodrigo Gómez Fernández

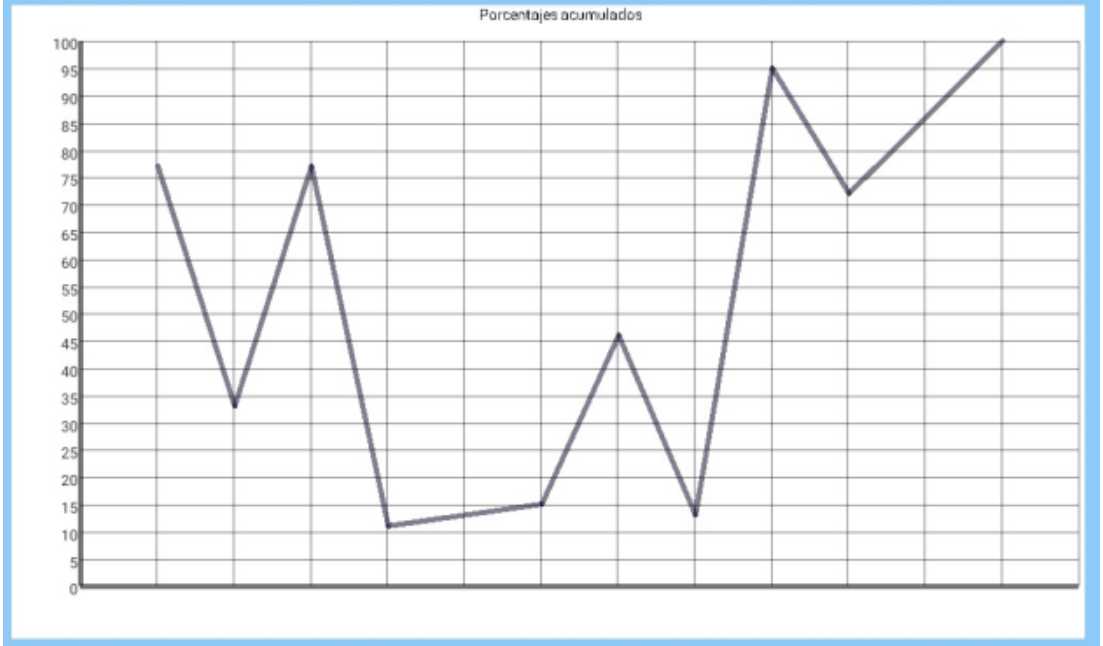
Test: test01

Tipo de test: TEDI-MATH: Primero de Educación Primaria, Periodo 2

Fecha realización: 10/02/2022

PUNTUACIONES

PUNTUACIONES BÁSICAS	PD	Intervalo de confianza del 90% Límite inferior	Límite superior	% acumulados
1. Contar	12	9	15	77
2. Numerar	11	9	13	33
3.A. Sistema numérico arábigo.	23	20	26	77
3.B. Sistema numérico oral.	24	21	27	11
3.C. Sistema en base 10.	N/A*	N/A*	N/A*	N/A*
3.D. Codificación.	11	9	13	15
4. Operaciones lógicas	13	10	16	46
5.A. Operaciones con apoyo en imágenes	5	4	6	13
5.B. Operaciones con enunciado aritmético	27	24	30	95
5.C. Operaciones con enunciado verbal	7	6	8	72
5.D. Conocimientos conceptuales	0	0	1	-
6. Estimación del tamaño	15	13	17	100



PUNTUACIONES COMPLEMENTARIAS	PD	% acumulados
3.A.1. Decisión numérica escrita	8	100
3.A.2. Comparación de números arábigos	15	88
3.B.1. Decisión numérica oral	11	75
3.B.2. Juicio gramatical	5	26
3.B.3. Comparación de números orales	8	6
3.C.1 Representación con palitos y 3.C.2 Representación con monedas	N/A*	N/A*
3.C.3 Reconocimiento de unidades, decenas y centenas	N/A*	N/A*
3.D.1. Escritura de números arábigos al dictado	6	100
3.D.2. Escritura de números arábigos en voz alta	5	9
4.A. Series numéricas	2	19
4.B. Clasificación numérica	2	100

	PD	% acumulados
4.C Conservación numérica	2	33
4.D. Inclusión numérica	2	20
4.E. Descomposición aditiva	5	65
5.B.1. Sumas simples	6	29
5.B.2. Sumas con huecos	3	27
5.B.3. Restas simples	4	31
5.B.4. Restas con huecos	3	83
5.B.5. Multiplicaciones simples	11	98
6.A. Comparación de modelos de puntos dispersos	5	12
6.B. Tamaño relativo	10	35

* N/A: No aplica. No se realiza la prueba al paciente ya que el test no la incluye.

Figura 11.1: Ejemplo de informe en PDF generado por la aplicación.

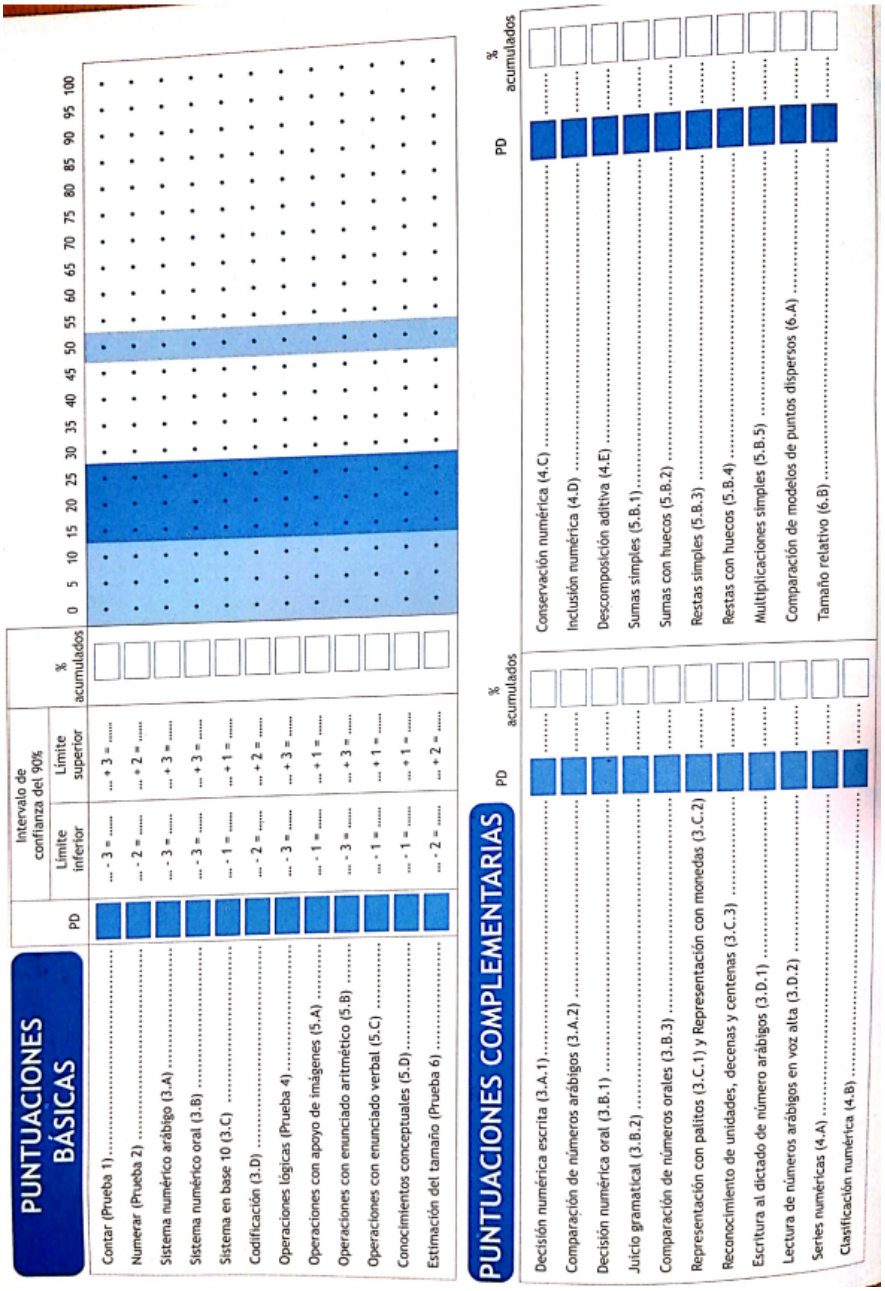


Figura 11.2: Ejemplo de informe TEDI-MATH.

Figura 11.3: Ejemplo de gráfica en la aplicación.

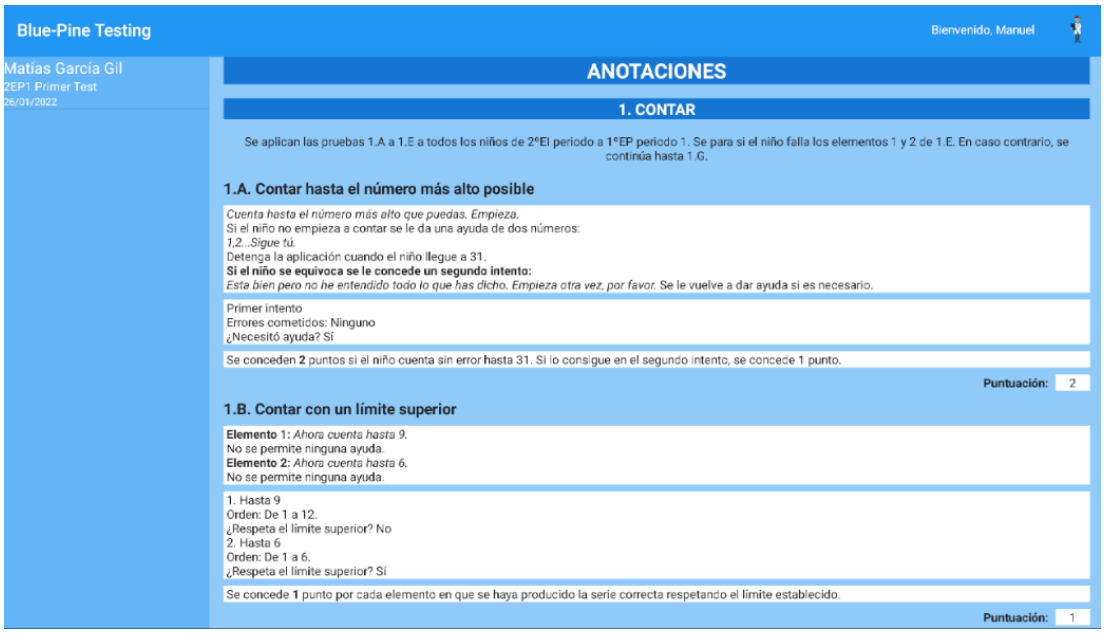


Figura 11.4: Ejemplo de anotaciones en la aplicación.

