UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Entrenamiento auditivo para músicos

ADA - Aplicación de Dictados para la Armonía

Franco Danilo Wanseéle González
Luis Martín Nogueira Techera

Programa en Ingeniería en Computación
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
Diciembre de 2021

UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Entrenamiento auditivo para músicos

ADA - Aplicación de Dictados para la Armonía

Franco Danilo Wanseéle González
Luis Martín Nogueira Techera

Tesis de Grado presentada en Facultad de Ingeniería de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Ingeniero/a en Computación.

Supervisor:
Dra.Ing. Prof. Regina Motz

Montevideo – Uruguay
Diciembre de 2021

RESUMEN

Durante la carrera de música existen cursos los cuales enfrentan a los estudiantes a varios desafíos, siendo uno de los principales la identificación de dictados musicales a través de la audición. Con el objetivo de lograr un buen entrenamiento auditivo, es necesario que los estudiantes puedan practicar con una gran variedad de dictados, siendo ejercitados presencialmente en el dictado de cursos.

En el presente trabajo se desarrolla una aplicación para dispositivos móviles la cual funciona como herramienta de apoyo al dictado de cursos musicales con dichas características. Esta consiste en generar una gran variedad de dictados musicales, de forma aleatoria, obteniendo dictados personalizados a los cursos, con poco esfuerzo por parte del docente.

Debido a que los cursos serán administrados por los docentes a través de la plataforma, éstos podrán guiar el aprendizaje de los alumnos inscriptos, subiendo configuraciones de dictados específicas para lo que desean entrenar.

Junto con los ejercicios generados se brinda el acceso a su solución, la cual consta del dictado graficado en un pentagrama. En base a esto es que a cada ejercicio que realiza el estudiante se le asigna una calificación correspondiente.

La aplicación móvil es desarrollada en *React Native* en el frontend, *NodeJs* en el backend y MongoDB como gestor de base de datos. La aplicación, de nombre ADA, fue publicada para las tiendas de dispositivos Android e iOS, la cual se encuentra disponible de forma gratuita. El desarrollo de la aplicación ADA es completamente de código libre y puede ser accedido en: *https://github.com/francowanseele/Entrenamiento_Auditivo* y *https://github.com/francowanseele/Entrenamiento_Auditivo_server*.

El desarrollo de esta herramienta se realizó en colaboración con un equipo interdisciplinario de docentes y estudiantes de la Escuela Universitaria de Música-Udelar y de la Carrera de Tecnólogo en Jazz y Música Creativa de la

Universidad Tecnológica del Uruguay.

Palabras clave:
Dictados musicales, Entrenamiento auditivo, Aplicación móvil, Generación aleatoria.

Tabla de contenidos

Capítulo 1

Introducción

Uno de los primeros desafíos con los que se enfrentan algunos estudiantes de música al momento de iniciar su carrera es la prueba de admisión, con la cual se evalúa que el estudiante cuente con una formación musical básica. Una sección de dicha prueba consiste en escuchar dictados musicales, y transcribirlos a una hoja pentagramada. La habilidad y el conocimiento necesario para afrontar este tipo de ejercicios es algo que se suele adquirir con la práctica y difícilmente se puede entrenar por cuenta propia. Durante la carrera de música existen cursos los cuales enfrentan a los estudiantes a varios desafíos, siendo uno de los principales la identificación de elementos musicales formados por figuras y notas, a través de la audición. Con el objetivo de lograr un buen entrenamiento auditivo, es necesario que los estudiantes puedan practicar con una gran variedad de dictados.

Para facilitar estos procesos de aprendizaje, el diseño de los cursos tiene un alto componente de ejercitación presencial en dictados musicales. Esto implica que el docente debe pensar en dictados, los cuales cumplan con las características que desea practicar con los alumnos y posteriormente reproducirlos con algún instrumento. Además esto presenta un fuerte carácter de subjetividad, ya que al momento de construir estos dictados, si bien dos profesores diferentes pueden querer entrenar el mismo aspecto, uno puede agregarle mayor complejidad que el otro. Esto implica que al momento de evaluar el desempeño de un estudiante, va a depender fuertemente de quien desarrolle la evaluación y por quien fue entrenado previamente.

A estos aspectos se le suma el hecho de que en el dictado de clases no presenciales hay una enorme pérdida de calidad formativa al verse afectada la

posibilidad del entrenamiento dirigido por el docente.

Para salvar estas limitaciones se desarrolla una aplicación para dispositivos móviles, la cual genera una variedad de dictados musicales, permitiendo que los estudiantes puedan entrenar su oído.

En el ámbito del dictado de un curso, el docente tiene la posibilidad de subir a la plataforma una configuración que corresponda con los objetivos que quiera entrenar en sus alumnos, y en base a esta, la aplicación genera dictados musicales de forma aleatoria, que son accedidas por los estudiantes. De esta forma, no solo se cubre la necesidad de generar ejercicios con poco esfuerzo del docente, sino que también se obtienen dictados personalizados a los cursos dictados.

Por otro lado, gracias a la generación aleatoria de dictados por parte de la aplicación, no existen variables de subjetividad, lo cual hace de la aplicación una herramienta de utilidad para ser utilizada como un recurso en el entrenamiento auditivo de cara a evaluaciones. Estas evaluaciones también contemplan la prueba de admisión para los estudiantes de ingreso ya que tienen la posibilidad de acceder a un curso inicial, previamente establecido, y así tener una forma de practicar.

Se contempla además el escenario en el que un estudiante quiera ejercitarse de forma autodidacta. En este caso el usuario es capaz de crear una configuración particular o incluso utilizar ciertas configuraciones de otros cursos, dado que las configuraciones dadas de alta en el sistema son publicas y de libre acceso por el resto de los usuarios.

Para cualquiera de las situaciones presentadas, la aplicación genera la solución en un pentagrama para que los estudiantes puedan corroborar la correctitud del ejercicio. Esto permite, desde el lado del profesor, monitorear la actividad de sus alumnos y desde el perfil del alumno, esto le brindará un feedback de cómo ha sido su evolución en los ejercicios planteados.

1.1. Marco metodológico

El marco metodológico utilizado en el presente trabajo consistió, principalmente, en técnicas de investigación cualitativas junto a un desarrollo basado en el diseño. Estos fueron encuentros con docentes y estudiantes de la carrera

de música de la UTEC[1], las cuales consistieron en presentar el trabajo e ideas desarrolladas y en base a estas abordar necesidades específicas del área de interés. La información recaudada era incorporada al desarrollo y presentada en futuros encuentros.

Adicionalmente se relevó información secundaria. Esto consistió en estudiar trabajos y aplicaciones relacionados al tema, lo cual permitió analizar la forma en que otras aplicaciones abordaban problemáticas similares y evaluar su posible reutilización.

1.2. Público de interés

Se considera como público de interés o *stakeholders* tanto a los usuarios directos de la aplicación como a toda entidad la cual tiene una interacción directa o indirectamente con el software. Se hará foco en aquellos *stakeholders* que son importantes en base a los objetivos de esta primera etapa del desarrollo, de la cual consiste el presente trabajo académico.

Este proyecto tiene como propósito construir una aplicación móvil para abordar los problemas mencionados en la sección anterior. Además de tener presente dicho objetivo a lo largo de todo el proyecto se tiene que tener definido el público objetivo, que son los usuarios o entidades al los que esta dirigido dicho software, ya que en base a estos, se priorizarán las funcionalidades a desarrollar, se analizará en mayor o menor medida el flujo de uso de ciertos usuarios e incluso determinará el modelado de la realidad. Además, diseñando de forma personalizada a las necesidades de los distintos públicos (estudiantes, docentes, músicos autodidactas) se espera que al cubrir sus necesidades, su satisfacción de uso pueda ser un elemento de recomendación fuerte para lograr llegar a más usuarios.

1.2.1. Público objetivo

Al trabajar en conjunto con docentes de música, concretamente pertenecientes a un instituto educativo, las funcionalidades serán definidas teniendo como objetivo cubrir sus necesidades y la de sus estudiantes. Si bien la aplicación se desarrolla teniendo en cuenta a los usuarios que deseen entrenarse de

[1]Universidad Tecnológica del Uruguay.

forma autodidacta, o incluso a profesores particulares, se entiende que el público objetivo son los docentes y estudiantes de música pertenecientes a institutos educativos. Esto es así porque se considera que dentro de dichos institutos es donde se encuentra la mayor necesidad del entrenamiento auditivo ya que es algo necesario que se entrene constantemente en profesionales del área. Además uno de los problemas a atacar con la aplicación es que futuros estudiantes de la carrera puedan entrenarse para pasar la prueba de admisión.

Por lo tanto, los puntos anteriores definen que el público objetivo serán personas mayormente universitarias, o por entrar a la educación terciaria. Esto, sumado a que el entrenamiento con dictados rítmicos y melódicos es algo que se hace gran énfasis en etapas tempranas de la carrera de música (aunque si es verdad que es un elemento interesante a seguir perfeccionando a lo largo de la carrera) se tiene que la mayor cantidad de usuarios serán personas jóvenes. Esto permite que el diseño y flujo de uso de la aplicación pueda contener elementos similares a otras de gran éxito en este tipo de usuarios, y de esta forma, al tener elementos familiares, le facilitaría al usuario la comprensión y uso de la aplicación.

Por otro lado se tiene a los docentes, usuarios claves para motivar a los estudiantes en el uso de la aplicación. Si bien la aplicación apunta concretamente al entrenamiento auditivo (funcionalidad enfocada a los usuarios estudiantes), será el docente quien decida si es una buena herramienta o no para ser usada en un curso educativo y por ende que a los estudiantes se les sugiera su uso. Si la configuración de los dictados, desde el lado del docente, logra abarcar todos los elementos musicales necesarios, además de ofrecer un correcto manejo de los mismos, la aplicación será capaz de generar dictados abarcando una gran cantidad de temáticas de un curso de música. Esto hará que la aplicación sea una buena herramienta que funcione como complemento al dictado de un curso.

Como último punto se tiene a las instituciones educativas. Teniendo en cuenta de que la aplicación busca entrenar a los estudiantes para que cuenten con un cierto conocimiento básico al momento de dar la prueba de ingreso, las instituciones educativas tendrán un rol fundamental en la difusión y recomendación de la herramienta. Por tal motivo es importante que la herramienta cuente con un curso inicial el cual esté abierto a que cualquier usuario estudiante se pueda inscribir y así acceder a las configuraciones de dictados (dadas de alta por docentes de música o estudiantes avanzados de la carrera) las cuales

les servirán como ejercicios de entrenamiento para la prueba de ingreso.

