



ugr | Universidad
de **Granada**

TRABAJO FIN DE GRADO
GRADO DE INGENIERÍA EN INFORMÁTICA

Aplicación multiplataforma para el aprendizaje del lenguaje musical

Subítulo del Proyecto

Autor
Sergio Hervás Cobo

Directores
Luis López Escudero
Germán Arroyo Moreno



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

—
Granada, mes de 201



Aplicación multiplataforma para el aprendizaje del lenguaje musical

Subtítulo del proyecto.

Autor

Sergio Hervás Cobo

Directores

Luis López Escudero
Germán Arroyo Moreno

Aplicación multiplataforma para el aprendizaje del lenguaje musical: Subtítulo del proyecto

Sergio Hervás Cobo

Palabras clave: palabra_clave1, palabra_clave2, palabra_clave3,

Resumen

Poner aquí el resumen.

Project Title: Project Subtitle

Sergio Hervás Cobo

Keywords: Keyword1, Keyword2, Keyword3,

Abstract

Write here the abstract in English.

Yo, **Nombre Apellido1 Apellido2**, alumno de la titulación **TITULACIÓN de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI XXXXXXXXX, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Nombre Apellido1 Apellido2

Granada a X de mes de 201 .

D. **Nombre Apellido1 Apellido2 (tutor1)**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

D. **Nombre Apellido1 Apellido2 (tutor2)**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Título del proyecto, Subtítulo del proyecto***, ha sido realizado bajo su supervisión por **Nombre Apellido1 Apellido2 (alumno)**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 201 .

Los directores:

Nombre Apellido1 Apellido2 (tutor1) **Nombre Apellido1 Apellido2 (tutor2)**

Agradecimientos

Poner aquí agradecimientos...

Índice general

1. Introducción	1
1.1. Motivacion	2
1.2. Objetivos	3
2. Estado del Arte	5
2.1. Software a desarrollar	5
2.1.1. Aplicaciones de aprendizaje	5
2.1.2. Aplicaciones de aprendizaje musical	10
2.2. Desarrollo de Software	12
2.2.1. Flutter	12
2.2.2. React Native	12
2.2.3. Dart	13
2.2.4. Kotlin	13
2.2.5. NodeJS	14
2.2.6. React	14
2.2.7. Angular	15
2.2.8. MongoDB	15
2.2.9. Pilas MERN & MEAN	16
2.2.10. MariaDB	16
3. Especificación de requisitos	19
3.1. Introducción	19
3.1.1. Propósito	19
3.1.2. Ámbito del Sistema	19
3.1.3. Definiciones, Acrónimos y Abreviaturas	20
3.1.4. Referencias	20
3.1.5. Visión General del Documento	20
3.2. Descripción General	21
3.2.1. Perspectiva del producto	21
3.2.2. Funciones del producto	22
3.2.3. Características de los usuarios	23
3.2.4. Restricciones	23
3.2.5. Suposiciones y dependencias	23

3.3. Requisitos Específicos	23
3.3.1. Interfaz de usuario	24
3.3.2. Requisitos funcionales	25
3.3.3. Requisitos no funcionales	26
4. Planificación	29
4.1. Introducción	29
4.2. Velocidad	31
4.3. Product Backlog	32
4.4. Sprints	34
4.4.1. Sprint #1 - Documentación inicial	34
4.4.2. Sprint #2 - Documentación de requisitos	34
4.4.3. Sprint #3 - Documentación de HU	35
4.4.4. Sprint #4 - Diseño y comienzo del desarrollo	35
4.4.5. Sprint #5 - Registro y login de usuarios	35
4.4.6. Sprint #6 - Tests y progreso	36
4.4.7. Sprint #7 - Entrada de micrófono y detección de tono	36
4.4.8. Sprint #8 - Funcionalidades del profesor con lecciones	37
4.4.9. Sprint #9 - Funcionalidades del profesor con tests .	37
4.4.10. Sprint #10 - Funcionalidades del administrador y ranking	38
4.4.11. Sprint #11 - Funcionalidades poco prioritarias y valor añadido	38
4.4.12. Sprint #12 - Mejoras de código	38
4.4.13. Sprint #13 - Pruebas	39
4.4.14. Sprint #14 - Conclusión y finalización del proyecto .	39
4.5. Diagrama de Gantt	39
5. Análisis del problema	43
5.1. Introducción	43
5.2. Historias de Usuario	43
5.3. Diagrama de Clases	48
6. Diseño	51
6.1. Introducción	51
6.2. Arquitectura del sistema	51
6.3. Diagrama de base de datos	52
6.4. Diagrama de clases	54
6.5. Diagramas de secuencia	54
6.6. Diseño de interfaces de usuario	56

7. Implementación	65
7.1. Herramientas utilizadas	65
7.1.1. Git & Github	65
7.1.2. Jira	66
7.1.3. Visual Studio Code & Android Studio	66
7.1.4. Mongo Compass	67
7.1.5. Postman	67
7.2. Estructura del proyecto	68
7.3. Definición de la API (Backend)	69
7.3.1. Rutas	69
7.3.2. Middleware	71
7.4. Funcionalidad de inicio	73
7.4.1. Drawer	73
7.4.2. Sesión	73
7.5. Funcionalidad de usuario	81
7.5.1. Lista de lecciones	81
7.5.2. Ver lección	82
7.5.3. Test	85
7.5.4. Revisar pregunta	93
7.5.5. Historial de tests	98
7.5.6. Ver Mi Perfil	98
7.5.7. Ver logro	99
7.5.8. Mis logros	100
7.6. Funcionalidad de gestión de la aplicación	102
7.6.1. Dashboard	102
7.6.2. Lista de lecciones	103
7.6.3. Borrar lección	104
7.6.4. Crear lección	105
7.6.5. Editar lección	108
7.6.6. Lista de preguntas	110
7.6.7. Borrar pregunta	111
7.6.8. Crear Pregunta	112
7.6.9. Editar Pregunta	117
7.6.10. Lista de usuarios	120
7.6.11. Borrar usuario	120
7.6.12. Crear usuario	122
7.6.13. Editar usuario	124
7.6.14. Lista de logros	125
7.6.15. Borrar logro	125
7.6.16. Crear logro	127
7.6.17. Editar logro	129

8. Pruebas	131
8.1. Paquetes utilizados	131
8.2. Pruebas de unidad	131
8.2.1. Lecciones	132
8.2.2. Preguntas	132
8.2.3. Usuarios	134
8.2.4. Logros	135
8.3. Pruebas de controlador o widget	136
8.3.1. Botón de inicio de sesión	136
8.3.2. Botón de acceso a lista de lecciones del profesor	136
8.3.3. Botón de acceso a lista de logros del profesor	136
8.3.4. Botón de acceso a lista de preguntas del profesor	137
8.3.5. Botón de acceso a lista de usuarios	137
8.3.6. Botón de acceso al perfil del usuario	137
8.3.7. Botón de acceso a una lección	138
8.3.8. Botón de acceso al inicio	138
8.3.9. Botón de acceso al dashboard	138
8.4. Pruebas de sistema o de integración	139
8.4.1. Creación de usuario	139
8.4.2. Edición de datos del perfil	141
8.4.3. Realización de un test	142
9. Conclusiones	145
9.1. Valoración personal	146
9.2. Trabajo futuro	146

Índice de figuras

2.1.	Logo de Duolingo	5
2.2.	Ejercicio de Duolingo donde el usuario debe adivinar qué palabra corresponde al audio	6
2.3.	Menú de selección de las lecciones de Artly clasificadas en movimientos artísticos	7
2.4.	Ciudad de BMath donde se seleccionan los niveles y se construyen los edificios para mostrar el progreso del usuario	8
2.5.	Ejercicio de BMath sobre cálculo matemático (sumas y restas) con calculadora, con papel y mentalmente	9
2.6.	Lista de lecciones que ofrece la aplicación Curso de Lenguaje Musical	10
2.7.	Logo de Flutter	12
2.8.	Logo de React Native	12
2.9.	Logo de Dart	13
2.10.	Logo de MongoDB	15
2.11.	Pilas MEAN y MERN	16
2.12.	Logo de MariaDB	16
3.1.	Diagrama general de la perspectiva del producto a desarrollar.	21
3.2.	Primer boceto de la interfaz de usuario de la página de inicio de la aplicación para los usuarios, donde se muestra la lista de las lecciones	24
4.1.	Diagrama de la arquitectura del sistema, donde se muestra la comunicación entre el servidor y la aplicación móvil.	30
4.2.	Diagrama de Gantt de la planificación del proyecto, divivido por Sprints.	40
4.3.	Diagrama de Gantt de los cuatro primeros Sprints.	41
4.4.	Diagrama de Gantt de los Sprints 5, 6, 7 y 8.	41
4.5.	Diagrama de Gantt de los Sprints 9, 10 y 11.	42
4.6.	Diagrama de Gantt de los Sprints 12, 13 y 14.	42
5.1.	Diagrama de clases de nuestro proyecto donde se muestran las relaciones entre las diferentes clases que componen el sistema.	48

6.1.	Diagrama de la arquitectura del sistema, donde se muestra la comunicación entre el servidor y la aplicación móvil.	52
6.2.	Diagrama de base de datos del sistema, donde se muestran los documentos que se van a almacenar en nuestra base de datos de MongoDB.	53
6.3.	Diagrama de clases del sistema donde se detallan las propiedades y las relaciones de las distintas clases o entidades que tendrá el software.	54
6.4.	Diagrama de secuencia de un usuario registrándose, el cual interactuará con la vista para poder hacer la petición al controlador y guardar su usuario en el modelo.	55
6.5.	Diagrama de secuencia de un usuario iniciando sesión, el cual interactuará con la vista para poder hacer la petición al controlador y buscar su usuario en el modelo para la validación. Tras esto, se cambiará la vista a la pantalla principal que lista las lecciones, por lo que la vista deberá hacer otra petición al controlador de lecciones para obtener las lecciones del modelo.	55
6.6.	Boceto de la pantalla de inicio de sesión, donde se pedirá al usuario el correo electrónico y la contraseña.	56
6.7.	Boceto de la pantalla de registro, donde se pedirá al usuario sus datos personales necesarios para crear la cuenta como el nombre, los apellidos, el correo y la contraseña.	57
6.8.	Boceto de la pantalla principal de la aplicación donde se muestran las lecciones disponibles para el usuario.	57
6.9.	Boceto de la pantalla del perfil de usuario con sus datos. . . .	58
6.10.	Boceto de la pantalla de logros conseguidos por el usuario. . . .	58
6.11.	Boceto de la pantalla de una lección, con el texto, el contenido multimedia y el botón para comenzar el test.	59
6.12.	Boceto de la pantalla de una pregunta de test de tipo selección única.	60
6.13.	Boceto de la pantalla de una pregunta de test de tipo selección multiple.	61
6.14.	Boceto de la pantalla de una pregunta de test de tipo escritura de texto.	62
6.15.	Boceto de la pantalla de una pregunta de test de tipo entrada por micrófono.	63
6.16.	Diagrama de navegación de la aplicación por los distintos bocetos presentados anteriormente que muestra las pantallas que seguirá el usuario cuando utilice la aplicación.	64
7.1.	Captura de pantalla de Jira. En ella se puede ver el Sprint Backlog, donde se encuentran las historias de usuario repartidas entre los sprints.	66

7.2. Captura de pantalla de Jira. En ella se puede ver el Sprint Backlog, donde se encuentran las historias de usuario repartidas entre los sprints.	67
7.3. Estructura del proyecto, donde se puede ver la parte de frontend y la parte de backend.	68
7.4. Drawer abierto en la aplicación con tres opciones (por ser desde la vista del profesor): Inicio, Dashboard y Cerrar sesión.	73
7.5. Diagrama que explica el proceso de encriptación en el inicio de sesión desarrollado en la aplicación.	74
7.6. Diagrama que explica el proceso de intercambio de token en el inicio de sesión desarrollado en la aplicación.	75
7.7. Diagrama que explica el proceso de sesión entre instancias en el inicio de sesión desarrollado en la aplicación.	76
7.8. Pantalla de inicio de sesión del usuario donde se puede ver el formulario de inicio de sesión con los campos de usuario (correo) y contraseña.	77
7.9. Pantalla de registro de los usuarios donde se puede ver el formulario de registro con los campos de usuario (correo), contraseña y repetir la contraseña.	79
7.10. Pantalla de la lista de lecciones con el título y la imagen de cada lección.	81
7.11. Pantalla de una lección en concreto con los contenidos (títulos, texto, multimedia...) del temario y los dos botones relacionados con los tests.	83
7.12. Pantalla de una lección en concreto con los contenidos (títulos, texto, multimedia...) del temario y los dos botones relacionados con los tests.	84
7.13. Pantalla de una pregunta de selección única en la que las opciones se muestran en forma de lista.	87
7.14. Pantalla de una pregunta de selección múltiple en la que las opciones se muestran en forma de lista.	88
7.15. Pantalla de una pregunta de entrada de texto.	89
7.16. Pantallas de una pregunta de micrófono.	90
7.17. Pregunta de micrófono con respuesta introducida.	91
7.18. Pantalla que muestra el resultado del test con el número de aciertos y el porcentaje de aciertos respecto a las preguntas con un Progress Bar.	92
7.19. Pantalla de la pregunta de selección única en revisión.	94
7.20. Pantalla de la pregunta de selección múltiple en revisión.	95
7.21. Pantalla de la pregunta de entrada de texto en revisión.	96
7.22. Pantalla de la pregunta de entrada de micrófono en revisión.	97
7.23. Pantalla de dashboard del profesor con las opciones de gestión disponibles para el profesor.	98
7.24. Pantalla del perfil del usuario.	99

7.25. Pantalla de visualización de logro asignado.	100
7.26. Pantalla de visualización de la lista de logros asignados.	101
7.27. Vista del dashboard	102
7.28. Vista de la lista de las lecciones.	103
7.29. Pantalla del borrado de una lección.	104
7.30. Pantalla de la creación de lecciones sin datos introducidos.	106
7.31. Pantalla de la creación de lecciones con datos introducidos.	107
7.32. Pantalla de la edición de lecciones existentes por parte del profesor.	109
7.33. Pantalla de lista de preguntas del profesor para la gestión de estas.	110
7.34. Pantalla del borrado de una pregunta.	111
7.35. Pantalla de creación de pregunta de tipo 'única' vacía. La pregunta de tipo múltiple y de tipo micrófono son iguales.	113
7.36. Pantalla de creación de pregunta de tipo 'única' con datos introducidos.	114
7.37. Pantalla de creación de pregunta de tipo 'múltiple' con datos introducidos.	114
7.38. Pantalla de creación de pregunta de tipo 'micrófono' con datos introducidos.	115
7.39. Pantalla de creación de pregunta de tipo 'texto'.	115
7.40. Pantalla de edición de pregunta de tipo 'única'.	117
7.41. Pantalla de edición de pregunta de tipo 'múltiple'.	118
7.42. Pantalla de edición de pregunta de tipo 'micrófono'.	118
7.43. Pantalla de edición de pregunta de tipo 'texto'.	119
7.44. Pantalla de lista de usuarios del administrador para la gestión de estos.	120
7.45. Pantalla de borrado de un usuario.	121
7.46. Pantallas de creación de usuarios.	123
7.47. Pantalla de edición de usuario existente.	124
7.48. Pantalla de lista de logros del administrador.	125
7.49. Pantalla del borrado de un logro.	126
7.50. Pantallas de creación de logros.	128
7.51. Pantalla de edición de logro existente.	130
8.1. Resulados de las pruebas de unidad de las lecciones	132
8.2. Resulados de las pruebas de unidad de las preguntas	134
8.3. Resulados de las pruebas de unidad de los usuarios	135
8.4. Resulados de las pruebas de unidad de los logros	135
8.5. Código de la prueba de controlador o widget de acceso a lista de logros	137
8.6. Código de la prueba de controlador o widget de acceso a la pantalla de lecciones	138
8.7. Resultados de las pruebas de controlador o widget	139

8.8. Pantallas por las que pasa la prueba de creación de usuario .	140
8.9. Pantallas por las que pasa la prueba de modificación de datos del perfil	141
8.10. Pantallas por las que pasa la prueba de realización de un test	143

Capítulo 1

Introducción

Durante toda la historia de la humanidad, la música ha sido siempre una parte muy importante de la vida de la mayoría de personas y, a día de hoy, es una pieza clave y fundamental en nuestra sociedad y tradiciones. Escuchamos canciones a todas horas: mientras andamos por la calle, cuando cocinamos, mientras hacemos ejercicio, conduciendo en el coche, etc. La música está presente en gran parte de nuestros hábitos y de nuestra cultura y es por esto que, además de ser una de las siete bellas artes, mucha gente se dedica a estudiarla y a aprender sobre esta.

Sin embargo, por desgracia, a lo largo de la historia el aprendizaje del lenguaje musical ha sido elitista y la mayoría de personas con pocos recursos económicos no ha podido acceder a gran parte de este conocimiento, ya que antiguamente solo las familias nobles y adineradas podían darse el lujo de disfrutar de las melodías clásicas. Puede parecer que no pero, incluso en la actualidad, la formación musical a veces puede ser vista como un privilegio reservado para unos pocos, pues solo quienes pueden permitirse ir a una escuela de música o conservatorio pueden adquirir esos conocimientos.

Por otro lado, pese a ser un aspecto fundamental en nuestras vidas, el aprendizaje de esta disciplina es a menudo visto como algo aburrido debido a clases desactualizadas, poco motivadoras y enfocadas únicamente a un género musical en específico: la música clásica.

Todo esto plantea las siguientes cuestiones, ¿es posible democratizar el aprendizaje musical? ¿Ha habido un cambio en la forma en la que se estudia y/o practica música con la aparición de los dispositivos móviles, de internet y de las aplicaciones software? ¿Es posible aprender música de forma autodidacta, divertida y gratuita?

1.1. Motivacion

El surgimiento de nuevas aplicaciones que ofrecen formas de aprendizaje lúdicas y divertidas en distintos ámbitos como idiomas, matemáticas, arte, geografía, etc, me ha hecho pensar en lo útiles que pueden llegar a ser estas para personas con pocos recursos que buscan una forma rápida, barata y sencilla de aprender sobre algún tema en específico. Además, el acceso a la música es algo fundamental que todas las personas deberían tener al alcance y siempre lo he creído así.

En cuanto al aprendizaje musical, al estar relacionado con la música clásica y tradicional, suele ser visto como algo aburrido, por lo que muchos no se llegan a interesar por esto o suelen abandonar su formación en mitad. Sin infravalorar las clases de música tradicionales, el mundo está cambiando y existe una sensación de que la docencia musical se ha quedado atrás y no se ha adaptado a las necesidades de los jóvenes ni de la sociedad actual. Habiendo pocas opciones que ofrezcan un aprendizaje de la música de forma divertida y lúdica, muchas personas pierden una oportunidad valiosa de llegar a un mundo muy valioso.

Por otro lado, mi experiencia estudiando en una escuela de música durante 6 años me ha hecho entender que el aprendizaje musical, pese a ser un proceso complejo y exigente, es gratificante y satisfactorio; y que cualquier persona, tenga o no recursos económicos, debería tener la oportunidad de poder aprender música y disfrutar de ella sin impedimentos. La música y la formación en esta debería ser accesible para cualquier persona.

Al unir el surgimiento de distintas aplicaciones software educativas que ofrecen una visión novedosa y entretenida del aprendizaje con mi experiencia en el conservatorio de música, se me ocurrió la idea de desarrollar un sistema de aprendizaje musical que permita a cualquier persona, independientemente de su nivel original de conocimientos, aprender conceptos básicos e intermedios sobre lenguaje musical, así como la posibilidad de instruirse para empezar a tocar instrumentos musicales con un enfoque didáctico, divertido y progresivo.

Todo esto y la falta de aplicaciones similares y de calidad en el mercado respecto al conocimiento musical, han convertido en una necesidad el desarrollo de este proyecto.

1.2. Objetivos

Se pretende desarrollar una aplicación software que permita al usuario aprender conceptos de lenguaje musical de forma autodidacta y sencilla, proporcionando un enfoque que anime a las personas a involucrarse en la adquisición de conocimientos y a entender la mayoría de conceptos que la envuelven, desde el lenguaje musical hasta la práctica de un instrumento.

También se desea un sistema que proporcione una metodología de enseñanza mediante gamificación; es decir, con contenidos lúdicos y divertidos y con un progreso organizado en logros y niveles que permita a los usuarios sentirse satisfechos con su progreso y motivados a seguir aprendiendo. Los usuarios podrán ver su progreso mediante un sistema de cuentas de usuario que permitirá también la posibilidad de compartir sus logros con otros usuarios.

Por último, otro de los objetivos será incorporar distintos tipos de ejercicios y actividades que faciliten este aprendizaje, como por ejemplo la inclusión de un sistema que detecte las notas musicales tocadas por el usuario o la incorporación de un teclado virtual simple y sencillo que permita al usuario practicar y aprender a tocarlo.

Capítulo 2

Estado del Arte

2.1. Software a desarrollar

Para tener una perspectiva más amplia y clara de lo que se pretende desarrollar, se ha realizado una búsqueda de software similar al de este proyecto, tanto de aprendizaje musical como de otros temas que utilicen las técnicas de gamificación y de aprendizaje lúdico.

2.1.1. Aplicaciones de aprendizaje

En la actualidad existen numerosas aplicaciones en el mercado que ayudan al aprendizaje de conceptos y a la adquisición de conocimientos sobre distintos temas, como los idiomas, las matemáticas o el arte, entre otros. Es tan grande el auge de este tipo de herramientas que muchas de ellas están siendo recomendadas por el colectivo docente como apoyo a los contenidos que se dan en clase. A continuación se detallarán las aplicaciones más destacadas encontradas:

2.1.1.1. Duolingo

Duolingo es una de las aplicaciones más populares que existen para aprender idiomas. Esta aplicación se basa en el método de aprendizaje por inmersión y en la gamificación: el usuario estudia el idioma mediante una serie de actividades lúdicas. Aunque su principal motivación es la enseñanza de hablar y escribir en un idioma, también se incluyen actividades de comprensión auditiva, de lectura y de vocabulario.



Figura 2.1: Logo de Duolingo

La aplicación se divide en lecciones, las cuales contienen una serie de ejercicios (Figura 2.2) que el usuario debe completar para ir avanzando por las distintas secciones.

Algunos de los ejercicios que presenta la aplicación son:

- Traducción de palabras o frases con el teclado
- Traducción seleccionando bloques de palabras
- Pronunciación de palabras o frases mediante el micrófono del dispositivo
- Selección de imagen para el vocabulario

Además, Duolingo ofrece en cada lección, antes de los ejercicios, una breve explicación de la gramática o vocabulario con imágenes y definiciones sencillas, que ayudarán al usuario a entender el temario antes de comenzar la evaluación de la lección. En cuanto al diseño, la herramienta ofrece una interfaz sencilla y fácil de usar, con botones intuitivos y coloridos que mejoran la experiencia de usuario, algo primordial en este tipo de aplicaciones. Por último, cabe destacar que esta aplicación es gratuita (aunque ofrece funcionalidad adicional de pago dentro de esta) y está disponible para dispositivos móviles (Android, iOS y Windows Phone).

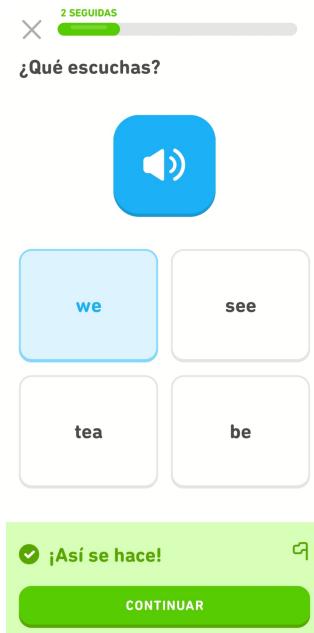


Figura 2.2: Ejercicio de Duolingo donde el usuario debe adivinar qué palabra corresponde al audio

2.1.1.2. Artly

Artly es una aplicación móvil que permite aprender cientos de obras de arte de forma interactiva. De forma parecida a Duolingo, cuando el usuario abre la aplicación, se encuentra una lista de secciones (Figura 3.2) correspondientes a los distintos movimientos artísticos. Estos apartados se irán desbloqueando uno a uno a medida que el usuario vaya completando las lecciones y ejercicios.

En cada uno de los apartados el usuario podrá ver una lista de cuadros y esculturas de los distintos artistas que pertenecen a ese movimiento. Al seleccionar uno de ellos, se abrirá una pantalla con la imagen del cuadro o escultura, en la que el usuario podrá ver la obra más de cerca, junto con el título, el autor y una descripción. Artly incorpora además una serie de preguntas sobre las obras vistas en dicho apartado (seleccionar el autor, seleccionar el título de la obra...). Al finalizar las preguntas, el usuario sumará un progreso en el tema que permitirá desbloquear otros apartados y conseguir ciertos logros en su perfil. La aplicación es gratuita pero tiene mejoras de pago que te permitirán avanzar con más facilidad y sin anuncios.



Figura 2.3: Menú de selección de las lecciones de Artly clasificadas en movimientos artísticos

2.1.1.3. BMATH

BMATH es un software para móviles para el aprendizaje de conceptos matemáticos mediante gamificación. Esta aplicación crea un programa personalizado para el usuario en función del curso académico en el que esté, adaptándose a su conocimiento y a su nivel. Pese a ser gratuita, el tiempo diario que te permite el modo básico es de 5 minutos, mientras que si pagas te permitirá ampliarlo en 10 minutos cada día.

Es un ejemplo de aprendizaje por gamificación ya que la aplicación es prácticamente un videojuego (de hecho, se consideran así), pues muestra una ciudad (Figura 2.4) por la que puedes mover la cámara y donde construyes distintos edificios. Para construirlos, deberás superar una serie de pruebas sobre álgebra, geometría, etc. También obtendrás puntos de experiencia al superar los ejercicios (Figura 2.5) y, al subir de nivel, recibirás nuevos edificios y decoraciones para tu ciudad. Con esto, se pretende que el usuario tenga la motivación de realizar ejercicios y superarlos correctamente para lograr que su ciudad prospere. Como alumno también puedes elegir tu propio personaje animado que te identificará en el juego.

Los diseños del software están muy cuidados y posee colores muy llamativos para que el usuario se sienta motivado y atraido. Además, la aplicación proporciona una experiencia de usuario agradable y motivadora, ayudando al alumno en todo momento a solucionar los ejercicios y enseñándole técnicas para resolverlos.

La aplicación también posee otras funciones como un apartado donde añadir recordatorios para acceder a la aplicación, preguntas para conocer cómo se siente el usuario tras cada sesión de estudio, sección parental, etc.



Figura 2.4: Ciudad de BMATH donde se seleccionan los niveles y se construyen los edificios para mostrar el progreso del usuario

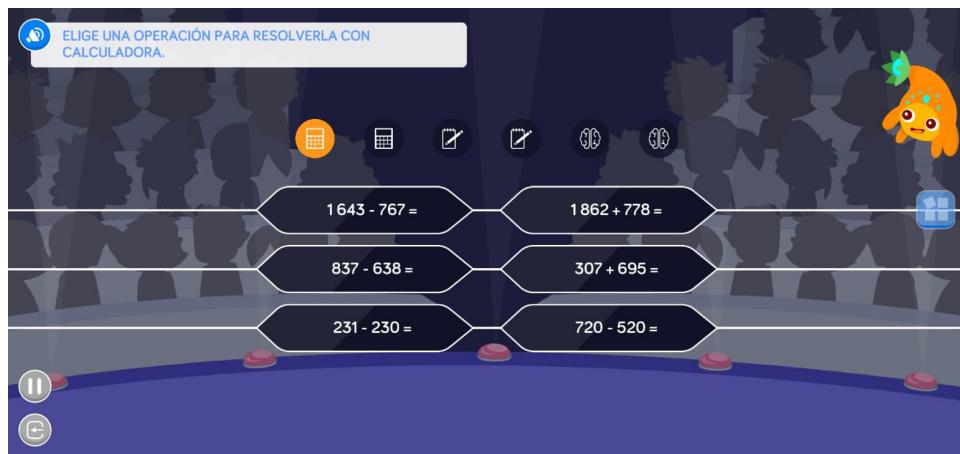


Figura 2.5: Ejercicio de BMATH sobre cálculo matemático (sumas y restas) con calculadora, con papel y mentalmente

2.1.2. Aplicaciones de aprendizaje musical

Centrándonos ya en el tema que nos ocupa (el aprendizaje musical) existen numerosas aplicaciones que ayudan a la adquisición de conocimientos musicales o a la ayuda a aprender a tocar un instrumento musical. A continuación se muestran algunas de ellas.

2.1.2.1. Curso de Lenguaje Musical

Esta herramienta para móvil tiene el objetivo de enseñar los conceptos básicos de lenguaje musical. La aplicación proporciona una forma de aprendizaje muy tradicional y que no está basada en la gamificación, ya que no dispone de ningún tipo de seguimiento del progreso ni de ejercicios para realizar sobre las lecciones aprendidas. La app contiene una lista de lecciones (Figura 2.6) y cada una de ellas muestra un vídeo de un docente explicando el contenido del temario.

Por último, en cuanto a la interfaz, la aplicación es muy sencilla y tiene un diseño poco atractivo, dando una simple lista con enlaces a los vídeos de las lecciones sin ningún tipo de decoración o imagen.



Figura 2.6: Lista de lecciones que ofrece la aplicación Curso de Lenguaje Musical

2.1.2.2. Sonid

Sonid consiste en una aplicación para el aprendizaje de piano a través de la gamificación. La aplicación se divide en distintas lecciones (Figura 2.7b) que el usuario irá desbloqueando al pasarlas correctamente. Cada lección explica unos conceptos básicos de música y piano y, al finalizar, el usuario deberá realizar una serie de ejercicios para comprobar que ha entendido y aprendido el contenido del temario.

Algunos de los ejercicios que posee esta aplicación son:

- Seleccionar la tecla en el piano que corresponde a una nota musical. (Figura 2.7a)
- Responder si la tecla que se ha pulsado es una nota musical o no.
- Decidir qué nota musical corresponde a una tecla del piano que se ha pulsado.

Además de las lecciones, el usuario también puede practicar con escalas y acordes del piano, así como aprenderlas en la wiki y en el diccionario que posee la aplicación. También dispone de un foro para que toda la gente pueda compartir sus dudas.

Por último, en cuanto al seguimiento, el usuario puede ver su progreso y sus estadísticas en su perfil, donde aparece el número de lecciones completadas, el número de errores que ha tenido, la experiencia que tiene. Además, el usuario consigue logros al completar las lecciones y comparte una clasificación global con otros usuarios que también usan la aplicación.

(a) Ejercicio de Sonid para conocer las notas de un piano

(b) Menú de selección de lecciones de Sonid dividido por escalas

2.2. Desarrollo de Software

En cuanto a las tecnologías y herramientas que existen para el desarrollo de aplicaciones, existen numerosas opciones que se pueden utilizar. En este apartado se muestran algunas de ellas.

2.2.1. Flutter

Flutter es un kit de desarrollo software (framework) de código abierto destinado al desarrollo de aplicaciones móviles multiplataforma. Está programado en Dart, fue creado por Google y su primera versión estable se lanzó en 2018. Se emplea para desarrollar aplicaciones para móvil, web y escritorio desde una sola base de código, lo cual permite mucha más agilidad y consistencia.

Flutter ofrece tres ventajas respecto a otros frameworks utilizados para el desarrollo de aplicaciones multiplataformas:

- **Compilación en nativo**
- **Flexibilidad** para crear interfaces gráficas
- **Desarrollo veloz**, permitiendo ver el resultado del código al instante.

Flutter está basado en el concepto de widgets, que son los elementos que componen la interfaz y la interacción del usuario. Estos widgets pueden ser modificados, añadidos o eliminados dinámicamente, permitiendo una gran flexibilidad a la hora de crear interfaces gráficas.

2.2.2. React Native

Framework de código abierto creado por Meta Platforms (Facebook) y destinado al desarrollo de aplicaciones multiplataforma para Android, iOS, Web, Windows, etc. Está basado en Javascript como lenguaje de programación y en React como herramienta para la interfaz de usuario. La diferencia con React es que, en lugar de trabajar con el navegador, trabaja con las plataformas móviles al transformar los componentes en nativos en función de la plataforma en la que se esté ejecutando la aplicación.

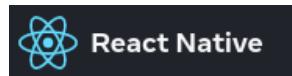


Figura 2.7: Logo de Flutter



Figura 2.8: Logo de React Native

Esta tecnología permite construir aplicaciones móviles usando solamente Javascript, con el mismo diseño que utiliza React permitiéndote realizar una interfaz de usuario completa mediante componentes.

Podemos encontrar dos ventajas en este framework:

- **Código compartido:** tu código puede ser compartido para diferentes plataformas.
- **Comunidad:** Existe una amplia comunidad sobre React y React Native, lo cual permite encontrar soluciones a los problemas que se puedan presentar en el desarrollo.

Como desventaja, encontramos una limitación en los componentes nativos, pues seguramente se necesite escribir algo de código específico para un plataforma en concreto

2.2.3. Dart

Dart es un lenguaje de programación de código abierto desarrollado por Google y utilizado por Flutter. Es orientado a objetos y de tipado estático. Surgió en 2011 como alternativa a Javascript.



Figura 2.9: Logo de Dart

Dart posee las siguientes ventajas:

- **Lenguaje fácil y sencillo** de aprender
- **Acceso gratuito**
- **Funciona en todas las plataformas**
- **Programación estructurada y flexible**
- Al ser **orientado a objetos**, facilita la encapsulación y reutilización de código gracias al uso de clases.

2.2.4. Kotlin

Kotlin es un lenguaje de programación construido sobre Java, desarrollado por JetBrains y lanzado en 2016. Es orientado a objetos y tiene tipado estático, pero también soporta la programación por procedimientos y funciones.

Algunas de sus ventajas son:

- **Compatibilidad con Java:** permite utilizar código Java en Kotlin y viceversa
- **Lenguaje fácil y sencillo** de aprender y usar por su sintaxis similar a Java
- **Robusto:** Es un lenguaje seguro con los valores nulos
- **Exactitud y claridad:** reduce el código repetido de forma sustancial y disminuye la probabilidad de error.

2.2.5. NodeJS

NodeJS es un entorno en tiempo de ejecución de código abierto, del lado del servidor y programado en Javascript. Es asíncrono y dirigido por eventos. Node es utilizado para el desarrollo de aplicaciones de red escalables y rápidas, ofreciendo beneficios en rendimiento, velocidad de desarrollo, etc.

Se utiliza en el desarrollo de aplicaciones web porque permite ejecutar Javascript del lado del servidor en lugar de en el navegador.

Los principales motivos para utilizar NodeJS son:

- **Simultaneidad de peticiones:** NodeJS es capaz de manejar múltiples peticiones al mismo tiempo gracias al modelo dirigido por eventos, lo que permite una mayor escalabilidad.
- **Lenguaje sencillo** basado en Javascript.
- **Gestión de paquetes de calidad** gracias a NPM.

2.2.6. React

React es una biblioteca Javascript de código abierto que se utiliza para la creación de interfaces de usuario de forma fácil y sencilla. Es mantenida por una gran comunidad de desarrolladores de software libre y fue lanzada en 2013.

En React se trabaja con componentes, los cuales son elementos o partes de la interfaz de usuario que facilitarán la reutilización de código y la modularidad.

Entre las ventajas de React encontramos:

- **Componentes reutilizables:** React permite crear componentes que pueden ser reutilizados en otras partes de la aplicación.
- **Fácil de aprender:** React es fácil de aprender y de utilizar, además de tener una documentación sólida y una gran cantidad de recursos online gratuitos.
- **Rendimiento:** React es rápido y eficiente gracias al DOM virtual.

2.2.7. Angular

Angular es un framework de código abierto para el desarrollo de aplicaciones web y móviles. Está basado en TypeScript y está desarrollado por Google. Esta tecnología también está basada en componentes, lo cual permite la creación de aplicaciones web escalables

2.2.8. MongoDB

MongoDB es una base de datos no relacional de código abierto y orientada a documentos. Es de tipo NoSQL, es decir, que no utiliza el modelo relacional de bases de datos tradicionales. Cada registro de la base de datos es un documento que consta de pares clave - valor (muy similar a los objetos JSON). Está escrito en C++ y las consultas se hacen pasando objetos JSON como parámetros.