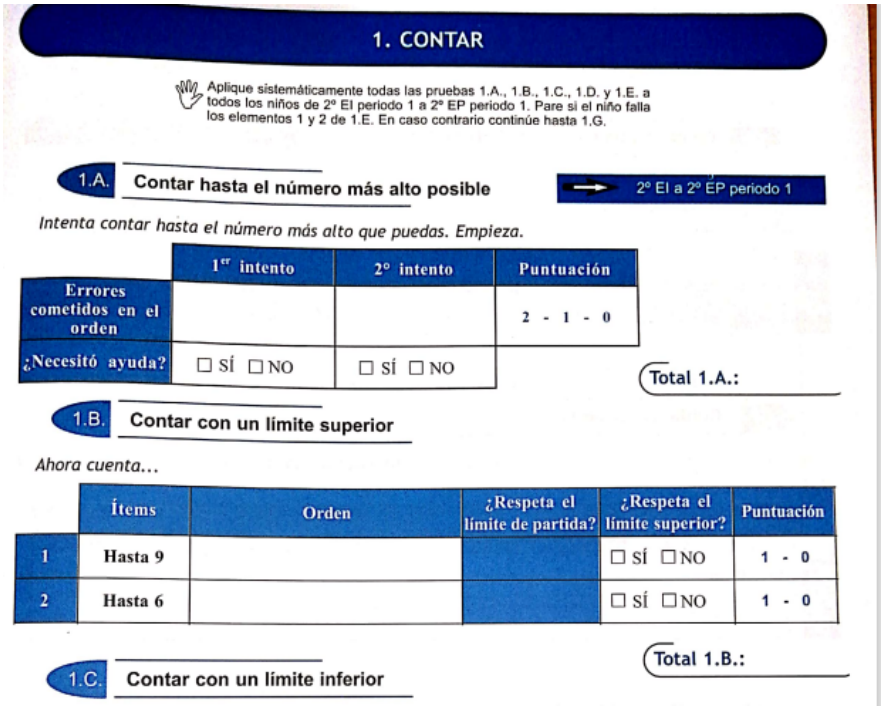


Figura 11.5: Ejemplo de anotaciones en el test original.

	NeurekaTEST	CAB-DC	Smartick	Blue-Pine Testing
Multiplataforma	No	Sí	Sí	No
Duración	15 minutos	30 a 40 minutos	15 minutos	45 minutos
Usuario	Paciente	Paciente	Paciente	Logopeda
Edad de pacientes	5 a 12 años	Más de 7 años	5 a 10 años	4 a 8 años
Informes	Sí	Sí	Sí	Sí
Gestión	No	No	Parcial	Total
Herr. refuerzo	Sí	No	Sí	No
Anotaciones	No	No	No	Sí

Cuadro 11.1: Comparativa final con el estado del arte.

11.1 Comparativa de la aplicación con el estado del arte

Una vez que se tiene el producto final, es posible actualizar la tabla 3.1 del capítulo 3, donde, haciendo una revisión de aplicaciones similares, se realizaba una comparativa inicial de sus funcionalidades con el producto esperado como resultado del desarrollo de este proyecto. Con el producto final real, obtenemos la actualización de esa tabla que se presenta en la tabla 11.1.

También se revisan los objetivos mencionados en ese capítulo:

- El evaluador es el logopeda, y sólo él realiza las evaluaciones.
- Se permite que el logopeda realice anotaciones sobre la evaluación, como en un caso clínico.
- Se replica con todos sus componentes la evaluación del test TEDI-MATH [2].
- La edad de evaluación es de 4 a 8 años, menor que en las otras herramientas.
- El logopeda tiene el control total de la gestión de la herramienta, sin dependencia de entidades externas.
- Se elaboran informes detallados en PDF.
- No se realiza una implementación multiplataforma. Sin embargo, la aplicación está preparada para realizar ese paso sin cambiar el código fuente, sino sólo modificando los ficheros XML con los datos de la colocación de los elementos de pantalla.

Adicionalmente, la aplicación está preparada para su traducción a cualquier idioma, nacional o internacional. Esto es debido a que se separaron los ficheros donde están los textos que ve el usuario del código fuente.

11.2 Verificación del cumplimiento de los requisitos

Se revisan los requisitos de primer nivel, que indican si la aplicación es de suficiente calidad como para considerarse un producto final:

- Se cumple el esquema de roles y sus funcionalidades: paciente, tutor y logopeda.
- Se implementa el test TEDI-MATH [2].
- La aplicación hace diagnóstico a niños en etapas de escolarización comprendidas entre segundo de Educación Infantil y tercero de Educación Primaria.
- Se implementa la aplicación en Android.
- Gestión de usuarios:
  - Alta y baja entre usuarios.
  - Gestión de contraseñas.
  - Garantía de protección de datos.
- Adaptación a diversidad, aunque los pacientes no sean objetivo del cambio:
  - Se puede cambiar el color de la aplicación.
  - Cada usuario puede escoger un avatar para su perfil.
- Se implementa para tabletas.
- Hay registro y almacenamiento de todos los datos deseados.
- Análisis de datos:
  - Visualización en el dispositivo de todos los objetos de la evaluación.
  - Se implementa el módulo de gráficas.
  - Se pueden generar los informes en PDF y almacenar en el dispositivo.
- Amplio módulo de evaluación, donde se puede analizar la posibilidad de estar afectado por discalculia o DAM mediante la realización de la batería de tests.

11.3 Líneas de trabajo futuras

La mejor forma de obtener las líneas de trabajo futuro es revisar los requisitos, secundarios y terciarios, que se sabe que podrían dar una mayor calidad a la aplicación. Por lo tanto, teniendo en cuenta esos requisitos que no han sido abordados o lo han sido parcialmente en el producto final entregado, las líneas de trabajo a corto y medio plazo serían las siguientes:

- Separación de logopedas entre aquellos que son administradores y los que no tienen acceso a ese módulo.
- Implementación de otros tests.
- Implementación en iOS.
- Adaptación a diversidad:
  - Variabilidad en la configuración del tamaño de fuente.
  - Mensajes de voz.
- Uso de pictogramas en menús e instrucciones.
- Capacidad de ser usado en móviles.
- Análisis de datos:
  - Exportación de datos a CSV.
  - Generación de informes grupales en PDF.
- Contacto, gestión de reuniones e intercambio de información entre familias y especialistas.

Como ya se ha comentado a lo largo de este trabajo, ninguna de estas funcionalidades afecta de forma importante a la usabilidad y utilidad del producto final entregado, que constituye una versión plenamente funcional, actualmente en fase de prueba por parte de los logopedas y terapeutas de la entidad ASPERGA con la que se ha colaborado para el desarrollo de este Trabajo Fin de Grado. En general, este tipo de aplicaciones, que automatizan los procesos de estas entidades, digitalizan pesados procedimientos tradicionalmente realizados en papel y permiten además un seguimiento no presencial de los usuarios a los que atienden estas asociaciones promueven la igualdad en oportunidades y la inclusión social de un colectivo que, si cuenta con terapias adecuadas desde temprana edad, puede reducir de forma importante las limitaciones en su vida diaria causadas por este tipo de trastornos.

Apéndices

Apéndice A

Manual de Uso

En este capítulo se describirá la metodología para usar la aplicación y sus funcionalidades.

A.1 Prerrequisitos

Para instalar la aplicación es necesario una tableta con el sistema operativo Android. La mínima versión de Interfaz de Programación de Aplicaciones (API) requerida para la aplicación es la 23, lo que equivale a Android 6.0.

En los ajustes del sistema de la tableta se tiene que permitir la instalación de aplicaciones de orígenes desconocidos.

A.2 Instalación de la aplicación

Junto con esta memoria se entregará un archivo con el nombre del proyecto y la versión, que incluirá todo lo necesario para que funcione la aplicación. Desde la tableta, ejecutaremos el fichero .apk . Con esa acción, la tableta instalará la aplicación.

A.3 Permisos

Antes de iniciar la aplicación, se le debe otorgar permisos de almacenamiento, ya que es necesario para el almacenamiento en memoria de informes de evaluación.

Para configurar el permiso, hay que ir a Ajustes del dispositivo > Aplicaciones > BluePine Testing > Permisos. Para activarlo tan sólo hace falta deslizar el botón que aparece a la derecha, como se indica en la figura A.1 (página 85).

En caso de que no se otorgue el permiso de almacenamiento ahora, se pedirá más adelante en la aplicación cuando sea necesario.

A.4 Primer inicio

A.4.1 Login

Ahora se iniciará la aplicación y lo primero que aparecerá será la pantalla de login. En el primer inicio no se introducirá ningún dato, tan sólo pulsaremos el botón Acceder, como se ve en la figura A.2 (página 85).

De esta forma se accederá a la aplicación como Administrador, un usuario que se usará para crear nuestro perfil y que se descartará al final del proceso.

A.4.2 Configuración de logopeda

Ahora el usuario se situará en la pantalla principal de aplicación (figura A.3). Desde ahí, hay que pulsar el icono del usuario en la esquina superior derecha. Una vez pulsado, se abrirán tres opciones: Editar Perfil, Configuración y Administrador. Hay que pulsar la tercera de ellas, es decir, Administrador.

Ahora, en la pantalla de Administrador, se seleccionará Gestionar Usuarios > Añadir Logopeda, donde se expandirán los campos que se podrán rellenar con nuestros datos. Este proceso se muestra en las figuras A.4 y A.5 (página 86).

Sólo es necesario para los usuarios introducir el nombre, los apellidos y el correo electrónico. En cualquier momento se podrá introducir el resto, completando la información del usuario si se desea. Una vez insertados los datos, se pulsará el botón de Añadir, y aparecerá un diálogo similar al de la figura A.6 de la página 86. En el diálogo se introducirán los siguientes datos:

- **Contraseña del usuario actual**: En este caso se dejará en blanco, pero en un futuro habrá que introducir la contraseña del usuario actual de la aplicación.
- **Nombre de login**: Nombre que se usará para iniciar sesión el nuevo logopeda.
- **Contraseña nueva**: Contraseña del nuevo logopeda.
- **Confirma la nueva contraseña**: Repetición de la contraseña del nuevo logopeda.

Ahora se pulsará confirmar y se puede verificar que se creó el usuario, ya que aparecerá en la pantalla de Gestión de Usuarios.

Por último, ahora o más adelante, se podrá realizar el proceso idéntico para crear más logopedas con acceso a la aplicación.

