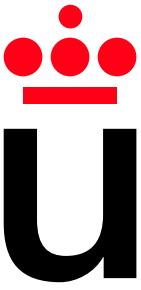

Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

Grado en Diseño y Desarrollo de Videojuegos

Curso 2022-2023

Trabajo Fin de Grado

GUITEAR, APLICACIÓN DE ENTRNAMIENTO MUSICAL ORIENTADA A GUITARRA

Autora: Abril Delgado Álvarez
Tutor: Manuel Rubio Sánchez

Agradecimientos

Quisiera agradecer a todo el mundo que me ha ayudado y ha hecho posible que este trabajo haya salido adelante.

Primero a mis padres, que han estado siempre ahí para cualquier cosa que he necesitado, y han hecho posible que llegase a estudiar lo que quería y en general, a hacer realidad cualquier cosa que me he propuesto.

Agradecer también a todos mis amigos, a los de siempre y a todos los que he conocido durante estos últimos años. Nunca habría conseguido nada de esto si no fuese por vosotros. Agradecer en especial a Lucía, Daniel, Jesús, Raúl, Rocío, Andrea y Pilar.

Por último, quiero agradecer a los profesores del grado por haber dedicado su tiempo a enseñarme todo lo que he aprendido durante estos años, también agradecer a mis compañeros de clase, de los que también he aprendido muchas cosas.

Resumen

Este trabajo de fin de grado consiste en la elaboración de una aplicación web enfocada al aprendizaje y práctica de la guitarra, aunque incluyendo también contenidos necesarios para cualquier músico en general.

La aplicación cuenta con apartados tanto de teoría musical y específica de guitarra; como con ejercicios para practicar lo visto en la teoría. El objetivo es desarrollar una buena base para la práctica de la guitarra y mejorar el oído musical, habilidad imprescindible para cualquier músico. Los ejercicios se realizan con una interfaz que se asemeja a la propia guitarra, lo que ayuda a trasladar fácilmente los contenidos aprendidos de la aplicación a la guitarra.

La aplicación también cuenta con un sistema de cuentas de usuario. Estas cuentas se usan para realizar un seguimiento del progreso y de la práctica de cada usuario. Cuenta con un calendario donde ver todos los días que se han practicado y con un sistema de puntuaciones que muestra las medias de los últimos días y las mejores puntuaciones del usuario.

La aplicación está desarrollada en Nextjs, un framework que funciona sobre React. Funcionando todo, tanto el cliente como el servidor, sobre JavaScript.

Para probar la aplicación se puede acceder al siguiente enlace:

https://guitear-305531c1234d.herokuapp.com

Palabras clave:

- Música
- Guitarra
- Entrenamiento auditivo musical
- Node.js
- React
- Firebase

Índice de contenidos

Índice de tablas

Índice de figuras

1 Introducción

1.1. Descripción

El objeto de este Trabajo de Fin de Grado es la creación de una aplicación que ayude a guitarristas, novatos y experimentados, a practicar y mejorar sus habilidades con la guitarra y a entrenar su oído musical. Para ello, la aplicación ofrecerá distintos ejercicios enfocados en los distintos conceptos a practicar, además de un apartado con la teoría referente a los ejercicios disponibles, para que los usuarios puedan repasarla en cualquier momento.

Además, los usuarios contarán con su propia cuenta, donde podrán ver que días han practicado y que tipos de ejercicio. También podrán consultar las puntuaciones de los distintos ejercicios y comparar sus mejores intentos y las medias de los últimos días.

Para la comodidad del usuario, la aplicación será una aplicación web, permitiendo a cualquiera acceder desde cualquier dispositivo, lo que facilita mantener una práctica continuada.

1.2. Motivación

La idea detrás de esta aplicación es juntar en un mismo lugar todo lo que pueda necesitar alguien que este aprendiendo a tocar la guitarra o que quiera practicar o repasar sus conocimientos. La mayoría de las webs de guitarra se centran en la teoría, sin ejercicios para practicarla y prácticamente la totalidad

de páginas con ejercicios están hechas de forma general o enfocadas en el piano.

Además, desde la parte técnica, durante el grado hemos trabajado con muchas cosas, y personalmente trabajo con una aplicación para Android. Por lo tanto, hacer una aplicación web como Trabajo de Fin de Grado me parecía una muy buena idea para profundizar un poco en otro campo de la programación y aprender cómo funcionan y se hacen las aplicaciones web.

1.3. Estructura de la memoria

A continuación se detallará la estructura que seguirá esta memoria:

- En el segundo capítulo, se detallarán los objetivos de la aplicación.
- En el tercer capítulo, se analizarán aplicaciones similares y se mostrarán las herramientas que se usarán en el desarrollo de la aplicación.
- En el cuarto capítulo, se mostrarán el método de trabajo elegido y el diseño de la aplicación.
- En el quinto capítulo, se explicará en detalle cómo se han realizado las distintas funcionalidades de la aplicación a nivel técnico.
- En el sexto capítulo, se explicará como se ha verificado el correcto funcionamiento de la aplicación y evaluado el producto final.
- En el séptimo capítulo, se analizarán los resultados del trabajo, se obtendrán conclusiones y se valorarán posibles mejoras y trabajos futuros.

Por último, se incluye un apéndice al final de la memoria con las instrucciones a seguir para configurar el proyecto en cualquier ordenador.

2

Objetivos

El trabajo por realizar consiste en una aplicación web. Esta decisión se ha tomado principalmente para que el usuario pueda acceder fácilmente en cualquier momento desde cualquier dispositivo. Por lo tanto, la aplicación debe funcionar en cualquier tipo de pantalla y debe estar preparada para aceptar diferentes tipos de interacción (principalmente ratón y teclado y pantalla táctil).

La aplicación está enfocada a guitarristas y gran parte de los ejercicios serán específicos para guitarra. Por consiguiente, la aplicación tiene que mostrar al usuario una interfaz familiar, lo más parecida a una guitarra, que le permita trasladar fácilmente los contenidos practicados al instrumento real. Esta interfaz debe adaptarse a los diferentes tipos de ejercicio, ya que distintos ejercicios requerirán distintas interacciones con la interfaz. Además, la interfaz dará opción para cambiar tu mano dominante, ya que la posición de las cuerdas, y, por lo tanto, de los ejercicios, se invierte.

La aplicación también presentará ejercicios para practicar el oído musical, de modo que la aplicación requerirá también de un sistema para reproducir sonidos, tanto notas sueltas como acordes. Aprovechando este sistema, la aplicación también permitirá escuchar cualquier configuración introducida en la interfaz de guitarra, para ayudar con una referencia auditiva a los usuarios.

3

Estado del arte

3.1. Estudio de alternativas

Durante la elaboración de la idea para este trabajo, se han comparado diferentes aplicaciones similares a la que se propone elaborar. Nos vamos a enfocar en dos aplicaciones en concreto que exponen claramente los problemas que se quieren abordar con esta aplicación.

La primera de ellas es TonedEar [1]. Esta aplicación está dedicada exclusivamente al entrenamiento del oído y cuenta con muchos ejercicios distintos (figura 3.1). Además, el usuario puede personalizar a su gusto, tanto los contenidos a practicar como el número de ejercicios. Por ejemplo, en la figura 3.2 se pueden ver las opciones de un ejercicio de intervalos. Cuenta con muchas opciones, como elegir cuales pueden aparecer, la velocidad de los audios o incluso la opción de empezar siempre en la misma nota.

El problema con esta aplicación no es tanto la aplicación en sí, como las necesidades del músico. Aunque la práctica del oído musical es muy importante, también lo son otros aspectos, que el usuario tendrá que buscar en otro lugar, repartiendo su tiempo de practica entre diversas aplicaciones.

La otra aplicación que quiero mencionar es ToneGym [2]. Esta aplicación también tiene ejercicios para practicar el oído musical y además incorpora ejercicios de teoría musical e incluso un apartado con teoría para repasar antes o después de los ejercicios. Cada día tienes una rutina de ejercicios por hacer que dependiendo de lo bien que te salgan se vuelven más complicados al día siguiente 3.4. También

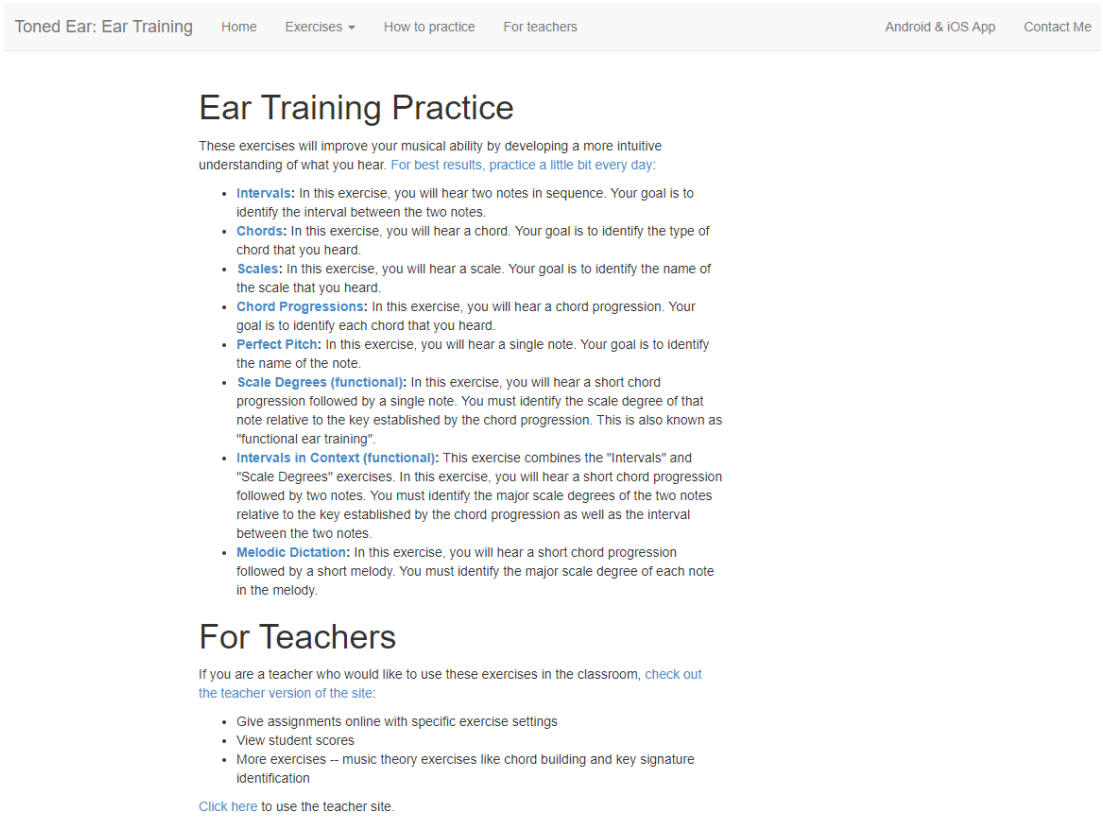

Figura 3.1: Ejercicios disponibles en TonedEar.

