

**Universidad
Rey Juan Carlos**

Escuela Técnica Superior
de Ingeniería Informática

Grado en Ingeniería Informática

Curso 2021-2022

Trabajo Fin de Grado

**EARFIT: APLICACIÓN PARA ENTRENAMIENTO
AUDITIVO MUSICAL**

Autor: Alberto Gómez Cano
Tutor: Manuel Rubio Sánchez

Agradecimientos

Quiero agradecer este TFG a mi familia, por siempre estar a mi lado aconsejándome en los momentos más difíciles de la carrera. A mi novia, por aguantar mis frustraciones y animarme a seguir adelante. A mi tutor académico, Manuel Rubio Sánchez, que me ha apoyado para realizar este trabajo. Y finalmente, me gustaría agradecer a todos mis compañeros que han compartido esta carrera conmigo.

¡A todos, mil gracias!

Resumen

La idea consiste en desarrollar una herramienta para ayudar a músicos a desarrollar su oído (Musical Ear Training). Por ejemplo, para identificar notas, intervalos y escalas. Estos ejercicios mejorarán su capacidad musical al desarrollar una comprensión más intuitiva de lo que se escucha.

Consistirá en una aplicación web basada en Next.js y TypeScript y que será desplegada en Vercel. Utilizando Metodologías Ágiles y prácticas de DevOps para llevar a cabo la organización y el desarrollo del producto. Además de Design Thinking y Lean Startup para la ideación, diseño y creación de la solución.

La aplicación se basa en un conjunto de ejercicios de entrenamiento auditivo divididos en la identificación de notas, intervalos y escalas. Donde los usuarios tratarán de adivinar el sonido de cada ejercicio al pulsar el botón de pregunta. Además, los usuarios cuentan con la posibilidad de personalizar estos ejercicios, variar el instrumento que suena, así como de tocar un pequeño piano para ayudarse en la obtención de sus respuestas.

Todo ello en una aplicación web progresiva (PWA) con renderizado en el lado del servidor que tiene como nombre Earfit. La aplicación se puede visitar y descargar en el siguiente enlace:

<https://earfit-alberttoggoca.vercel.app/>

Palabras clave:

- Entrenamiento Auditivo
- Nextjs
- React
- TypeScript
- Vercel
- Agile

Índice de contenidos

Índice de figuras	XIII
1. Introducción	1
1.1. Entrenamiento Auditivo	1
1.1.1. Oído Absoluto	2
1.1.2. Oído Relativo	2
1.2. Estado del Arte	2
1.2.1. ToneGym	3
1.2.2. TonedEar	4
1.2.3. EarMaster	5
1.3. Objetivos	6
1.4. Estructura del documento	6
2. Análisis y Diseño	8
2.1. Design Thinking	8
2.1.1. Empatizar	9
2.1.2. Definir	9
2.1.3. Idear	10
2.1.4. Prototipar	11
2.1.5. Testear	12
2.2. Lean Startup	13
3. Metodologías Ágiles	14
3.1. Scrum	14
3.1.1. User Story Map	16
3.1.2. Scrum Board	17
3.1.3. User Stories	18
3.2. DevOps	19
3.2.1. Integración Continua (CI)	20
3.2.2. Despliegue Continuo (CD)	21
4. Descripción Informática	22
4.1. Stack Tecnológico	22

4.1.1.	Next.js	23
4.1.2.	React	24
4.1.3.	Vercel	24
4.1.4.	TypeScript	24
4.1.5.	Node.js	25
4.1.6.	VSCode	25
4.2.	Detalles de Implementación	26
4.2.1.	Arquitectura	27
4.2.2.	Estructura de Archivos	28
4.2.3.	Infraestructura	30
4.2.4.	Datos y Servicios	33
4.2.5.	Hooks y Context	33
4.2.6.	Componentes y Pages	36
4.2.7.	Progressive Web App	39
4.2.8.	Buenas Prácticas	41
4.3.	Testing	42
4.3.1.	Pruebas Funcionales	42
4.3.2.	Pruebas No Funcionales	44
5.	Conclusiones	47
5.1.	Objetivos Alcanzados	47
5.2.	Trabajos futuros	49
5.3.	Opinión Personal	49
Bibliografía		51
Apéndices		56
A. Proceso de Creación Detallado		58
A.1.	Resumen	58
A.2.	Design Thinking	59
A.2.1.	Empatizar	59
A.2.2.	Definir	59
A.2.3.	Idear	59
A.2.4.	Prototipar	60
A.2.5.	Testear	60
A.3.	Lean Startup	60
A.3.1.	Crear	60
A.3.2.	Medir	60
A.3.3.	Aprender	61
A.3.4.	Conclusión	62

B. Diseños del Prototipo	63
B.1. Prototipo Pantallas Pequeñas	64
B.2. Prototipo Pantallas Medianas	65
B.3. Prototipo Pantallas Grandes	67
C. Historias de Usuario	69
D. Capturas de la Aplicación	84
D.1. Pantalla Pequeña	84
D.2. Pantalla Media	86
D.3. Pantalla Grande	89
E. Clean Code	93
F. Limitaciones Personales	94

Índice de figuras

1.1. Página de ejercicio de ToneGym.	3
1.2. Página de ejercicio de TonedEar.	4
1.3. Página de inicio de EarMaster.	5
1.4. Precios de EarMaster.	5
1.5. Proceso combinado de Design Thinking, Lean Startup y Agile. . .	6
2.1. Etapas de Design Thinking.	9
2.2. MindMap de los principales problemas y soluciones.	10
2.3. Método MoSCoW con las prioridades del proyecto.	11
2.4. Prototipo para pantallas pequeñas.	12
2.5. Circuito de feedback de información.	13
3.1. Marco de trabajo Scrum.	15
3.2. User Story Map del producto.	16
3.3. Scrum Board con el Product Backlog y Sprint Backlog.	17
3.4. Historia de usuario “Cambiar Instrumento”.	18
3.5. Diagrama de DevOps.	19
3.6. Modelo de creación de ramas Gitflow.	20
3.7. Flujo de trabajo DPS: Develop, Preview and Ship.	21
4.1. Stack Tecnológico.	22
4.2. Arquitectura de la aplicación.	27
4.3. Estructura de Archivos.	28
4.4. Tipo Instrument.	30
4.5. Tipo Noteplayer.	30
4.6. Tipo VariantExercise.	31
4.7. Tipo Answer.	31
4.8. Diagrama de un componente React.	36
4.9. Jerarquía de Componentes de la aplicación.	36
4.10. Código de la página de intervalos.	37
4.11. Código del componente AnswerButtons.	38
4.12. Instalar la aplicación como PWA desde Chrome.	39
4.13. Captura de la página de intervalos.	40

4.14. Puntuaciones de Lighthouse.	44
4.15. Vista de análisis de las Web Vitals.	45
5.1. Código QR de Earfit.	48
B.1. Elementos IU de Atomic Design.	63
B.2. Prototipo de la Página de Inicio y de Notas (Pantallas Pequeñas).	64
B.3. Prototipo de la Página de Intervalos y de Escalas (Pantallas Pequeñas).	64
B.4. Prototipo de la Página de Inicio (Pantallas Medianas).	65
B.5. Prototipo de la Página de Notas (Pantallas Medianas).	65
B.6. Prototipo de la Página de Intervalos (Pantallas Medianas).	66
B.7. Prototipo de la Página de Escalas (Pantallas Medianas).	66
B.8. Prototipo de la Página de Inicio (Pantallas Grandes).	67
B.9. Prototipo de la Página de Notas (Pantallas Grandes).	67
B.10. Prototipo de la Página de Intervalos (Pantallas Grandes).	68
B.11. Prototipo de la Página de Escalas (Pantallas Grandes).	68
C.1. Product Backlog con las historias de usuario.	69
C.2. Historia de usuario “Seleccionar Ejercicio Menú Principal”.	70
C.3. Historia de usuario “Cambiar Ejercicio Menú Lateral”.	71
C.4. Historia de usuario “Escuchar Nota”	72
C.5. Historia de usuario “Elegir Respuesta Nota”	73
C.6. Historia de usuario “Escuchar Intervalo”	74
C.7. Historia de usuario “Elegir Respuesta Intervalo”	75
C.8. Historia de usuario “Escuchar Escala”	76
C.9. Historia de usuario “Elegir Respuesta Escala”	77
C.10. Historia de usuario “Piano”	78
C.11. Historia de usuario “Cambiar Opciones de Respuesta”	79
C.12. Historia de usuario “Cambiar Instrumento”	80
C.13. Historia de usuario “Cambiar Escala Notas”	81
C.14. Historia de usuario “Cambiar Dirección Intervalo”	82
C.15. Historia de usuario “Cambiar Dirección Escala”	83
D.1. Captura de la Página de Inicio y Notas (Pantallas Pequeñas) . . .	84
D.2. Captura de la Página de Intervalos y Escalas (Pantallas Pequeñas) .	85
D.3. Captura de la Página de Escalas, Piano y About (Pantallas Pequeñas) .	85
D.4. Captura de la Página de About (Pantallas Pequeñas)	85
D.5. Captura de la Página de Inicio (Pantallas Medianas)	86
D.6. Captura de la Página de Notas (Pantallas Medianas)	86
D.7. Captura de la Página de Intervalos (Pantallas Medianas)	87
D.8. Captura de la Página de Escalas (Pantallas Medianas)	87

D.9. Captura de la Página de Piano (Pantallas Medianas).	88
D.10.Captura de la Página de About 1 (Pantallas Medianas).	88
D.11.Captura de la Página de About 2 (Pantallas Medianas).	89
D.12.Captura de la Página de Inicio (Pantallas Grandes).	89
D.13.Captura de la Página de Notas (Pantallas Grandes).	90
D.14.Captura de la Página de Intervalos (Pantallas Grandes).	90
D.15.Captura de la Página de Escalas (Pantallas Grandes).	91
D.16.Captura de la Página de Piano (Pantallas Grandes).	91
D.17.Captura de la Página de About 1 (Pantallas Grandes).	92
D.18.Captura de la Página de About 2 (Pantallas Grandes).	92

1

Introducción

En este capítulo se explica en qué consiste el entrenamiento auditivo para que se pueda comprender mejor el objetivo general y alcance del trabajo. Después, se analiza el estado del arte actual de aplicaciones relacionadas. Por último, se establecen los objetivos del trabajo y la estructura del documento.

1.1. Entrenamiento Auditivo

Los oídos son la herramienta más importante a la hora de hacer música. Pero si no se entrena, nunca se desarrollarán por completo. Los músicos, productores y DJs pueden beneficiarse del entrenamiento auditivo, ya que puede resultar muy útil a la hora de mezclar música y componer canciones.

El entrenamiento auditivo es el proceso de identificar los elementos de la música en su forma más sencilla y conectarlos con la forma en que sentimos el sonido físicamente. Tradicionalmente, el entrenamiento auditivo incluye habilidades como identificar notas, intervalos y escalas.

Para los músicos es importante porque la escucha es una habilidad al igual que tocar el piano. Por ejemplo, las melodías son simplemente series de intervalos. Con el entrenamiento necesario para identificar los intervalos, los músicos pueden aprender a tocar una melodía de oído.

Para los productores de música y DJs, el entrenamiento auditivo sirve para identificar los rangos de frecuencias más rápidamente y ayudarlos también a conseguir los efectos buscados. [1]

En conclusión, practicar el entrenamiento auditivo te lleva al siguiente nivel como músico ya que te permite sacar canciones más rápido, con mayor precisión, improvisar mejor, llevar al instrumento las melodías que imaginas con mayor facilidad, y en general te permitirá ser mucho mejor músico. [2]

1.1.1. Oído Absoluto

Es la habilidad para reconocer notas musicales sin tener otras como referencia. Es relativamente raro encontrar personas con oído absoluto. Se considera que menos del uno por ciento de la población tiene oído absoluto. Las posibilidades de tener oído absoluto aumentan si has recibido mucho entrenamiento musical desde muy pequeño. [3]

1.1.2. Oído Relativo

Es la habilidad para reconocer notas musical relacionándolas entre sí. Es una habilidad indispensable para los músicos y es más sencilla de entrenar que el oído absoluto. Esta característica te puede permitir, por ejemplo, interpretar canciones sin disponer de partitura.

Las personas que disponen de oído relativo son capaces de:

- Denotar la distancia de una nota musical desde una nota de referencia establecida.
- Seguir la notación musical, esto permite cantar correctamente una melodía entonando cada nota de acuerdo a la distancia con la nota anterior.
- Identificar intervalos entre notas dadas, de manera independiente a la afinación elegida.

Los ejercicios más comunes de entrenamiento auditivo ayudan a desarrollar este oído relativo principalmente. [3]

1.2. Estado del Arte

En la actualidad ya existen algunas aplicaciones que contemplan el entrenamiento auditivo como pueden ser **ToneGym**, **TonedEar** y **EarMaster**. En esta sección se describen sus pros y contras con el objetivo de obtener información útil de cara a las fases de análisis y diseño de este trabajo.

1.2.1. ToneGym

Esta aplicación cuenta con los ejercicios clásicos de entrenamiento musical y un menú siempre visible a la izquierda para que puedas moverte entre ellos fácilmente (ver Figura 1.1). Cuando aciertas la respuesta en un ejercicio recibes feedback del botón mediante sonido, el botón se vuelve verde unos segundos y automáticamente se pasa al siguiente sonido, lo que agiliza la práctica. [4]

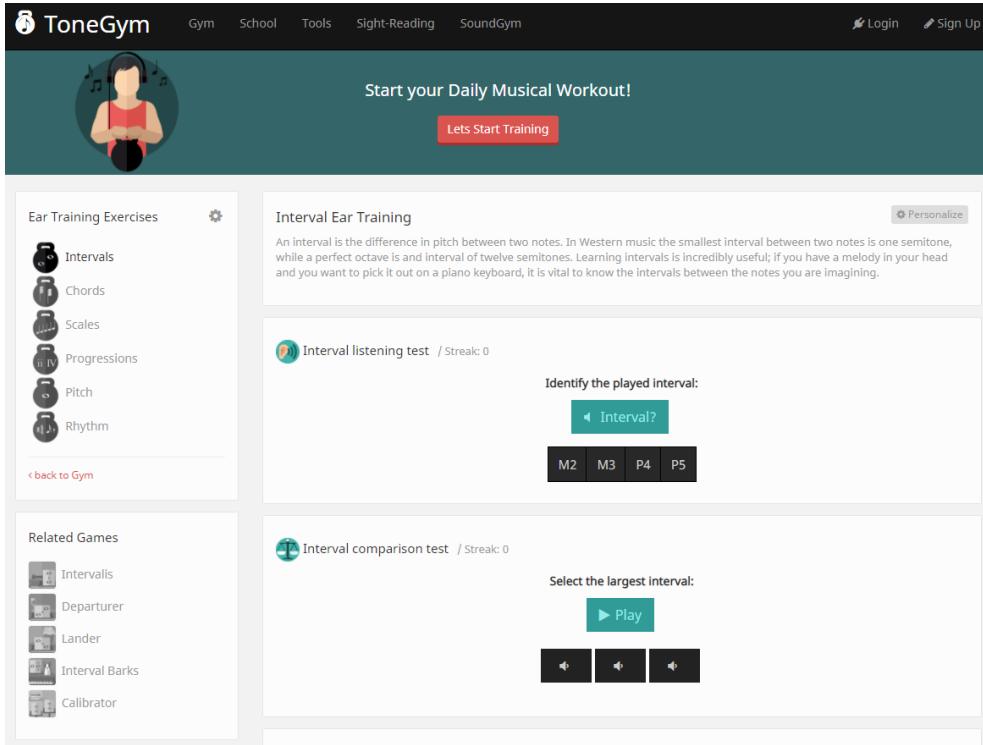


Figura 1.1: Página de ejercicio de ToneGym.

Sin embargo, lo primero que vemos al entrar en la aplicación es una landing page en la que si pulsas Lets Start Training nos lleva a crearnos una cuenta en lugar de a los ejercicios. Esto es algo que echa para atrás a probar una aplicación que no conoces. [5]

La sección de ejercicios está un poco escondida y hay que deslizar hacia abajo para encontrarla. Aunque el diseño está bastante bien, la personalización de los ejercicios es nula. Además, una vez hayas fallado en la respuesta no se te permite continuar probando. Por último, no incorpora ninguna opción para cambiar el instrumento que suena. [6]

1.2.2. TonedEar

Esta aplicación es la que más se parece a lo que queremos construir (ver Figura 1.2). Contiene varios tipos de ejercicios como identificación de intervalos, escalas, etc. Permite la personalización de los ejercicios mediante un panel a la derecha, aunque no es inmediato. Y ofrece la posibilidad de continuar con el ejercicio aunque falles en la primera respuesta. Además, es una aplicación web gratuita, sencilla, fácil de entender y usar. [7]

The screenshot shows a web-based ear training application. At the top, there's a navigation bar with links to 'Home', 'Exercises ▾', 'How to practice', 'For teachers', 'Android & iOS App', and 'Contact Me'. Below the navigation is a main section titled 'Intervals Quiz' with a progress bar showing '0 of 0 correct'. A 'Hear Again' button is available for each question. To the right is a 'Choices' section with three buttons: 'Major 3rd', 'Perfect 5th', and 'Octave'. Further right is an 'Options' section where users can select intervals ('Minor 2nd', 'Major 2nd', 'Minor 3rd', 'Major 3rd', 'Perfect 4th', 'Tritone', 'Perfect 5th', 'Minor 6th', 'Major 6th', 'Minor 7th', 'Major 7th', 'Octave') and speed ('Ascending', 'Medium Speed'). A note says 'Select the intervals on which you would like to be tested'.

Figura 1.2: Página de ejercicio de TonedEar.

Algo que se hace notar analizando el código de la aplicación es que en el ejercicio de identificación de intervalos no se encuentran todos los intervalos posibles. En concreto, una de las dos notas del intervalo será un número MIDI desde 40 hasta 65 y la segunda nota será la primera más un número del 1 al 14. Esto nos da un rango de notas desde 40 hasta 79 cuando el rango de notas MIDI va desde 20 hasta 108 para un piano. Además, puede ocurrir que la primera nota del intervalo suene después de la segunda.

Por último, la aplicación permite personalizar los ejercicios pero no permite cambiar entre instrumentos. El instrumento que suena siempre al pulsar el botón de pregunta es un piano. [8]

1.2.3. EarMaster

Parece ser la más completa y elaborada de todas con más de 2500 ejercicios (ver Figura 1.3). Contiene ejercicios para que puedas reconocer, transcribir, leer, cantar y tocar: Intervalos, acordes, progresiones armónicas, escalas, ritmos y melodías. [9]



Figura 1.3: Página de inicio de EarMaster.

Sin embargo, su contenido gratuito sólo consta de identificación de intervalos y acordes. Para poder hacer uso de la aplicación completa ofrece precios desde los 59,95€ hasta los 99,95€ (ver Figura 1.4). Aparte, necesitas tener la aplicación instalada para utilizarla, no es una aplicación web. Todo esto resulta en que el usuario al que nos estamos enfocando no pueda usarla. Esta aplicación parece estar más destinada a centros educativos que puedan hacerse cargo de la licencia. [10]



Figura 1.4: Precios de EarMaster.

1.3. Objetivos

El objetivo principal del TFG es desarrollar una aplicación que permita a músicos desarrollar su oído musical mediante el entrenamiento auditivo.

Subobjetivos:

- Desarrollar una interfaz interactiva.
- Implementar diferentes tipos de ejercicio personalizables de entrenamiento auditivo.
- Incluir varios instrumentos para practicar con diferentes sonidos.

1.4. Estructura del documento

Este trabajo se centra en el siguiente proceso combinando de Design Thinking, Lean Startup y Metodologías Ágiles para la creación de un producto mínimo viable. De esta manera conseguimos un proceso que parte de la nada, se centra en el usuario, gestiona el desarrollo, integra el error y es iterable. [11]

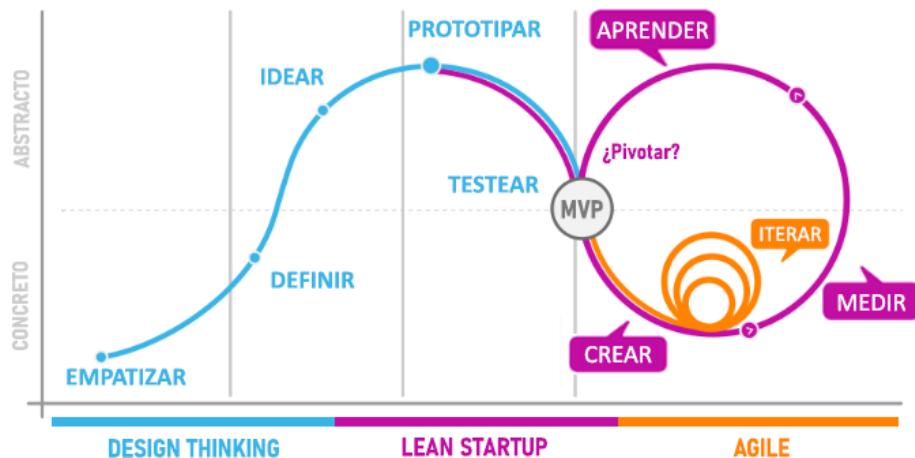


Figura 1.5: Proceso combinado de Design Thinking, Lean Startup y Agile.

En la Figura 1.5 se ilustra este proceso que se ha dividido en la memoria en los siguientes tres capítulos:

- **Análisis y Diseño:** Este capítulo recoge todo lo relacionado con la fase de ideación y puesta en marcha de la solución. Recoge explicaciones del proceso de Design Thinking y Lean Startup. Aquí se pueden encontrar el diseño y el prototipo de la aplicación.

- **Metodologías Ágiles:** Este capítulo recoge lo relacionado para llevar a cabo la gestión y desarrollo del software utilizando Scrum y prácticas DevOps. Aquí se puede encontrar cómo se ha realizado la gestión, especificación de requisitos, integración continua y despliegue continuo.
- **Descripción Informática:** Este capítulo se centra en lo relacionado con la implementación de la solución. Aquí se puede encontrar las tecnologías usadas, los detalles de implementación del código, buenas prácticas y testing.

Por último, se encuentran las **Conclusiones** derivadas del trabajo con los objetivos alcanzados y una propuesta para trabajos futuros. Y los **Apéndices**, que complementan la memoria principal con un mayor desarrollo de alguno de sus apartados, capturas de la aplicación e imágenes grandes.

2

Análisis y Diseño

Para encontrar una solución viable al problema que se plantea y que aporte un valor real a los usuarios, necesitamos de algún método que nos permita diseñar una solución de manera efectiva bajo unas condiciones de incertidumbre.

Para crear esta propuesta de valor, se puede utilizar **Design Thinking** como método de generación de ideas innovadoras y **Lean Startup** como método de aprendizaje validado para la puesta en marcha de la solución. [12]

2.1. Design Thinking

Design Thinking es un método para **generar ideas innovadoras** que centra su eficacia en entender y dar solución a las necesidades reales de los usuarios. Proviene de la Universidad de Stanford en California (EEUU) y la forma en la que trabajan los diseñadores de producto, de ahí su nombre. [13]

Esta metodología sirve en el proceso de creación para conocer al cliente en profundidad y encontrar soluciones prácticas ante sus problemas de manera ágil. Su objetivo es partir desde las necesidades de los clientes, para generar productos o servicios que las satisfagan.

El proceso de Design Thinking se compone de cinco etapas y no es lineal. Por tanto, en cualquier momento se puede saltar entre etapas si se cree oportuno. De modo que a lo largo del proceso se va afinando el contenido hasta desembocar en una solución que cumpla con los objetivos. [14]

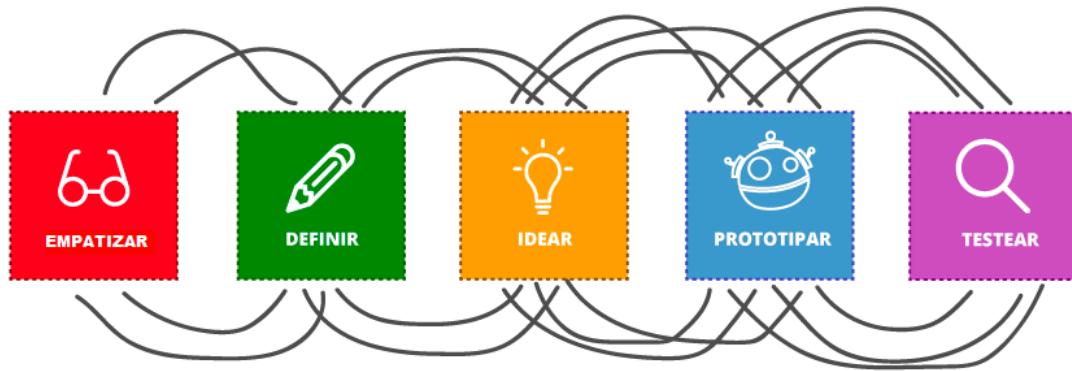


Figura 2.1: Etapas de Design Thinking.

En la Figura 2.1 podemos observar las etapas de las que se compone Design Thinking. El objetivo que pretende abordar cada etapa es el siguiente:

- **Empatizar:** Comprender el entorno y necesidades del usuario.
- **Definir:** Identificar los problemas a solucionar.
- **Idear:** Dar solución a estos problemas.
- **Prototipar:** Convertir las soluciones en un prototipo.
- **Testear:** Probar el prototipo con el usuario.

2.1.1. Empatizar

En esta etapa se realizaron entrevistas al usuario y un estudio profundo de teoría musical para entender al usuario y sus necesidades. Además, se analizaron diversas herramientas ya existentes de entrenamiento musical como se ha explicado en el estado del arte. Se puede encontrar una explicación más detallada en el apéndice [A.2 Design Thinking](#).

2.1.2. Definir

El problema parte de la necesidad de músicos amateur que quieren mejorar su nivel de percepción de la música. Sin los medios adecuados resulta imposible llevar a cabo el entrenamiento auditivo que es una parte fundamental para llevar a cabo su propósito.

2.1.3. Idear

Para idear una solución que aporte valor, realizamos un **MindMap** [15]. Esta es una herramienta gráfica para representar ideas o conceptos y encontrar soluciones. En ella tratamos de enfocarnos en los principales problemas del usuario y sus posibles soluciones.