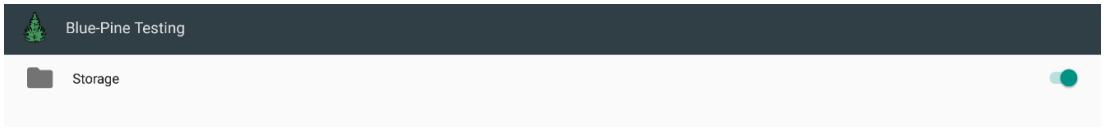


Figura A.1: Permiso otorgado a la aplicación.

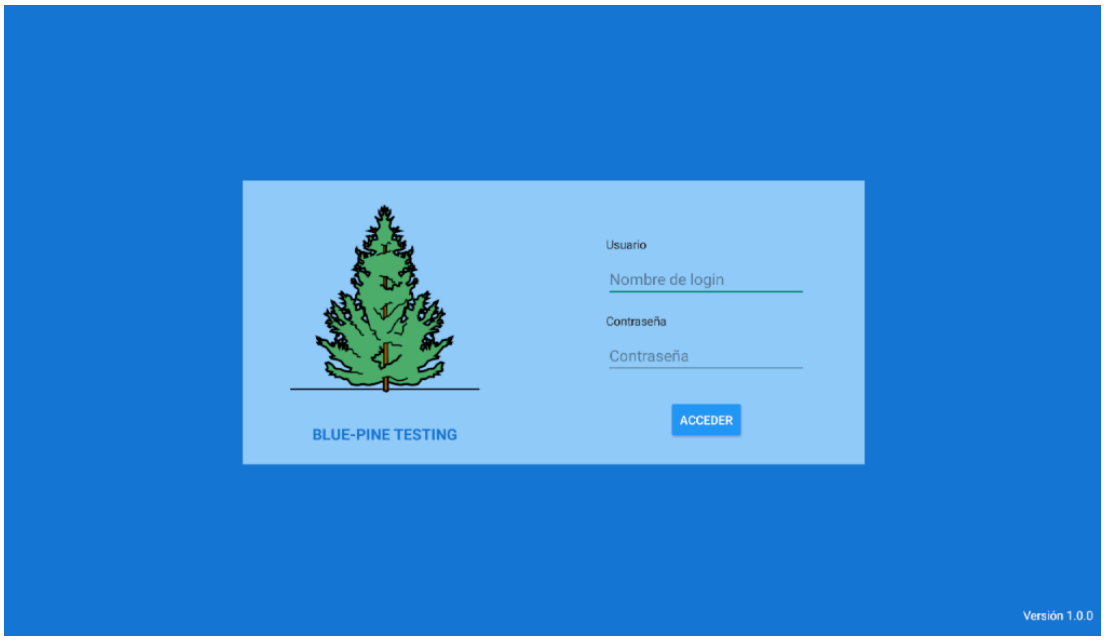


Figura A.2: Pantalla de login.

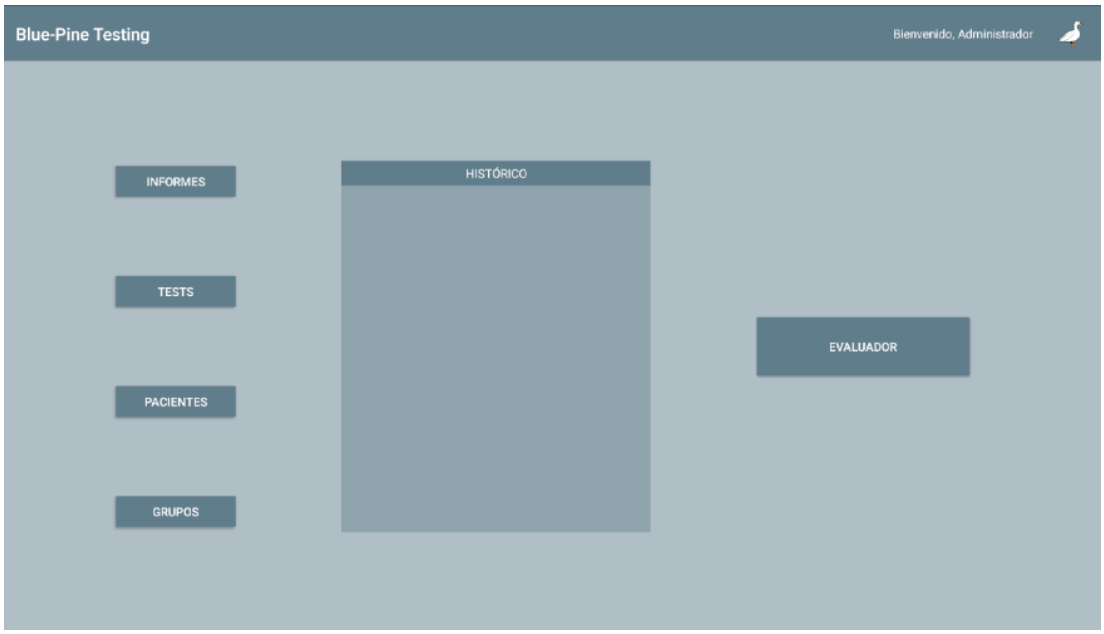


Figura A.3: Pantalla inicial.

Figura A.4: Pantalla de Gestión de Usuarios, desde donde se puede añadir un logopeda.

Figura A.5: Pantalla para introducir los datos del primer logopeda.

Figura A.6: Introducción de los datos de login del logopeda nuevo.

A.4.3 Eliminación del usuario Administrador

Una vez que se haya creado el primer usuario logopeda, hay que proceder a eliminar el usuario Administrador, ya que no hará falta para nada más. Para ello, volveremos a la pantalla de login (figura A.2) y se entrará en la aplicación con el usuario que acabamos de crear.

Para eliminar el usuario Administrador, se vuelve a la pantalla de Gestión de Usuarios (figura A.4), en la que se podrá ver el usuario Administrador. En la casilla de dicho usuario, se pulsará el botón de Eliminar Usuario y se nos pedirá la contraseña de nuestro usuario como medida de seguridad. Una vez introducida la contraseña, se pulsa el botón de Confirmar para eliminar el usuario administrador inicial.

Por último, se puede verificar que el usuario está eliminado porque ya no aparece en la lista de usuarios de esta sección.

A.5 Pantalla principal

La pantalla principal (figura A.18) contiene accesos a los siguientes elementos:

- **Informes**: Informes de evaluaciones realizadas anteriormente.
- **Tests**: Visualización y creación de tests.
- **Pacientes**: Listado de pacientes y opción de dar de alta nuevos pacientes.
- **Grupos**: Exploración de los grupos de pacientes y formación de nuevos grupos.
- **Histórico**: Acceso directo a la lista de las últimas veinte evaluaciones realizadas.
- **Evaluador**: Módulo en el que se realizan las evaluaciones a pacientes.
- **Menú Usuario**: En la esquina superior derecha, si se pulsa el icono de usuario, se desplegarán las siguientes opciones:
  - **Editar Perfil**: Formulario para la modificación de datos del usuario actual de la aplicación.
  - **Configuración**: Módulo para cambiar la contraseña del usuario actual de la aplicación.
  - **Administración**: Acceso al módulo de administración, en el que se puede gestionar usuarios, grupos y datos de los usuarios.

A.6 Tests

Los tests son la representación de una evaluación que se hará a un paciente. Un paciente podrá evaluarse con un test y, con ello, generar el informe correspondiente. Los tests implementados son los de TEDI-MATH, para distintos cursos de infantil y primaria.

A.6.1 Creación de Tests

Desde la pantalla principal (figura A.3), se pulsa el botón Tests. En esa pantalla se encuentra la opción de Crear Test (ver esquina inferior izquierda en la figura A.7 en la página 88).

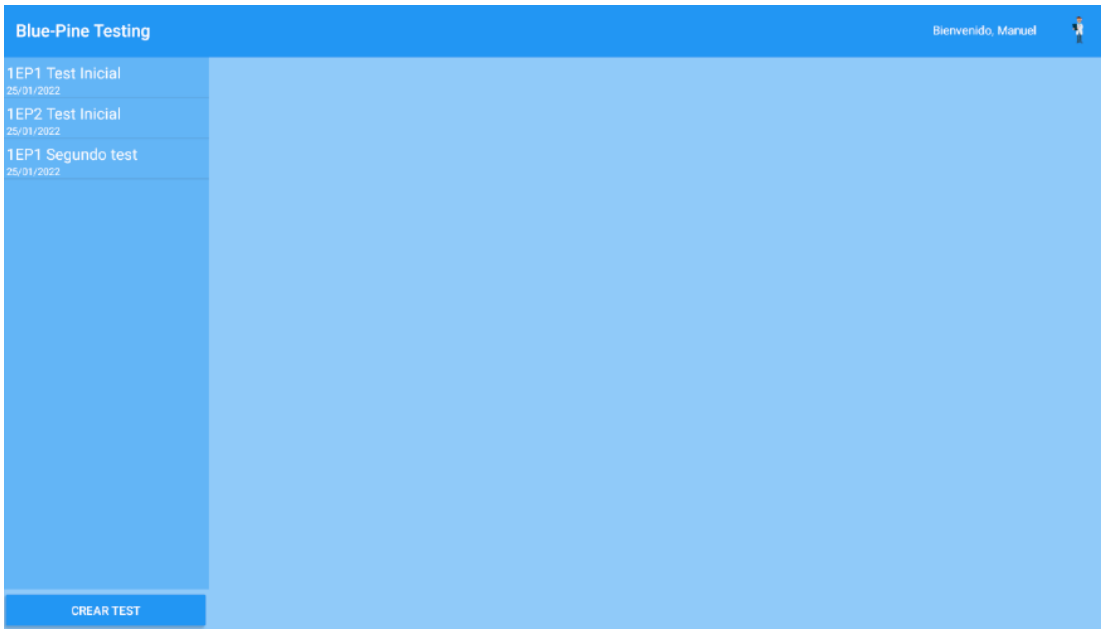


Figura A.7: Lista de tests y opción de Crear Tests.

En la pantalla de Crear Tests, se introducirán los siguientes datos:

- **Test**: Nombre del test.
- **Fecha de creación**: Campo generado automáticamente. Es la fecha en la que se crea el test.
- **Tipo de test**: Tipo de test al que pertenece.

Estos son los distintos tipos de test que se han implementado. Cada uno de ellos corresponde al test TEDI-MATH en un período:

- TEDI-MATH: Segundo de Educación Infantil, Periodo 1.