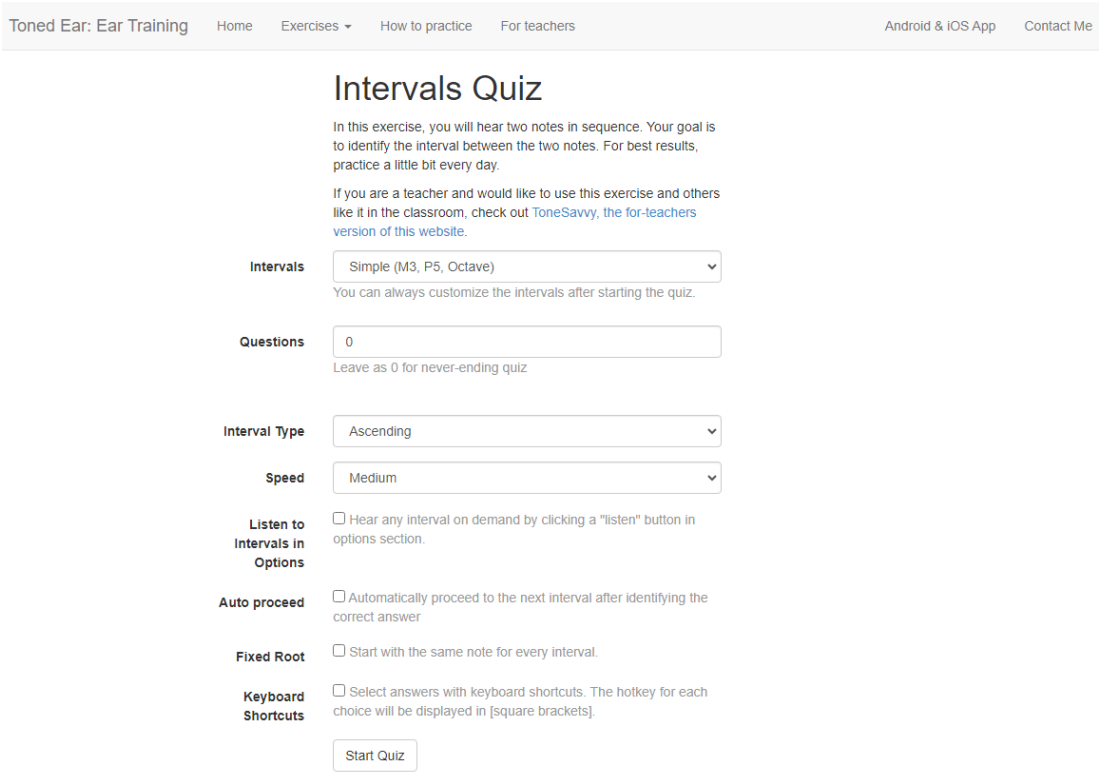

Figura 3.2: Opciones del ejercicio de intervalos.

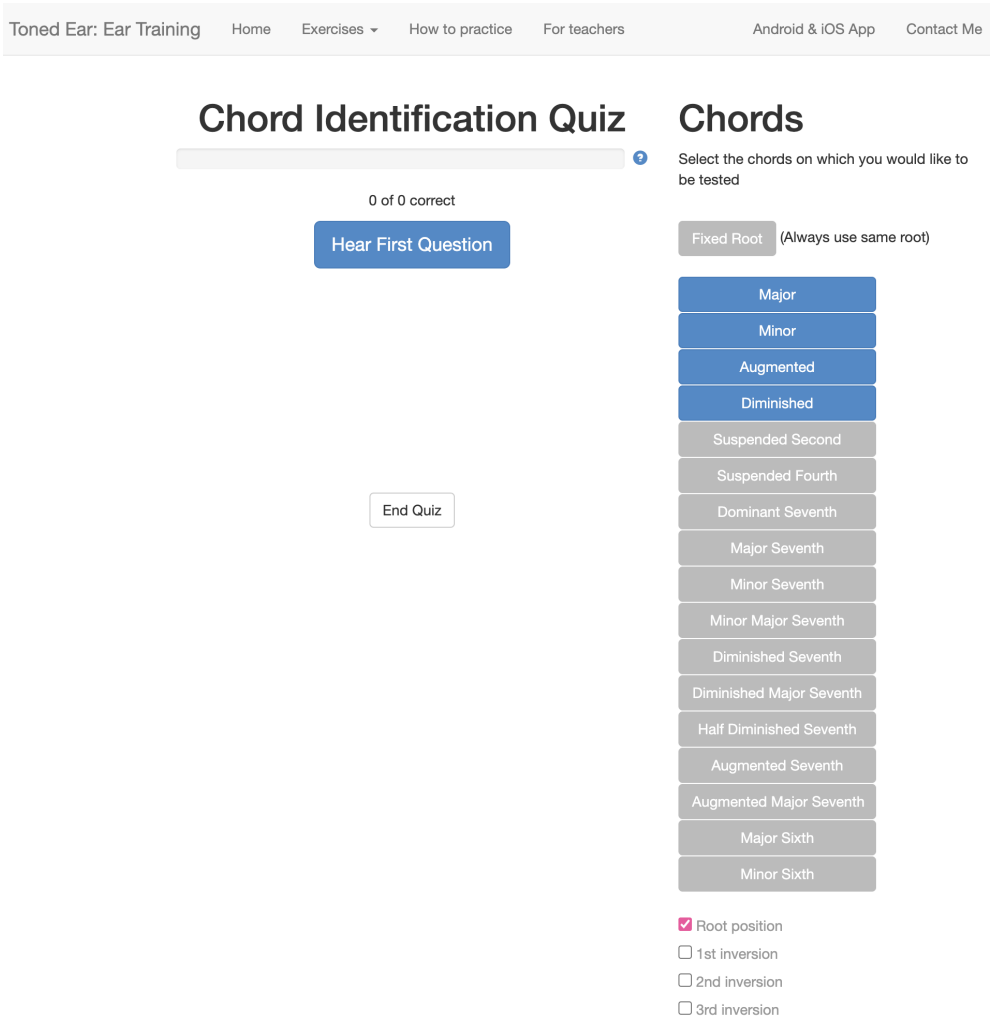

Figura 3.3: Ejercicio de acordes en la aplicación TonedEar.

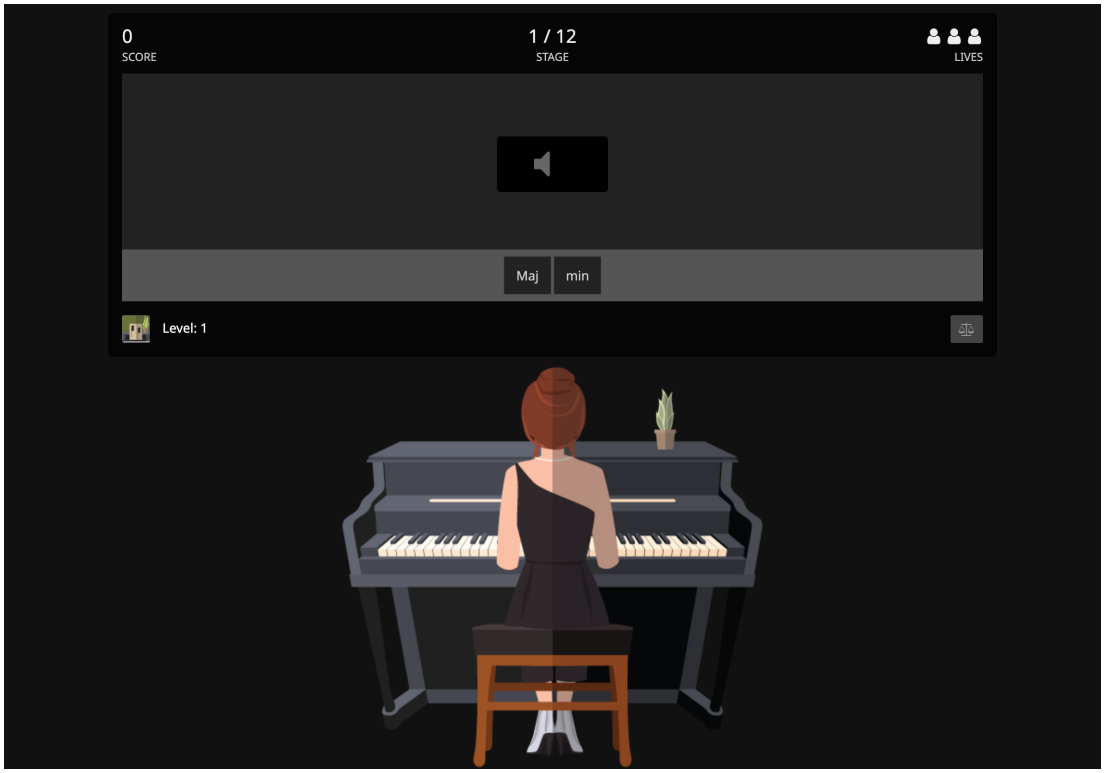

Figura 3.4: Ejercicio de acordes en la aplicación ToneGym.

puedes practicar en cualquier momento unos ejercicios más sencillos que llaman "de calentamiento". Además, a parte de la teoría para los ejercicios, la aplicación cuenta con cursos más elaborados, de diferentes temas 3.6.

Esta aplicación está enfocada a los músicos en general, independientemente del instrumento que toquen, por lo tanto, todos los ejercicios y explicaciones están basados en el piano. Esta es una buena idea para conseguir esa generalidad, pero deja atrás las particularidades de cada instrumento, que igual que con la aplicación anterior, provoca que tengas que acceder a otros recursos, ya sean aplicaciones, libros, etc., para aprender o aplicar cosas a tu instrumento en específico.

Por último, también se ha planteado la posibilidad de usar otra plataforma para la aplicación, principalmente Android o iOS. Pero, aparte de los motivos mencionados en el capítulo anterior sobre la motivación, el objetivo principal de la aplicación es facilitar la práctica diaria y hacerlo lo más sencillo de usar, por lo tanto una aplicación web es lo que más versatilidad ofrece en cuestión del mayor número de dispositivos compatibles.

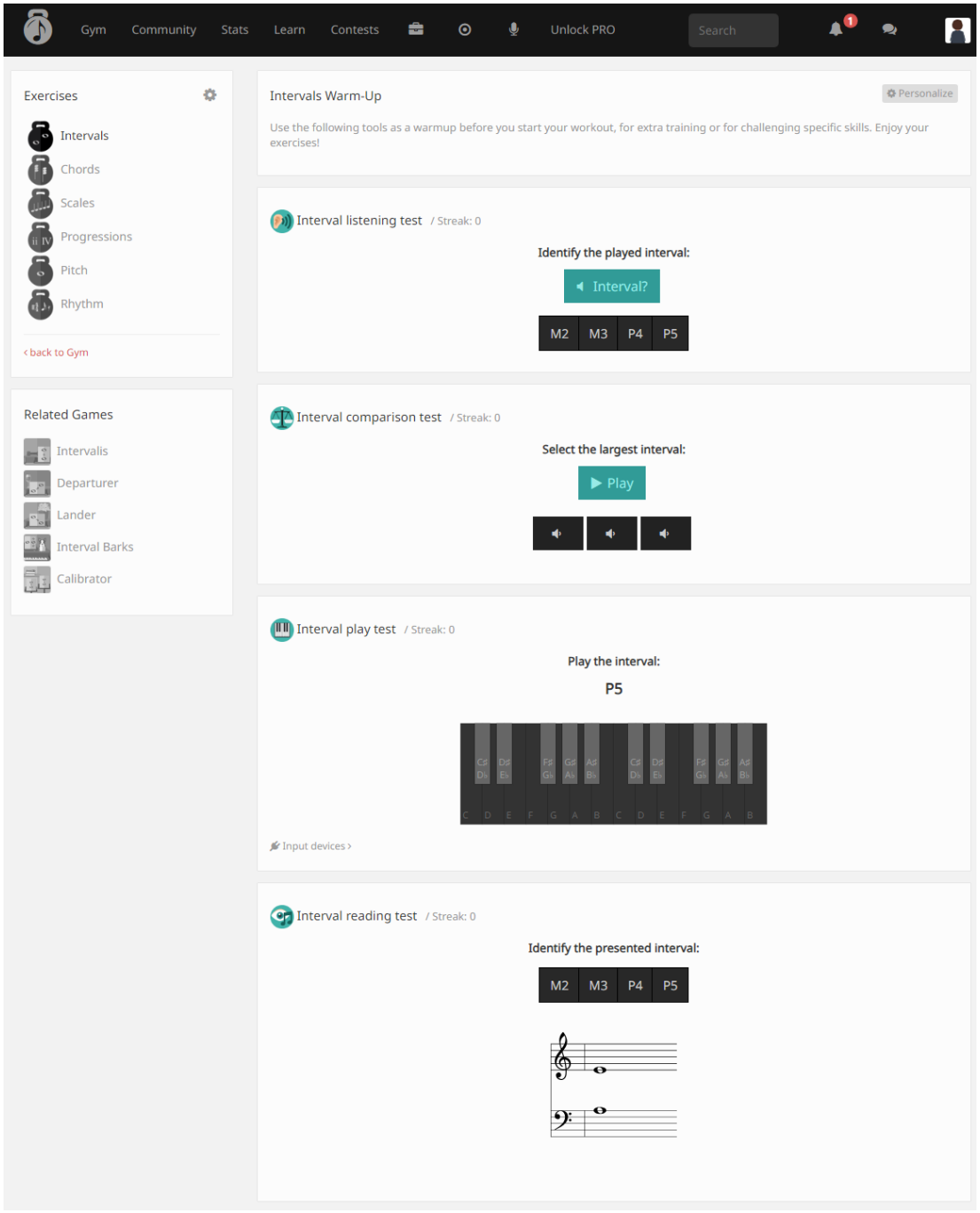

Figura 3.5: Ejercicios de calentamiento de ToneGym.