MongoDB proporciona las siguientes ventajas:

- **Rendimiento:** MongoDB es más rápido porque almacena los datos directamente en un mismo sitio (colección).
- **Simplicidad:** No hay que seguir un esquema ni unir tablas como en SQL.
- **Flexibilidad** para diseñar el esquema y crear las relaciones.



Figura 2.10: Logo de MongoDB

2.2.9. Pilas MERN & MEAN

MERN es un conjunto de tecnologías que se utilizan para el desarrollo de aplicaciones web. Está compuesto por las siglas de MongoDB, Express (ayuda para crear y gestionar el backend en Node), React y NodeJS. MEAN es igual, pero sustituye React por Angular.

Hay una gran cantidad de beneficios al utilizar una de estas pilas para desarrollar una aplicación web como por ejemplo:

- **Cubre todo el ciclo de desarrollo:** Desde el backend hasta el frontend.
- **Facilita el trabajo** con la arquitectura modelo-vista-controlador.
- **Rápido desarrollo:** El desarrollo de una aplicación web con estas tecnologías es rápido y eficiente.
- **Código abierto:** Todas las tecnologías que componen estas pilas son de código abierto y gratuitas y están respaldadas por la comunidad.
- **Fácil mantenimiento:** El mantenimiento de una aplicación web con estas tecnologías es fácil y sencillo.



Figura 2.11: Pilas MEAN y MERN

2.2.10. MariaDB

MariaDB es un sistema de gestión de bases de datos relacional de código abierto. Es un fork de MySQL y está desarrollado por su comunidad de usuarios. Surgió en 2009 como consecuencia de la compra de MySQL por parte de Sun Microsystems. Sigue un modelo relacional, lo cual significa que el contenido de los datos se almacena en tablas con un esquema más rígido que en MongoDB u otra base de datos noSQL.



Figura 2.12: Logo de MariaDB

Pese a haber mencionado solamente MariaDB como ejemplo de sistema de gestión de bases de datos relacional, existen muchos otros como por ejemplo PostgreSQL, MySQL, etc.

Capítulo 3

Especificación de requisitos

3.1. Introducción

En esta sección se describirá qué sistema vamos a construir y cómo lo vamos a hacer, con sus restricciones específicas. Esto se realizará especificando y describiendo los requisitos del sistema que vamos a desarrollar.

3.1.1. Propósito

En este capítulo se pretende describir de forma clara y precisa las funciones, características y restricciones del sistema que se va a desarrollar. Estas definiciones servirán al equipo de desarrollo para conocer las necesidades del sistema y a los usuarios finales. Además, este capítulo será consultado como base para el desarrollo de las funcionalidades descritas en la sección de implementación.

3.1.2. Ámbito del Sistema

El sistema que vamos a desarrollar es una aplicación multiplataforma llamada "Meloudy" que permitirá a los usuarios aprender conceptos musicales de una forma amena, fácil y divertida. Esto se logrará mediante el método de gamificación, utilizando un sistema de logros y de recompensas con la superación de los distintos ejercicios y actividades de distintos tipos que el usuario deberá completar. Además, la aplicación llevará el progreso de los usuarios que les permitirá saber cómo van en su aprendizaje.

3.1.3. Definiciones, Acrónimos y Abreviaturas

A continuación se detallará el significado de algunos conceptos importantes para la comprensión del capítulo y de nuestro sistema.

- **Requisito:** Es una condición o característica que debe cumplir el sistema para satisfacer una necesidad o cumplir una función.
- **Funcionalidad:** Descripción de lo que debe hacer el producto software.
- **Restricción:** Condición que limita la funcionalidad del sistema.
- **Interfaz de usuario:** Requisitos que describen los diseños de pantallas que utilizará el usuario para utilizar el software.
- **Usuario:** Persona que utilizará la aplicación para la intención final de esta.
- **Administrador:** Persona encargada del sistema software y del mantenimiento de este para su buen funcionamiento.
- **RF / RNF:** Requisito funcional / Requisito No Funcional.

3.1.4. Referencias

Este capítulo de Especificación de requisitos ha sido redactada consultando los documentos del estándar IEEE Recommended Practice for Software Requirements Specification ANSI/IEEE 830, 1998.

3.1.5. Visión General del Documento

La Especificación de Requisitos consta de tres partes bien diferenciadas:

- **Introducción:** Proporciona una visión general sobre el apartado de Especificación de Requisitos sin profundizar en los requisitos como tal.
- **Descripción General:** Se describirá el sistema a construir para saber las funciones principales, los datos necesarios, las restricciones y otros aspectos que puedan afectar al desarrollo de la aplicación.
- **Requisitos Específicos:** Se profundiza en las necesidades del usuario definiendo los requisitos que debe tener nuestro sistema tras el desarrollo y la implementación de este.

3.2. Descripción General

A continuación, se procederá a describir la visión del sistema y los factores que afectan al producto de una forma general.

3.2.1. Perspectiva del producto

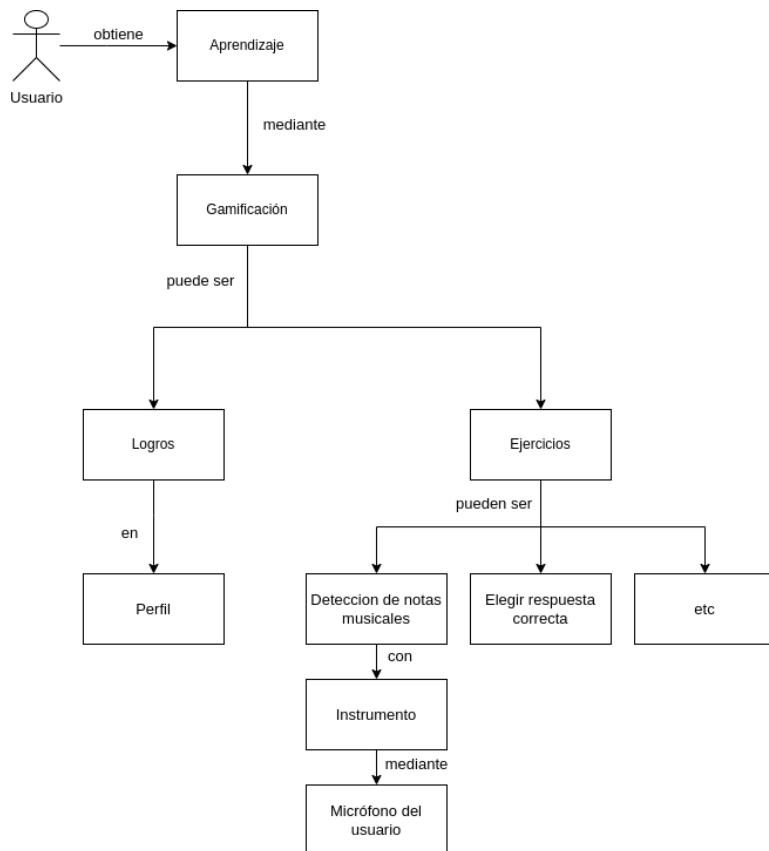


Figura 3.1: Diagrama general de la perspectiva del producto a desarrollar.

Se pretende implementar un sistema que permita el aprendizaje del usuario mediante técnicas de gamificación y ejercicios a resolver. Algunas de estas actividades utilizarán librerías para la detección de notas musicales a partir de la frecuencia captada por el micrófono del usuario y, por tanto, la aplicación dependerá de dichas librerías que se incluyan. Por otro lado, el sistema de administración y el progreso de los usuarios se llevará a cabo utilizando una base de datos en la que se almacenará la información necesaria de los usuarios y su progreso (con un sistema de logros asociados al perfil).

La interacción de los usuarios con la aplicación será mediante una interfaz

gráfica que podrá ser utilizada con la pantalla táctil o el ratón del dispositivo usado.

3.2.2. Funciones del producto

Las principales funciones que el usuario podrá realizar dentro de la aplicación son:

- Selección de lecciones a aprender.
- Respuesta (mediante selección o escritura) de las preguntas y actividades que ofrezca la aplicación.
- Consulta del progreso individual, de los logros obtenidos y de la clasificación global.
- Modificación de los datos del perfil.

Además, el encargado/profesor se encargará de parte de la administración de las lecciones mediante:

- Modificación y creación del texto de cada lección
- Gestión de contenido multimedia de cada lección (modificación, creación y eliminación)
- Gestión de las preguntas y actividades (modificación, creación y eliminación)
- Creación de nuevas lecciones
- Eliminación de lecciones

Por último, el administrador, además de las funcionalidades del encargado/profesor, podrá realizar las siguientes funcionalidades

- Creación de nuevos usuarios
- Modificación de los datos de los usuarios
- Borrado de usuarios
- Creación de nuevas lecciones
- Borrado de lecciones

3.2.3. Características de los usuarios

Los usuarios que usarán la aplicación tendrán distintos perfiles y abarcarán edades muy distintas. Aún así, el perfil objetivo para el uso del sistema será las personas jóvenes, pues suelen utilizar mucho más las nuevas tecnologías y estarán más familiarizados con este tipo de aplicaciones. No se necesita conocimiento musical previo para utilizar el software y, de hecho, las lecciones pueden ser bastante básicas y sencillas para las personas que ya conozcan conceptos y aspectos avanzados del lenguaje musical.

3.2.4. Restricciones

Las restricciones del sistema a desarrollar son:

- La aplicación del cliente estará desarrollada en Flutter y/o Dart y para la del servidor se utilizará NodeJS con Express.
- Se utilizarán las consultas, inserciones, modificaciones y borrados de MongoDB para la base de datos, usando un modelo no relacional.
- Al hacer uso de librerías externas para desarrollar algunas de las funciones, el rendimiento y las limitaciones hardware pueden depender de estas.

Además, es posible que el sistema tenga que adaptarse a los cambios del entorno y necesite agregar con el paso del tiempo funcionalidades que añadan valor al producto.

3.2.5. Suposiciones y dependencias

Los requisitos especificados en este capítulo pueden estar sujetos a cambios por motivos técnicos, de alcance o de refinamiento por la necesidad de la adaptación al entorno y a la evolución continua del proyecto.

No habrá ninguna suposición respecto al sistema operativo a utilizar puesto que pretendemos desarrollar una aplicación multiplataforma, aunque se priorizará el buen funcionamiento en Android y en navegadores web.

Asumiremos también que las bibliotecas a utilizar para el desarrollo de ciertos requisitos funcionan correctamente y son estables.

3.3. Requisitos Específicos

A continuación se listarán todos los requisitos que debe cumplir el sistema a desarrollar. Se dividirán en distintos tipos en función de sus objetivos.

3.3.1. Interfaz de usuario

Se tratará que la interfaz de usuario siga un diseño minimalista y sencillo para facilitar el uso de la aplicación a todo tipo de personas y garantizar cierto grado de accesibilidad. En cuanto a los colores, se usarán tonalidades de azul junto con grises. En móviles la pantalla estará dividida en dos partes principales:

- La cabecera con el título de la aplicación, un menú estilo hamburguesa que se abrirá lateralmente y un ícono para el acceso al perfil del usuario.
- El cuerpo, donde se encuentra el contenido principal de la sección correspondiente.

A continuación se presenta un primer boceto de lo que podría ser la interfaz de usuario para móviles.

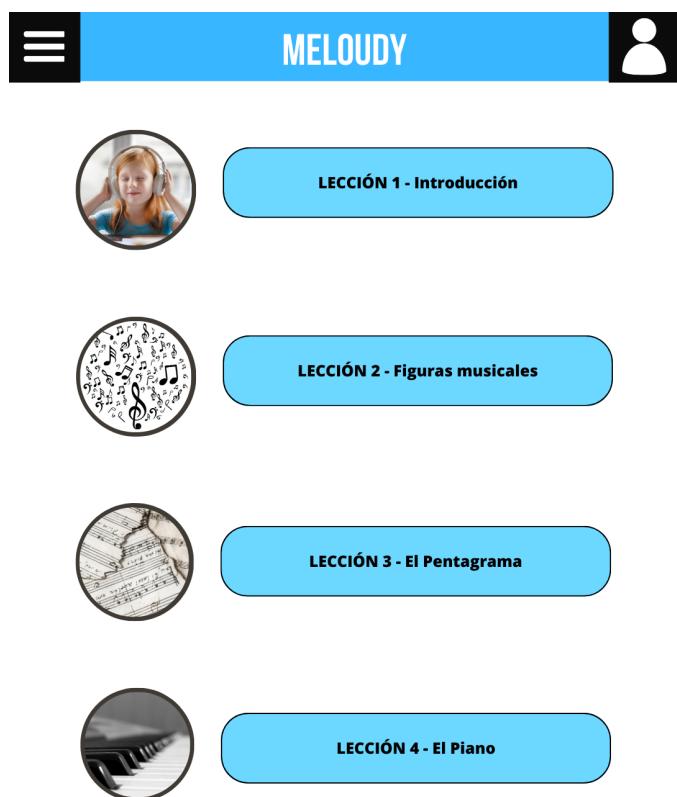


Figura 3.2: Primer boceto de la interfaz de usuario de la página de inicio de la aplicación para los usuarios, donde se muestra la lista de las lecciones

3.3.2. Requisitos funcionales

En esta sección se presentan los requisitos relacionados con las funcionalidades del sistema y su comportamiento, sus servicios y las tareas que este debe realizar.

▪ General

- **RF 1 - Registro de usuarios:** Los usuarios deberán registrarse en el sistema para utilizar la funcionalidad de la aplicación.
- **RF 2 - Identificar usuarios:** Los usuarios deberán identificarse en el sistema con su cuenta para poder utilizar la funcionalidad de la aplicación.

▪ Usuarios

- **RF 3 - Consultar datos de la cuenta:** Los usuarios podrán consultar los datos de su perfil.
- **RF 4 - Modificar datos de la cuenta:** Los usuarios podrán modificar los datos de su perfil (imagen de perfil, nombre, contraseña...).
- **RF 5 - Selección de lección:** Los usuarios podrán seleccionar la lección a estudiar de la lista de lecciones de la pantalla principal.
- **RF 6 - Responder preguntas:** Los usuarios podrán responder las preguntas de cada tipo en la lección que estén.
 - **RF 6.1 - Responder con texto:** Los usuarios podrán responder preguntas que necesiten insertar texto como respuesta.
 - **RF 6.2 - Responder seleccionando varias opciones:** Los usuarios podrán responder preguntas que necesiten seleccionar varias opciones de un conjunto.
 - **RF 6.3 - Responder seleccionando una opción:** Los usuarios podrán responder preguntas que necesiten seleccionar una opción de un conjunto.
 - **RF 6.4 - Responder tocando una nota:** Los usuarios podrán responder preguntas que necesiten tocar una nota de un instrumento y captarla por el micrófono.
- **RF7 - Consultar ranking:** Los usuarios podrán consultar la clasificación global de los usuarios.
- **RF8 - Consultar progreso de perfil:** Los usuarios podrán consultar su progreso y sus logros obtenidos en su perfil.
- **RF9 - Consultar perfiles ajenos:** Los usuarios podrán visualizar el perfil de otros usuarios y ver sus logros.

- **RF22 - Consultar progreso de lección:** Los usuarios podrán consultar el progreso de una lección en concreto.

- **Profesores/Encargados**

- **RF 10 - Obtener lista de lecciones:** Los profesores podrán obtener una lista de todas las lecciones que existen en el sistema.
- **RF 11 - Crear lección:** Los profesores podrán crear nuevas lecciones.
- **RF 12 - Eliminar lección:** Los profesores podrán eliminar lecciones ya existentes.
- **RF 13 - Modificar lecciones:** Los profesores podrán modificar el texto y el contenido multimedia de las distintas lecciones.
- **RF 14 - Crear preguntas:** Los profesores podrán crear nuevas preguntas sobre las distintas lecciones.
- **RF 15 - Obtener lista de preguntas:** Los profesores podrán obtener una lista de todas las preguntas que existen en el sistema.
- **RF 16 - Modificar preguntas:** Los profesores podrán modificar preguntas ya creadas.
- **RF 17 - Eliminar preguntas:** Los profesores podrán eliminar preguntas del sistema.

- **Administradores**

- **RF 18 - Modificar datos de usuarios:** Los administradores podrán modificar los datos de un usuario.
- **RF 19 - Crear usuario:** Los administradores podrán crear usuarios en el sistema.
- **RF 20 - Obtener lista de usuarios:** Los administradores podrán obtener una lista con todos los usuarios registrados en el sistema.
- **RF 21 - Eliminar usuario:** Los administradores podrán eliminar usuarios registrados en el sistema.

3.3.3. Requisitos no funcionales

Estos requisitos se refieren a aquellas condiciones que debe cumplir el sistema en cuanto a rendimiento, accesibilidad, seguridad, robustez...

- **RNF 1 -** La aplicación cifrará las claves en todo momento para garantizar la seguridad de los usuarios.

- **RNF 2** - El número de usuarios simultáneos solo estará limitado por la capacidad del hardware del servidor.
- **RNF 3** - La cantidad máxima de datos a almacenar solo estará limitada por la capacidad del hardware del servidor de la base de datos.
- **RNF 4** - La aplicación será multiplataforma, pudiéndose ejecutar en dispositivos (móvil, ordenador, tablet...) diferentes con distintos sistemas operativos (Windows, Android...)
- **RNF 5** - La aplicación será lo más accesible posible para personas con distintas discapacidades.
- **RNF 6** - La aplicación será tolerante a fallos (será fiable).
- **RNF 7** - Se garantizará el cumplimiento de la Ley de Protección de Datos de Carácter Personal.

Capítulo 4

Planificación

4.1. Introducción

En este capítulo se describirá la planificación para el desarrollo del proyecto, la cual va a servir para poder realizar un control de los avances del producto y asegurar que se cumplan los objetivos marcados en cada sprint/iteración. Para ello, se utilizarán ciertos elementos de la metodología SCRUM, el cual definiremos brevemente a continuación:

SCRUM

SCRUM es un marco de trabajo iterativo e incremental que se utiliza para desarrollar proyectos de software. Este marco de trabajo se basa en la idea de que el desarrollo de software es un proceso complejo que se adapta continuamente a las circunstancias. Por ello, se divide en pequeñas iteraciones que se van realizando de forma incremental y utiliza una serie de elementos para poder controlar el desarrollo del proyecto:

- **Artefactos:** Son documentos que se utilizan para controlar el desarrollo del proyecto.
- **Reuniones:** Se realizan cuatro tipos de reuniones de forma periódica para poder controlar y seguir el progreso del proyecto.
- **Roles:** Representan la responsabilidad de cada persona en el proceso.

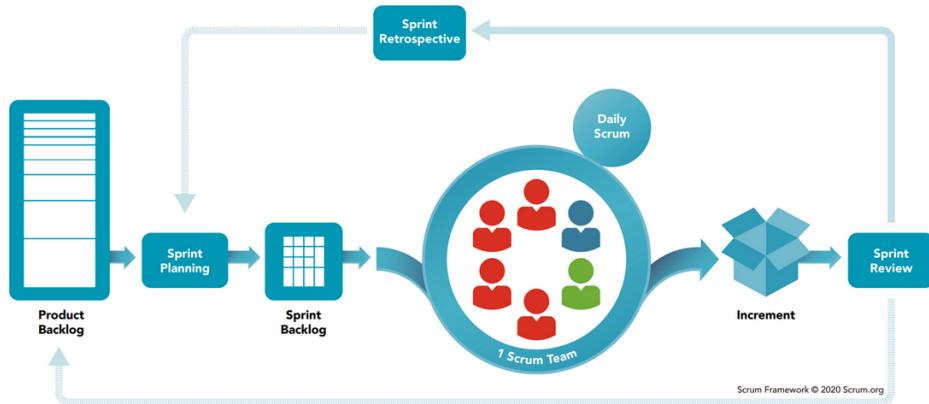


Figura 4.1: Diagrama de la arquitectura del sistema, donde se muestra la comunicación entre el servidor y la aplicación móvil.

Es importante recalcar que no voy a seguir la metodología SCRUM, sino que solo usaré algunos principios y conceptos de metodologías ágiles debido a las condiciones en las que se va a desarrollar el proyecto (no hay un cliente real, no hay un equipo de desarrollo real, etc.). Por tanto, se utilizarán los siguientes aspectos de SCRUM:

- **Sprints:** Se definirán los Sprints (iteraciones) que se realizarán para el desarrollo del proyecto, explicando el objetivo de cada uno de ellos y detallando la duración y las fechas de inicio y de fin.
- **Historias de Usuario:** Se listarán y describirán las historias de usuario que se van a implementar en cada sprint a partir de los requisitos funcionales del proyecto especificados en el capítulo de Especificación de Requisitos.
- **Product Backlog:** Se definirá el product backlog, que es una lista de todas las funcionalidades (Historias de usuario) que se pretenden implementar en el proyecto. Esta lista se ordenará por prioridad de acuerdo a la importancia que tienen para el usuario final.
- **Velocidad:** Se calculará la velocidad de trabajo que se tendrá en el desarrollo del proyecto a partir de la cantidad de horas que se pretenden dedicar al proyecto cada día y la duración de los Sprints.
- **Puntos de historia:** Se asignarán los puntos de historia a cada historia de usuario para poder estimar la cantidad de trabajo que se debería realizar en cada sprint. Los valores de estos puntos de historia se calcularán a partir de la estimación de la complejidad de cada historia de usuario.

- **Reuniones:** Pese a no encajar exactamente en ninguna de las reuniones que hay en SCRUM, se realizarán reuniones cada dos semanas para poder controlar el progreso del proyecto y realizar un seguimiento de los avances. En estas reuniones se tratará:
 - **Qué se ha hecho:** Se comentará el trabajo realizado desde la anterior reunión para comentar las posibles mejoras.
 - **Qué falta por hacer:** Se comparará la planificación realizada para el Sprint con el trabajo realizado, comprobando qué cosas han podido faltar.
 - **Qué se piensa hacer:** Se volverá a planificar si es necesario y se determinará qué trabajo se planea realizar durante el nuevo Sprint.

4.2. Velocidad

A continuación vamos a estimar la velocidad inicial de trabajo que se tendrá en el desarrollo. Esta será una aproximación que nos ayudará a estimar la cantidad de trabajo que se debería realizar en cada iteración. Sin embargo, esta podrá variar y ajustarse a lo largo del proyecto en función de la cantidad de trabajo que se logre completar en cada sprint.

Para calcular la velocidad, vamos a considerar las horas de trabajo que se pretenden dedicar al proyecto cada día y tratar de estimar cuanta cantidad de trabajo se podría realizar.

- Partimos de un “equipo de desarrollo” de 1 miembro.
- Se pretende dedicar 5 horas al día de trabajo aproximadamente.
- Cada Sprint dura 2 semanas (14 días). Si cada día trabajamos 5 horas de forma aproximada, obtenemos un total de 70 horas en cada Sprint.
- En mi entorno, se estima que 1 PH son dos días de trabajo ideal (es decir, 10 horas). Esto significa que cada dos jornadas de trabajo real, se debería completar 1 PH. Sin embargo, en la realidad, esto no siempre es así ya que estamos realizando una aproximación.
- Si multiplicamos un programador por las 70 horas de trabajo y lo dividimos por el número de horas de trabajo por cada punto de historia (10 horas), obtenemos que deberíamos completar **7 PH** por sprint aproximadamente.

Nota: La duración de los Sprints puede variar en función de factores externos al proyecto. Pero se tratará de que duren 2 semanas para seguir un ritmo de trabajo constante.

Nota 2: Puesto que estamos realizando estimaciones y suposiciones para la planificación, esta está sujeta a cambios a lo largo del proyecto.

4.3. Product Backlog

A continuación, se listarán todas las historias de usuario ordenadas por prioridad de acuerdo a la importancia que tienen para el usuario final del producto.

- **HU1** - Como usuario quiero registrarme en la aplicación para poder comenzar a utilizar sus funcionalidades. (Prioridad: Alta — Puntos de Historia: 2)
- **HU2** - Como usuario quiero iniciar sesión en la aplicación para poder acceder a sus funcionalidades. (Prioridad: Alta — Puntos de Historia: 2)
- **HU6** - Como usuario quiero seleccionar una lección para leer el temario. (Prioridad: Alta — Puntos de Historia: 1)
- **HU7** - Como usuario quiero empezar un test de una lección. (Prioridad: Alta — Puntos de Historia: 1)
- **HU26** - Como usuario quiero responder una pregunta de un test del tipo escritura de texto. (Prioridad: Alta — Puntos de Historia: 1)
- **HU31** - Como usuario quiero ver todas las lecciones. (Prioridad: Alta — Puntos de Historia: 0.5)
- **HU8** - Como usuario quiero responder una pregunta de un test del tipo selección múltiple. (Prioridad: Alta — Puntos de Historia: 1)
- **HU9** - Como usuario quiero responder una pregunta de un test del tipo selección única. (Prioridad: Alta — Puntos de Historia: 1)
- **HU10** - Como usuario quiero responder una pregunta de un test del tipo respuesta por micrófono. (Prioridad: Alta — Puntos de Historia: 8)
- **HU29** - Como profesor quiero ver una lista de todas las preguntas. (Prioridad: Alta — Puntos de Historia: 0.5)

- **HU11** - Como usuario quiero ver el resultado de un test. (Prioridad: Media — Puntos de Historia: 2)
- **HU13** - Como usuario quiero ver mi progreso en las lecciones. (Prioridad: Media — Puntos de Historia: 2)
- **HU28** - Como profesor quiero ver una lista de todas las lecciones. (Prioridad: Media — Puntos de Historia: 0.5)
- **HU14** - Como profesor quiero crear una lección. (Prioridad: Media — Puntos de Historia: 1)
- **HU15** - Como profesor quiero modificar el texto de una lección. (Prioridad: Media — Puntos de Historia: 2)
- **HU16** - Como profesor quiero añadir contenido multimedia a una lección. (Prioridad: Media — Puntos de Historia: 2)
- **HU17** - Como profesor quiero eliminar contenido multimedia de una lección. (Prioridad: Media — Puntos de Historia: 1)
- **HU27** - Como profesor quiero añadir preguntas a un test del tipo entrada de texto. (Prioridad: Media — Puntos de Historia: 1)
- **HU18** - Como profesor quiero añadir preguntas a un test del tipo selección múltiple. (Prioridad: Media — Puntos de Historia: 1)
- **HU19** - Como profesor quiero añadir preguntas a un test del tipo selección única. (Prioridad: Media — Puntos de Historia: 1)
- **HU20** - Como profesor quiero añadir preguntas a un test del tipo respuesta por micrófono. (Prioridad: Media — Puntos de Historia: 2)
- **HU21** - Como profesor quiero eliminar preguntas de un test. (Prioridad: Media — Puntos de Historia: 0.5)
- **HU22** - Como profesor quiero modificar preguntas de un test. (Prioridad: Media — Puntos de Historia: 1)
- **HU30** - Como administrador quiero ver una lista de todos los usuarios. (Prioridad: Media — Puntos de Historia: 0.5)
- **HU24** - Como administrador quiero modificar los datos de un usuario. (Prioridad: Media — Puntos de Historia: 2)
- **HU25** - Como administrador quiero eliminar un usuario. (Prioridad: Media — Puntos de Historia: 0.5)
- **HU3** - Como usuario quiero ver los datos personales y los logros de mi perfil. (Prioridad: Media — Puntos de Historia: 1)

- **HU4** - Como usuario quiero editar los datos de mi perfil. (Prioridad: Media — Puntos de Historia: 2)
- **HU23** - Como administrador quiero crear un usuario. (Prioridad: Baja — Puntos de Historia: 1)
- **HU12** - Como usuario quiero ver la lista de mis logros. (Prioridad: Baja — Puntos de Historia: 1)
- **HU5** - Como usuario quiero ver con detalle uno de mis logros. (Prioridad: Baja — Puntos de Historia: 1)

4.4. Sprints

Nuestro proyecto se pretende desarrollar en 14 Sprints. Cada Sprint tendrá una duración de 2 semanas, y se pretenden realizar una media de 7 puntos de historia por Sprint. A continuación se especificará el contenido de cada Sprint con las fechas de inicio y fin de cada uno de ellos.

4.4.1. Sprint #1 - Documentación inicial

24/11/2022 - 08/12/2022

En este Sprint se realizará:

- Definición del proyecto y del alcance.
- Redactar Capítulo 1 - Introducción.
- Redactar Capítulo 2 - Estado del Arte.

4.4.2. Sprint #2 - Documentación de requisitos

08/12/2022 - 12/01/2023

En este Sprint se planea realizar:

- Revisión de la documentación inicial y corrección de errores.
- Redactar Capítulo 3 - Especificación de Requisitos.

4.4.3. Sprint #3 - Documentación de HU

12/01/2023 - 03/02/2023

En este Sprint se realizará:

- Revisión de la documentación de requisitos y corrección de errores.
- Redactar Capítulo 4 - Planificación.
- Redactar Capítulo 5 - Análisis del problema.

4.4.4. Sprint #4 - Diseño y comienzo del desarrollo

03/02/2023 - 17/02/2023

En este Sprint se terminará la documentación y comenzará el desarrollo con:

- Revisión de la planificación y el análisis y corrección de errores.
- Redactar Capítulo 6 - Diseño.
- Preparación del backend para el desarrollo.
 - Creación de la base de datos.
 - Creación de los modelos, rutas y controladores del servidor con NodeJS.
- Preparación del frontend para el desarrollo (creación de la aplicación de Flutter).

4.4.5. Sprint #5 - Registro y login de usuarios

17/02/2023 - 03/03/2023

En este Sprint se comenzará el desarrollo de la aplicación realizando las funcionalidades iniciales del usuario que utilizará la aplicación, tales como el registro, el login, la selección de lección, etc.

- HU1 - Como usuario quiero registrarme en la aplicación.
- HU2 - Como usuario quiero iniciar sesión en la aplicación.
- HU6 - Como usuario quiero seleccionar una lección para leer el temario.

- HU31 - Como usuario quiero ver todas las lecciones.
- HU28 - Como profesor quiero ver una lista de todas las lecciones.

4.4.6. Sprint #6 - Tests y progreso

03/03/2023 - 17/03/2023

Este Sprint tendrá como objetivo el desarrollo de los tests permitiendo al usuario responder a las preguntas y ver el resultado. Además, podrá comprobar su progreso en cada lección y en la pantalla principal que muestra todas las lecciones (desbloqueando las correspondientes en su caso).

- HU7 - Como usuario quiero empezar un test de una lección.
- HU26 - Como usuario quiero responder una pregunta de un test del tipo escritura de texto.
- HU8 - Como usuario quiero responder una pregunta de un test del tipo selección múltiple.
- HU9 - Como usuario quiero responder una pregunta de un test del tipo selección única.
- HU11 - Como usuario quiero ver el resultado de un test.
- HU13 - Como usuario quiero ver mi progreso en las lecciones.

4.4.7. Sprint #7 - Entrada de micrófono y detección de tono

17/03/2023 - 31/03/2023

En este Sprint se realizará la funcionalidad de la entrada de micrófono y la detección de tono. Puesto que la historia de usuario es muy grande y compleja, se reservará un Sprint entero para su desarrollo. Para desarrollar esta funcionalidad, se utilizarán librerías de Flutter que permitan la entrada de audio y trabajar con el mismo.

- HU10 - Como usuario quiero responder una pregunta de un test del tipo respuesta por micrófono.

4.4.8. Sprint #8 - Funcionalidades del profesor con lecciones

31/03/2023 - 14/04/2023

En la octava iteración se comenzarán a realizar las funcionalidades relacionadas con el profesor y la gestión de las lecciones. Además, puede que se necesite terminar la funcionalidad de la entrada de micrófono de la iteración anterior debido a que su estimación supera los puntos de historia de un Sprint.

- HU29 - Como profesor quiero ver una lista de todas las preguntas.
- HU14 - Como profesor quiero crear una lección.
- HU15 - Como profesor quiero modificar el texto de una lección.
- HU16 - Como profesor quiero añadir contenido multimedia a una lección.
- HU17 - Como profesor quiero eliminar contenido multimedia de una lección.

4.4.9. Sprint #9 - Funcionalidades del profesor con tests

14/04/2023 - 28/04/2023

En esta iteración se realizarán las funcionalidades relacionadas con el profesor y la gestión de los tests, tales como la creación, modificación y eliminación de las preguntas de los distintos tipos. Además, se realizará la funcionalidad de ver todos los usuarios registrados en la aplicación.

- HU27 - Como profesor quiero añadir preguntas a un test del tipo escritura de texto.
- HU18 - Como profesor quiero añadir preguntas a un test del tipo selección múltiple.
- HU19 - Como profesor quiero añadir preguntas a un test del tipo selección única.
- HU20 - Como profesor quiero añadir preguntas a un test del tipo respuesta por micrófono.
- HU21 - Como profesor quiero eliminar preguntas de un test.
- HU22 - Como profesor quiero modificar preguntas de un test.
- HU30 - Como administrador quiero ver una lista de todos los usuarios.

4.4.10. Sprint #10 - Funcionalidades del administrador y ranking

28/04/2023 - 12/05/2023

En la decima iteración se pretenden implementar todas las funcionalidades que tienen que ver con la administración de los usuarios, además de dos historias relacionadas con el perfil propio del usuario.

- HU24 - Como administrador quiero modificar los datos de un usuario.
- HU25 - Como administrador quiero eliminar un usuario.
- HU3 - Como usuario quiero ver los datos personales y los logros de mi perfil.
- HU4 - Como usuario quiero editar los datos de mi perfil.
- HU23 - Como administrador quiero crear un usuario.

4.4.11. Sprint #11 - Funcionalidades poco prioritarias y valor añadido

12/05/2023 - 26/05/2023

El objetivo de este Sprint es acabar todas las historias de usuario que quedan del product backlog y cuya prioridad es baja como lo es la interacción entre los distintos usuarios de la aplicación. Además, se añadirá una funcionalidad de valor añadido en función del tiempo disponible y de las funcionalidades que se hayan podido implementar en los Sprints anteriores.

- HU5 - Como usuario quiero ver la lista de mis logros.
- HU12 - Como usuario quiero ver con detalle uno de mis logros.
- Realizar mejoras de código y arreglar errores.
- Mejorar el diseño de la aplicación.

4.4.12. Sprint #12 - Mejoras de código

26/05/2023 - 09/06/2023

En este Sprint se pretende realizar mejoras de código en cuanto a estilo, formato y calidad del mismo. Además, se pretende dejar terminado la redacción del séptimo capítulo del documento.

- Revisar Capítulo 7 - Implementación
- Realizar mejoras de código y arreglar errores.

4.4.13. Sprint #13 - Pruebas

09/06/2023 - 23/06/2023

Este Sprint se dedicará a realizar todas las pruebas necesarias para comprobar que la aplicación funciona correctamente y que no hay errores en el código, tanto en el frontend como en el backend. También se redactará el octavo capítulo del documento con las pruebas realizadas.

- Realizar pruebas de integración.
- Realizar pruebas de unidad.
- Redactar Capítulo 8 - Pruebas.

4.4.14. Sprint #14 - Conclusión y finalización del proyecto

23/06/2023 - 15/07/2023

El último Sprint se dedicará por completo a la redacción del último capítulo del documento, en el que se recogerán las conclusiones del proyecto y se realizará una revisión de todos los capítulos para corregir errores y mejorar la redacción. Se preparará también la presentación del proyecto para la defensa ante el tribunal.

- Redactar Capítulo 9 - Conclusiones.
- Revisión de todo el documento y corrección de errores.
- Preparación de la presentación del proyecto.

4.5. Diagrama de Gantt

Por último, se muestran los diagramas de Gantt de la planificación del proyecto.

El primero es más general y representa la división del desarrollo del producto en Sprints. En él se puede ver la duración de cada Sprint y la proporción de estos en el tiempo total del proyecto.