- TEDI-MATH: Segundo de Educación Infantil, Periodo 2.
- TEDI-MATH: Tercero de Educación Infantil, Periodo 1.
- TEDI-MATH: Tercero de Educación Infantil, Periodo 2.
- TEDI-MATH: Primero de Educación Primaria, Periodo 1.
- TEDI-MATH: Primero de Educación Primaria, Periodo 2.
- TEDI-MATH: Segundo de Educación Primaria, Periodo 1.
- TEDI-MATH: Segundo de Educación Primaria, Periodo 2.
- TEDI-MATH: Tercero de Educación Primaria, Periodo 1.
- TEDI-MATH: Tercero de Educación Primaria, Periodo 2.

Todos ellos son distintos y forman distintas pruebas, idénticas a las del método TEDI-MATH.

Por último hay que pulsar el botón Generar, que creará el test con los datos introducidos y estará listo para su evaluación.

En la figura A.8 se crea un test, al que se denomina *2EP2 Segundo Test*, y que representará un segundo test de evaluación para los pacientes que cursen segundo de educación primaria en la segunda parte del curso. Cuando se cree aparecerá un mensaje en la aplicación que indica ”Test generado correctamente”, confirmando que el proceso de creación del test fue correcto. De ahora en adelante ese test aparecerá en la pantalla de Test (figura A.7).

A.6.2 Visualización de un Test

Para ver la información relativa a un test, desde la pantalla principal (figura A.3) se pulsará el botón Tests. Esto volverá a mostrar la lista de tests, además del botón de Crear Test. En este caso, se selecciona un test de la lista y a continuación se visualizará su información, como se muestra en la figura A.9:

- **Test**: Nombre del test.
- **Fecha de creación**: Fecha en la que se ha creado el test.
- **Tipo de test**: Nivel del test TEDI-MATH.
- **Realizado en**: Pacientes a los que se le ha realizado el test.
- **Realizado en grupo**: Grupos a los que se ha realizado el test a todos los pacientes.

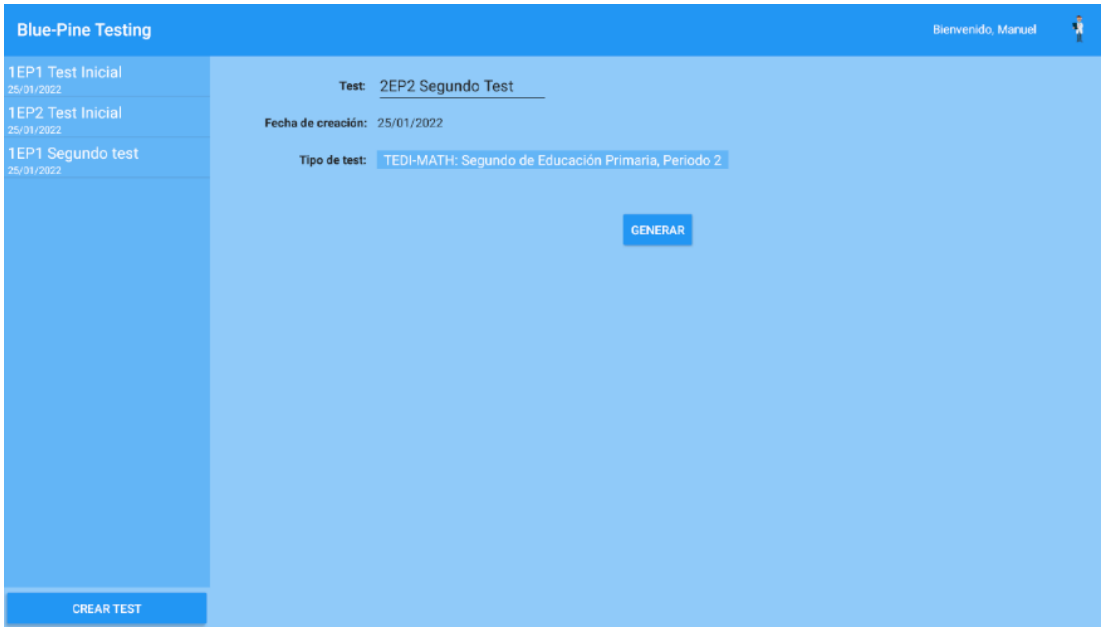


Figura A.8: Creación de un test.

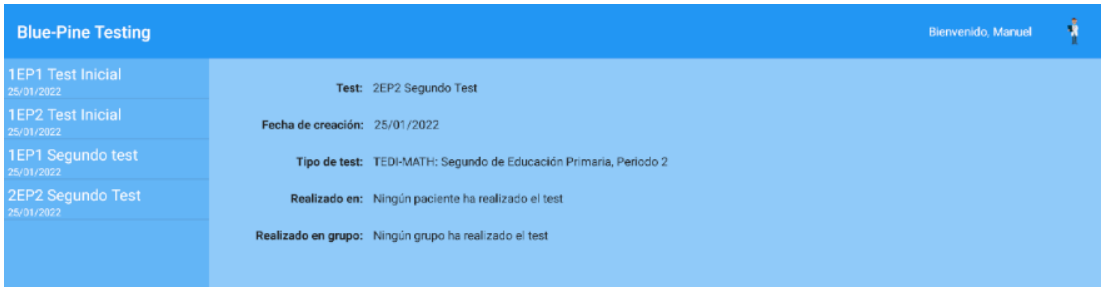


Figura A.9: Creación de un test.

A.7 Pacientes

Además de tests, se necesitarán pacientes para realizar las evaluaciones.

A.7.1 Añadir un paciente

Desde la pantalla principal (figura A.3) se pulsará el botón de Pacientes, lo que llevará a una pantalla como la que se muestra en A.10, donde se verá la lista de pacientes existentes (inicialmente vacía) y un botón para añadir un paciente nuevo.

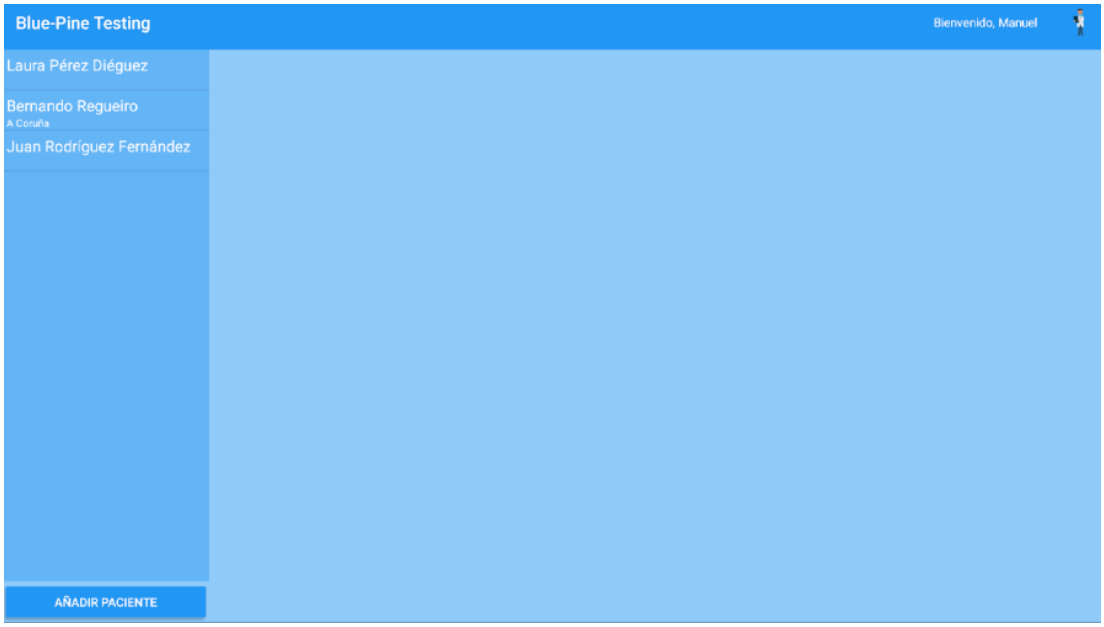


Figura A.10: Pantalla de pacientes.

Se pulsará el botón Añadir Paciente, que mostrará un formulario en el que se podrá introducir tanto sus datos como el de, al menos, un tutor. También se podrá añadir un segundo tutor, pero es opcional.

Como mínimo, para cada paciente y tutor, será necesario introducir su nombre y los apellidos y, en el caso de los tutores, un correo de contacto.

También será posible, para los pacientes, escoger un tutor que ya esté dado de alta en la aplicación.

En la figura A.11 se puede ver el formulario completado con los datos mínimos. A continuación, se pulsa el botón de Añadir para crear el paciente con los tutores. Un mensaje ”Paciente añadido correctamente” confirmará que el paciente se ha añadido a la lista.

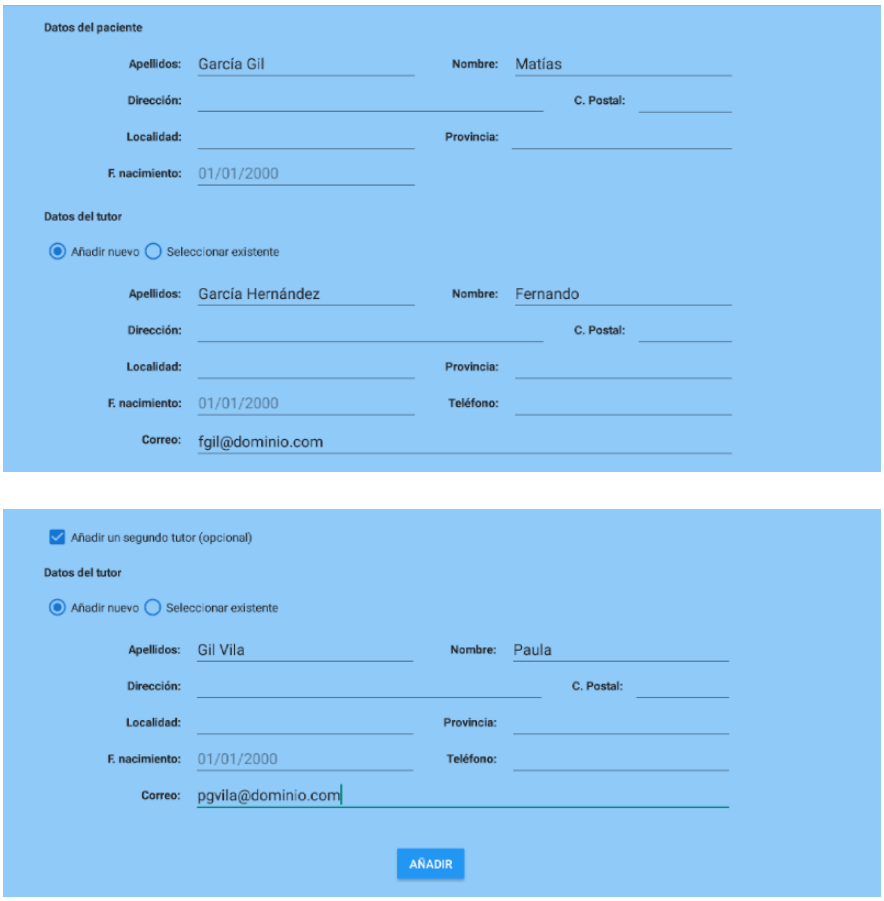


Figura A.11: Añadir paciente con los datos mínimos.