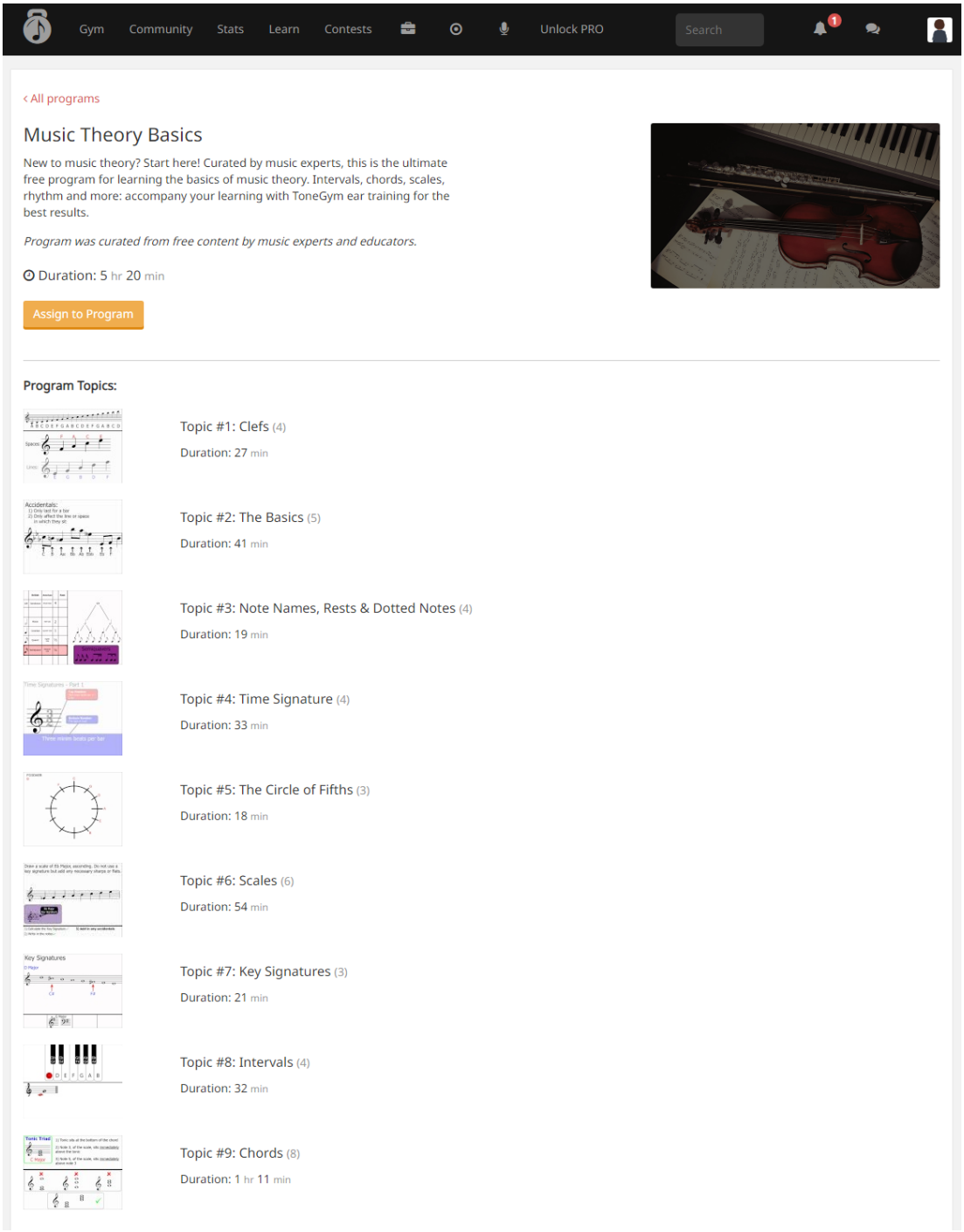

Figura 3.6: Curso de aprendizaje ToneGym.

Figura 3.7: Logos de herramientas y tecnologías.

3.2. Herramientas utilizadas

Para la realización del trabajo, se han utilizado diversas herramientas. A continuación, se describirán en este orden las aplicaciones, las tecnologías y librerías y por último las plataformas usadas.

- Aplicaciones:
  - **Visual Studio Code**: este es el entorno de desarrollo elegido para la aplicación. Es uno de los entornos de desarrollo más usados para desarrollo web. Funciona con los lenguajes más usados para web y permite la instalación de extensiones que amplían sus funcionalidades para cualquier caso concreto.
  - **GIT**: es una aplicación para el control de versiones. Permite organizar versiones y trabajar en distintas partes de una aplicación simultáneamente, con herramientas para después juntar todo el trabajo. Se ha utilizado para este propósito y como método para publicar la aplicación, cosa que se comentará más adelante.
  - **Trello**: es una aplicación web que permite organizar elementos en tarjetas, listas y tableros. Se ha utilizado para organizar las tareas a realizar durante la elaboración del trabajo.
- Tecnologías:
  - **Nodejs**: es un entorno en tiempo de ejecución que permite usar JavaScript como servidor. Es de código abierto y multiplataforma y es el entorno sobre el que funcionan las siguientes tecnologías que se mencionarán.
  - **React**: es una librería desarrollada por Facebook que permite construir archivos HTML a partir de instrucciones en JavaScript, proceso

denominado renderizar. Permite organizar la aplicación en componentes autocontenidos que se pueden reutilizar y ayudar a realizar páginas web complejas de forma más sencilla [3].

- **Nextjs**: es un framework de React que añade funciones como renderización en el servidor o la opción de usar Typescript [4]. Se irán mencionando las funciones en el capítulo 3 a medida que resulten relevantes.
- **TypeScript**: es un lenguaje de programación compilado basado en JavaScript. Su principal característica es el añadido de tipado fuerte a JavaScript. Gracias a TypeScript, es más fácil localizar errores y problemas que puedan afectar a la aplicación.
- **Nookies**: es una librería que permite manejar de forma sencilla las cookies. Se utilizará principalmente para la gestión de la sesión del usuario.
- **Tonejs**: es una librería que permite la creación de instrumentos virtuales a partir de muestras de sonido. Se utilizará para los ejercicios de oído y para reproducir los sonidos de la guitarra.

- Plataformas:

  - **Heroku**: es una plataforma de alojamiento para aplicaciones web, servidores y otras soluciones. En este trabajo solo se utilizará para publicar la aplicación. Una función muy útil que proporciona es la publicación automática de las nuevas versiones que se creen con Git.
  - **Firebase**: es una plataforma desarrollada por Google que ofrece alojamiento y herramientas para aplicaciones [5]. En este trabajo se usarán dos de sus principales utilidades, Firestore Database para almenar todos los datos de los usuarios y de los ejercicios; y Firebase Auth, para gestionar las cuentas de usuario, así como su registro e inicio de sesión.

4

Análisis y Diseño

4.1. Metodología empleada

Para realizar cualquier proyecto, especialmente uno técnico como es el desarrollo de una aplicación, hay que tener una buena organización y planificar como se pretenden alcanzar los objetivos. Hay diferentes formas de organizar un proyecto y para este trabajo de fin de grado se ha optado por seguir una metodología ágil, más concretamente Kanban [6].

Las metodologías ágiles son aquellas que se basan en ciclos de trabajo cortos. Son ideales para aplicaciones como la de este trabajo de fin de grado porque reducen el trabajo a tareas sencillas que permiten ven un progreso rápido y constante del proyecto. Además, al trabajar en ciclos, ofrece más flexibilidad ante los problemas que puedan surgir durante el desarrollo.

De entre todas las metodologías ágiles disponibles, se ha elegido Kanban. Esta metodología se basa en columnas con tarjetas. Cada columna representa un estado posible para las tareas: pendientes, en progreso, completadas... Cada tarjeta representa una tarea a realizar y se va moviendo entre columnas a medida que cambia su estado. Todo esto permite la organización del trabajo de forma muy visual, desglosa un proyecto grande en muchas tareas pequeñas y más fáciles de abordar y permite ir adaptándose a las dificultades del proyecto. Para poner en funcionamiento esta metodología se va a utilizar Trello [7]. Trello es una herramienta que ya funciona con tarjetas similares a las descritas anteriormente y que permite organizarlas por columnas, haciendo muy fácil la puesta en marcha de

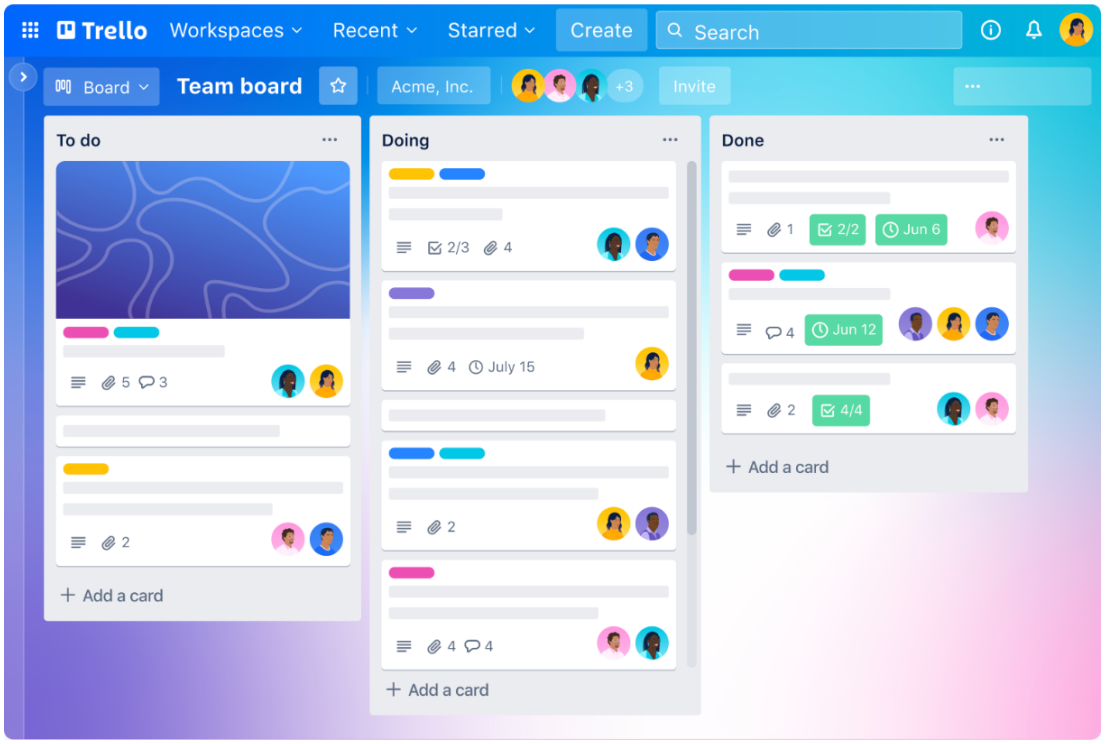

Figura 4.1: Ejemplo de tablero de Trello.

Kanban.

Por último, también es importante tener controladas y organizadas las diferentes versiones de la aplicación a medida que se realizan las tareas de Kanban, por lo que se va a usar una herramienta de gestión de versiones, en este caso Git. Git permite almacenar diferentes revisiones de la aplicación en un árbol y además permite visualizar fácilmente los cambios que han ocurrido entre versiones.