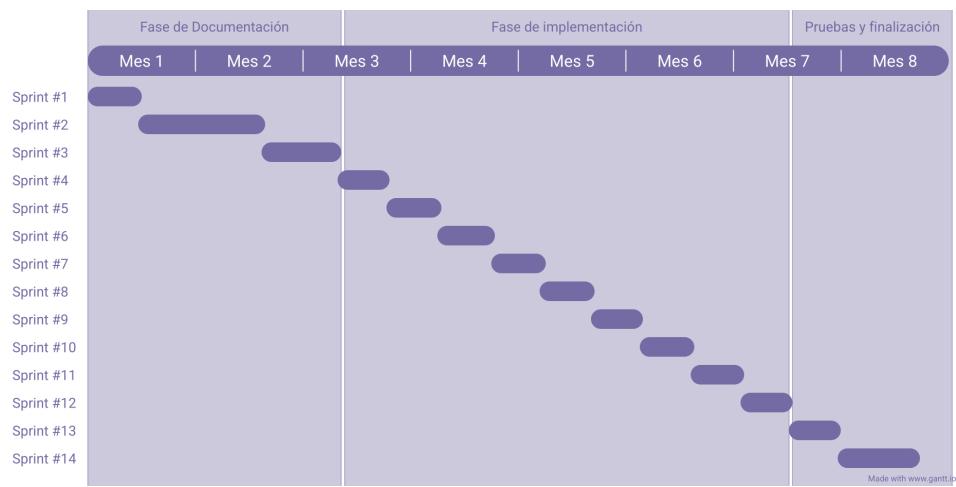


Figura 4.2: Diagrama de Gantt de la planificación del proyecto, divivido por Sprints.

Como se puede ver, el proyecto se ha dividido en 14 Sprints, cada uno de ellos con una duración de dos semanas aproximadamente. El desarrollo de cada Sprint se realizará normalmente de forma secuencial, es decir, no se empezará a desarrollar el siguiente Sprint hasta que no se haya terminado el anterior. Además, es importante recalcar que los Sprints del proyecto se dividen en tres fases:

- **Fase de Documentación:** en esta fase se redactará la documentación necesaria para el desarrollo del proyecto, como puede ser la especificación de requisitos, el estado del arte, la planificación, el diseño... A pesar de que esta sea la fase donde se redactará la mayor parte de la documentación, también se realizará esta en las fases posteriores.
- **Implementación:** en esta fase se desarrollará el producto, es decir, se implementarán las funcionalidades del proyecto.
- **Pruebas y finalización:** en esta fase se realizarán las pruebas necesarias para comprobar que el producto funciona correctamente y que no hay errores en el código. Además, se finalizará la documentación del proyecto y se preparará la presentación del mismo.

Por otro lado, a continuación se encuentra un diagrama de Gantt más detallado, en el que se puede ver la división de cada Sprint en historias de usuario o tareas y la duración de cada una de ellas. El diagrama se ha dividido en cuatro partes para facilitar su visualización y su lectura, debido a la gran cantidad de información que contiene.

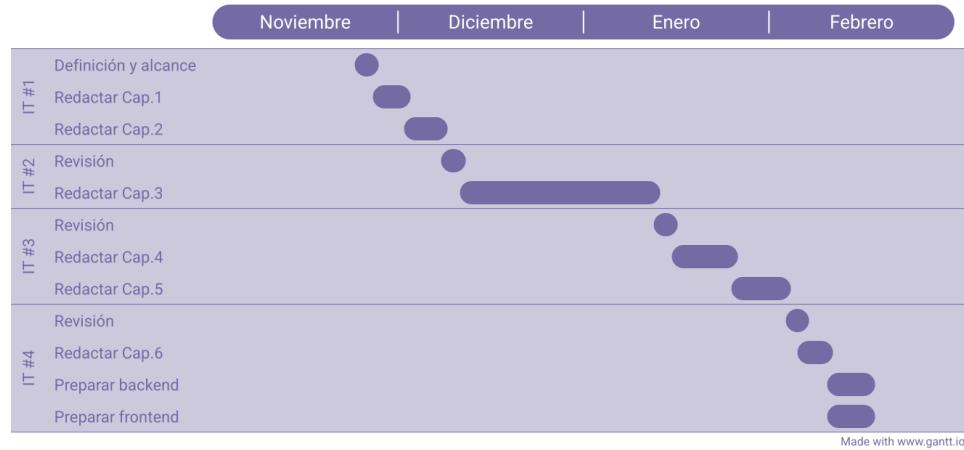


Figura 4.3: Diagrama de Gantt de los cuatro primeros Sprints.

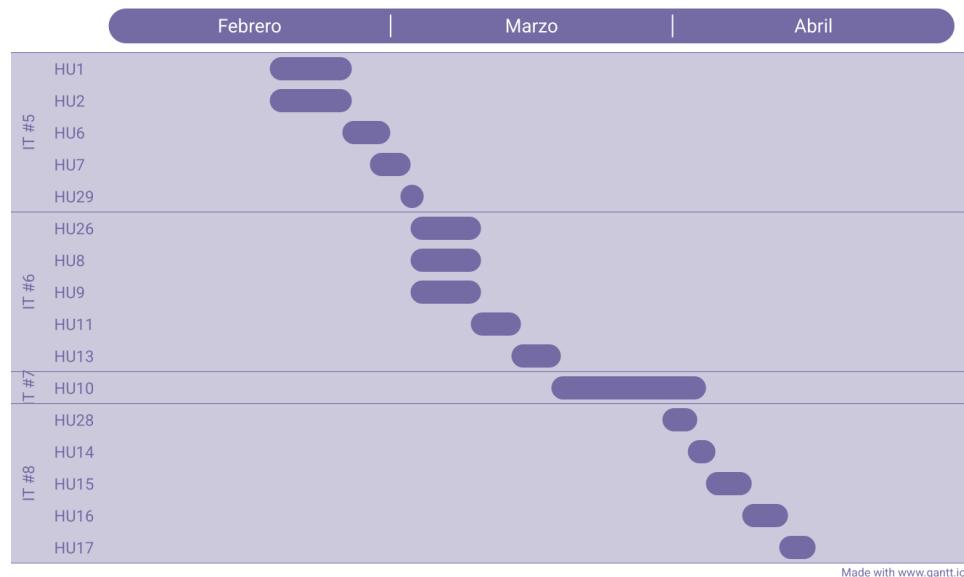


Figura 4.4: Diagrama de Gantt de los Sprints 5, 6, 7 y 8.

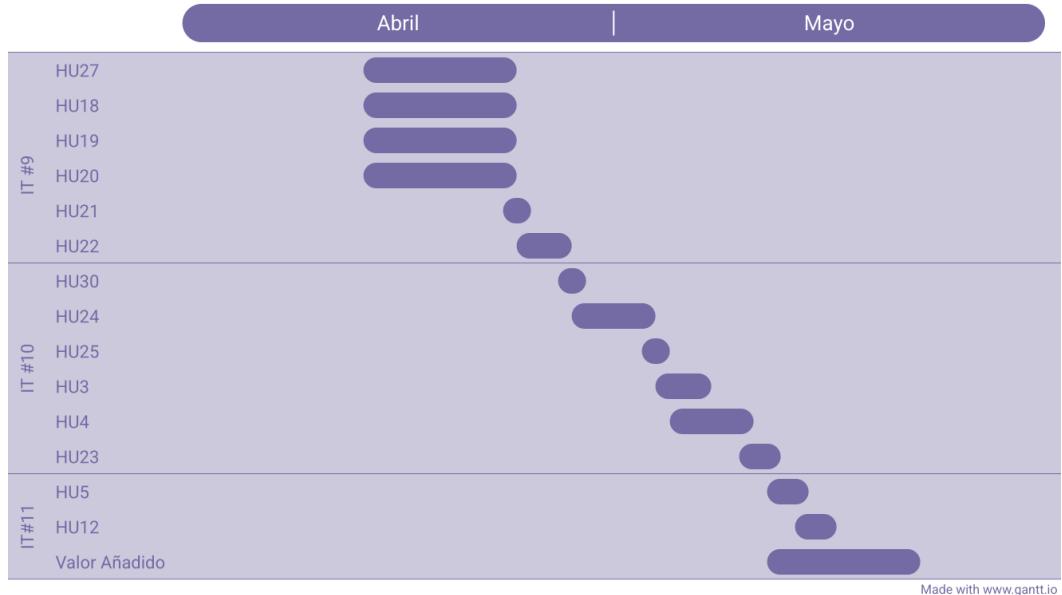


Figura 4.5: Diagrama de Gantt de los Sprints 9, 10 y 11.

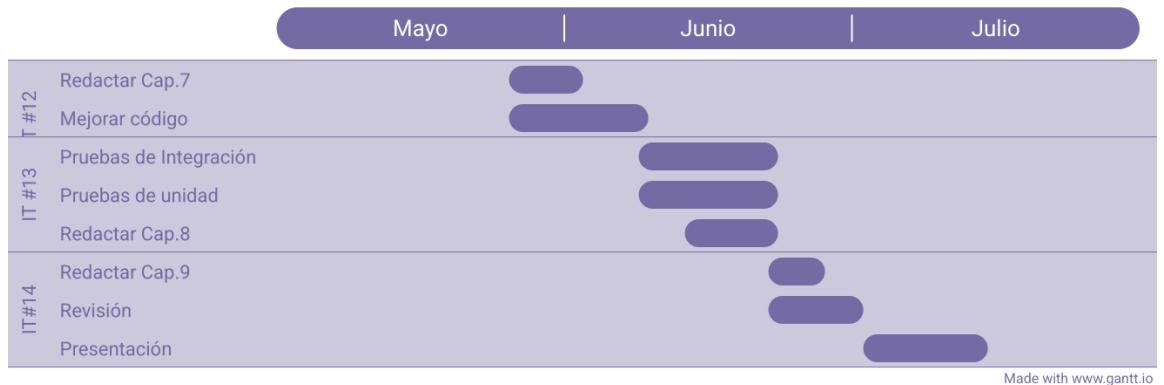


Figura 4.6: Diagrama de Gantt de los Sprints 12, 13 y 14.

Capítulo 5

Análisis del problema

5.1. Introducción

En este capítulo se va a describir de forma más detallada el problema que se va a resolver con la aplicación a desarrollar. De esta forma, vamos a especificar qué es lo que tiene que hacer el sistema para su futuro desarrollo. El objetivo del análisis es definir, estructurar y describir los requisitos o historias de usuario para conseguir una comprensión más precisa y detallada del problema a resolver.

5.2. Historias de Usuario

A continuación, se procederá a describir las historias de usuario que se han definido para el proyecto.

Funcionalidad del usuario

ID	HU1	Nombre	Como usuario quiero registrarme en la aplicación para poder comenzar a utilizar sus funcionalidades.
PH	2	Descripción	El usuario podrá registrarse en el sistema llenando un formulario con campos como el nombre de usuario, el correo electrónico, la contraseña...
Prioridad	Alta	Dependencias	— Requisitos Funcionales RF 1
Pruebas de aceptación		1. Todos los campos llenados por el usuario serán almacenados en la base de datos. 2. La contraseña se almacenará cifrada para no comprometer la seguridad del usuario. 3. Si el usuario introduce valores erróneos, se informará del campo en el que está el error. 4. Si el usuario no introduce datos en los campos obligatorios, se informará de los campos que faltan por llenar.	

ID	HU2	Nombre	Como usuario quiero iniciar sesión en la aplicación para poder acceder a sus funcionalidades.
PH	2	Descripcion	El usuario podrá iniciar sesión en el sistema rellenando un formulario con campos como el nombre de usuario o correo electrónico y la contraseña...
Prioridad	Alta	Dependencias	— Requisitos Funcionales RF 2
Pruebas de aceptacion			1. Se compararán los campos llenados por el usuario con los almacenados para comprobar la identidad del usuario. 2. La contraseña procesará de forma cifrada para no comprometer la seguridad del usuario. 3. Si el usuario introduce datos incorrectos, se informará del error al usuario.

ID	HU3	Nombre	Como usuario quiero ver los datos personales y los logros de mi perfil.
PH	1	Descripcion	El usuario podrá visualizar todos los datos de su propio perfil, así como los logros obtenidos en su aprendizaje.
Prioridad	Media	Dependencias	— Requisitos Funcionales RF 3
Pruebas de aceptacion			1. Los datos mostrados al usuario en su perfil se corresponderán con la información almacenada en la base de datos de dicho usuario.

ID	HU4	Nombre	Como usuario quiero editar los datos de mi perfil.
PH	2	Descripcion	El usuario podrá modificar los datos de su perfil como su nombre de usuario, correo electrónico, contraseña, etc...
Prioridad	Media	Dependencias	HU 3 Requisitos Funcionales RF 4
Pruebas de aceptacion			1. Los datos introducidos por el usuario en los campos del formulario se actualizarán en la base de datos 2. En caso de modificar la contraseña, esta se almacenará cifrada para no comprometer la seguridad del usuario. 3. Si el usuario introduce valores erróneos, se informará del campo en el que está el error.

ID	HU5	Nombre	Como usuario quiero ver el perfil de otros usuarios.
PH	1	Descripcion	El usuario podrá visitar los perfiles de otros usuarios registrados y ver sus logros
Prioridad	Baja	Dependencias	HU 3 Requisitos Funcionales RF 9
Pruebas de aceptacion			1. Los datos mostrados al usuario en su perfil se corresponderán con la información almacenada en la base de datos de dicho usuario. 2. Los datos privados del usuario no se mostrarán, solo los que no sean personales. 3. El usuario que no haya iniciado sesión no podrá ver ningún perfil.

ID	HU6	Nombre	Como usuario quiero seleccionar una lección para leer el temario.
PH	1	Descripcion	El usuario podrá hacer click en la lección que desea de la lista ubicada en la pantalla principal.
Prioridad	Alta	Dependencias	— Requisitos Funcionales RF 5
Pruebas de aceptacion			1. El contenido (textos, imágenes, videos...) almacenado en la base de datos de la lección seleccionada aparecerá en pantalla. 2. No se podrá elegir una lección que esté bloqueada.

ID	HU7	Nombre	Como usuario quiero empezar un test de una lección.
PH	1	Descripcion	El usuario pulsará el botón de empezar test ubicado dentro de la lección para poder responder a las preguntas.
Prioridad	Alta	Dependencias	— Requisitos Funcionales RF 6
Pruebas de aceptacion			1. Las preguntas almacenadas en la base de datos de dicha lección aparecerán en orden.

ID	HU8	Nombre	Como usuario quiero responder una pregunta de un test del tipo selección múltiple.
PH	1	Descripcion	El usuario podrá responder a una pregunta seleccionando más de una opción de la lista de posibles respuestas.
Prioridad	Alta	Dependencias	HU7 Requisitos Funcionales RF 6.2
Pruebas de aceptacion			1. La respuesta del usuario quedará registrada en el sistema para un futuro procesamiento. 2. El usuario deberá seleccionar una respuesta como mínimo.

ID	HU9	Nombre	Como usuario quiero responder una pregunta de un test del tipo selección única.
PH	1	Descripcion	El usuario podrá responder a una pregunta seleccionando una sola opción de la lista de posibles respuestas.
Prioridad	Alta	Dependencias	HU7 Requisitos Funcionales RF 6.3
Pruebas de aceptacion			1. La respuesta del usuario quedará registrada en el sistema para un futuro procesamiento. 2. El usuario deberá seleccionar una opción obligatoriamente.

ID	HU10	Nombre	Como usuario quiero responder una pregunta de un test del tipo respuesta por micrófono.
PH	8	Descripcion	El usuario podrá responder a una pregunta mediante la entrada de un sonido de un instrumento externo a través del micrófono.
Prioridad	Alta	Dependencias	HU7 Requisitos Funcionales RF 6.4
Pruebas de aceptacion			1. La respuesta del usuario quedará registrada en el sistema para un futuro procesamiento.

ID	HU26	Nombre	Como usuario quiero responder una pregunta de un test del tipo escritura de texto.
PH	1	Descripcion	El usuario podrá responder a una pregunta escribiendo la respuesta en un campo de texto.
Prioridad	Media	Dependencias	HU7 Requisitos Funcionales RF 6.1
Pruebas de aceptacion			1. La respuesta del usuario quedará registrada en el sistema para un futuro procesamiento.

ID	HU11	Nombre	Como usuario quiero ver el resultado de un test.
PH	2	Descripcion	El usuario podrá visualizar el resultado del test una vez finalizado.
Prioridad	Media	Dependencias	HU7, HU8, HU9, HU10, HU26 Requisitos Funcionales RF 6
Pruebas de aceptacion			1. El resultado del test será almacenado en la base de datos. 2. La puntuación mostrada en pantalla corresponderá a las preguntas acertadas en dicho test.

ID	HU12	Nombre	Como usuario quiero ver el ranking de usuarios.
PH	1	Descripcion	El usuario podrá visualizar la lista ordenada de usuarios con mayor progreso.
Prioridad	Baja	Dependencias	— Requisitos Funcionales RF 22
Pruebas de aceptacion			1. La lista mostrará a los usuarios con más puntuación en el sistema.

ID	HU13	Nombre	Como usuario quiero ver mi progreso en las lecciones.
PH	2	Descripcion	El usuario podrá ver su progreso en cada lección, los resultados de cada test y los puntos que le faltan para completar la lección.
Prioridad	Media	Dependencias	HU6 Requisitos Funcionales RF 1
Pruebas de aceptacion			1. Se mostrarán los datos relacionados con el progreso del usuario de la lección correspondiente.

Gestión de lecciones y tests

ID	HU14	Nombre	Como profesor quiero crear una lección.
PH	1	Descripcion	El profesor podrá crear una lección, añadiendo el texto y el contenido multimedia que desee.
Prioridad	Media	Dependencias	HU28 Requisitos Funcionales RF 11
Pruebas de aceptacion			1. Todos los campos rellenados por el profesor serán almacenados en la base de datos. 2. Si el usuario introduce valores erróneos, se informará del campo en el que está el error. 3. Si el usuario no introduce datos en los campos obligatorios, se informará de los campos que faltan por llenar.

ID	HU15	Nombre	Como profesor quiero modificar el texto de una lección.
PH	2	Descripcion	El profesor podrá modificar el texto de una lección en particular.
Prioridad	Media	Dependencias	HU14, HU28 Requisitos Funcionales RF 13
Pruebas de aceptacion			1. Todos los campos llenados por el profesor serán actualizados en la base de datos. 2. Si el usuario introduce valores erróneos, se informará del campo en el que está el error. 3. Si el usuario no introduce datos en los campos obligatorios, se informará de los campos que faltan por llenar.

ID	HU16	Nombre	Como profesor quiero añadir contenido multimedia a una lección.
PH	2	Descripcion	El usuario podrá añadir contenido multimedia al cuerpo de una lección.
Prioridad	Media	Dependencias	HU28, HU14 Requisitos Funcionales RF 13
Pruebas de aceptacion			1. Se agregará el contenido multimedia a la tabla de la lección en la base de datos. 2. Si el usuario introduce valores erróneos, se informará del campo en el que está el error. 3. Si el usuario no introduce datos en los campos obligatorios, se informará de los campos que faltan por llenar.

ID	HU17	Nombre	Como profesor quiero eliminar contenido multimedia de una lección.
PH	1	Descripcion	El usuario podrá eliminar contenido multimedia del cuerpo de una lección.
Prioridad	Media	Dependencias	HU28, HU14 Requisitos Funcionales RF 13
Pruebas de aceptacion			1. Se eliminará el contenido multimedia de la tabla de la lección en la base de datos.

ID	HU18	Nombre	Como profesor quiero añadir preguntas a un test del tipo selección múltiple.
PH	1	Descripcion	El profesor creará preguntas de tipo selección multiple y las añadirá a un test.
Prioridad	Media	Dependencias	— Requisitos Funcionales RF 14
Pruebas de aceptacion			1. Todos los campos llenados por el profesor serán almacenados en la base de datos. 2. La pregunta se mostrará en su test correspondiente.

ID	HU19	Nombre	Como profesor quiero añadir preguntas a un test del tipo selección única.
PH	1	Descripcion	El profesor creará preguntas de tipo selección única y las añadirá a un test.
Prioridad	Media	Dependencias	— Requisitos Funcionales RF 14
Pruebas de aceptacion			1. Todos los campos llenados por el profesor serán almacenados en la base de datos. 2. La pregunta se mostrará en su test correspondiente.

ID	HU20	Nombre	Como profesor quiero añadir preguntas a un test del tipo respuesta por micrófono.
PH	2	Descripcion	El profesor creará preguntas de tipo respuesta por micrófono y las añadirá a un test.
Prioridad	Media	Dependencias	— Requisitos Funcionales RF 14
Pruebas de aceptacion			1. Todos los campos llenados por el profesor serán almacenados en la base de datos. 2. La pregunta se mostrará en su test correspondiente.

ID	HU21	Nombre	Como profesor quiero eliminar preguntas de un test.
PH	0.5	Descripcion	El usuario podrá eliminar preguntas existentes de un test.
Prioridad	Media	Dependencias	HU29 Requisitos Funcionales RF 17
Pruebas de aceptacion			1. Se eliminará la información de dicha pregunta de la base de datos

ID	HU22	Nombre	Como profesor quiero modificar preguntas de un test.
PH	1	Descripcion	El profesor podrá modificar tanto el enunciado (la cuestión) como las posibles respuestas de cada pregunta de un test a través de un formulario.
Prioridad	Media	Dependencias	HU29 Requisitos Funcionales RF 16
Pruebas de aceptacion			1. Todos los campos llenados por el profesor serán actualizados en la base de datos 2. La contraseña se almacenará cifrada para no comprometer la seguridad del usuario. 3. Si el usuario introduce valores erróneos, se informará del campo en el que está el error. 4. Si el usuario no introduce datos en los campos obligatorios, se informará de los campos que faltan por llenar.

ID	HU27	Nombre	Como profesor quiero añadir preguntas a un test del tipo escritura de texto.
PH	1	Descripcion	El profesor creará preguntas de tipo escritura de texto y las añadirá a un test.
Prioridad	Media	Dependencias	— Requisitos Funcionales RF 14
Pruebas de aceptacion		1. Todos los campos rellenados por el profesor serán almacenados en la base de datos. 2. La pregunta se mostrará en su test correspondiente.	

ID	HU28	Nombre	Como profesor quiero ver una lista de todas las lecciones.
PH	0.5	Descripcion	El profesor podrá ver una lista de todas las lecciones almacenadas en el sistema.
Prioridad	Media	Dependencias	— Requisitos Funcionales RF 10
Pruebas de aceptacion		1. Todas las lecciones almacenados en la base de datos del sistema se mostrarán en pantalla	

ID	HU29	Nombre	Como profesor quiero ver una lista de todas las preguntas.
PH	0.5	Descripcion	El profesor podrá ver una lista de todas las preguntas almacenadas en el sistema.
Prioridad	Media	Dependencias	— Requisitos Funcionales RF 15
Pruebas de aceptacion		1. Todas las preguntas almacenadas en la base de datos del sistema se mostrarán en pantalla	

Gestión de usuarios

ID	HU23	Nombre	Como administrador quiero crear un usuario.
PH	1	Descripcion	El administrador podrá crear usuarios llenando un formulario con campos como el nombre de usuario, el correo electrónico, la contraseña...
Prioridad	Baja	Dependencias	— Requisitos Funcionales RF 19
Pruebas de aceptacion		1. Todos los campos rellenados por el administrador serán almacenados en la base de datos 2. La contraseña se almacenará cifrada para no comprometer la seguridad del usuario. 3. Si el usuario introduce valores erróneos, se informará del campo en el que está el error. 4. Si el usuario no introduce datos en los campos obligatorios, se informará de los campos que faltan por llenar.	

ID	HU24	Nombre	Como administrador quiero modificar los datos de un usuario.
PH	2	Descripcion	El administrador podrá modificar los datos de un usuario registrado en el sistema rellenando un formulario con campos como el nombre de usuario, el correo electrónico, la contraseña...
Prioridad	Media	Dependencias	HU30 Requisitos Funcionales RF 18
Pruebas de aceptacion		1. Todos los campos rellenados por el usuario serán almacenados en la base de datos 2. La contraseña se almacenará cifrada para no comprometer la seguridad del usuario. 3. Si el usuario introduce valores erróneos, se informará del campo en el que está el error. 4. Si el usuario no introduce datos en los campos obligatorios, se informará de los campos que faltan por llenar.	

ID	HU25	Nombre	Como administrador quiero eliminar un usuario.
PH	0.5	Descripcion	El administrador podrá eliminar un usuario registrado en el sistema.
Prioridad	Media	Dependencias	HU30 Requisitos Funcionales RF 21
Pruebas de aceptacion		1. Los datos del usuario serán eliminados de la base de datos.	

ID	HU30	Nombre	Como administrador quiero ver una lista de todos los usuarios.
PH	0.5	Descripción	El profesor podrá ver una lista de todos los alumnos registrados en el sistema.
Prioridad	Media	Dependencias	— Requisitos Funcionales RF 20
Pruebas de aceptación			1. Todos los usuarios almacenados en la base de datos del sistema se mostrarán en pantalla

5.3. Diagrama de Clases

Para finalizar esta sección, se muestra un primer diagrama de clases de nuestro proyecto, que nos permite ver las relaciones entre las diferentes clases que compondrán el sistema y que nos ayudará a entender mejor el funcionamiento del mismo. Para ello, primero se han identificado las entidades en nuestro sistema a partir de las historias de usuario y de los requisitos funcionales. Tras esto, se han hallado las relaciones entre las diferentes entidades. Por último, se han definido los atributos (las propiedades) de dichas entidades.

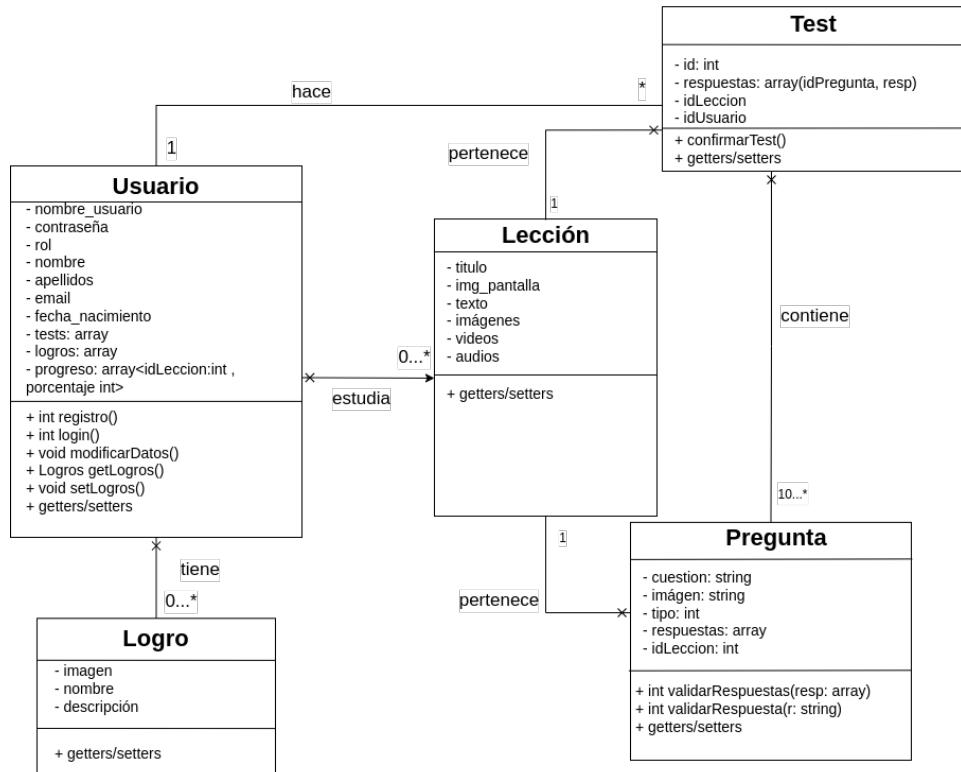


Figura 5.1: Diagrama de clases de nuestro proyecto donde se muestran las relaciones entre las diferentes clases que componen el sistema.

Como se puede observar, disponemos de cinco clases principales: Usuario, Lección, Test, Pregunta y Logro. Estas entidades se han relacionado

siguiendo la descripción del sistema que se ha realizado hasta ahora.

Capítulo 6

Diseño

6.1. Introducción

En este capítulo se realizará el diseño del software a desarrollar. Para ello, se detallará cómo se va a desarrollar la aplicación, qué arquitectura y tecnologías se van a utilizar y cómo se va a realizar el diseño de la interfaz de usuario. Esta fase tiene como objetivo definir la estructura del sistema y la función de cada una de sus partes, lo cual permitirá que el desarrollo sea más eficiente y que el resultado final tenga mayor calidad.

6.2. Arquitectura del sistema

Para el desarrollo de la aplicación se ha decidido utilizar una arquitectura **cliente-servidor**, donde el servidor será responsable de las operaciones relacionadas con la gestión (añadir, extraer, eliminar y/o modificar) de la información almacenada en la base de datos, mientras que el cliente será la aplicación móvil que se encargará de mostrar la información al usuario y de realizar las operaciones que el usuario solicite. Además, también será una arquitectura basada en el patrón de diseño **Modelo / Vista / Controlador (MVC)**, donde el modelo gestionará y mantendrá la estructura de la información de la base de datos, la vista se hará cargo de mostrar la información al usuario y facilitar la interacción de este con el sistema y el controlador se encargará de gestionar las operaciones que el usuario solicite. Todo esto mediante las tecnologías **Node.js** para el controlador, **Mongoose** para el modelo, **MongoDB** para la base de datos y **Flutter** para las distintas vistas que los usuarios tendrán.

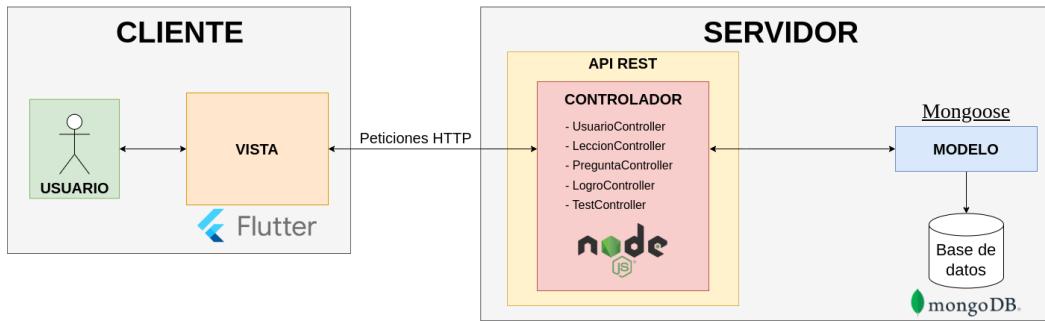


Figura 6.1: Diagrama de la arquitectura del sistema, donde se muestra la comunicación entre el servidor y la aplicación móvil.

Como podemos ver en el diagrama, el controlador tendrá una API REST que será la encargada de gestionar las peticiones que se realicen desde la aplicación móvil. La API REST es una interfaz que permite la comunicación entre dos sistemas de computación (en nuestro caso, entre el cliente y el servidor) para el intercambio de información de manera segura. REST definirá las restricciones a cumplir dentro de nuestra arquitectura, como por ejemplo la necesidad de utilizar los principales verbos HTTP (GET, POST, PUT y DELETE) para realizar las operaciones de lectura, creación, actualización y eliminación de la información.

6.3. Diagrama de base de datos

En cuanto al diagrama de base de datos, se ha realizado un diagrama de base de datos no relacional, que representa los distintos documentos que tendrá nuestra base de datos de MongoDB. En este caso, se han definido 5 documentos, uno para los usuarios, otro para las lecciones, otro para los tests, otro para las preguntas de los tests y otro para los logros. También se han definido las referencias que habrá entre los documentos, como por ejemplo que un usuario tiene varios logros.

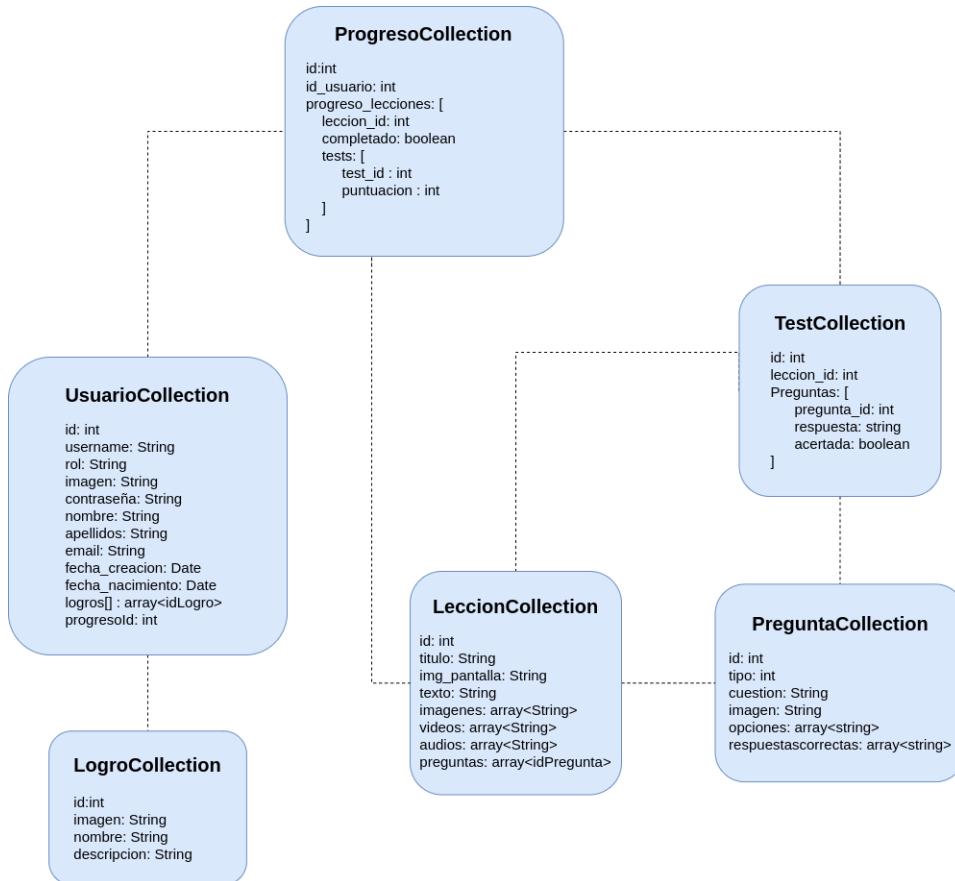


Figura 6.2: Diagrama de base de datos del sistema, donde se muestran los documentos que se van a almacenar en nuestra base de datos de MongoDB.

En el diagrama se pueden ver varias referencias entre las colecciones de datos que se explicarán a continuación:

- Un usuario tendrá los logros que consiga al progresar en la aplicación.
- Un usuario tiene un progreso, que contendrá las lecciones que realice junto con los tests que haya contestado.
- Un test tendrá varias preguntas (alrededor de 10) y estas se generarán de forma aleatoria cuando se cree el test.
- Una lección tendrá varias preguntas y de dicho conjunto de preguntas se seleccionarán algunas al azar para formar el test.
- Un test pertenecerá a una lección: un test solo podrá pertenecer a una lección (aquella donde se generó dicho test).

6.4. Diagrama de clases

A continuación se muestra el diagrama de clases de diseño, el cual se ha realizado a partir del diagrama de clases de análisis del capítulo anterior. En esta ocasión se han detallado más los atributos y los métodos de cada clase, facilitando así la comprensión de las relaciones entre las clases y de la estructura del sistema.