1.2.2. Otros públicos

En base a los objetivos planteados es que se definió el público objetivo, el cual forma parte de los *stakeholders* de la aplicación. Se entiende que a medida que el software evolucione y se vayan solucionando los problemas abordados, el público objetivo cambiará y se le dará foco a otros *stakeholders*. Un ejemplo de esto son los estudiantes que desean entrenarse de forma autodidacta y profesores particulares, los cuales, si bien se los contempla en las funcionalidades desarrolladas en la aplicación, no fueron el principal objetivo abarcar sus necesidades.

1.3. Resultados obtenidos

El presente trabajo tuvo como clientes institucionales a la UTEC, representada por docentes de la carrera de música, y a la EUM, a quienes se les presentó el proyecto como herramienta para mejorar la educación musical. Esto desencadenó en la formación de un equipo interdisciplinario, con la participación Belén Algorta y Juan Diego Fernández como estudiantes avanzados de la carrera de música y Agustín Pardo como docente, en el cual estuvo a cargo del desarrollo del proyecto. Esto dio como resultado la aplicación "ADA - Entrenamiento Auditivo"[2], con la cual se brinda una solución a las necesidades planteadas. Además el proyecto es de código abierto y se encuentra disponible en en *github*[3].

La plataforma cuenta con dos flujos de uso principales. Por un lado existe la posibilidad de publicar configuraciones para dictados musicales. Esto se realiza ajustando una gran cantidad de parámetros lo cual permite flexibilidad al momento de indicar qué es lo que se quiere entrenar. En segundo lugar se encuentra el flujo de uso para la generación y reproducción de dictados, con el objetivo de que el usuario entrene su oído musical. A partir de una configuración, la aplicación genera dictados de forma aleatoria (la cantidad que el

[2]aplicación disponible para Android en: https://play.google.com/store/apps/details?id=prueba.apk y disponible en iOS: https://apps.apple.com/us/app/ada-entrenamiento-auditivo/id1592608904?ign-mpt=uo%3D2

[3]Repositorio frontend: https://github.com/francowanseele/Entrenamiento$_A$*uditivo.Repositoriobackend* : *https* : *//github.com/francowanseele/Entrenamiento*$_A$*uditivo*$_s$*erver*.

usuario desee) y de esta forma el usuario pueda escucharlo y transcribir dicho dictado a una hoja pentagramada. Este flujo tiene un segundo paso, que consiste en acceder a la solución (desplegada en la aplicación en un pentagrama), y de esta forma brindarle al usuario la información necesaria para saber la correctitud del ejercicio realizado.

Por otro lado la aplicación cuenta con la capacidad de gestionar instituciones, cursos, estudiantes y docentes. Esto consiste en que hayan docentes responsables de cursos, quienes serán los encargados de subir las configuraciones de los dictados y por el otro lado que existan perfiles con el rol de estudiantes, quienes estén inscriptos a cursos y puedan entrenarse en base a las configuraciones allí subidas.

A modo de poder brindarle al usuario información sobre los progresos, se tiene una sección calificaciones, la cual le brinda a los estudiantes una noción de su desempeño y a los docentes un indicador de qué tan difícil o fácil resultó ser la configuración establecida.

Capítulo 2

Marco referencial

2.1. Enseñanza de la música

El área de la música dio sus comienzos en civilizaciones muy antiguas y hasta la actualidad sigue presente de forma muy activa. A partir del siglo XX hasta la fecha, la música ha ido tomado una gran participación en la educación dentro del área perceptiva, expresiva y comunicativa (Sagredo, 2020).

Hoy en día existe una gran variedad de opciones al momento de entrar en la educación musical. Existen carreras del área las cuales cuentan con actividades que implican que un docente sea el encargado de reproducir dictados musicales y los alumnos los encargados de identificarlo. El presente proyecto se encarga de analizar estas prácticas mencionadas con el objetivo de brindar herramientas que sirvan como complemento a las metodologías utilizadas en la actualidad.

2.2. Distinción de conceptos

En la presente sección se abordan los elementos musicales que son manejados por la aplicación y de esta forma poder ubicar y dar contexto a ciertos temas desarrollados en capítulos posteriores. En primer lugar se presenta un diagrama (Figura 2.1) de los principales conceptos a ser abordados, de forma de poder mostrar la relación que existen entre ellos.

2.2.1. Configuración rítmica

La rítmica hace referencia a las duraciones de los sonidos, haciendo que se diferencien entre largos y breves. Dentro de esta configuración estarán todos los

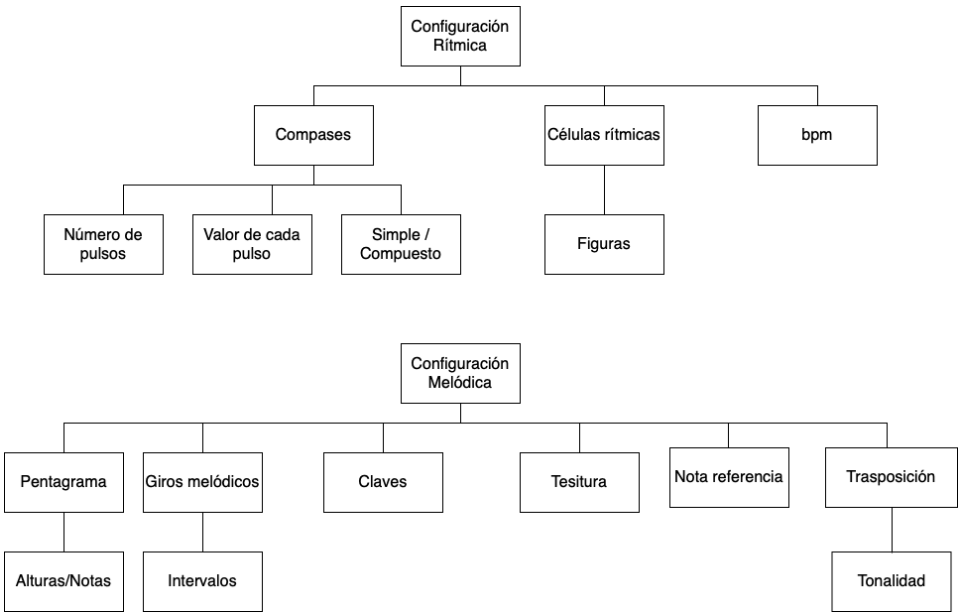

Figura 2.1: Diagrama de conceptos

elementos musicales que hagan referencia a la duración y tiempo de los sonidos, estando también incluida configuraciones de la métrica musical. Esta es la estructura que indica la aparición periódica de sonidos (o silencios) expresada en intervalos.

Compases:

Los compases de un dictado son indicados al comienzo de una partitura y es lo que determina la métrica del dictado. Este es representado por dos números formando una fracción, en el cual, el numero de arriba indica el número de pulsos dentro de un compás y el número inferior indica el valor que debe haber por cada pulso. Es decir, cada figura musical tiene asociado un valor (tema abordado en detalle en definiciones de conceptos posteriores) por lo que una o más figuras deben colocarse dentro de un pulso hasta que sus valores sumados sean igual al valor que debe tener un pulso. Cabe aclarar que una figura puede llegar a cubrir más de un pulso, los cuales pertenezcan a un mismo compás. De esta forma se completan todos los pulsos correspondientes a un compás, volviendo a repetirse el proceso para el número total de compases que tenga el dictado.

Además los compáses pueden ser simples o compuestos. En los compases simples cada uno de sus pulsos se puede subdividir en mitades. Por ejemplo, si los pulsos se subdividen en dos y se agrupan de a dos se tiene un compás de 2/4: el valor de cada pulso (correspondiente a la nota negra) se puede dividir

NOMBRE	FIGURA	FIGURA DE SILENCIO	DURACIÓN EN 4/4	VALOR DE NOTA	DENOMINADOR (en la indicación de compás)
REDONDA (Unidad)			4 Tiempos	4/4	1
BLANCA (Mitad)			2 Tiempos	2/4	2
NEGRA (Cuarto)			1 Tiempo	1/4	4
CORCHEA (Octavo)			1/2 Tiempo	1/8	8
SEMICORCHEA (Dieciseisavo)			1/4 Tiempo	1/16	16
FUSA (Treintaidosavo)			1/8 Tiempo	1/32	32
SEMIFUSA (Sesentaicuatroavo)			1/16 Tiempo	1/64	64

Figura 2.2: Tabla de figuras

en dos (dos corcheas) y además el compás es una agrupación de dos pulsos.

Por otro lado, en los compáses compuestos, los pulsos o tiempos se pueden subdividir en tercios. Por ejemplo, si los pulsos se subdividen en tres y se agrupan de a dos, el compás corresponde a 6/8: esto representaría que el compás está subdividido en 6, cada una con un valor correspondiente a una corchea. Por convección se establece este compás se divide en dos pulsos con tres corcheas cada uno, eso hace que aparezca la figura negra con puntillo, la cual tiene el valor de tres corcheas.

Células rítmicas:

Son conjuntos de figuras musicales las cuales van juntas y en el orden que aparecen, dentro del dictado. Las células rítmicas son las que son utilizadas en el dictado para completar el valor de los pulsos, siendo, el valor de cada célula rítmica correspondiente a la suma del valor de todas sus figuras. A su vez, este valor está asociado a la duración de cada nota, conceptos que están representados en la tabla (Figura 2.2):

La duración está establecida en base al valor de la nota negra correspondiendole el valor de tiempo 1, siendo dos corcheas el correspondiente en cuanto a duración en tiempo (y valor) de la nota negra. Esta misma lógica se extrapola al resto de las figuras.

Por último, las figuras de silencio se comportan de igual forma que las

figuras, a excepción que no tienen ninguna nota musical asociada, ya que en la duración de dicha nota no hay sonido.

bpm:

Los *Beat per Minute* o golpes por minuto sirve para establecer la velocidad de las figuras musicales con exactitud. Si bien se vio que cada nota tiene asociada una duración, los bpm sirven para asociarles una duración de sonido en el tiempo. Es decir, si a la nota negra se le asocia un valor de 128bpm, según la duración de las figuras (vistas en la figura 2.2), la nota blanca tendrá una duración de 256bpm. Cabe aclarar que en este caso la nota negra fue tomada como ejemplo, pero cualquier figura puede ser tomada como referencia para los bpm.

2.2.2. Configuración melódica

Los elementos pertenecientes a esta área corresponden a la representación de la altura de las notas en un pentagrama.

Pentagrama:

El pentagrama está formado por cinco líneas horizontales, representando cada una de estas (incluido los espacios en blanco) una altura diferente. Esta altura representa la frecuencia del sonido, es decir, a mayor frecuencia, mayor la altura y más agudo el sonido. De esta forma cada línea representa una nota musical diferente. Las figuras musicales son escritas sobre el pentagrama y dependiendo sobre que línea o espacio en blanco esté es la nota que representa dicha figura.

Un pentagrama escrito con figuras musicales, en conjunto con otros posibles símbolos musicales, indica cómo debe interpretarse una melodía, y esto recibe el nombre de partitura.