A.7.2 Ver información de paciente.

Para mostrar la información de un paciente, desde la pantalla de Pacientes (figura A.10), se pulsará un paciente de la lista. Con esto, aparecerá una pantalla, como la de la figura A.12, con los datos del paciente[1] y de su tutor o tutores, en caso de tener dos.

Figura A.12: Información de un paciente.

Por último, se ha añadido el botón Ver Informes, que permite tener un acceso rápido a todos los informes de las evaluaciones del paciente.

A.8 Grupos

Un grupo es un conjunto de pacientes dirigidos por un logopeda gestor. Esto sirve para establecer una gestión de logopedas a pacientes.

Un paciente puede estar en un único grupo. Un logopeda puede ser gestor de un único grupo.

[1] En caso de que, para cualquier usuario, no se introduzca una fecha de nacimiento, se muestra un valor por defecto.

A.8.1 Creación de grupos

Para crear un grupo, desde la pantalla principal (figura A.3), hay que pulsar el botón Grupos. Esto llevará a la herramienta a la pantalla de Grupos (figura A.13), en el que se mostrará una lista de los grupos existentes y un botón Crear Grupo, que se pulsará para crear el grupo de pacientes.

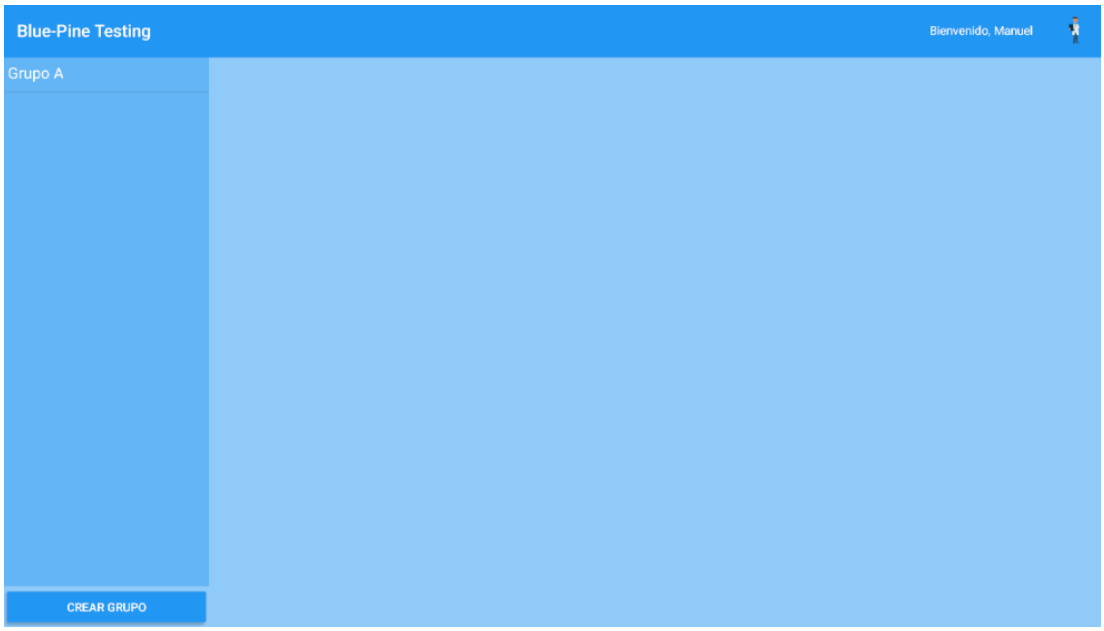


Figura A.13: Pantalla de grupos.

Al pulsar el botón Crear Grupo, aparece un formulario para rellenar los siguientes campos:

- **Nombre de Grupo**: Nombre que tiene el grupo.
- **Fecha de Creación**: Generado automáticamente. Fecha en la que se forma el grupo.
- **Gestor**: Menú desplegable que permite seleccionar el logopeda gestor del grupo.
- **Participantes**: Listado de pacientes que se incluirán en el grupo. Para añadir un paciente al grupo que se va a crear sólo hace falta pulsarlo en la lista de pacientes disponibles y se añadirá a la lista de participantes.

En la figura A.14 se refleja este proceso, donde se añaden a un grupo nuevo dos pacientes que fueron creados anteriormente.

Por último, se pulsará el botón Crear Grupo que hay debajo de la lista de participantes y, al pulsarlo, el usuario volverá a la pantalla principal y se confirmará mediante un mensaje ”Grupo creado correctamente”.

A.8.2 Ver Información de Grupo

Para ver la información de un grupo, se tiene que volver a la sección de Grupos desde la pantalla principal (figura A.3), pulsando el botón de Grupos.

Una vez que se está en el sección de grupos, se podrá ver el listado de los grupos actuales. Para ver la información del grupo, se pulsa la casilla correspondiente al mismo.

La información que se muestra es la siguiente:

- **Nombre de Grupo**: Nombre que tiene el grupo.
- **Fecha de Creación**: Fecha en la que se forma el grupo.
- **Gestor**: Logopeda gestor del grupo.
- **Participantes**: Listado de pacientes que pertenecen al grupo.

Por último, como se muestra en la figura A.15, hay un acceso rápido a los informes del grupo, pulsando para ello el botón Ver Informes.

A.9 Evaluador

Una vez que tenemos un test y un paciente, se podrá realizar una evaluación. Para ello, desde la pantalla principal (figura A.3), se pulsará el botón de Evaluador. Esto mostrará el diálogo A.16, desde el que se podrá escoger un test y un paciente.

En caso de que el paciente ya haya realizado el test, se mostrará una advertencia, ya que si se repite la evaluación se sobrescribirán los datos de la anterior.

Una vez que se han seleccionado el test y el paciente, se pulsará el botón confirmar para empezar la evaluación.

El evaluador consiste en una serie de pruebas, que son replicadas del test TEDI-MATH para el nivel del test seleccionado. En la figura A.17 se muestra un ejemplo de prueba.

De forma equivalente a la prueba realizada con el test original, se muestran los siguientes campos:

- **Test**: Test que se realiza, con el nivel en el que se ha configurado.
- **Paciente**: Nombre y apellidos del paciente al que se realiza el test.
- **Prueba**: Prueba del test TEDI-MATH que se está evaluando.
- **Subtest**: Paso concreto de la prueba que se está evaluando.
- **Material**: Material necesario para realizar la prueba. Algunas pruebas del test TEDI-MATH requieren material adicional.

Figura A.14: Formulario de creación de grupo.

Figura A.15: Evaluador: selección de paciente y test.

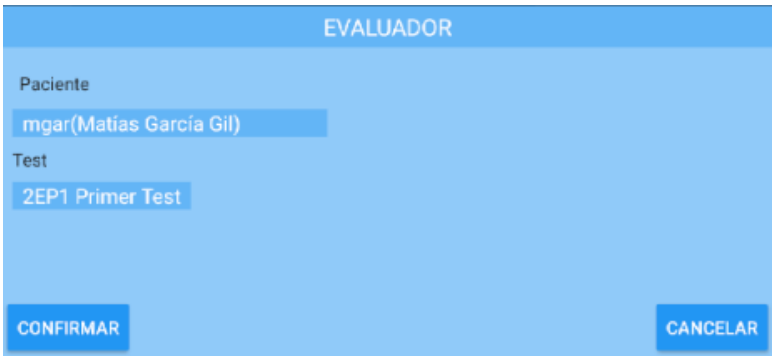


Figura A.16: Evaluador: selección de paciente y test.

- **Instrucciones**: Pasos que se deben seguir para evaluar al paciente en la prueba.
- **Anotación**: Campo de texto en el que se pueden guardar las notas con respecto a la prueba, que serán representadas en el informe de evaluación al visualizarlo. Por defecto, viene con la información necesaria para representar los posibles resultados de la prueba.
- **Criterio de éxito**: Explicación de cómo se debe puntuar al paciente en la prueba.
- **Puntuación**: En base al criterio de éxito, este campo sirve para almacenar el resultado de la prueba. Viene con una pista del rango de puntos que se le puede asignar a la prueba, por ejemplo, ”De 2 a 0”.

Además, se puede pulsar el botón Atrás, para repetir una prueba anterior, y el botón Siguiente para avanzar a la siguiente prueba. En la última prueba, el botón Siguiente se reemplazará por el botón Finalizar y Guardar para guardar los resultados y las anotaciones de la evaluación.

Por último, si se pulsa Finalizar y Guardar se volverá a la pantalla principal, con un mensaje ”Resultados guardados correctamente”, y con la evaluación que se acaba de realizar en la primera posición del acceso rápido a los últimos históricos de evaluaciones, como se muestra en la figura A.18.

A.10 Informes

Para ver los informes, primero el usuario se situará en la pantalla principal (figura A.3), y pulsará luego el botón Informes. Eso llevará a una pantalla donde se encontrará la lista de informes de las evaluaciones realizadas a pacientes, como se muestra en la figura A.19.

En la pantalla de informes, si se pulsa uno de ellos, se muestra toda la información de la evaluación, como en el test TEDI-MATH. Hay tres partes a destacar:

- **Información de la evaluación**: Se muestra a quién se ha realizado el test, qué test se hizo, de qué nivel es dicho test y la fecha en la que se evaluó al paciente.
- **Puntuaciones**: Puntuaciones básicas, su gráfica y puntuaciones complementarias. Las puntuaciones básicas son el nivel más alto de análisis, donde se muestra la puntuación del paciente en distintos ámbitos, así como su intervalo de confianza y el porcentaje de acumulados.

  La gráfica permite ver, para cada puntuación básica, cuál es el porcentaje de acumulados correspondiente de forma directa.

  Las puntuaciones complementarias son las pruebas individuales de algunas de las puntuaciones básicas, para ver en detalle posibles fallos en algunas de las pruebas.