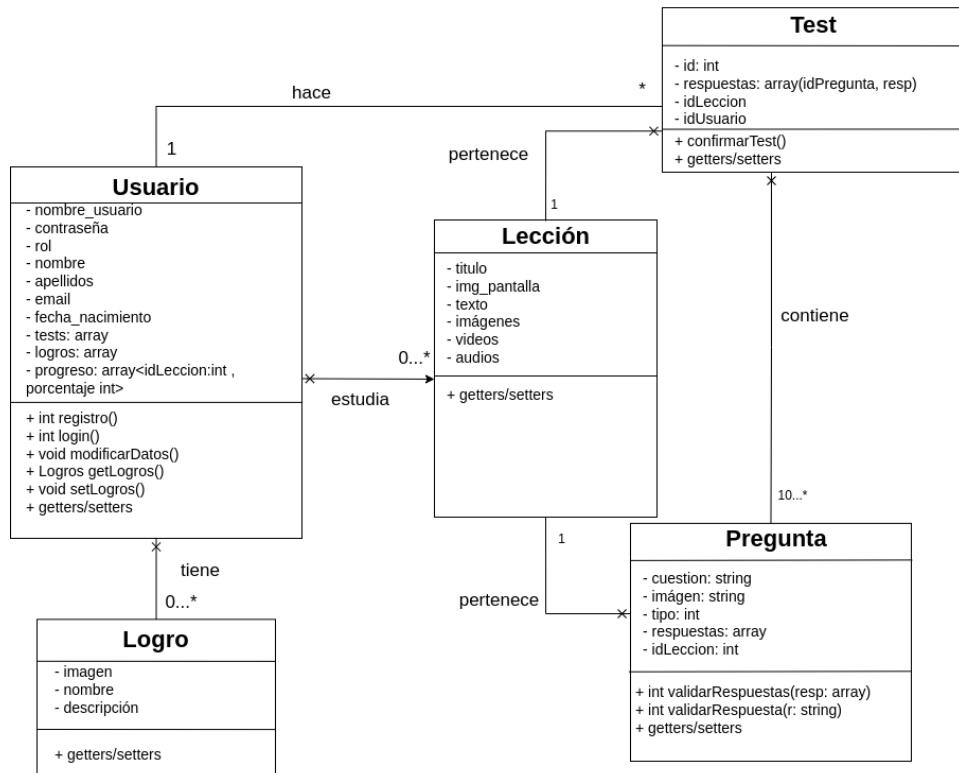


Figura 6.3: Diagrama de clases del sistema donde se detallan las propiedades y las relaciones de las distintas clases o entidades que tendrá el software.

6.5. Diagramas de secuencia

En lo que respecta a los diagramas de secuencia, se han realizado algunos ejemplos de cómo sería el flujo de una operación en el sistema. Como hemos visto anteriormente, se seguirá una arquitectura basada en el patrón de diseño MVC, por lo que las peticiones pasarán por la vista, luego por el controlador y finalmente por el modelo. En este caso, se ha realizado un diagrama de secuencia de cómo sería el flujo de una operación en la que un usuario se registra en la aplicación, otro para un usuario que contesta un test y otro para un profesor que modifica una lección.

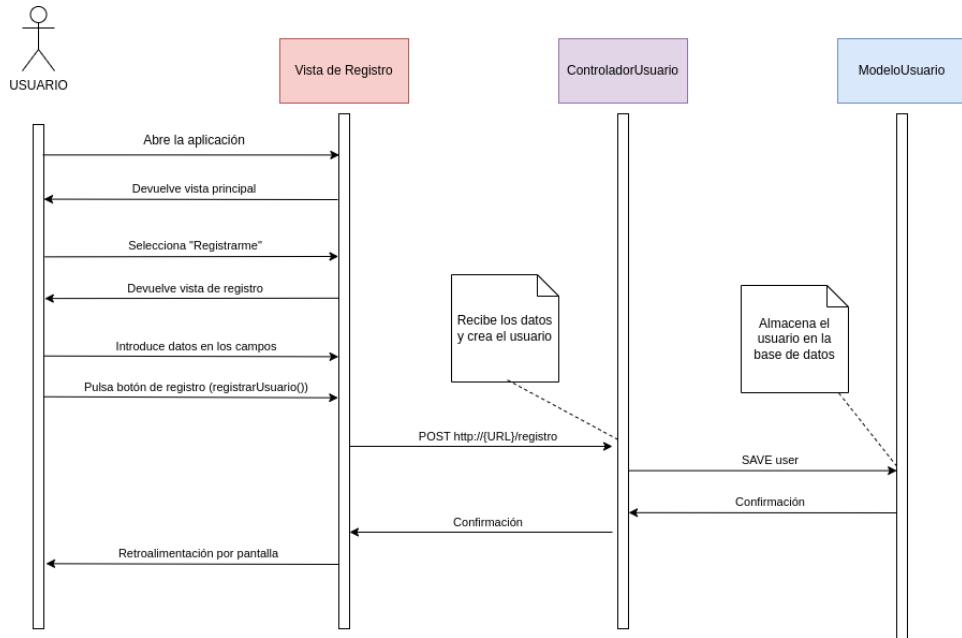


Figura 6.4: Diagrama de secuencia de un usuario registrándose, el cual interactuará con la vista para poder hacer la petición al controlador y guardar su usuario en el modelo.

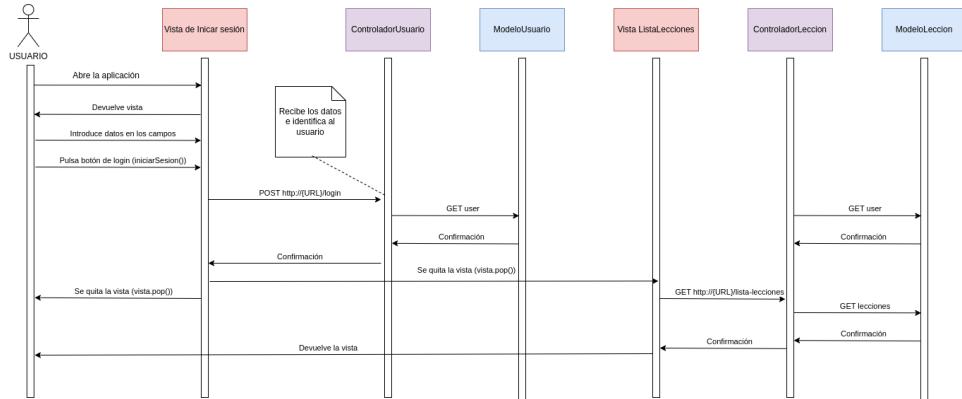


Figura 6.5: Diagrama de secuencia de un usuario iniciando sesión, el cual interactuará con la vista para poder hacer la petición al controlador y buscar su usuario en el modelo para la validación. Tras esto, se cambiará la vista a la pantalla principal que lista las lecciones, por lo que la vista deberá hacer otra petición al controlador de lecciones para obtener las lecciones del modelo.

6.6. Diseño de interfaces de usuario

Por último, en este apartado se presentan los bocetos de las interfaces de usuario que se han diseñado para la aplicación. Estos bocetos se han realizado con la herramienta Canva y serán de ayuda para la realización del diseño final de las interfaces de usuario, que, pese a que seguirán una estructura similar a la de los bocetos, podrán variar en algunos detalles y aspectos de diseño.

Inicio de sesión

Esta vista se mostrará al usuario cuando inicie la aplicación y no tenga la sesión iniciada. En ella se pedirá al usuario que introduzca su correo electrónico y su contraseña para poder iniciar sesión en la aplicación. Si el usuario no tiene una cuenta, podrá pulsar en el botón de registro para poder crear una nueva cuenta.



Figura 6.6: Boceto de la pantalla de inicio de sesión, donde se pedirá al usuario el correo electrónico y la contraseña.

Registro

Esta vista se mostrará al usuario cuando pulse en el botón de registro de la pantalla de inicio de sesión. En ella se pedirá al usuario que introduzca sus datos personales para poder crear una nueva cuenta en la aplicación. Si el usuario ya tiene una cuenta, podrá pulsar en el botón de inicio de sesión para poder iniciar sesión en la aplicación.



Figura 6.7: Boceto de la pantalla de registro, donde se pedirá al usuario sus datos personales necesarios para crear la cuenta como el nombre, los apellidos, el correo y la contraseña.

Lista de lecciones

Esta vista se mostrará al usuario cuando inicie la aplicación y tenga la sesión iniciada. En ella se mostrarán las lecciones disponibles para el usuario y podrá seleccionar una de ellas para poder acceder a su contenido.



Figura 6.8: Boceto de la pantalla principal de la aplicación donde se muestran las lecciones disponibles para el usuario.

Perfil de usuario

Esta vista se mostrará al usuario cuando pulse en el botón de perfil de la pantalla principal de la aplicación. En ella se mostrarán los datos del usuario y podrá acceder a sus logros y a la modificación de sus datos.



Figura 6.9: Boceto de la pantalla del perfil de usuario con sus datos.

Lista de logros

Esta vista se mostrará al usuario cuando pulse en el botón de logros de la pantalla del perfil de usuario. En ella se mostrarán los logros conseguidos por el usuario. Pulsando en cada logro se mostrará el nombre y la descripción de este.



Figura 6.10: Boceto de la pantalla de logros conseguidos por el usuario.

Lección

Esta vista se mostrará al usuario cuando pulse en una lección de la pantalla principal de la aplicación. En ella se mostrará el texto de la lección, el contenido multimedia y el botón para comenzar el test.

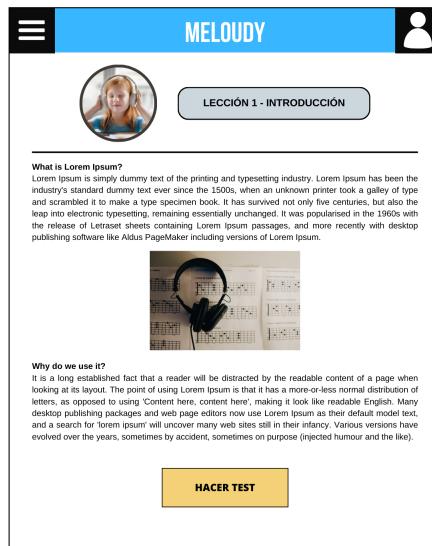


Figura 6.11: Boceto de la pantalla de una lección, con el texto, el contenido multimedia y el botón para comenzar el test.

Pregunta de selección única

Con esta vista se mostrará al usuario una pregunta de test de tipo selección única. En ella se mostrará la pregunta y las opciones de respuesta. El usuario podrá seleccionar solamente una de ellas y pulsar en el botón de comprobar para comprobar si la respuesta es correcta o no. Si la respuesta es correcta, se mostrará un mensaje de felicitación y se mostrará el botón de siguiente para pasar a la siguiente pregunta. Si la respuesta es incorrecta, se mostrará un mensaje de error y se mostrará el botón de siguiente para pasar a la siguiente pregunta. Si el usuario no ha seleccionado ninguna opción, se mostrará un mensaje de error y no se podrá pasar a la siguiente pregunta hasta que no seleccione una opción.



Figura 6.12: Boceto de la pantalla de una pregunta de test de tipo selección única.

Pregunta de selección múltiple

Con esta vista se mostrará al usuario una pregunta de test de tipo selección múltiple. En ella se mostrará la pregunta y las opciones de respuesta. El usuario podrá seleccionar todas las opciones que considere correctas y pulsar en el botón de comprobar para comprobar si las respuestas son correctas o no. Si las respuestas son correctas, se mostrará un mensaje de felicitación y se mostrará el botón de siguiente para pasar a la siguiente pregunta. Si las respuestas son incorrectas, se mostrará un mensaje de error y se mostrará el botón de siguiente para pasar a la siguiente pregunta. Si el usuario no ha seleccionado ninguna opción, se mostrará un mensaje de error y no se podrá pasar a la siguiente pregunta hasta que no seleccione una opción.

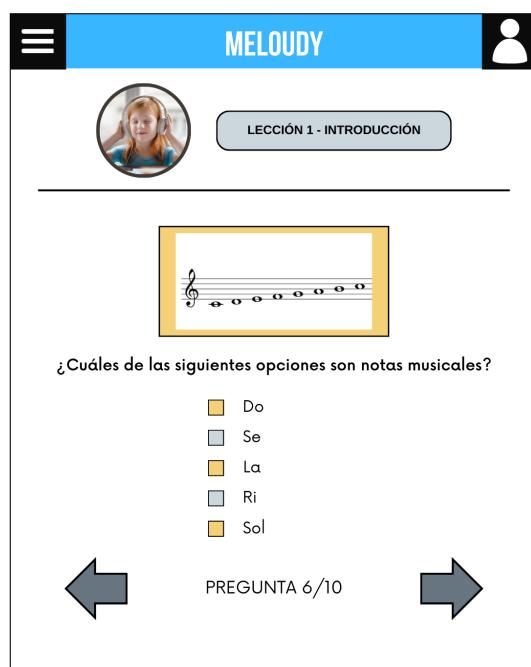


Figura 6.13: Boceto de la pantalla de una pregunta de test de tipo selección múltiple.

Pregunta de escritura de texto

Con esta vista se mostrará al usuario una pregunta de test de tipo escritura de texto. En ella se mostrará la pregunta y el usuario deberá escribir la respuesta en el campo de texto. El usuario podrá pulsar en el botón de comprobar para comprobar si la respuesta es correcta o no. Si la respuesta es correcta, se mostrará un mensaje de felicitación y se mostrará el botón de siguiente para pasar a la siguiente pregunta. Si la respuesta es incorrecta, se mostrará un mensaje de error y se mostrará el botón de siguiente para pasar a la siguiente pregunta. Si el usuario no ha escrito ninguna respuesta, se mostrará un mensaje de error y no se podrá pasar a la siguiente pregunta hasta que no escriba una respuesta.



Figura 6.14: Boceto de la pantalla de una pregunta de test de tipo escritura de texto.

Pregunta de entrada por micrófono

Con esta vista se mostrará al usuario una pregunta de test de tipo entrada por micrófono. En ella se mostrará la pregunta y el usuario deberá pulsar en el botón de grabar para grabar la respuesta, que deberá ser la melodía que pida la pregunta. El usuario podrá pulsar en el botón de comprobar para comprobar si la respuesta es correcta o no. Si la respuesta es correcta, se mostrará un mensaje de felicitación y se mostrará el botón de siguiente para pasar a la siguiente pregunta. Si la respuesta es incorrecta, se mostrará un mensaje de error y se mostrará el botón de siguiente para pasar a la siguiente pregunta. Si el usuario no ha grabado ninguna respuesta, se mostrará un mensaje de error y no se podrá pasar a la siguiente pregunta hasta que no grabe una respuesta.

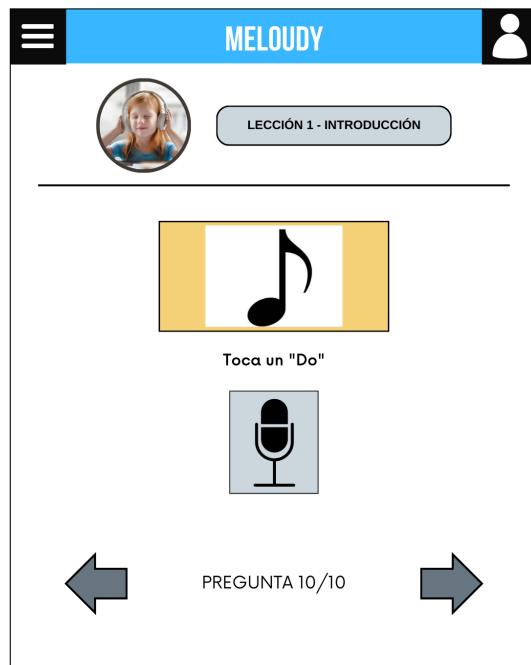


Figura 6.15: Boceto de la pantalla de una pregunta de test de tipo entrada por micrófono.

Diagrama de navegación

Para finalizar la sección se presenta un diagrama de navegación de la aplicación, el cual muestra las distintas pantallas que tendrá la aplicación siguiendo el flujo que un usuario tendría al utilizar la aplicación.

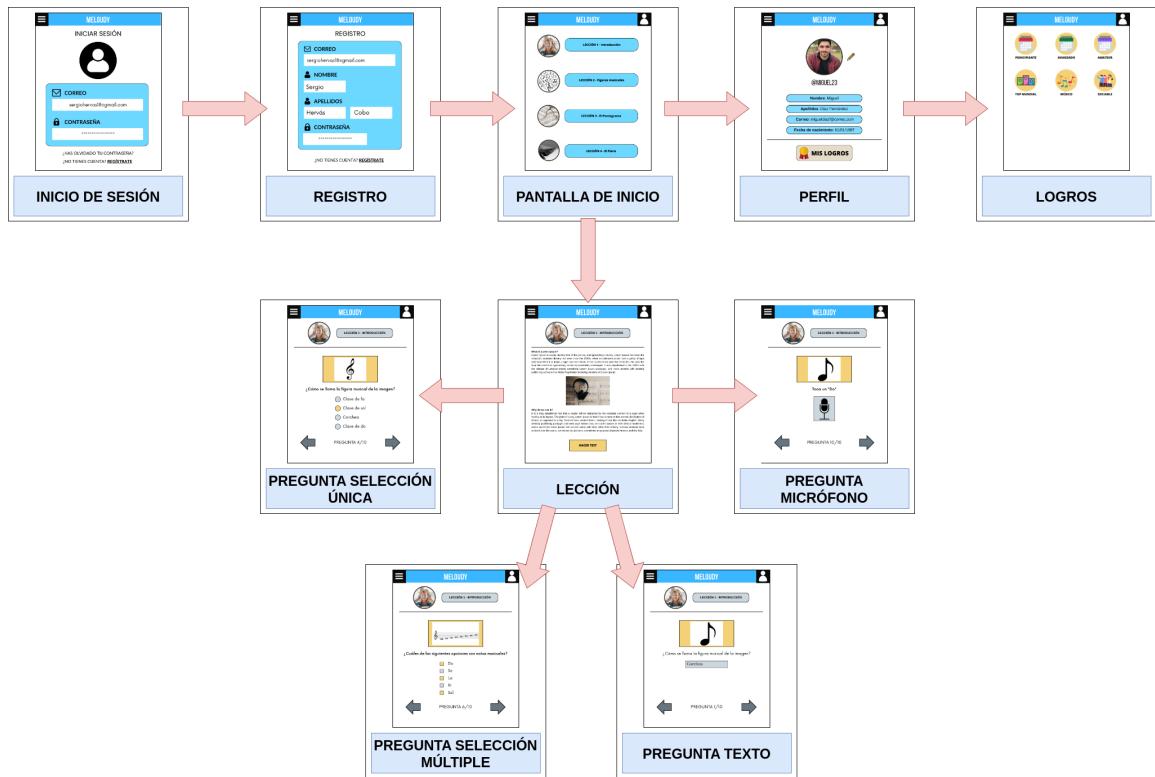


Figura 6.16: Diagrama de navegación de la aplicación por los distintos bocetos presentados anteriormente que muestra las pantallas que seguirá el usuario cuando utilice la aplicación.

Capítulo 7

Implementación

En este séptimo capítulo se detallará la implementación del sistema. En primer lugar, se presentará la estructura del proyecto que se ha realizado para el frontend y del backend y, a continuación, se describirá la implementación de cada una de las funcionalidades del sistema, presentando la documentación de cada ruta de la API y describiendo el código escrito durante el desarrollo.

7.1. Herramientas utilizadas

En el desarrollo de la aplicación se utilizarán varias herramientas externas que ayudarán en el cumplimiento de los objetivos y facilitarán la implementación de cada una de las historias de usuario. A continuación se muestran algunas de ellas:

7.1.1. Git & Github

Git es una herramienta para el control de versiones. Nos permitirá mantener un historial de cambios que nos ayudarán en la implementación cuando necesitemos "volver atrás", así como poder trabajar en distintas ramas cuando la situación lo requiera (por ejemplo, si encontramos un error puedo abrir una nueva rama para poder trabajar exclusivamente en arreglarlo).

Como complemento a Git, usaremos Github, una plataforma para alojar repositorios de Git. En Github tendremos un repositorio para el frontend y para el backend, donde se irán subiendo los cambios que se vayan realizando. Esto facilitará la posibilidad de trabajar en el desarrollo en varios dispositivos permitiendo un desarrollo más flexible en cuanto a en qué dispositivo y cuándo trabajar.

7.1.2. Jira

Jira es una herramienta para la gestión de proyectos. Usaré Jira para facilitar el cumplimiento de algunos de los principios de metodologías ágiles que usamos en la planificación y controlar el seguimiento de esta última. En Jira tendré un Sprint Backlog, cuyas historias de usuario y tareas irán moviéndose a los distintos Sprints para su desarrollo.

Proyectos / Meloudy

Backlog

SC Insights

#5 - Registro y Login Usuarios 17 feb - 3 mar (5 incidencias)

MEL-1 Como usuario quiero registrarme en la aplicación.

MEL-2 Como usuario quiero iniciar sesión en la aplicación.

MEL-6 Como usuario quiero seleccionar una lección para leer el temario.

MEL-28 Como profesor quiero ver una lista de todas las lecciones.

MEL-31 Como usuario quiero ver todas las lecciones.

+ Crear Incidencia

2 0 0 Iniciar sprint

#6 - Tests y progreso 3 mar - 17 mar (6 incidencias)

MEL-7 Como usuario quiero empezar un test de una lección.

MEL-8 Como usuario quiero responder una pregunta de un test del tipo selección múltiple.

MEL-9 Como usuario quiero responder una pregunta de un test del tipo selección única.

MEL-10 Como usuario quiero responder una pregunta de un test del tipo respuesta por micrófono.

MEL-11 Como usuario quiero ver el resultado de un test.

MEL-13 Como usuario quiero ver mi progreso en las lecciones.

+ Crear incidencia

0 0 0 Iniciar sprint

Figura 7.1: Captura de pantalla de Jira. En ella se puede ver el Sprint Backlog, donde se encuentran las historias de usuario repartidas entre los sprints.

7.1.3. Visual Studio Code & Android Studio

Visual Studio Code es el editor de código fuente que se usará para implementar la parte backend (NodeJS) de nuestro sistema. Nos permitirá un desarrollo más cómodo y eficiente, ya que nos ofrece una gran variedad de herramientas para el desarrollo, como la posibilidad de depurar el código, herramientas para el formato de los distintos lenguajes de programación, visualizador de versiones de Git, etc.

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android. Se utilizará para implementar la parte de frontend (flutter)

y para ejecutar la aplicación en un simulador de Android o en nuestro propio dispositivo físico.

7.1.4. Mongo Compass

Mongo Compass es una herramienta interactiva para la consulta, la gestión y el análisis de los datos en MongoDB. Se utilizará para comprobar si las consultas se realizan correctamente y para añadir datos a la base de datos (lecciones, preguntas...) de forma fácil, cómoda y rápida.

meloudydb.questions

The screenshot shows the Mongo Compass interface with the 'meloudydb.questions' collection selected. The 'Documents' tab is active. At the top, there's a 'FILTER' button with a query '{ field: 'value' }'. Below it is an 'ADD DATA' button with a dropdown arrow. The main area displays three documents:

```
_id: ObjectId('6405144d20563dd96aaf6702')
tipo: "unica"
cuestion: "¿Qué es la música?"
> respuestascorrectas: Array
> opciones: Array
imagen: "queeslamusica.png"
lección: ObjectId('63f94104d79a4308398b1f07')

_id: ObjectId('6405155820563dd96aaf6704')
tipo: "unica"
cuestion: "¿Qué es el sonido?"
> respuestascorrectas: Array
> opciones: Array
imagen: "queeselsonido.png"
lección: ObjectId('63f94104d79a4308398b1f07')

_id: ObjectId('6405156b20563dd96aaf6705')
tipo: "multiple"
cuestion: "¿Cuáles de las siguientes opciones son cualidades de un sonido?"
> respuestascorrectas: Array
> opciones: Array
imagen: "cualidadessonido.png"
```

Figura 7.2: Captura de pantalla de Jira. En ella se puede ver el Sprint Backlog, donde se encuentran las historias de usuario repartidas entre los sprints.

7.1.5. Postman

Postman es una plataforma API para diseñar, construir, probar e iterar APIs. Se utilizará en la implementación de nuestra aplicación para comprobar el funcionamiento de las rutas especificadas y codificadas en el backend. Además, podremos realizar una documentación de nuestra API mediante esta herramienta para facilitar la comprensión y el mantenimiento de cada ruta.

7.2. Estructura del proyecto

Como se mencionó en la arquitectura del sistema en el anterior capítulo, el proyecto se ha desarrollado en dos partes, una parte de frontend y otra de backend. La parte de frontend se ha desarrollado en Flutter y la parte de backend se ha desarrollado en Node.js.

La estructura del proyecto se puede ver en la siguiente imagen:



Figura 7.3: Estructura del proyecto, donde se puede ver la parte de frontend y la parte de backend.

En la figura anterior podemos ver que el desarrollo del proyecto se ha realizado en dos partes, una parte de frontend y otra de backend.

La parte de backend se ha desarrollado en Node.js, por lo que se han dividido los archivos en carpetas de acuerdo a su funcionalidad:

- *app.js*: Archivo principal de la aplicación. En él se configura el servidor y se importan las rutas.
- *routes*: Carpeta que contiene las rutas de la API. Dentro se encuentran distintos archivos que contienen las rutas de cada entidad de la base de datos (user, lesson, question...).
- *controllers*: Carpeta que contiene los controladores de las rutas de la API. Dentro se encuentran distintos archivos que contienen los controladores de cada entidad de la base de datos (user, lesson, question...)
- *models*: Carpeta que contiene los modelos de la base de datos.
- *middleware*: Carpeta que contiene los middleware de la aplicación. Por dichos middleware pasarán algunas peticiones que se hagan a la API.

La parte de frontend se ha desarrollado en Flutter, por lo que se han dividido los archivos en carpetas de acuerdo a su funcionalidad, dentro de la carpeta *lib*:

- *main.dart*: Archivo principal de la aplicación. En él se configura la aplicación y se importan las rutas.
- *providers*: Carpeta que contiene los providers de la aplicación. Los providers son clases que se encargan de gestionar el estado de la aplicación.
- *screen*: Carpeta que contiene las pantallas de la aplicación. Dentro se encuentran distintos archivos que implementan cada pantalla de la aplicación.
- *widgets*: Carpeta que contiene los widgets de la aplicación. Dentro se encuentran los archivos de los widgets que se utilizan en las distintas pantallas de la aplicación.

7.3. Definición de la API (Backend)

En esta sección se describirá la API que se ha desarrollado para el sistema. La API se ha desarrollado en Node.js y se ha utilizado el framework Express para trabajar con el protocolo HTTP.

7.3.1. Rutas

7.3.1.1. Usuarios

La API cuenta con las siguientes rutas para la gestión de usuarios:

- *POST /api/user/registro*: Ruta para registrar un usuario. Recibe los datos del usuario en el cuerpo de la petición, encripta la contraseña y almacena todos los datos en la base de datos. Si el usuario se registra correctamente, se devuelve un token que se utilizará para la autenticación de las peticiones que se hagan a la API.
- *POST /api/user/login*: Ruta para iniciar sesión. Recibe los datos del usuario en el cuerpo de la petición y comprueba si el usuario existe en la base de datos. En caso de que el usuario exista, se comprueba que la contraseña sea correcta. Si la contraseña es correcta, se devuelve un token que se utilizará para la autenticación de las peticiones que se hagan a la API.

- *GET /api/user/get-user/{id}*: Ruta para obtener los datos de un usuario. Recibe por parámetro el id del usuario y devuelve los datos de dicho usuario.
- *GET /api/user/get-users*: Ruta para obtener los datos de todos los usuarios. No recibe ningún parámetro y devuelve una lista de los datos de todos los usuarios.
- *DELETE /api/user/delete-user/{id}*: Ruta para eliminar un usuario. Recibe por parámetro el id del usuario y lo elimina de la base de datos.

7.3.1.2. Lecciones

La API cuenta con las siguientes rutas para la gestión de lecciones:

- *POST /api/lesson/create-lesson*: Ruta para crear una lección. Recibe los datos de la lección en el cuerpo de la petición y los almacena en la base de datos. Si la lección se crea correctamente, se devuelve la lección creada.
- *GET /api/lesson/get-lessons/{id}*: Ruta para obtener los datos de una lección. Recibe por parámetro el id de la lección y se devuelven los datos de esta.
- *PUT /api/lessons/:id*: Ruta para actualizar los datos de una lección. Recibe los datos de la lección en el cuerpo de la petición y los actualiza en la base de datos. Si la lección se actualiza correctamente, se devuelve la lección actualizada.
- *DELETE /api/lessons/:id*: Ruta para eliminar una lección. Recibe el token del usuario en el encabezado de la petición y comprueba si el token es correcto. Si el token es correcto, se elimina de la base de datos la lección cuyo id coincide con el recibido.
- *POST /api/lesson/upload-image/*: Ruta para subir una imagen de una lección al servidor. Recibe por parámetro el id de la lección y la imagen en el cuerpo de la petición.

7.3.1.3. Preguntas

La API cuenta con las siguientes rutas para la gestión de preguntas:

- *GET /api/question/get-questions*: Ruta para obtener todas las preguntas almacenadas en la base de datos.

- *GET /api/question/get-questions/{idLeccion}*: Ruta para obtener las preguntas de una lección. Recibe por parámetro el id de la lección y devuelve todas las preguntas existentes de dicha lección.
- *GET /api/question/get-question-test/{idTest}*: Ruta para obtener las preguntas de un test. Recibe por parámetro el id del test y devuelve todas las preguntas existentes de dicho test junto con el propio test.
- *GET /api/question/get-question/{id}*: Ruta para obtener los datos de una pregunta. Recibe por parámetro el id de la pregunta y devuelve los datos de dicha pregunta.

7.3.1.4. Tests

La API cuenta con las siguientes rutas para la gestión de tests:

- *GET /api/progress/get-tests-progress/{idUser}/{idLeccion}*: Ruta para obtener todos los tests de un usuario en una lección. Recibe por parámetro el id del usuario y el de la lección, busca el progreso a partir de estos identificadores, busca todos los tests de dicho progreso y los devuelve.

7.3.1.5. Progress

La API cuenta con las siguientes rutas para la gestión del progreso de los usuarios:

- *POST /api/progress/create-test-and-progress/*: Ruta para crear un test y el progreso de un usuario. Recibe los datos del test y del progreso en el cuerpo de la petición y los almacena en la base de datos. Si el progreso ya estaba creado, solamente se almacena el nuevo test y se añade a la lista de tests del progreso. Si el test y el progreso se crean correctamente, se devuelve el test y el progreso creados.

7.3.2. Middleware

Un middleware es una función intermediaria que se utiliza para realizar acciones comunes a varias funcionalidades. La API cuenta con los siguientes middleware:

- *auth*: Middleware que comprueba si el token del usuario es correcto (es decir, si el usuario está identificado en la aplicación). Si el token es correcto, se pasa a la siguiente función. Si el token no es correcto, se devuelve un error.

A continuación se explicarán todas las pantallas y funcionalidades de la aplicación. Para cada una de ellas se explicará la vista y la lógica de la misma. Para la vista, se mostrará una captura de pantalla de la pantalla en cuestión y un diagrama de la estructura de widgets que se han utilizado para construir la pantalla. Este diagrama detallará los principales widgets utilizados en cada vista. *Cabe destacar que en algunas pantallas se han omitido algunos widgets para simplificar el diagrama y se han utilizado los nombres C1, C2, C3... para referirse a los hijos del widget distribuidor en columna (Column) y F1, F2, F3... para referirse a los hijos del widget distribuidor en fila (Row).*

7.4. Funcionalidad de inicio

A continuación presentamos la funcionalidad básica de la aplicación que usará cualquier usuario que se registre:

7.4.1. Drawer

Un drawer es un menú lateral que se puede abrir y cerrar y que se utiliza para acceder a distintas pantallas de la aplicación. En este caso, se abre y se cierra clickando en el ícono superior izquierdo de la aplicación. Se ha utilizado el llamado "menú hamburguesa". En la siguiente imagen se puede ver el drawer abierto:

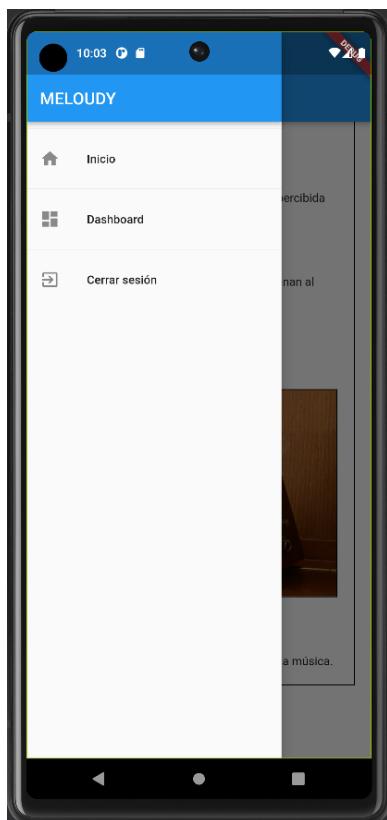


Figura 7.4: Drawer abierto en la aplicación con tres opciones (por ser desde la vista del profesor): Inicio, Dashboard y Cerrar sesión.

7.4.2. Sesión

Se han desarrollado tres características adicionales relacionadas con la sesión de usuario y que explicaremos con más detalle en cada uno de los

apartados de la sesión.

- **Encriptación de la contraseña:** La contraseña del usuario se encripta antes de ser almacenada en la base de datos para evitar que sea visible por cualquier persona que tenga acceso a esta. De esta forma garantizamos que se cumple el requisito de seguridad de la contraseña. Para esto se ha utilizado un algoritmo hash de la biblioteca *bcrypt*. El diagrama que explica el proceso de encriptación se muestra en la siguiente imagen:

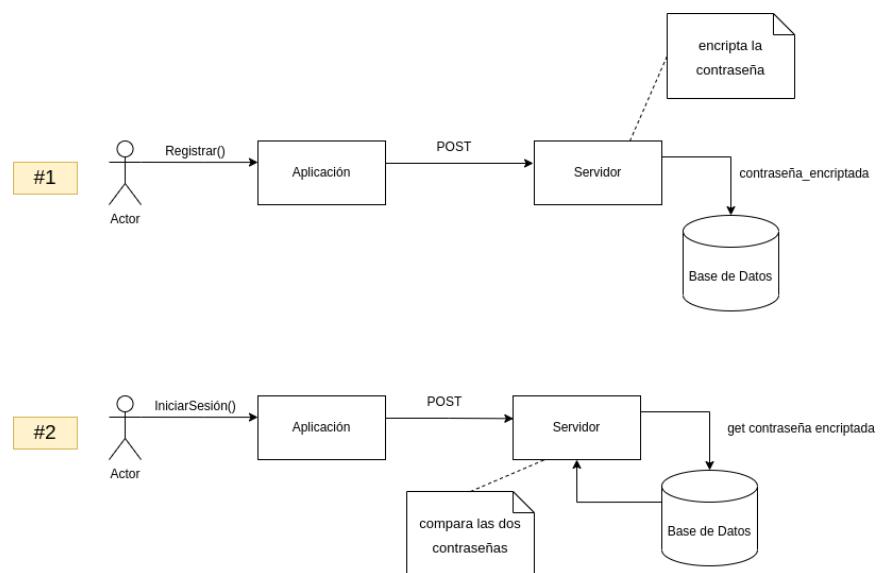


Figura 7.5: Diagrama que explica el proceso de encriptación en el inicio de sesión desarrollado en la aplicación.

- **Intercambio de token:** El servidor proporciona un token al usuario cuando se registra o inicia sesión. Este token se añade a las peticiones que se hagan a la API. De esta forma, garantizamos que se cumple el requisito de seguridad de la sesión y que todas las peticiones válidas que se hagan a la API son de usuarios registrados. Además, el token fija una fecha de expiración al crearse para aumentar la seguridad del usuario. Esta funcionalidad se ha logrado con el paquete *jsonwebtoken* de NodeJS. El diagrama que explica el proceso de intercambio de token se muestra en la siguiente imagen:

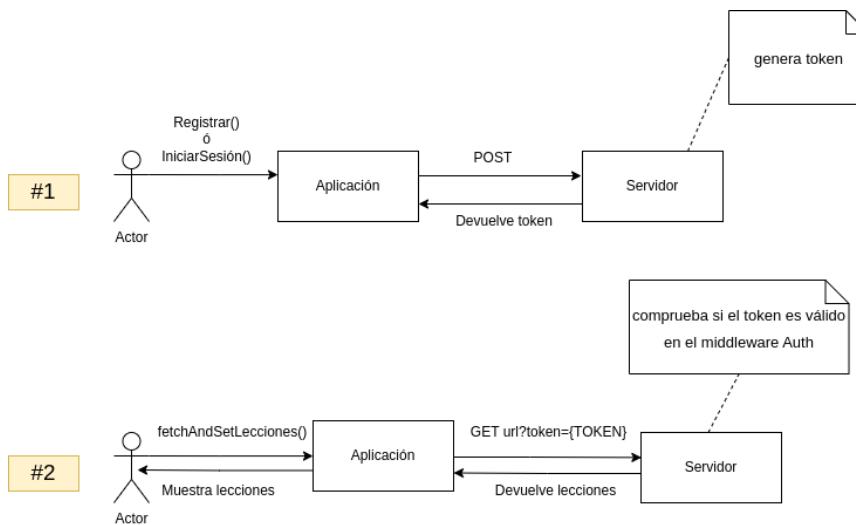


Figura 7.6: Diagrama que explica el proceso de intercambio de token en el inicio de sesión desarrollado en la aplicación.