Giros melódicos:

Los giros melódicos son listas ordenadas de notas las cuales indican qué nota puede ir a continuación de cada nota. Es decir, si se tiene la lista de notas musicales A, B, C, D; dado un dictado que en cierto punto contenga a la nota B, seguida de ésta puede colocarse la nota A o C, no así la D. En el caso que la nota B aparezca en más de un giro melódico, para dicho dictado puede aplicarse tanto uno u otro giro melódico.

Estos giros melódicos son vistos como intervalos musicales, lo cual refiere a la distancia que hay entre dos notas. De esta forma cuando a los giros melódicos

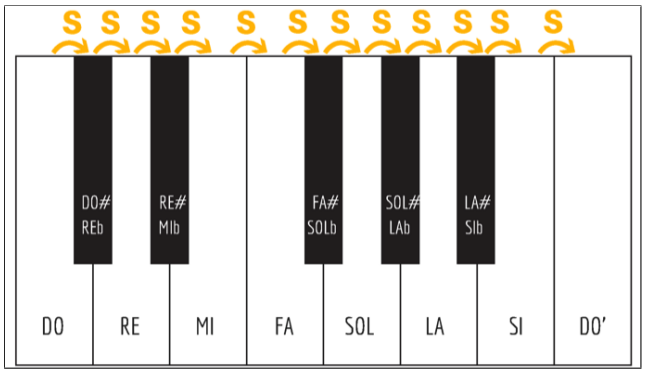

Figura 2.3: Intervalos

se les realiza una trasposición, las notas cambiarán pero no los intervalos entre ellos. Los semitonos es el menor intervalos que hay entre dos notas consecutivas, ejemplificado en la Figura 2.3.

Cada letra "S" representa un semitono y las teclas negras del piano corresponden a alteraciones de las notas como bemol o sostenido (b o $\#$). Esto simplemente quiere decir que al colocarle un sostenido ($\#$) a una nota se le aumenta un semitono. De forma contraria, al colocarle un bemol (b) se le disminuye un semitono a dicha nota.

Claves:

La clave es un signo establecido al comienzo de un pentagrama y cumple la función de indicar la altura de la música escrita. Es decir, asigna una determinada nota (o altura) a una línea del pentagrama, a partir de la cual sirve como referencia para establecer el resto de notas (o alturas). A este elemento se le pueden agregar armaduras en clave, la cual consiste en agregarle alteraciones al comienzo de la partitura. Por ejemplo si se le agrega un sostenido ($\#$) a la línea del pentagrama que representa la nota Sol, todas las notas que aparezcan sobre las líneas que representan la nota Sol serán $Sol\#$, a menos que se indique lo contrario. Es decir, las alteraciones indican que por defecto las notas alteradas estarán un semitono arriba o abajo, dependiendo si corresponde a un sostenido o un bemol respectivamente.

Cabe aclarar que a las armaduras en clave les corresponde una tonalidad, por ejemplo, la tonalidad Sol mayor le corresponde una armadura en clave la cual altera la nota de Fa agregándole un sostenido.

Tesitura:

La tesitura indica entre qué dos notas estará contenido un dictado. En la Figura 2.4 se ve un ejemplo de notas en un pentagrama con clave de sol y fa

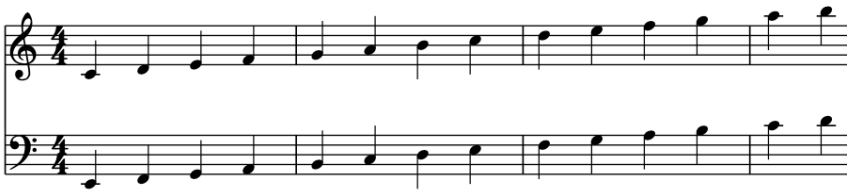

Figura 2.4: Pentagrama con claves de sol y fa. *Imagen obtenida de la herramienta: https://flat.io/es*

(la de arriba y abajo respectivamente).

En esta imagen, las primeras ocho notas representadas en la clave de sol corresponden a Do4, Re4, Mi4, Fa4, Sol4, La4, Si4, Do5; y para la clave fa son Mi2, Fa2, Sol2, La2, Si2, Do3, Re3, Mi3. Con esta lógica es que aumentan y disminuyen las notas. A partir de éste ejemplo, una tesitura válida para la clave de sol podría ser el rango Do4, La5; lo que indica que todas las notas comprendidas entre éstas son válidas para un dictado. De la misma forma podría establecerse una tesitura para la clave de fa.

Nota de referencia:

Esta nota sirve para quienes vayan a escuchar una melodía para poder situarse en un contexto con el fin de poder calcular las relaciones entre las notas que suenan. Esta suele ser escuchada previo a dicha melodía sabiendo a que nota corresponde el sonido.

Trasposición:

Dado un conjunto de notas escritas en una cierta tonalidad, por ejemplo las notas de un dictado, una trasposición consiste en escribir dichas notas en una tonalidad diferente. Esto se hace, en primer lugar, moviendo todas las notas un cierto intervalo, y en segundo lugar, aplicando las alteraciones que corresponden a la nueva tonalidad. Cabe destacar que esto mantiene las relaciones entre las notas pero desde un punto de partida diferente.

2.2.3. Dictados

Existen tres tipos de dictados que se abordarán a lo largo del presente trabajo: dictados rítmicos, melódicos y armónicos.

Dictado rítmico:

Consiste en una secuencia de figuras escritas en un pentagrama, en el cual siempre suena una única nota musical. El objetivo con este tipo de dictado es reconocer las figuras musicales de acuerdo a su duración. Este tipo de dictado

generalmente es abordado previo a los otros dos tipos de dictados, ya que sirve de base para abordar lo siguiente.

Dictado melódico:

En este tipo de dictado se le suman las notas musicales. Esto se refiere a que a cada figura musical se le asigna una nota, teniendo como objetivo el reconocimiento, tanto de figuras como de notas musicales.

Dictado armónico:

En los dictados armónicos aparecen los acordes. Esto consiste en tocar varias notas musicales de forma simultánea, teniendo el objetivo de reconocer, además de las figuras musicales, las estructuras resultantes de cada conjunto de notas.

2.3. Trabajos relacionados

El estudio de trabajos relacionados se realizó con el principal objetivo de obtener información sobre cómo son resueltos problemas similares a los planteados para el presente trabajo. De este relevamiento de información, junto con la colaboración de docentes y estudiantes de música, se obtuvo formas de representar elementos musicales que resultan intuitivos para personas del ámbito, los cuales fueron adaptados para poder incorporarlos a la aplicación desarrollada. Cabe destacar que muchas aplicaciones presentadas en esta sección resuelven problemas diferentes a los planteados en el presente trabajo, incluso con un público objetivo diferente, por lo que muchas funcionalidades, si bien fue de utilidad su estudio, no se alinean con los objetivos planteados.

2.3.1. Live music programming in Haskell

El objetivo de *Live music programming in Haskell* es componer musica mediante algoritmos, sin que se tenga la necesidad de generar la melodía nota por nota, sino que se pueda describir la estructura musical, especificando, por ejemplo, un patrón dado por una secuencia armónica. El usuario puede establecer todos los parámetros necesarios mediante un sub-lenguaje del lenguaje de programación funcional *Haskell*, haciendo que esté principalmente dirigido a usuarios del área de la programación. Mediante este lenguaje, el usuario está habilitado a especificar una estructura, para la melodía generada, de forma estricta o más laxa. Esto quiere decir que se puede tanto especificar las notas

de forma manual como siguiendo un patrón de notas y que simplemente se complete con un ritmo totalmente aleatorio.

Uno de los modos de operar para lograr componer musica es generarla como una lista de eventos MIDI, es decir, eventos como "tecla presionada", "tecla liberada", "instrumento cambiado", "mando girado" e instrucciones de espera. A cada uno de estos eventos, en el caso que corresponda, se le asigna una lista de notas musicales o bien un patrón de notas. Todos estos datos son ingresados a través de la interfaz gráfica en código *Haskell*. Una vez escrito esto, se transfiere al *bufer* del interpretador y el programa es ejecutado, mostrando en pantalla las llamadas a las funciones las cuales se expanden en eventos MIDI, permitiendo al usuario rastrear la melodía visualmente (Thielemann, 2013).

En este proyecto resulta muy interesante el manejo de la aleatoriedad y que en base a un patrón se generen melodías. Esta característica hace que la música generada sea variada, haciendo que no sea predecible los resultados que brinda. El inconveniente con esta forma de operar es que no se pueden priorizar elementos musicales, lo cual es algo deseable en la enseñanza de dictados, para que el docente pueda introducir de forma paulatina ciertos elementos musicales a lo largo del curso.

Por otro lado, este proyecto está orientado a componer música, ofreciendo una gran libertad al momento de brindarle los datos de entrada necesarios para que sean procesados. Esto, si bien permite una amplia gama de posibilidades, tiene la contra de que si no se compone musica con una buena estructura, los intervalos de tiempos puede que no sean exactos (Thielemann, 2013), lo cual es un punto que se desea tener controlado.

Estos puntos, sumado a que el usuario final debe contar con conocimientos en el lenguaje de programación *Haskell*, hace que sea una herramienta dirigida a un público muy acotado, a pesar de su posible potencial. Esto hace que no sea una herramienta adecuada para instituciones educativas del área de la música ya que estudiantes y docentes de dicha carrera deberían ser capaces de utilizarla.

2.3.2. The rhythmic dictator

The rhythmic dictator consiste en el desarrollo de la aplicación Trubadur[4] proveniente de Eslovenia la cual es una aplicación web que hace foco en que sea adaptada a dispositivos móviles, haciendo especial énfasis en incluir un diseño intuitivo de los elementos musicales que se manejan. Está orientada al entrenamiento auditivo en los estudiantes de música mediante ejercicios rítmicos, melódicos y armónicos. Los ejercicios son abordados a través de juegos y desafíos en los cuales se comienza en un cierto nivel y se avanza a medida se completen de forma correcta.

Esta aplicación brinda un camino de aprendizaje el cual consiste en que los estudiantes resuelvan dictados rítmicos, melódicos y armónicos, y que se muestre de forma inmediata un feedback de su desempeño en el ejercicio. Para lograrlo, la aplicación hace énfasis en aspectos de la interfaz de usuario, ya que, luego que el estudiante escucha el dictado, se presentan elementos gráficos para poder transcribirlo en notación musical.