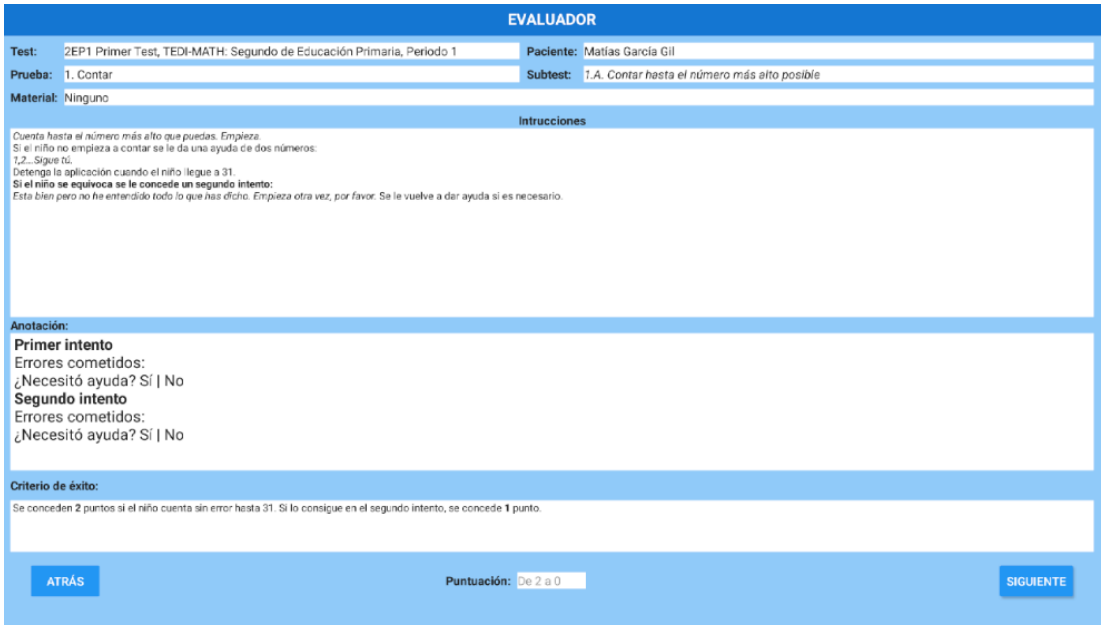


Figura A.17: Evaluador: ejemplo de prueba.

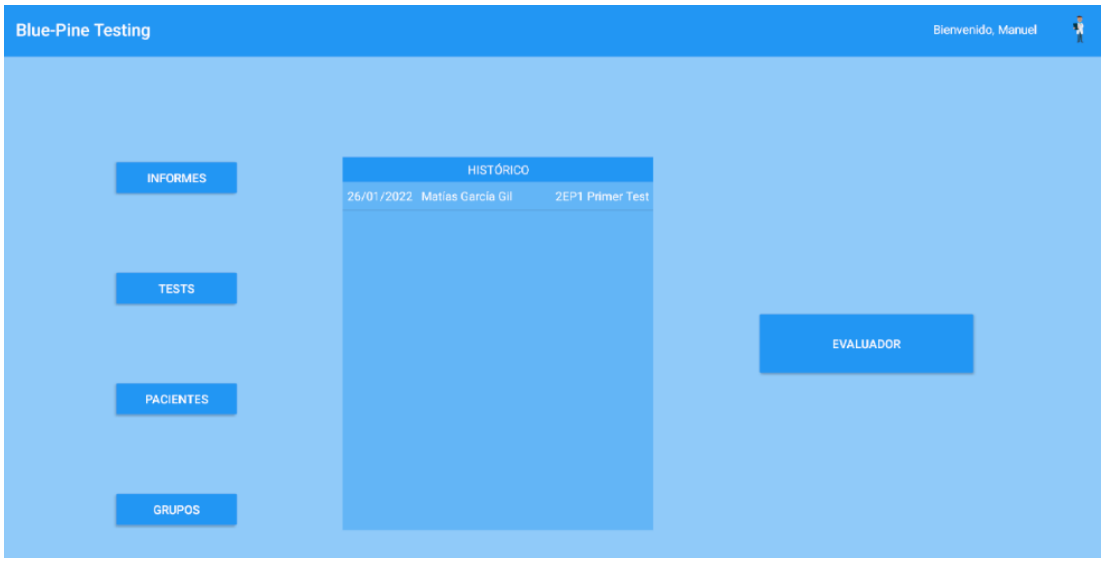


Figura A.18: Pantalla principal con un histórico cargado.

- **Anotaciones**: Para cada prueba que se ha realizado, se muestran las instrucciones de la prueba, las anotaciones del logopeda, el criterio de éxito y la puntuación.

En la figura A.20 se muestra un ejemplo de la cabecera del informe.

A.10.1 Exportar informe a PDF

Por último, pulsando el botón Exportar A PDF, se exportan los campos ”Información de evaluación” y ”Puntuaciones” (puntuaciones básicas, gráfica y puntuaciones complementarias), que se almacenarán en el directorio de descargas (Downloads) del dispositivo. Tanto el nombre como la localización del fichero se mostrarán en un mensaje una vez que la descarga haya finalizado con éxito.

Si no se otorgaron permisos de almacenamiento anteriormente, se solicitarán la primera vez que se pulse el botón Exportar A PDF. Una vez que éstos hayan sido concedidos, hará falta pulsar el botón Exportar A PDF de nuevo.

A.11 Editar Perfil

Desde la pantalla principal (figura A.3), se pulsará el icono de usuario en la esquina superior derecha, lo que desplegará un menú. En el menú desplegado, hay que pulsar Editar Perfil.

Se mostrará una pantalla con los datos del usuario actual de la aplicación, como se representa en la figura A.21.

Pulsando el botón Editar, se podrá modificar la siguiente información:

- **Avatar**: Icono de usuario. Se modifica pulsando el botón Cambiar Avatar.
- **Color**: Conjunto de colores de la aplicación. Para cambiarlo hay que pulsar el botón Cambiar Color.
- **Datos personales**: Información personal del usuario.
- **Login**: Nombre que usa el usuario para entrar en la aplicación. No se puede modificar.

Se puede pulsar el botón Cancelar para no sobrescribir los datos del usuario o Guardar para guardar los cambios. La segunda opción nos devolverá a la pantalla principal con los cambios aplicados, como se muestra en la figura A.22.

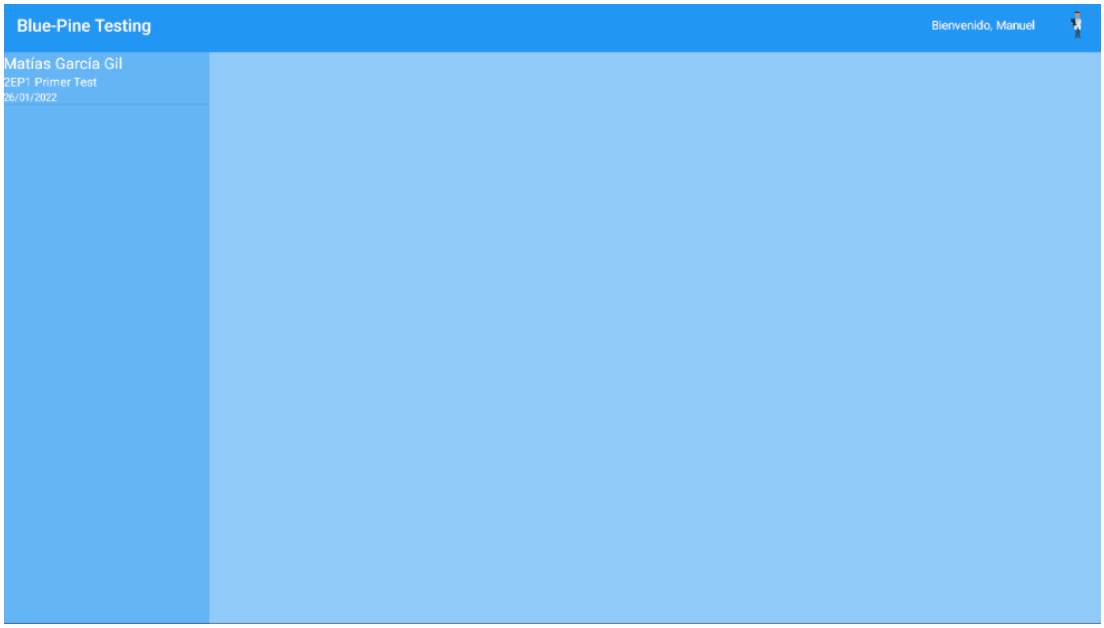


Figura A.19: Listado de informes.

Figura A.20: Ejemplo de cabecera de informe.

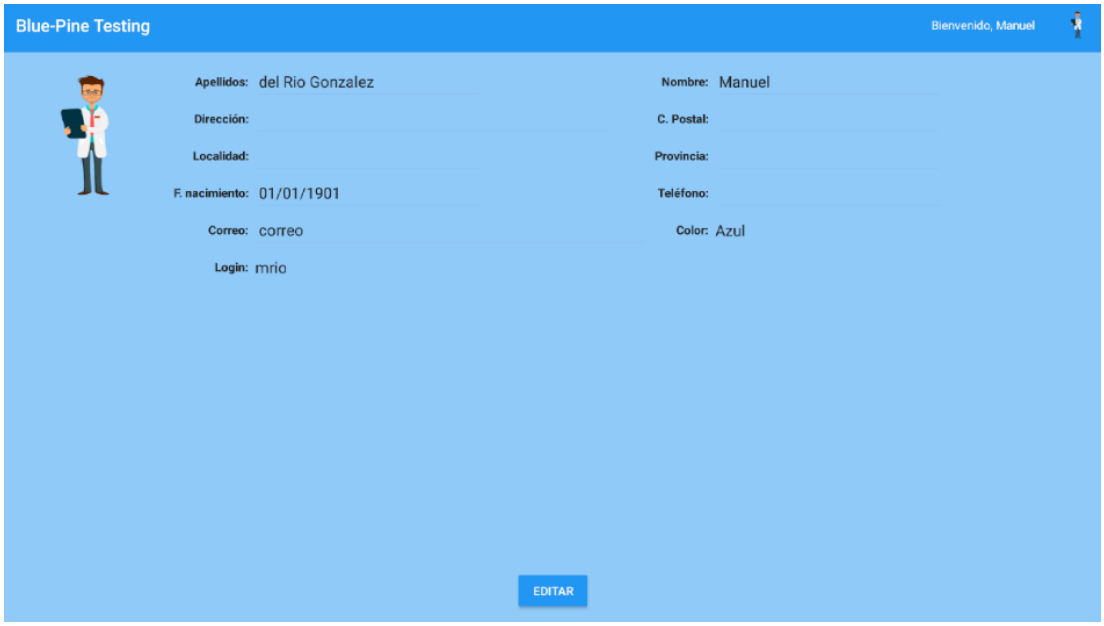


Figura A.21: Pantalla de editar perfil.