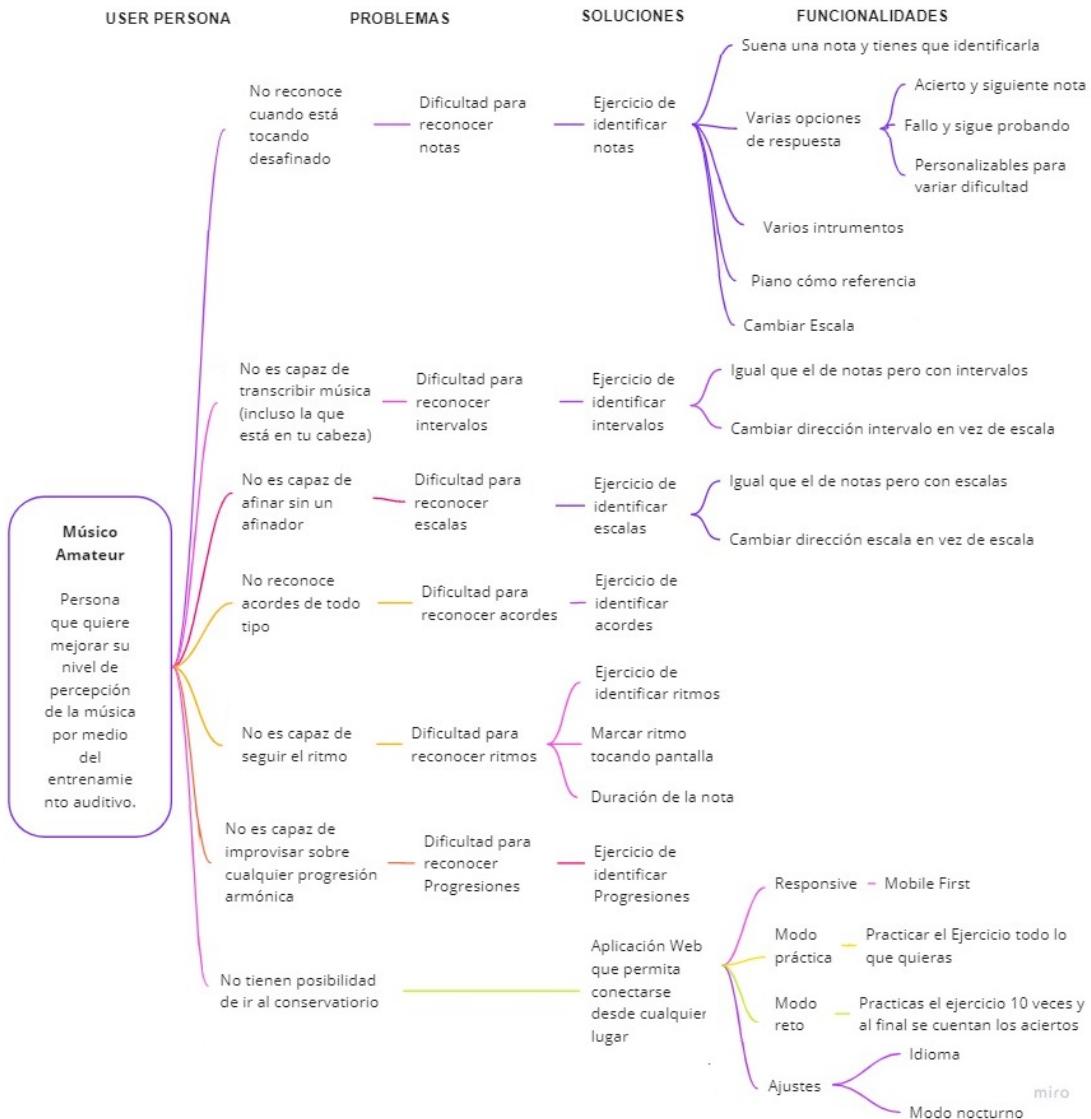


Figura 2.2: MindMap de los principales problemas y soluciones.

En la Figura 2.2 se observa este Mindmap donde se concluye que una posible forma de solucionar estos problemas es mediante una aplicación web con ejercicios de entrenamiento auditivo.

2.1.4. Prototipar

En este etapa utilizamos la técnica MoSCoW [16]. Esta es una técnica de priorización utilizada en gestión de proyectos. En ella tratamos de establecer las prioridades del proyecto, teniendo en cuenta nuestras limitaciones: poco tiempo para desarrollar, aprender conceptos musicales y nuevas tecnologías.

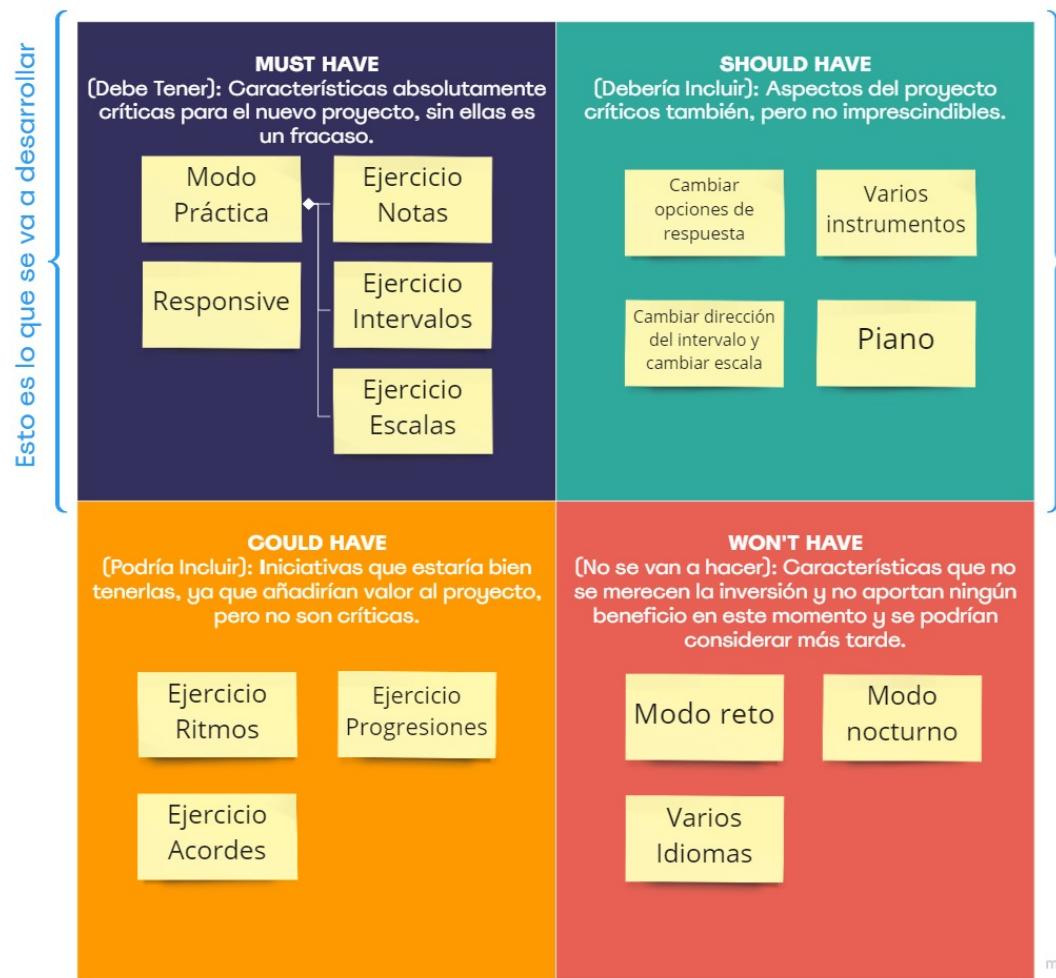


Figura 2.3: Método MoSCoW con las prioridades del proyecto.

En la Figura 2.3 se observa esta técnica de MoSCoW. Esta técnica se basa en definir lo que debe tener (Must have), debería incluir (Should have), podría incluir (Could have) y no va a tener (Won't have) la aplicación.

Para crear nuestro **producto mínimo viable** se decidió centrarse en desarrollar los Must haves y más tarde los Should haves. Los Could haves y los Won't haves se deciden dejar para trabajos futuros por complejidad y tiempo de desarrollo.

Por último, una vez establecidas las prioridades del proyecto pasamos a visualizarlas diseñando el prototipo teniendo en cuenta la filosofía de diseño **Atomic Design** [17] y **Mobile First** [18]. Diseñando primero para pantallas pequeñas y después adaptándolo a pantallas más grandes.

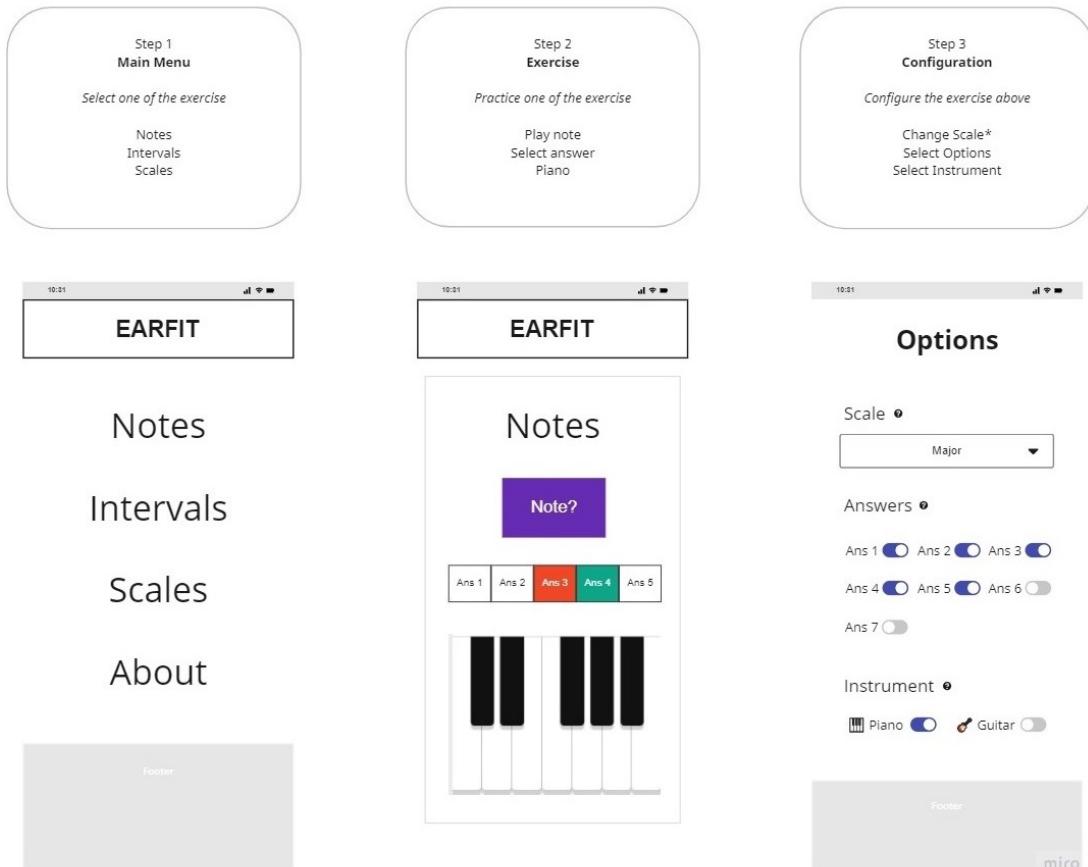


Figura 2.4: Prototipo para pantallas pequeñas.

En la Figura 2.4 se puede apreciar una pequeña porción del prototipo para pantallas pequeñas. Todos los demás diseños del prototipo de la aplicación, incluyendo pantallas pequeñas, medianas y grandes, se pueden encontrar en el apéndice **B. Prototipos**.

2.1.5. Testear

Después de haber testeado los primeros diseños con el usuario, haber recogido feedback, haber hecho correcciones y haber validado el último prototipo que hemos visto anteriormente, pasamos a crear esta solución aplicando el método **Lean Startup**.

2.2. Lean Startup

Es un método riguroso para **agilizar la puesta en marcha** de soluciones y optimizarlas con base en un proceso de aprendizaje y de corrección iterativa. Comenzó con el método de desarrollo de clientes y el método Lean en los sistemas de fabricación japoneses popularizado por Toyota.

La metodología Lean Startup se basa en el circuito de feedback de información: **crear, medir, aprender** (ver Figura 2.5). Crear una hipótesis, diseñar un experimento (producto mínimo viable) para testear esa hipótesis, llevar a cabo el experimento, reunir datos, reflexionar y ver si validan o rechazan la hipótesis para pivotar. [19]

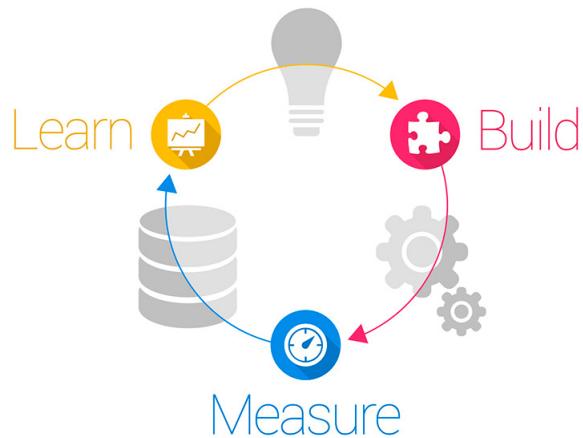


Figura 2.5: Circuito de feedback de información.

Lean Startup sirve para acortar los ciclos de desarrollo, medir el progreso y ganar feedback por parte de los usuarios. Esto se consigue porque se basa en el aprendizaje validado, la experimentación científica y la iteración en los lanzamientos del producto. Debido a esto es considerado una herramienta imprescindible para la puesta en marcha de soluciones software.

Para no extender este trabajo se puede encontrar una explicación detallada de como fue este proceso de crear, medir, aprender en el apéndice [A.3 Lean Startup](#). A continuación, nos centraremos sólo en la fase de creación del software. [20]

Después haber establecido nuestra hipótesis y diseñado un producto mínimo viable usando **Design Thinking** pasamos a desarrollarlo bajo el paraguas de **Metodologías Ágiles**.

3

Metodologías Ágiles

Para gestionar el desarrollo del software de manera exitosa necesitamos asegurar un proceso visible, controlado, repetitivo, eficiente y predecible. Necesitamos adaptar las formas de trabajo a las necesidades del proyecto, prolongando respuestas rápidas y flexibles para acomodar el desarrollo.

La agilidad es la habilidad, que facilita la adaptación, de manera rápida y efectiva en circunstancias cambiantes, para garantizar la entrega de valor continuo, en ciclos cortos de tiempo y con el mínimo coste. [21]

Para ser ágiles mientras desarrollamos podemos beneficiarnos de **DevOps** como filosofía de desarrollo y **Scrum** como modelo organizativo. [22]

3.1. Scrum

Scrum es un **proceso de gestión** que reduce la complejidad en el desarrollo de productos para satisfacer las necesidades de los clientes. Se considera un marco de gestión de proyectos ágil.

Scrum no es una técnica para desarrollar productos, es un marco donde se puede emplear un conjunto de diferentes procesos y técnicas.

Es ideal para crear nuevos productos poniendo el foco en el cliente y detectar fallos pronto. Lo que se traduce en una mayor calidad del producto, reducción de costes y optimización del riesgo. [23]

Con Scrum, un producto se basa en una serie de iteraciones llamadas **sprints** que dividen proyectos grandes y complejos en partes más pequeñas. Priorizadas por el beneficio que aportan al receptor del producto. [24]

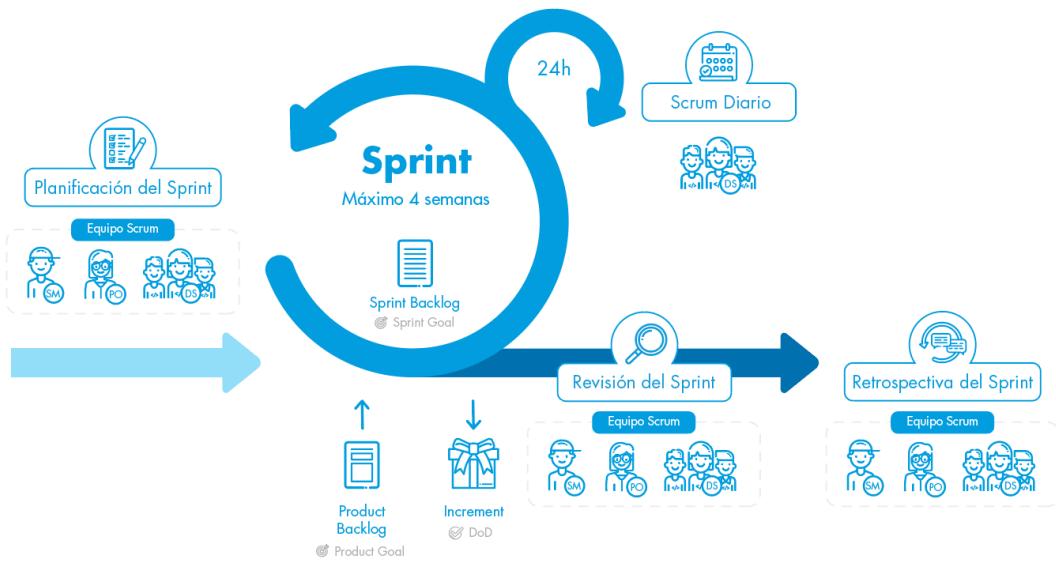


Figura 3.1: Marco de trabajo Scrum.

En la Figura 3.1 podemos observar el **flujo de trabajo** de Scrum. Este flujo de trabajo está formado por los siguientes pasos:

1. Un Product Owner ordena el trabajo de un problema complejo en un Product Backlog.
2. El Equipo Scrum convierte una selección del trabajo en un Incremento de valor durante un Sprint.
3. El Equipo Scrum y sus partes interesadas inspeccionan los resultados y se ajustan para el próximo Sprint.
4. *Repetir*

Implementar este marco de trabajo nos permite obtener resultados pronto, reduciendo el **Time to Market**, es decir, a tenerlo antes posible en el mercado nuestro producto o una de sus características, aumentando la satisfacción del cliente. Además, permite lidiar con requisitos cambiantes o poco definidos, aportando tolerancia al cambio.

3.1.1. User Story Map

Es un método de mapeo de UX (experiencia de usuario) que se utiliza para delinear las interacciones que se espera que realicen los usuarios para completar sus objetivos en un producto digital. Ayuda a definir el **víaje o casos de uso** del usuario en el producto. [25] [26] [27]

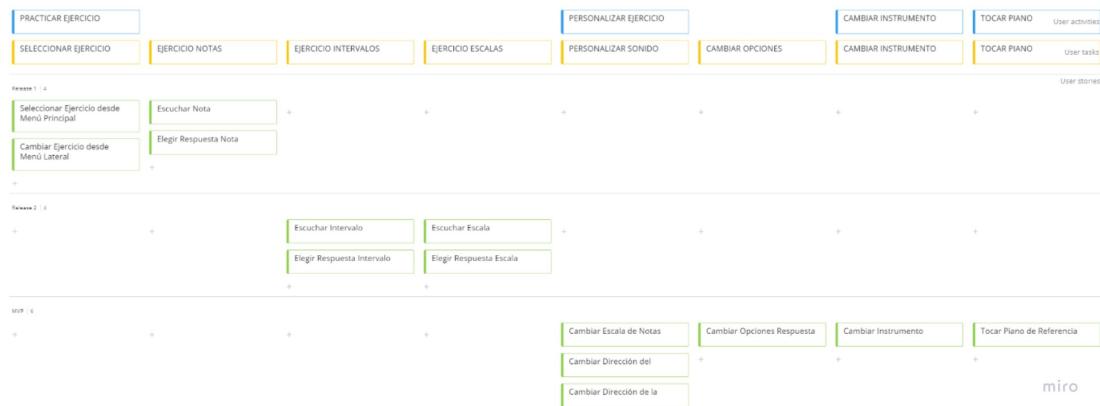


Figura 3.2: User Story Map del producto.

En la Figura 3.2 podemos observar el User Story Map o mapa de historias de usuario del producto. Este User Story Map representa 3 tipos de acciones:

- Las **User Activities**: Representan las tareas de alto nivel que los usuarios pretenden completar en el producto. Son los principales objetivos del usuario en la aplicación.
- Las **User Task**: Representan las subtareas específicas que los usuarios realizarán en el producto para completar la actividad superior. Son las actividades a realizar por el usuario y siguen un flujo narrativo.
- Las **User Stories** o historias de usuario: Describen las interacciones concretas que experimentarán los usuarios para completar el paso anterior. Definen las funcionalidades del producto y están agrupadas por releases.

Las User Activities y User Task se muestran horizontalmente en la parte superior y las User Stories se apilan verticalmente debajo en orden de prioridad.

Se recomienda visitar este mural de Miro, donde se puede encontrar este road-map junto con más esquemas y diseños de la aplicación, en el siguiente enlace:

<https://miro.com/app/board/uXjVOMmPLjM=/>

3.1.2. Scrum Board

Es un método para **visualizar el trabajo** y gestionar el desarrollo. Esto nos permite separar en pequeñas tareas cada historia de usuario y minimizar los riesgos de entrega. Realizando una construcción iterativa incremental donde se integra al final de cada sprint para validar el resultado. [28] [29]

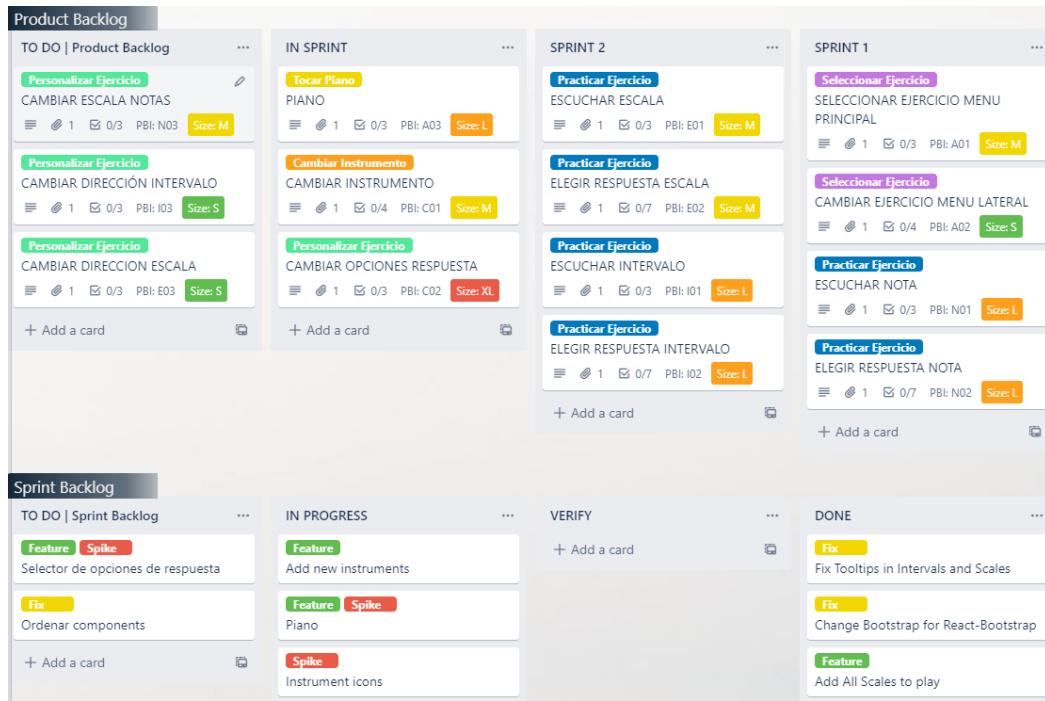


Figura 3.3: Scrum Board con el Product Backlog y Sprint Backlog.

En la Figura 3.3 podemos observar el Scrum Board del proyecto en un punto a mitad del desarrollo. Este Scrum Board está compuesto por:

- **El Product Backlog:** Es la lista de todas las tareas o historias de usuario detalladas pendientes para el desarrollo. Es una evolución de los documentos de requisitos tradicionales y desglosa el roadmap del producto en un documento vivo de tareas realizables.
- **El Sprint Backlog:** Es una lista con todas las tareas técnicas que se compromete a llevar a cabo dentro de un Sprint. Cada historia de usuario se divide en estas tareas más pequeñas a completar. Estas tareas pasan por diferentes etapas hasta completarse.

Este Scrum Board se puede visitar en el siguiente enlace:

<https://trello.com/b/ZnsnNP2h/earfit>.

3.1.3. User Stories

Las historias de usuarios son descripciones breves y sencillas de las **características o requisitos** del sistema contadas desde la perspectiva del usuario o cliente del sistema. Cada una es un incremento en el valor del producto. [30] [31]

The screenshot shows a Trello card for a user story:

- Title:** CAMBIAR INSTRUMENTO
- Labels:** Cambiar Instrumento
- Description:**
 - Como: Músico Amateur
 - Quiero: Cambiar entre varios instrumentos
 - Para: Practicar los ejercicios con sus sonidos
- Custom Fields:**
 - PBI: C01
 - Size: M
- Attachments:**
 - Instrument: Piano, Guitar
 - Prototipo: Added yesterday at 8:00 PM - Comment - Delete - Edit
Make cover
- Criterios de Aceptación:**
 - 0% completed
 - checkboxes for acceptance criteria:
 - Dado: Un selector de instrumentos en la sección de opciones situada a la derecha del ejercicio
 - Cuando: Pulses sobre un botón de instrumento
 - Entonces: El sonido que emita el botón de play será el del instrumento seleccionado
 - Condiciones: Siempre tiene que haber un instrumento seleccionado

Figura 3.4: Historia de usuario “Cambiar Instrumento”.

En la Figura 3.4 se puede observar una de las historias de usuario. Todas siguen el método INVEST para garantizar que ofrecen valor. Y el método SMART para descomponerlas en tareas. [32]

Estas historias de usuario se encuentran en el apéndice **C. Historias de Usuario**. También se pueden ver en el ScrumBoard en el siguiente enlace:

<https://trello.com/b/ZnsnNP2h/earfit>.

3.2. DevOps

DevOps, es una combinación de los términos desarrollo (development) y operaciones (operations), significa la unión de personas, procesos y tecnología para ofrecer valor a los clientes de forma constante. (Ver Figura 3.5).

Se enfoca en **agilizar los procesos** con los que una nueva funcionalidad, mejora o corrección pasa del desarrollo a la implementación, a un entorno de producción que pueda generar valor para el usuario.