Un aspecto importante que se maneja es la generación aleatoria de ejercicios. Las melodías generadas son generadas en base a conceptos que los docentes de música suelen utilizar para el dictado de cursos, y a medida que el usuario avanza de nivel, se incorporan nuevos conceptos ya establecidos en la plataforma (Pesek et al. 2020). En este punto cabe aclarar que si bien la generación aleatoria de dictados y la enseñanza de la música a través de estos ejercicios es un concepto que también se maneja en la aplicación ADA, en la aplicación *The rhythmic dictator* los conceptos musicales a enseñarse ya están establecidos por la aplicación. Es decir, para cada uno de los niveles por los que pasa el estudiante los conceptos musicales que se abordan ya están preestablecidos, lo cual hace que el estudiante no pueda elegir entrenarse en cierto tema concreto. Además, esto deja de lado el rol del docente dentro dado que no se puede establecer una trayectoria de enseñanza.

A pesar de que esta aplicación no se alinea con los objetivos que se plantearon en secciones anteriores para el presente proyecto, *The rhythmic dictator* resuelve de forma acertada cuestiones sobre cómo manejar ciertos conceptos musicales desde la interfaz de un dispositivo móvil. Existen funcionalidades que fueron resueltas de tal forma que resulta interesante tenerlas en cuenta para el desarrollo del proyecto, como ser:

[4]Acceso a la aplicación en: https://trubadur.si

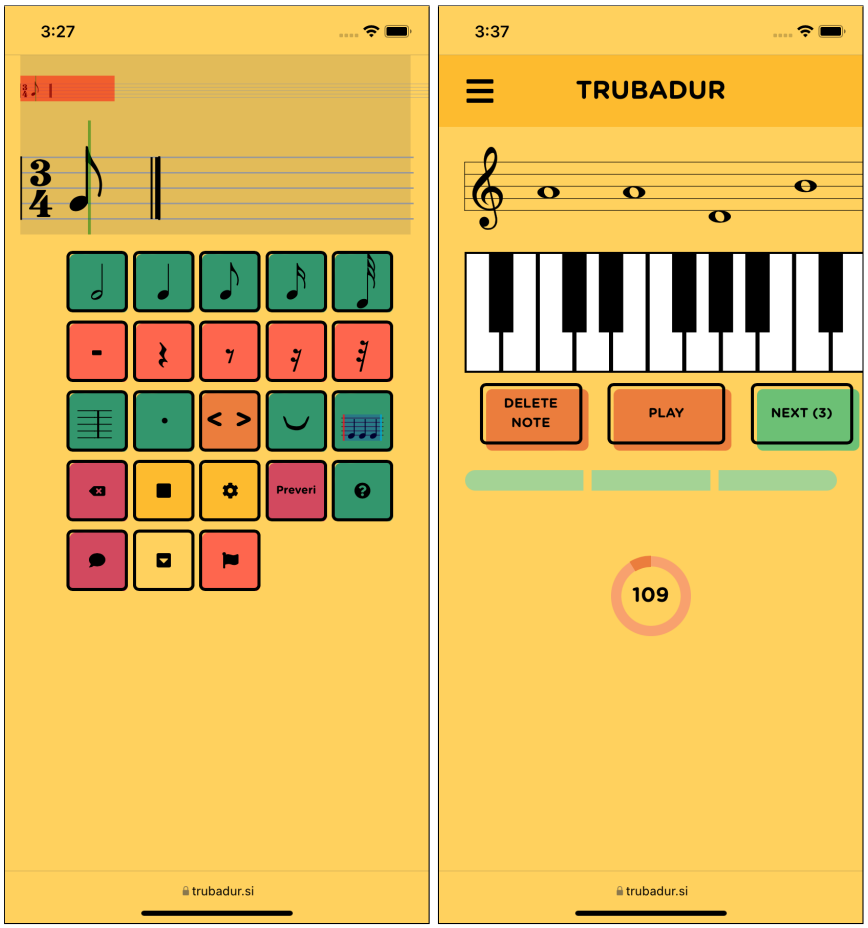

Figura 2.5: Captura de pantallas aplicación Trubadur

- **Escribir notas musicales**
  Para escribirlas se presenta en pantalla un teclado en forma de piano el cual resulta intuitiva, ya que es una aplicación dirigida a usuarios pertenecientes al área musical .
- **Escribir figuras musicales**
  Las mismas son presentadas en modo de teclado en pantalla, las cuales son agrupadas en base a diferentes conceptos, como son las figuras con sonido y silencios.
- **Representación**
  Cada nota y figura musical que se escribe es mostrado en un pentagrama, con la opción de borrado, similar al uso de cualquier teclado.

Estos elementos gráficos (Figura 2.5) presentados en esta aplicación (como puede ser un teclado en forma de piano o la representación de lo escrito en un pentagrama) resulta adecuado e intuitivo. Debido a que el público objetivo de

la aplicación ADA son estudiantes y docentes de música, algunos elementos de los mencionados fueron tomados como base y adaptados para funcionalidades similares, los cuales son abordados en mayor detalle en el Capítulo 4 de diseño.

2.3.3. Aplicación *Teoría: Music Theory Web*

El portal web de *Teoría: Music Theory Web*[5] está dedicado al estudio y práctica de la teoría musical. Esta plataforma fue desarrollada por José Rodríguez Alvira como un recurso para complementar sus clases, la cual fue premiada en 2006 con el *Merlot Classic Award* en el área de música, premio que se otorga para reconocer recursos en línea destacados, los cuales fueron desarrollados para mejorar la enseñanza y el aprendizaje.

Dentro de la plataforma se tiene una sección bajo el título de ejercicios de entrenamiento auditivo (*Ear traning exercises*) en la cual se brinda una muy completa variedad de elementos musicales a entrenar. Si bien esto es un punto fuerte de esa plataforma, no se tiene la posibilidad de personalizar al detalle algunos características del dictado, como puede ser los bpm de las figuras. Además esta configuración debe ser realizada por el usuario que desea entrenarse, por lo que no existe un rol de docente que sea quien guíe el aprendizaje.

Como punto a destacar, que se repiten en la revisión de otros trabajos del área, se puede ver que la escritura de las notas musicales se realiza mediante el uso de un teclado de piano en pantalla. En cuanto a lo rítmico, la escritura de las figuras musicales se presentan a modo de lista en pantalla. Ambas representaciones de los elementos musicales son tenidas en cuenta, junto a otros trabajos del área, al momento del desarrollo de la aplicación ADA.

Finalmente, en cuanto a los dictados que genera esta plataforma se observa que no resultan repetitivos dado a la gran variedad de elementos melódicos que se pueden incluir o dejar de lado. El hecho de que el usuario se entrene con dictados de a cuerdo a sus necesidades depende de la configuración previa que este realice. Al igual que en otros trabajos revisados anteriormente no se tiene en cuenta de forma explicita en la aplicación el rol del docente.

[5]Acceso al portal: https://www.teoria.com

2.3.4. Quizlet

La plataforma *Quizlet*[6] no es exactamente un software orientado a la música, sino orientado a la enseñanza y aprendizaje. Es una herramienta que permite estudiar ciertos temas mediante ejercicios, tanto de forma individual o grupal, con la participación de un docente. La forma de operar consiste en crear tarjetas de estudio con contenido educativo, por ejemplo, colocando al frente de esta un concepto y en su reverso la definición correspondiente. Una vez que esto se sube a la plataforma, la herramienta ofrece una serie de dinámicas para interactuar con estas tarjetas, por ejemplo, revelando de forma aleatoria el contenido de una de sus caras, teniendo el usuario que ingresar en el sistema el contenido del reverso de la tarjeta.

Lo interesante de la plataforma resulta en la forma que estructura los cursos, ya que están organizados en carpetas, y cada una de éstas contienen diferentes unidades de estudio. A su vez se tiene la opción de hacer públicas o privadas las unidades de estudio que se den de alta, conceptos los cuales son similares a los utilizados en la aplicación ADA.

Por otro lado, cuenta con una sección de avances, en la cual se puede ver el progreso que un usuario ha tenido, separado por cursos y unidades de estudio. Esta forma de manejar los datos y brindarlos al usuario resulta de valor, sobre todo a usuarios con roles educativos, ya sea estudiante como docente.

2.4. Otros trabajos

Adicionalmente se estudiaron otras aplicaciones que, si bien no tenían objetivos similares a los planteados en el proyecto, fueron de ayuda para saber como manejaban ciertos elementos musicales.

La aplicación que se tuvo mayormente en consideración fue *iWriteMusic*[7]. Este software para dispositivos móviles consiste principalmente en una herramienta para escribir en una hoja pentagramada. Si bien no es una herramienta de generación de dictados musicales orientados al aprendizaje, está enfocada a la escritura en un pentagrama, manejando de forma acertada e intuitiva los elementos como notas y figuras musicales.

Si bien en otros trabajos relacionados, al igual que en este, se repiten pa-

[6] Acceso a la aplicación en https://quizlet.com/.
[7] Acceso a la aplicación: http://iwritemusicapp.com

trones de cómo abordan, por ejemplo, la escritura de las notas musicales (las cuales se escriben mediante el uso de un teclado del piano), este software presenta una característica destacable y es que al asignarle una figura a una nota musical, no solo es mostrada en el pentagrama, sino que también se reproduce su sonido. Esto puede resultar de gran ayuda para estudiantes que recién comienzan, ya que al escribir la solución de un dictado, el poder escuchar las notas escritas le brinda cierta ayuda.

Capítulo 3

Relevamiento de requerimientos

Los requerimientos fueron relevados mediante el uso de diferentes técnicas, principalmente interactuando con los posibles *stakeholders* que estarán en contacto con la aplicación. Estos encuentros se abordaron en modo entrevistas abiertas, en donde no se tenía una agenda definida, sino que se buscaba comprender las necesidades generales. Avanzada esta etapa, los encuentros fueron abordados a modo de *workshops*, lo cual consiste en un trabajo en conjunto para definir y refinar requisitos.

Una vez se tuvo mayor noción del área, lo cual fue acompañado con el desarrollo de pruebas de conceptos, se procedió a construir un prototipado de la aplicación. En primer lugar, esto se abordó presentando bocetos en los que se representen las pantallas principales, lo cual permitió tener algo tangible sobre lo cual definir requerimientos funcionales. Esto fue de gran valor ya que permitió enfocar todas las energías en abordar temas del flujo de uso y distribución de conceptos, dejando de lado aspectos visuales.

Esta etapa dio lugar al desarrollo de un prototipo evolutivo, el cual se contempló que provea una base arquitectónica sólida para poder desarrollar el producto de forma incremental, al mismo tiempo que los requerimientos se vuelven más claros y detallados. Esto permitió también acotar posibles riesgos de insatisfacción en el producto, además de abordar aspectos como la percepción de la interfaz. Este prototipo, una vez fue refinado, permitió poder relevar requerimientos mediante la técnica de *focus groups*. Esta consistió en presentar a un grupo de estudiantes avanzados de la carrera de música dicho desarrollo y así poder relevar contribuciones e ideas, desde el lado de la música. Esto fue de gran valor para poder entender en mayor profundidad las necesidades

de los usuarios, las impresiones que le genera el software y poder evaluar qué tan intuitivo resulta la herramienta desarrollada. Cabe destacar que la información recaudada es subjetiva y sirve principalmente a modo de evaluación y refinamiento de la aplicación.