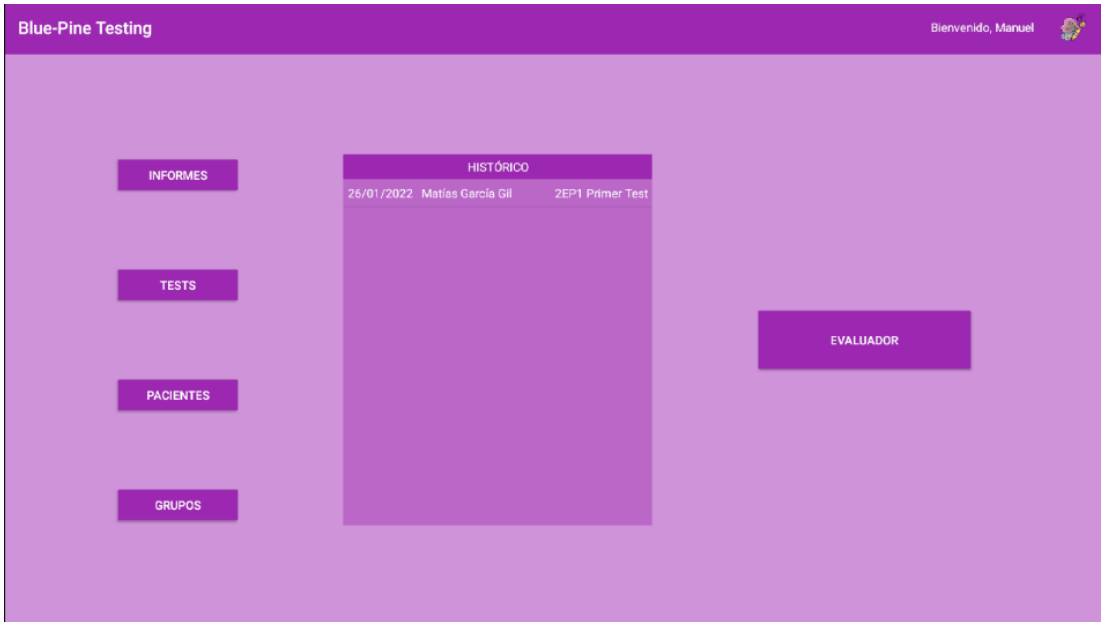


Figura A.22: Pantalla principal con el perfil editado.

A.12 Configuración

Desde la pantalla principal (figura A.3), se pulsará el icono de usuario en la esquina superior derecha, lo que desplegará un menú. En el menú desplegado, hay que pulsar Configuración.

A.12.1 Cambiar Contraseña

Para cambiar la contraseña del usuario actual, hay que pulsar el botón Cambiar Contraseña desde la pantalla de configuración. Esto mostrará un formulario para cambiar la contraseña del usuario, como se muestra en la figura A.23.

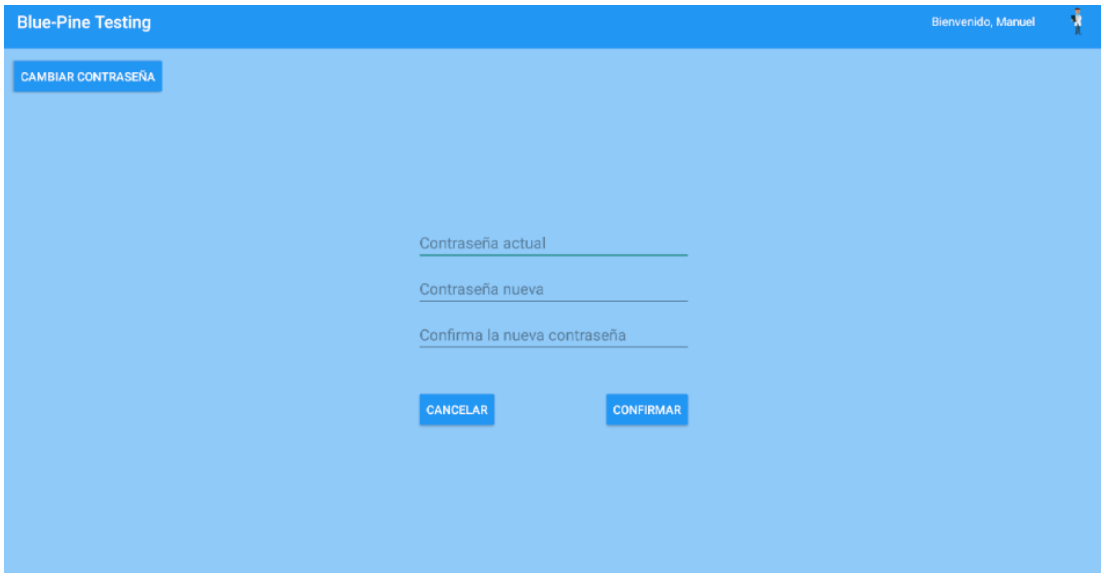


Figura A.23: Formulario para cambiar contraseña.

Habrá que introducir los siguientes datos:

- **Contraseña actual**: Contraseña actual del usuario de la aplicación.
- **Contraseña nueva**: Contraseña nueva para el usuario actual.
- **Repetir contraseña**: Repetición de la contraseña nueva.

Para confirmar el cambio, habrá que pulsar el botón Confirmar, lo que emitirá un mensaje ”Contraseña modificada correctamente” y la aplicación volverá a la pantalla principal.

A.13 Administrador

Desde la pantalla principal (figura A.3), se pulsará el icono de usuario en la esquina superior derecha, lo que desplegará un menú. En el menú desplegado, hay que pulsar Administrador.

Todos los logopedas tienen acceso a este menú, en el que se muestran los siguientes campos (figura A.24):

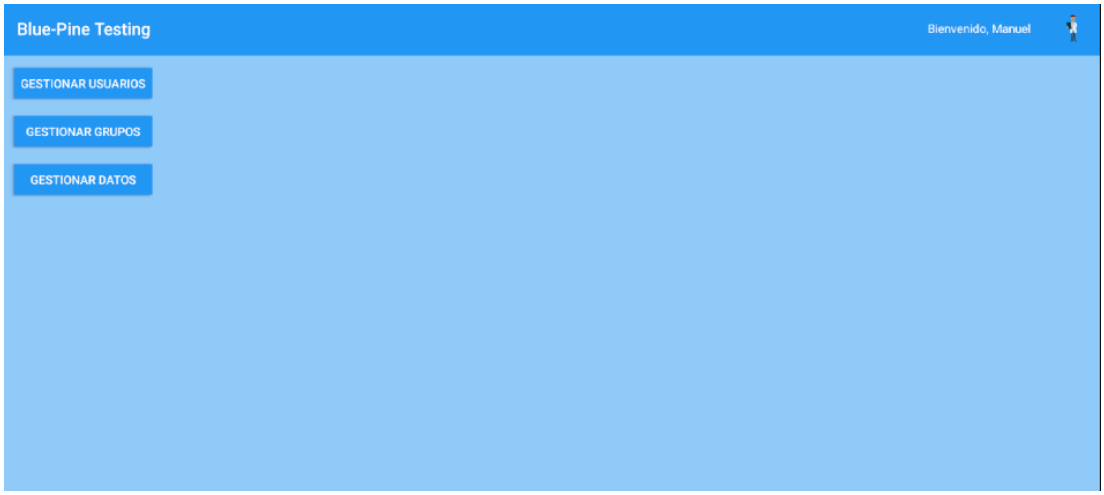


Figura A.24: Pantalla de administrador.

- **Gestión Usuarios**: Menú para la gestión de usuarios y añadir logopedas nuevos.
- **Gestionar Grupos**: Sección para gestionar, para cada grupo, su logopeda gestor y participantes.
- **Gestionar Datos**: Módulo dedicado a la gestión de datos personales de los usuarios.

A.13.1 Gestionar Usuarios

Desde la pantalla de Administrador (figura A.24), se pulsará el botón Gestionar Usuarios. Esto mostrará, como en la figura A.25, para cada usuario, la siguiente información:

- **Login**: Nombre del usuario, con el que puede entrar a la aplicación si es Tutor o Logopeda.
- **Rol**: El usuario puede ser Paciente, Tutor o Logopeda.
- **Nombre y apellidos**: Nombre completo del usuario.

Además, para cada usuario, habrá las siguientes opciones:

- **Editar usuario**: Si se pulsa, este botón llevará a la sección Editar Perfil, pero en este caso se editará el usuario seleccionado.
- **Eliminar usuario**: Si se pulsa, se eliminará el usuario de la aplicación, y no se mostrará en la misma ni podrá tener acceso a la aplicación en el login. Se requiere confirmación por parte del usuario actual de la aplicación.
- **Restablecer Contraseña**: Para los usuarios Tutor y Logopeda, se podrá cambiar su contraseña. Esto está diseñado para casos en los que un usuario olvide su contraseña. Se requiere confirmación por parte del usuario actual de la aplicación.

Por último, se podrá configurar un logopeda nuevo pulsando el botón Añadir Logopeda.

A.13.2 Gestionar Grupos

Desde la pantalla de Administrador (figura A.24), se pulsará el botón Gestionar Grupos. Esto mostrará, como en la figura A.26, para cada grupo, la siguiente información:

- **Nombre**: Nombre del Grupo.
- **Gestor**: Nombre y apellidos del logopeda gestor del grupo.

Por último, para cada grupo, habrá las siguientes opciones:

- **Editar Grupo**: Si se pulsa, este botón llevará a un formulario para modificar el logopeda gestor del grupo y se podrá añadir o eliminar pacientes del mismo.
- **Eliminar Grupo**: Si se pulsa, se eliminará el grupo. Esto requiere confirmación por parte del usuario actual de la aplicación.

A.13.3 Gestionar Datos

Con motivo de la Ley Orgánica de Protección de Datos Personales y garantía de los derechos digitales, se crea una sección dedicada a la gestión de los datos de usuarios.

Desde la pantalla de Administrador (figura A.24), se pulsará el botón Gestionar Datos. Esto mostrará, como en la figura A.27, para cada usuario eliminado, la siguiente información, el nombre y apellidos del usuario.

Por último, para cada usuario eliminado, habrá las siguientes opciones:

- **Restablecer Usuario**: Si se pulsa, este botón restablecerá al usuario seleccionado en la aplicación, como si nunca hubiera sido borrado. Esto requiere confirmación por parte del usuario actual de la aplicación.