4.2. Requisitos de la aplicación

Para la realización de la aplicación, se van a definir unos requisitos. Estos requisitos tienen como objetivo definir la aplicación de forma concreta y ayudar al desarrollo de la apliacción. Hay dos tipos de requisitos:

- **Requisitos funcionales**: son aquellos requisitos que representan una función en concreto de la aplicación. Definen el comportamiento de la aplicación y las formas de interactuar con ella.
- **Requisitos no funcionales**: son aquellos que más que definir algo en concreto de la aplicación, definen sus propiedades, como cuestiones de seguridad

o de diseño.

A continuación se detallarán los requisitos funcionales (tabla 4.1) y los requisitos no funcionales (tabla 4.2)

Tabla 4.1: Requisitos funcionales

Requisito	Nombre	Descripción
RF1	Registro de usuario	La aplicación debe permitir que cada usuario se cree una cuenta personal
RF2	Recuperación de cuentas	Se debe ofrecer un método para recuperar una cuenta
RF3	Página del usuario	La aplicación debe contar con una página donde el usuario pueda ver sus datos y progreso
RF4	Calendario	La página del usuario debe tener un calendario
RF5	Seguimiento de ejercicios	El usuario podrá visualizar que ejercicios ha realizado y cuando las ha realizado
RF6	Puntuaciones	En la página del usuario se mostrarán las puntuaciones de los diferentes ejercicios
RF7	Progreso del usuario	El usuario deberá tener estadísticas que le muestren su progreso a lo largo del tiempo
RF8	Página de ejercicios	La aplicación debe contar con una sección que muestre todos los ejercicios disponibles
RF9	Ejercicios de teoría	La aplicación tendrá ejercicios dedicados a practicar la teoría y fundamentos de la guitarra
RF10	Ejercicios de oído	La aplicación tendrá ejercicios dedicados a mejorar el oído musical
RF11	Personalización de ejercicios	El usuario podrá personalizar los ejercicios basándose en sus necesidades

Tabla 4.1: Requisitos funcionales

Requisito	Nombre	Descripción
RF12	Página de teoría	La aplicación debe contar con una sección que permita al usuario aprender los conceptos que se usarán en los ejercicios
RF13	Teoría de guitarra	La teoría de guitarra se debe mostrar de una forma que sea fácil y sencilla de entender y practicar
RF14	Interfaz de guitarra	La aplicación debe conta con una interfaz de guitarra que se asemeje al instrumento real y que permita la realización de los ejercicios de una forma intuitiva
RF15	Modos de la interfaz de guitarra	La interfaz de la guitarra se deberá adaptar a los diferentes ejercicios, presentando una funcionalidad acorde al ejercicio a realizar
RF16	Comprobación de errores	La interfaz de la guitarra deberá mostrar los errores que ha cometido el usuario, y permitir su corrección
RF17	Sonidos	La aplicación debe contar con una forma de reproducir sonidos de los instrumentos
RF18	Interfaz de guitarra reproducible	La interfaz de guitarra debe poder escucharse en cualquier momento con el estado que haya introducido el usuario
RF19	Cambio de contraseña	El usuario deberá poder cambiar la contraseña de su cuenta

Tabla 4.2: Requisitos no funcionales

Requisito	Nombre	Descripción
RNF1	Acceso a la base de datos	El acceso a la base de datos se realizará solamente desde el servidor
RNF2	Adaptabilidad	La aplicación debe adaptarse al dispositivo que use el usuario
RNF3	Selección de mano dominante	El usuario deberá poder seleccionar cuál es su mano dominante y la aplicación adaptarse

4.3. Diseño de la aplicación

Para desarrollar una aplicación es muy importante planificar con antelación lo que se va a hacer. Esto crea un camino a seguir durante el desarrollo de la aplicación, que asegura la implementación de todas las funcionalidades previamente definidas. La mayor parte de la aplicación va a consistir en páginas simples que conectan unas con otras, un esquema básico de esta estructura se puede ver en la figura 4.2.

La página de ejercicios permitirá navegar entre los siguientes ejercicios:

- Ejercicios de guitarra:
  - Ejercicio para aprender las notas de las cuerdas: los usuarios tendrán la opción de recibir notas en la interfaz a identificar o colocar ellos las notas en la propia interfaz.
  - Ejercicio para aprender acordes: al igual que con las notas, los usuarios podrán identificar acordes ya construidos o colocarlos ellos mismos en la guitarra.
  - Ejercicio para aprender escalas: de la misma forma que los anteriores ejercicios, los usuarios podrán identificar o colocar escalas.
- Ejercicios de oído:
  - Ejercicio de intervalos: los usuarios recibirán en la interfaz de la guitarra una nota y tendrán un audio a su disposición con dos sonidos. El primer sonido se corresponde a la nota mostrada, el segundo es el que marca el intervalo a adivinar.
  - Ejercicio de acordes: en este ejercicio se recibirá un sonido con un acorde, los usuarios deberán adivinar qué tipo de acorde es.

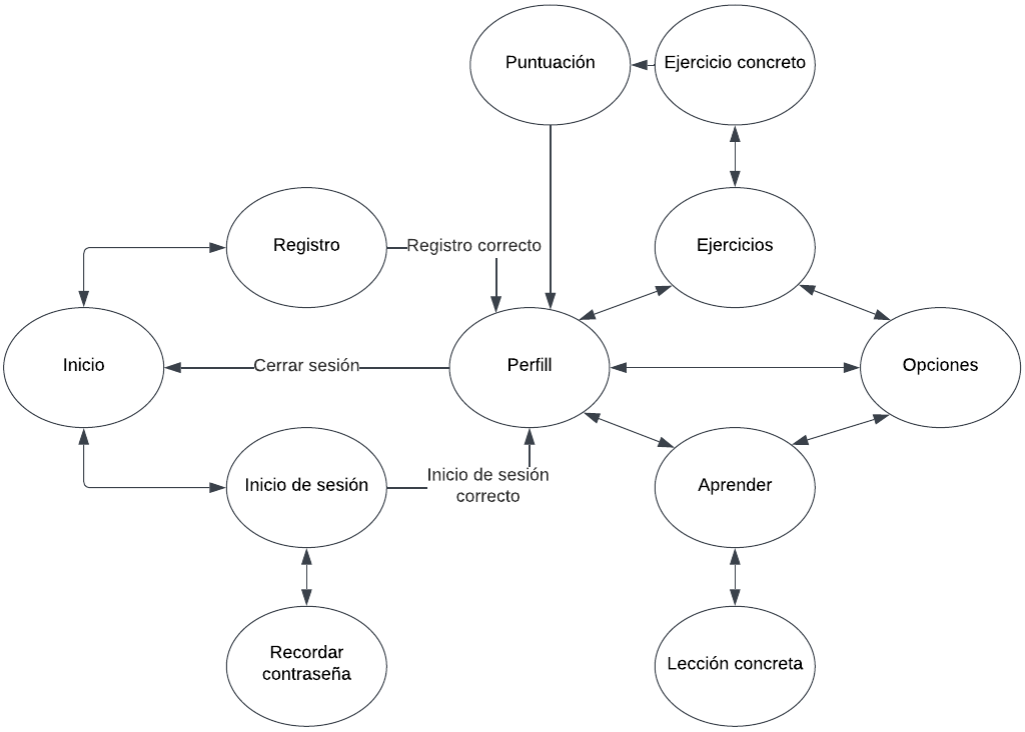

Figura 4.2: Esquema básico de la aplicación.

Todos estos ejercicios cuentan con distintas categorías para practicar diferentes acordes, escalas o cualquiera que sea el tema del ejercicio. También contarán con una última opción que permite a los usuarios crear sus categorías personalizadas, para practicar lo que deseen.

La aplicación contará también con una sección de teoría (la página de aprender), donde repasar de un vistazo los conocimientos pertinentes a cada ejercicio. Para ello se mostrarán siempre que sea posible, en la propia interfaz de la aplicación, haciendo más sencillo su aprendizaje.

Por último, la aplicación contará con un servicio de cuentas de usuario donde ver su progreso y actividad. Se mostrará un calendario con los días que han practicado y que tipo de ejercicio han realizado. También se mostrarán las puntuaciones máximas de cada ejercicio, así como la media en los últimos 7 días y la media en los últimos 30 días, para ayudar a medir el progreso.

A parte de este diseño básico, hay elementos que requieren de algo más de desarrollo previo. Para esto se han creado diagramas UML de las clases que serán necesarias para estos elementos. Se han preparado tres, uno para los instrumentos que reproducirán los ejercicios, otro para la interfaz de la guitarra y un último para el calendario.

Para los instrumentos (figura 4.3) tendremos una clase "instrument"que inicializa los sonidos dependiendo de con qué instrumento se la llame. Después tendremos dos clases que heredan de "instrument"; "Guitar", que simularía una guitarra y "Piano", que simularía un piano.

Para la interfaz de guitarra (figura 4.4) tendremos la clase "GuitarInterface", que dibujará la guitarra usando componentes de tipo "Fret", que es la clase que se encargará de dibujar los trastes y dibujará las cuerdas usando componentes de tipo "StringNote". Todas estas clases tendrán acceso a la estructura de datos de la guitarra y a las distintas opciones, tanto de funcionamiento como del usuario, para dibujar sus elementos acorde a ello. Todas las clases tienen un método "return"porque realmente son componentes de Next.js que dibujan elementos en la web.

Para el calendario (figura 4.5) tendremos la clase "Calendar", que dibujará el calendario en sí, usando los componentes "CalendarDay", que dibujan cada día en concreto. La clase "DayData"se utilizará para encapsular los datos de cada día del conjunto de datos del mes. Al igual que la interfaz de guitarra, algunas clases tienen el método "return"porque son componentes de Next.js que dibujan elementos en la web.

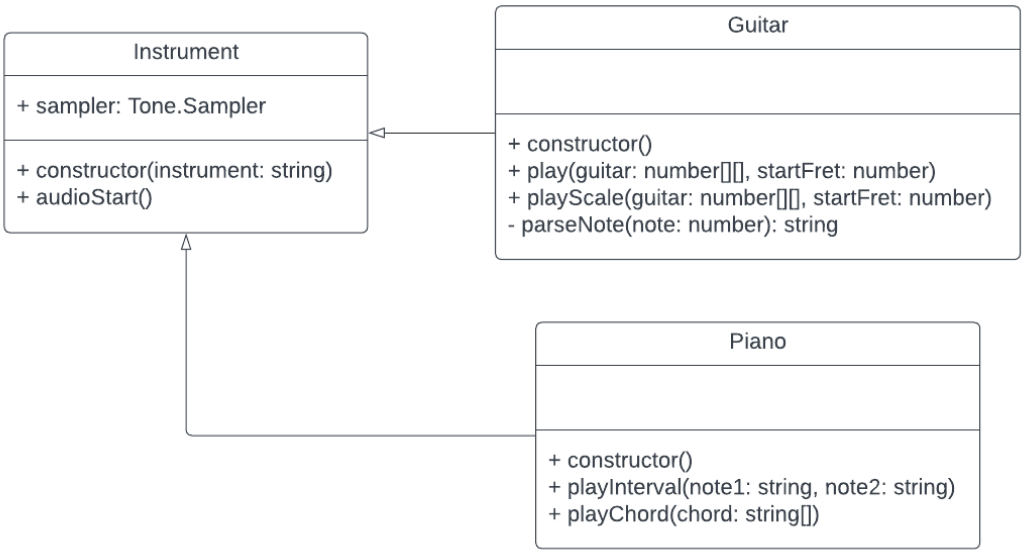

Figura 4.3: Diagrama UML de los instrumentos.

Figura 4.4: Diagrama UML de la interfaz de guitarra.