Algunas de las prácticas recomendadas de DevOps que ya se han visto son: la gestión de proyectos de manera ágil con Scrum, creación de registros de trabajo pendiente, visualización del progreso y recopilación de feedback continuo entre otros.

El desarrollo de **aplicaciones modernas** requiere procesos diferentes a los enfoques del pasado. Las nuevas empresas utilizan enfoques ágiles para desarrollar sistemas de software. [33]

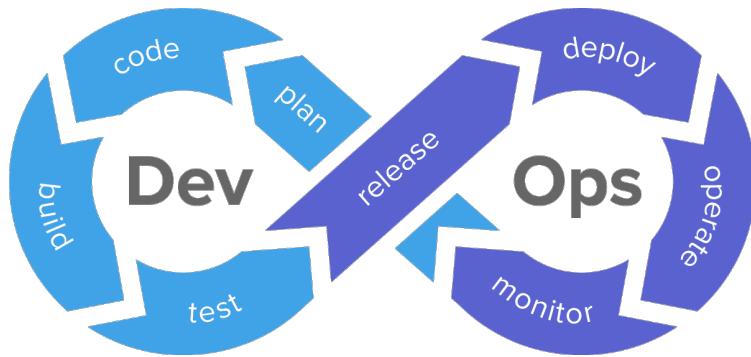


Figura 3.5: Diagrama de DevOps.

DevOps permite fabricar software más rápidamente, con mayor calidad, menor coste y una alta frecuencia de releases. Al adoptar prácticas de DevOps, se asegura la confiabilidad, la alta disponibilidad y el objetivo de ningún tiempo de inactividad del sistema.

El primero de los 12 principios del Manifiesto Ágil es el siguiente: “Satisfacer a los clientes mediante la distribución de software continua y oportuna” [21]. Este es uno de los motivos por el que es importante aplicar prácticas de DevOps, como la **Integración Continua** y el **Despliegue Continuo**. [34]

3.2.1. Integración Continua (CI)

La integración continua es una práctica de DevOps mediante la cual los desarrolladores combinan los cambios en el código en un repositorio central de forma periódica. Para implantar integración continua se suele definir un “pipeline”, es decir, un conjunto de etapas y de fases automatizadas por las que va pasando el software hasta integrarse con el resto. [35]

En nuestro caso usamos **Github** [36], que es un servicio basado en la nube, como herramienta de control de versiones **Git** [37]. Y utilizamos **Gitflow** [38], que es un modelo alternativo de creación de ramas en Git.

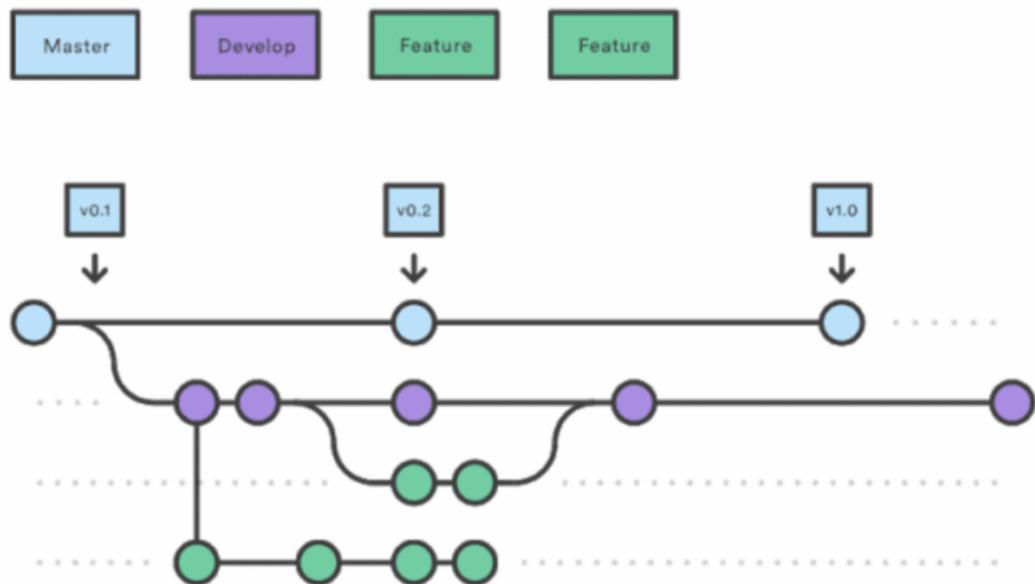


Figura 3.6: Modelo de creación de ramas Gitflow.

En la Figura 3.6 podemos observar este modelo de creación de ramas. Este se caracteriza por el uso de varias ramas de función, una rama de desarrollo y una rama de producción:

- **Ramas de Función:** Cada una de estas ramas se crea desde la rama de desarrollo para implementar una nueva funcionalidad del sistema.
- **Rama de Desarrollo:** Cada vez que una rama de funcionalidad es completada se une a ésta integrando los nuevos cambios.
- **Rama de Producción:** Cada vez que una nueva versión está lista en la rama de desarrollo se une a ésta generando una nueva release.

3.2.2. Despliegue Continuo (CD)

El despliegue continuo es una estrategia de DevOps en la que los cambios de código de una aplicación se publican automáticamente. Es útil para entregar funcionalidades del software de forma frecuente. [39]

Para ello utilizamos **GitHub Actions** [40] el cuál nos permite personalizar y automatizar estos despliegues desde nuestro repositorio. Y para alojar nuestra aplicación web usamos **Vercel** [41], que es una plataforma en la nube optimizada para proyectos **Nextjs**.

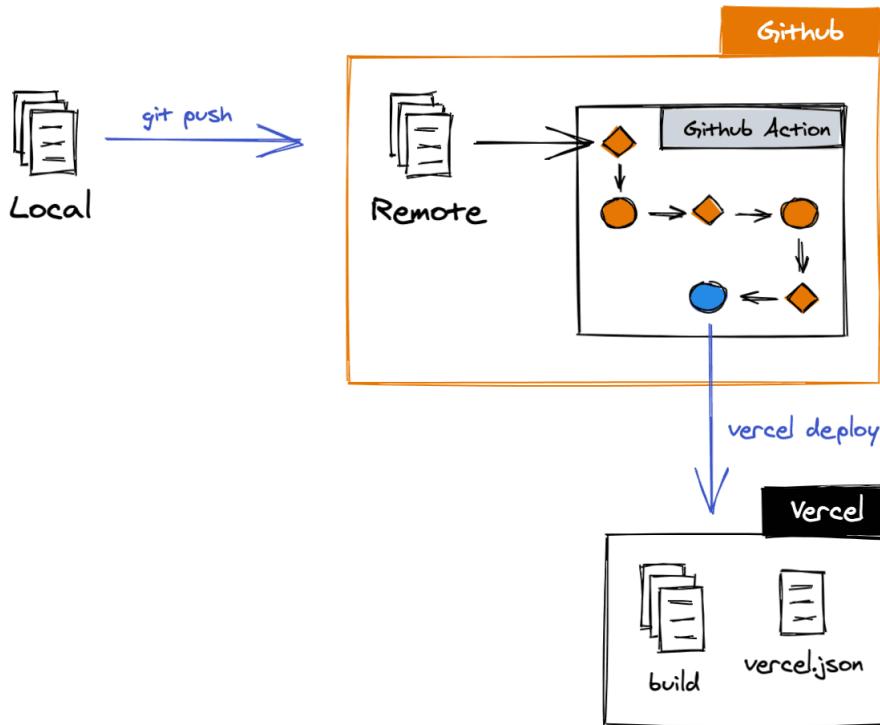


Figura 3.7: Flujo de trabajo DPS: Develop, Preview and Ship.

En la Figura 3.7 podemos observar el flujo de trabajo. Este se basa en que cada vez que se realiza una subida de código, Github Actions hace un despliegue en Vercel, siguiendo el **flujo de trabajo DPS** [42]:

- **Desarrollo:** Escribimos código en Next.js y subimos los cambios a una rama en Github.
- **Vista previa:** Vercel crea un despliegue de cada rama que estará disponible a través de una URL. Podemos compartir esta URL para recibir feedback.
- **Enviar a Producción:** Fusionamos la rama creada con la rama “main” para enviar a producción.

4

Descripción Informática

Para desarrollar una aplicación de calidad es necesario el uso de unas guías de estándares de calidad de código, un moderno stack tecnológico, así como buenas prácticas. A continuación se encuentra una detallada descripción informática incluyendo **tecnologías**, **implementación** y **pruebas**.

4.1. Stack Tecnológico

Después de una comparación de tecnologías actuales se llegó a la conclusión de que estas herramientas, frameworks y lenguajes eran los ideales para el desarrollo (ver Figura 4.1). Es necesario comprender cómo funcionan estas tecnologías para entender más tarde la implementación de la aplicación.

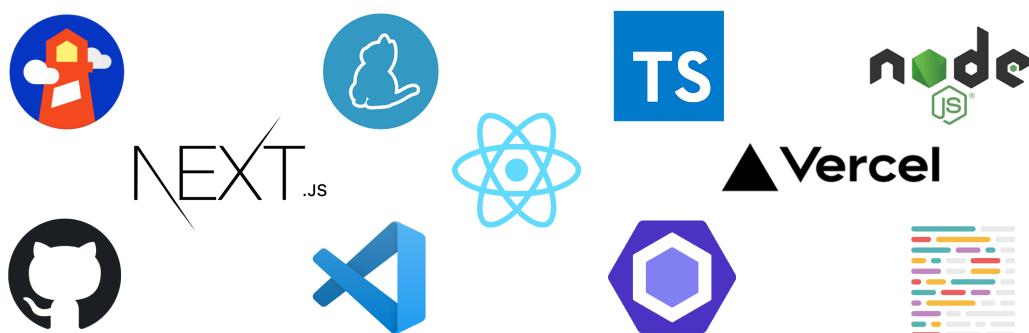


Figura 4.1: Stack Tecnológico.

4.1.1. Next.js

Creado por Vercel, es un framework creado sobre **Node.js** y basado en **React**. Permite, con una sola dependencia, tener todo configurado para crear una aplicación web de alto rendimiento de React usando **SWC** y **Webpack**. [43]

Estas son las características [44] que ofrece sin apenas configuración:

- **Sistema de enrutamiento basado en páginas** con soporte para rutas dinámicas: Cada página está asociada con una ruta basada en su nombre de archivo.
- **Pre-rendering**: El HTML para cada página es generado por adelantado, en lugar de que JavaScript en el cliente lo haga todo. Esto resulta en un mejor rendimiento y SEO. Cuando llegue el robot de Google recibirá el contenido ya renderizado, esto permitirá posicionar igual que una web estática.

Next.js tiene dos formas de pre-rendering:

- **Server-side Rendering**: El HTML se genera en cada solicitud.
- **Generación Estática**: El HTML se genera en el momento de la compilación y se reutilizará en cada solicitud. Esta forma es la utilizada por razones de rendimiento.
- **Code Splitting** para cargas de página más rápidas: La división automática del código en varios paquetes o componentes que luego se pueden cargar a medida y en paralelo.
- **Enrutamiento optimizado con prefetching** para SPA: La aplicación precarga los recursos mostrados en la página actual para que su acceso sea mucho más rápido al navegar.
- **Fast Refresh y HMR** (Hot Module Replacement) para el entorno de desarrollo: Permite actualizar todo tipo de módulos y componentes de React en tiempo de ejecución sin necesidad de un refresco completo. Lo que mejora la experiencia y velocidad del desarrollo.
- **Soporte integrado de CSS, Sass y CSS-in-JS**: Lo que permite abstraer CSS al nivel del componente, utilizando JavaScript para describir estilos de forma declarativa y mantenible.
- **Rutas API** para crear endpoints con Serverless Functions: Cualquier archivo dentro de la carpeta “pages/api” se asigna a “/api/*” y se tratará como un endpoint en lugar de una página.
- **Compatible con TypeScript** de forma predeterminada y con tipos integrados para las páginas y la API.

4.1.2. React

Es una biblioteca de **JavaScript** mantenida por Facebook, que permite crear interfaces de usuario mediante componentes interactivos y reutilizables [45].

Estos componentes se crean usando una sintaxis llamada JSX permitiendo escribir HTML y CSS dentro de objetos JavaScript y se combinan para crear componentes mayores hasta configurar una web completa.

Además, genera el DOM de forma dinámica, hace los cambios en un **DOM virtual** y después la compara con la versión actual del DOM. Aplica los cambios sólo al componente actualizado, evitando renderizar toda la página. Esto propicia una mejor experiencia de usuario, gran rendimiento y fluidez [46].

4.1.3. Vercel

Es una plataforma en la nube que permite **alojar sitios web** y servicios web que escalan automáticamente y no requieren supervisión [47].

Proporciona dominios personalizados, variables de entorno, HTTPS automático y una vista previa de cada rama de Github. Además, desplegar una aplicación **Next.js** tiene las siguientes ventajas:

- Las páginas que usan generación estática y assets se publican automáticamente desde el **Vercel Edge Network**, que es increíblemente rápido.
- Las páginas que usan pre-rendering y rutas API se convierten en **Serverless Functions**, permitiendo que el renderizado y las solicitudes puedan escalar.

Por último, ofrece **Vercel Analytics** [48] para medir el rendimiento de la web, recopilando las web vitals de los dispositivos reales que utilizan los usuarios.

4.1.4. TypeScript

Es un lenguaje de programación desarrollado y mantenido por Microsoft. Es un superconjunto de **JavaScript**, que añade **tipos estáticos** y objetos basados en clases [49].

Es un lenguaje más limpio, robusto y mantenible. Permite escribir código con menos errores, más sencillo, coherente y fácil de probar. Además, ayuda a implementar patrones de diseño e incrementa la agilidad en el refactoring.

Los navegadores no entienden TypeScript, por lo que es necesario transpilarlo a JavaScript antes de usarlo [50]. Al crear nuestra aplicación con **Nextjs** ya obtenemos transpilación y empaquetado automáticos con **SWC** y **Webpack**.

4.1.5. Node.js

Es un entorno de tiempo de ejecución de **JavaScript**. Orientado a eventos asíncronos, incluye todo lo necesario para ejecutar un programa JavaScript [51].

Permite utilizar un único lenguaje para el Backend y el FrontEnd. Además, cuenta con un repositorio de código abierto lleno de **librerías** útiles.

Debe estar instalado con **NPM** [52] o **Yarn** [53], dos gestores de paquetes para proyectos de Node.js. A veces, la **instalación de paquetes** con NPM no es lo suficiente consistente o rápida, dando incluso errores. Este es el motivo para usar Yarn, una alternativa construida por Facebook, Google, Exponent y Tilde [54].

4.1.6. VSCode

Es un **editor de código** optimizado para crear aplicaciones modernas. Incluye depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente y refactorización de código [55].

Además, cuenta con una biblioteca de **extensiones** para ser más eficiente programando. Desde extensiones de lenguajes de programación hasta herramientas para visualizar o estructurar el código.

Las extensiones más importantes para mejorar la experiencia de desarrollo y buenas prácticas de este proyecto fueron **ESLint** y **Prettier**. Configurar estas herramientas es una inversión que se hace una vez y sus beneficios notan durante todo el desarrollo.

ESLint

Es una herramienta de **análisis de código** para identificar patrones problemáticos que se puede instalar como extensión en VSCode [56].

Su función es analizar el código, detectar problemas y si puede:

- Corregir errores de sintaxis.
- Corregir código poco intuitivo o difícil de mantener.
- Evitar el uso de “malas prácticas”.
- Hacer uso de un estilo de código consistente.

Está diseñado para ser flexible y configurable por lo que añadimos ejecutarlo como parte del proceso de integración continua. Se pueden usar guías de estilo como Airbnb, Standard o Google. En este caso la configuración es la **guía de estilo** recomendada de ESLint con reglas para TypeScript y React.

Prettier

Es una herramienta para **formatear el código** que se puede instalar como extensión en VSCode [57].

Su función es analizar el código y aplicar un estilo consistente, dando formato según sus propias reglas a cosas como:

- La longitud máxima de línea.
- El tipo de identación.
- Los saltos de línea.
- El orden de los imports.

Su objetivo es acabar con los debates sobre el **estilo del código**. Para ello, analiza el código y lo da formato cada vez que se guarda el archivo. Para usarlo junto con ESLint hay que configurar este último para que use Prettier.

En definitiva, Prettier se usa para problemas de formato de código y ESLint para problemas de calidad de código.

4.2. Detalles de Implementación

La principal documentación con la que contamos es el estilo de nomenclatura, la coherencia a la hora de nombrar cosas y una arquitectura general coherente. Esta debe emanar de una serie de patrones que se repiten a lo largo del proyecto, tanto a nivel arquitectónico como de diseño. Esta información reside en el código: en el nombre de los componentes, de las variables y de las funciones. Este código se puede encontrar en el siguiente **repositorio de Github**:

<https://github.com/alberttoggoca/EarFit>.

Para dejar reflejados estos detalles en la memoria se explicarán todas las partes de las que se compone esta implementación:

- [Arquitectura](#).
- [Estructura de Archivos](#).
- [Tipos](#).
- [Librerías](#).
- [Servicios](#).
- [Hooks](#).
- [Componentes](#).

No hará falta estar familiarizado con los conceptos propios de React para haber entendido esta documentación al acabar.

4.2.1. Arquitectura

La arquitectura indica la **estructura, funcionamiento e interacción** entre las diferentes partes de las que esta compuesto el software. En este diagrama se muestra el diseño de más alto nivel de la estructura del sistema.

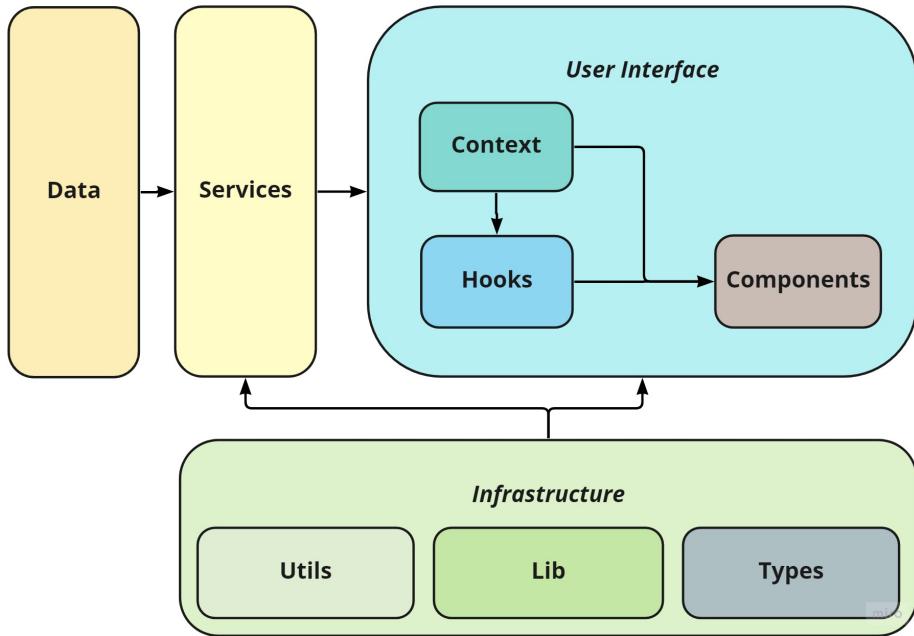


Figura 4.2: Arquitectura de la aplicación.

En la Figura 4.2 se ilustra la arquitectura de la aplicación. Ésta está formada por **varias capas desacopladas**: infraestructura, datos, servicios e interfaz de usuario. Estas capas interactúan entre sí de la siguiente manera:

- **Infraestructura**: Se compone de funciones, librerías y tipos de Typescript que apoyan la lógica de los servicios y de la interfaz.
- **Servicios**: Proveen los datos y alguna lógica independiente a la capa de interfaz de usuario.
- **Contexto y Hooks**: Usan los servicios y se hacen cargo de crear y manejar los estados necesarios para los componentes.
- **Páginas y Componentes**: Hacen uso del contexto y los Hooks, y renderizan la interfaz de usuario.

4.2.2. Estructura de Archivos

Next.js tiene algunos archivos y **directorios especiales** [44]. Como mínimo, se necesita una carpeta “pages” con un archivo “index.js”. Además, la carpeta “public” tiene una función específica en Next.js. Aparte de estos archivos y directorios, todo vale [58].

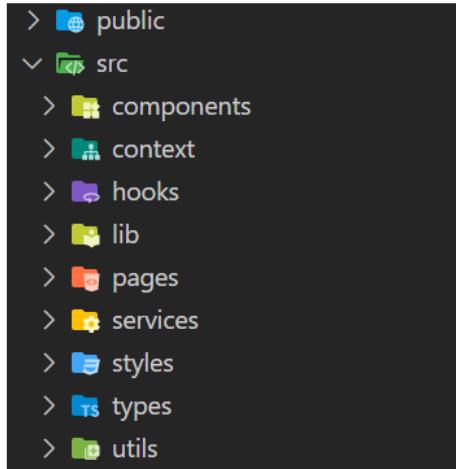


Figura 4.3: Estructura de Archivos.

En la Figura 4.3 se puede observar la estructura de archivos. En este caso se han separado las **carpetas por conceptos** o intereses. Cada carpeta contiene archivos similares basandose en la arquitectura:

- **Pages:** Aquí se encuentran todas las páginas de la aplicación. Cada página está asociada con una ruta basada en su nombre de archivo. Por ejemplo: en “pages/notes” se exporta un componente de React que será accesible en la URL “/notes”.

El archivo “index.js” es la página que se representa cuando el usuario visita la ruta raíz de la aplicación.

Aparte, para tener un Layout persistente está el archivo “_app.tsx”. En este archivo se encuentran los metadatos de la aplicación y se envuelve la aplicación con el Layout.

Por otro lado, en el archivo “_document.tsx” se actualizan las etiquetas “html” y “body”.

- **Public:** Aquí se encuentran los iconos, imágenes, Service Worker, Manifest y archivos fuente de sonido en formato MIDI.js. En Next.js el código puede hacer referencia a los archivos estáticos dentro de una “public” en el directorio raíz a partir de la URL base (/).

- **Components:** Aquí se encuentran todos los componentes de la aplicación agrupados por carpetas. El modelo React nos permite deconstruir una página en una serie de componentes. Muchos de estos componentes a menudo se reutilizan entre páginas.
- **Context:** Aquí se encuentra el contexto de la aplicación. El Context está diseñado para compartir estados que pueden considerarse “globales” para un árbol de componentes en React.
- **Hooks:** Aquí se encuentran todos los Custom Hooks creados para extraer la lógica de los componentes en funciones reutilizables. Los Hooks son una nueva característica en React 16.8. Estos permiten usar el estado y otras características de React sin escribir una clase.
- **Lib:** Aquí se encuentra “soundfont-wrapper”, una librería personalizada para la aplicación. Para hacer uso de la librería “soundfont-player”, que se encarga de cargar los archivos fuente de sonido y usarlos mediante la Web Audio API. Se creó esta librería envoltorio (wrapper library) para refinar la complejidad de su interfaz y simplificar su uso.
- **Services:** Aquí se encuentran los servicios que serán consumidos desde los Hooks. Los servicios proporcionan los datos y un conjunto de métodos responsables de alguna lógica específica independiente a la interfaz.
- **Styles:** Aquí se encuentran algunos estilos css globales. Aunque en general se hace uso de “react-bootstrap” junto con “bootswatch” para los estilos de la aplicación.
- **Types:** Aquí se encuentran todos los tipos usados en los demás archivos de la aplicación. Para usar Typescript todos los archivos “.js” se convierten en “.ts” y “.jsx” en “.tsx”.
- **Utils:** Aquí encuentran los archivos con métodos genéricos para ser usados en cualquier parte de la aplicación.

Archivos de configuración: En el directorio raíz del proyecto se pueden encontrar los archivos “.eslintrc.js”, “.gitignore”, “.prettier.js”, “next-env.d.ts”, “next.config.js”, “tsconfig.json”, “package.json” y “yarn.lock”. También se encuentra un archivo “README.md” con una breve presentación y explicación de la aplicación.

4.2.3. Infraestructura

Tipos

A continuación, se explican los tipos de **TypeScript** creados para la implementación. Estos tipos son una manera de documentar el código y evitar errores que Javascript no proporciona. Ayudan a leer y entender el código de una forma más efectiva.

```
export type Instrument = {
  displayName: string;
  emoji: string;
  instrumentName: InstrumentName;
  notePlayer: NotePlayer;
  isLocal: boolean;
};
```

Figura 4.4: Tipo Instrument.

- **Instrument**: Este tipo define a los instrumentos en la aplicación. Cada instrumento tiene un nombre, un emoji, un nombre identificador del instrumento, un “notePlayer” para tocar notas y un booleano que especifica si se tiene el archivo de sonido en una carpeta local. (Ver Figura 4.4).
- **InstrumentName**: Este tipo define el nombre identificador del instrumento. Esta definido por la librería “Soundfont-player-js” como una lista de instrumentos disponibles. Sirve para buscar el archivo MIDI de dicho instrumento y crear su “notePlayer”.

```
type NotePlayer = {
  play: (
    noteName: string,
    when?: number,
    opts?: {
      duration?: number;
      gain?: number;
      attack?: number;
      decay?: number;
      sustain?: number;
      release?: number;
      adsr?: [number, number, number, number];
      loop?: boolean;
    }
  ) => Player;
  stop: (noteName?: string) => void;
  schedule: (when?: number, events?: any[]) => Player;
  instrumentName: InstrumentName;
};
```

Figura 4.5: Tipo Noteplayer.

- **Noteplayer:** Este tipo proporciona funciones para reproducir los sonidos de dicho instrumento. Este tipo es proporcionado por la librería “Soundfont-wrapper”. (Ver Figura 4.5).



```
TypeScript
export type VariantExercise = 'notes' | 'intervals' | 'scales';
```

Figura 4.6: Tipo VariantExercise.

- **VariantExercise:** Este tipo define la tres variantes del ejercicio: “notes”, “intervals” y “scales”. En la aplicación se tratan todas las páginas de ejercicio como variantes del mismo. Es decir, todas son el mismo ejercicio pero con valores distintos. Lo que nos permite reutilizar toda la lógica como veremos en los Hooks. (Ver Figura 4.6).



```
export type Answer = {
  id: string;
  notes: string[];
  displayName: string;
};

export type IsSelectable = {isSelected: boolean};
export type VariantColor = 'success' | 'secondary' | 'danger';
export type WithColor = {color: VariantColor};

export type SelectableAnswer = Answer & IsSelectable;
export type SelectableAnswerWithColor = SelectableAnswer & WithColor;
```

Figura 4.7: Tipo Answer.