Por otro lado, se investigaron antecedentes, una técnica que, si bien no reveló gran cantidad de requerimientos, fue de utilidad al principio del proyecto para poner en contexto y brindar una perspectiva (aunque algo limitada, ya que es una desventaja de esta técnica) del área musical. Esto se abordó desde dos puntos, en primer lugar una investigación de antecedentes interna, estudiando cómo es que funcionan algunas instituciones educativas al momento de enseñar ciertos temas, indagando tanto desde el lado de la enseñanza como desde el lado de la música. Por otro lado se estudió el aspecto externo, investigando aplicaciones ya existentes, y la forma en que éstas son de utilidad o en qué aspectos poseen limitaciones.

Los requerimientos obtenidos en el transcurso de estas etapas fueron registrados mediante historias de usuario. Estas consisten en pequeñas descripciones de alto nivel, escritas en términos del cliente, en este caso utilizando términos del área musical. La principal idea es evitar perder tiempo en describir detalles y escribirlas de forma general (Sommerville, 2011).

En las siguientes secciones se especifican los requerimientos obtenidos en forma de historias de usuario obtenidas principalmente desde tres lados diferentes: desde el área de la música, desde el equipo de desarrollo y desde la educación. Además se tendrá un breve resumen del análisis que se hizo a cada historia de usuario y cómo estas necesidades son abordadas y solucionadas en la aplicación.

3.1. Requerimientos no funcionales

[Req. 001] Como usuario de la aplicación (tanto estudiante como docente) quiero que a la aplicación se le realice mantenimiento y se actualice en función de las necesidades que vayan surgiendo para mantener el software alineado con las metodologías de enseñanzas y aprendizajes actuales.

Para cubrir este requerimiento es que se decide que el desarrollo sea de código abierto y de esta forma no limitar el desarrollo futuro de la aplicación. Esto además podría cubrir la implementación de ciertas funcionalidades que

quedan planteados como trabajo a futuro

[Req. 002] Como usuario de la aplicación quiero que la misma sea escalable para así poder soportar un número considerablemente alto de cursos con sus respectivos estudiantes y docentes.

Siguiendo por la misma línea que el requerimiento anterior, y mirando hacia el futuro de la aplicación, se quiere que el desarrollo pueda soportar, tanto un crecimiento en usuarios activos, como un crecimiento en nuevas funcionalidades. Esta última idea apunta concretamente a que el sistema sea lo suficientemente flexible como para incorporar funcionalidades no previstas inicialmente. Este requerimiento es cubierto con la arquitectura en que se desarrolla el proyecto, además de la elección de una base de datos no relacional. Dichos puntos son abordados en el Capítulo 5 de desarrollo.

[Req. 003] Como equipo se quiere que las funcionalidades sean testeadas y validadas antes de su lanzamiento para poder poder minimizar errores de concepto pertenecientes al área de la música.

Este requerimiento se aborda con un desarrollo modular. De esta forma, al completarse ciertos conjuntos de funcionalidades se pueden ir validando por personas conocedoras de la música. Esto consiste tanto en mostrar la aplicación en uso como en publicar versiones previas para que efectivamente puedan ser usadas por posibles futuros usuarios.

[Req. 004] Como equipo se quiere que el desarrollo del proyecto sea capaz de adaptarse a cambios y modificaciones que surjan en el transcurso del proyecto para que la dinámica de desarrollo se adapte a los procesos de investigación y no al revés.

Las funcionalidades pueden sufrir modificaciones en el transcurso del proyecto e incluso surgirán nuevas. A esto se le suma que, la música y la ingeniería, al ser dos áreas muy diferentes, las funcionalidades centrales de la aplicación deberán ser validadas con personas conocedoras del tema, teniendo una gran probabilidad de tener que re-analizar y re-ver funcionalidades ya desarrolladas. Dada esta situación es que se plantea utilizar metodologías ágiles de desarrollo, permitiendo un desarrollo flexible, lo cual hará que el sistema responda rápidamente a cambios que surjan.

[Req. 005] Como usuario quiero que mis datos en el sistema no sean accedidos por cualquier persona para que mi progreso no se vea alterado.

Este requerimiento esta muy relacionado con la seguridad del sistema y para esto se decide implementar un sistema de *tokens*, lo cual permite que solamente un usuario registrado en el sistema esté habilitado a acceder a ciertos datos y funciones que el sistema provee.

[Req. 006] Como usuario perteneciente a un instituto educativo quiero poder utilizar las plataformas ya existentes para poder conectarme a la aplicación y no tener que manejar dos plataformas separadas.

La necesidad de que el desarrollo sea interoperable, es decir, que la aplicación interactúe con otros sistemas, surge a partir de las plataformas que ya se encuentran en uso en instituciones educativas y la fomentación de su uso en cursos educativos.

[Req. 007] Como usuario quiero que la aplicación sea intuitiva y sencilla para que el diseño no sea un obstáculo al momento de su uso.

Dada la complejidad que tiene la aplicación en ciertas funcionalidades, este requerimiento se vuelve esencial al momento de evitar que los usuarios dejen de utilizar la aplicación por ser tediosa o frustrante. Esto se aborda principalmente realizando una fuerte investigación sobre temas de diseño y flujo de uso, haciendo énfasis tanto en la interfaz de usuario como en la experiencia de usuario.

3.2. Requerimientos funcionales

A continuación se abordan los requerimientos funcionales relevados, agrupándolos en tres categorías: ingreso de datos, salida de datos y administración de datos. A su vez, dentro de cada una de éstas, existen dos agrupaciones. Por un lado se tendrá todo lo relacionado a la generación de dictados, lo cual implica aspectos y términos mucho más técnicos desde el punto de vista musical. Por otro lado se presentará los requerimientos relacionados a la gestión, es

decir, todo lo relacionado al manejo de cursos pertenecientes a instituciones, con sus respectivos docentes y estudiantes.

3.2.1. Ingreso de datos

3.2.1.1. Generación de dictados

[Req. 008] Como docente quiero poder especificar qué elementos entrenar con los dictados generados para poder organizar un curso en diferentes temas.

Esta necesidad es abordada permitiéndole al docente una gran libertad en la configuración de parámetros para la generación de un dictado. En primer lugar, al establecer una configuración a partir de la cual se van a generar dictados aleatorios, se tiene la posibilidad de generar dictados rítmicos o dictados melódicos, lo cual apunta a entrenar dos aspectos musicales diferentes.

Por otro lado, se tiene la posibilidad de establecer diferentes parámetros tanto para la configuración rítmica como para la melódica. Algunos de estos parámetros son los giros melódicos, la clave de sol y fa, las diferentes tonalidades, los compases y las células rítmicas. A cada uno de estos se tiene la posibilidad de asignarle una prioridad diferente, esto quiere decir que el docente es capaz de decirle al sistema cuáles elementos quiere que aparezcan con mayor frecuencia (o incluso que no aparezcan), haciendo énfasis en los elementos que quiera entrenar.

Dentro de los elementos a entrenar también surge la necesidad de querer entrenar con dictados con diferentes velocidades, para lo cual el docente establece un rango de bpm (beats per minute) lo cual hará que la velocidad de un dictado musical varíe desde ritmos más lentos a más rápidos.

[Req. 009] Como estudiante quiero poder generar nuevas configuraciones para poder entrenar en aspectos musicales que quiero enfatizar.

Este requerimiento surge de la necesidad de que los usuarios pueden querer entrenarse siendo autodidactas o incluso quienes quieren ingresar a carreras relacionadas con la música y desean tener cierto entrenamiento previo. Para esto se decidió que la misma funcionalidad de crear una configuración de dictados que está disponible para el docente lo esté para el estudiante, con la limitante de que sólo el propietario podrá generar dictados aleatorios a partir de dicha

configuración.

[Req. 010] Como usuario quiero poder especificar los giros melódicos como intervalos al generar una configuración de dictado para que resulte más intuitivo para los músicos.

En primer lugar, el especificar los giros melódicos va a ser sumamente necesario ya que las notas de los dictados deben seguir ciertas reglas al momento de generarse. Las notas que especifiquen los giros melódicos va a depender de la tonalidad en que se genere el dictado por lo que el especificarlas como intervalos y no como notas específicas no genera confusión. Esta necesidad es cubierta habiendo desarrollado un teclado (para el momento de escribir los giros melódicos) en el cual se indican diferentes intervalos. Las notas que genere dicho teclado dependerá de si se le especifica la armadura en clave mayor o menor, ya que dependiendo de esto último va a depender de qué nota se calculan dichos intervalos.

[Req. 011] Como usuario quiero tener la opción de poder especificar los giros melódicos seleccionándolos a partir de una lista para que no resulte tedioso tener que escribirlos cada vez.

Dado que muchos giros melódicos que se vayan a dar de alta en las configuraciones de dictados van a repetirse, se tiene una lista de giros melódicos dada de alta en el sistema la cual contempla una gran cantidad de posibles casos. Esto no descarta que exista la posibilidad de ingresar giros melódicos personalizados.

3.2.1.2. Gestión

[Req. 012] Como docente perteneciente a un instituto educativo quiero poder dar de alta nuevos cursos dentro del instituto para poder tener una mayor libertad al momento de gestionar el dictado de cursos.

Los docentes serán capases de crear nuevos cursos y los mismos deberán de pertenecer a un instituto educativo para el cual dicho docente deberá estar formando parte. Esto será de gran importancia para poder agilizar el uso de la aplicación dado que si la creación de cursos depende de un usuario administrador, esto dificultará y hará más lento el proceso.

[Req. 013] Como docente particular quiero poder dar de alta nuevos cursos y que los mismos queden asociados al propietario para que estudiantes puedan inscribirse.

La aplicación contemplará también a docentes quienes no pertenezcan a ningún instituto educativo. Estos serán interpretados como docentes particulares y los mismos podrán tener diferentes cursos asociados a su usuario, con sus respectivos estudiantes inscriptos.

[Req. 014] Como docente quiero poder organizar el dictado del curso para tener una planificación adecuada.

Esta necesidad es resuelta con la estructura que maneja la aplicación para la gestión de los cursos. Estos contienen diferentes módulos y dentro de cada uno de estos existirán diferentes configuraciones de dictados que serán dada de alta por el docente responsable del curso.

[Req. 015] Como docente que crea un curso quiero tener la opción de asociar una clave para ser requerida al momento de que un estudiante se inscriba y así poder tener un cierto control sobre los usuarios que realicen los ejercicios.

Si bien esto no garantiza que se inscriban solamente estudiantes, ya que la clave puede ser compartida y quien tenga dicha clave puede inscribirse al curso, si es un filtro de importancia para los docentes, sobre todo al momento de acceder a las calificaciones y ver el desempeño de sus estudiantes en los dictados del curso.