- **Sesión entre instancias:** La sesión se mantiene entre instancias de la aplicación. Es decir, si el usuario cierra la aplicación y la vuelve a abrir, no tendrá que volver a iniciar sesión si no ha cerrado la sesión o si el token no ha expirado. Esto se ha conseguido mediante el almacenamiento del token en la memoria local del dispositivo con la biblioteca *shared_preferences*. De esta forma, cuando el usuario inicia sesión, el token se almacena en el almacenamiento local del dispositivo y cuando el usuario cierra la aplicación y la vuelve a abrir, el token se recupera del almacenamiento local del dispositivo y se comprueba si es correcto. Si es correcto, se pasa a la siguiente pantalla. Si no es correcto, se vuelve a la pantalla de inicio de sesión. Este token además estará siendo comprobado constantemente para que si el este expira, se cierre la sesión y se vuelva a la pantalla de inicio de sesión evitando accesos no autorizados.

El diagrama que explica el proceso de sesión entre instancias se muestra en la siguiente imagen:

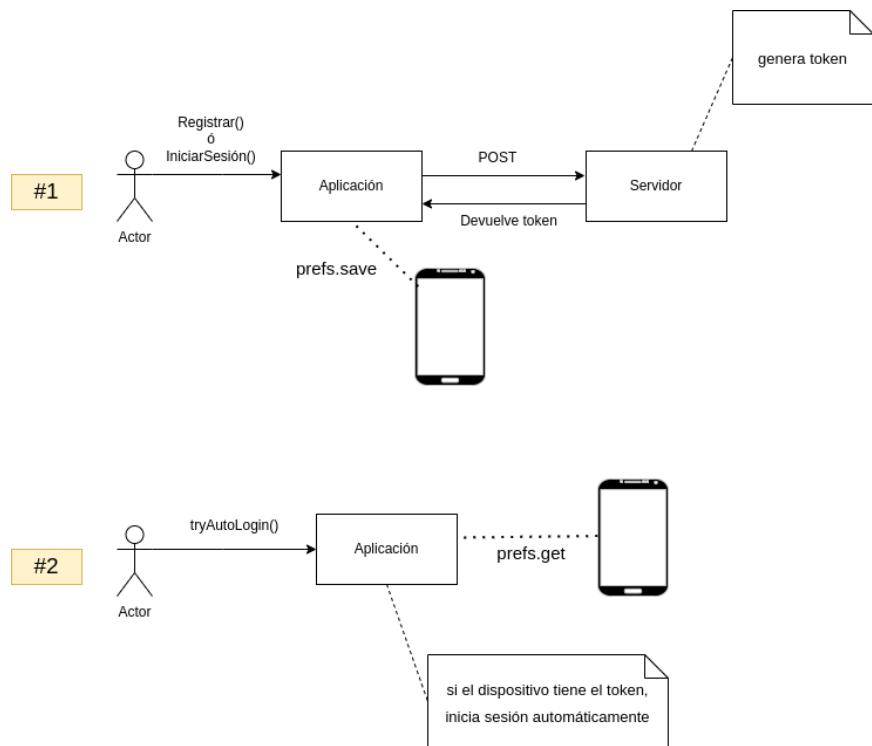


Figura 7.7: Diagrama que explica el proceso de sesión entre instancias en el inicio de sesión desarrollado en la aplicación.

7.4.2.1. Inicio de sesión

Desarrollado en el Sprint 5

La aplicación cuenta con una pantalla de inicio de sesión para que los usuarios puedan acceder a la aplicación con su cuenta registrada previamente. Para el inicio de sesión, el usuario deberá introducir su nombre de usuario y su contraseña. Si el usuario no tiene una cuenta, podrá registrarse pulsando en el botón 'Registrarse'. En la siguiente imagen se puede ver la pantalla de inicio de sesión:

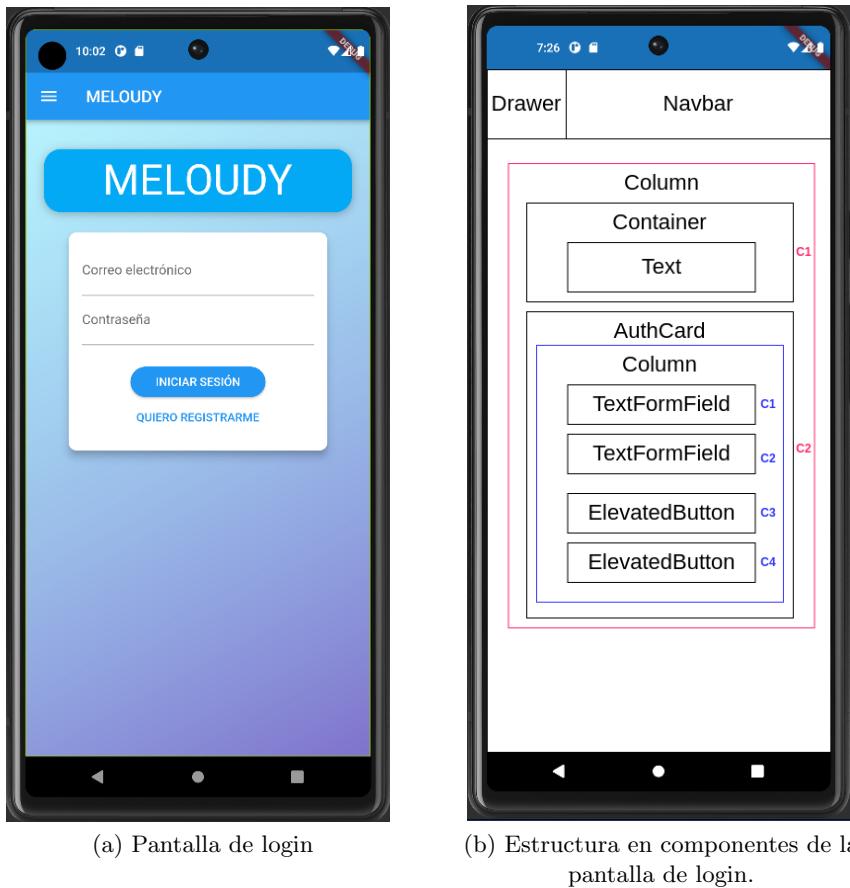


Figura 7.8: Pantalla de inicio de sesión del usuario donde se puede ver el formulario de inicio de sesión con los campos de usuario (correo) y contraseña.

Vista Empezando por la vista, se ha desarrollado un widget para la pantalla de inicio de sesión. Este widget se compone de un formulario con dos campos de texto, uno para el nombre de usuario y otro para la contraseña. Además, el formulario contiene un botón para iniciar sesión y otro para ir a registrarse si el usuario así lo desea.

La pantalla tiene una pila que contiene dos widgets: un container con un fondo en degradado conseguido con el atributo *gradient* de la clase *BoxDecoration* de Flutter, y un *SingleChildScrollView* que consta de un widget *Column* para poder colocar los elementos de forma vertical. Dichos elementos son: un *Container* para el título de la aplicación y un Widget personalizado que contiene el formulario de inicio de sesión *AuthCard()*.

Este último no es más que un widget de la clase *Card* con dos *TextField* (uno para el correo electrónico y otro para la contraseña) y un *ElevatedButton* para proceder al inicio de sesión.

Lógica En cuanto a la lógica, una vez el cliente pulsa el botón de 'Iniciar sesión', se llama a la función asíncrona *submit*, la cual se encarga de comprobar que los datos introducidos son correctos. Cuando los datos son comprobados, se procede a iniciar sesión en la aplicación mediante el método de inicio de sesión del Provider Auth. Para esto, se envía una petición HTTP POST a la ruta `/user/login` de la API con los datos escritos en el formulario de inicio de sesión.

Por otro lado, en el servidor cuando la petición es recibida a dicha ruta, se ejecuta una función en el controlador para el inicio de sesión de usuarios. Esta función capta los valores y comprueba que el usuario existe en la base de datos. Si el usuario existe, se comprueba que la contraseña introducida es correcta mediante el método *verify* de bcrypt. Si los datos son correctos, se genera el token de sesión y se devuelve al cliente. Si el usuario no existe o la contraseña es incorrecta, se devuelve un código de estado 400 y un mensaje de error.

Cuando el cliente recibe la respuesta del servidor, se comprueba si el código de estado es 200, lo que significa que el inicio de sesión ha sido correcto. En este caso, se almacena el token de sesión en el dispositivo tal y como se ha explicado en la sección 7.4.2.

7.4.2.2. Registro

Desarrollado en el Sprint 5

La aplicación cuenta con una pantalla de registro para que los usuarios puedan registrarse en la aplicación. Para el registro, el usuario deberá introducir su nombre de usuario y su contraseña dos veces. Si el usuario ya tiene una cuenta, podrá iniciar sesión pulsando en el botón 'Iniciar sesión'. En la siguiente imagen se puede ver la pantalla de registro:

Vista La vista del registro es muy similar a la del inicio de sesión. La única diferencia es que el formulario contiene un campo adicional para introducir la contraseña otra vez por razones de seguridad.



Figura 7.9: Pantalla de registro de los usuarios donde se puede ver el formulario de registro con los campos de usuario (correo), contraseña y repetir la contraseña.

Lógica En cuanto a la lógica, una vez el cliente pulsa el botón de 'Registrarse', también se llama a la función asíncrona *submit*, que se encarga de comprobar que los datos introducidos son correctos. Cuando se ha hecho esto, si los datos son válidos, la función llama al método de registro del Pro-

vider Auth, el cual envia una petición HTTP POST a la ruta `/user/registro` de la API con los datos escritos en el formulario de registro.

Para poder procesar esta petición de Flutter, una vez el servidor recibe la petición a la ruta de registro, se ejecuta una función en el controlador para el registro de usuarios. Esta función capta los valores, encripta la contraseña y crea el usuario en la base de datos. Si los datos son correctos, se genera un token de sesión y se devuelve al cliente junto con un código de estado 201 y un mensaje de éxito. Si el registro no es correcto, se devuelve un código de estado 400 y un mensaje de error.

Cuando Flutter recibe la respuesta del servidor, almacena el token de sesión en el dispositivo tal y como se ha explica en la sección 7.4.2.

7.4.2.3. Cierre de sesión

Desarrollado en el Sprint 5

La aplicación cuenta con una opción para cerrar sesión. Para cerrar sesión, el usuario deberá pulsar en el botón *Cerrar sesión* que se encuentra en el drawer (barra lateral) de la aplicación. También se cerrará sesión automáticamente si el token ha expirado.

Vista Para el cierre de sesión, se ha añadido una fila al drawer (barra lateral) de la aplicación, el cual el usuario clicará cuando desee cerrar su sesión.

Lógica La lógica del cierre de sesión es sencilla: se llama a la función *logout* de la clase *Auth* donde se restablecen los valores de la sesión y se limpian los datos de la sesión almacenados en el dispositivo con el paquete *shared_preferences*. Esta limpieza de datos se realiza en el método *clear*. Además, se ha añadido un timer que se encarga de cerrar la sesión automáticamente si el token ha expirado. Este timer se ejecuta cada cierto tiempo y comprueba si el token ha expirado. Si es así, se llama al método *logout* mencionado anteriormente.

7.5. Funcionalidad de usuario

7.5.1. Lista de lecciones

Desarrollado en el Sprint 5

La aplicación cuenta con una lista de lecciones que el usuario puede ver. Esta lista es la pantalla inicial de la aplicación y el usuario la visualizará al iniciar sesión. También puede acceder a ella pulsando el botón *Inicio* que se encuentra en el drawer (barra lateral) de la aplicación.

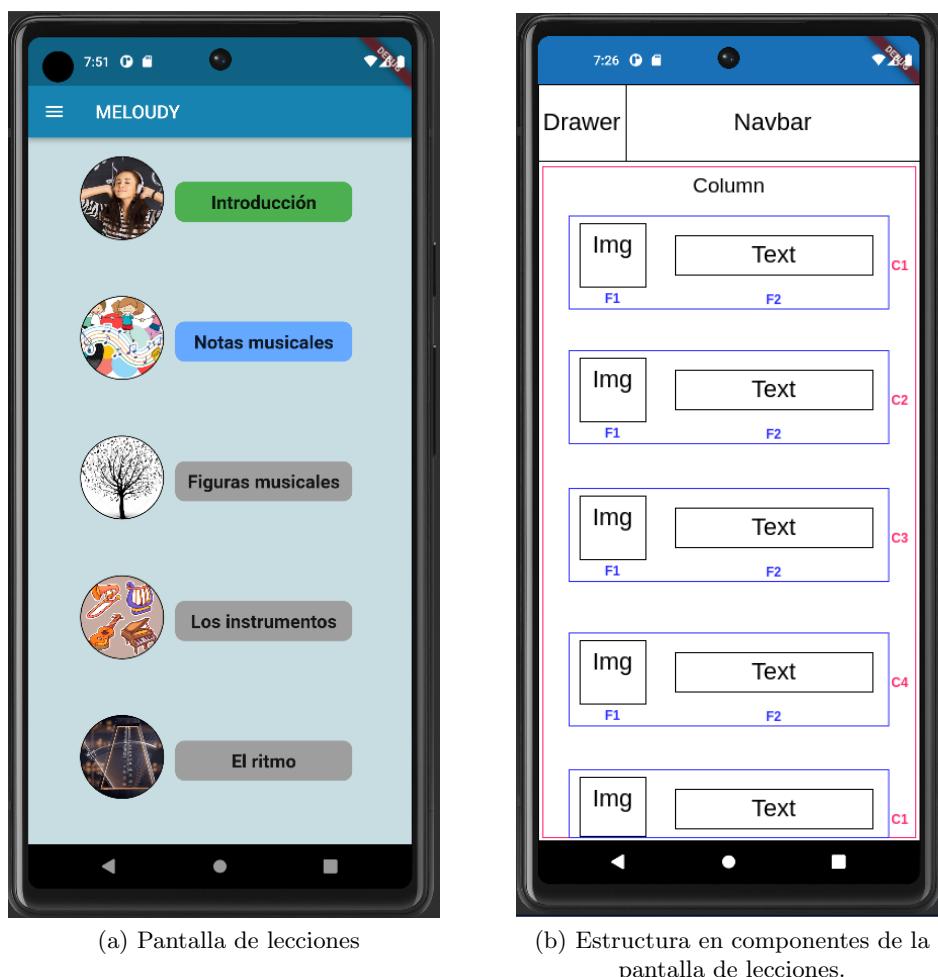


Figura 7.10: Pantalla de la lista de lecciones con el título y la imagen de cada lección.

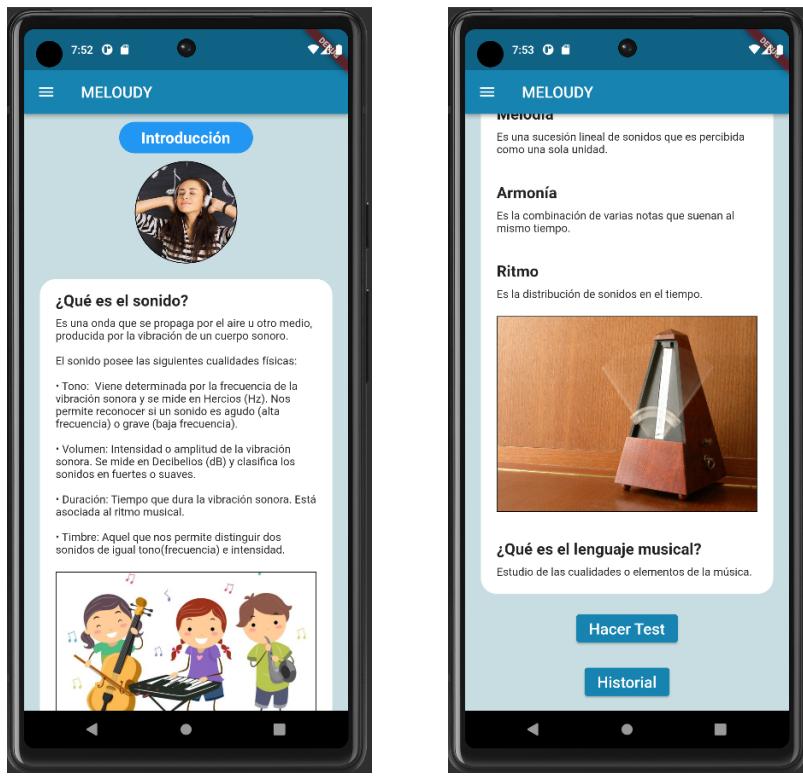
Vista Para que el usuario pueda ver la lista de lecciones, se ha desarrollado un widget principal para la pantalla. Este widget contiene el widget *ListaLecciones*, que contiene un componente para la distribución de widgets en una columna. Cada uno de estos widgets se llama *LeccionItem* y contendrá la información principal de una lección, como el título y la imagen de perfil. En la figura anterior se puede ver la pantalla de lista de lecciones.

Lógica Para la lógica de esta pantalla se ha utilizado el patrón de diseño *Provider* para la gestión de estados. Un provider es un objeto que se encarga de gestionar el estado de la aplicación y proporcionarselo a los widgets que lo necesiten. En este caso, se ha utilizado para gestionar el estado de la lista de lecciones y se le ha llamado *Lecciones*. Este Provider se encarga de obtener la lista de lecciones de la API (llamada HTTP con el verbo GET a la ruta (*GET /api/lesson/get-lessons/{id}*)) y almacenar cada uno de sus elementos en variables de la clase *Lección*. Además, se ha creado un método público que devuelve la lista de lecciones. Este método se utiliza en el widget *ListaLecciones* para obtener la lista de lecciones, añadir cada lección en forma de widget de forma iterativa y mostrarla en pantalla.

7.5.2. Ver lección

Desarrollado en el Sprint 5

La aplicación cuenta con una vista para cada lección. Esta vista se puede acceder pulsando en el título o en la imagen de una lección en la lista de lecciones. En esta vista se puede ver el contenido de la lección, como el título, el texto, las imágenes, los vídeos, etc. Además, al final de la lección se puede hacer un test para comprobar los conocimientos adquiridos y revisar los tests realizados anteriormente por el usuario.



(a) Parte 1.

(b) Parte 2.

Figura 7.11: Pantalla de una lección en concreto con los contenidos (títulos, texto, multimedia...) del temario y los dos botones relacionados con los tests.

Vista La vista de una lección proporciona al usuario información más detallada de esta. En primer lugar se muestra la imagen de la lección y el título en un componente para distribuir widgets en una fila. (Un widget *Row* compuesto de un *Text* y un *Image*). A continuación, se muestra el contenido de la lección en forma de texto, imágenes, videos, etc. Esto se ha conseguido construyendo widgets de forma dinámica dependiendo del tipo de contenido almacenado y añadiéndolos a un widget distribuidor en columna (*Column*). Finalmente, se muestra un botón en pantalla para poder hacer un test y otro para ver el historial de tests realizados, ambos mediante un widget *ElevatedButton*. En la siguiente imagen se puede ver la pantalla de una lección:

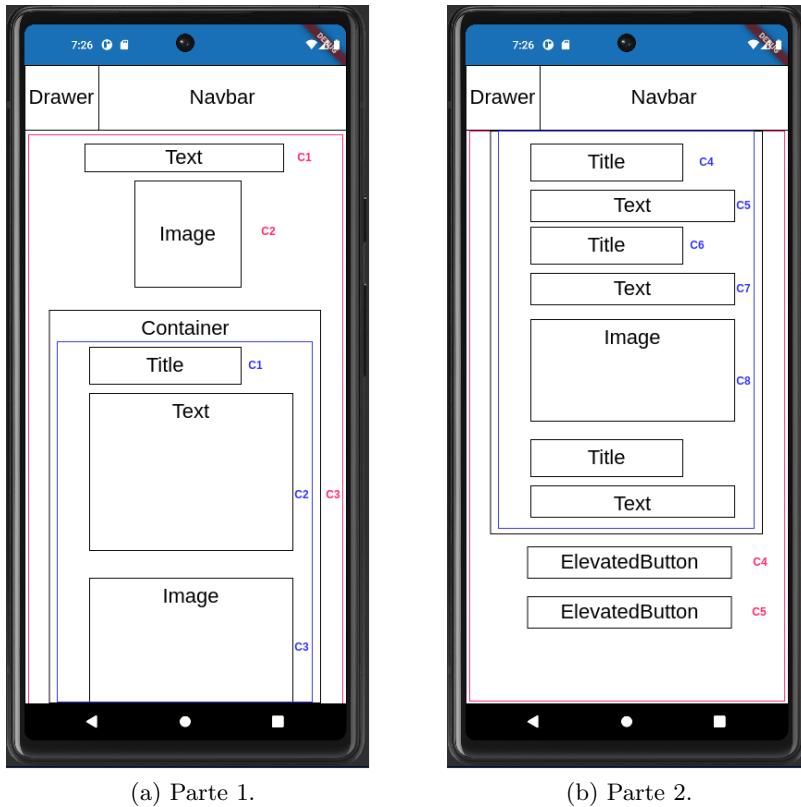


Figura 7.12: Pantalla de una lección en concreto con los contenidos (títulos, texto, multimedia...) del temario y los dos botones relacionados con los tests.

Lógica Para la lógica de esta pantalla se ha utilizado el provider *Lecciones* mencionado anteriormente. En este caso, el provider se encarga de obtener la lección que se quiere mostrar en pantalla de la lista de lecciones y devolverla al widget que la mostrará en pantalla. Además, se ha implementado una funcionalidad para poder construir el contenido de la lección de forma dinámica. Para esto se ha realizado un bucle que recorre el contenido de la lección y dependiendo del tipo de contenido, se construye un widget diferente. Por ejemplo, si el contenido es un texto, se construye un widget *Text* y se añade a la lista de widgets que se devolverán al final del método. Si el contenido es una imagen, se construye un widget *Image* y se añade a la lista de widgets que se devolverán al final del método. De esta forma se consigue que el contenido de la lección se muestre en pantalla en función del tipo y facilitaremos la adición de contenido a cada lección por parte del profesor en un futuro.

7.5.3. Test

Desarrollado en el Sprint 6

La lección contiene un test que se puede realizar al final de la lección. Este test consta de una serie de preguntas que el usuario deberá responder.

El usuario podrá responder a las preguntas de la siguiente forma: seleccionando una opción de la lista de opciones, seleccionando varias, escribiendo una respuesta en un campo de texto... El usuario podrá responder la opción que desee y pasar a la siguiente pregunta. A continuación se muestran las diferentes preguntas que se pueden realizar en el test.

Vista La vista de un test consiste en una serie de preguntas que el usuario deberá responder. Al seleccionar el botón de *Hacer Test* ubicado en la pantalla de la lección, se muestra la pantalla de un test. En esta pantalla se presenta una pregunta a la vez, y el usuario deberá responderla. Para pasar a la siguiente pregunta, pulsará la flecha ubicada a la derecha de la imagen y para retroceder a la pregunta anterior, pulsará la flecha izquierda. Esto se ha realizado mediante un widget que distribuye otros widget de forma horizontal en una fila (*Row*). Este widget se encarga de mostrar la imagen de la pregunta actual y las flechas para pasar a la siguiente o anterior pregunta.

Debajo de lo anterior, se mostrará la pregunta actual y las opciones de respuesta. Esto se ha realizado mediante un widget que distribuye otros widget de forma vertical en una columna (*Column*). Las opciones de respuesta dependerán del tipo de pregunta que se esté mostrando. Por ejemplo, si la pregunta es de selección única o múltiple, se muestra una lista de opciones de respuesta, mientras que si la pregunta es de entrada de texto, se muestra un campo de texto para que el usuario escriba su respuesta.

En la última pregunta, aparece un botón para enviar las respuestas y ver el resultado del test. Dicho botón evita que el usuario no pueda enviar las respuestas antes de responder o pasar por todas las preguntas.

Lógica Para la lógica de esta pantalla se ha utilizado el patrón de diseño *Provider* para la gestión del estado del test en curso. Este Provider se encarga de obtener todas las preguntas del test de la API y almacenarlas en una lista privada. Para recorrer las preguntas, se ha utilizado un índice que se incrementa o decrementa en función de si el usuario quiere pasar a la siguiente o a la anterior pregunta. Este índice se almacena también en una variable privada e irá aumentando o disminuyendo en función de si el usuario quiere pasar a la siguiente pregunta o volver a la anterior. Cada vez que el usuario responde una pregunta, se almacenan las respuestas en una variable privada para conservarlas durante toda la navegación del test (si vuelve para

atrás, por ejemplo) y se produce una retroalimentación al usuario cambiando el color de la opción a un azul más oscuro para que el usuario sepa qué opción ha respondido.

Cuando el usuario pulsa el botón de enviar respuestas, estas se envían al servidor para crear el test y añadirlo al progreso (se envía la petición HTTP a la ruta de la API *POST /api/progress/create-test-and-progress/*:) y aparece la pantalla para ver el resultado del test.

7.5.3.1. Pregunta de selección única.

Desarrollado en el Sprint 6

A continuación se muestra la pantalla de una pregunta de selección única, en la que el usuario deberá seleccionar una opción de la lista de opciones:

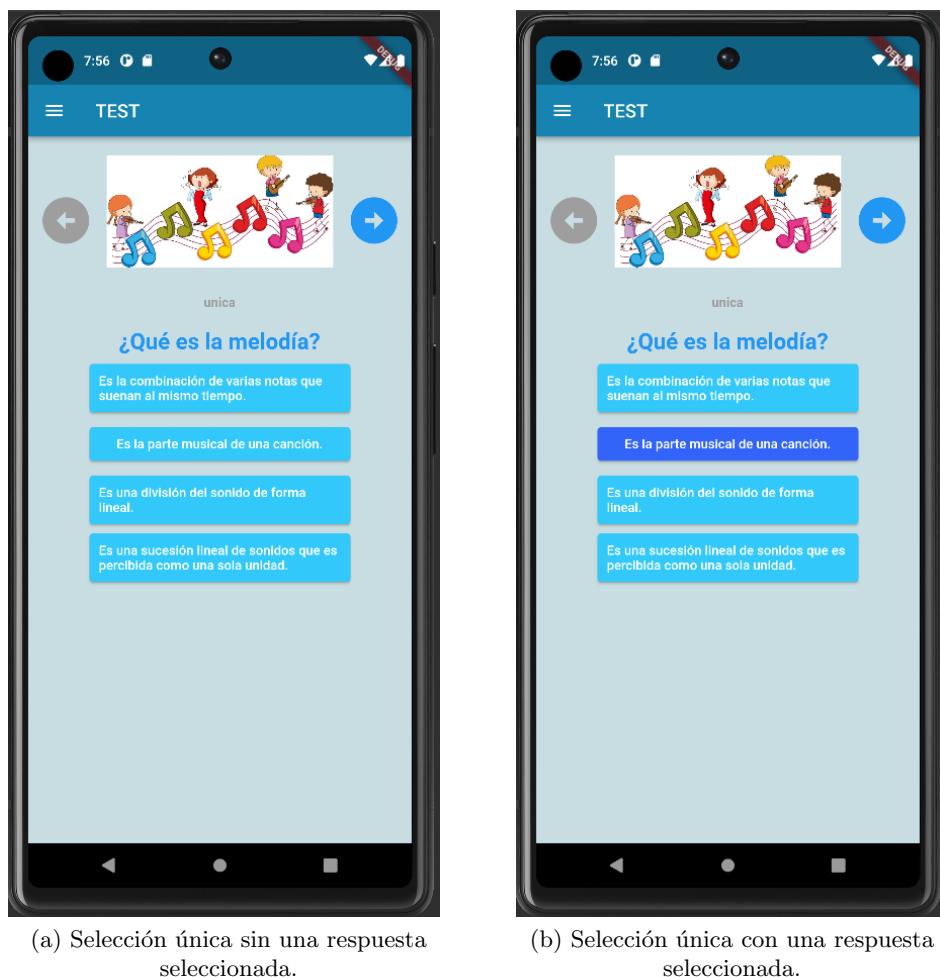


Figura 7.13: Pantalla de una pregunta de selección única en la que las opciones se muestran en forma de lista.

7.5.3.2. Pregunta de selección múltiple

Desarrollado en el Sprint 6

A continuación se muestra la pantalla de una pregunta de selección múltiple, en la que el usuario deberá seleccionar una o varias opciones de la lista de opciones:

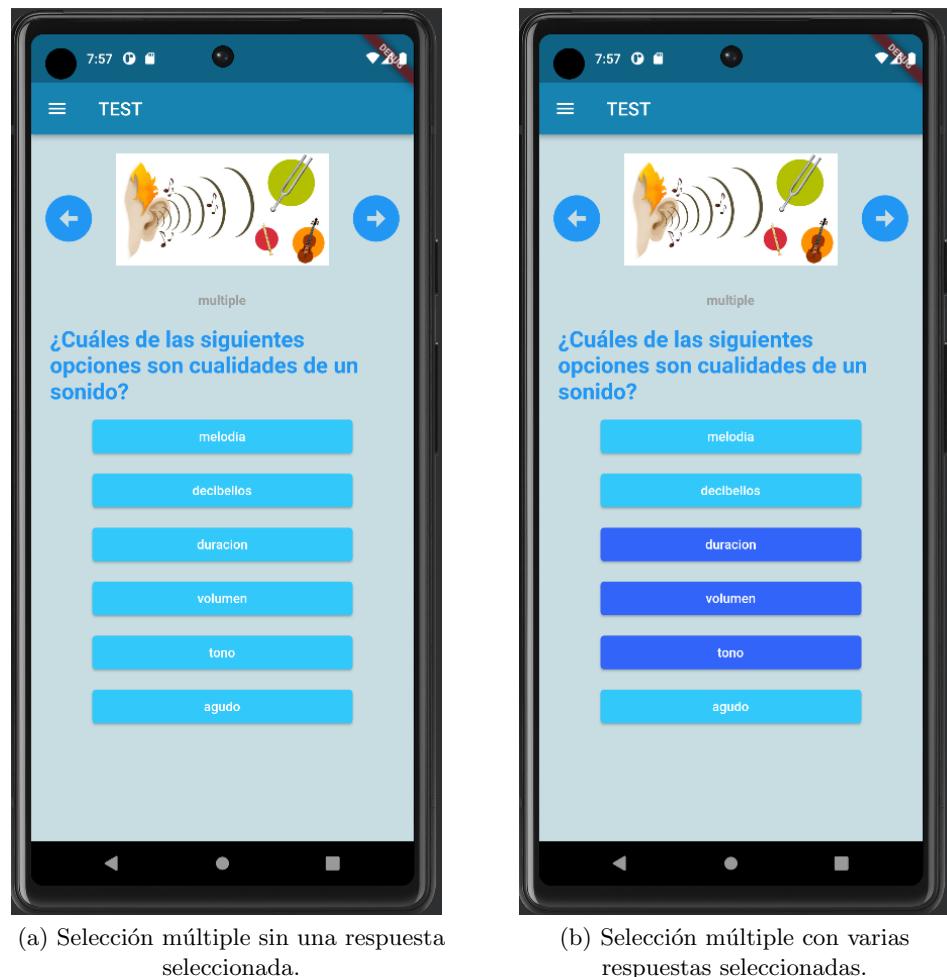


Figura 7.14: Pantalla de una pregunta de selección múltiple en la que las opciones se muestran en forma de lista.

7.5.3.3. Pregunta de entrada de texto

Desarrollado en el Sprint 6

A continuación se muestra la pantalla de una pregunta de entrada de texto, en la que el usuario deberá escribir la respuesta en un campo de texto:

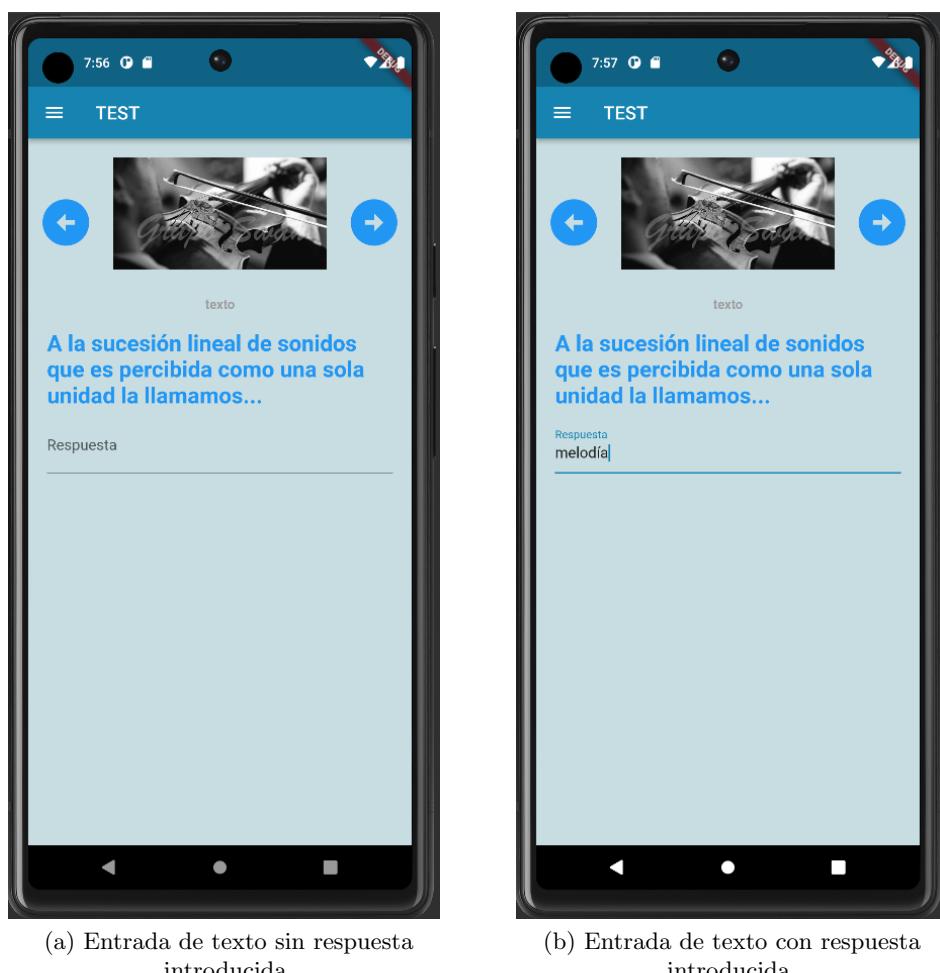
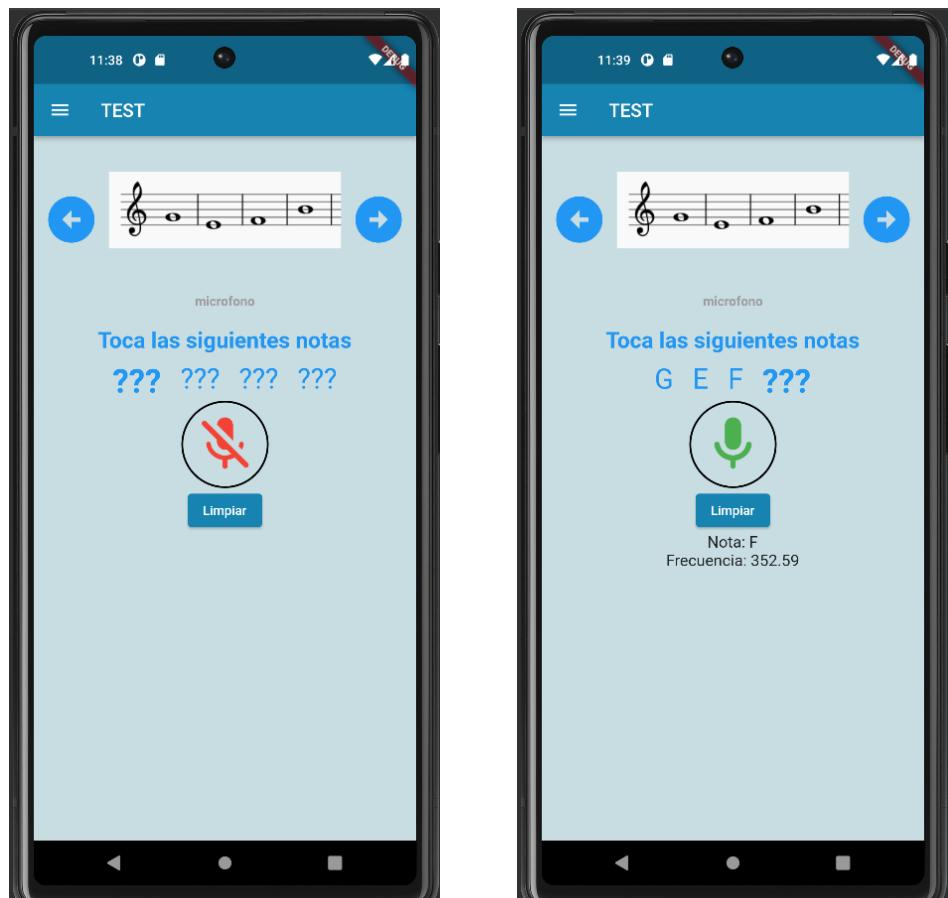


Figura 7.15: Pantalla de una pregunta de entrada de texto.