Cada variante de ejercicio utiliza los tipos Answer, SelectableAnswer y SelectableAnswerWithColor. (Ver Figura 4.7).

- **Answer:** Este tipo define a una respuesta del ejercicio. Cada respuesta tiene un id, un nombre para mostrar y un array de notas para ser reproducidas. Es el tipo base del ejercicio.
- **SelectableAnswer:** Este tipo añade propiedades por intersección al tipo Answer, necesarias para implementar los botones de selección renderizados por el componente “AnswerToggles”. La propiedad “isSelected” especifica si la respuesta está disponible para preguntar.
- **SelectableAnswerWithColor:** Este tipo añade propiedades por intersección al tipo SelectableAnswer, necesarias para implementar los botones de respuesta renderizados por el componente “AnswerButtons”. La propiedad “color” especifica el color que debe tomar el botón de respuesta.

Librerías

A continuación, se explican los paquetes de **Node.js** o librerías usadas en la implementación. Todas estas librerías se pueden encontrar definidas con sus versiones en el archivo “package.json” o en la carpeta “lib” en el caso de “soundfont-wrapper”.

- **Tonaljs**: Es una librería de teoría musical. Contiene funciones para manipular elementos musicales como notas, intervalos y escalas. Se tratan de abstracciones, no de música o sonido reales [59].
- **Soundfont-player**: Sirve para cargar archivos MIDI.js y reproducir sonidos utilizando la WebAudio API [60]. Estos archivos contienen los sonidos de los instrumentos codificados. No hace falta contar con estos archivos en local ya que en tal caso los buscaría en el siguiente repositorio: **midi-js-soundfonts** [61].
- **Soundfont-wrapper**: Es una librería personalizada para refinar la complejidad de “soundfont-player” y simplificar su uso. Proporciona la función “getSoundfontInstrument” que devuelve un “notePlayer” con funciones para tocar notas. Se encarga de crear el “audioContext” y gestionar los nodos de audio.
- **React-piano**: Proporciona un teclado de piano interactivo para React. No implementa la reproducción de audio de cada nota, por lo que se debe implementar aparte y pasar por Props dos funciones: playNote y stopNote [62].
- **React-use-measure**: Se usa para conseguir que “react-piano” sea responsive. Lo que hace es obtener la referencia de ancho del componente padre para usarla en el renderizado del piano [63].
- **React-bootstrap**: Es la librería de estilos CSS usada. Cada componente de React-Bootstrap es un verdadero componente de React, sin dependencias innecesarias como jQuery [64].
- **Next-pwa**: Es un plugin que permite registrar y generar un Service Worker para convertir una aplicación web en una aplicación web progresiva [65].

Utils

En el archivo “**arrayUtils.ts**” está el método “getRandomItem” que recibe un array y devuelve un elemento al azar. Usado en el Hook “useAnswer” para seleccionar la respuesta para el ejercicio. El objetivo era ir añadiendo aquí todas funciones complementarias que se fuesen necesitando. Al final resultó ser una.

4.2.4. Datos y Servicios

A continuación, se explican los servicios creados para la aplicación. Por comodidad y tamaño, los datos que proporcionan los servicios se encuentran en su mismo archivo. Ahorrándonos los pertinentes imports y ganando sencillez a la hora de leer el código.

- **instrumentService.ts:** Este servicio provee una función “getInstruments” que devuelve un array de instrumentos de tipo “Instrument[]”. Usa la librería “soundfont-wrapper” para obtener los “notePlayer” para cada instrumento definido en la constante “InstrumentData”. De tal manera que añadir o quitar un instrumento es tan sencillo como añadirlo a “InstrumentData”.
- **noteService.ts:** Este servicio provee una función “getNotes” que recibe una escala de tipo “string” y devuelve un array de respuestas de tipo “Answer[]” correspondiente a la variante de ejercicio “notes”. Utiliza la librería Tonal para obtener las notas según la escala.
- **intervalService.ts:** Este servicio provee una función “getIntervals” que devuelve un array de respuestas de tipo “Answer[]” correspondiente a la variante de ejercicio “intervals”. En este caso, como hay varias combinaciones de notas para un mismo intervalo, proporciona una función “calcInterval” que recibe un intervalo y lo devuelve con una de estas combinaciones. Utiliza la librería Tonal para calcular las notas de los intervalos.
- **scaleService.ts:** Este servicio provee una función “getScales” que devuelve un array de respuestas de tipo “Answer[]” correspondiente a la variante de ejercicio “scales”. Utiliza la librería Tonal para obtener las escalas.
- **menuService.ts:** Este servicio provee una función “getMenuInfo” que devuelve un array con la información de cada ítem del menú principal.

4.2.5. Hooks y Context

Según React: “Las clases presentan problemas para las herramientas de hoy en día. Las clases no minifican bien, y hacen que la recarga en caliente sea confusa y poco fiable. Los componentes complejos se vuelven difíciles de entender. Es difícil reutilizar la lógica de estado entre componentes. Y las clases confunden tanto a las personas como a las máquinas.”

Para resolver estos problemas, Hooks permite usar **React sin clases**. Aunque se necesita un cambio de mentalidad para empezar a “pensar en Hooks”. Conceptualmente, los componentes de React siempre han estado más cerca de las funciones. Es por esto por lo que se usan funciones en lugar de clases para guardar el estado de la aplicación. [66]

Hooks

Los Hooks sirven para extraer **lógica de estado** de tal forma que un componente pueda ser reusado independientemente. Cuando se tiene lógica que debe ser utilizada por varios componentes, se puede extraer a un Custom Hook. Esto permite dividir un componente en funciones más pequeñas. [67]

El estado de cada componente es independiente. Cada llamada a un Hook tiene un estado completamente aislado. Además, como los Hooks son funciones, se puede pasar información entre ellos. [68]

A continuación, se explican los **Custom Hooks** creados para la aplicación, cada uno cubre un caso de uso. Hay una relación entre el nombre de los Hooks y de los componentes, por lo que se pueden asociar fácilmente más adelante.

- **useExercise.tsx:** Este Hook recibe la variante de ejercicio de tipo “VariantExercise”. Devuelve las respuestas de tipo “Answer[]” asociadas a dicha variante. Obtiene estas respuestas gracias a los servicios “noteService”, “intervalService” y “scaleService”.
- **useAnswerToggles.tsx:** Este Hook recibe las respuestas de tipo “Answer[]”. Devuelve un nuevo array de respuestas disponibles de tipo “AnswerToggles[]” con tres seleccionadas, la función “updateIsSelected” para seleccionar una respuesta y la función “selectAllOrThree” para alternar entre seleccionarlas todas o dejar tres seleccionadas.
- **useAnswer.tsx:** Este Hook recibe la variante de ejercicio de tipo “VariantExercise” y las respuestas disponibles de tipo “AnswerToggles[]”. Calcula y devuelve la respuesta actual de tipo “Answer” dependiendo de la variante, la función “setNewAnswer” para calcular una nueva respuesta y la función “isCorrectAnswer” para comprobar si una respuesta es correcta.
- **useAnswerButtons.tsx:** Este Hook recibe las respuestas disponibles de tipo “AnswerToggles[]” y las funciones “setNewAnswer”, “isCorrectAnswer” y “playAnswer”. Devuelve un nuevo array compuesto sólo por las respuestas seleccionadas de tipo “answerButtons[]” cada una con color, la función “handleAnswerButtonClick” que se hace cargo de la lógica cuando se pulsa sobre un botón de respuesta y la racha actual de respuestas correctas de tipo “string” haciendo uso de “useStreak”.
- **useStreak.tsx:** Este Hook devuelve el contador actual de respuestas acertadas consecutivamente de tipo “number”, la función “incrementStreak” para incrementar el contador y la función “clearStreak” para resetar el contador.

- **useDirectionSelector.tsx:** Este Hook devuelve la dirección en la que se reproducirán las notas de la respuesta de tipo “boolean” y la función “changeDirection” para actualizarla.
- **usePlayButton.tsx:** Este Hook recibe la variante de ejercicio de tipo “VariantExercise”, la respuesta actual de tipo “Answer” y la dirección de tipo “boolean”. Devuelve la función “playAnswer” para reproducir las notas de la respuesta actual, en su dirección y a la velocidad correcta según la variante.
- **useScaleDropdown.tsx:** Este Hook devuelve la lista de escalas de tipo “string[]”, la escala seleccionada de tipo “string”, para obtener las respuestas de la variante del ejercicio de notas, y la función “setNewSelectedScale” para actualizarla.
- **usePiano.tsx:** Este Hook recibe opcionalmente un rango de notas de tipo “string”. Devuelve el rango de notas, los atajos de teclado y las funciones “stopNote” y “playNote” necesarias para el componente “Piano”.

Por último, dentro de estos se usan también dos Hooks propios de React:

- **useState:** Permite añadir un estado local de React a un componente. Este Hook recibe el estado inicial y devuelve una pareja de valores: el valor de estado actual y una función que permite actualizarlo. [69]
- **useEffect:** Permite llevar a cabo efectos secundarios en los componentes como peticiones de datos, suscripciones o actualizaciones manuales del DOM. Estos efectos pueden afectar a otros componentes y no se pueden hacer durante el renderizado. Este Hook recibe una función “efecto” que se ejecutará después de renderizar el DOM. [70]

Context

En una aplicación típica de React, los datos se pasan de padre a hijo a través de Props. Esta forma puede resultar incómoda para Props que son necesarias para muchos componentes. Context es una forma de **compartir valores** como estos entre componentes sin tener que pasar Props a través de cada nivel del árbol. [71]

- **EarfitContext.tsx:** Este contexto provee todos los instrumentos de tipo “Instrument[]”, el instrumento seleccionado de tipo “Instrument” y la función “selectInstrument” para actualizarlo. Obtiene estos instrumentos gracias al servicio “instrumentService”.

Este contexto se encuentra envolviendo la aplicación en “_app.tsx” e implementa el Hook “useInstrumentContext” para ser usado por diferentes Hooks y componentes.

4.2.6. Componentes y Pages

Los componentes permiten separar la interfaz de usuario en piezas independientes y reutilizables. Aceptan entradas llamadas **Props** y retornan **elementos de React**. También pueden tener estado, que es similar a las Props, pero es privado. (Ver Figura 4.8).

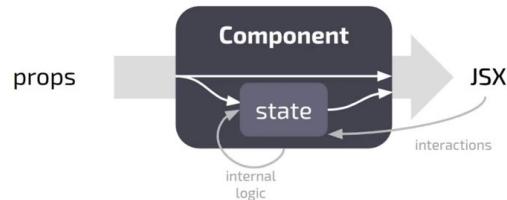


Figura 4.8: Diagrama de un componente React.

Los componentes deben actuar como **funciones puras**, es decir, no tratar de cambiar sus entradas, y devolver siempre el mismo resultado para las mismas entradas. En consecuencia, las Props deben ser de solo lectura. [72]

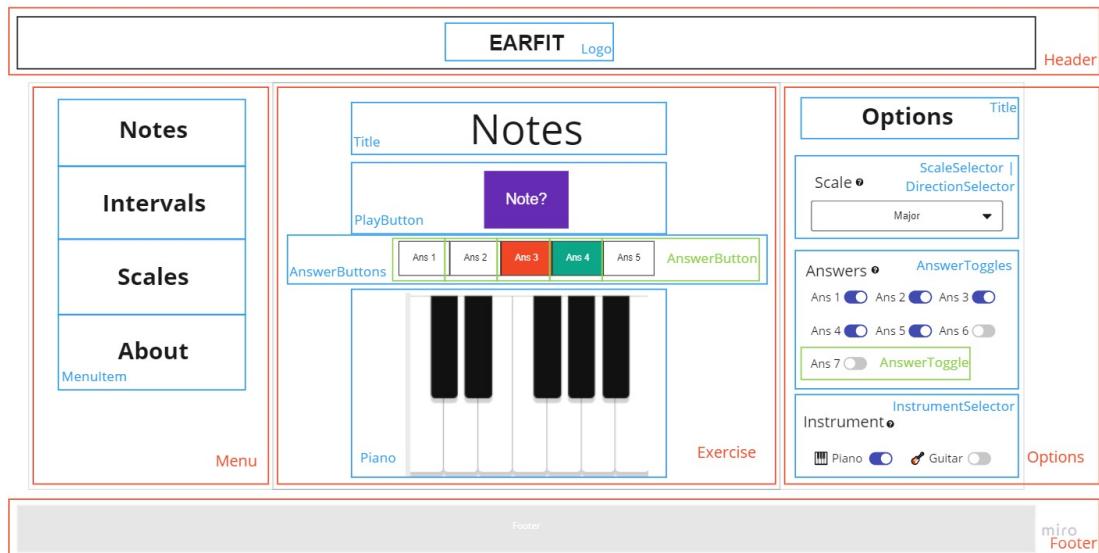


Figura 4.9: Jerarquía de Componentes de la aplicación.

En la Figura 4.9 se puede observar todos los componentes de los que se compone la aplicación y que son **reutilizados entre páginas**.

Las páginas no son más que componentes de React exportados desde un archivo .tsx en el directorio de páginas. Sólo que cada una está asociada con una ruta basada en su nombre de archivo.

Pages

```

import { Exercise, Menu, Options, PageLayout } from 'components';
import { useAnswer, useAnswerButtons, useAnswerToggles, useDirectionSelector,
useExercise, usePlayButton } from 'hooks';

export default function Intervals(): JSX.Element {
  const { direction, changeDirection } = useDirectionSelector();
  const { answers } = useExercise('intervals');
  const { answerToggles, updateIsSelected, selectAllOrThree } = useAnswerToggles(answers);
  const { answer, setNewAnswer, isCorrectAnswer } = useAnswer('intervals', answerToggles);
  const { playAnswer } = usePlayButton('intervals', answer, direction);
  const { answerButtons, handleAnswerButtonClick, streak } = useAnswerButtons(answerToggles,
    isCorrectAnswer,
    setNewAnswer,
    playAnswer
  );

  return (
    <PageLayout
      leftCol={<Menu />}
      rightCol={
        <Options
          direction={direction}
          handleDirectionChange={changeDirection}
          answerToggles={answerToggles}
          handleAnswerToggleAllChange={selectAllOrThree}
          handleAnswerTogglesChange={updateIsSelected}
        />
      }
    >
      <Exercise
        title="Intervals"
        playButtonLabel="Interval?"
        handlePlayButtonClick={playAnswer}
        answerButtons={answerButtons}
        handleAnswerButtonClick={handleAnswerButtonClick}
        streak={streak}
      />
    </PageLayout>
  );
}

```

Figura 4.10: Código de la página de intervalos.

En la Figura 4.10 se observa el código de una de las páginas de la aplicación. Explicando su lógica se puede entender el comportamiento de todas las demás:

1. Se importan los Hooks y Componentes de la página.
2. Se hacen las llamadas a los Hooks que aportan los datos y Callbacks.
3. Se le pasan estos datos y Callbacks a los componentes por Props.

Components

```

import { AnswerButton } from 'components/Exercise/AnswerButtons/AnswerButton';
import { ButtonGroup } from 'react-bootstrap';
import { SelectableAnswerWithColor } from 'types';

interface Props {
    answerButtons: SelectableAnswerWithColor[];
    handleAnswerButtonClick: (answerButton: SelectableAnswerWithColor) => void;
}

export const AnswerButtons = ({ answerButtons, handleAnswerButtonClick }: Props): JSX.Element => {
    return (
        <ButtonGroup className="btn-group btn-group-toggle d-flex justify-content-center" data-toggle="buttons">
            <div>
                {answerButtons.map((answerButton) => (
                    <AnswerButton
                        key={answerButton.id}
                        answerButton={answerButton}
                        handleAnswerButtonClick={handleAnswerButtonClick}
                    />
                ))}
            </div>
        </ButtonGroup>
    );
};

```

Figura 4.11: Código del componente AnswerButtons.

En la Figura 4.10 se observa el código de uno de los componentes de la aplicación. Explicando su lógica se puede entender el comportamiento de todos los demás:

1. Le llegan los datos y Callbacks a través de Props.
2. Renderiza el componente usando estos datos.
3. El usuario interactúa con el componente.
4. El componente hace la llamada a su función Callback.
5. Esta función actualizará los datos que le llegan por Props.
6. Al llegar nuevas Props al componente se vuelve a renderizar mostrando los cambios.

Como los componentes solo pueden actualizar su propio estado, el componente superior necesita pasar funciones (Callbacks) al hijo que este ejecutará cada vez que el estado deba actualizarse. Esto se llama flujo unidireccional.

4.2.7. Progressive Web App

Para conseguir que una aplicación web sea confiable, instalable y se comporte como una app nativa en ordenador, móvil y tablet se puede convertir en PWA o Progressive Web App. Con ella, el usuario puede **instalar la app** en su dispositivo y seguir consultándola, aunque haya perdido la conexión. [73]

Las PWA deben tener las siguientes características:

- **Una conexión segura:** La aplicación web se debe servir a través de una red segura HTTPS.
- **Cargue sin conexión:** Esto significa que las aplicaciones web progresivas requieren de un **Service Worker**.
- **Información sobre la aplicación:** como nombre, autor, ícono y descripción en un documento JSON llamado **Manifest**.
- **Un ícono** de al menos 144x144 de tamaño grande en formato PNG.

Siendo una PWA, al visitarla cualquier navegador ya nos da la opción de instalarla en nuestro dispositivo. En iOS, la opción se llama “añadir a pantalla de inicio” desde el menú “compartir” de **Safari** y en **Chrome** aparece un ícono en la barra de búsqueda (ver Figura 4.12).

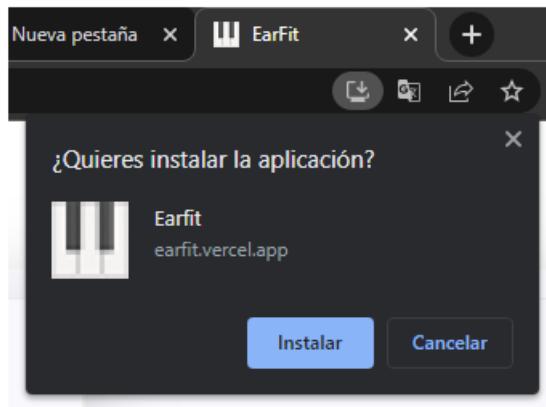


Figura 4.12: Instalar la aplicación como PWA desde Chrome.

Al instalarla se crea un **ícono en la pantalla de inicio** y ya está lista para usarse. Puede usarse sin conexión y en cualquier tamaño de ventana ya que al ser responsiva se adapta. Para desinstalarla se puede hacer como una aplicación normal o pulsando en los tres puntos de la ventana nos da la opción.

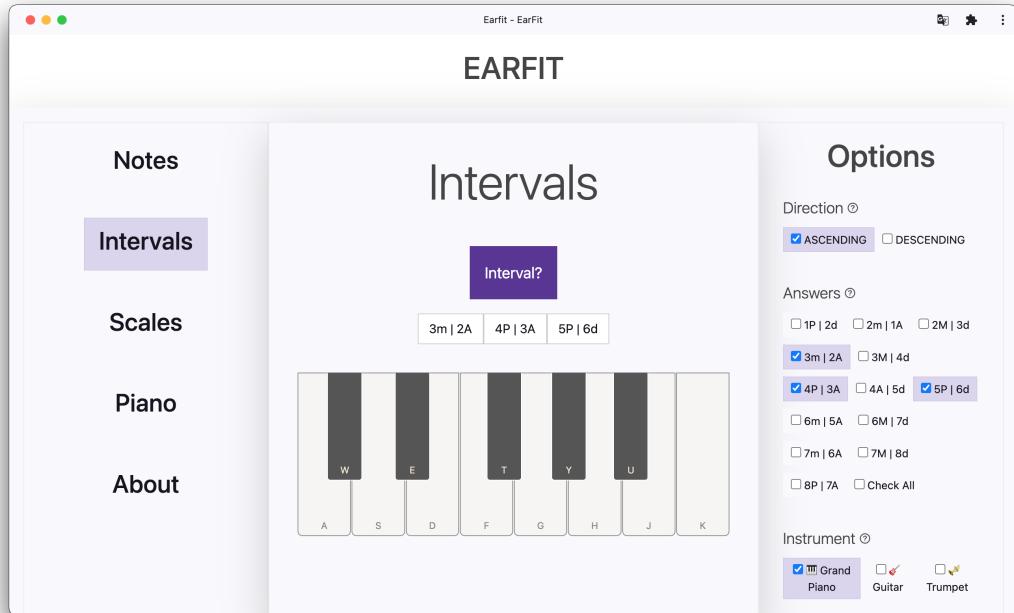


Figura 4.13: Captura de la página de intervalos.

En la Figura 4.13 aparece una captura de la aplicación en MacOS. Hay capturas de toda la aplicación en el apéndice [D. Capturas de la Aplicación](#).

Service Worker

Se encuentra en la carpeta “public” con el nombre de “**sw.js**”. Es un script que permite interceptar y controlar cómo un navegador maneja las solicitudes de red y el almacenamiento en caché. Este es un archivo autogenerado por el plugin “next-pwa”, que permite registrar y generar un Service Worker optimizado. El archivo “**workbox.js**” también está relacionado con este plugin. [74]

Manifest

Se encuentra en la carpeta “public” con el nombre de “**manifest.json**”. Es un archivo que controla cómo se muestra la aplicación al usuario y garantiza que las PWA sean detectables. Describe el nombre de la aplicación, la URL de inicio, los iconos y todos los demás detalles necesarios para transformar el sitio web en un formato similar al de una aplicación. [75]

IOS Safari no implementa manifiestos, por lo que la mayoría de los metadatos de PWA se pueden definir a través de extensiones específicas de Apple para las metaetiquetas. Estos están definidos en el archivo personalizado “**_app.tsx**”.

4.2.8. Buenas Prácticas

Las buenas prácticas son conjunto de técnicas, principios y metodologías para que el software se vuelva fácil, rápido y seguro de desarrollar, mantener y desplegar. En este caso se han usado **Guías de Estilo de Código**, **Principios de Clean Code** y **Otras Buenas Prácticas**.

Guía de Estilo de Código

Se han usado las reglas recomendadas de ESLint y Prettier. Las reglas se pueden encontrar en el archivo “`.eslintrc.js`” y “`.prettierrc.js`”. Además se han añadido las siguientes reglas personales:

- Tipos de TypeScript para los componentes en lugar de Props.
- No es necesario importar React cuando se usa Next.js.
- Desactivar la regla para enlaces predeterminada, no es compatible con los componentes “`<Link />`” de Next.js.
- Eliminar variables sin usar.
- Las funciones tienen que devolver siempre un tipo.
- Uso de comas finales.
- Indentación a 2 espacios.
- Uso de comillas simples para strings.
- Longitud de línea máximo 120 caracteres.
- No usar tabulaciones.
- Finales de linea automaticos (“LF” to “CRLF”).

Principios Clean Code

Los principios de Clean Code [76] permiten obtener código limpio, reutilizable, escalable y con mayor cambiabilidad. Algunos de los principios adaptados React tenidos presentes se encuentran en el apéndice **E. Clean Code**.

Otras Buenas Prácticas

- Crear un archivo “`.gitignore`”: Para excluir archivos generados y sensibles.
- Visualizar el flujo de trabajo: Con Scrum Board y User Story Map.
- Circuitos de retroalimentación: Haciendo reuniones periódicas con el tutor y recogiendo feedback de los usuarios constantemente.
- Fomentar la visibilidad con el proceso bien definido, publicado y promovido: Usando las herramientas Notion, Miro y Trello.
- Mejorar colaborando, usando modelos y el método científico: Aplicando Lean Startup y otras metodologías.

4.3. Testing

El testing de software o **software QA**, es un proceso para verificar y validar la funcionalidad de un programa o una aplicación de software. Su propósito principal es asegurar que la aplicación desarrollada cumpla con los estándares y se ofrezca al cliente un producto de calidad [77]. Para asegurarlo es importante realizar tanto **Pruebas Funcionales** como **Pruebas No Funcionales**.

4.3.1. Pruebas Funcionales

El objetivo de estas pruebas es evaluar las suposiciones hechas en las especificación de historias de usuario y diseño para asegurar que el software se comporta según lo definido.

Dado que es una aplicación pequeña y el desarrollo de **pruebas automáticas** tiene asociado un coste de implementación y mantenimiento, hay que llegar a un compromiso de valor aportado (defectos que no llegan a producción) frente al coste de desarrollarlas. Debido a esto, las pruebas se realizaron de forma manual.

En un futuro se pretende añadir este tipo de pruebas automáticas al proceso de integración continua. Para así asegurar un porcentaje de cobertura de código y su fiabilidad.

Cada prueba que se ha realizado pretende abordar un objetivo en concreto donde se han intentado abordar todos los test cases o casos de prueba posibles. Estas pruebas se clasifican en: **Pruebas Unitarias**, **Pruebas de Componentes**, **Pruebas de Integración** y **Pruebas End to End (E2E)**.

Pruebas Unitarias

Estas pruebas permiten probar que los elementos más fundamentales del software como objetos, funciones, eventos, funcionan como se espera. Su objetivo es probar que para cada elemento las mismas entradas proporcionen siempre las mismas salidas, verificando también si son correctas.

En este caso según se iba desarrollando el código se comprobaba que cada Hook, función y estado, devolvía el resultado esperado y funcionaba correctamente antes de darle uso en el código. Realizando pequeños tests por consola y usando las herramientas para desarrolladores de Google Chrome cuando era necesario.

Pruebas de Componentes

En estas pruebas se considera el SUT como cada componente de React. Con el objetivo de probar si la funcionalidad descrita en las historias de usuario se corresponde con su comportamiento. Permiten identificar fallos en los componentes que incluyen varias funciones o elementos internos para que se puedan corregir antes de que el software llegue a producción.

En este tipo de pruebas se validaron los criterios de aceptación descritos en las historias de usuario. Probando cada componente según se desarrollaba. Cualquier problema detectado se reparó de forma inmediata antes de continuar con el desarrollo.

Pruebas de Integración

El objetivo es probar la interacción entre componentes al integrar uno nuevo en la aplicación. Permiten probar el comportamiento y posibles fallos en la interacción entre los componentes entre sí, y demás elementos del software.