3.2.2. Salida de datos

3.2.2.1. Generación de dictados

[Req. 016] Como usuario quiero que la aplicación genere dictados rítmicos, melódicos y armónicos de forma aleatoria para poder entrenar el oído.

El requerimiento que los dictados generados sean aleatorios es uno de los principales puntos que se atacó durante el desarrollo del proyecto. Es crucial que éstos no sean repetitivos ya que estudiantes de música han manifestado haber utilizado software similares, los cuales tienen dictados para practicar,

pero al cabo de un tiempo se vuelve repetitivo, lo que conlleva a dejar de utilizarlo.

La necesidad de la aleatoriedad en los dictados esta cubierta por la forma en que se especifican los datos y el procesamiento de los mismos. La configuración de dictados consiste en una serie de reglas, muchas de ellas con una probabilidad asociada, lo cual, al momento de la generación del dictado, se construyen diferentes caminos, llegando a obtener un amplio abanico de dictados generados (ver Capítulo 5 sección Implementación).

[Req. 017] Como usuario quiero que los dictados generados comiencen y terminen en determinadas notas para que los mismos sean coherentes.

Si bien esta necesidad es cubierta en la aplicación como una regla más dentro de la configuración del dictado, la misma se detalla como un requerimiento separado debido a su complejo estudio y desarrollo.

Al construir un dictado se parte de una nota base y cada una de los giros melódicos que se va eligiendo va construyendo un camino diferente, lo que podría asociarse a una estructura arborescente, en la cual, estando en un cierto nodo, se va a ir hacia uno de sus hijos dependiendo del giro melódico que se elija. El tener la condición de que el dictado deba terminar en una cierta nota hace que este árbol deba ser podado. Además se tiene en cuenta el número de notas totales, dada por las células rítmicas generadas (ver Capítulo 5).

[Req. 018] Como usuario quiero que la aplicación genere dictados simples y compuestos para tener un mayor espectro de ejercicios cubiertos.

Lo que determina que un dictado sea simple o compuesto es el compás de este, limitando así el tipo de figuras que se utilizarán dentro de las células rítmicas. Es por esto que, al dar de alta una configuración de dictado se debe seleccionar si se desea un dictado simple o compuesto, y en base a esta elección es que se mostrará un determinado conjunto de compáses, además de condicionar también las células rítmicas disponibles para seleccionar.

[Req. 019] Como usuario quiero que la aplicación genere dictados en clave de sol y fa, con una tesitura establecida para tener

una buena calidad en la generación de ejercicios.

Los dictados se generan en clave de sol y fa de forma aleatoria, teniendo, el docente, la capacidad de asignarle a cada clave una prioridad para que haya mayor predominancia de una que de otra, según se requiera. A su vez, al generarse el dictado, el mismo es ajustado dentro de una tesitura establecida para cada clave y de esta forma contemplar el requerimiento.

[Req. 020] Como usuario quiero que la aplicación genere dictados "humanizados" para que los mismos sean más realistas.

Esta necesidad surge a partir de que muchos softwares existentes incorporan un parámetro denominado "humanizer" el cual agrega ciertas imperfecciones al momento de reproducir música generada por un software. En la aplicación desarrollada, este requerimiento fue abordado desde el lado de los tiempos rítmicos. Esto consiste en agregar de forma aleatoria una pequeña alteración en el momento en que comienza un pulso. Esto hace que los pulsos comiencen unas milésimas de segundo antes o después, sin que llegue a haber cambios drásticos.

[Req. 021] Como estudiante quiero tener una nota de referencia para escuchar previo al dictado a modo de guía.

Este requerimiento consiste en que se brinde la opción de reproducir una nota, la cual se le va a indicar cual es. De esta forma el estudiante tendrá una referencia para poder identificar el resto de las notas que sonarán en el dictado.

[Req. 022] Como estudiante quiero que, previo a la reproducción del dictado, se marque los pulsos de cada compás para tener una referencia de los tiempos.

Este punto fue abordado introduciendo, previo al comienzo de cada dictado, el sonido de "sticks". Estos sirven para indicar, en primer lugar el tiempo de cada pulso y en segundo lugar cuantos pulsos por compás habrá en el dictado generado.

[Req. 023] Como estudiante quiero poder acceder a la solución de cada dictado para poder evaluar mi desempeño.

Este requerimiento es contemplado por dos funcionalidades en la aplicación.

Por un lado, luego de la reproducción del dictado se tiene la opción de poder ver la solución, lo cual consiste en representar de forma gráfica el dictado en un pentagrama y así el estudiante corroborará la correctitud de su solución. Cabe aclarar que la solución brindada por el estudiante está pensada para que sea escrita en una hoja pentagramada, ya que si la misma fuera escrita en la aplicación directamente haría mucho más lento su escritura, pudiendo no seguir la velocidad del dictado.

Por otro lado, como segundo paso, el estudiante tiene la opción de indicar cuantos errores tuvo. De esta forma el estudiante podrá realizar un seguimiento a la evolución de su desempeño.

3.2.2.2. Gestión

[Req. 024] Como usuario que quiere crear una nueva configuración de dictado, quiero poder tener a disposición una guía para facilitar la comprensión de la herramienta.

La aplicación está pensada para que, al momento de estar estableciendo una configuración de dictado, se tenga una ayuda en cada sección. Esta ayuda aparecerá cada vez que el usuario lo requiera, describiendo en qué consiste y cómo se configura el elemento consultado. Además de facilitar la comprensión del software, ayudará en el uso de la aplicación, evitando que ésta pueda llegar a ser frustrante debido a su grado de complejidad.

A esto se le agrega una validación de forma automática por parte de la aplicación, la cual hace foco en dos aspectos. En primer lugar, a medida que el usuario va estableciendo los parámetros de la configuración, la aplicación indicará si existen incongruencias entre ellos, indicándole cual es el error. En segundo lugar, al querer querer finalizar la configuración, la aplicación verificará si efectivamente es posible generar dictados a partir de ésta. Esto contempla ciertos casos borde en los cuales, si bien no existen incongruencias, no es posible llegar a generar un dictado con las características dadas.

[Req. 025] Como usuario quiero poder acceder a configuraciones de dictados ya existentes para que la plataforma sea colaborativa.

Dado que, tanto docentes como estudiantes van a poder crear configuraciones de dictados, cualquier usuario va a poder acceder a las que se hayan dado de alta en el sistema. Del lado del estudiante, esto le permitirá generar una

variedad de dictados muy amplia en cuanto a elementos a entrenar. Desde el lado del docente permitirá tener acceso a configuraciones de otros cursos que servirán como guía para configurar su propio dictado del curso.

[Req. 026] Como usuario quiero que las configuraciones de dictados que se encuentran en el sistema contengan ciertas etiquetas para que funcionen a modo de filtro y facilitar así la búsqueda.

La información que maneja el sistema está pensado para que crezca a medida que crecen los usuarios activos, lo cual implica que buscar elementos de una lista pueda resultar tedioso. En este caso, al darse de alta configuraciones de dictados se les asignarán etiquetas (por el usuario propietario) las cuales indicarán temática, dificultad o elementos a entrenar de los dictados que se generen. Esto facilitará la búsqueda tanto de los docentes para sus armados de cursos como para estudiantes y su entrenamiento a partir de dichas configuraciones.

[Req. 027] Como estudiante y docente quiero poder ver mi progreso y el de mis alumnos respectivamente para evaluar el desempeño.

Para contemplar esta necesidad se tiene un menú dedicado a la parte de calificaciones, en donde, si se ingresa como estudiante, se podrá ver el desempeño que tuvo en los dictados realizados así como también si fueron realizados de forma reiterada. Desde el lado del docente, esta pantalla contendrá el desempeño de los estudiantes en las configuraciones de dictados de cada curso. Estos datos son presentados sin indicar la identidad de los estudiantes, ya que el objetivo en este punto no es evaluar el desempeño de cada uno individualmente, sino de tener un indicativo del nivel general de los alumnos, además de funcionar como feedback si los dictados generados resultaron muy complejos o fáciles.

3.2.3. Administración de datos

3.2.3.1. Generación de dictados

3.2.3.2. Gestión

[Req. 028] Como usuario de la aplicación quiero poder loguearme como estudiante y docente con un email dado para no restringir el uso de la plataforma.

Este requerimiento se basa en que un docente podría querer loguearse como estudiante para practicar por su cuenta con dictados rítmicos y melódicos. Además podría darse el caso que alguien registrado como estudiante quiera generar cursos para que otros estudiantes se inscriban, ya sea con el objetivo de ser docente o simplemente para querer compartir configuraciones con el resto.

Dado esta necesidad es que se incorpora un botón estilo switch, previo a iniciar sesión el cual indicará si se quiere loguear como estudiante o docente.

[Req. 029] Como docente quiero que mis estudiantes se inscriban a mis cursos para que accedan a su contenido.

Para abordar esta necesidad, cada estudiante tendrá un conjunto de cursos a los cuales esté inscripto y así poder estar actualizado con el contenido que sube cada profesor a dichos cursos. Cabe destacar que en la entidad docente no solamente están aquellos quienes pertenecen a un instituto educativo sino también los profesores particulares de música.

[Req. 030] Como usuario de la aplicación no quiero que cualquier otro usuario dicte cursos pertenecientes a instituciones educativas para estar seguro que el contenido de dichos cursos haya sido creado por docentes avalados por la institución.

Este requerimiento está asociado a la gestión de los institutos, y consiste principalmente en controlar a los docentes teniendo como principal objetivo que ningún desconocido se pueda establecer como docente de un instituto del cual no forma parte. Por tal razón es que, cuando un docente configure en su perfil que forma parte de cierto instituto, dicha configuración quedará pendiente a que un usuario, con rol administrador lo apruebe. A su vez, un rol administrador tendrá la posibilidad de asignar y des-asignar roles de admi-

nistrador y de esta forma tener un control auto-gestionado de los docentes pertenecientes a un instituto educativo.

[Req. 031] Como administrador del sistema quiero que mi rol sea exclusivo para poder gestionar un instituto.

Para abordar este requerimiento se le asignarán a ciertos usuarios un perfil administrador. Luego de analizar la realidad de las instituciones educativas se decidió que dicho perfil administrador será asignado, en un principio, a algunos docentes dentro de cada institución educativa, teniendo éstos la posibilidad de sumar a dicho rol a otros docentes dentro del instituto. La idea de modelar esta situación de la forma planteada es no tener centralizado en un solo usuario este rol administrativo y así poder hacer la gestión de forma mucho más ágil entre docentes de un instituto.

[Req. 032] Como usuario quiero poder guardar mis configuraciones de dictado sin darlas de alta en un curso para poder probarlas.