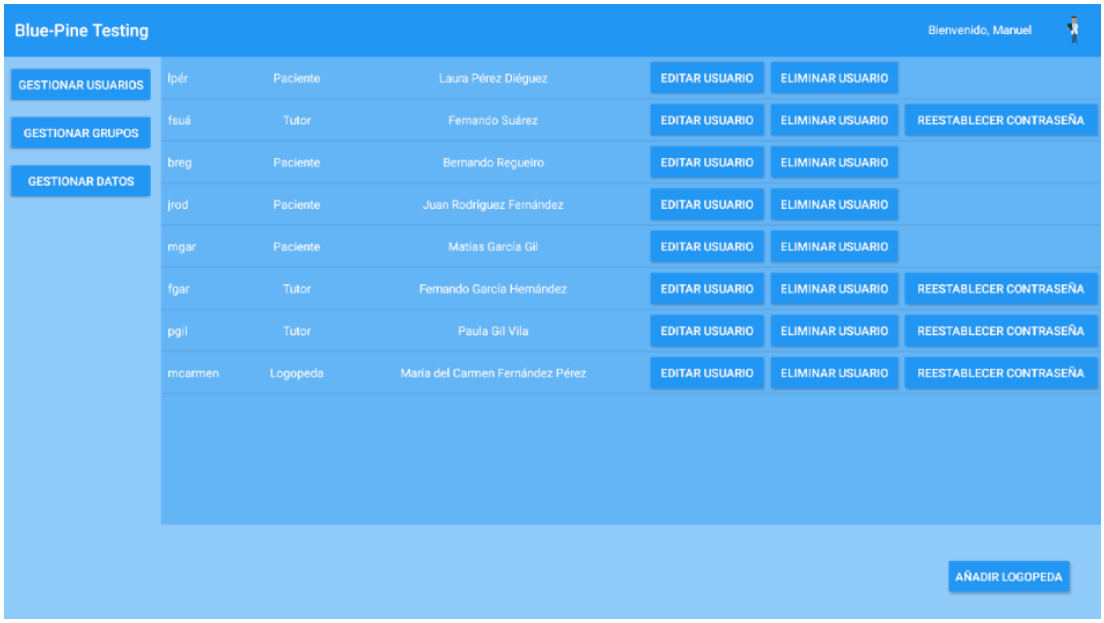


Figura A.25: Pantalla de gestión de usuarios.

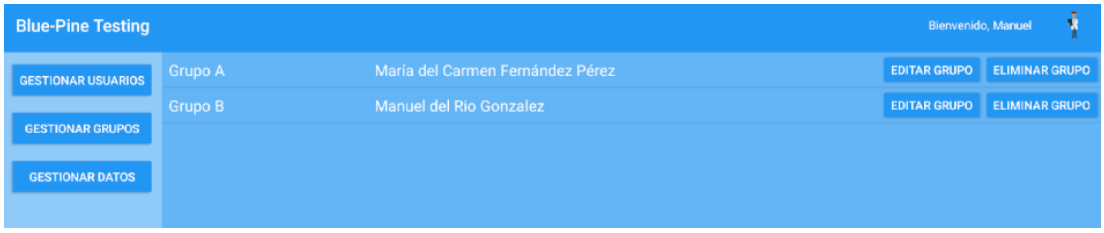


Figura A.26: Pantalla de gestión de grupos.

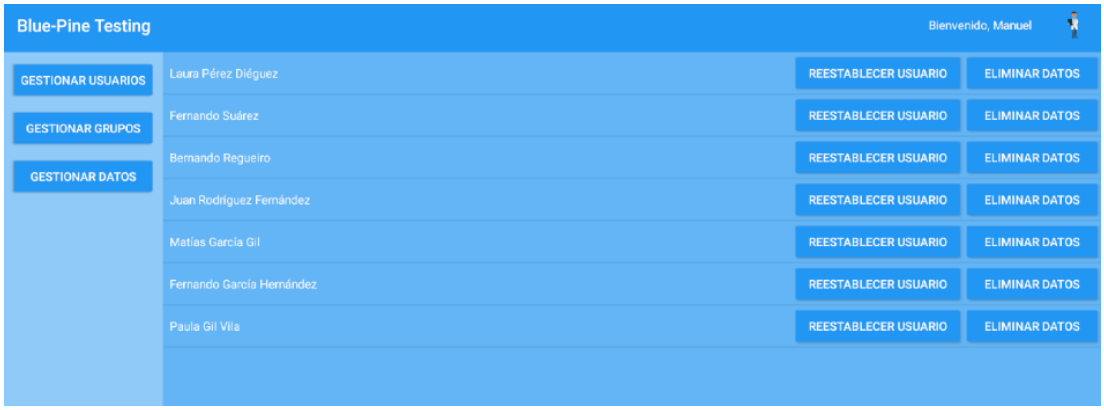


Figura A.27: Pantalla de gestión de datos.

- **Eliminar Datos**: Si se pulsa, se eliminarán los datos del usuario seleccionado. Esta acción es completamente irreversible, ya que elimina todo registro del usuario en la aplicación. Esta acción requiere confirmación por parte del usuario actual de la aplicación.

Lista de acrónimos

API Interfaz de Programación de Aplicaciones. 83

DAM Dificultad en el Aprendizaje de las Matemáticas. 1, 5, 6, 36, 38, 55, 73, 79

DAO Data Access Objet. 16, 60–64, 66, 67

FIC Falcultade de informática da Coruña. 17

LA Listado - Acción. 47, 50

LIC Listado - Información - Creación. 25, 47, 49, 51

PD Puntuación Directa. 45, 47

TDAH Trastorno de déficit de atención con hiperactividad. 5, 11, 12

TFG Trabajo de Fin de Grado. 3

Bibliografía

[1] L. Giordano, *Discalculia escolar: dificultades en el aprendizaje de las matemáticas*, 3rd ed. Buenos Aires, 1976.

[2] "TEDI-MATH." [En línea]. Disponible en: https://web.teaediciones.com/tedi-math.aspx

[3] M. A. Palacián, "La discalculia en Educación Infantil. Un estudio de caso," 2020. [En línea]. Disponible en: https://www.researchgate.net/publication/344021389_La_discalculia_en_Educacion_Infantil_Un_estudio_de_caso

[4] J. R. Alonso, "Salir de la discalculia," 2019. [En línea]. Disponible en: https://jralonso.es/2019/05/02/salir-de-la-discalculia/

[5] J. García-Orza, "Dislexia y discalculia. ¿Extraños compañeros de viaje?" 2012. [En línea]. Disponible en: http://psibasica.uma.es/javiergarciaorza/upload/personal/JGORZA_Dislexia%20y%20discalculia.pdf

[6] C. A. Panadero, "Las consecuencias sociales de las dificultades de aprendizaje en niños y adolescentes." [En línea]. Disponible en: http://demo.psicoaragon.es/sites/default/files/las_consecuencias_sociales_de_las_dificultades_de_aprendizaje_en_ninos_y_adolescentes.pdf

[7] "Evaluación Cognitiva para investigaciones sobre Discalculia." [En línea]. Disponible en: https://www.cognifit.com/es/evaluacion-cognitiva/test-discalculia

[8] "NEUREKA-TEST." [En línea]. Disponible en: https://neureka-test.web.app/lang/es/neurekatests.html

[9] "BERDE." [En línea]. Disponible en: http://ladiscalculia.es/wp-content/uploads/2018/04/berde.pdf

[10] "Smartick." [En línea]. Disponible en: https://www.smartick.es/dyscalculia.html

[11] e. d. e. Universidad Internacional de Valencia, "Como trabajar la discalculia en el aula ordinaria." [En línea]. Disponible en: https://www.universidadviu.com/es/actualidad/nuestros-expertos/como-trabajar-la-discalculia-en-el-aula-ordinaria

[12] "NEUREKA-SUM." [En línea]. Disponible en: Véaselapáginaweb:https://neurekalab.es/lang/es/neurekanum.html

[13] "Microsoft Word." [En línea]. Disponible en: https://www.microsoft.com/es-es/microsoft-365/word

[14] "Dia." [En línea]. Disponible en: http://dia-installer.de/index.html.es

[15] "Android Studio." [En línea]. Disponible en: https://developer.android.com/studio

[16] "Java." [En línea]. Disponible en: https://www.java.com/es/

[17] "XML." [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/XML/XML_introduction

[18] "Room." [En línea]. Disponible en: https://developer.android.com/training/data-storage/room?hl=es-419

[19] "Microsoft Excel." [En línea]. Disponible en: https://www.microsoft.com/es-es/microsoft-365/excel

[20] "Git." [En línea]. Disponible en: https://git-scm.com/

[21] "GitLab." [En línea]. Disponible en: https://about.gitlab.com/

[22] "Microsoft Teams." [En línea]. Disponible en: https://www.microsoft.com/es-es/microsoft-teams/log-in

[23] "Extreme Programming: Qué es y cómo aplicarlo." [En línea]. Disponible en: https://openwebinars.net/blog/extreme-programming-que-es-y-como-aplicarlo/

[24] "Scrum: Aprende a utilizar scrrum con lo mejor de él." [En línea]. Disponible en: https://www.atlassian.com/es/agile/scrum

[25] "Qué es Kanban: Definición, características y ventajas." [En línea]. Disponible en: https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban

[26] "¿Cuál es el sueldo de un Programador de Android?" [En línea]. Disponible en: https://www.tokioschool.com/noticias/cual-es-el-sueldo-de-un-programador-android/

[27] "PC expansión." [En línea]. Disponible en: https://www.pcexpansion.es/asus_k55vd_sx009h.php

[28] "Mediamarkt." [En línea]. Disponible en: https://www.mediamarkt.es/es/product/_tablet-bq-aquaris-m10-16-gb-blanco-wifi-10-1-wxga-2-gb-ram-mediatek-mt8163b-android-12975.html

[29] "Microsft Planes para Empresas." [En línea]. Disponible en: https://www.microsoft.com/es-es/microsoft-365/business/compare-all-microsoft-365-business-products

[30] "Aprendizaje-Servicio." [En línea]. Disponible en: https://www.udc.es/es/ocv/Aprendizaxe_servizo/

[31] "Asperga: Asociación Galega de Asperger." [En línea]. Disponible en: https://asperga.org/

[32] "Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales." [En línea]. Disponible en: https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673

[33] "Introducción a actividades." [En línea]. Disponible en: https://developer.android.com/guide/components/activities/intro-activities?hl=es-419

[34] "Fragmentos." [En línea]. Disponible en: https://developer.android.com/guide/components/fragments?hl=es-419

[35] "Cómo solicitar permisos de la app." [En línea]. Disponible en: https://developer.android.com/training/permissions/requesting?hl=es-419

[36] "Cómo ejecutar apps en Android Emulator." [En línea]. Disponible en: https://developer.android.com/studio/run/emulator?gclid=CjwKCAiA6Y2QBhAtEiwAGHybPS3NhbC6BsuFrnMBwI_TasYNiGaNJS_nZWmcvxgUJXbCmRBr0MrPbhoC8XEQAvD_BwE&gclsrc=aw.ds