En este caso, al integrar un nuevo componente en la aplicación, se probó si su funcionamiento y el de los componentes con los que interactúa seguía siendo el adecuado. De esta manera en cada momento se tiene una visión general del estado del proyecto alertándonos de regresiones en el código.

Pruebas End to End (E2E)

Estas pruebas se realizaron sobre el sistema una vez desplegado desde el punto de vista de los usuarios. En estas pruebas el SUT es el sistema completo de principio a fin. Se realizan con el objetivo de verificar si el software se comporta como se espera según las historias de usuario.

Estas pruebas se han realizado con los siguientes objetivos:

- **Pruebas de sanidad (sanity check):** Probar las funcionalidades básicas del sistema, usadas después de un despliegue.
- **Pruebas de instalación y desinstalación:** Probar las funcionalidades básicas del sistema, después de instalarse como Progressive Web App.

En este caso cada historia de usuario se probó para asegurarse de que se siguen obteniendo los resultados esperados. Estas pruebas fueron realizadas en Windows, MacOS, Android y Iphone usando Chrome y Safari.

4.3.2. Pruebas No Funcionales

El objetivo de estas pruebas es evaluar el rendimiento, accesibilidad, buenas prácticas, SEO y si es progresiva, para la optimización del software y asegurar su calidad. Siendo el SUT el sistema una vez desplegado. Estas pruebas se ejecutan de manera automática mediante [Google Lighthouse](#) y [Vercel Analytics](#).

Google Lighthouse

Es una herramienta para medir la calidad de las páginas web. Incluye la capacidad de probar PWA para el cumplimiento de estándares, mejores prácticas. Permite detectar errores y oportunidades de optimización.



Figura 4.14: Puntuaciones de Lighthouse.

En la Figura 4.14 se observan los resultados de su análisis de la aplicación. Estos constan de las siguientes cinco categorías:

- **Performance:** En este apartado, Lighthouse no está optimizado y da falsas mediciones para aplicaciones [Nextjs](#). Sin embargo, como hemos desplegado nuestra aplicación en Vercel, podemos hacer uso de [Vercel Analytics](#), que además nos aporta algunas ventajas como veremos más adelante.
- **Accessibility:** En este ámbito se comprueba si la página es fácil de usar para personas con limitaciones físicas. Se comprueba si los elementos importantes, como los botones y los enlaces, se describen de forma clara.
- **Best Practices:** En este área se analizan los aspectos de seguridad de la web. Se comprueban las tecnologías de codificación como TLS, si los recursos, bibliotecas JavaScript, bases de datos y APIs son seguras.
- **SEO:** En este ámbito se analiza el nivel de visibilidad de la web en diferentes buscadores. Se comprueba la idoneidad de la web para terminales móviles, es decir, si las etiquetas y los metadatos se han optimizado.
- **Progressive Web Apps:** Esta es la función principal de Google Lighthouse. Analiza si la página web funciona según lo previsto. Comprueba si todos los elementos y contenidos dinámicos se representan correctamente, si la página registra un Service Worker y si está disponible la función offline.

Vercel Analytics

Mientras que otras herramientas como Lighthouse estiman la experiencia de usuario ejecutando una simulación en el ordenador del desarrollador, el **Real Experience Score** de Vercel se calcula utilizando datos reales recopilados de los dispositivos de los usuarios reales. Gracias a esto proporciona una calificación real de cómo los usuarios realmente experimentan la aplicación. [48]

Esto permite un flujo continuo de mediciones de rendimiento, a lo largo del tiempo, integrado en el flujo de trabajo de desarrollo. Con lo que se puede correlacionar fácilmente los cambios en el rendimiento con las nuevas implementaciones.

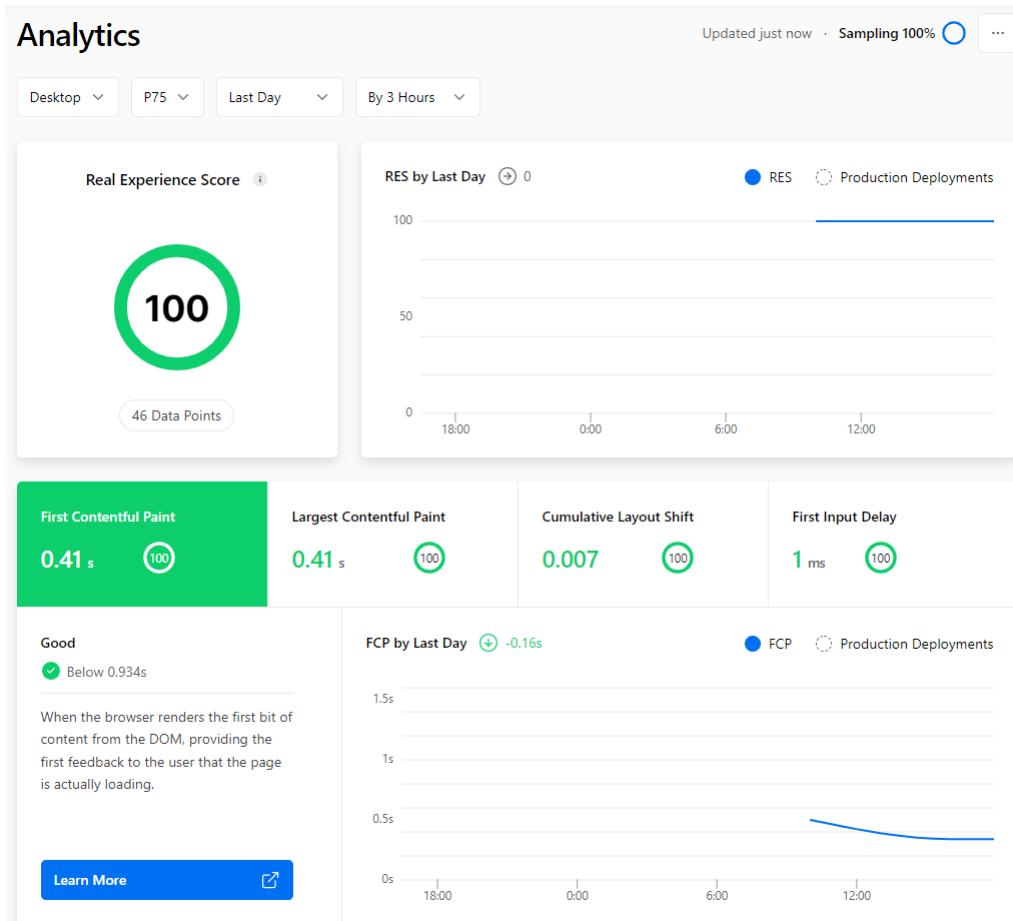


Figura 4.15: Vista de análisis de las Web Vitals.

En la Figura 4.15 se observa la vista de análisis de la aplicación. Esta se calcula en base a las **Web Vitals**, que son una colección de métricas establecidas por Google [78] y el Web Performance Working Group [79]. En ella pueden distinguir las siguientes cuatro categorías:

- **First Contentful Paint (FCP):** Mide la velocidad de carga, o cuando se muestra el primer contenido de la página.
- **Largest Contentful Paint (LCP):** Mide la velocidad de carga percibida, o cuando se puede visualizar todo el contenido de la página.
- **Cumulative Layout Shift (CLS):** Mide la estabilidad visual, o cuánto se mueven los elementos después de mostrarse al usuario. Por ejemplo, cuando un botón se mueve porque una imagen se cargó tarde, eso es CLS.
- **First Input Delay (FID):** Mide la capacidad de respuesta de la página, o cuánto tiempo esperan los usuarios para ver la reacción de su primera interacción con la página. Por ejemplo, la cantidad de tiempo entre que se hace click en “Aregar al carrito” y el incremento del número de artículos en el carrito es FID.

5

Conclusiones

En este capítulo se detallan las conclusiones derivadas del trabajo, los objetivos alcanzados y la propuesta para posibles trabajos futuros. Por último, habrá una pequeña opinión personal que dará cierre a esta memoria.

5.1. Objetivos Alcanzados

El presente trabajo se proponía el desarrollo de una herramienta que permitiera a los músicos desarrollar su oído musical mediante la práctica de ejercicios de **entrenamiento auditivo**.

Para lograr una solución que aportase valor, se definieron las necesidades del usuario, se establecieron una serie de hipótesis que solucionasen sus problemas y se priorizaron las funcionalidades más relevantes para diseñar un prototipo de un producto mínimo viable aplicando **Design Thinking**.

Este producto mínimo viable se desarrolló aplicando **Lean Startup**, en un proceso de aprendizaje validado siguiendo un circuito iterativo de crear, medir, aprender. Los requisitos podían cambiar al aprender del impacto en los usuarios de una versión del software y tener que pivotar.

Por eso se aplicó **Scrum** para planificar y gestionar el proyecto. Un User Story Map definía el viaje o casos de uso del usuario en el producto y las releases de cada versión. Las historias de usuario definían los requisitos del sistema y se encontraban en el Scrum Board que servía para visualizar y gestionar el trabajo

de cada sprint. Al final de cada sprint se producía un incremento en el software generando una nueva versión que era directamente entregada al usuario para obtener su feedback.

Usar prácticas **DevOps** agilizó este proceso. Aplicando integración continua en Github siguiendo el modelo Gitflow. Y despliegue continuo, utilizando Github Actions para tener un despliegue por cada rama de desarrollo en Vercel siguiendo el flujo de trabajo Develop, Preview and Ship.

Para su implementación se usaron diferentes tecnologías como **Next.js**, **React**, **TypeScript** y **Nodejs**. Con esto se logró convertir la solución en una aplicación web progresiva o PWA, haciendo que sea confiable e instalable como una aplicación nativa en ordenador, móvil y tablet.

Además, durante su desarrollo en **VSCODE** se aplicaron buenas prácticas como guías de estilo de código y principios de Clean Code. Mientras, se testeaba su rendimiento usando **Google Lighthouse** y **Vercel Analytics** asegurando una buena experiencia de usuario.

Al final, se logró desarrollar el MVP propuesto cumpliendo con los objetivos que se habían establecido en este trabajo. Aportando una solución única para el entrenamiento auditivo, con diferentes tipos de ejercicios personalizables y varios instrumentos.

El resultado final de esta aplicación se puede encontrar en el apéndice **D. Capturas de la Aplicación**. También se puede probar la aplicación escaneando el código QR (ver Figura 5.1) y en el siguiente enlace:

<https://earfit.vercel.app/>

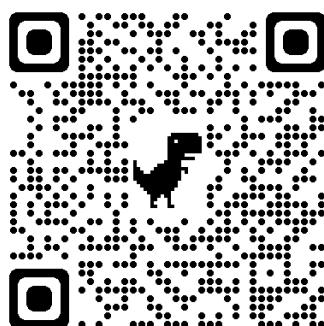


Figura 5.1: Código QR de Earfit.

5.2. Trabajos futuros

Esta memoria recoge todo lo necesario para seguir con el trabajo aún si no se conocen las tecnologías y metodologías empleadas. Por lo que, se podrá considerar mejorar la aplicación o implementar algunas de las funcionalidades que no tuvieron cabida en el desarrollo.

Algunas de las **posibles extensiones** para la aplicación, que no se llevaron a cabo por falta de tiempo son:

- **Funcionalidades** que no se establecieron como prioridad en la etapa de diseño. Éstas se pueden encontrar en la Figura 2.3 de la técnica de MosCow o en la Figura 2.2 del Mindmap de Design Thinking: ejercicio de acordes, ritmos, progresiones, modo nocturno y varios idiomas.
- **Cobertura de código** y test automáticos. Debido al coste de aprender, implementar y mantener estos test automáticos de los componentes, se optó por realizarlos manuales. Se podrían añadir estos test al flujo de desarrollo en un futuro.
- **Software libre vs Negocio**. De momento esta aplicación es gratuita. Se podría plantear alguna forma de monetización, si se considerase que la aplicación pudiera tener algún hueco en el mercado, aunque este no era su propósito original.

5.3. Opinión Personal

En general este trabajo ha sido demasiado ambicioso por mi parte. El tener que aprender prácticamente todo desde cero me ha servido mucho para crecer pero también me ha limitado a la hora de desarrollar la aplicación.

El tiempo de desarrollo de la aplicación no fue muy elevado si no se cuenta la fase de aprendizaje de todas las metodologías y tecnologías empleadas, acabando el desarrollo según lo previsto. En cambio, lo que más tiempo llevó fue realizar esta memoria explicando todos estos conceptos que no se dan en la carrera y me siento en la obligación de explicar, por si no se entiende la manera ágil en la que se han hecho las cosas.

En conclusión, creo que el resultado de la aplicación es bueno y la forma moderna en que se ha desarrollado es acorde a la actualidad. Espero que la memoria haya sido lo suficiente completa y se haya podido entender todo sin conocimientos previos.

Bibliografía

- [1] J. Plana, “Entrena tus oídos,” *LANDR*, 2018. [Online]. Available: <https://blog.landr.com/es/entrena-tus-oidos-las-8-mejores-aplicaciones-para-mejorar-tus-habilidades-de-escucha/>
- [2] J. D. Lopera, “Introducción al entrenamiento auditivo,” 2014. [Online]. Available: <https://www.ladomicilio.com/es/cursos-de-musica/entrenamiento-auditivo/introduccion-entrenamiento-auditivo>
- [3] C. T. Flix, “El oído musical,” *Research Gate*, 2003. [Online]. Available: https://www.researchgate.net/publication/46772334_El_ocio_musical
- [4] ToneGym, “Ear training for musicians.” [Online]. Available: <https://www.tonegym.co/>
- [5] ——, “Create a free account.” [Online]. Available: <https://www.tonegym.co/user/registration>
- [6] ——, “Interval ear training.” [Online]. Available: <https://www.tonegym.co/exercise/intervals>
- [7] TonedEar, “Ear training practice.” [Online]. Available: <https://tonedear.com/>
- [8] ——, “Intervals quiz.” [Online]. Available: <https://tonedear.com/ear-training/intervals>
- [9] EarMaster, “Conviértete en un mejor músico.” [Online]. Available: <https://www.earmaster.com/es/>
- [10] ——, “Tienda online de earmaster.” [Online]. Available: <https://www.earmaster.com/es/shop.html>
- [11] D. P. Gámir, “Cómo se aplican las metodologías design-thinking, lean, agile y growth hacking de manera eficiente,” *Emprende a Conciencia*, 2017. [Online]. Available: <https://www.emprendeaconciencia.com/blog/como-se-aplican-las-metodologias-design-thinking-lean-agile-y-growth-hacking>
- [12] ContentLab, “Lean startup vs. design thinking: Ventajas y desventajas,” *Gestión*, 2019. [Online]. Available: <https://gestion.pe/especial/businessstyle/innovacion/lean-startup-vs-design-thinking-ventajas-y-desventajas-noticia-1994455>
- [13] D. L. de Innovación, “Design thinking en español.” [Online]. Available: <https://www.designthinking.es/inicio/index.php>
- [14] D. A. Carrasco, “Despierta a tu creativo en 5 etapas,” *ENAE Business School*, 2020. [Online]. Available: <https://www.enae.es/blog/despierta-tu-creativo-en-5-etapas>
- [15] S. Gibbons, “Cognitive maps, mind maps, and concept maps: Definitions,” *NNgroup*, 2019. [Online]. Available: <https://www.nngroup.com/articles/cognitive-mind-concept/>
- [16] ——, “5 prioritization methods in ux roadmapping,” *NNgroup*, 2021. [Online]. Available: <https://www.nngroup.com/articles/prioritization-methods/>

- [17] C. Busquets, “Atomic design: Qué es y qué ventajas tiene,” 2019. [Online]. Available: <https://www.uifrommars.com/atomic-design-ventajas/>
- [18] QualityDevs, “Qué es «mobile first» y porqué es importante,” 2019. [Online]. Available: <https://www.qualitydevs.com/2019/02/07/que-es-mobile-first-y-por-que-es-importante/>
- [19] E. Ries, *El Método de Lean Startup*. Deusto, 2012.
- [20] Y. P. Alexander Osterwalder, *Diseñando la Propuesta de Valor*. Deusto, 2015.
- [21] M. B. Kent Beck, “Manifiesto Ágil.” [Online]. Available: <https://agilemanifesto.org/iso/es/manifesto.html>
- [22] S. G. Sotomayor, “Las metodologías Ágiles más utilizadas y sus ventajas dentro de la empresa,” *IEBS*, 2021. [Online]. Available: <https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/>
- [23] C. Drumond, “Scrum: Aprende a utilizar scrum con lo mejor de él,” Atlassian. [Online]. Available: <https://www.atlassian.com/es/agile/scrum>
- [24] J. S. Ken Schwaber, *The Definitive Guide to Scrum: The Rules of the Game*. Scrum.org, 2020. [Online]. Available: <https://scrumguides.org/scrum-guide.html>
- [25] G. H. Sola, “¿cómo hacer un user story mapping?” Scrum.org. [Online]. Available: <https://www.scrum.org/resources/blog/como-hacer-un-user-story-mapping>
- [26] A. Kaley, “Mapping user stories in agile,” *NNgroup*, 2021. [Online]. Available: <https://www.nngroup.com/articles/user-story-mapping/>
- [27] A. Alonso, “5 pasos para construir nuestro user story mapping,” *Medium*, 2019. [Online]. Available: <https://adrianalonso.dev.medium.com/5-pasos-para-construir-nuestro-user-story-mapping-2674be8aeb60>
- [28] P. Galiana, “Scrum board: Qué es y cómo hacer uno,” *IEBS*, 2021. [Online]. Available: <https://www.iebschool.com/blog/scrum-board-que-es-y-como-hacer-uno/>
- [29] R. G. Tamarit, “Product backlog y sprint backlog,” *Muy Agile*, 2019. [Online]. Available: <https://muyagile.com/product-backlog-y-sprint-backlog/>
- [30] S. Mexico, “Escribiendo historias de usuario,” *Scrum.mx*, 2018. [Online]. Available: <https://scrum.mx/informe/historias-de-usuario>
- [31] M. Rehkopf, “Historias de usuario con ejemplos y plantilla,” Atlassian. [Online]. Available: <https://www.atlassian.com/es/agile/project-management/user-stories>
- [32] B. A. M. Friend, “Las mejores historias de usuario invest y smart.” [Online]. Available: <https://beagilemyfriend.com/historias-de-usuario-invest-smart/>
- [33] Azure, “¿qué es devops?” Microsoft. [Online]. Available: <https://azure.microsoft.com/es-es/overview/what-is-devops/#devops-overview>
- [34] RedHat, “El concepto de devops,” RedHat. [Online]. Available: <https://www.redhat.com/es/topics/devops>
- [35] J. Garzas, “Aprende a implantar integración continua desde cero,” 2014. [Online]. Available: <https://www.javiergarzas.com/2014/08/implantar-integracion-continua.html>
- [36] GitHub, *GitHub Docs*, GitHub. [Online]. Available: <https://docs.github.com/es/get-started>
- [37] B. S. Scott Chacon, *Pro Git*, Git. [Online]. Available: <https://git-scm.com/book/en/v2>
- [38] Atlassian, *Flujo de Trabajo de Gitflow*, Atlassian. [Online]. Available: <https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow>

BIBLIOGRAFÍA

- [39] I. C. Education, “Despliegue continuo,” IBM, 2019. [Online]. Available: <https://www.ibm.com/es-es/cloud/learn/continuous-deployment>
- [40] GitHub, *GitHub Actions*, GitHub. [Online]. Available: <https://docs.github.com/es/actions>
- [41] Vercel, *Next.js on Vercel*, Vercel. [Online]. Available: <https://vercel.com/docs/concepts/next.js/overview>
- [42] Nextjs, *Deploying Your Next.js App*, Vercel. [Online]. Available: <https://nextjs.org/learn/basics/deploying-nextjs-app>
- [43] ——, *Introduction a Nextjs*, Vercel. [Online]. Available: https://nextjs.org/learn/foundations/about-nextjs?utm_source=next-site&utm_medium=nav-cta&utm_campaign=next-website
- [44] ——, *Documentación de Nextjs*, Vercel. [Online]. Available: <https://nextjs.org/docs>
- [45] React, *Documentación de React*, Meta. [Online]. Available: <https://es.reactjs.org/docs/getting-started.html>
- [46] Drauta, “¿qué es react y para qué sirve?” [Online]. Available: <https://www.drauta.com/que-es-react-y-para-que-sirve>
- [47] Vercel, *Documentación de Vercel*, Vercel. [Online]. Available: <https://vercel.com/docs>
- [48] ——, *Vercel Analytics*, Vercel. [Online]. Available: <https://vercel.com/docs/concepts/analytics>
- [49] Typescript, *Documentación de TypeScript*, Microsoft. [Online]. Available: <https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html>
- [50] J. L. Chacón, “TypeScript: Qué es, diferencias con javascript y por qué aprenderlo,” *Profile*, 2021. [Online]. Available: <https://profile.es/blog/que-es-typescript-vs-javascript>
- [51] Nodejs, *Acerca de Node.js*, OpenJS Foundation. [Online]. Available: <https://nodejs.org/es/about/>
- [52] NPM, *About NPM*, Npm Inc. [Online]. Available: <https://docs.npmjs.com/about-npm>
- [53] Yarn, *Introduction to Yarn*, Meta. [Online]. Available: <https://yarnpkg.com/getting-started>
- [54] J. Salazar, “Yarn vs. npm, todo lo que necesitas saber,” *Tekzup*, 2016. [Online]. Available: <https://tekzup.com/yarn-vs-npm-lo-necesitas-saber/>
- [55] VSCode, *Documentación de VSCode*, Microsoft. [Online]. Available: <https://code.visualstudio.com/docs>
- [56] ESLint, *Documentación de ESLint*, OpenJS Foundation. [Online]. Available: <https://eslint.org/docs/about/>
- [57] Prettier, *Documentación de Prettier*. [Online]. Available: <https://prettier.io/docs/en/index.html>
- [58] React, *Estructura de Archivos*, Meta. [Online]. Available: <https://es.reactjs.org/docs/faq-structure.html>
- [59] Tonaljs, *A Functional Music Theory Library for Javascript*. [Online]. Available: <https://github.com/tonaljs/tonal>
- [60] M. W. Docs, *Web Audio API*, Mozilla Corporation. [Online]. Available: https://developer.mozilla.org/es/docs/Web/API/Web_Audio_API
- [61] D. G. B., *Quick Soundfont Loader and Player for Browser*. [Online]. Available: <https://github.com/danigb/soundfont-player>

- [62] K. Qi, *An Interactive Piano Keyboard for React*. [Online]. Available: <https://github.com/kevinsqi/react-piano>
- [63] Poimandres, *Utility to Measure View Bounds*. [Online]. Available: <https://github.com/pmnndrs/react-use-measure>
- [64] R. Bootstrap, *The Most Popular Front-End Framework*. [Online]. Available: <https://react-bootstrap.github.io/>
- [65] W. Wang, *Zero Config PWA Plugin for Next.js*. [Online]. Available: <https://github.com/shadowwalker/next-pwa>
- [66] React, *Presentando Hooks*, Meta. [Online]. Available: <https://es.reactjs.org/docs/hooks-intro.html>
- [67] ——, *Construyendo tus Propios Hooks*, Meta. [Online]. Available: <https://es.reactjs.org/docs/hooks-custom.html>
- [68] ——, *Un Vistazo a los Hooks*, Meta. [Online]. Available: <https://es.reactjs.org/docs/hooks-overview.html>
- [69] ——, *Usando el Hook de Estado*, Meta. [Online]. Available: <https://es.reactjs.org/docs/hooks-state.html>
- [70] ——, *Usando el Hook de Efecto*, Meta. [Online]. Available: <https://es.reactjs.org/docs/hooks-effect.html>
- [71] ——, *Context*, Meta. [Online]. Available: <https://es.reactjs.org/docs/context.html>
- [72] ——, *Componentes y Propiedades*, Meta. [Online]. Available: <https://es.reactjs.org/docs/components-and-props.html>
- [73] M. W. Docs, *Aplicaciones Web Progresivas*, Mozilla Corporation. [Online]. Available: https://developer.mozilla.org/es/docs/Web/Progressive_web_apps
- [74] ——, *Service Worker API*, Mozilla Corporation. [Online]. Available: https://developer.mozilla.org/es/docs/Web/API/Service_Worker_API
- [75] ——, *Web App Manifest*, Mozilla Corporation. [Online]. Available: <https://developer.mozilla.org/es/docs/Web/Manifest>
- [76] R. C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson, 2008.
- [77] M. M. Canelo, “Qué es el testing de software,” *Profile*, 2021. [Online]. Available: <https://profile.es/blog/que-es-el-testing-de-software/>
- [78] Google, *Web Vitals*, Google. [Online]. Available: <https://web.dev/vitals/>
- [79] G. de Trabajo de Rendimiento Web, *Application Performance of User Agent Features and APIs*, w3. [Online]. Available: <https://www.w3.org/webperf/>

Apéndices

A

Proceso de Creación Detallado

Este apéndice recoge una descripción detallada de lo que no se ha podido incluir en la memoria principal sobre el proceso de creación de la solución. El objetivo principal es complementar el apartado de Lean Startup explicando su proceso de crear, medir, aprender. También incluye un mayor desarrollo para las primeras etapas de Design Thinking. Se espera que estos dos apartados sirvan para entender mejor cómo se han implementado y para qué han servido estas dos metodologías.

Este apéndice consta de tres secciones:

- Un resumen del proceso de creación general a modo de recordatorio.
- Una explicación más detalla de las etapas de Design Thinking.
- Una explicación detalla del proceso de Lean Startup, que no se ha explicado en la memoria.

A.1. Resumen

La idea parte de la necesidad de jóvenes músicos que quieren aprender a tocar un instrumento. Una parte fundamental del aprendizaje consiste en entrenar el oído, que sin los medios adecuados puede resultar difícil.

Junto con mi tutor Manuel Rubio, que es un apasionado de la música, intentamos dar forma a esta solución. Tuvimos varias reuniones en las que él, como músico, me explicaba las dificultades por las que pasan a la hora de entrenar el oído. Una vez tenidas claras sus necesidades era hora de crear una solución que aportase valor.

Para idear nuestra propuesta de valor se utilizaron prácticas de Design Thinking. También, el método de aprendizaje validado de Lean Startup que se basa en el Circuito de feedback de

información: crear, medir, aprender. Y por último, Metodologías Ágiles como Scrum para la gestión y desarrollo del software.