7.5.3.4. Pregunta de micrófono

Desarrollado en el Sprint 7

A continuación se muestra la pantalla de una pregunta de micrófono, en la que el usuario deberá grabar la respuesta mediante el micrófono del dispositivo. Esta respuesta debe ser fiel a la partitura que se muestra en la imagen de la pregunta.



(a) Pregunta de micrófono sin respuesta introducida.

(b) Pregunta de micrófono introduciendo respuesta.

Figura 7.16: Pantallas de una pregunta de micrófono.



Figura 7.17: Pregunta de micrófono con respuesta introducida.

Lógica Para realizar la lógica de esta pantalla se ha utilizado la biblioteca *flutter_fft*, que utiliza la transformada de Fourier de una señal de audio para obtener la frecuencia que se utiliza para determinar el tono del sonido, y obtener la nota musical que se está reproduciendo a partir de dicho valor de frecuencia. Para ello, la biblioteca utiliza un *StreamBuilder* que se encarga de obtener la frecuencia de la señal de audio. Se han realizado algunas modificaciones al código de la biblioteca para que la detección de notas sea más precisa y se pueda obtener la nota musical real que se está reproduciendo (había cierto desajuste que detectaba medio semitono más de lo que se estaba tocando).

Cabe mencionar que *flutter_fft* se le pueden ajustar varios parámetros para que la detección de notas sea más o menos precisa, como el número de muestras que se toman por segundo, el intervalo entre muestras, etc.

Cuando el usuario pulsa el botón del micrófono, se inicia el *StreamBuilder*, se cambia el estado del widget para establecer que se está grabando (variable *isRecording*) y se muestra una retroalimentación al usuario cambiando el color del botón para notificar al usuario.

Además, las notas musicales se muestran en pantalla conforme el algoritmo las identifica. De esta manera, el usuario puede comprobar la exactitud

de su interpretación en tiempo real. Cuando se completan todas las notas de la partitura, se detiene el *StreamBuilder* automáticamente. Además, en el momento en el que el usuario pulsa el botón del micrófono cuando esté grabando, también se detiene el *StreamBuilder*.

Si el usuario no está conforme con la respuesta, puede pulsar el botón de “Reiniciar” para resetear el ejercicio y puede volver a grabar pulsando el botón de micrófono. Por otro lado, si está conforme, puede pulsar el botón de siguiente para pasar a la siguiente pregunta.

7.5.3.5. Resultado del test

Desarrollado en el Sprint 6

A continuación se muestra la pantalla de resultado del test, en la que se muestra el número y el porcentaje de aciertos respecto a las preguntas. Además, se muestra una barra de progreso que muestra el porcentaje de aciertos respecto al número de preguntas preguntas. (Figura 7.18)

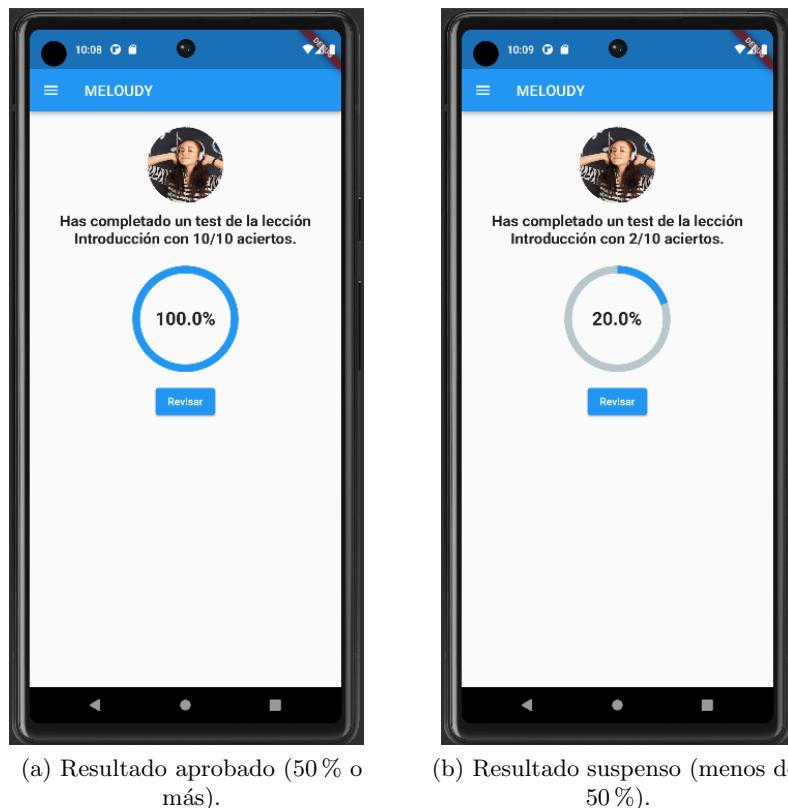


Figura 7.18: Pantalla que muestra el resultado del test con el número de aciertos y el porcentaje de aciertos respecto a las preguntas con un Progress Bar.

Vista Para realizar la vista de la pantalla se ha utilizado de nuevo el widget *Column* para poder mostrar los elementos de la pantalla de forma vertical. Dentro de dicho widget tendremos:

- la imagen de la lección (widget *Image*).
- un campo de texto para mostrar el número de aciertos (widget *Text*).
- una barra de progreso para mostrar el porcentaje de aciertos (widget *CircularPercentIndicator*). Para este test se ha utilizado la biblioteca de flutter *percent_indicator*.
- un botón para poder revisar el test (widget *ElevatedButton*).

Lógica La lógica para mostrar el resultado del test es bastante sencilla. Tras el envío del test al servidor, se comparan las respuestas del usuario con las respuestas correctas almacenadas en cada una de las preguntas (estas últimas se obtienen mediante el Provider de las preguntas). Una vez obtenidos estos datos, se calcula el porcentaje de aciertos y se envía a la vista.

7.5.4. Revisar pregunta

El usuario podrá revisar las preguntas que haya contestado en un test. Como se puede ver en la pantalla anterior del resultado del test, existe un botón para poder revisarlo. El usuario podrá ver las preguntas que haya contestado y las respuestas que haya seleccionado con la validación de estas.

Vista La vista de la pantalla de revisión de preguntas es muy similar a la vista de la pantalla de la pregunta. La diferencia es que se muestran las respuestas que ha seleccionado el usuario si son correctas o no. Por tanto, solo se han cambiado los colores de los botones y de los textos para mostrar si la respuesta es correcta o no.

Lógica La lógica para mostrar la revisión de las preguntas se ha realizado comparando las respuestas del usuario con las respuestas correctas almacenadas en cada una de las preguntas (estas últimas se obtienen mediante el Provider de las preguntas).

Para mostrar si la respuesta es correcta o no, se ha cambiado el color del botón o del campo de texto. Si la respuesta es correcta, se muestra en verde y, si es incorrecta, se muestra en rojo.

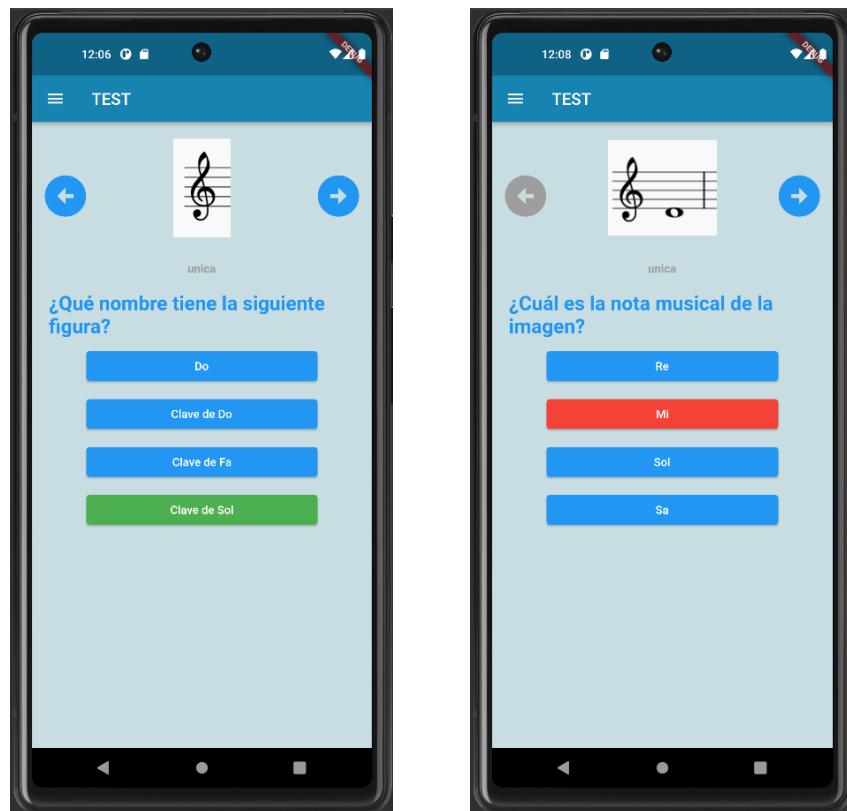
Para la validación, se ha llamado a la función *validarRespuesta* que se ha creado en la clase provider *Preguntas*. Esta función comprueba si la(s)

respuesta(s) del usuario de la pregunta actual coincide con la(s) respuesta(s) correcta(s). En caso de hacerlo, se devuelve un valor booleano *true*, y si no coincide, se devuelve un valor booleano *false*.

Para mostrar el color de los botones o de los campos de texto, se ha utilizado la función *ternario* de Dart. Esta función recibe tres parámetros: una condición, un valor si la condición es *true* y un valor si la condición es *false*. En nuestro caso, la condición es la función *validarRespuesta* que hemos creado antes. Si la condición es *true*, se devuelve el color verde, y si es *false*, se devuelve el color rojo.

7.5.4.1. Revisión de pregunta de selección única

Desarrollado en el Sprint 6



(a) Revisión de selección única acertada.

(b) Revisión de selección única fallida.

Figura 7.19: Pantalla de la pregunta de selección única en revisión.

7.5.4.2. Revisión de pregunta de selección múltiple

Desarrollado en el Sprint 6

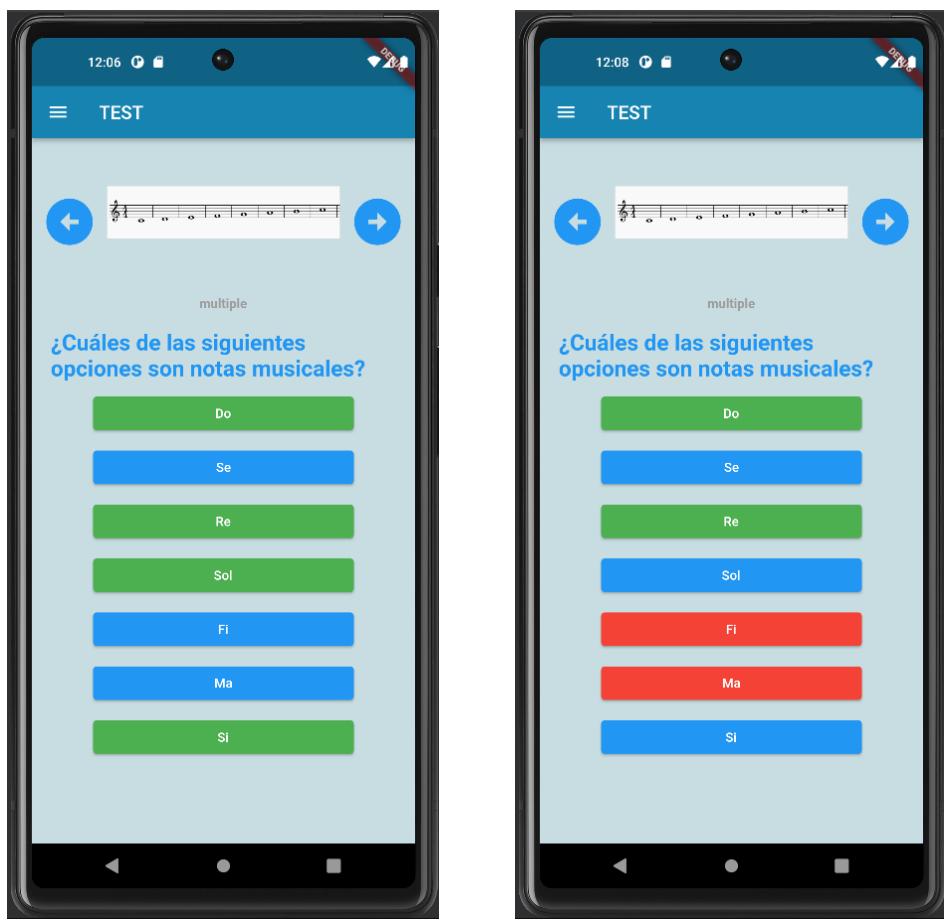
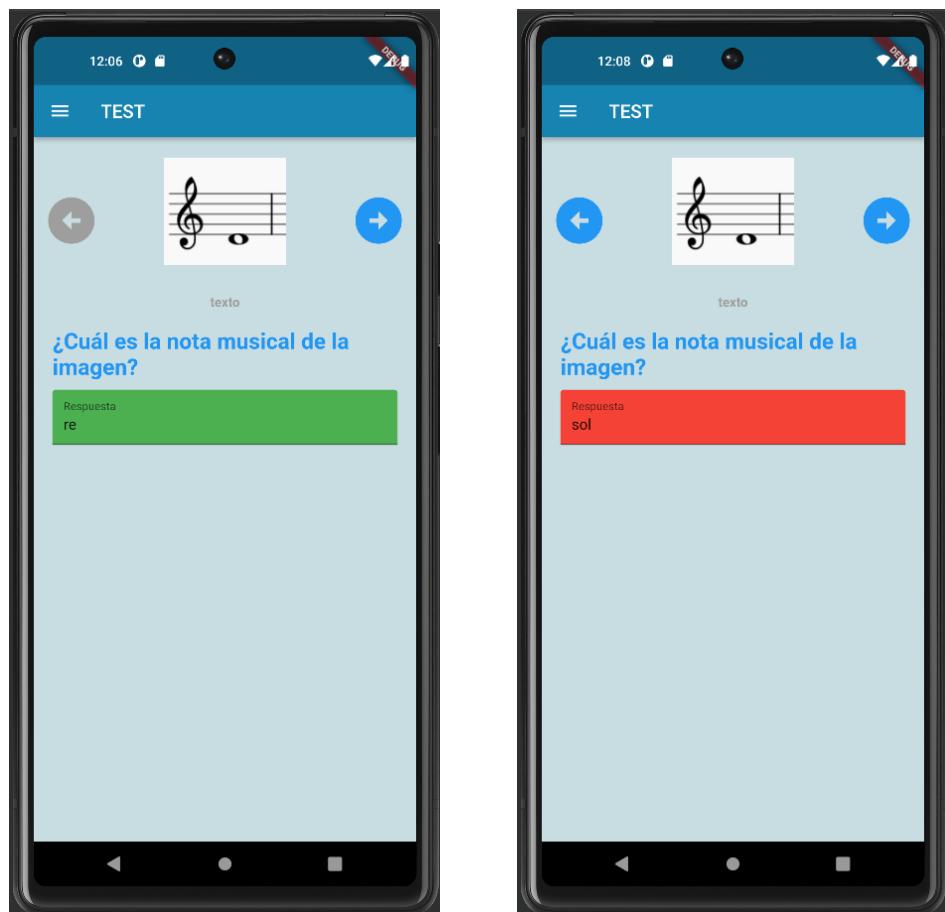


Figura 7.20: Pantalla de la pregunta de selección múltiple en revisión.

7.5.4.3. Revisión de pregunta de entrada de texto

Desarrollado en el Sprint 6



(a) Revisión de entrada de texto acertada.

(b) Revisión de entrada de texto fallida.

Figura 7.21: Pantalla de la pregunta de entrada de texto en revisión.

7.5.4.4. Revisión de pregunta de entrada de micrófono

Desarrollado en el Sprint 7

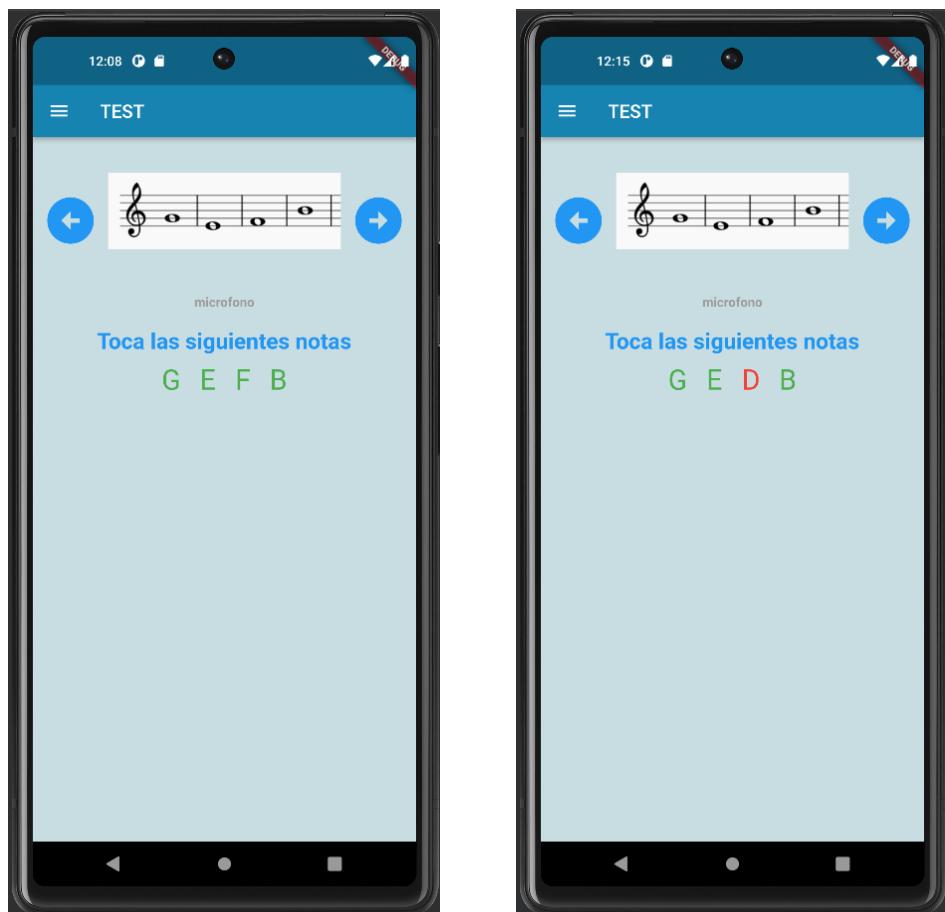


Figura 7.22: Pantalla de la pregunta de entrada de micrófono en revisión.

7.5.5. Historial de tests

Desarrollado en el Sprint 6

Vista La vista de esta pantalla consiste en un distribuidor de columnas (Column) que contiene una lista de tests (ListView) verticalmente.

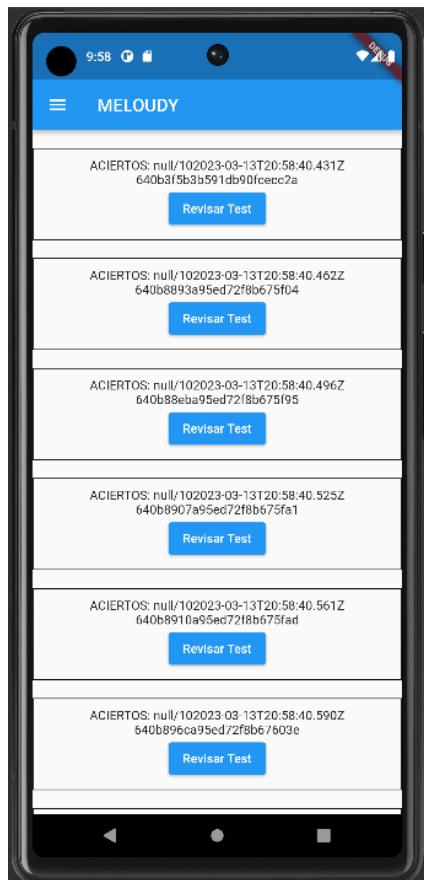


Figura 7.23: Pantalla de dashboard del profesor con las opciones de gestión disponibles para el profesor.

7.5.6. Ver Mi Perfil

Desarrollado en el Sprint 10

Vista Se ha utilizado un distribuidor de columnas (Column) que contienen los datos de los usuarios. En la parte superior de la pantalla se muestra la foto de perfil del usuario, y debajo de la foto se muestra el nombre, el apellido, el nombre de usuario (username), el correo electrónico y el rol.



Figura 7.24: Pantalla del perfil del usuario.

Lógica En primer lugar, cuando se accede al perfil se obtienen los datos del usuario mediante el método `fetchAndSetUser` de un nuevo Provider llamado `UsuarioPerfil`. Este método manda una petición *GET* a la ruta `(/api/user/get-user/{id})`. Una vez obtenidos los datos del usuario mediante el provider `UsuarioPerfil`, estos se muestran en la vista.

7.5.7. Ver logro

Desarrollado en el Sprint 11

Los usuarios podrán ver de forma detallada cada logro que hayan conseguido. Para ello, se ha desarrollado una pantalla que muestra la información de un único logro (imagen, título y descripción). A esta pantalla se accederá cuando el usuario pinche en uno de los logros dentro de su perfil.

Vista Esta pantalla consta de un distribuidor de columnas (Column) de contenedores que contienen la información a mostrar. En la siguiente imagen se puede ver la pantalla de un logro:



Figura 7.25: Pantalla de visualización de logro asignado.

Lógica Para cargar la información del logro no ha hecho falta llamar al provider pues la información del logro se encuentra en el provider Usuario-Perfil. Por tanto, solo ha hecho falta obtener dichos datos y cargarlos en la nueva vista.

7.5.8. Mis logros

Desarrollado en el Sprint 11

Los usuarios podrán ver una lista de todos los logros que hayan conseguido. Hay que tener en cuenta que en el perfil del usuario solo se muestran algunos. A esta lista se accederá cuando el usuario pulse el botón "Más" de la sección de logros del perfil.

Vista La información visual de esta funcionalidad consiste en un distribuidor de columnas (Column) de contenedores que contienen la información a mostrar de cada logro. Cada una de las filas contiene un distribuidor de filas (Row) que se encargará de visualizar a la izquierda la imagen de perfil y a la derecha la información de texto (esta última información a su vez está dentro de otro widget Column). En la siguiente imagen se puede ver la pantalla de los logros del usuario::



Figura 7.26: Pantalla de visualización de la lista de logros asignados.

Lógica Para cargar la lista de logros no ha hecho falta llamar al provider pues los datos de los logros se encuentran en el provider UsuarioPerfil. Por tanto, solo ha hecho falta obtener dichos datos y cargarlos en la nueva vista.

7.6. Funcionalidad de gestión de la aplicación

A continuación presentamos las funcionalidades de gestión de la aplicación. El profesor podrá gestionar las lecciones y las preguntas de la aplicación. Por otro lado, el administrador podrá gestionar, además de las lecciones y las preguntas, los usuarios y los logros de la aplicación.

7.6.1. Dashboard

Desarrollado en el Sprint 5

Un dashboard es una interfaz de usuario que muestra información de forma resumida y que permite al usuario acceder a las diferentes funcionalidades de la aplicación. En este caso, el dashboard del profesor muestra las opciones de gestión de lecciones y preguntas.

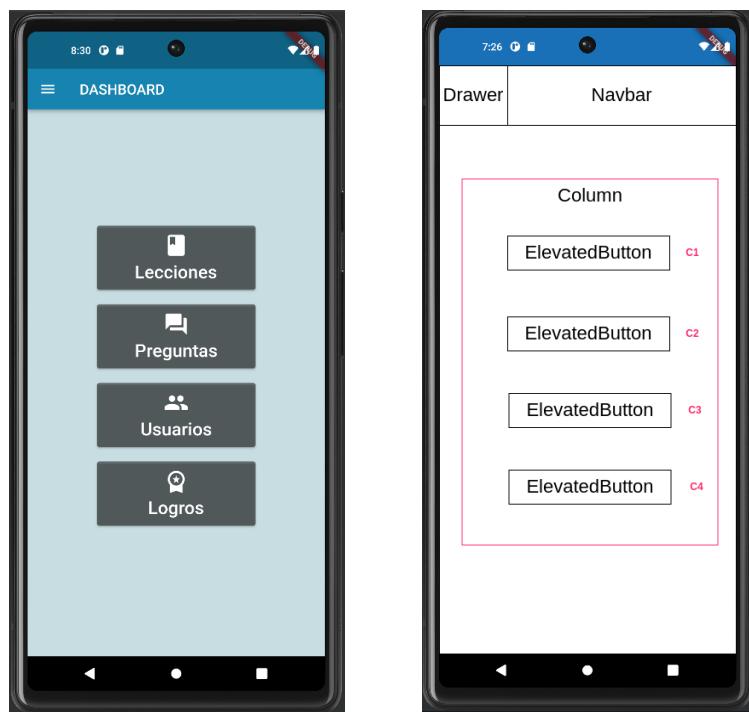


Figura 7.27: Vista del dashboard

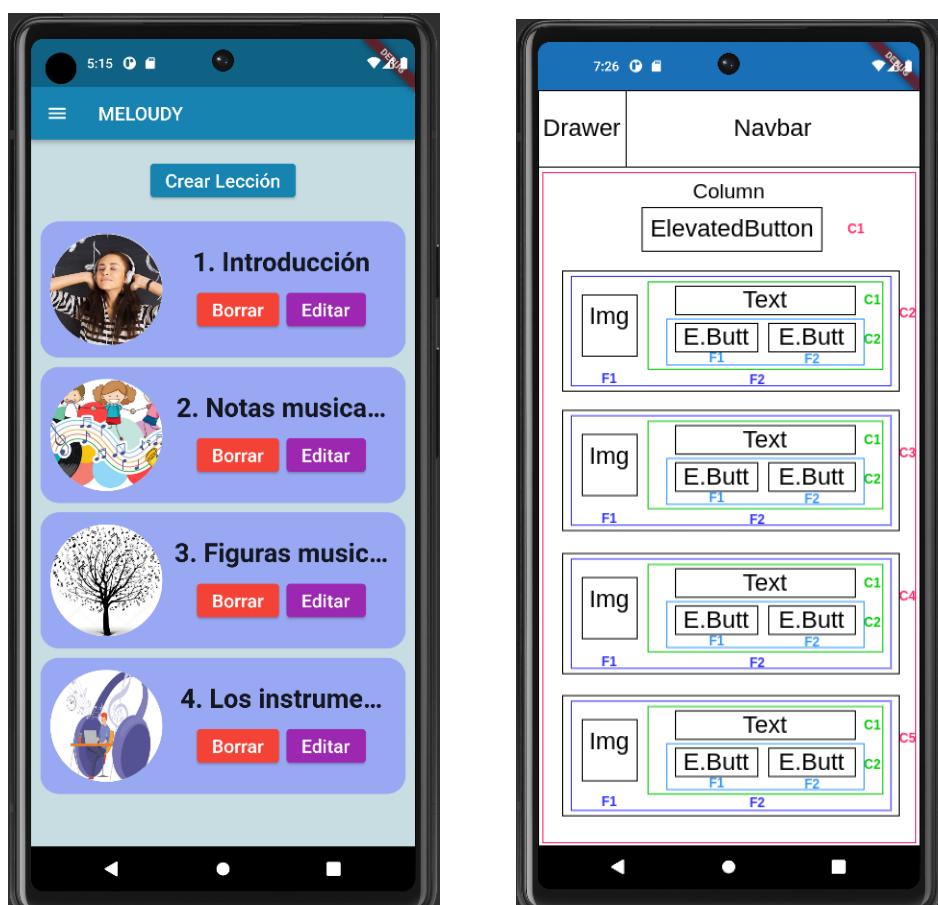
Vista Se ha implementado una pantalla de dashboard para la gestión de la aplicación. En esta pantalla, se muestran las opciones de gestión disponi-

bles para la persona que utiliza la aplicación en función de su rol mediante una columna de ElevatedButton. En la figura 7.27 imagen se puede ver la pantalla de dashboard.

7.6.2. Lista de lecciones

Desarrollado en el Sprint 5

Vista La lista de lecciones del profesor es muy similar a la del usuario. Las diferencias, aparte de los estilos, son los botones de tipo ElevatedButton que se han añadido para la gestión de las lecciones y que permiten al profesor crear, modificar y eliminar lecciones. En la siguiente imagen se puede ver la pantalla de lista de lecciones del profesor:



(a) Pantalla de lista de lecciones del profesor para la gestión de estas.

(b) Composición de la pantalla de la lista de lecciones.

Figura 7.28: Vista de la lista de las lecciones.

7.6.3. Borrar lección

Desarrollado en el Sprint 8 El profesor podrá borrar una lección de la lista de lecciones al pulsar el ícono de la papelera mostrado en la vista del apartado anterior. Una vez pulsado, se mostrará un diálogo de confirmación para que el profesor confirme que desea borrar la lección.

Vista La pantalla del borrado de lecciones utiliza el widget *AlertDialog*, que posee un título, un texto descriptivo y dos botones: uno para cancelar el borrado y otro para confirmarlo. En la siguiente imagen se puede ver la pantalla de borrado de lecciones:



Figura 7.29: Pantalla del borrado de una lección.

Lógica Una vez el botón de *Confirmar* es pulsado, se muestra un diálogo de confirmación para que el profesor confirme que desea borrar la lección. Este borrado se realiza llamando a la función `borrarLeccion()` del provider *Lecciones*. En dicha función se borrará la lección de la lista del provider y se actualizará la base de datos enviando una petición `DELETE` a la API con el id de la lección a borrar. La ruta de la petición es `DELETE /api/lesson/delete-lesson`. La API devolverá un código de estado 200 si la operación se ha realizado correctamente y se actualizará la lista de lecciones del provider. En caso contrario, se mostrará un mensaje de error.

7.6.4. Crear lección

Desarrollado en el Sprint 8

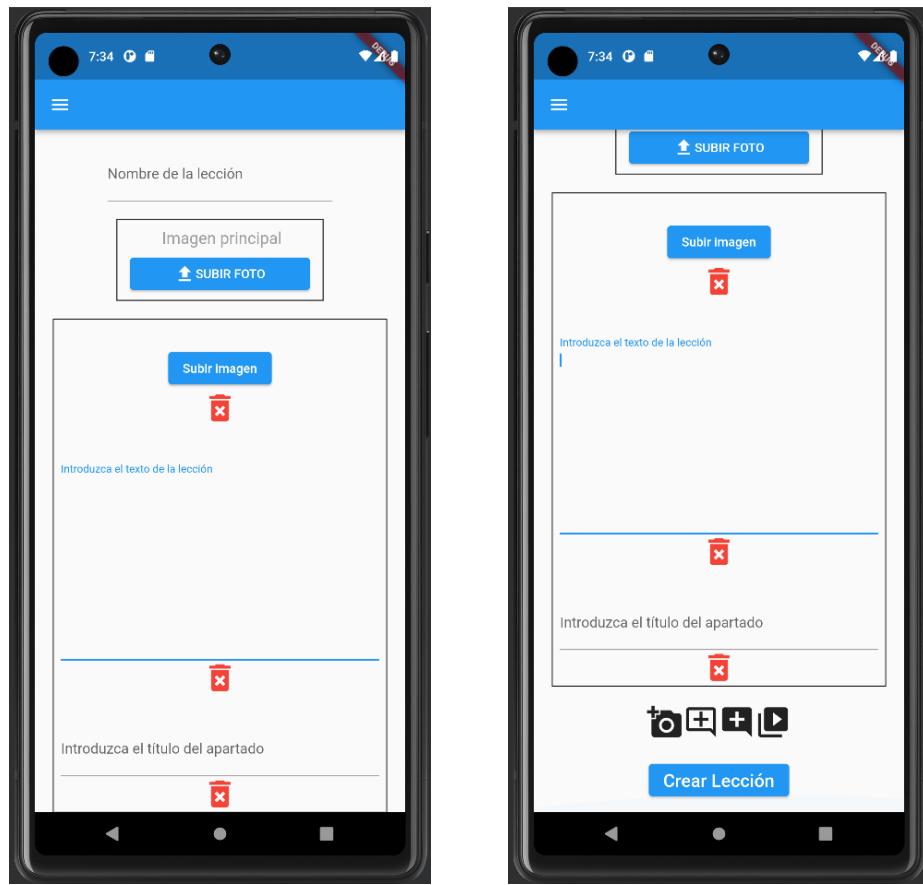
Los profesores podrán crear lecciones para que los usuarios puedan estudiarlas. Para ello, se ha implementado una pantalla de creación de lecciones que permite al profesor añadir títulos, textos explicativos y contenido multimedia a la lección. Una vez creada, la lección se añadirá a la lista de lecciones del profesor.

Vista La vista de esta pantalla consiste en un formulario de creación de lecciones. Este formulario consta de:

- Un campo de texto para el título de la lección.
- Un botón para subir la imagen de la lección.
- Un número variable de campos de texto para los títulos.
- Un número variable de campos de texto para los textos explicativos.
- Un número variable de botones (*ElevatedButton*) para las imágenes.
- Un número variable de campos de texto para los vídeos.

Todo esto está distribuido mediante el widget que organiza los elementos de forma vertical llamado *Column*. En la parte inferior de la pantalla, se encuentran varios iconos organizados en una fila, que permiten incorporar nuevas widgets al contenido de la lección ya que pulsando en uno de estos iconos, se añade un nuevo widget a la lista.

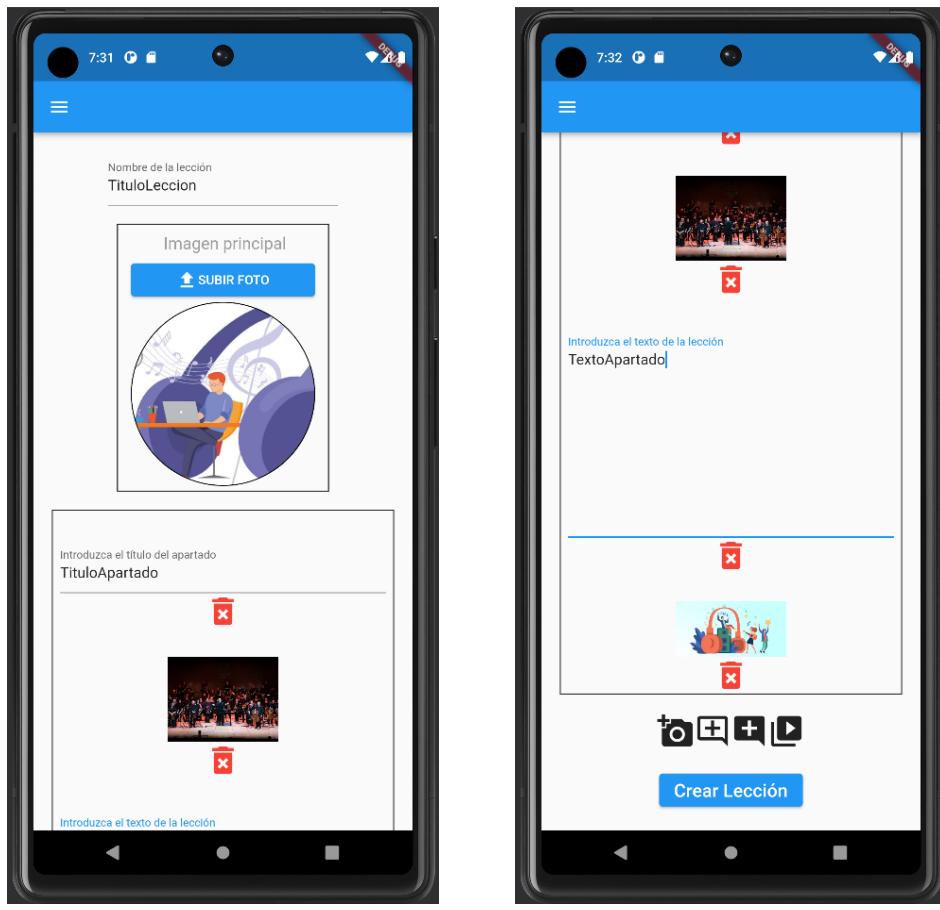
En las siguientes imágenes se puede ver la pantalla de creación de lecciones:



(a) Creación de lección vacía #1.