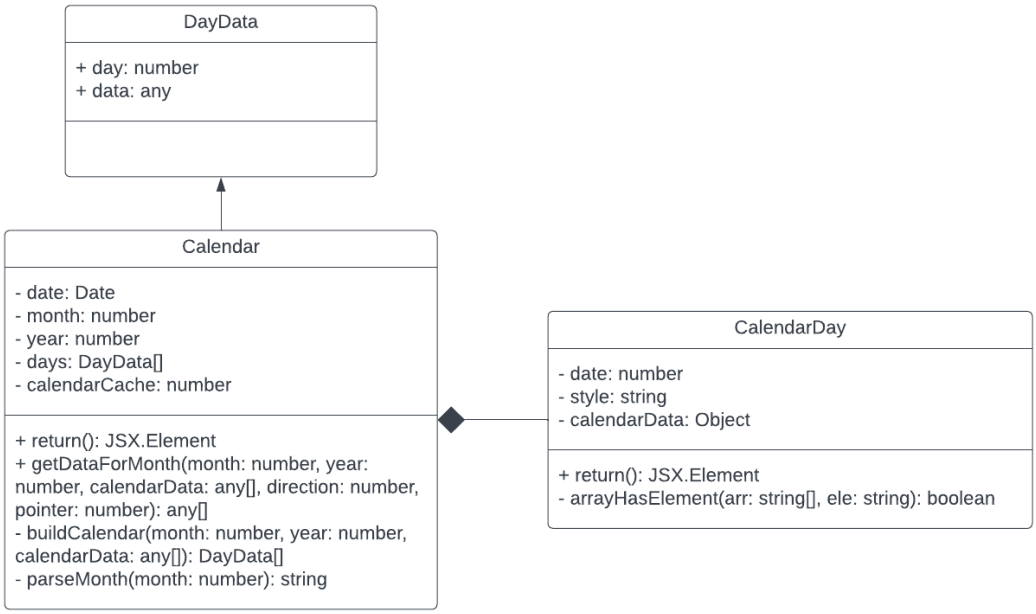

Figura 4.5: Diagrama UML del calendario.

5

Descripción técnica

5.1. Introducción

Como se ha mencionado anteriormente, la aplicación está hecha con Nextjs, un framework de React. La mayor diferencia con una web que no usa React, es que aquí no se escriben los archivos HTML directamente. Se programan en JavaScript unas instrucciones que le dicen a Nextjs y React, como deben construir el HTML.

Para conseguir esto y mantener la legibilidad del código, React usa una extensión de JavaScript llamada JSX. JSX permite escribir HTML directamente en el código JavaScript, evitando todas las llamadas a "createElement()" o "appendChild()" que provocarían que el código fuese muy difícil de leer [8]. Además, al tener HTML incrustado dentro de JavaScript, puedes, por ejemplo, hacer bucles con elementos u ocultar y mostrar elementos de forma muy dinámica y sencilla.

Por otra parte, Nextjs lleva esta funcionalidad un paso más adelante, ya que permite elegir donde se realiza el proceso de construcción del HTML, en el cliente, en el servidor, o de forma híbrida en ambos. Además, es capaz de detectar si el contenido de una página puede o no cambiar en el momento de renderizarla. De no cambiar (ser estática), mientras se inicia el servidor, crea una copia prerenderizada, que es la que se manda a los clientes cuando intentan acceder a esa página, reduciendo así la carga tanto del servidor como del cliente.

Para la aplicación, salvo las páginas que sean estáticas, se ha preferido la renderización en el servidor, para acelerar el funcionamiento de la aplicación y para eliminar completamente cualquier acceso a la base de datos desde el cliente.

5.2. Guitarra

Esta es, sin duda, la parte más importante de la aplicación. Y para crearla se han utilizado dos características muy importantes de React: los componentes y los estados. El mástil de la guitarra está compuesto por trastes y pulsando las cuerdas en cada traste es lo que hace que cambie la nota que da dicha cuerda. Para construir la interfaz se ha creado un componente que la engloba entera, un componente que representa cada traste y un último componente que representa cada nota de cada cuerda en ese traste.

Los componentes en React son como páginas web completas que se pueden incluir dentro de otras páginas o componentes. Una vez creados, funcionan como una etiqueta HTML e incluso pueden recibir parámetros u otros componente o elementos como hijos.

Para almacenar toda la información de la interfaz, se hace uso de los estados. React permite definir una variable como estado. Esto se hace para almacenar datos que afectan a como se visualiza la página, ya que, al actualizar una variable de estado, React vuelve a renderizar las partes afectadas de la web para reflejar los cambios que sean necesarios.

El componente que engloba toda la interfaz recibe una matriz como variable de estado que representa el mástil de la guitarra. También recibe otros parámetros de configuración como el traste a empezar, el número de trastes a mostrar, el modo de interacción y la mano dominante del usuario. Este componente se encarga de construir la interfaz, controlar las cuerdas silenciadas y propagar las variables de configuración a los componentes hijo, en este caso, los trastes.

El componente del traste recibe los mismos parámetros que el componente general, salvo el número total de trastes. En vez de eso, recibe un índice que indica en que traste está, ya que el trabajo de este componente es dibujar el mástil con los puntos en los trastes correctos. Además, también se encarga de crear las notas de cada cuerda en ese traste, invirtiendo su orden si el usuario es zurdo. Por último, se pasa a cada nota los mismos parámetros, salvo que el índice ahora indica la cuerda, de más grave a más aguda.

Por último, tenemos el componente de la nota. Este es el más complejo, ya que es el que contiene la lógica de cómo funciona la interfaz. Se encarga de dibujar las cuerdas sobre los trastes y permite seleccionar las notas en la interfaz. Para ello, cuenta con tres modos de funcionamiento, además de la opción de desactivar completamente la interacción:

- Modo de una nota: este modo solo permite una única nota activa en la interfaz. Si la nota pulsada está ya seleccionada, la deselecciona, de lo contrario, limpia toda la interfaz y selecciona la pulsada.

- Modo de múltiples notas: este modo permite múltiples notas por cada cuerda. Cada nota funciona como un interruptor, independientemente del estado del resto de notas.
- Modo normal: este modo es el que simula el uso de una guitarra real. No se permiten múltiples notas por cuerda salvo que haya una cejilla y solo si se encuentran después de esta. Este modo también permite agrupar notas contiguas en cejillas, así como ampliarlas y reducirlas, y automáticamente deselecciona las notas que no pueden llegar a sonar.

El estado de la guitarra se almacena en la matriz anteriormente mencionada en forma de números enteros, cada uno significando un tipo de nota o su ausencia:

- Un 0 indica que la nota no está seleccionada.
- Un 1 indica que la nota está seleccionada.
- Un 2 indica que la nota está seleccionada y que se trata de la parte interior de una cejilla.
- Un 3 indica que la nota está seleccionada y que se trata del extremo izquierdo de una cejilla.
- Un 4 indica que la nota está seleccionada y que se trata del extremo derecho de una cejilla.

Además, estos números pueden tener un 1 en las decenas. Esto ocurre si el usuario comprueba su respuesta y contiene errores, lo que añade 10 a cada nota colocada incorrectamente. De esta forma, el componente puede mostrar al usuario sus errores.

5.3. Ejercicios

Esta aplicación contiene diferentes tipos de ejercicios y por cada tipo, muchos ejercicios diferentes. Para evitar código redundante, se ha agrupado los ejercicios en cuatro tipos, dos de guitarra y dos de oído. Además, estos ejercicios hacen uso de una característica de Nextjs muy útil, su enrutador.

El enrutador de Nextjs permite entre otras cosas, tener URLs variables que posteriormente podemos leer. Gracias a esta funcionalidad, podemos crear las páginas en una ruta cuyas carpetas tienen el nombre entre corchetes. El nombre de estas carpetas sirve como variable a la hora de crear el enlace y con el enrutador podemos acceder a ese dato de la URL en el momento en que se renderiza la

página. De esta forma y con parámetros de URL clásicos, podemos filtrar todos los tipos y combinaciones posibles de ejercicios a los cuatro tipos anteriormente mencionados.

Los dos tipos de ejercicio de guitarra funcionan de forma muy similar. El primer tipo muestra un enunciado con el concepto a colocar en la interfaz, y esta permite introducirlo. El segundo tipo muestra en la interfaz un concepto y le pide al usuario que lo identifique con un campo de texto.

Ambos tipos reciben los ejercicios de la misma base de datos, pero los procesan de forma distinta. Estos ejercicios contienen una representación del mástil, el concepto representado, el traste en el que comenzar la interfaz, el número de trastes a mostrar y el tipo de ejercicio obtenido de la URL. En el ejercicio de colocar, se muestra el concepto y la representación del mástil se utiliza para comprobar la respuesta, en el de identificar esto funciona justo al contrario.

Los dos tipos de ejercicio de oído son un poco más diferentes, aunque funcionan de manera similar. Estos ejercicios contienen un sonido inicial y un sonido secundario, en caso de necesitarlo. También reciben un conjunto de botones que representan las posibles respuestas; y, por último, recibe el tipo de ejercicio obtenido de la URL.

El primer tipo de ejercicio es el general y con solo esos datos permite al usuario reproducir el sonido y elegir la opción que crea correcta. En caso de error, el usuario es notificado y puede responder entre las opciones restantes.

El otro tipo de ejercicio es exclusivo para los intervalos y además de lo mencionado anteriormente, los ejercicios contienen la posición de una nota a mostrar en la interfaz de guitarra, para permitir al usuario ayudarse de su propia guitarra a la hora de hacer los ejercicios.

Todos los ejercicios, los cuatro tipos, registran los aciertos y fallos del usuario y cuando finaliza el ejercicio, le transmiten esa información a la pantalla de puntuaciones, junto con el tipo de ejercicio concreto. En la pantalla de puntuaciones, se muestra el resultado del ejercicio al usuario y se suben los resultados a la base de datos, para posteriormente poder mostrarlos al usuario.

5.4. Cuentas de usuario

Todos los usuarios de la aplicación tienen una cuenta de usuario donde se almacena su progreso. Para gestionar la creación de cuentas y el inicio de sesión, se usa uno de los módulos de Firebase, Firebase Auth. Este módulo permite registrarse a los usuarios de muchas formas posibles, aunque en la aplicación se usa un correo electrónico y una contraseña. También incorpora funcionalidades como recuperar una cuenta si se ha olvidado la contraseña. Para esto, se manda

un correo al usuario que se puede configurar desde el panel de control de Firebase.

Para implementar esto en la aplicación, debemos primero configurar el api de Firebase con unos datos únicos para la aplicación, que te proporcionan en el panel de control. Esto hay que realizarlo dos veces, una para el servidor y otra para el cliente. El cliente de Firebase se utiliza para iniciar sesión y registrarse; mientras que, en el servidor, usaremos Firebase cuando se quiera acceder a una página, para comprobar si el usuario tiene la sesión iniciada y permitirle el acceso o redirigirle a la página de inicio de sesión.

Para registrarse o iniciar sesión solo se necesita llamar a los métodos específicos de Firebase Auth, que gestionan las comprobaciones necesarias y si todo es correcto, devuelven la sesión del usuario con sus datos. De lo que se tiene que encargar la aplicación es de asegurarse que la sesión no caduca y en caso de hacerlo, pedir al usuario que inicie sesión de nuevo. Para esto vamos a volver a usar los componentes y estados de React.

Toda la aplicación va a estar englobada por un componente que gestiona la sesión del usuario. En este componente se ha inicializado un método de Firebase que se llama cada vez que la sesión del usuario sufre algún cambio, entre estos, que inicie sesión o que caduque la sesión. Cada vez que esto ocurra, actualizaremos el usuario en la aplicación, y guardaremos el identificador de la sesión en una cookie, que usaremos en el resto de las páginas para comprobar si el usuario tiene la sesión iniciada y sigue siendo esta válida. Para este propósito se ha usado una librería llamada Nookies, que permite guardar y leer cookies fácilmente. Además, este componente se va a encargar, mediante el uso de un temporizador, de refrescar la sesión cada cierto tiempo.

Tanto el temporizador como el método anteriormente mencionado estarán englobados en un "useEffect". Esto es una herramienta de React que asegura que un código no se ejecute múltiples veces, algo necesario ya que el componente se encuentra en la raíz de la página y está siendo constantemente recargado. Por último, el usuario se almacena como una variable de contexto, que se comporta como una variable global accesible a los componentes hijo.

5.5. Base de datos

En los anteriores apartados se ha mencionado como se inicializa Firebase y que algunas partes de la aplicación, como los ejercicios o las puntuaciones, se encuentran en la base de datos. La base de datos que se ha utilizado es la que proporciona Firebase, llamada Firestore Database. Es una base de datos bastante sencilla, pero perfecta para las necesidades de esta aplicación. Permite guardar colecciones, el equivalente a las tablas, de documentos, el equivalente a las entradas de una tabla.