Todo este proceso se basa en crear una hipótesis, diseñar un experimento para testear esa hipótesis, llevar a cabo el experimento, reunir datos, reflexionar y ver si validan o rechazan la hipótesis.

A.2. Design Thinking

Lo primero de todo es definir nuestra visión. ¿Cuáles son nuestros objetivos? El objetivo es conocer qué quieren los potenciales usuarios. Pero nos encontramos bajo unas condiciones de incertidumbre extrema. Nuestra tarea es diseñar para crear un nuevo producto o servicio bajo estas condiciones de incertidumbre extrema.

A.2.1. Empatizar

Una forma de conocer que quieren los usuarios podría ser entrevistando a los potenciales clientes para estar seguros de que el problema que queremos solventar existe.

En este caso el cliente más cercano es el propio Manuel Rubio. Así no diseñamos sobre hipótesis propias no validadas sino que lo hacemos teniendo en cuenta lo que es necesario para el usuario.

Tenemos varias reuniones por Teams en las que, como músico, me explica las dificultades por las que pasan a la hora de entrenar el oído y la importancia que el entrenamiento auditivo tiene.

Además, para entender mejor la situación necesito de un proceso de investigación. Internet es un inmenso campo de búsqueda de información. Por eso, paralelamente a las entrevistas busco información adicional que me permita entender mejor a nuestro usuario, el problema y cómo solucionarlo. Dado que mis conocimientos sobre música son básicos necesito de un estudio profundo de la teoría musical.

Una vez entendido al usuario y sus necesidades, podemos pasar a definir el problema.

A.2.2. Definir

El problema principal que se encuentran estos jóvenes músicos es la dificultad para entrenar el oído sumado a la dificultad para entrar en un conservatorio. Si quieren aprender a tocar por su cuenta no tienen los recursos necesarios para llevar a cabo este entrenamiento auditivo que Manuel nos ha explicado que es tan importante.

A.2.3. Idear

Vamos a intentar construir una solución que aporte valor. Para ello realizamos un pequeño MindMap donde nos enfocamos en los principales problemas y las posibles soluciones que podríamos llevar a cabo.

Después de varias iteraciones concluimos que una posible forma de solucionar estos problemas es mediante una aplicación web con ejercicios de entrenamiento auditivo. Esta es nuestra hipótesis.

A.2.4. Prototipar

Ahora toca diseñar un producto mínimo viable que pueda crear para testear esta hipótesis y obtener resultados. Este MVP no debe ser perfecto, lo que queremos es aprender lo más rápido posible. Añadir características a nuestro producto que todavía ni sabemos cómo va a ser aceptado es una perdida de tiempo.

Para este objetivo, la técnica MoSCoW nos permitió establecer las prioridades del proyecto. Teniendo nuestras limitaciones: poco tiempo para el desarrollo, aprender conceptos musicales y aprender tecnologías que no conocemos.

Luego, una vez establecidas las prioridades del proyecto pasamos a visualizarlas en forma de Sketches en papel, y más tarde realizamos el Prototipo teniendo en cuenta la filosofía de diseño Mobile First y Atomic Design. Tratamos de realizar un diseño simple pero efectivo basado en tres columnas.

A.2.5. Testear

En esta fase, es importante que entendamos que no estamos vendiendo. Se trata de aprender del feedback del usuario para hacer posteriormente una nueva versión mejorada de nuestra solución.

Después de haber testeado los primeros diseños con el usuario, haber hecho correcciones y haber validado el prototipo, pasamos a desarrollar la solución bajo el paraguas de Metodologías Ágiles siguiendo el método de Lean Startup.

A.3. Lean Startup

A.3.1. Crear

Para crear esta solución se decidió utilizar Scrum, prácticas de DevOps y diversas tecnologías. La aplicación se fue creando y testeando en un proceso iterativo incremental, generando siempre una versión que aportase valor. Esta parte es en la que se centra la memoria principal.

A.3.2. Medir

Es hora de llevar a cabo el experimento, lo más común es poner los usuarios delante del producto y recoger así información sobre su comportamiento. Recoges los datos y reflexionas sobre ellos para pivotar, es decir, seguir adelante o cambiar de dirección. Lo importante es qué tan rápido puedes desarrollar tus experimentos para hacer evolucionar la aplicación.

Nuestro objetivo es saber si el usuario realmente quiere lo que se está ofreciendo. Para ello,

se siguió la técnica del Conserje, empezando con un sólo usuario para luego escalar y hacer pruebas con más usuarios.

Cada versión se testeó con el tutor recogiendo feedback y realizando los ajustes pertinentes. Más tarde, se empezó a testear también con conocidos, en concreto dos estudiantes de conservatorio y una persona que está empezando a tocar.

Las métricas que se tuvieron en cuenta son: Engagement y Tiempo en el producto por usuario. También se tuvo en cuenta la retención (el usuario lo usa nuevamente) y la referencia (el usuario comparte el producto con sus amigos).

Además, se utilizaron test A/B: enseñando dos versiones diferentes del producto al mismo tiempo para tomar decisiones acertadas. Aprovechando las características que nos ofrecía Vercel de despliegue por rama de desarrollo.

A.3.3. Aprender

Lo que pudimos aprender en este continuo proceso de crear, medir, aprender fue lo siguiente:

- De los **test con el tutor** pudimos corregir lo siguiente:

En un principio no se incluían todos los intervalos existentes, sólo los de notas naturales. Un test con el tutor nos hizo darnos cuenta de que faltaban las notas alteradas.

En consecuencia, también pudimos corregir el nombre de estos, ya que descubrimos que dos intervalos con diferente nombre pueden sonar igual. Esto es debido a que los intervalos se nombran no sólo por la distancia de sus notas sino también por cómo están escritas en el pentagrama. Por ejemplo: entre Do y Do' hay 1 semitono y entre Do y Reb también hay 1 semitono, la misma distancia, suenan igual, pero son intervalos distintos. Lo que técnicamente se conoce como enarmonía.

Por último, se decidió añadir también un sonido al acertar. Se aprovechó el sonido de la misma respuesta para que así los usuarios puedan interiorizarla.

- De los **test con más usuarios** pudimos mejorar lo siguiente:

En un principio se mostraban todas las opciones disponibles, lo que hacía el ejercicio muy difícil. Por lo que, se ajustó la dificultad inicial dejando sólo tres opciones al azar.

Además, descubrimos de que resultaba engoroso una vez seleccionadas varias opciones volver a deseleccionarlas por lo que se añadió un botón que seleccionase y deselectionase todas de golpe.

Por último, descubrimos que había un problema de compatibilidad con Iphone que hacía inservible la aplicación, el cual pudimos corregir rápido. Y se ajustó el volumen de sonido, ya que al principio no era lo suficientemente alto.

- De los **test A/B** aprendimos lo siguiente:

Se mostraron dos versiones, una con piano en los ejercicios y otras sin él. Lo que pudimos observar es que las versiones con el piano generaban mayor engagement y retención por parte de los usuarios. Además, los usuarios lo utilizaban para tener notas de referencia, lo que podía ayudar a la obtención de la respuesta correcta. Esto resultó ser una incorporación muy útil sobretodo en el ejercicio de notas.

A.3.4. Conclusión

Cómo se puede observar gracias a este proceso iterativo de crear, medir, aprender pudimos corregir fallos ágilmente y añadir funcionalidades que ni nos habíamos planteado en un primer momento gracias al feedback de los propios usuarios.

Este proceso es necesario para mantenerse ágiles mientras crecemos. No invertir demasiado en grandes mejoras, sino hacer lotes de pequeñas mejoras más a menudo para aprender más. Lo que lleva a un círculo de interacción más rápido ya que puedes detectar problemas de calidad y tener mayor feedback sin tener que esperar a que el trabajo este hecho lo que propicia a que haya menos trabajo que rehacer.

Una vez pasado por todo este proceso iterativo, el resultado es un MVP listo para sacar al mercado y que se debe seguir mejorando. Los Early Adopters, los primeros en utilizar el producto no son muy exigentes con la calidad pero esto no es así para los nuevos clientes que vendrán más tarde. Por lo que habría que seguir un proceso de innovación sostenida, que se basaría en ir incrementando mejoras al producto ya existente.

B

Diseños del Prototipo

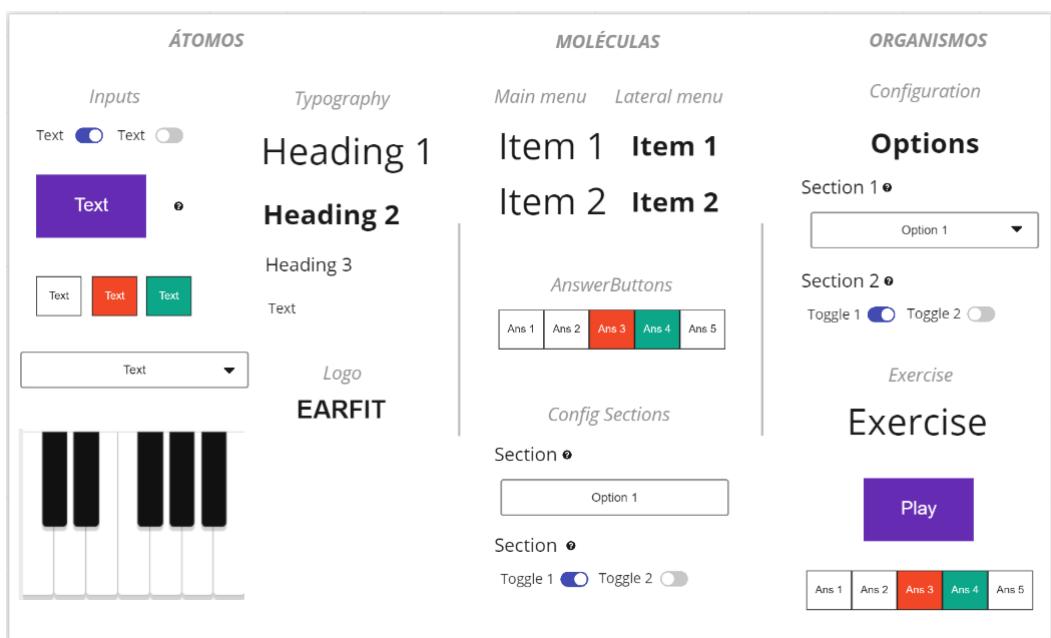


Figura B.1: Elementos IU de Atomic Design.

B.1. Prototipo Pantallas Pequeñas

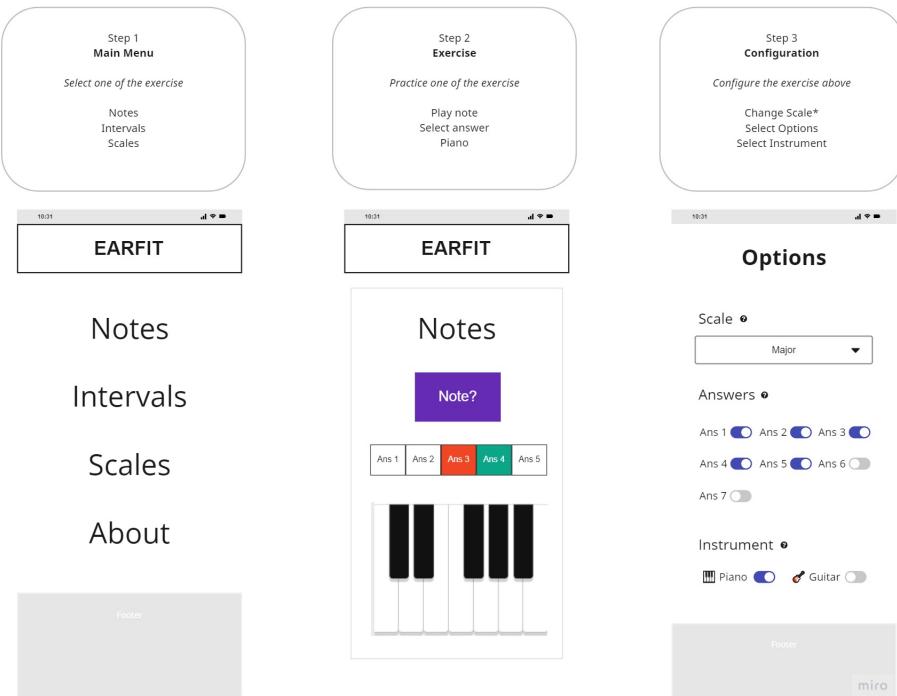


Figura B.2: Prototipo de la Página de Inicio y de Notas (Pantallas Pequeñas).

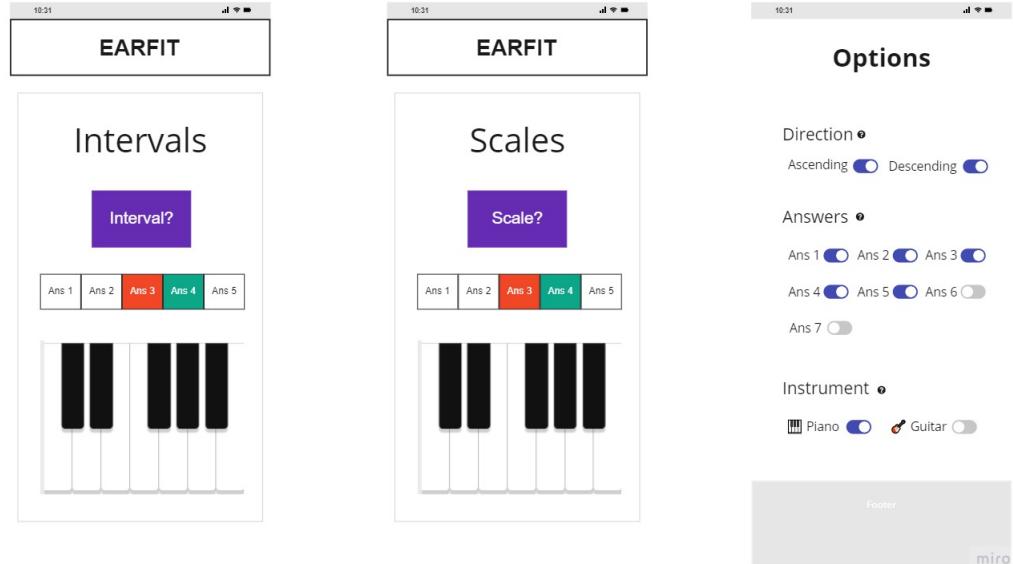


Figura B.3: Prototipo de la Página de Intervalos y de Escalas (Pantallas Pequeñas).

B.2. Prototipo Pantallas Medias

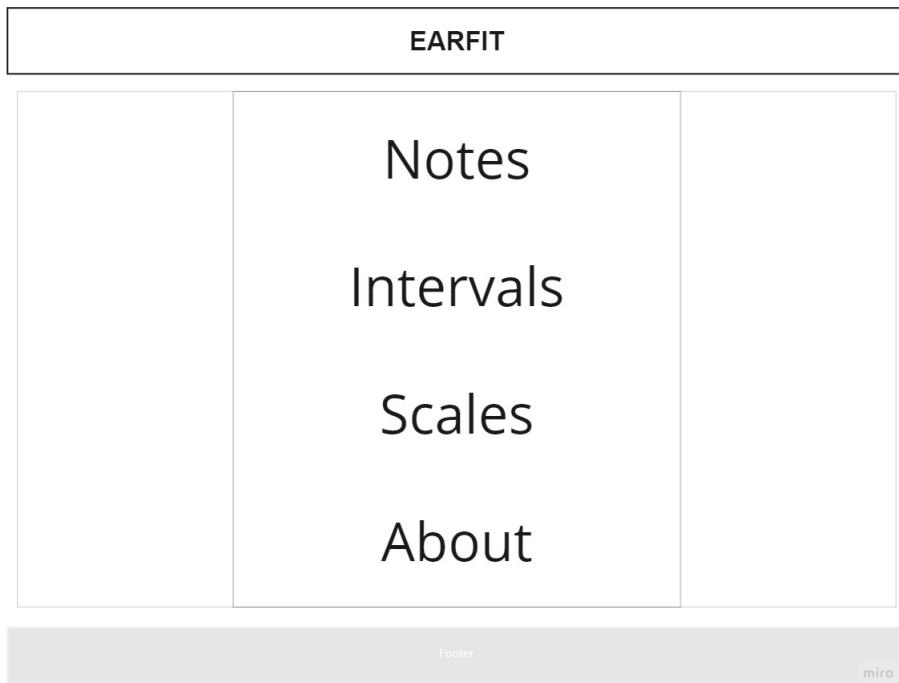


Figura B.4: Prototipo de la Página de Inicio (Pantallas Medias).

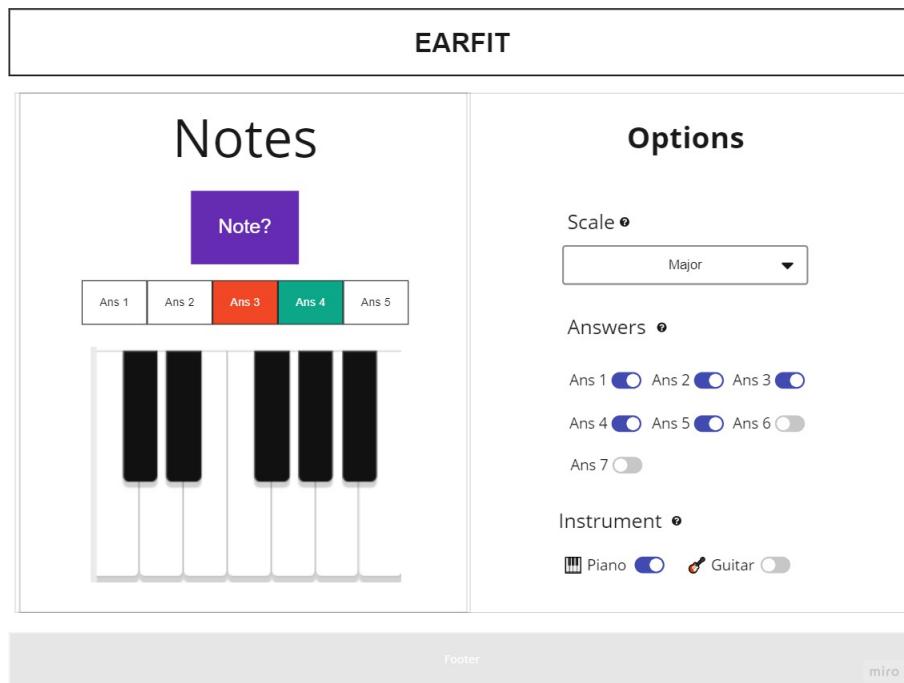


Figura B.5: Prototipo de la Página de Notas (Pantallas Medias).

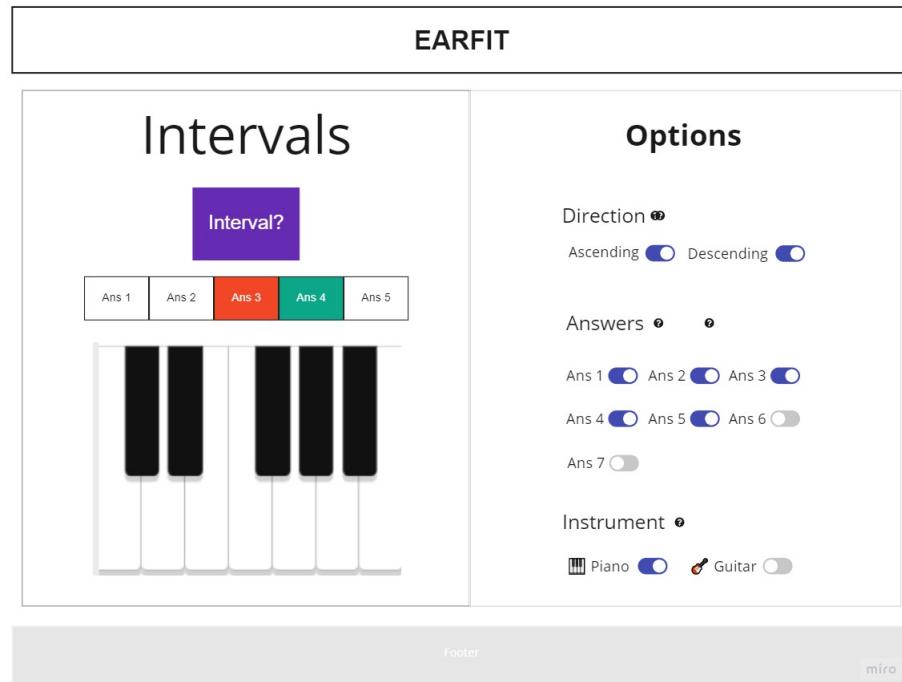


Figura B.6: Prototipo de la Página de Intervalos (Pantallas Medianas).

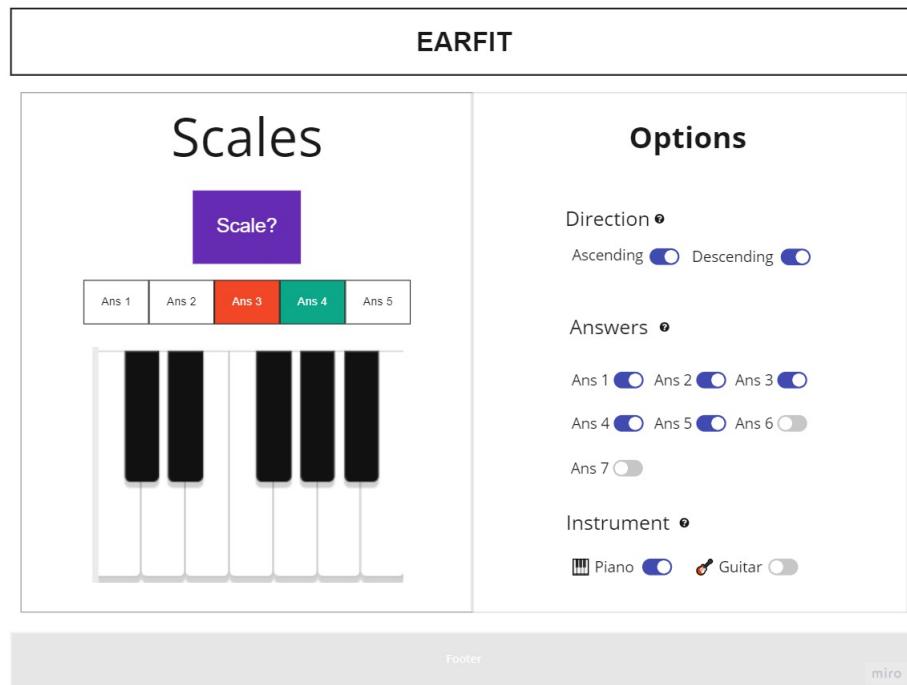


Figura B.7: Prototipo de la Página de Escalas (Pantallas Medianas).

B.3. Prototipo Pantallas Grandes

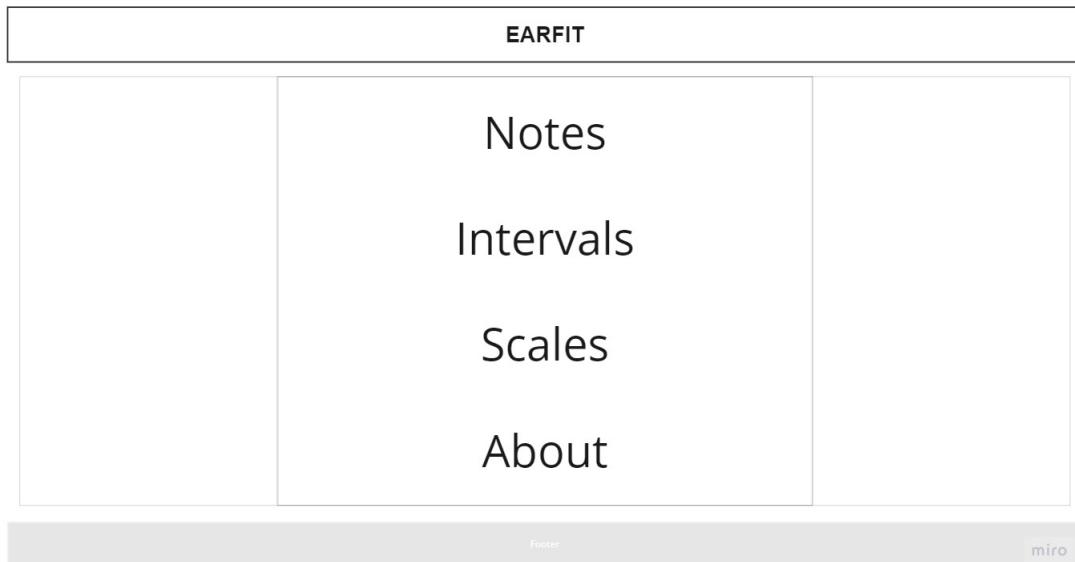


Figura B.8: Prototipo de la Página de Inicio (Pantallas Grandes).

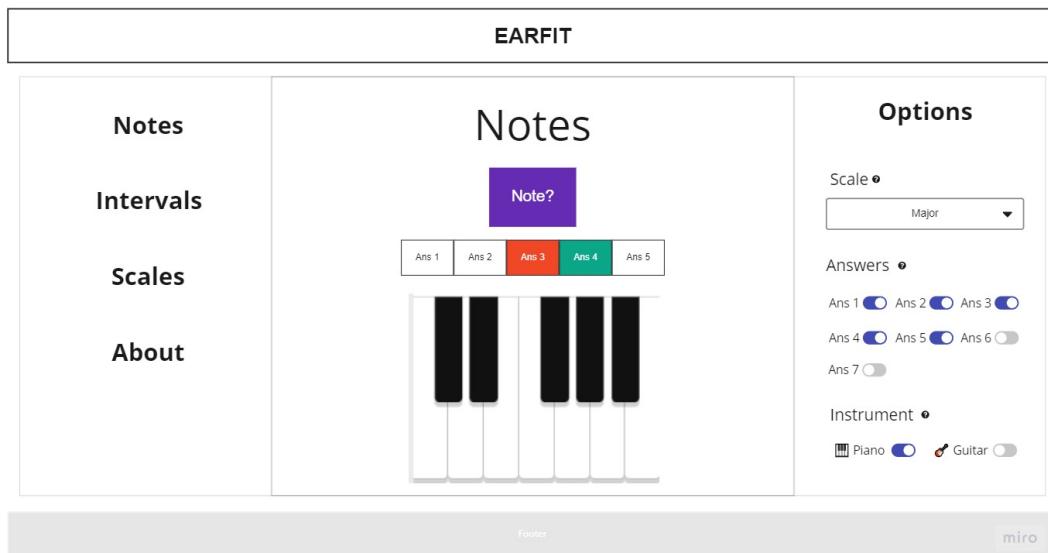


Figura B.9: Prototipo de la Página de Notas (Pantallas Grandes).