(b) Creación de lección vacía #2.

Figura 7.30: Pantalla de la creación de lecciones sin datos introducidos.



(a) Creación de lección rellena #1.

(b) Creación de lección rellena #2.

Figura 7.31: Pantalla de la creación de lecciones con datos introducidos.

Lógica La funcionalidad de crear lección ha necesitado desarrollar varias utilidades debido a su complejidad. La primera de ellas es la de añadir campos de texto a la pantalla de creación de lecciones. Para ello, se han creado varios widgets que contienen un campo de texto y un botón para eliminar dicho campo. Estos widgets se han llamado *SubirTexto*, *SubirTitulo* y *SubirVideo* y se añaden de forma dinámica a un vector ubicado en la pantalla de creación de lecciones al pulsar el botón correspondiente.

Para subir una imagen se ha hecho uso de la biblioteca *image_picker*, que permite al usuario seleccionar una imagen de su galería y devuelve la ruta de la imagen seleccionada. Una vez tenemos el archivo de la imagen, se manda una petición de tipo multipart con la imagen a la API para que la suba al servidor. El almacenamiento de la imagen utiliza la funcionalidad proporcionada por el paquete de NodeJS llamado *multer*. Este paquete permite

la subida de archivos al servidor y la creación de carpetas para almacenar dichos archivos. La ruta de la petición es *POST /api/lesson/upload-image*. La API devolverá un código de estado 200 si la operación se ha realizado correctamente. En caso contrario, se mostrará un mensaje de error.

El profesor también podrá eliminar widgets de la pantalla de creación de lecciones. Para ello, se ha implementado un botón en cada widget que permite al profesor eliminar dicho widget. Al pulsar este botón, se llama al método *borrar()*. En este se elimina el widget de la lista y se actualiza la pantalla de creación de lecciones al ser un widget de tipo *StatefulWidget*.

Una vez el usuario ha llenado el formulario de creación de lecciones, se ha implementado un botón que permite al profesor guardar la lección. Al pulsar este botón, se envía una petición POST a la API con los datos de la lección. La ruta de la petición es *POST /api/lesson/create-lesson*. La API devolverá un código de estado 200 si la operación se ha realizado correctamente y se actualizará la lista de lecciones del provider. En caso contrario, se mostrará un mensaje de error.

7.6.5. Editar lección

Desarrollado en el Sprint 8 Los profesores podrán editar la información y el contenido de las lecciones. Para ello, se ha implementado una pantalla de edición de lecciones que permite al profesor modificar el título, la imagen y todo el contenido.

Vista La vista de esta pantalla es muy similar a la de la creación de lecciones. La diferencia es que los campos de texto contienen la información de la lección que se está editando.

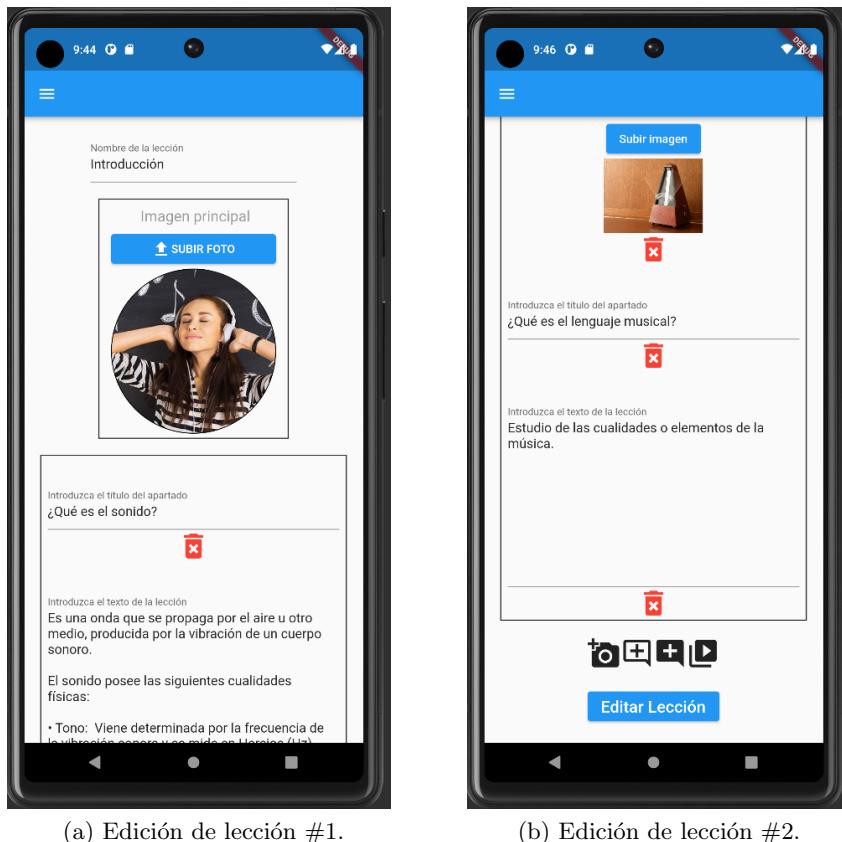


Figura 7.32: Pantalla de la edición de lecciones existentes por parte del profesor.

Lógica En cuanto a la lógica, muchas de las funcionalidades de esta pantalla son iguales a las de la pantalla de creación de lecciones.

La única diferencia es que, al pulsar el botón de guardar, se envía una petición PUT a la API con los datos de la lección. La ruta de la petición es *PUT /api/lesson/update-lesson*. La API devolverá un código de estado 200 si la operación se ha realizado correctamente y se actualizará la lista de lecciones del provider. En caso contrario, se mostrará un mensaje de error.

7.6.6. Lista de preguntas

Desarrollado en el Sprint 8

Vista La lista de preguntas contiene todas las preguntas existentes en el sistema. Se ha utilizado un distribuidor de columnas /textit(Column) para mostrar las preguntas verticalmente. Cada pregunta tiene un distribuidor de elementos horizontal /textit(Row) que contiene a la izquierda la imagen de la pregunta y a la derecha el texto de la cuestión, el tipo de pregunta y su identificador. Además de esto, cada pregunta tiene un icono de tipo Icon que permite al usuario modificar la pregunta.



Figura 7.33: Pantalla de lista de preguntas del profesor para la gestión de estas.

7.6.7. Borrar pregunta

Desarrollado en el Sprint 9

El profesor podrá borrar una pregunta de la lista al pulsar el ícono de la papelera mostrado en la vista del apartado anterior. Una vez pulsado, se mostrará un diálogo de confirmación para que el profesor confirme que desea borrar la pregunta.

Vista La pantalla del borrado de preguntas utiliza el widget *AlertDialog*, que posee un título, un texto descriptivo y dos botones: uno para cancelar el borrado y otro para confirmarlo. En la siguiente imagen se puede ver la pantalla de borrado de preguntas:



Figura 7.34: Pantalla del borrado de una pregunta.

Lógica Una vez el botón de *Confirmar* es pulsado, se muestra un diálogo de confirmación para que el profesor confirme que desea borrar la pregunta. Este borrado se realiza llamando a la función `borrarPregunta()` del provider *PreguntasProfesor*. En dicha función se borrará la pregunta de la lista del provider y se actualizará la base de datos enviando una petición `DELETE` a la API con el id de la pregunta a borrar. La ruta de la petición es `DELETE /api/question/delete-question`. La API devolverá un código de estado 200 si la operación se ha realizado correctamente y se actualizará la lista de preguntas del provider. En caso contrario, se mostrará un mensaje de error.

7.6.8. Crear Pregunta

Desarrollado en el Sprint 9

Los profesores podrán crear preguntas para que los usuarios puedan contestarlas en los tests. Para ello, se ha implementado una pantalla de creación de preguntas que permite al profesor elegir el tipo de pregunta y modificar toda la información relacionada con esta (imagen, opciones, respuestas correctas, etc). Una vez creada, la lección se añadirá a la lista de lecciones del profesor.

Vista La vista de esta pantalla consiste en un formulario de creación de preguntas. Este formulario consta de:

- Un desplegable para elegir el tipo de pregunta (entre *única*, *múltiple*, *micrófono* y *texto*).
- Un botón para subir la imagen de la pregunta.
- Un desplegable para elegir la lección a la que pertenece la pregunta.
- Un campo de texto para el enunciado/cuestión de la pregunta.
- Si es una pregunta de tipo 'única' o 'múltiple', un número variable de campos de texto para las opciones de la pregunta.
- Si es una pregunta de tipo 'texto', un campo de texto para la respuesta correcta.
- Si es una pregunta de tipo 'micrófono', un número variable de desplegables para elegir las notas musicales correctas.

Todo esto está distribuido mediante el widget que organiza los elementos de forma vertical llamado *Column*. Dentro de este, se encuentran los widgets del formulario, que serán desde widgets de desplegables (*DropdownButton*) hasta widgets de campos de texto (*TextField*), pasando por casillas de verificación (*Checkbox*). En las preguntas de tipo 'única', 'múltiple' y 'micrófono', en la parte inferior de la pantalla se encuentra un ícono que permite al profesor añadir más opciones a la pregunta.

En las siguientes imágenes se puede ver la pantalla de creación de preguntas de todos los tipos con los campos de texto vacíos y rellenos.

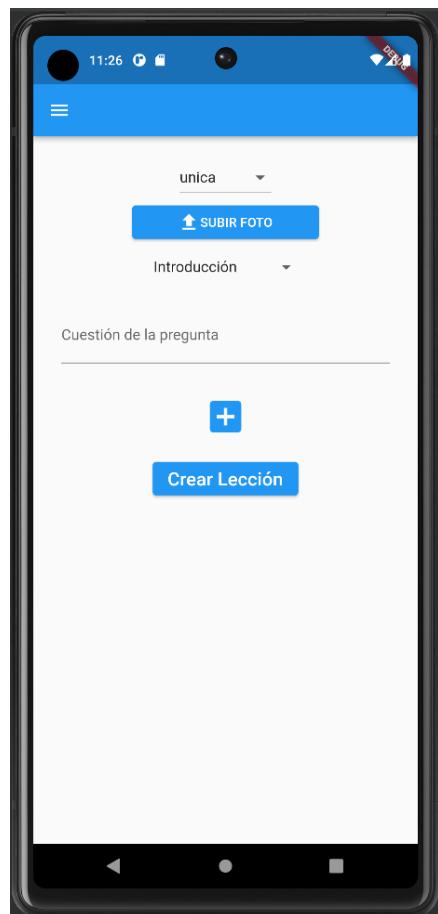
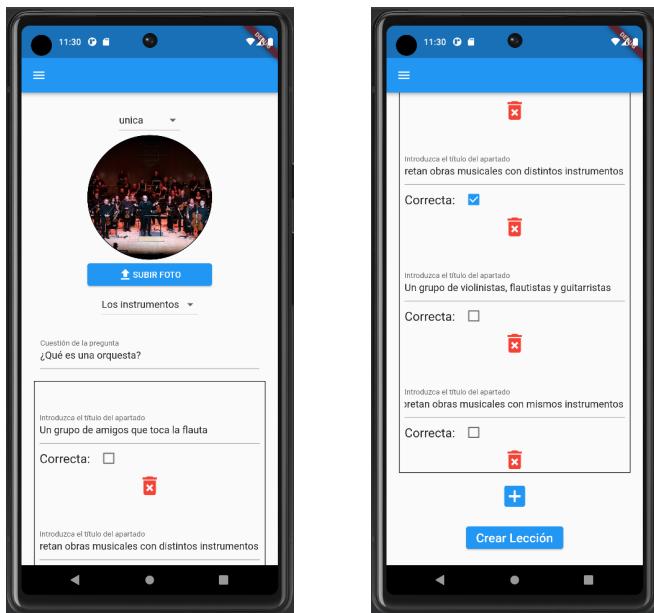


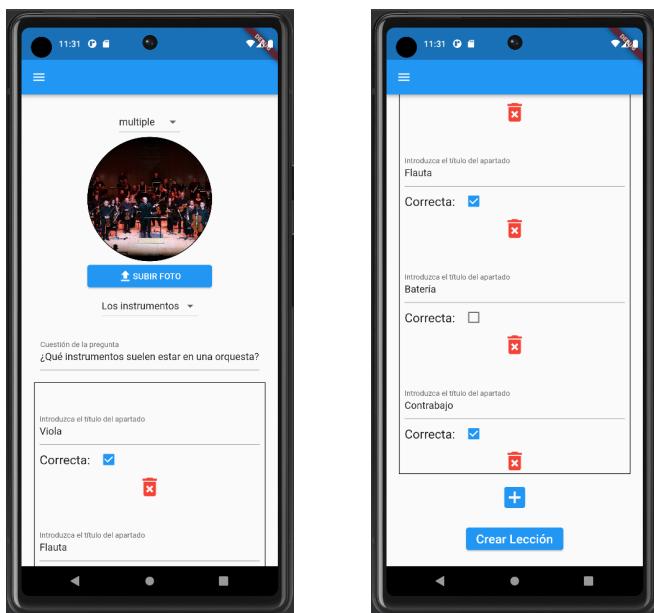
Figura 7.35: Pantalla de creación de pregunta de tipo 'única' vacía. La pregunta de tipo múltiple y de tipo micrófono son iguales.



(a) Creación de pregunta 'única' rellena #1.

(b) Creación de pregunta 'única' rellena #2.

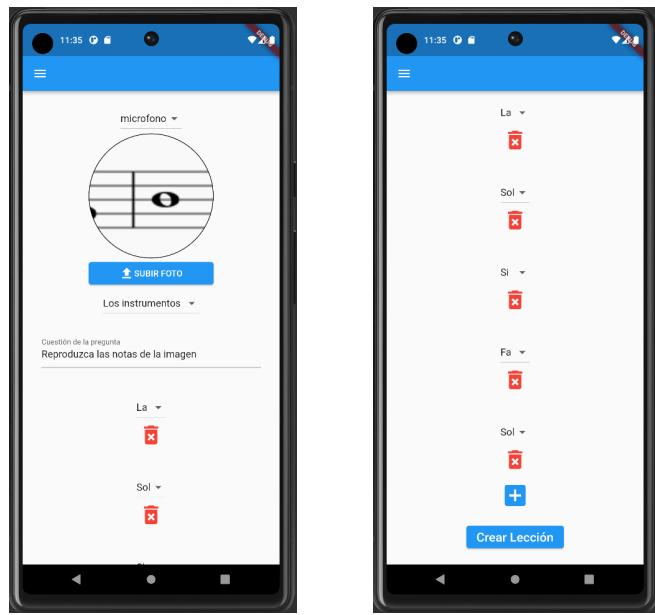
Figura 7.36: Pantalla de creación de pregunta de tipo 'única' con datos introducidos.



(a) Creación de pregunta 'múltiple' rellena #1.

(b) Creación de pregunta 'múltiple' rellena #2.

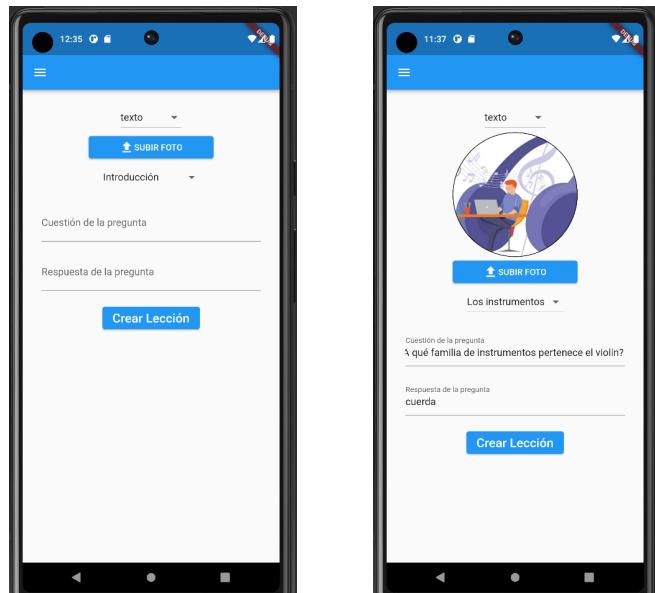
Figura 7.37: Pantalla de creación de pregunta de tipo 'múltiple' con datos introducidos.



(a) Creación de pregunta 'micrófono' rellena #1.

(b) Creación de pregunta 'micrófono' rellena #2.

Figura 7.38: Pantalla de creación de pregunta de tipo 'micrófono' con datos introducidos.



(a) Creación de pregunta 'texto' vacía.

(b) Creación de pregunta 'texto' rellena.

Figura 7.39: Pantalla de creación de pregunta de tipo 'texto'.

Lógica La funcionalidad de crear preguntas se ha implementado utilizando el Provider de preguntas anteriormente mencionado. De esta forma se ha podido comunicar varias pantallas entre sí y compartir datos entre ellas.

Para empezar, se ha creado un nuevo widget llamado *SubirOpcion*, que actúa de forma muy parecida a los widgets mencionados en la creación de lecciones (*SubirImagen*, *SubirTitulo*, *SubirTexto...*). Este widget se encargará de encapsular toda la vista y la lógica relacionada con una opción concreta. Además he visto necesaria la creación de un nuevo Provider para las opciones de las preguntas. Este Provider contiene un vector de opciones y un vector de respuestas correctas. El vector de opciones contiene los textos de las opciones de la pregunta y el vector de respuestas correctas contiene los índices de las opciones correctas. Estos dos vectores se actualizan cada vez que el profesor añade o elimina una opción de la pregunta. De esta forma, la comunicación entre la pantalla y la clase "*SubirOpcion*" ha sido más sencilla tanto para borrar una opción como para desmarcar una opción cuando se marca otra distinta. El widget de *SubirOpcion* se añade de forma dinámica a un vector de widgets ubicado en la pantalla de creación de preguntas al pulsar el botón correspondiente.

Para subir una imagen se ha hecho uso de nuevo de la biblioteca *image-picker*, que detallé en la funcionalidad de crear lecciones. El profesor podrá eliminar widgets mediante el botón apropiado en cada uno de ellos. Dicho botón llama al método *borrar()* y éste elimina el widget de la listas y actualiza la pantalla de creación de preguntas.

El botón de guardar la lección envía los datos de las preguntas al servidor mediante una petición POST a la ruta `/api/question/create-question`. La API devolverá un código de estado 200 si la operación se ha realizado correctamente y se actualizará la lista de preguntas del provider. En caso contrario, se mostrará un mensaje de error.

7.6.9. Editar Pregunta

Desarrollado en el Sprint 9 Los profesores podrán editar la información y el contenido de las preguntas. Para ello, se ha implementado una pantalla de edición de preguntas que permite al profesor modificar la cuestión, la imagen y todos los demás datos.

Vista La vista de esta pantalla es muy similar a la de la creación de las preguntas. La diferencia es que los widgets disponen de la información de la pregunta que se está editando.

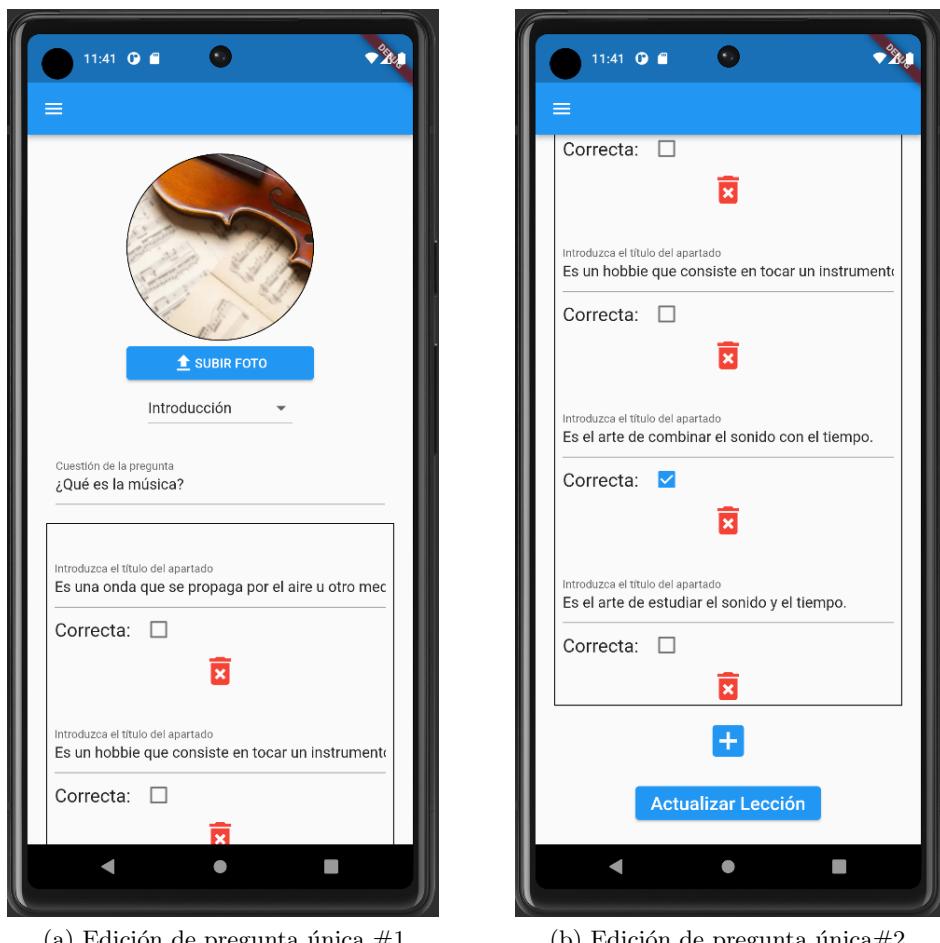


Figura 7.40: Pantalla de edición de pregunta de tipo 'única'.

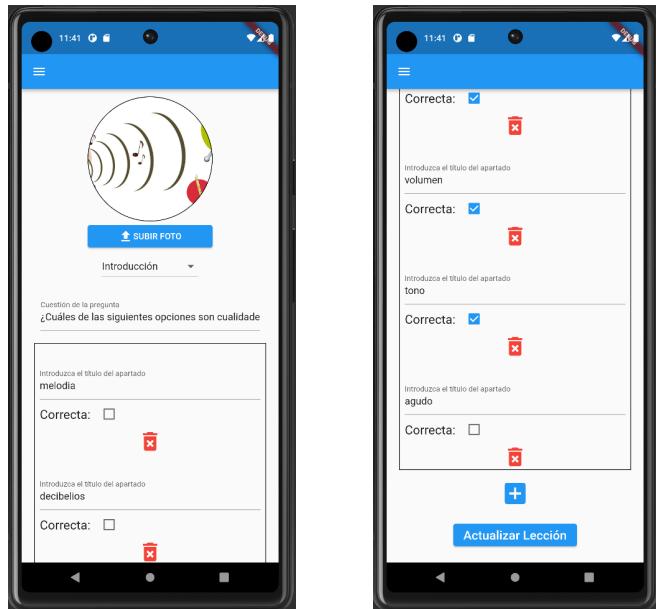


Figura 7.41: Pantalla de edición de pregunta de tipo 'múltiple'.



Figura 7.42: Pantalla de edición de pregunta de tipo 'micrófono'.



Figura 7.43: Pantalla de edición de pregunta de tipo 'texto'.

Lógica En cuanto a la lógica, muchas de las funcionalidades de esta pantalla son iguales a las de la pantalla de creación de preguntas. Entre las diferencias notamos que al entrar se deben inicializar los valores de cada uno de los campos con aquellos obtenidos de la base de datos y que, al pulsar el botón de guardar, se envía una petición PUT a la API con los datos de la pregunta. La ruta de la petición es *PUT /api/question/update-question*. La API devolverá un código de estado 200 si la operación se ha realizado correctamente y se actualizará la lista de preguntas del provider. En caso contrario, se mostrará un mensaje de error.

7.6.10. Lista de usuarios

Desarrollado en el Sprint 9

Vista La lista de usuarios contiene todos los usuarios registrados en el sistema. Se ha utilizado un distribuidor de columnas /textit(Column) para mostrar los usuarios de forma vertical. Cada usuario tiene un distribuidor de elementos horizontal /textit(Row) que contiene a la izquierda la foto del usuario y a la derecha su nombre en negrita y el correo en un color grisáceo.

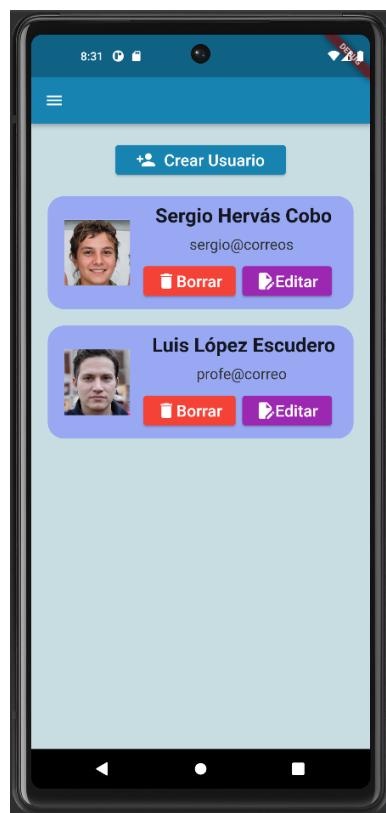


Figura 7.44: Pantalla de lista de usuarios del administrador para la gestión de estos.

7.6.11. Borrar usuario

Desarrollado en el Sprint 10

El administrador podrá borrar un usuario de la lista al pulsar en el botón de "Borrar" mostrado en la Figura 7.44. Una vez pulsado, se mostrará un diálogo para que el administrador confirme que desea borrar el usuario.

Vista La pantalla del borrado de usuarios utiliza el widget *AlertDialog*, que posee un título, un texto descriptivo y dos botones: uno para cancelar el borrado y otro para confirmarlo. En la siguiente imagen se puede ver la pantalla de borrado de usuarios:



Figura 7.45: Pantalla de borrado de un usuario.

Lógica Una vez pulsado el botón de *Borrar*, se muestra un diálogo para que el administrador confirme que desea borrar el usuario. Este borrado se hará llamando a la función `borrarUsuario()` del provider *Usuarios*. En dicha función se borrará el usuario de la lista del provider y se actualizará la base de datos enviando una petición `DELETE` a la API con el id del usuario a borrar. La ruta de la petición es `DELETE /api/user/delete-user`. La API devolverá un código de estado 200 si la operación se ha realizado correctamente y se actualizará la lista de usuarios del provider. En caso

contrario, se mostrará un mensaje de error.

7.6.12. Crear usuario

Desarrollado en el Sprint 10

Los administradores podrán crear usuarios en el sistema. Para ello, se ha implementado una pantalla de creación de usuarios que permite introducir los datos del usuario: nombre, el correo, la contraseña, la foto... Una vez creado, el usuario se añadirá a la lista de usuarios del profesor.

Vista La vista de esta pantalla consiste en un formulario de creación de usuarios. Este formulario consta de:

- Un botón para subir la imagen del usuario.
- Un desplegable para elegir el rol del usuario.
- Campos de texto para introducir el nombre, los apellidos, el nombre de usuario, el correo y la contraseña.

Todo esto está distribuido mediante el widget que organiza los elementos de forma vertical llamado *Column*. Dentro de este, se encuentran los widgets del formulario, que serán tanto widgets de desplegables (*DropdownButton*) como widgets de campos de texto (*TextField*).

En las siguientes imágenes se puede ver la pantalla de creación de usuarios de todos los tipos con los campos de texto vacíos y llenos.

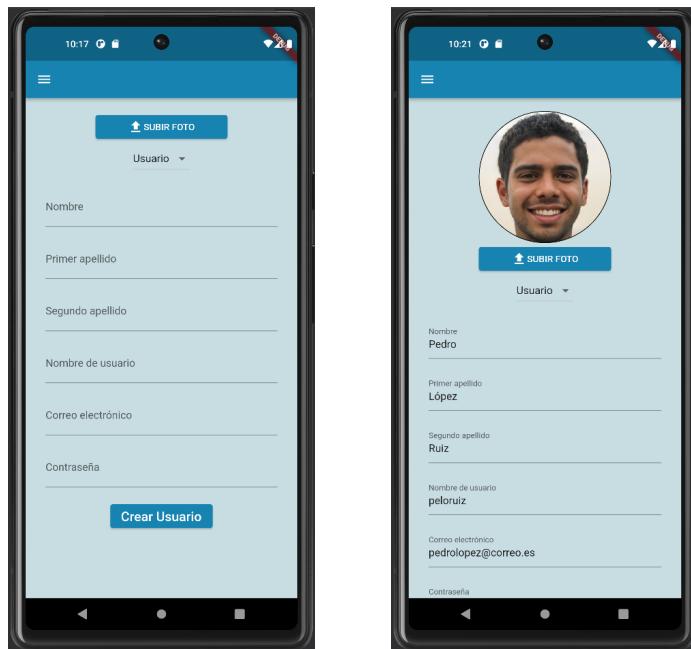


Figura 7.46: Pantallas de creación de usuarios.

Lógica La funcionalidad de crear usuarios se ha implementado utilizando un nuevo método de creación de usuarios del Provider llamado usuarios. De esta forma se ha podido comunicar varias pantallas entre sí y añadir el usuario creado a la lista de usuarios del administrador.

Para subir una imagen se ha hecho uso de nuevo de la biblioteca *image_picker*, que detallé en la funcionalidad de crear lecciones.

El botón de guardar el usuario envía los datos de los usuarios al servidor mediante una petición POST a la ruta /api/user/registro. La API devolverá un código de estado 200 si la operación se ha realizado correctamente y se actualizará la lista de usuarios del provider. En caso contrario, se mostrará un mensaje de error.

7.6.13. Editar usuario

Desarrollado en el Sprint 10

Los administradores podrán editar la información de los usuarios. Para ello, se ha implementado una pantalla de edición de usuarios que permite al profesor modificar el nombre, los apellidos, la foto y todos los demás datos.

Vista La vista de esta pantalla es muy similar a la de la creación de los usuarios. La diferencia es que los widgets disponen de la información del usuario que se está editando.

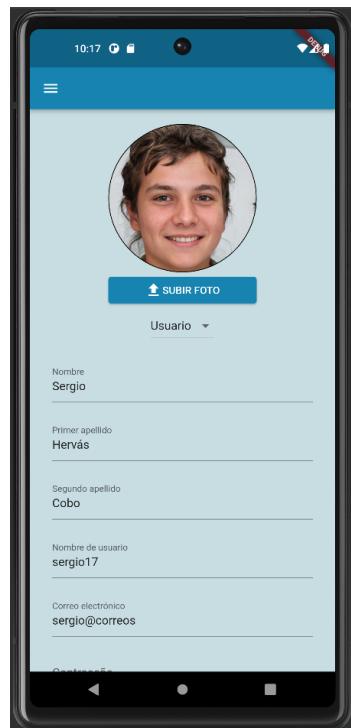


Figura 7.47: Pantalla de edición de usuario existente.

Lógica Los datos del usuario se inician con la información almacenada en la base de datos. Una vez confirmadas las modificaciones, se envía una petición PUT a la API con los datos actualizados del usuario a la ruta `/api/user/update-user`. La API devolverá un código de estado 200 si la operación se ha realizado correctamente y se actualizará la lista de usuarios del provider. En caso contrario, se mostrará un mensaje de error.

7.6.14. Lista de logros

Desarrollado en el Sprint 10

Vista La lista de logros contiene todos los logros creados en el sistema. Se ha utilizado un distribuidor de columnas /textit(Column) para mostrar los logros de forma vertical. Cada logro tiene un distribuidor de elementos horizontal /textit(Row) que contiene a la izquierda la imagen del logro y a la derecha el nombre en negrita.

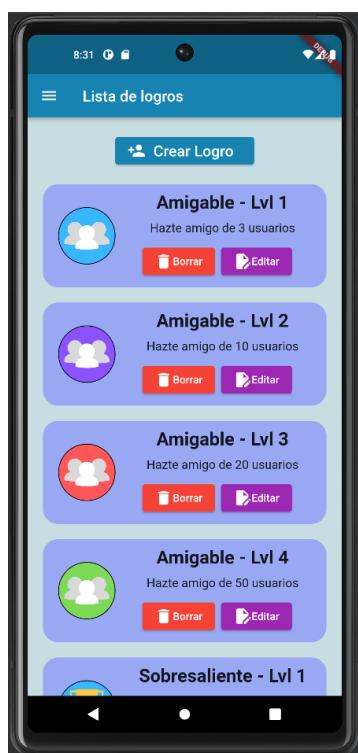


Figura 7.48: Pantalla de lista de logros del administrador.

7.6.15. Borrar logro

Desarrollado en el Sprint 10 El administrador podrá borrar un logro de la lista al pulsar el botón Borrarrmostrado en la vista de la figura 7.48. Una vez pulsado, se mostrará un diálogo de confirmación para que el administrador confirme que desea borrar el logro.

Vista La pantalla del borrado de logros utiliza el widget *AlertDialog*, que posee un título, un texto descriptivo y dos botones: uno para cancelar el borrado y otro para confirmarlo. En la siguiente imagen se puede ver la pantalla de borrado de logros:



Figura 7.49: Pantalla del borrado de un logro.

Lógica Una vez pulsado el botón de *Borrar*, se muestra un diálogo para que el administrador confirme que desea borrar el logro. Este borrado se realiza llamando a la función `borrarLogro()` del provider *Logros*. En dicha función se borrará el logro de la lista del provider y se actualizará la base de datos enviando una petición *DELETE* a la ruta `/api/achievement/delete-achievement`. La API devolverá un código de estado 200 si la operación se ha realizado correctamente y se actualizará la lista de logros del provider. En caso contrario, se mostrará un mensaje de error.

7.6.16. Crear logro

Desarrollado en el Sprint 10

Los administradores podrán crear logros en el sistema. Para ello, se ha implementado una pantalla de creación de logros que permite introducir el nombre, la imagen, la descripción, el tipo y la condición del logro. Una vez creado, se añadirá a la lista de logros de la pantalla del administrador.

Vista La vista de esta pantalla consiste en un formulario de creación de logros. Este formulario consta de:

- Un botón para subir la imagen del logro.
- Un campo de texto para introducir el nombre del logro.
- Un campo de texto para introducir la descripción del logro.
- Un desplegable para elegir el tipo de logro (entre *amigos*, *lección*, *lecciones*, *tests*, *preguntas*, *preguntasunica*, *preguntasmultiple*, *preguntas-microfono* y *preguntastexto*).
- Si el logro no es de tipo lección; un campo de texto para introducir la condición dependiente del tipo.
- Si el logro es de tipo lección; un desplegable para elegir la lección a la que pertenece el logro.