A esta base de datos se puede acceder tanto desde el cliente como desde el servidor. Si accedes desde el cliente, los permisos y acceso a la base de datos están sincronizados con el módulo de Firebase Auth para mayor simplicidad. Pero en la aplicación, como se ha mencionado anteriormente, accederemos a la base de datos exclusivamente desde el servidor. Debido a que las credenciales del servidor tienen todos los permisos, no se necesita de ninguna configuración especial, se pueden realizar consultas directamente.

La base de datos de la aplicación consta de siete colecciones principales, cinco para las diferentes categorías de ejercicios y dos para almacenar configuraciones y puntuaciones máximas de los usuarios.

- Colecciones de ejercicios:
  - "exercices-theory-notes": contiene los ejercicios de guitarra en los que solo hay una sola nota en la interfaz.
  - "exercices-theory-chords": contiene los ejercicios de guitarra en los que se muestran acordes.
  - "exercices-theory-scales": contiene los ejercicios de guitarra en los que se muestran escalas.
  - "exercices-ear-intervals": contiene los ejercicios de oído que tratan sobre intervalos.
  - "exercices-ear-chords": contiene los ejercicios de oído que tratan sobre acordes.
- Colecciones de usuario:
  - "user-scores": contiene las puntuaciones máximas para cada ejercicio.
  - "user-data": contiene las opciones del usuario.

Estas colecciones de usuario contienen un documento con el id de cada usuario, para poder identificarlos. También cabe destacar que actualmente no se pueden guardar listas de listas dentro de un documento, cosa necesaria para guardar las puntuaciones de cada día para cada usuario. Por este motivo, a parte de las colecciones anteriormente mencionadas, hay una colección por cada usuario con nombre "id-calendar" que almacena estos datos. A todas las colecciones se accede directamente por su nombre completo, por lo que esto, aunque lejos de ideal, no afecta en nada al rendimiento de la aplicación ya que no se tiene que buscar entre un numero de colecciones en crecimiento.

5.6. Perfil de usuario

Cada usuario tiene una página dedicada a ellos, donde pueden ver un calendario con los días que han practicado y con las puntuaciones de los ejercicios.

El calendario muestra todos los días en los que el usuario ha practicado y especifica que tipo de ejercicio ha hecho. Para conseguir esto se han usado dos componentes, uno para el calendario en general y otro para cada día.

El componente del día recibe su día para mostrar, un estilo dependiendo de si es domingo o no y los datos del usuario para ese día. De esos datos se obtiene si ha hecho ejercicios de guitarra u oído y se muestran en el día.

El componente del calendario recibe los datos del historial del usuario y se encarga de navegarlos y generar los meses a medida que el usuario interactúa con el calendario.

Antes de dibujar el calendario, el componente tiene que obtener los datos del usuario que corresponden a ese mes. Para ello, se usa una variable a modo de puntero, que guardará donde ha terminado de leer los datos, de esta forma no hay que recorrer todos los datos cada vez que se cambie de mes. Una vez recopilados todos los datos del mes, estos se utilizan para poblar el calendario.

A la hora de representar el mes, primero comprueba en que día de la semana empieza el mes y rellena los días previos a ese mes con los del mes anterior. A partir de ahí genera todos los días, les asigna el estilo adecuado y sus datos si los hay. Tras esto, rellena los días que sobran al final con los del siguiente mes.

A parte de usar los datos del usuario para el calendario, también se usan para calcular las estadísticas. Se buscan los datos de todos los ejercicios en los últimos 30 días y se hacen las medias. También se hace esto mismo con los últimos 7 días, para poder comparar el progreso. Además, se recogen de la base de datos las puntuaciones más altas de cada ejercicio, ya que cada vez sería más costoso obtenerlas de todos los datos.

5.7. Sonido

Para poner sonido a los ejercicios e interfaz de guitarra se ha usado la librería Tonejs. Con esta librería se han creado los dos instrumentos que se utilizan en la aplicación, una guitarra y un piano.

Para esto se ha creado una clase instrumento que inicializa el instrumento. Esto consiste en indicarle a la librería donde están los sonidos y con que nota se corresponde cada uno. Dependiendo de si se indica guitarra o piano, lo sonidos se leerán de su respectiva carpeta.

Después hay otras dos clases que implementan métodos específicos para cada instrumento. La guitarra tiene un método que simula que se toquen todas las cuerdas a la vez y otro que simula que se tocan todas las notas de una escala en orden. Estos métodos reciben los datos de la interfaz y los procesan para hacer el sonido.

La clase del piano tiene dos métodos que solo se usan en los ejercicios de oído. Uno está dedicado a los intervalos y el otro a los acordes.

Por último, cabe destacar que esta librería tiene que configurar los sonidos de los instrumentos en el cliente, por lo que la inicialización no se puede realizar antes del renderizado del HTML. Por esto, antes de usar cualquier instrumento, se comprueba si se ha inicializado y si no lo está, se hace en ese momento.

5.8. Interfaz de la aplicación

La interfaz de la aplicación es uno de los aspectos más importantes de esta, ya que al tratarse de una herramienta para aprender, su uso no debería interponerse en el aprendizaje. Para evitar esto, la interfaz debe ser simple y fácil de usar, para que el usuario pueda dedicar su tiempo en mejorar, y no en aprender a utilizar la aplicación. Además debe asemejarse lo máximo posible a lo que el usuario está acostumbrado, una guitarra real. Esto ayudará a que sea mucho más fácil transferir el conocimieto adquirido en la aplicación a la práctica real con el instrumento.

A continuación se mostrarán las diferentes partes de la aplicación y se explicarán las decisiones tomadas en su diseño.

Para navegar por la aplicación se ha incluido una barra de navegación lateral, que permite al usuario moverse libremente por toda la aplicación (figuras 5.1 y 5.2). Además, en las figuras se pueden observar los botones de navegación de ejercicios y teoría. Son iguales a través de toda la aplicación e indican su tipo con el icono de guitarra u oído.

Para la elección del ejercicio, los botones son los mismos que los de navegación, con la opción de seleccionar el tipo de ejercicio o, en el caso de la opción personalizable, con botones para activar y desactivar los contenidos que quieres introducir en el ejercicio (figura 5.3).

Para los ejercicios se ha intentado hacer la interfaz de guitarra lo más fiel posible a la realidad, ya que el objetivo último es aplicar los conocimientos a una guitarra real (figura 5.4). Cuando el usuario se equivoca en algún ejercicio, se indican en rojo los errores para que el usuario los pueda corregir (figura 5.5). Los ejercicios de adivinar muestran el enunciado en la propia interfaz.

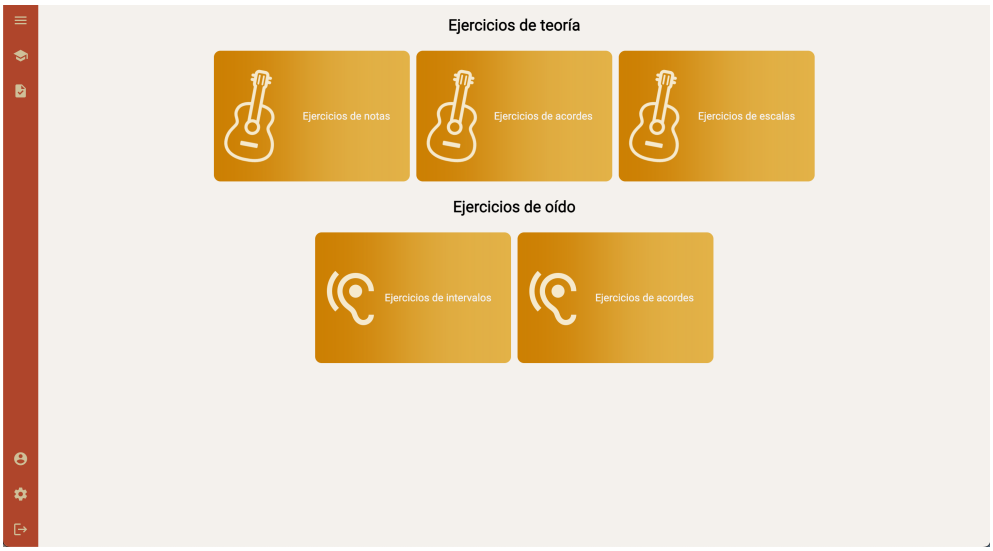

Figura 5.1: Navegación de ejercicios.

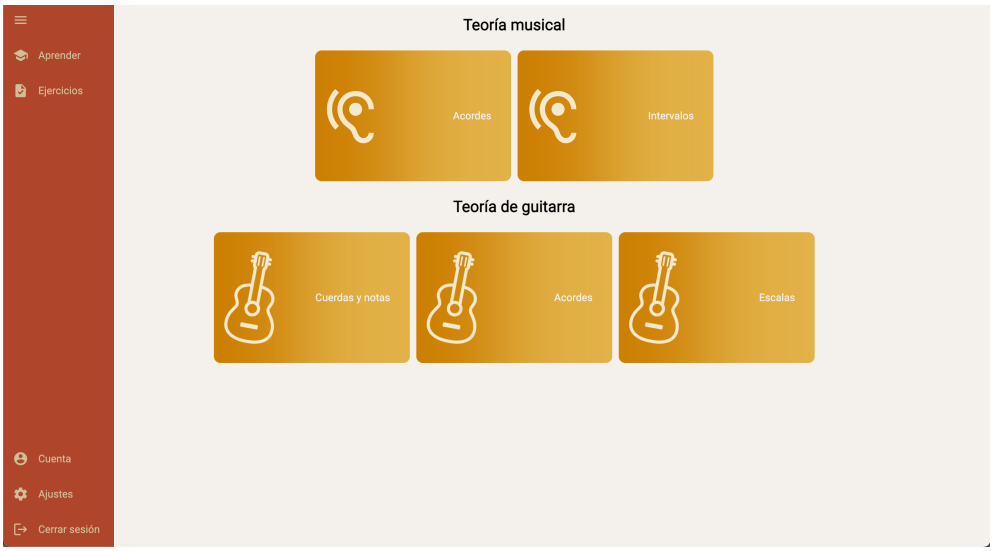

Figura 5.2: Navegación de teoría.

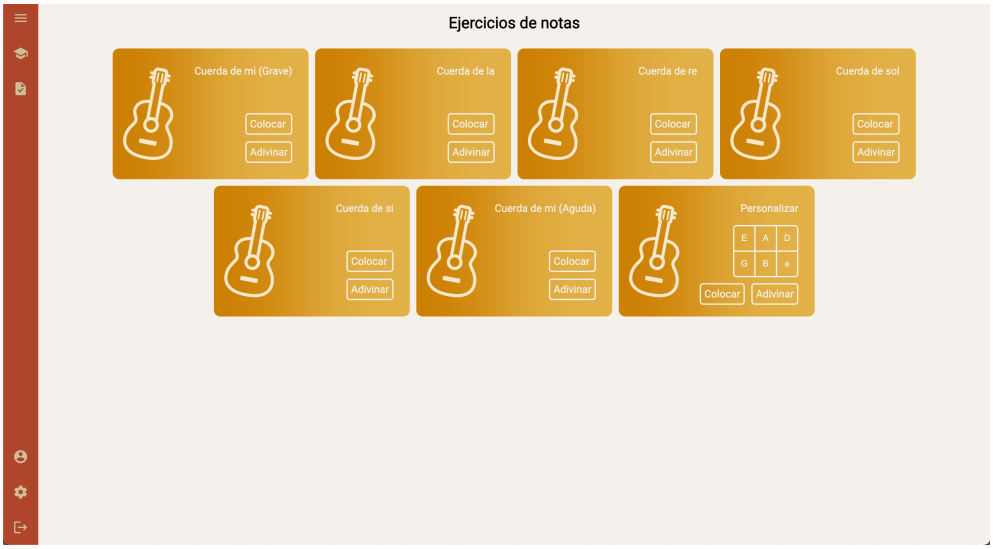

Figura 5.3: Selección de ejercicios.