B.3. Prototipo Pantallas Grandes

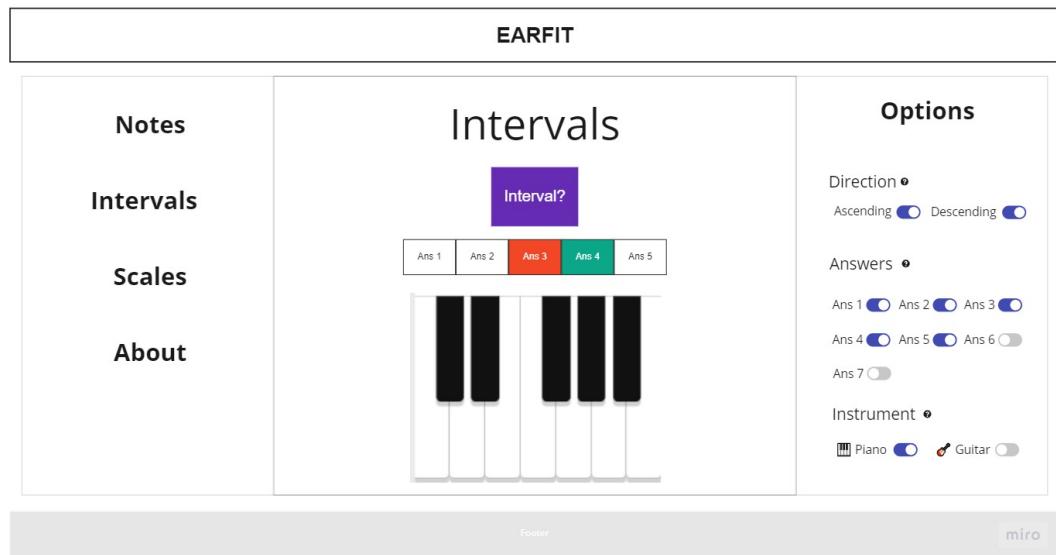


Figura B.10: Prototipo de la Página de Intervalos (Pantallas Grandes).

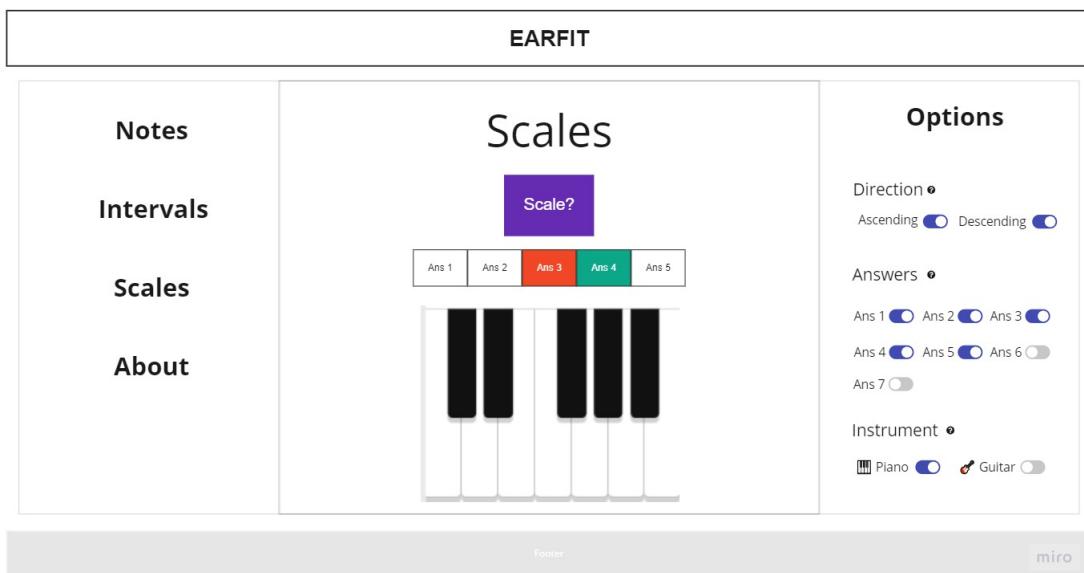


Figura B.11: Prototipo de la Página de Escalas (Pantallas Grandes).

C

Historias de Usuario

Product Backlog

TO DO Product Backlog	IN SPRINT	SPRINT 2	SPRINT 1
Personalizar Ejercicio CAMBIAR ESCALA NOTAS 1 0/3 PBI: N03 Size: M	Tocar Piano PIANO 0/1 0/3 PBI: A03 Size: L	Practicar Ejercicio ESCUCHAR ESCALA 0/1 0/3 PBI: E01 Size: M	Seleccionar Ejercicio SELECCIONAR EJERCICIO MENU PRINCIPAL 0/1 0/3 PBI: A01 Size: M
Personalizar Ejercicio CAMBIAR DIRECCIÓN INTERVALO 0/1 0/3 PBI: I03 Size: S	Cambiar Instrumento CAMBIAR INSTRUMENTO 0/1 0/4 PBI: C01 Size: M	Practicar Ejercicio ELEGIR RESPUESTA ESCALA 0/1 0/7 PBI: E02 Size: M	Seleccionar Ejercicio CAMBIAR EJERCICIO MENU LATERAL 0/1 0/4 PBI: A02 Size: S
Personalizar Ejercicio CAMBIAR DIRECCION ESCALA 0/1 0/3 PBI: E03 Size: S	Personalizar Ejercicio CAMBIAR OPCIONES RESPUESTA 0/1 0/3 PBI: C02 Size: XL	Practicar Ejercicio ESCUCHAR INTERVALO 0/1 0/3 PBI: I01 Size: L	Practicar Ejercicio ELEGIR RESPUESTA INTERVALO 0/1 0/7 PBI: I02 Size: L
+ Add a card	+ Add a card	+ Add a card	+ Add a card

Figura C.1: Product Backlog con las historias de usuario.

SELECCIONAR EJERCICIO MENU PRINCIPAL

in list SPRINT 1

Labels

Seleccionar Ejercicio +

Description Edit

Como: Músico Amateur
Quiero: Elegir un ejercicio
Para: Comenzar a practicarlo

Custom Fields

T PBI Size
A01 M

Attachments

Prototipo ↗
Added Jan 25 at 7:57 PM - [Comment](#) - [Delete](#) - [Edit](#)
[Make cover](#)

Add an attachment

Criterios de Aceptación Delete

0%

Dado: Un Menú con todos los ejercicios en la página de Inicio
 Cuando: Seleccionas un ejercicio del menú
 Entonces: Te tiene que llevar a su página de ejercicio

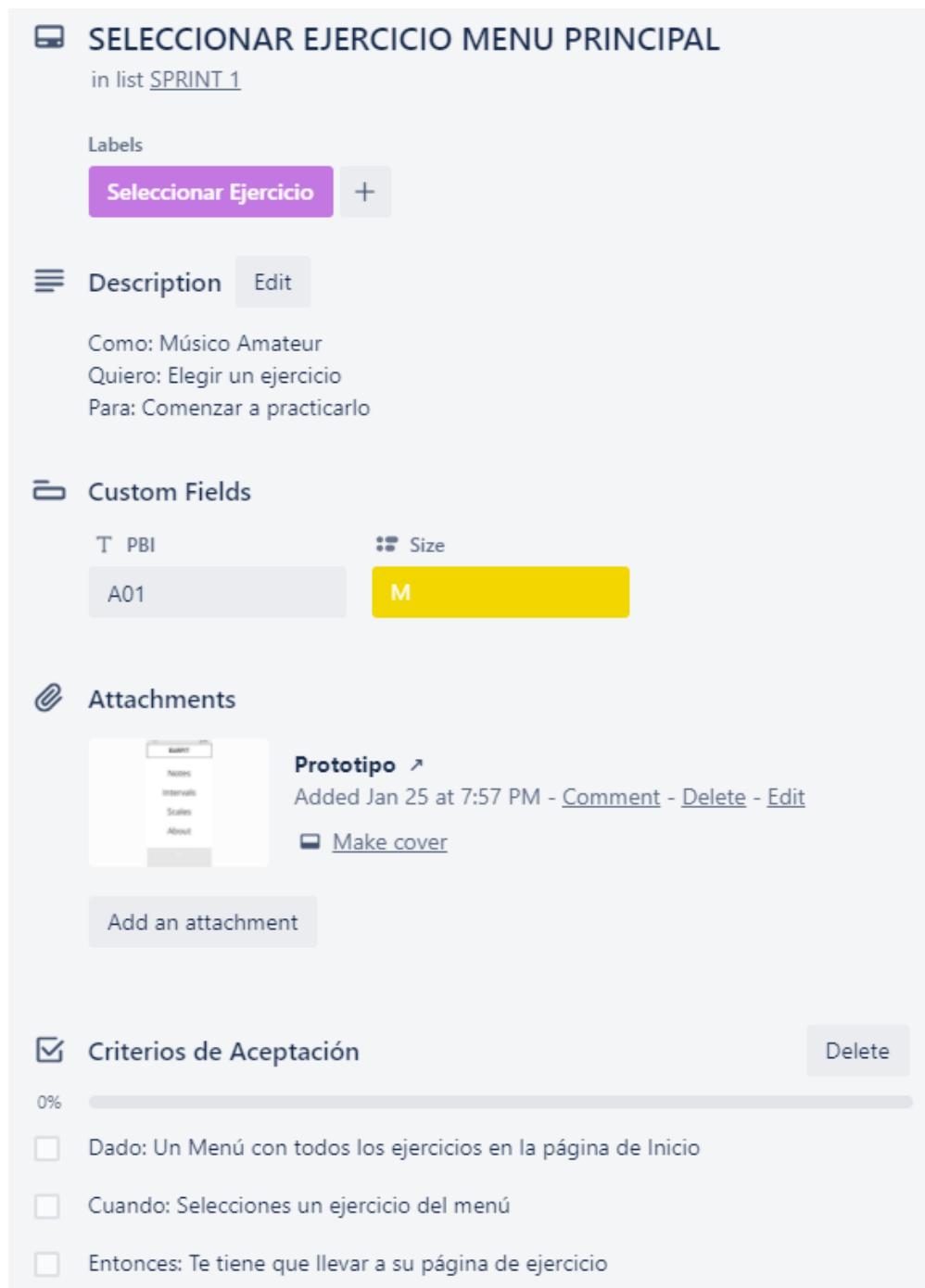


Figura C.2: Historia de usuario “Seleccionar Ejercicio Menú Principal”.

 **CAMBIAR EJERCICIO MENU LATERAL**

in list [SPRINT 1](#)

Labels

[Seleccionar Ejercicio](#) +

 **Description** [Edit](#)

Como: Músico Amateur
Quiero: Cambiar de ejercicio desde una página de ejercicio
Para: Comenzar a practicar otro ejercicio

 **Custom Fields**

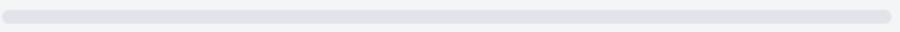
PBI	Size
A02	S

 **Attachments**

 **Prototipo** ↗
Added Jan 25 at 7:58 PM - [Comment](#) - [Delete](#) - [Edit](#)
[Make cover](#)

Add an attachment

Criterios de Aceptación [Delete](#)

0% 

Dado: Un menú con todos los ejercicios situado a la izquierda en las páginas de ejercicio

Cuando: Seleccionas un ejercicio del menú

Entonces: Te tiene que llevar a su página de ejercicio

Condiciones: Este menú sólo será visible en pantallas grandes

Figura C.3: Historia de usuario “Cambiar Ejercicio Menú Lateral”.

The screenshot shows a user story card with the following details:

- Title:** ESCUCHAR NOTA
- Labels:** Practicar Ejercicio
- Description:**
 - Como: Músico Amateur
 - Quiero: Escuchar una nota de la escala mayor
 - Para: Tratar de identificarla
- Custom Fields:**
 - PBI: N01
 - Size: L
- Attachments:**
 - Notes:
 - Prototipo ↗
Added Jan 25 at 7:59 PM - [Comment](#) - [Delete](#) - [Edit](#)
[Note?](#)
 - Add an attachment
- Criterios de Aceptación:**
 - Dado: Un botón de play en el centro del ejercicio
 - Cuando: Se pulse
 - Entonces: Emitirá una nota al azar de entre las opciones de respuesta disponibles (notas de la escala)

Figura C.4: Historia de usuario “Escuchar Nota”.

Capítulo C. Historias de Usuario

ELEGIR RESPUESTA NOTA
in list SPRINT 1

Labels
Practicar Ejercicio +

☰ Description Edit
Como: Músico Amateur
Quiero: Seleccionar una respuesta
Para: Tratar de identificar la nota que he escuchado

⊖ Custom Fields
T PBI Size
N02 L

📎 Attachments
Prototipo ↗
Added 7 minutes ago - [Comment](#) - [Delete](#) - [Edit](#)
 [Make cover](#)

Add an attachment

Criterios de Aceptación Delete
0%

- Dado: Un grupo de botones de respuesta (notas de la escala) justo debajo del botón de play
- Cuando: Pulse un botón de respuesta
- Entonces: si no coincide con la nota que se ha escuchado, el botón pasará a color rojo y el usuario podrá seguir pulsando botones
- Entonces: si coincide con la nota que se ha escuchado, el botón pasará a color verde durante 1s
- Entonces: si coincide con la nota que se ha escuchado, emitirá un sonido de acierto
- Entonces: si coincide con la nota que se ha escuchado, reseteará todos los colores del grupo
- Entonces: si coincide con la nota que se ha escuchado, se calculará una nueva nota para el botón de play

Figura C.5: Historia de usuario “Elegir Respuesta Nota”.

 **ESCUCHAR INTERVALO**

in list [SPRINT 2](#)

Labels

Practicar Ejercicio +

 **Description** [Edit](#)

Como: Músico Amateur
Quiero: Escuchar un intervalo de dos notas
Para: Tratar de identificarlo

 **Custom Fields**

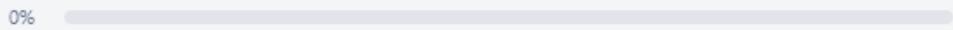
T PBI Size
I01 L

 **Attachments**

Intervals [Prototipo ↗](#)
Added Jan 25 at 8:23 PM - [Comment](#) - [Delete](#) - [Edit](#)
[Interval?](#) [Make cover](#)

Add an attachment

 **Criterios de Aceptación** [Delete](#)

0% 

Dado: Un botón de play en el centro del ejercicio
 Cuando: Se pulse
 Entonces: Emitirá un intervalo al azar de entre las opciones de respuesta disponibles (intervalos)

Figura C.6: Historia de usuario “Escuchar Intervalo”.

 **ELEGIR RESPUESTA INTERVALO**
in list SPRINT 2

Labels

Practicar Ejercicio +

☰ Description Edit

Como: Músico Amateur
Quiero: Seleccionar una respuesta
Para: Tratar de identificar el intervalo que he escuchado

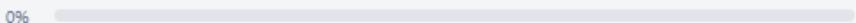
Custom Fields

T PBI Size
I02 L

📎 Attachments

Prototipo ↗
Added 12 minutes ago - [Comment](#) - [Delete](#) - [Edit](#)

Add an attachment

Criterios de Aceptación Delete
0% 

- Dado: Un grupo de botones de respuesta (intervalos) justo debajo del botón de play
- Cuando: Pulse un botón de respuesta
- Entonces: si no coincide con la nota que se ha escuchado, el botón pasará a color rojo y el usuario podrá seguir pulsando botones
- Entonces: si coincide con la nota que se ha escuchado, el botón pasará a color verde durante 1s
- Entonces: si coincide con la nota que se ha escuchado, emitirá un sonido de acierto
- Entonces: si coincide con la nota que se ha escuchado, reseteará todos los colores del grupo
- Entonces: si coincide con la nota que se ha escuchado, se calculará una nueva nota para el botón de play

Figura C.7: Historia de usuario “Elegir Respuesta Intervalo”.

ESCUCHAR ESCALA
in list SPRINT_2

Labels

Practicar Ejercicio +

Description Edit

Como: Músico Amateur
Quiero: Escuchar una escala de entre los siete modos griegos
Para: Tratar de identificarla

Custom Fields

T	PBI	Size
E01		M

Attachments

Scales

Prototipo ↗
Added Jan 25 at 8:23 PM - [Comment](#) - [Delete](#) - [Edit](#)
[Scale?](#)

Add an attachment

Criterios de Aceptación Delete

0%

Dado: Un botón de play en el centro del ejercicio

Cuando: Se pulse

Entonces: Emitirá una escala al azar de entre las opciones de respuesta disponibles (modos griegos)

Figura C.8: Historia de usuario “Escuchar Escala”.

 **ELEGIR RESPUESTA ESCALA**
in list SPRINT 2

Labels

Practicar Ejercicio +

☰ Description Edit

Como: Músico Amateur
Quiero: Seleccionar una respuesta
Para: Tratar de identificar la escala que he escuchado

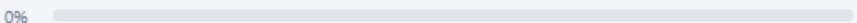
⊖ Custom Fields

T PBI Size
E02 M

📎 Attachments

Prototipo ↗
Added 12 minutes ago - [Comment](#) - [Delete](#) - [Edit](#)
 [Make cover](#)

Add an attachment

Criterios de Aceptación Delete
0% 

- Dado: Un grupo de botones de respuesta (modos griegos) justo debajo del botón de play
- Cuando: Pulse un botón de respuesta
- Entonces: si no coincide con la escala que se ha escuchado, el botón pasará a color rojo y el usuario podrá seguir pulsando botones
- Entonces: si coincide con la escala que se ha escuchado, el botón pasará a color verde durante 1s
- Entonces: si coincide con la escala que se ha escuchado, emitirá un sonido de acierto
- Entonces: si coincide con la escala que se ha escuchado, reseteará todos los colores del grupo
- Entonces: si coincide con la escala que se ha escuchado, se calculará una nueva escala para el botón de play

Figura C.9: Historia de usuario “Elegir Respuesta Escala”.

PIANO
in list [IN SPRINT](#)

Labels

Tocar Piano +

Description [Edit](#)

Como: Músico Amateur
Quiero: Tocar notas de referencia durante los ejercicios
Para: Tratar de ayudarme en la obtención de la respuesta correcta
[Oído Relativo]

Custom Fields

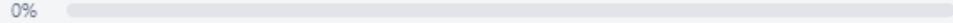
T PBI Size
A03 L

Attachments

 **Prototipo** ↗
Added Jan 25 at 7:49 PM - [Comment](#) - [Delete](#) - [Edit](#)
[Make cover](#)

Add an attachment

Criterios de Aceptación [Delete](#)

0% 

Dado: Un pequeño piano debajo del ejercicio
 Cuando: Pulse las teclas del piano
 Entonces: Emitirán el sonido correspondiente a esas teclas

Figura C.10: Historia de usuario “Piano”.

 **CAMBIAR OPCIONES RESPUESTA**
in list [IN SPRINT](#)

Labels

[Personalizar Ejercicio](#) +

 **Description** [Edit](#)

Como: Músico Amateur
Quiero: Cambiar el número de opciones de respuesta disponibles del ejercicio
Para: Ajustar la dificultad del ejercicio

 **Custom Fields**

T PBI  Size
C02 

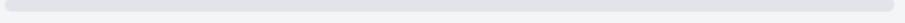
 **Attachments**

Answers •
Ans 1 Ans 2 Ans 3
Ans 4 Ans 5 Ans 6
Ans 7

Prototipo 
Added 9 minutes ago - [Comment](#) - [Delete](#) - [Edit](#)
 [Make cover](#)

Add an attachment

 **Criterios de Aceptación** [Delete](#)

0% 

Dado: Un selector de opciones de respuesta en la sección de opciones situada a la derecha del ejercicio
 Cuando: Pulses sobre un botón de opción de respuesta
 Entonces: se marcará cómo disponible y se añadirá al grupo de botones de respuesta justo debajo del botón de play

Figura C.11: Historia de usuario “Cambiar Opciones de Respuesta”.

CAMBIAR INSTRUMENTO
in list IN SPRINT

Labels

Cambiar Instrumento +

Description Edit

Como: Músico Amateur
Quiero: Cambiar entre varios instrumentos
Para: Practicar los ejercicios con sus sonidos

Custom Fields

T PBI Size
C01 M

Attachments

Prototipo ↗
Added Jan 25 at 8:00 PM - [Comment](#) - [Delete](#) - [Edit](#)
[Make cover](#)

Add an attachment

Criterios de Aceptación Delete

0%

Dado: Un selector de instrumentos en la sección de opciones situada a la derecha del ejercicio

Cuando: Pulses sobre un botón de instrumento

Entonces: El sonido que emita el botón de play será el del instrumento seleccionado

Condiciones: Siempre tiene que haber un instrumento seleccionado

Figura C.12: Historia de usuario “Cambiar Instrumento”.

CAMBIAR ESCALA NOTAS
in list [TO DO](#) | [Product Backlog](#)

Labels

Personalizar Ejercicio +

Description Edit

Como: Músico Amateur
Quiero: Cambiar la escala del ejercicio de notas
Para: Practicar notas de diferentes escalas

Custom Fields

PBI N03 Size S

Attachments

Prototipo ↗
Added Jan 25 at 8:01 PM - [Comment](#) - [Delete](#) - [Edit](#)
 [Make cover](#)

Add an attachment

Criterios de Aceptación Delete

0%

Dado: Un menú desplegable con los siete modos griegos situado en la sección de opciones situada a la derecha del ejercicio

Cuando: Seleccione uno de los modos griegos (escala)

Entonces: Las opciones de respuesta pasarán a ser las notas de la escala seleccionada

Figura C.13: Historia de usuario “Cambiar Escala Notas”.

 **CAMBIAR DIRECCIÓN INTERVALO**

in list [TO DO](#) | [Product Backlog](#)

Labels

Personalizar Ejercicio +

 **Description** Edit

Como: Músico Amateur
Quiero: Cambiar la dirección en la que suenan las dos notas del ejercicio de intervalos
Para: Practicar intervalos ascendentes y descendentes

 **Custom Fields**

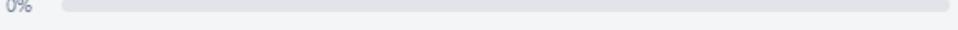
T PBI	Size
I03	S

 **Attachments**

Prototipo ↗
Added Jan 25 at 8:23 PM - [Comment](#) - [Delete](#) - [Edit](#)
[Make cover](#)

Add an attachment

 **Criterios de Aceptación** Delete

0% 

- Dado: Un selector de dirección (ascendente o descendente) situado en la sección de opciones de la derecha del ejercicio
- Cuando: Pulses sobre un botón de dirección
- Entonces: El orden en el que emita las dos notas el botón de play será en el de la dirección seleccionada

Figura C.14: Historia de usuario “Cambiar Dirección Intervalo”.

 **CAMBIAR DIRECCION ESCALA**

in list [TO DO](#) | [Product Backlog](#)

Labels

[Personalizar Ejercicio](#) +

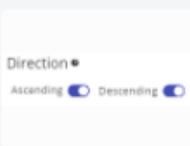
 **Description** [Edit](#)

Como: Músico Amateur
Quiero: Cambiar la dirección en la que suenan las notas del ejercicio de escalas
Para: Practicar escalas ascendentes y descendentes

 **Custom Fields**

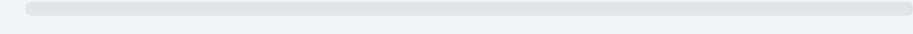
PBI	Size
E03	S

 **Attachments**


Direction •
Ascending Descending
[Prototipo](#) ↗
Added Jan 25 at 8:23 PM - [Comment](#) - [Delete](#) - [Edit](#)
[Make cover](#)

[Add an attachment](#)

Criterios de Aceptación [Delete](#)

0% 

- Dado: Un selector de dirección (ascendente o descendente) situado en la sección de opciones de la derecha del ejercicio
- Cuando: Pulses sobre un botón de dirección
- Entonces: El orden en el que emita las notas de la escala el botón de play será en el de la dirección seleccionada

Figura C.15: Historia de usuario “Cambiar Dirección Escala”.

D

Capturas de la Aplicación

D.1. Pantalla Pequeña



Figura D.1: Captura de la Página de Inicio y Notas (Pantallas Pequeñas).

Capítulo D. Capturas de la Aplicación

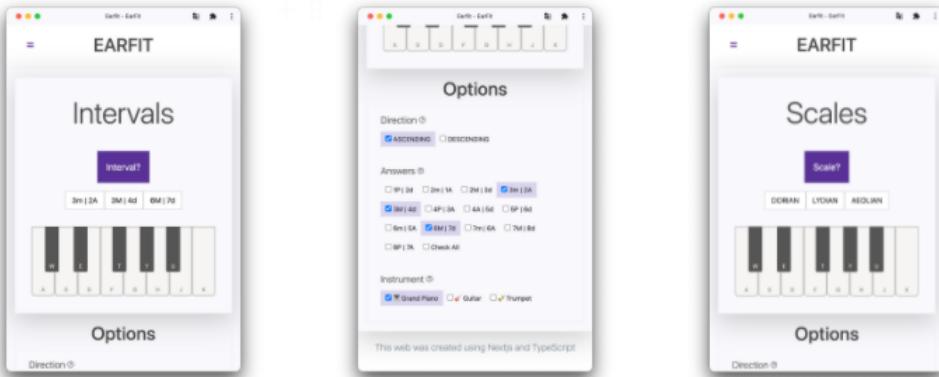


Figura D.2: Captura de la Página de Intervalos y Escalas (Pantallas Pequeñas).



Figura D.3: Captura de la Página de Escalas, Piano y About (Pantallas Pequeñas).

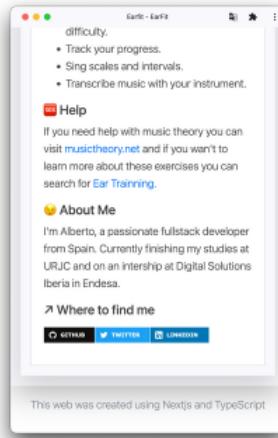


Figura D.4: Captura de la Página de About (Pantallas Pequeñas).