Esta necesidad se contempla con el concepto de curso personal. Todos los usuarios de la aplicación tendrán uno por defecto y todos los módulos y configuraciones de dictados que allí se guarden serán privadas. De esta forma los usuarios podrán probar dichas configuraciones (pudiendo escuchar algunos dictados que genera) o incluso generar todo el contenido necesario de un curso y cuando no requiera más modificaciones darlo de alta. Cabe destacar que este último punto es importante ya que una vez que los dictados se dan de alta en un curso los mismos no podrán ser modificados. Esto se debe a que, una vez que se sube el contenido a un curso, algún estudiante podría acceder para comenzar a generar dictados y así comenzar su entrenamiento. Si dicho contenido es modificado por parte del docente, los dictados generados y las configuraciones del dictado serían inconsistentes, y borrar los dictados que los estudiantes ya generaron no es una opción ya que se le estaría borrando el progreso que han tenido.

Capítulo 4

Diseño - UI/UX

En el presente capítulo se abordarán aspectos del diseño de la aplicación los cuales tuvieron lugar desde etapas tempranas del proyecto. Esto se debe a que no solo se hizo foco en aspectos de estética sino que el principal enfoque estuvo en abordar aspectos del flujo de uso, priorizando siempre que la aplicación sea intuitiva y de fácil entendimiento para el usuario.

Desde el lado de la música, el poder llegar a un consenso, junto con personas expertas en el área, de qué parámetros van a ser necesarios configurar desde la aplicación, para dar de alta configuraciones de dictados, fue un trabajo complejo. Esto desencadena que la forma en que el usuario establecerá dichos parámetros no resulte en algo sencillo. Es por esto que se hace especial foco en este aspecto, en cuanto a que la aplicación pueda, de cierta forma, facilitar el entendimiento de los parámetros a configurar y el flujo a través de los diferentes pasos.

Desde el lado educativo se enfatizó el flujo de uso en el que los estudiantes acceden a los dictados generados, ya que esto trae consigo una serie de pasos que implican: reproducción del dictado, acceso a la solución y calificación. En esta área entra también la sección de calificaciones, tanto desde el lado del estudiante como del docente.

Otros aspectos más generales como son la gestión de cursos y la administración de perfiles fueron analizados con una menor prioridad, ya que en la etapa actual, la prioridad estará en las funcionalidades principales de la aplicación.

4.1. Distinción entre UI/UX

Los términos UI/UX hacen referencia a User Interface y User Experience, por sus siglas en inglés, y son muy utilizados en el ambiente del diseño, sobre todo en lo que respecta al diseño de software, tanto aplicaciones para pc como para dispositivos móviles. Cabe destacar que la UX, en la teoría es una práctica no necesariamente orientada a productos digitales a pesar de que son varios los artículos que le hacen referencia utilizando términos provenientes de las industrias digitales solamente. Por otro lado, la UI hace referencia solamente a productos digitales, siendo esta el punto de interacción concretamente entre el usuario y el dispositivo (Lamprecht, 2021).

La User Experience se aplica a cualquier cosa que genere una experiencia en su uso, eso puede ser tanto una aplicación o un celular, como también una maquina de café o una visita al supermercado. Por lo tanto, la UX refiere a cualquier interacción entre el usuario y el producto o servicio con el que se relacione. Esta área del diseño consiste principalmente de las siguientes tareas (Lamprecht, 2021):

- El diseño de la UX consiste en desarrollar y mejorar la calidad de la interacción entre el usuario y el producto a construir
- El área del diseño de la UX no se basa en aspectos visuales, sino que se enfoca concretamente en la sensación que se genera en el usuario al usar el producto

Por lo mencionado, se puede decir que la UX es una agrupación de tareas con el objetivo de optimizar el uso eficaz y proporcionar sensaciones positivas al usar el producto. Es acá donde entra en juego el diseño de la UI, que funcionará como una tarea más para cumplir los objetivos que se propone el diseño de la UX, aportando desde la apariencia.

El diseño de la UI aborda temas sobre los iconos, botones, tipografía, paleta de colores, galería de imágenes utilizadas y cómo todos estos elementos se relacionan entre ellos, considerando también un diseño *responsive*. Ya que el diseño UI refiere solamente a productos digitales, este ultimo término hace referencia a que los elementos presentes en la pantalla estén bien diseñados en diferentes tipos de pantallas, como puede ser la pantalla de una PC, celular o incluso reloj inteligente.

Como tareas principales dentro de la UI se puede destacar las siguientes (Lamprecht, 2021):

- El objetivo principal del diseño de la UI es brindar una guía visual para que el usuario pueda hacer uso de dicho producto digital a través de la interfaz.
- El diseño UI debe transmitir lo que la marca o los dueños detrás del producto deseen que transmita, manteniendo la consistencia y coherencia del producto.

Para poder contemplar los aspectos mencionados y cubrir los puntos claves tanto en el diseño UI como UX se decidió investigar otro tipo de aplicaciones y estudiar cómo contemplan aspectos que son de interés para el proyecto. En particular se investigó en dos líneas principales: softwares de música y aplicaciones móviles de gran uso actual. Las primeras fueron estudiadas para ver cómo resuelven aspectos de usabilidad específicos de la música, como puede ser la forma de escribir notas y figuras musicales. Luego se contrasta con las características que debe tener un buen diseño UI/UX para tomar la decisión de incorporarlas o dejarlas de lado. Por otro lado se estudian aplicaciones móviles de gran uso actual, como pueden ser redes sociales o aplicaciones de entretenimiento. Esto se hace con el objetivo de adquirir flujos de usos, que sean adecuados para la aplicación, los cuales ya estén instaurados y así su uso resulte familiar e intuitivo. Estos aspectos serán explicados en mayor detalle en las siguientes secciones.

4.2. Prototipado

El construir un prototipo permitió, no solo obtener nuevos requisitos, sino que también poder enfocar la energía en abordar temas concretos del diseño. El testear de forma temprana las decisiones tomadas dentro de ésta área permite poder construir un camino sólido para tener un buen diseño de la UX (Pinto, 2018), y es justamente mediante un prototipo que esto es posible.

En primer lugar se construyó un prototipo de baja fidelidad, en el cual se bajó a tierra conceptos adquiridos dentro del área de la música y permitió tomar decisiones sobre el manejo de dichos conceptos. Esto fue la base para luego construir un prototipo de alta fidelidad, con una arquitectura sólida, para poder ir haciendo incrementos y en base a esto construir la aplicación.

4.2.1. Prototipo de baja fidelidad

El prototipo de baja fidelidad se realiza con el objetivo de abordar aspectos generales del sistema sin entrar en grandes detalles. Esto permitió, también, tener algo tangible sobre lo cual abordar conceptos e ideas importantes con personas del área de la música.

El maquetado de este prototipo se realizó en la herramienta *uizard*[8] con un template sencillo y con muy pocos detalles visuales. Esta decisión se tomó dado que el objetivo era abordar temas del diseño UX y discutir flujos de uso de los usuarios, sin perder tiempo en detalles de la interfaz.

En primer lugar, para dar inicio al flujo de uso se tiene la pantalla de bienvenida o inicio (Figura 1.2 Anexo 1).

Si bien esta pantalla no tiene una gran complejidad, permitió validar la idea de que existen dos roles de usuarios diferentes: docentes y estudiantes, quienes, dependiendo con que rol inicien sesión, se les habilitará funcionalidades diferentes. Además al tener la opción de registrarse se dará de alta a un nuevo usuario, tanto estudiante como docente. En el caso de un docente perteneciente a una institución educativa, este deberá ser habilitado por un usuario administrador.

Antes de pasar a abordar temas concretos de las pantallas, se explicarán algunos conceptos que estarán presentes en todas (o al menos en la mayoría) de las pantallas. En la parte superior se tendrá el título correspondiente a cada pantalla y en la parte inferior estará siempre presente la barra de navegación. Ambos conceptos suelen estar presentes en aplicaciones actuales, además de ser adecuadas en la aplicación a desarrollar. En primer lugar, cada pantalla engloba un concepto diferente al resto, por lo que asociarles un título ayuda a fácilmente reconocer en que parte del flujo de uso se encuentra el usuario. Por otro lado la barra de navegación engloba las ideas principales de la aplicación. Las dos grandes núcleos son la reproducción de dictados y la creación de los mismos, estando, estas funcionalidades, dentro del primer y segundo menú. Como tercer y cuarto menú se tiene la sección de calificaciones y de perfil, ambos necesarios para brindarle información de utilidad al usuario y poder tener una adecuada administración de los elementos que maneja la aplicación. Esto último se refiere a que dentro del perfil se tendrán funcionalidades de configuración o edición respecto al usuario.

[8]https://uizard.io/ sitio web de la herramienta.

4.2.1.1. Reproducción de dictados

Una de las funcionalidades principales consiste en poder escuchar los dictados que la aplicación genera. Esto es algo que está presente solamente para los usuarios estudiantes y es la funcionalidad principal para este tipo de usuarios, por lo que, una vez logueados en la aplicación, se los redirige a la primera de las dos pantallas que se muestran en la Figura 1.3 Anexo 1.

En esta pantalla se encuentras varios conceptos importantes. En primer lugar, sobre la parte superior se tiene una sección *scrollable* de forma horizontal, en la cual aparecerán todos los cursos a los que esté inscripto el estudiante. De esta forma siempre habrá uno seleccionado y en base éste se muestra su contenido. Cabe destacar que si el estudiante no se encuentra inscripto a ningún curso aparecerá por defecto seleccionado su curso personal.

Esta sección, en donde se *scrollea* a través de los cursos a los que se está inscripto, es un diseño muy utilizado actualmente en aplicaciones modernas como los son Instagram, LinkedIn o Twitter. En estas aplicaciones, este diseño se utiliza para que el usuario pueda ver las "historias"de otros usuarios, lo cual consiste en listar los perfiles que el usuario sigue (en esta sección *scrolleable* horizontalmente) y al seleccionar uno se mostrarán las fotos que subió a la sección de "historias". En la aplicación desarrollada, la funcionalidad consiste en que al presionar un curso, éste quede seleccionado y todo el contendido, como son los módulos, configuraciones y dictados, sean los pertenecientes a dicho curso. Si bien son dos funcionalidades diferentes (en comparación con las aplicaciones mencionadas), se decidió incorporar este diseño ya que la forma de navegabilidad resultará familiar para el usuario. Además, dado el público al que está dirigida la aplicación, es probable que la mayoría de usuarios tengan contacto o estén de cierta forma familiarizados con las aplicaciones de redes sociales mencionadas. Esto hará que el software a desarrollar resulte intuitivo y familiar al momento de usarlo. Finalmente, el primer elemento de esta sección horizontal será diferente al resto. Éste, en lugar de representar un curso, cumplirá la función de inscribirse a un nuevo curso, por lo que, al ser presionado, se mostrará una ventana en la que se tendrá la opción de desplegar y acceder a todos los cursos a los que el estudiante desee inscribirse. De esta forma se tiene todos los elementos relacionadas a la gestión de cursos concentradas en un sólo lugar, facilitándole al usuario el acceso a las mismas (Ripalda y Garrido, 2020).