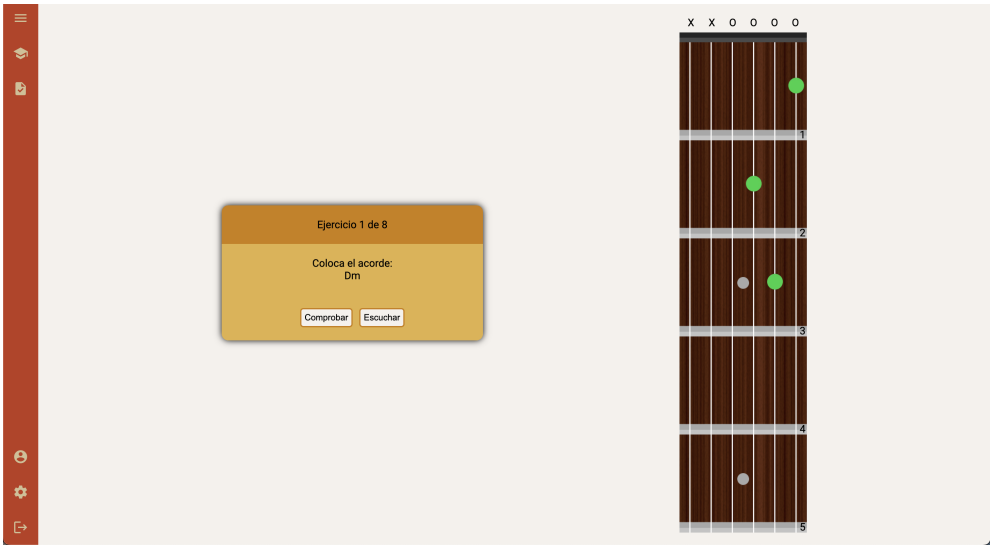

Figura 5.4: Ejemplo de ejercicio de colocar acordes.

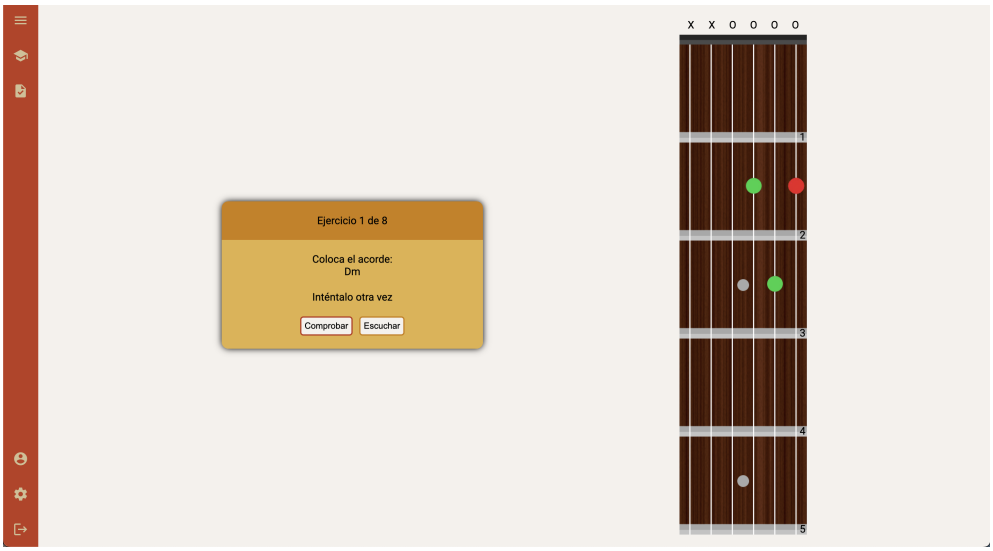

Figura 5.5: Ejemplo de ejercicio con errores.

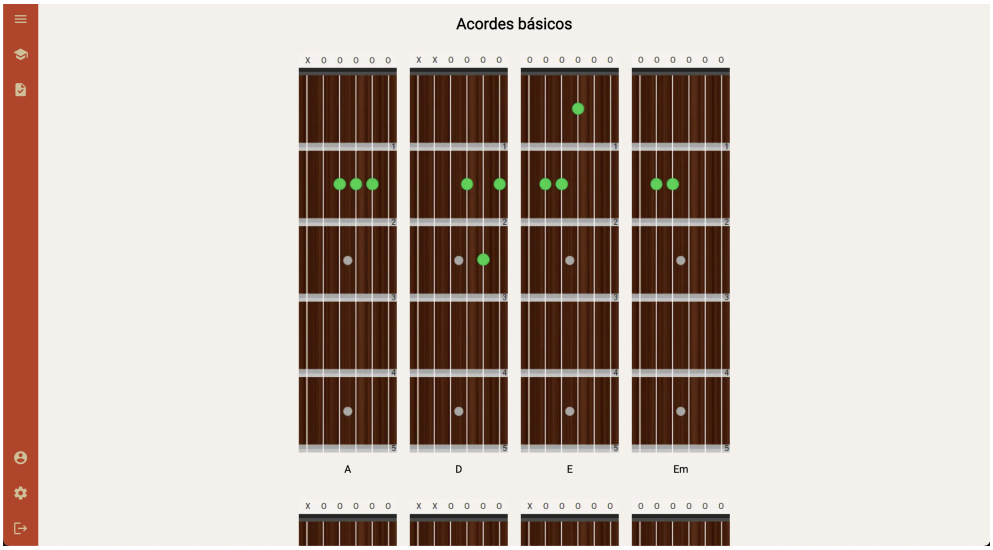

Figura 5.6: Teoría de acordes.

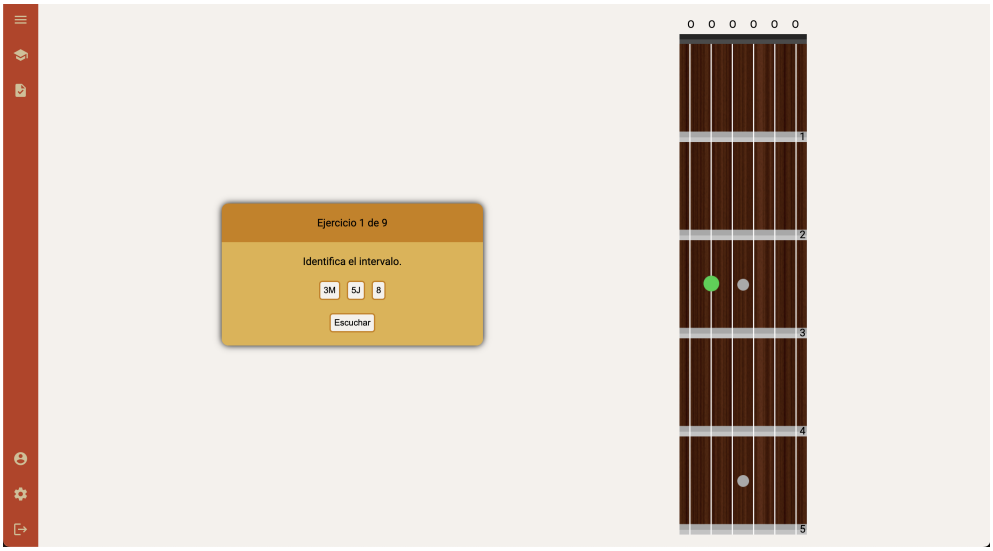

Figura 5.7: Ejercicio de oído de intervalos.

Para la teoría en la aplicación, se ha reutilizado la interfaz de guitarra, haciendo más fácil la aplicación de los conocimientos en los ejercicios (figura 5.6). También se ha reutilizado en los ejercicios de sonido, para facilitar al usuario la posibilidad de probar con su guitarra el ejercicio (figura 5.7).

Para el calendario se ha usado un diseño común con seis semanas. En la esquina de cada día se muestran, con los mismos iconos que en la navegación de la app, que ejercicios se han practicado ese día (figura 5.8).

Las puntuaciones se muestran en la página del usuario. Se muestra una categoría para cada tipo de ejercicio, con la media del mes y los últimos 7 días, como forma de seguir el progreso del usuario. También se muestra la puntuación máxima de cada ejercicio (figura 5.9).

Por último, en la sección de opciones, a parte del cambio de contraseña, se ofrece la opción de la mano dominante (figura 5.10). Cambiar tu mano dominante cambia como se presenta la teoría y la solución de los ejercicios de forma acorde.

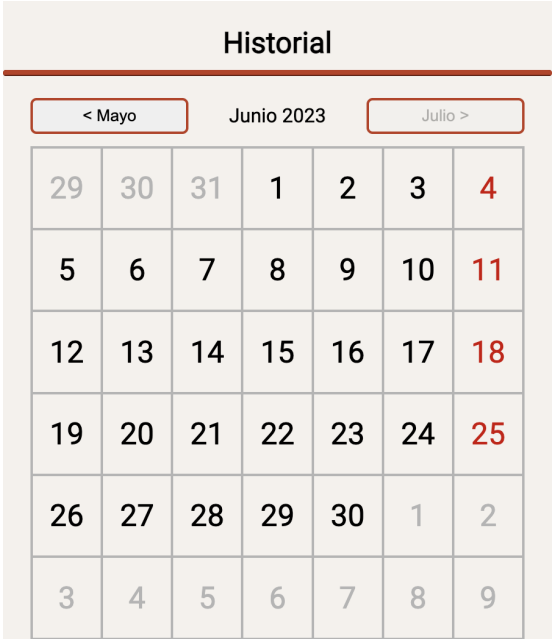

Figura 5.8: Calendario de prácticas.

Figura 5.9: Sección de puntuaciones.

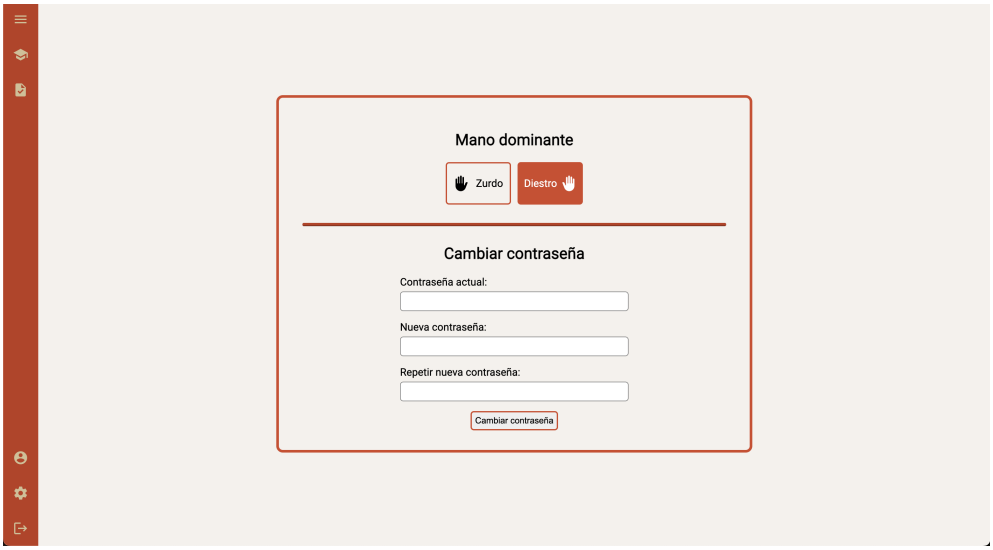

Figura 5.10: Opciones del usuario.

6 Evaluación

6.1. Evaluación técnica

Durante el desarrollo y tras completar la aplicación se han realizado pruebas para asegurarse que todos los sistemas funcionan correctamente. A continuación se detalla como se ha realizado este proceso y que herramientas se han utilizado.

Para probar la funcionalidad del cliente se han usado las herramentas de depuración incluídas en el navegador web Google Chrome. Se ha utilizado el visor de almacenamiento para comprobar el funcionamiento del inicio de sesión con las cookies. También se ha utilizado la consola y el depurador incluídos para probar la mayoría de funciones del cliente. Por último, se ha utilizado el visor de elementos para comprobar la correcta compilación de los archivos HTML.