D.2. Pantalla Media

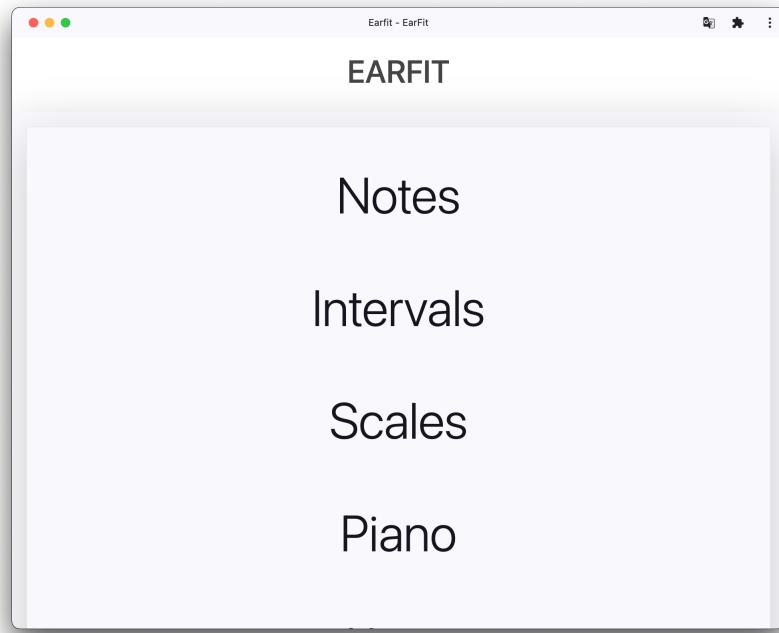


Figura D.5: Captura de la Página de Inicio (Pantallas Medianas).

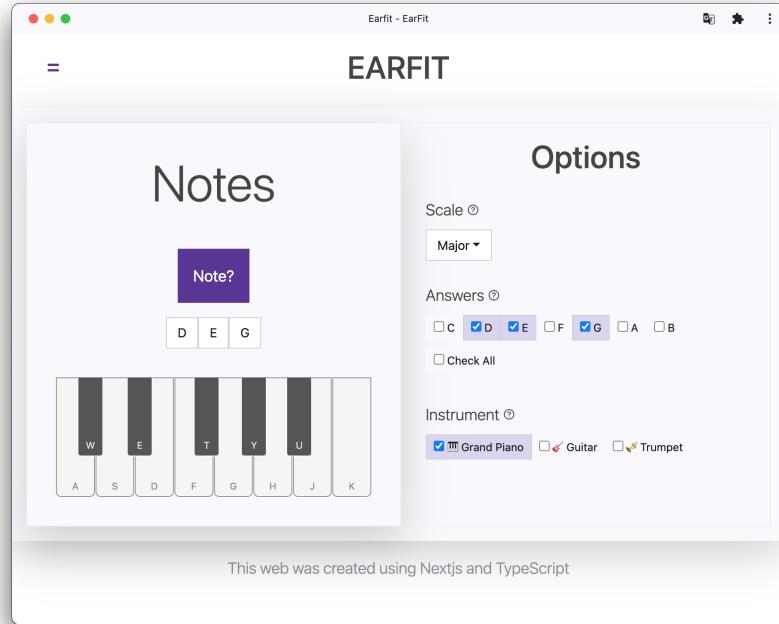


Figura D.6: Captura de la Página de Notas (Pantallas Medianas).

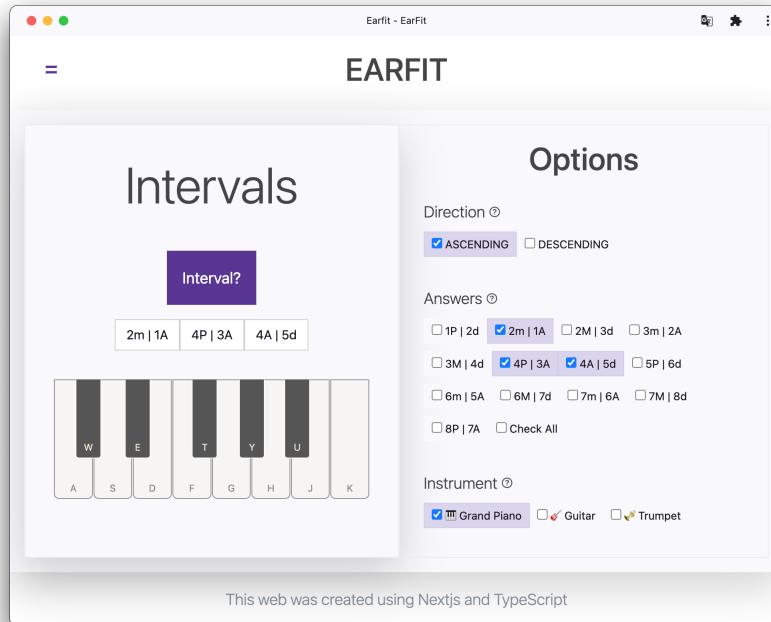


Figura D.7: Captura de la Página de Intervalos (Pantallas Medias).

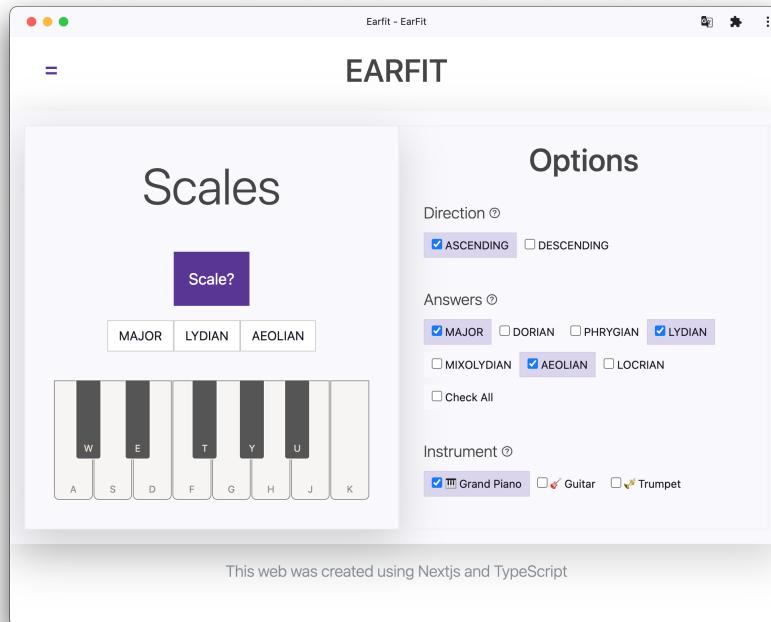


Figura D.8: Captura de la Página de Escalas (Pantallas Medias).

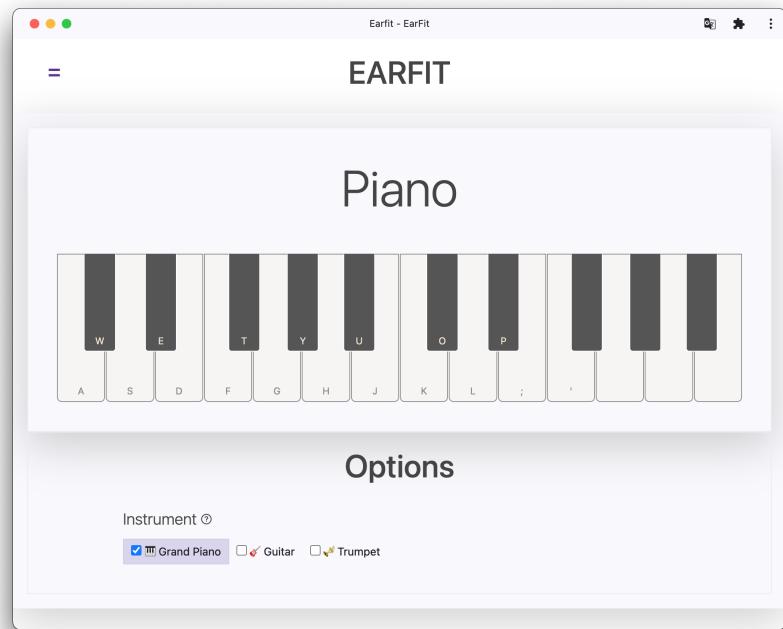


Figura D.9: Captura de la Página de Piano (Pantallas Medias).

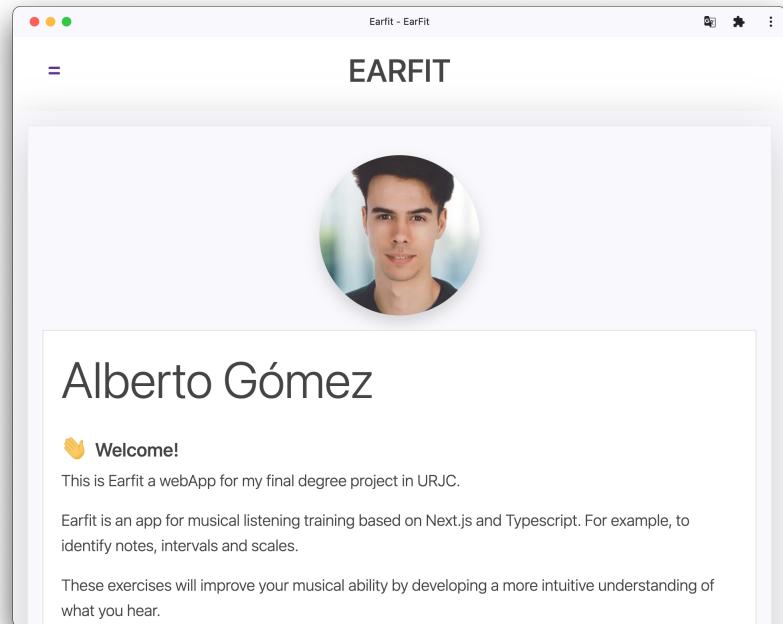


Figura D.10: Captura de la Página de About 1 (Pantallas Medias).

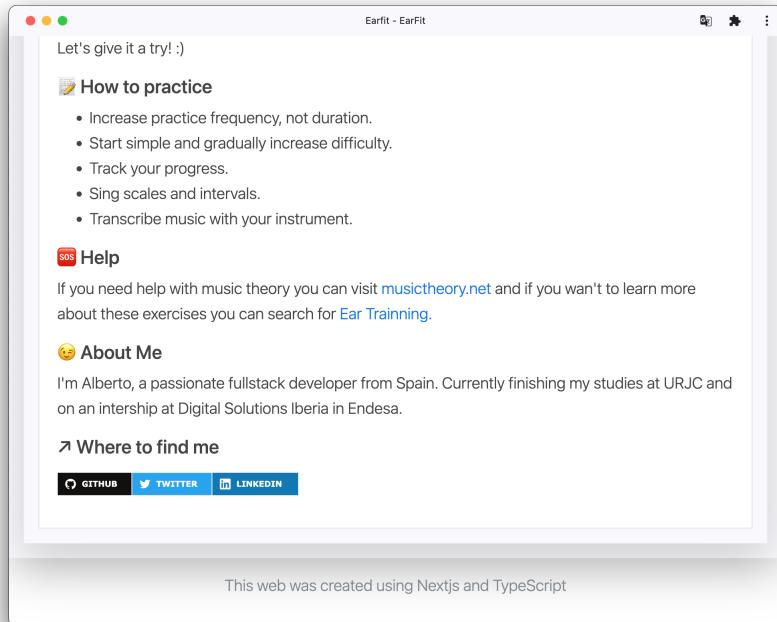


Figura D.11: Captura de la Página de About 2 (Pantallas Medianas).

D.3. Pantalla Grande

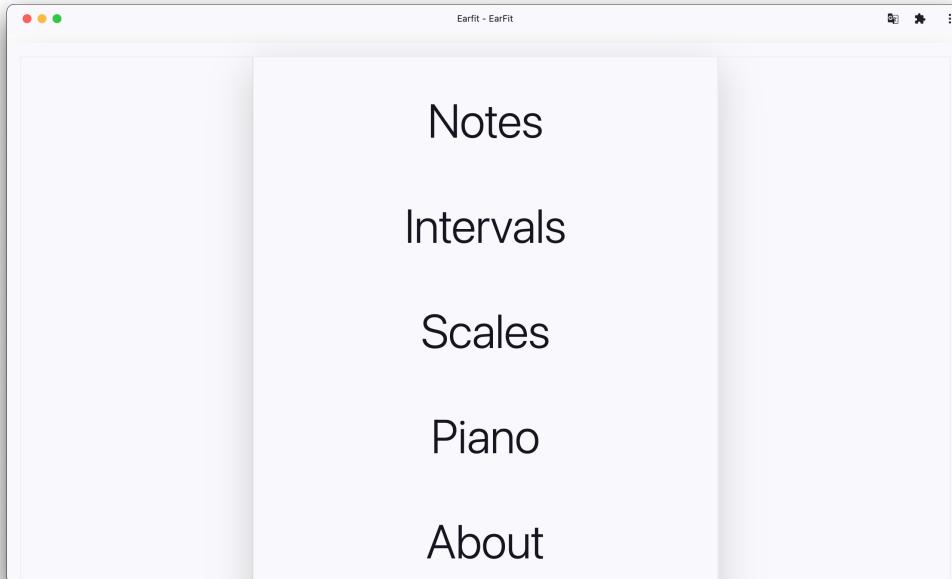


Figura D.12: Captura de la Página de Inicio (Pantallas Grandes).

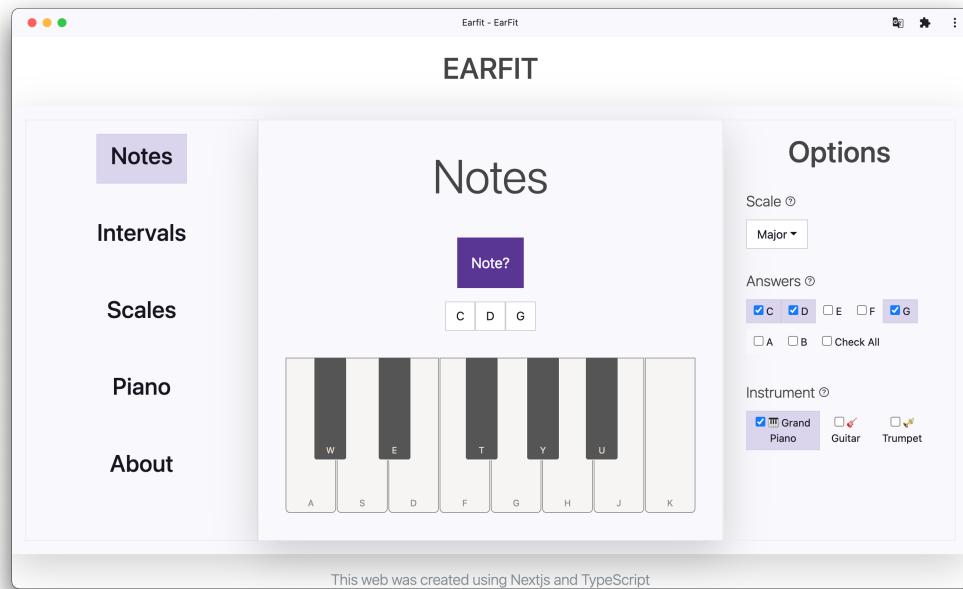


Figura D.13: Captura de la Página de Notas (Pantallas Grandes).

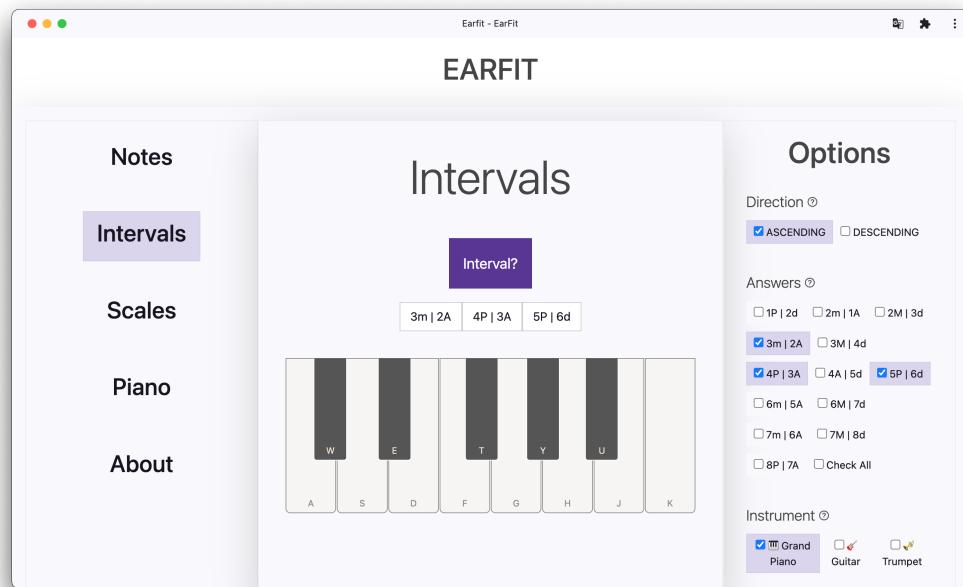


Figura D.14: Captura de la Página de Intervalos (Pantallas Grandes).

Capítulo D. Capturas de la Aplicación

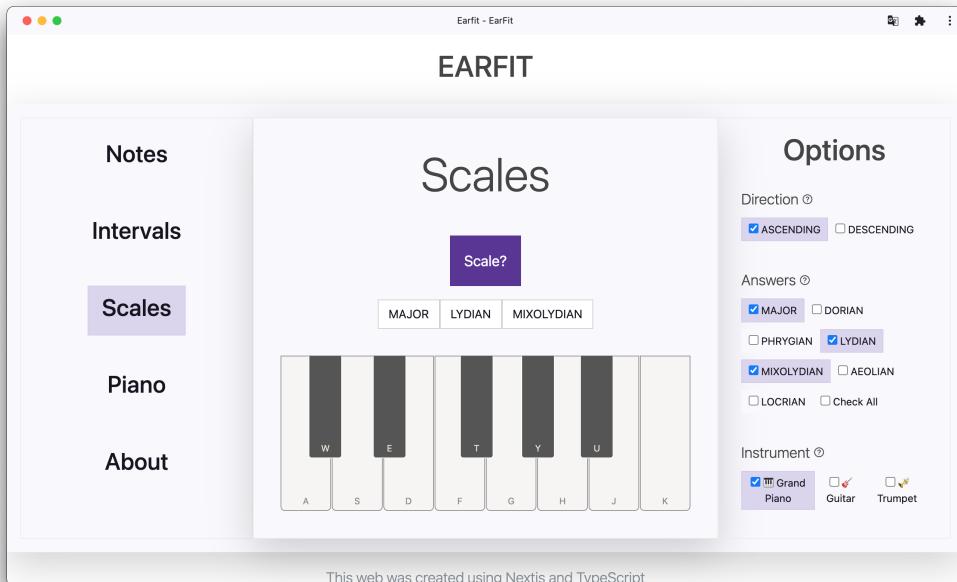


Figura D.15: Captura de la Página de Escalas (Pantallas Grandes).



Figura D.16: Captura de la Página de Piano (Pantallas Grandes).

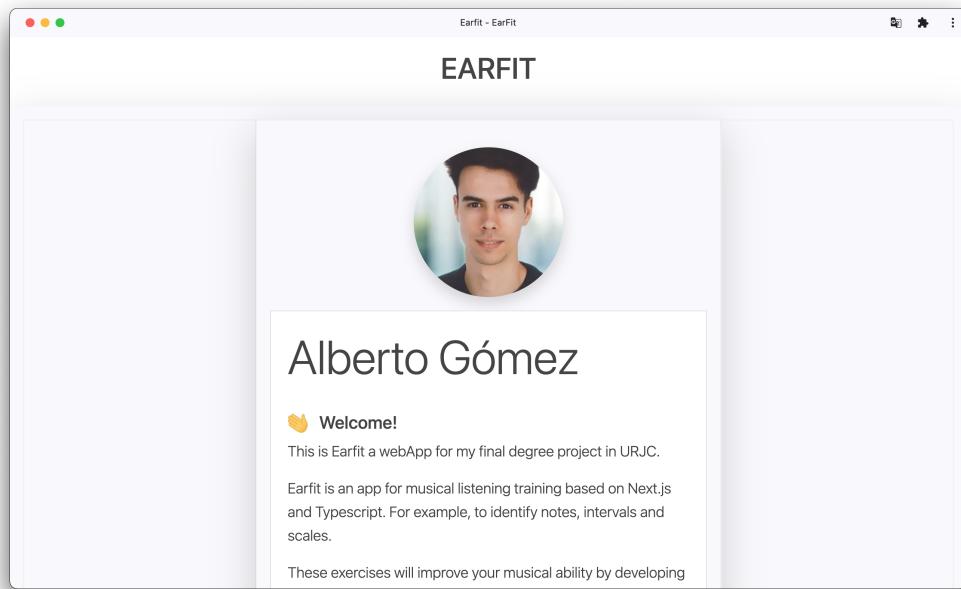


Figura D.17: Captura de la Página de About 1 (Pantallas Grandes).

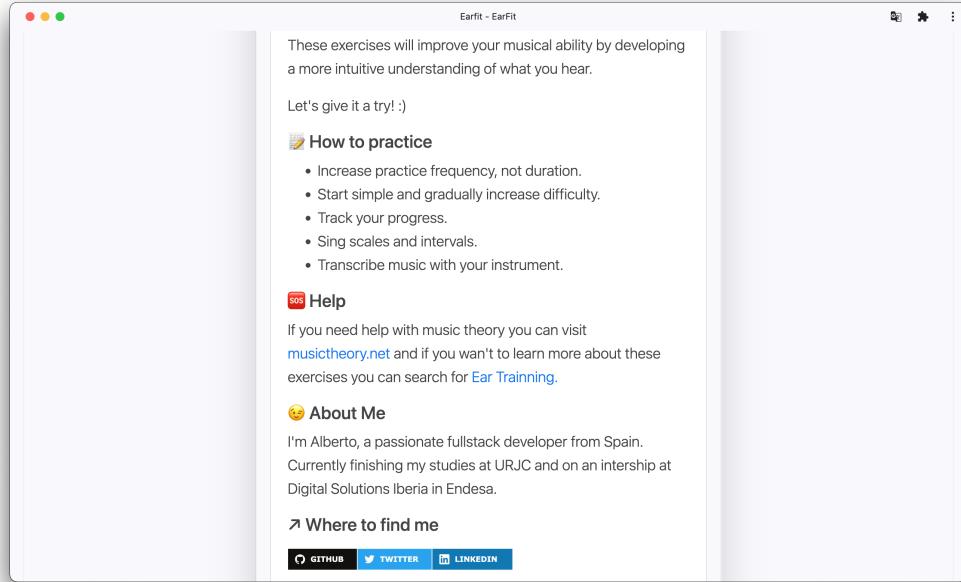


Figura D.18: Captura de la Página de About 2 (Pantallas Grandes).

E

Clean Code

A continuación se enumeran algunos de los principales principios de Clean Code que se han intentado tener presentes a la hora de desarrollar el código, adaptados a React.

- Follow Standard Conventions: Seguir convenciones estandarizadas.
- DRY Principle (Don't Repeat Yourself): No repetir código.
- The Principle of Least Surprise: Las funciones o Hooks deben hacer lo que se espera que hagan.
- The Boy Scout Rule: Dejar el código más limpio de como te lo encontraste.
- Keep It Simple Stupid: Reducir la complejidad tanto como sea posible.
- You Are Not Gonna Needed: Sólo se debe añadir el código que sea estrictamente necesario.
- Choose Descriptive Names: El código debe ser legible para otros desarrolladores.
- Be Consistent: Si haces algo de cierta manera, haz todas las cosas similares de la misma manera.
- Single Responsibility Principle: Los Hooks deben tener responsabilidad sobre una sola parte de la funcionalidad proporcionada.
- Open/Closed Principle: Los Hooks deben estar abiertos a extensiones pero cerrados a modificaciones.
- Liskov Substitution Principle: Los componentes derivados deben poder sustituirse por sus componentes base.
- Interface Segregation Principle: Los componentes/Hooks sólo deberían conocer los métodos que realmente usan.
- Dependency Inversion Principle: Depende de abstracciones, no de detalles concretos.

F

Limitaciones Personales

Hay que tener en cuenta algunas de las limitaciones que obstaculizaron y aumentaron el tiempo de desarrollo del proyecto. Este proyecto se realizó en el tiempo libre que me quedaba entre mi trabajo de ocho horas y el master que estaba cursando a la vez. Lo que se traduce en **poco tiempo** para desarrollar, aprender teoría musical, nuevas tecnologías, metodologías y realizar la memoria.

Mis conocimientos sobre **teoría musical** al inicio del proyecto eran muy básicos. Prácticamente todo lo aplicado sobre música lo he tenido que aprender de cero para el desarrollo de los ejercicios. Algo que no es fácil si nunca has estudiado sobre notación musical, semitonos, modos griegos, etc. Lo que produjo algún que otro error sobre todo en el ejercicio de intervalos.

Sobre las **metodologías** empleadas, ya había utilizado Design Thinking en la carrera. Pero nunca había implementado metodologías ágiles como Scrum, por lo que he tenido que aprender sobre cómo realizar historias de usuario, el Scrum Board, el User Story Map y cómo gestionarlas. Tampoco había tratado de implementar Lean Startup aunque tenía una ligera idea. Por último, si que había llevado control de versiones anteriormente pero nunca usando el flujo Gitflow.

Sobre las **tecnologías** empleadas, nunca había utilizado ninguna aparte de Nodejs y VS-Code. Por lo que hay una gran cantidad de tiempo invertido en aprender Nextjs, React y TypeScript. Tampoco había utilizado nunca Github Actions y Vercel para realizar un despliegue continuo de la aplicación.

Cómo consecuencias el desarrollo se vio entorpecido por algunos errores que tenía al actualizar el estado cuando era un array o tener que hacer una migración de Bootstrap a react-bootstrap a mitad del proyecto, cuando me di cuenta que algunos componentes no funcionaban bien.

Por último, tampoco había usado nunca la WebAudio API algo que fue un quebradero de cabeza para hacer funcionar la aplicación en IOS, ya que su documentación es complicada.