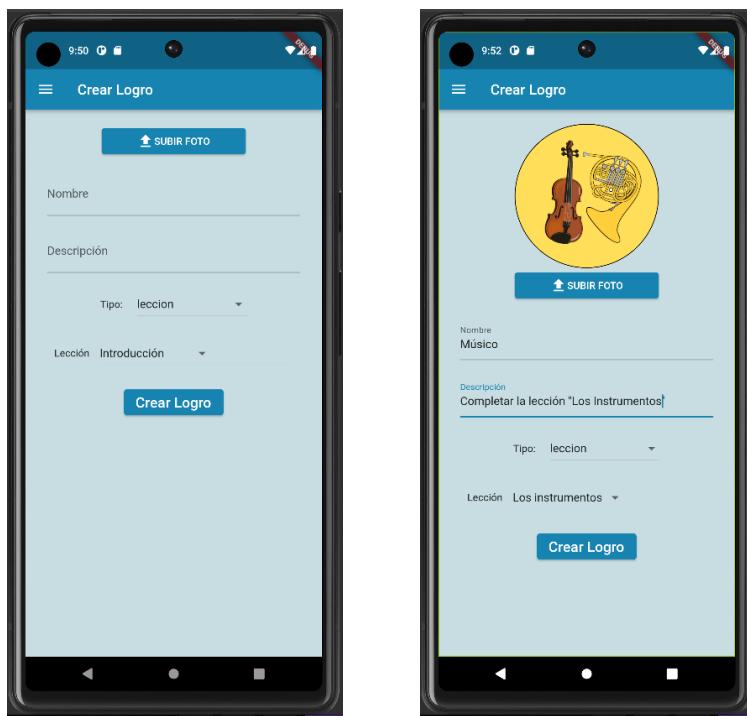


Figura 7.50: Pantallas de creación de logros.

Todo esto está distribuido mediante el widget que organiza los elementos de forma vertical llamado *Column*.

En las imágenes anteriores se puede ver la pantalla de creación de logros con los campos de texto vacíos y llenos.

Lógica Se ha implementado utilizando un nuevo método de creación de logros del Provider llamado logros. De esta forma se ha podido comunicar varias pantallas entre sí y añadir el logro creado a la lista de logros del administrador.

Para subir una imagen se ha hecho uso de nuevo de la biblioteca *image_picker*, que detallé en la funcionalidad de crear lecciones.

El botón de guardar el logro envía los datos de los logros al servidor mediante una petición POST a la ruta /api/achievement/create-achievement. La API devolverá un código de estado 200 si la operación se ha realizado correctamente y se actualizará la lista de logros del provider. En caso contrario, se mostrará un mensaje de error.

7.6.17. Editar logro

Desarrollado en el Sprint 10

Los administradores podrán editar la información de los logros. Para ello, se ha implementado una pantalla de edición de logros que permite al administrador modificar el nombre, la imagen, la descripción, el tipo y la condición del logro.

Vista La vista de esta pantalla es muy similar a la de la creación de los logros. La diferencia es que los widgets disponen de la información del logro que se está editando.



Figura 7.51: Pantalla de edición de logro existente.

Lógica Los datos del logro se inician con la información almacenada en la base de datos. Una vez confirmadas las modificaciones, se envía una petición PUT a la API con los datos actualizados del logro a la ruta `/api/achievement/update-achievement`. La API devolverá un código de estado 200 si la operación se ha realizado correctamente y se actualizará la lista de logros del provider. En caso contrario, se mostrará un mensaje de error.

Capítulo 8

Pruebas

En este capítulo se describen las pruebas realizadas al sistema, tanto de forma individual como de forma integrada. Se describen los casos de prueba, los resultados obtenidos y las conclusiones extraídas de los mismos. El objetivo de las pruebas es comprobar que el sistema cumple con los requisitos establecidos en el capítulo (Figura 3) y que funcione correctamente.

8.1. Paquetes utilizados

Para realizar las pruebas se han utilizado los siguientes paquetes:

- **flutter_test**: paquete que contiene las clases necesarias para realizar las pruebas de unidad.
- **nock**: paquete que permite simular las peticiones HTTP realizadas por la aplicación y devolver una respuesta simulada
- **flutter_driver**: paquete que contiene las clases necesarias para realizar las pruebas de integración.

8.2. Pruebas de unidad

Las pruebas de unidad son aquellas que se realizan sobre los componentes más pequeños del sistema, como pueden ser las funciones o los métodos. Estas pruebas se realizan de forma individual, aislando el componente a probar de los demás componentes del sistema. De esta forma nos aseguramos de que el componente funciona correctamente y que no depende de otros componentes para su correcto funcionamiento.

8.2.1. Lecciones

A continuación se describen las pruebas realizadas sobre las lecciones:

- **Obtención y guardado:** Las lecciones deben obtenerse y guardarse en el vector del provider (método fetchAndSetLecciones). *Tras la ejecución del método el provider debe contener una lista no vacía de lecciones.*
- **Obtención del índice mediante el id:** Dado un id de una lección, se debe obtener el índice de la lección en el vector del provider (método getIndice). *El método no debe devolver el valor '-1'.*
- **Creación de una lección:** Se debe crear una lección con los datos correctos (método crearLeccion). *Tras ejecutar el método, el vector de lecciones del provider debe tener una longitud mayor que antes de ejecutarlo.*
- **Modificación de una lección:** Se debe modificar una lección con los datos correctos (método modificarLeccion). *Tras ejecutar el método, los datos antiguos y nuevos de la lección deben ser distintos.*
- **Eliminación de una lección:** Se debe eliminar una lección coorectamente (método eliminarLeccion). *Tras ejecutar el método, el vector de lecciones del provider debe tener una longitud menor que antes de ejecutarlo.*

Test Results		4 sec 597 ms
✓	✓ lecciones.dart	4 sec 597 ms
✓	✓ Las lecciones deben obtenerse y guardarse en el vector del provider	771 ms
✓	✓ El indice de una lección en el vector de lecciones es el correcto	754 ms
✓	✓ El número de tests aprobados de una lección es correcto	712 ms
✓	✓ Se crea una lección correctamente	805 ms
✓	✓ Se modifica una lección correctamente	772 ms
✓	✓ Se borra una lección correctamente	783 ms

Figura 8.1: Resulados de las pruebas de unidad de las lecciones

8.2.2. Preguntas

Las pruebas realizadas sobre las preguntas son las siguientes:

- **Obtención y guardado de preguntas del profesor:** Las preguntas de la vista del profesor deben obtenerse y guardarse en el vector del provider PreguntasProfesor (método fetchAndSetPreguntas). *Después*

de ejecutar el método, el provider debe contener una lista no vacía de preguntas.

- **Obtención del índice mediante el id:** Dado un id de una pregunta, se debe obtener el índice de la pregunta en el vector del provider (método getIndice). *El método no debe devolver el valor '-1'.*
- **Creación de una pregunta:** Se debe crear una pregunta con los datos correctos (método crearPregunta). *Tras ejecutar el método, el vector de preguntas del provider debe tener una longitud mayor que antes de ejecutarlo*
- **Modificación de una pregunta:** Se debe modificar una pregunta con los datos correctos (método modificarPregunta). *Tras la ejecución del método, los datos antiguos y nuevos de la pregunta deben ser distintos.*
- **Eliminación de una pregunta:** Se debe eliminar una pregunta correctamente (método eliminarPregunta). *Tras ejecutar el método, el provider no debe contener la pregunta.*
- **Obtención y guardado de preguntas:** Las preguntas deben obtenerse y guardarse en el vector del provider Preguntas (método fetchAndSetPreguntas). *Después de ejecutar el método, el provider debe contener una lista no vacía de preguntas.*
- **Asignar opción:** Se debe asignar una opción a una pregunta correctamente cuando se crea o modifica una pregunta (método setValor). *Tras ejecutar el método, las opciones de la pregunta deben coincidir con las opciones asignadas.*
- **Asignar respuesta:** Se debe asignar una respuesta a una pregunta correctamente cuando el usuario responde (método setRespuestas). *Después de la ejecución, la respuesta de la pregunta debe coincidir con la respuesta asignada.*
- **Asignar pulsado:** Se debe asignar el valor de la variable "pulsado" correctamente cuando el usuario pulsa una opción (método setPulsado). *Después de la ejecución, los valores del array "pulsado" deben coincidir con los asignados.*

Test Results		498 ms
✓	preguntas.dart	498 ms
✓	Las preguntas del profesor deben obtenerse y guardarse en el vector del provider PreguntasProfesor	76 ms
✓	El índice de una pregunta en el vector de preguntas es el correcto	48 ms
✓	Se crea una pregunta correctamente	98 ms
✓	Se modifica una pregunta correctamente	92 ms
✓	Se borra una pregunta correctamente	74 ms
✓	Las preguntas deben obtenerse y guardarse en el vector del provider Preguntas	36 ms
✓	Las opciones de las preguntas asignan el valor correctamente	4 ms
✓	Las respuestas de las preguntas asignan el valor correctamente	37 ms
✓	El estado pulsado de las respuestas se asigna correctamente	33 ms

Figura 8.2: Resultados de las pruebas de unidad de las preguntas

8.2.3. Usuarios

Las pruebas realizadas sobre los usuarios son las siguientes:

- **Obtención y guardado:** Los usuarios deben obtenerse y guardarse en el vector del provider (método `fetchAndSetUsuarios`). *Tras la ejecución del método el provider debe contener una lista no vacía de usuarios.*
- **Obtención del índice mediante el id:** Dado un id de un usuario, se debe obtener el índice del usuario en el vector del provider (método `getIndice`). *El método no debe devolver el valor '-1'.*
- **Creación de un usuario:** Se debe crear un usuario con los datos correctos (método `crearUsuario`). *Tras ejecutar el método, el vector de usuarios del provider debe tener una longitud mayor que antes de ejecutarlo.*
- **Modificación de un usuario:** Se debe modificar un usuario con los datos correctos (método `modificarUsuario`). *Tras ejecutar el método, los datos antiguos y nuevos del usuario deben ser distintos.*
- **Eliminación de un usuario:** Se debe eliminar un usuario correctamente (método `eliminarUsuario`). *Tras ejecutar el método, el vector de usuarios del provider debe tener una longitud menor que antes de ejecutarlo.*

Test Results		414 ms
✓	✓ usuarios.dart	414 ms
✓	Los usuarios deben obtenerse y guardarse en el vector del provider	73 ms
✓	Se crea una lección correctamente	148 ms
✓	Se modifica una lección correctamente	122 ms
✓	Se borra una lección correctamente	71 ms

Figura 8.3: Resulados de las pruebas de unidad de los usuarios

8.2.4. Logros

Las pruebas realizadas sobre los logros son las siguientes:

- **Obtención y guardado:** Los logros deben obtenerse y guardarse en el vector del provider (método fetchAndSetLogros). *Tras la ejecución del método el provider debe contener una lista no vacía de logros.*
- **Obtención del índice mediante el id:** Dado un id de un logro, se debe obtener el índice del logro en el vector del provider (método getIndice). *El método no debe devolver el valor '1'.*
- **Creación de un logro:** Se debe crear un logro con los datos correctos (método crearLogro). *Tras ejecutar el método, el vector de logros del provider debe tener una longitud mayor que antes de ejecutarlo.*
- **Modificación de un logro:** Se debe modificar un logro con los datos correctos (método modificarLogro). *Tras ejecutar el método, los datos antiguos y nuevos del logro deben ser distintos.*
- **Eliminación de un logro:** Se debe eliminar un logro coorectamente (método borrarLogro). *Tras ejecutar el método, el vector de logros del provider debe tener una longitud menor que antes de ejecutarlo.*

Test Results		360 ms
✓	✓ logros.dart	360 ms
✓	Los logros deben obtenerse y guardarse en el vector del provider	67 ms
✓	El indice de una pregunta en el vector de preguntas es el correcto	42 ms
✓	Se crea un logro correctamente	103 ms
✓	Se modifica un logro correctamente	73 ms
✓	Se borra un logro correctamente	75 ms

Figura 8.4: Resulados de las pruebas de unidad de los logros

8.3. Pruebas de controlador o widget

Las pruebas de controlador o widget son aquellas que se realizan sobre los controladores o widgets de la aplicación (Figura 8.7). Estas pruebas se realizan de forma individual, aislando el controlador o widget a probar de los demás componentes del sistema. A continuación se muestran las pruebas realizadas sobre algunos widgets de la aplicación.

8.3.1. Botón de inicio de sesión

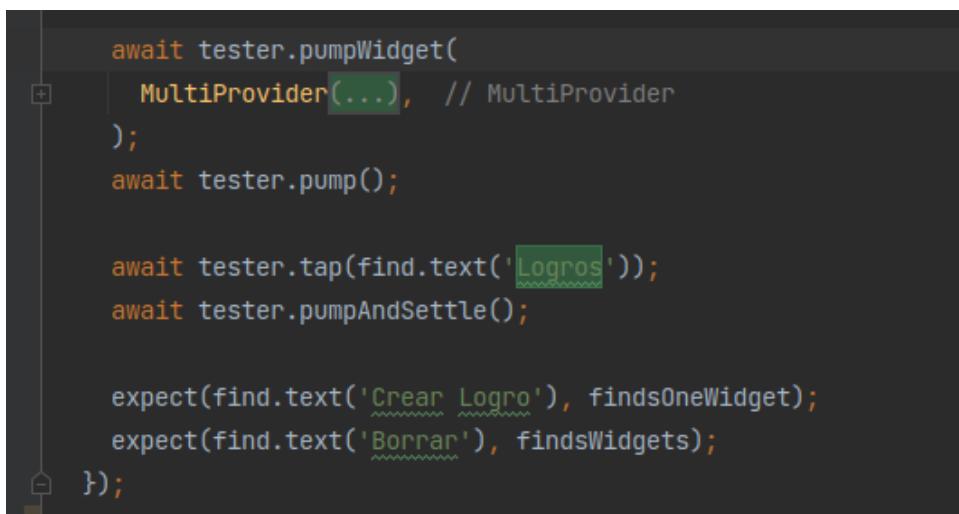
El botón de inicio de sesión se encuentra en la pantalla de inicio de la aplicación. Se ha realizado una prueba sobre el botón, comprobando que al pulsar el botón se accede a la pantalla de lecciones (pantalla principal de la aplicación). El test verifica que al pulsar dicho botón aparece un texto perteneciente a la pantalla de lecciones, en este caso “Los instrumentos”.

8.3.2. Botón de acceso a lista de lecciones del profesor

El dashboard del usuario contiene un botón para acceder a las lecciones. Se ha realizado una prueba sobre el botón, comprobando que al pulsar el botón se accede a la pantalla de lista de lecciones. El test verifica que al pulsar dicho botón se muestra un texto perteneciente a la pantalla de lecciones, en este caso “Crear Lección” y “Borrar”.

8.3.3. Botón de acceso a lista de logros del profesor

El dashboard del usuario contiene un botón para acceder a los logros. Se ha realizado una prueba sobre el botón, comprobando que al pulsar el botón se accede a la pantalla de lista de logros. El test verifica que al pulsar dicho botón se muestra un texto perteneciente a la pantalla de logros, en este caso “Crear Logro” y “Borrar” (Figura 8.5).



```
await tester.pumpWidget(  
    MultiProvider(...), // MultiProvider  
);  
await tester.pump();  
  
await tester.tap(find.text('Logros'));  
await tester.pumpAndSettle();  
  
expect(find.text('Crear Logro'), findsOneWidget);  
expect(find.text('Borrar'), findsWidgets);  
});
```

Figura 8.5: Código de la prueba de controlador o widget de acceso a lista de logros

8.3.4. Botón de acceso a lista de preguntas del profesor

El dashboard del usuario contiene un botón para acceder a las preguntas. Se ha realizado una prueba sobre el botón, comprobando que al pulsar el botón se accede a la pantalla de lista de preguntas. El test verifica que al pulsar dicho botón aparece un texto perteneciente a la pantalla de preguntas, en este caso “Crear Pregunta” y “Borrar”.

8.3.5. Botón de acceso a lista de usuarios

El dashboard del usuario contiene un botón para acceder a los usuarios. Se ha realizado una prueba sobre el botón, comprobando que al pulsar el botón se accede a la pantalla de lista de usuarios. El test verifica que al pulsar dicho botón se muestra un texto perteneciente a la pantalla de usuarios, en este caso “Crear Usuario” y “Borrar”.

8.3.6. Botón de acceso al perfil del usuario

El drawer contiene un botón para acceder al perfil del usuario. Se ha realizado una prueba sobre el botón, comprobando que al pulsar el botón se accede a la pantalla de perfil del usuario. El test verifica que al pulsar dicho botón aparece un texto perteneciente a la pantalla de perfil del usuario, en este caso “Mis logros”.

8.3.7. Botón de acceso a una lección

Al pulsar sobre una lección de la lista de lecciones, se accede a la pantalla de lección. Se ha realizado una prueba sobre el botón, comprobando que dicho acceso se realiza correctamente. El test verifica que al pulsar dicho botón aparece un texto perteneciente a la pantalla de lección, en este caso “Historial”.

8.3.8. Botón de acceso al inicio

El drawer contiene un botón para acceder a la pantalla de lecciones. Se ha realizado una prueba sobre el botón, comprobando que dicho acceso se realiza correctamente. El test verifica que al pulsar dicho botón se muestra un texto perteneciente a la pantalla de lecciones, en este caso “Los instrumentos” (Figura 8.6).



```
await tester.pumpWidget(
    MultiProvider(...), // MultiProvider
);
await tester.pump();

final ScaffoldState scafState = tester.firstState(find.byType(Scaffold));
scafState.openDrawer();
await tester.pumpAndSettle();

await tester.tap(find.text('Inicia'));
await tester.pumpAndSettle();

expect(find.text('Los instrumentos'), findsOneWidget);
});
```

Figura 8.6: Código de la prueba de controlador o widget de acceso a la pantalla de lecciones

8.3.9. Botón de acceso al dashboard

El drawer contiene un botón para acceder al dashboard. Se ha realizado una prueba sobre el botón, comprobando que dicho acceso se realiza correctamente. El test verifica si al pulsar dicho botón encuentra un texto perteneciente a la pantalla de dashboard, en este caso “Lecciones”.

Test Results		2 sec 86 ms
✓	widget_test.dart	2 sec 86 ms
✓	El botón de Iniciar sesión funciona correctamente	1 sec 65 ms
✓	El botón de Lecciones del dashboard funciona correctamente	146 ms
✓	La lección se carga correctamente	121 ms
✓	El botón de Logros del dashboard funciona correctamente	265 ms
✓	El botón de Preguntas del dashboard funciona correctamente	185 ms
✓	El botón de Usuarios del dashboard funciona correctamente	106 ms
✓	El botón de Inicio del Drawer funciona correctamente	115 ms
✓	El botón de Perfil del Drawer funciona correctamente	83 ms

Figura 8.7: Resultados de las pruebas de controlador o widget

8.4. Pruebas de sistema o de integración

Las pruebas de sistema o de integración son aquellas que se realizan sobre el sistema completo. Estas pruebas tratan de verificar el flujo de uso de una determinada funcionalidad de la aplicación. De esta forma nos aseguramos de que el sistema funciona de forma adecuada simulando varias acciones que podría realizar un usuario.

8.4.1. Creación de usuario

En este test se comprueba que se crea un usuario correctamente. Para ello se siguen los siguientes pasos:

1. Se comienza en la pantalla de inicio de sesión.
2. Se rellenan los campos de inicio de sesión con los datos del profesor.
3. Se pulsa el botón de inicio de sesión.
4. Se abre el drawer y se pulsa el botón de acceso a la dashboard.
5. Se pulsa el botón de acceso a la gestión de Usuarios.
6. Se pulsa el botón de crear usuario.
7. Se rellenan los campos de creación de usuario con los datos del alumno.
8. Se pulsa el botón de crear usuario.
9. Se comprueba que se ha creado el usuario.

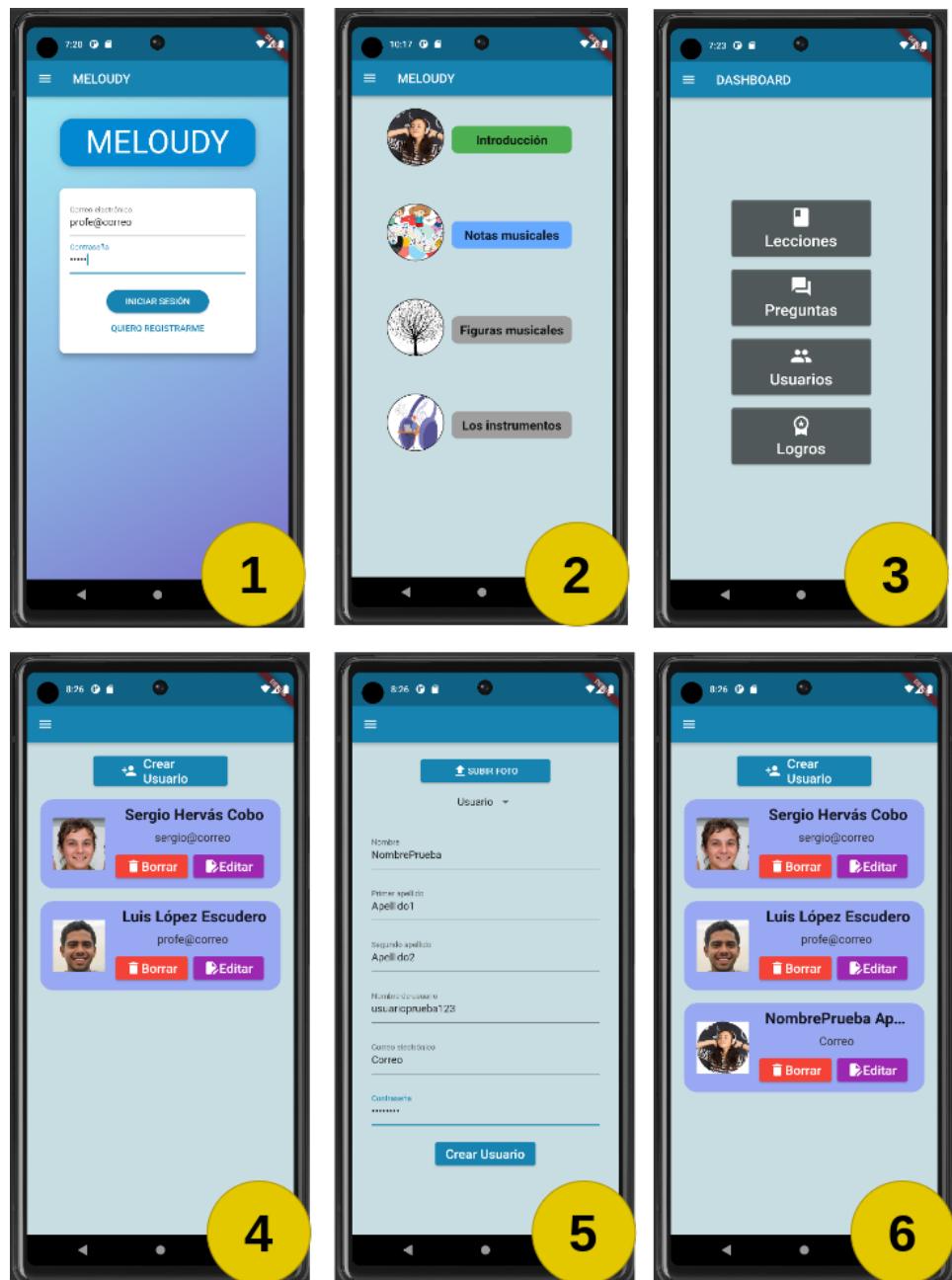


Figura 8.8: Pantallas por las que pasa la prueba de creación de usuario

8.4.2. Edición de datos del perfil

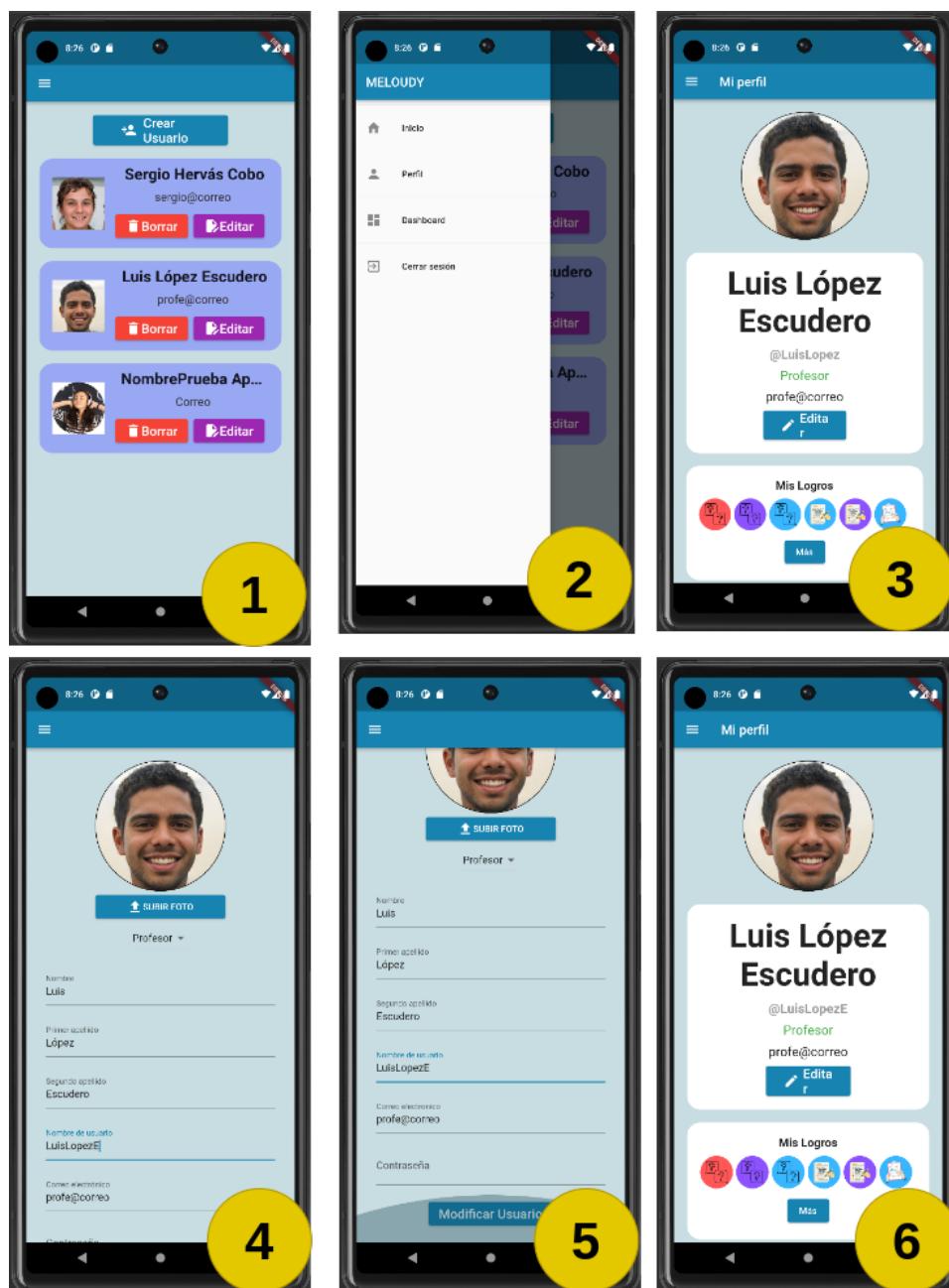


Figura 8.9: Pantallas por las que pasa la prueba de modificación de datos del perfil

Esta prueba verifica que se editen los datos del perfil de un usuario (Figura 8.9). Para ello se siguen los siguientes pasos:

1. Se comienza en la pantalla de lista de lecciones. (la sesión ya está iniciada del test anterior)
2. Se abre el drawer y se pulsa el botón de acceso al perfil del usuario.
3. Se pulsa el botón de editar perfil.
4. Se llenan los campos de edición de perfil con los nuevos datos del profesor.
5. Se pulsa el botón de editar perfil.
6. Se comprueba que se han editado los datos del perfil.

8.4.3. Realización de un test

Esta prueba verifica que se puede realizar un test (Figura 8.10) y contestar a las preguntas de este. Para ello se siguen los siguientes pasos:

1. Se comienza en la pantalla de lista de lecciones.
2. Se pulsa sobre la lección "Introducción" (por ejemplo).
3. Se pulsa el botón de "Hacer Test".
4. Se responde a las preguntas en función del tipo:
 - a) Si es de tipo "única", se pulsa sobre la segunda opción.
 - b) Si es de tipo "multiple", se pulsa sobre la segunda y la tercera opción.
 - c) Si es de tipo "texto", se escribe una respuesta cualquiera. En este caso se ha escrito "melodía".
5. Se pulsa el botón de ".Enviar".
6. Se comprueba que se ha realizado el test.

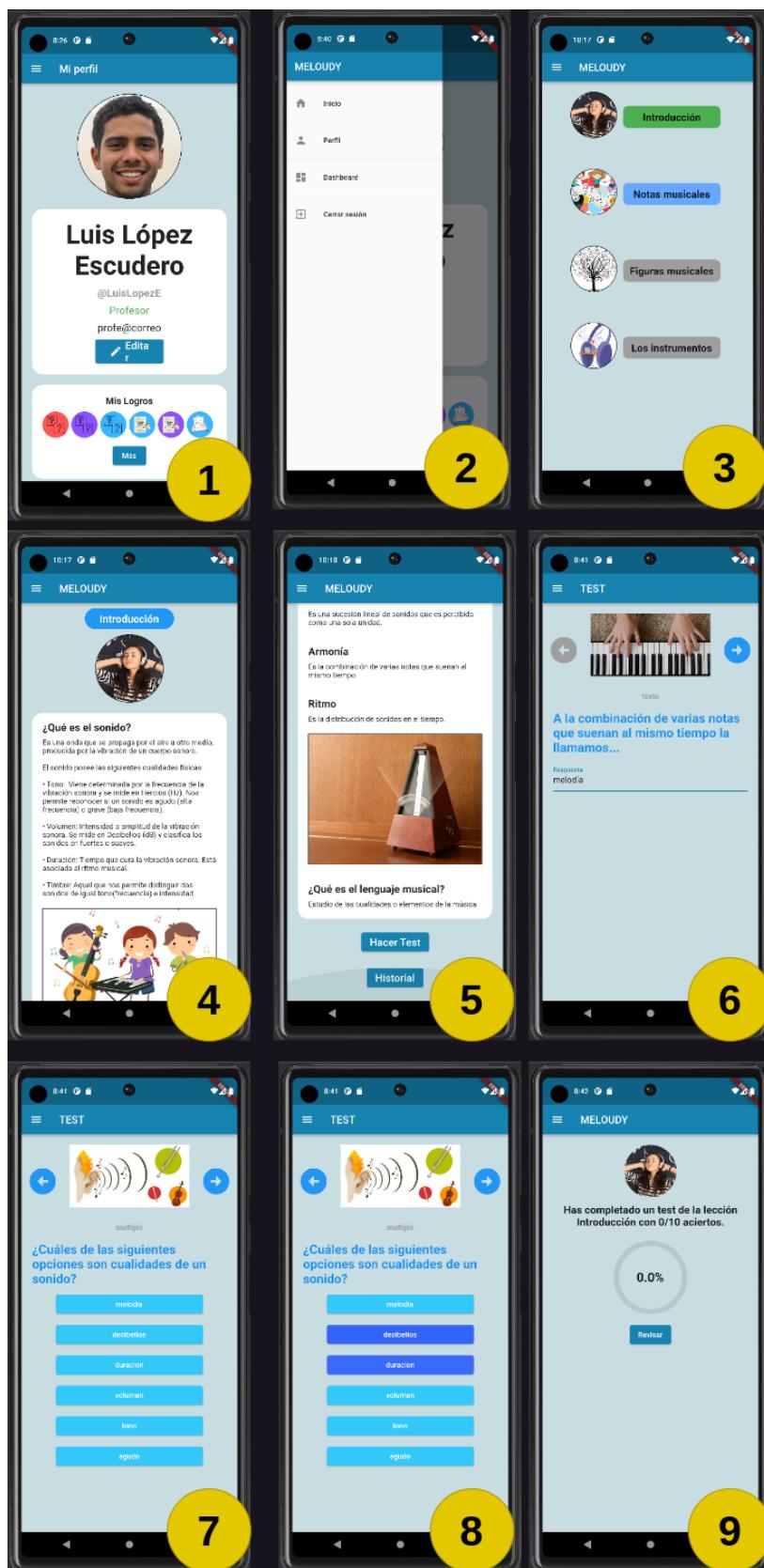


Figura 8.10: Pantallas por las que pasa la prueba de realización de un test

Capítulo 9

Conclusiones

A modo de cierre puedo decir que, tras meses de trabajo, se ha conseguido realizar la aplicación multiplataforma que se pretendía al comienzo del proyecto. Se ha logrado desarrollar un sistema para el aprendizaje musical de forma interactiva y basado en métodos de gamificación. En concreto, se han conseguido cumplir con éxito los siguientes objetivos:

- Se ha construido un sistema seguro y robusto, que permite a los usuarios registrarse y acceder a la aplicación con seguridad.
- Se ha desarrollado un sistema de gestión de usuarios, que permite a los usuarios guardar su progreso y acceder a él desde cualquier dispositivo.
- Se ha implementado un sistema de administración para el profesor y el administrador, que permite gestionar los usuarios, las lecciones, las preguntas y los logros de forma fácil y cómoda.
- Se ha desarrollado un sistema de lecciones, que permite a los usuarios aprender los conceptos básicos de la música de forma interactiva.
- Se ha implementado un sistema de preguntas, que permite a los usuarios poner a prueba sus conocimientos de forma interactiva. En concreto, uno de los mayores retos del proyecto ha sido la implementación de la funcionalidad de preguntas de tipo *micrófono* y tras mucho esfuerzo se ha logrado desarrollar con un resultado satisfactorio.
- Se ha desarrollado un sistema de logros, que permite a los usuarios obtener recompensas por su progreso y sus logros.

El desarrollo de este proyecto ha sido una experiencia muy enriquecedora, ya que ha permitido aprender y poner en práctica muchos de los conocimientos adquiridos durante los cuatro años de grado. Además, me ha

permitido afianzar mis habilidades de programación y aprender nuevas tecnologías y herramientas que no había utilizado anteriormente o de forma muy superficial.

9.1. Valoración personal

Mi experiencia con este proyecto ha sido muy positiva. Además de todos los conocimientos adquiridos, ha sido muy gratificante ver cómo poco a poco la aplicación iba tomando forma y se iba convirtiendo en lo que había imaginado al comienzo del proyecto. Aunque ha sido un proyecto muy ambicioso, he conseguido completarlo con éxito y estoy muy satisfecho con el resultado final. En cuanto al seguimiento del proyecto, también he logrado completar el proyecto en el tiempo estimado, aunque he tenido que dedicar más tiempo del esperado a la implementación de algunas funcionalidades, como por ejemplo la funcionalidad de preguntas de tipo *micrófono* o a las pruebas de integración. Considero que mi motivación durante todos los meses de trabajo se ha visto alzada por la temática y el dominio del proyecto, ya que desde pequeño he estado muy interesado en el aprendizaje musical y esta idea la llevaba pensando unos meses antes de comenzar con el proyecto.

9.2. Trabajo futuro

Aunque el proyecto ha sido completado con éxito, existen algunas funcionalidades que no han podido ser implementadas ni entraban en el alcance de este proyecto, pero que podrían ser implementadas en un futuro para mejorar la aplicación. Algunas de estas funcionalidades son:

- **Sistema de tienda:** Se podría implementar un sistema de tienda, que permita a los usuarios canjear puntos por premios dentro de la aplicación (insignias, fondos, etc.)
- **Sistema de amigos:** Se podría implementar un sistema de amigos, que permita a los usuarios añadir a otros usuarios como amigos y ver sus perfiles.
- **Sistema de chat:** Se podría implementar un sistema de chat, que permita a los usuarios comunicarse entre ellos.
- **Sistema de comentarios:** Se podría implementar un sistema de comentarios, que permita a los usuarios comentar las lecciones y las preguntas o incluso sugerir nuevas preguntas para una lección.

- **Sistema de estadísticas:** Se podría implementar un sistema de estadísticas, que permita a los usuarios ver sus estadísticas de progreso en el perfil de forma más detallada.
- **Más tipos de preguntas:** Se podrían implementar más tipos de preguntas, como por ejemplo preguntas de tipo *arrastrar y soltar* o preguntas de tipo *piano* (tocar en un pequeño piano de la pantalla la(s) nota(s) que se pidan).

Como toda aplicación, Meloudy continuará evolucionando y mejorando con el tiempo, pero por el momento se puede decir que el proyecto ha sido completado con éxito y que la aplicación está lista para ser utilizada por los usuarios sin ningún tipo de limitación.