Para probar la funcionalidad de la parte del servidor, se han aprovechado las herramientas de depuración que incluye Visual Studio Code, así como el compilador de TypeScript, que facilita la depuración del código de JavaScript

Para evaluar la base de datos, se ha utilizado la aplicación Postman, que permite realizar peticiones a servidores y analizar los resultados de estas de forma detallada. Se ha utilizado tanto para probar el correcto funcionamiento de la base de datos como para asegurarse de que es solo accesible por las aplicaciones autorizadas, en este caso, el servidor.

6.2. Evaluación heurística

Una evaluación heurísitca es una técnica que se utiliza para analizar la usabilidad de la interfaz de una aplicación. Se trata de un conjunto de comprobaciones para asegurarse de que la aplicación es usable. Para esta aplicación se van a usar las diez heurísticas de Jakob Nielsen, que describe en su libro Usability Engineering [9]. Las diez heurísticas son las siguientes:

- **Visibilidad del estado del sistema**: La aplicación debe informar al usuario en todo momento de lo que está sucediendo.
- **Coincidencia entre el sistema y el mundo real**: La aplicación debe adaptarse al mundo que conoce el usuario. Debe usar conceptos familares que eviten que el usuario deba buscar su significado.
- **Dale al usuario el control y la libertad**: El usuario debe tener siempre la opción y la libertad de modificar sus acciones y estas deben mostrarse claramente.
- **Consistencia y estándares**: La aplicación debe resolver las situaciones de la misma forma. Ser consistente tanto dentro de la aplicación con su interfaz, como en relación a otras ya existentes, básandose en estándares.
- **Prevención de errores**: La aplicación debe adaptarse a los errores que puedan comenter los usuarios y tener lista una respuesta ante ellos.
- **Minimizar la carga de memoria del usuario**: Se debe minimizar la necesidad de memorizar del usuario, mostrando indicaciones y ayudas con la información necesaria en cualquier momento.
- **Flexibilidad y eficiencia de uso**: La aplicación debe proporcionar herramientas para que todos los usuarios, independientemente de su familiaridad con la aplicación, puedan hacer un uso eficiente.
- **Diálogos estéticos y diseño minimalista**: El diseño tiene que ser claro y priorizar el contenido de la aplicación.
- **Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de los errores**: Lo errores deben expresarse de forma clara y sencilla, ayudando al usuario a solucionarlos.
- **Ayuda y documentación**: Aunque lo ideal es tener una interfaz que no requiera ningún aprendizaje previo, contar con la ayuda necesaria hará que el usuario pueda entender como realizar algunas tareas.

Una vez explicadas en que consisten las heurísticas de Nielsen, se va analizar la aplicación y evaluar cada una ellas, con el fin de detectar que aspectos podrían mejorarse para conseguir una mejor experiencia de usuario.

- **Visibilidad del estado del sistema**: A medida que el usuario interactúa con la interfaz se muestra instantáneamente y el estado actual puede ser escuchado en cualquier momento. Una posible mejora puede ser al escuchar la guitarra, que se muestren las notas a medida que se tocan. Esta mejora ayudarí sobretodo en los ejercicios de escalas, haciendo más sencillo el identificar la nota que está sonando.

- **Coincidencia entre el sistema y el mundo real**: La interfaz de guitarra se ha hecho basándose en una guitarra real y también cuenta con una opción para zurdos. Además otras opciones como la teoría y el calendario de prácticas se han creado asemejándose lo más posible a la realidad por lo que creo que este punto se cumple correctamente.

- **Dale al usuario el control y la libertad**: La navegación en toda la aplicación se ha pensado con este objetivo en mente. El usuario siempre tiene la opción de retroceder en cualquier menú y la barra de navegación permite volver a cualquier sección de la aplicación con un solo click.

- **Consistencia y estándares**: Todos los menús de la aplicación tiene un diseño similar, los cuadros interactivos son todos iguales y tanto para los ejercicios como para la teoría se ha utilizado la misma interfaz de guitarra, haciendo muy fácil la aplicación de los conceptos aprendidos. Aparte de esto, los formularios encontrados en la aplicación siguen los estándares de la mayoría de aplicaciones web y los aciertos y errores en los ejercicios están marcados con colores fácilmente reconocibles por cualquier usuario.

- **Prevención de errores**: Al ser una aplicación de práctica con ejercicios, los errores son casi seguros. Los ejercicios están preparados para ello e incluso situaciones como abandonar ejercicios a la mitad no provocan ningún error visible al usuario.

- **Minimizar la carga de memoria del usuario**: Este es uno de los puntos que la aplicación no cumple parcialmente, pero no es necesariamente un problema. Esto se observa principalmente en la realización de ejercicios, donde no hay ninguna indicación más que el enunciado del ejercicio. Pero tratándose de el objetivo de la aplicación no es un fallo preocupante.

  El resto de la aplicación si que tiene indicado con claridad donde te encuentras y cuales son tus opciones, de forma que el usuario no tiene que recordar donde se encuentra el ejercicio que quiere realizar o el concepto que quiere repasar, por ejemplo.

- **Flexibilidad y eficiencia de uso**: El uso de la aplicación en este momento es bastante simple y no requiere de grandes atajos o funciones avanzadas que simplifiquen usos ya dominados por usuarios más avanzados. La única herramienta que mejora esta eficiencia de uso es la personalización de los ejercicios, que fácilita a los usuarios más avanzados practicar más conceptos a la vez. A medida que la aplicación cuente con más contenido, se podría ampliar con opciones como selección del número de ejercicios o selección de la dificultad de estos.

- **Diálogos estéticos y diseño minimalista**: La aplicación sigue un esquema de colores básico y minimalista, que reutiliza de la misma forma para elementos similares. La información se muestra claramente sin elementos que distraigan la atención del usuario.

- **Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de los errores**: En la aplicación, los errores que realmente puede cometer el usuario se encuentran principalmente durante los ejercicios. Cuando el usuario comete un error, este se resalta en rojo hasta que el usuario lo corrige. En los formularios de la aplicación, cualquier error introducido se ve reflejado con un diálogo que lo explica y ayuda al usuario a corregirlo.

- **Ayuda y documentación**: Salvo la información localizada en la sección de "aprender"de la aplicación, esta no cuenta con ninguna forma de ayuda o documentación que facilite la realización de las tareas que realizará el usuario. Alguna propuesta para resolver este punto se detalla en el apartado de posibles mejoras, en las conclusiones de la memoria.

7

Conclusiones

7.1. Resultados

Ya terminada la aplicación, se cumple el objetivo principal que se planteó al principio del trabajo de crear una herramienta multiplataforma fácil de usar.

La aplicación cuenta con las funcionalidades propuestas en el comienzo de la realización del trabajo. También cuenta con un buen número de ejercicios como para que los usuarios puedan empezar a practicar tanto con la guitarra como el oído.

Los requisitos funcionales y no funcionales se han cumplido y aunque como se ha comentado en la evaluación, hay algunos apartados, principalmente de usabilidad, que se pueden mejorar, el resultado se acerca mucho a lo propuesto.

7.2. Posibles mejoras y futuros trabajos

Aunque la aplicación cumple bastante bien las expectativas presentadas al principio del trabajo, hay algunos detalles que creo que podrían mejorar la aplicación en un futuro.

El primero es el que resulta más obvio, pero son más ejercicios. Más ejercicios, tanto en número como en tipo haría a la aplicación una mejor herramienta para más gente. Y ligado a este punto de los ejercicios, también una posible adición serían más opciones a la hora de personalizar los ejercicios. No solo que los filtre por tipo, si no que también permita diferentes formas de colocar acordes, o intervalos descendentes, por ejemplo.

Relacionado con los ejercicios, y tal como se muestra de forma evidente en la evaluación heurística de la aplicación, se debería añadir una forma de explicación o ejercicio tutorial que ayude al usuario a familiarizarse con la interfaz de los ejercicios. Que explique al usuario como interactuar con la interfaz de guitarra y todas las opciones y posibilidades con las que cuenta. Además se podrían mostrar recordatorios de ciertas funcionalidades si son necesarias para el ejericio en cuestión y el usuario falla en utilizarlas.

Otra parte que se podría mejorar es el calendario. Se podría mostrar detalladamente que ejercicios has hecho cada día y cuáles han sido tus resultados. Habría que encontrar una forma de mostrar los ejercicios personalizados pero el usuario ganaría mucha más información y de mejor calidad.

Y por último se podrían mostrar gráficas de las puntuaciones. Para que el usuario pueda ver su progreso mes a mes o semana a semana. También podría comparar que ejercicios ya ha dominado y cuales sigue necesitando más práctica para centrarse en esos.

Aparte de lo mencionado anteriormente, un futuro trabajo relacionado podría ser una revisión de este mismo proyecto o otras versiones enfocadas en diferentes instrumentos a la guitarra, ya que el piano es el que suele predominar en este tipo de aplicaciones, dejando olvidados otros instrumentos.

Otro futuro trabajo relacionado con la aplicación podría ser realizar pruebas con usuarios. Estas pruebas servirían por un lado para evaluar la usabilidad de la aplicación y por otro lado para evaluar la eficacia de la aplicación a la hora de aprender a tocar la guitarra.

Además, gracias a los conocimientos y experiencias adquiridos en este trabajo, tendría una base para en el futuro poder trabajar en el desarrollo de aplicaciones web, tanto en la parte del cliente como en la parte del servidor.

Bibliografía

[1] TonedEar. Aplicación tonedear. [Online]. Available: https://tonedear.com/

[2] ToneGym. Aplicación tonegym. [Online]. Available: https://www.tonegym.co/site/index

[3] Meta Platforms, Inc. Documentación oficial de react. [Online]. Available: https://reactjs.org/docs/getting-started.html

[4] Vercel, Inc. Documentación oficial de next.js. [Online]. Available: https://nextjs.org/docs/getting-started

[5] Google, LLC. Documentación oficial de firebase. [Online]. Available: https://firebase.google.com/docs

[6] D. Anderson, *Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results.* Prentice Hall, 2003.

[7] Atlassian. Aplicación trello. [Online]. Available: https://trello.com/es

[8] Múltiples autores. Mdn (documentación de html, css y javascript). [Online]. Available: https://developer.mozilla.org/en-US/

[9] J. Nielsen, *Usabillity Engineering.* Academic Press Inc, 1993.

Apéndices

A

Configuración del proyecto

En este apéndice se explicará como preparar el proyecto para su ejecución. Independientemente de estas instrucciones, la aplicación se puede probar en el siguiente enlace: https://guitear-305531c1234d.herokuapp.com. Antes de empezar se recomienda tener instalado Visual Studio Code u otro IDE similar, para facilitar la visualización del código de la aplicación, aunque este paso no es necesario.

Como único requisito, es necesario tener instalada la versión 14.15.5 de Node.js, que es la que usa la aplicación como servidor.

1. El primer paso es descargar los archivos del proyecto y descomprimirlos. Para mayor facilidad se recomienda abrir el proyecto con un IDE , pero el resto de pasos se pueden seguir usando la consola del sistema operativo.
2. Tras tener los archivos listos, y encontrándose la consola dentro la carpeta raíz del proyecto, escribir el comando "npm install", que descargará e instalará todas las dependencias necesarias para el funcionamiento de la aplicación. Si todo se ha instalado correctamente, se debería mostrar el mensaje de la figura A.1
3. Después de esto, el proyecto se puede ejecutar con el comando "npm run dev". Iniciará Node.js y procederá a ejecutar la aplicación, mostrando el mensaje de la figura A.2 si todo ha funcionado correctamente:
4. Por último, queda introducir la siguiente dirección web en su navegador para probar la aplicación: localhost:3000

```
added 492 packages from 363 contributors and audited 494 packages in 8.858s
```

Figura A.1: Resultado del comando "npm install".

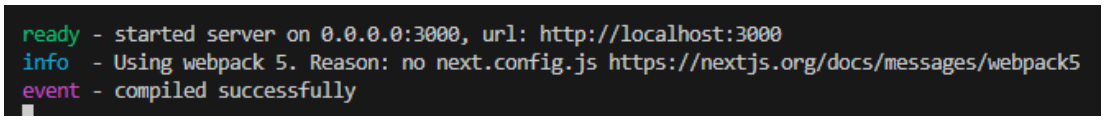

Figura A.2: Resultado del comando “npm run dev”.