Continuando con el análisis de las pantallas, (Figuras 1.3 del Anexo 1) se

tiene una lista de módulos, y a través de cada uno de éstos se puede desplegar la lista de configuraciones de dictados que contengan. Todo este contenido listado depende del curso que se tenga seleccionado en el panel superior.

Al seleccionar una configuración de dictado se redirige a la siguiente pantalla del flujo la cual contiene los dictados propiamente dichos. La primera vez que un estudiante acceda a una configuración de dictado, la aplicación genera cinco dictados de forma aleatoria. En caso de que el estudiante desee seguir practicando con más dictados, existe un botón ubicado en la parte inferior derecha que da la posibilidad de seguir generando cuantos dictados se quiera, siempre en base a la configuración seleccionada.

Cuando el estudiante desee entrenarse, deberá seleccionar un dictado lo cual lo redirigirá a las siguientes pantallas del flujo (Figuras 1.4 del Anexo 1).

En la en la primer pantalla aparecen representados varios conceptos de la música, los cuales son reordenados en función de cómo son utilizados por los usuarios. La ubicación de cada elemento es algo a tener en cuenta ya que el sistema debe comunicarse en el mismo idioma que los usuarios de la aplicación, haciendo que la información aparezca en un orden lógico para este (Ripalda y Garrido, 2020). A esto se le suma que la lectura en dispositivos digitales, por parte del usuario, se realiza en forma de escaneo, siguiendo ciertos patrones. Existen estudios que muestran que en la lectura de un sitio web, los usuarios suelen seguir un patrón en F o patrón en Z (Moreno, 2021). El primero consiste en que las personas suelen dar un barrido lector en los primeros párrafos del sitio, de forma horizontal y luego continúan con su escaneo hacia el final del sitio por el costado izquierdo. El patrón Z suele aplicarse en sitios web donde no necesariamente está presenta la lectura en forma de párrafos, sino que existe otro tipo de interacción con el software. Esto consiste en que el ojo del lector primero escanea la parte superior del sitio, en donde puede encontrar información importante, luego hacia la esquina inferior izquierda, en busca de generar alguna acción, terminando sobre la parte inferior derecha, buscando terminar el flujo de uso de la pantalla actual, lo cual sigue la forma de la letra Z.

Si bien el software desarrollado no es un sitio web, se entiende que el usuario seguirá un patrón Z ya que interactúa con la aplicación en busca de un objetivo: realizar ejercicios auditivos. Es por esto que, sobre la parte superior de la pantalla aparece el primer elemento que el usuario necesita para comenzar con su ejercicio: la nota de referencia escrita en un pentagrama. Al terminar con

este barrido lector horizontal, y continuando con el patrón en forma de Z, es que el usuario tiene la opción de reproducir dicha nota con el botón "Escuchar referencia". Con esto el usuario tiene la información necesaria para reproducir el dictado y poder escribir en una hoja pentagramada la solución. Finalmente se encuentra, sobre la parte inferior de la pantalla, el botón de "Ver resultado" para acceder a la solución.

Como último paso del flujo, al acceder a la solución, se redirige a una pantalla en donde se sigue el mismo patrón anterior. En la parte superior se muestra un pentagrama con la solución, esta vez conteniendo un *scroll* horizontal para poder visualizar todas las notas graficadas. Seguido a esto se tiene la opción de marcar si el dictado fue correcto o en caso contrario la cantidad de errores que se tuvo. Al presionar "Finalizar revisión" se le asigna una nota en base a los errores que haya marcado y se lo redirige a la lista de dictados. Cabe destacar que el rango de errores que se muestra en la pantalla es solo informativo, siendo este calculado en función de la cantidad total de notas del dictado. Por ejemplo, si se tiene un dictado de 50 notas, las opciones que se tendrán disponibles serán las mostradas en la siguiente tabla, donde cada fila representa una opción diferente para elegir. La columna de la izquierda muestra la forma de calcularlas y la columna de la derecha lo que se muestra en base al ejemplo:

Forma de calcular	**Ejemplo**
-	Todas correctas
De 1 al (`10 % del total de notas`)	De 1 a 5 errores
Del (`10 % del total de notas + 1`) al (`20 % del total de notas`)	De 6 a 10 errores
Del (`20 % del total de notas + 1`) al (`40 % del total de notas`)	De 11 a 20 errores
Del (`40 % del total de notas + 1`) al (`65 % del total de notas`)	De 21 a 33 errores
Más del (`65 % del total de notas`)	Más de 33 errores

Se siguió esta escala ya que resulta de mayor interés, tanto para el estudiante como para el docente, saber la cantidad de errores con mayor exactitud cuando no tuvo un gran número de equivocaciones. Además, las opciones de autoevaluación se presentan a modo de selección (que el usuario seleccione qué rango de errores tuvo) y no se le pide que escriba exactamente la cantidad de errores ya que esto podría resultar tedioso. La idea es hacer de esta funciona-

lidad algo sencillo para que el usuario no se saltee esta autoevaluación ya que esto se convierte en datos de utilidad tanto para el propio estudiante como para el docente que dicta el curso.

4.2.1.2. Creación de dictados

Además de la reproducción de dictados, el poder dar de alta una configuración de dictado (a partir del cual se generarán dictados) es de las funcionalidades más importantes de la aplicación. En esta sección se abordará la interacción que tiene que tener el usuario con el sistema para lograr dar de alta una configuración de dictado. En capítulos siguientes se tratará en detalle los algoritmos utilizados para la generación aleatoria de estos.

Si bien la creación de configuraciones de dictados está presente tanto en usuarios con el rol de estudiante como docente, es una funcionalidad dirigida principalmente a usuarios que estén dictando un curso. De todas formas, para ambos usuarios se tendrá en el menú de navegación inferior una sección, representada en los diagramas con el texto "Crear Dicts.", la cual redirige a las pantallas de la Figura 1.5, Anexo 1.

En esta pantalla se encuentras dos secciones: la pestaña de "Configuración melódica" y la de "Configuración rítmica". Sobre la parte superior se encuentra la sección de configuraciones generales, esto consiste en que allí se muestra el instituto, el curso y el módulo al que pertenecerá la configuración de dictado al crearse. Además, tocando en dicha sección se puede establecer dicha información además de asignarle un nombre y una descripción a la configuración de dictado. Cabe destacar que una vez un usuario crea una configuración de dictado, el instituto, curso y módulo que seleccionó quedan guardados en el *storage* del celular. De esta forma, la próxima vez que este usuario vuelva a crear una configuración de dictado, le aparecerá preseleccionado estas opciones, ya que es probable que, al momento de configurar un curso, este proceso se repita.

Por debajo de esto se tiene la opción de seleccionar si se quiere configurar un dictado puramente rítmico, esto hará que la pestaña de "Configuración melódica" no sea accedida, ya que solo serán necesarias las configuraciones rítmicas. En caso de no tener seleccionada esta opción se estará configurando un dictado melódico para lo cual serán necesarias ambas pestañas.

Dentro de la configuración melódica se tienen cinco secciones las cuales ma-

nejan conceptos diferentes al momento de crear una configuración de dictado:

- **Giros melódicos**
  Al presionar el botón "Agregar" se tiene la opción de ingresar un nuevo giro melódico, asignándole una prioridad de 1 a 5. Esto permitirá hacer mayor o menor énfasis en ciertos giros melódicos al momento de la generación del dictado.
- **Notas inicio, Notas fin**
  Tanto para las notas de inicio como las notas de fin se establecen un conjunto de notas musicales, de las cuales, al momento de generar el dictado, se tomará de forma aleatoria una nota de cada grupo, en la cual deberá comenzar y terminar el dictado.
- **Clave sol, Clave fa**
  A cada una de estas se les asigna una prioridad de 1 a 5. Esto determinará la probabilidad con que el dictado sea generado en dicha clave.
- **Tonalidades**
  En esta sección se tiene la opción de desplegar las 13 tonalidades musicales que maneja la aplicación y asignarles una prioridad de 1 a 5 a cada una. En base a esto es que se determinará la tonalidad del dictado generado.
- **Nota de referencia**
  Esta es una única nota fija la cual podrá sufrir una alteración al momento de generar el dictado, en función a la tonalidad en que se genere éste.

Por otro lado, se tiene la configuración rítmica la cual es manejada de forma similar. El primer conocimiento que el usuario debe saber es si desea configurar un dictado simple o compuesto, ya que en función de dicha decisión son los datos que se muestran en secciones siguientes. Una vez establecido esto se tiene tres secciones identificadas por conceptos musicales diferentes:

- **Compás**
  Aquí se selecciona, por un lado los compases en que se podría generar un dictado, cada uno de estos con una probabilidad asociada de 1 a 5, y por otro lado el número total de compáses que tendrá el dictado generado.
- **Células rítmicas**
  En esta sección se deben agregar células rítmicas, cada una de éstas con una probabilidad de 1 a 5. Al generarse el dictado, estos elementos serán

tomados teniendo en cuenta, además de la probabilidad asociada, si la célula rítmica elegida, junto con el resto de células rítmicas del dictado, es coherente con el compás.

- **BPM**
  Se define un rango de bpm para la figura negra, de esta forma, al generarse el dictado se tomarán valores aleatorios dentro de este rango. Además, en base a los bpm de la figura negra es que se calculan los bpm del resto de las figuras.

Cabe aclarar que tanto en las configuraciones melódicas como en las rítmicas se maneja el concepto de probabilidad, en donde 1 significa poco probable y 5 muy probable de que dicho elemento aparezca en los dictados generados (o que aparezca con mayor frecuencia).

Un buen diseño debe permitir revertir acciones realizadas ya sea volviendo a la pantalla anterior o borrando contenido establecido previamente (Gong y Tarasewich, 2004). En las imágenes presentadas como prototipo para esta sección se puede ver un icono (a modo representativo solamente) al costado de cada elemento que al presionarlo permite modificar los datos correspondientes. Esto es de utilidad para modificar datos ingresados erróneamente o incluso borrarlos.

Todo lo mencionado anteriormente corresponde a establecer una configuración de dictado por parte del usuario, pero adicionalmente, en el botón de "Consultar configuraciones" se tiene la opción de acceder a otras que ya fueron dadas de alta por otros usuarios. Este botón redirige a una pantalla en la cual se listan todas las configuraciones de dictado, teniendo opciones de filtrado. Una vez se seleccione una, todos los parámetros son cargado en las pantallas mostradas, así se le da al usuario la opción de modificar los elementos que desee.

Una vez se presione el botón "Crear" se redirige a una pantalla en la cual se muestra un resumen de la configuración del dictado establecida. Esto se hace con el objetivo de que el usuario pueda verificar la correctitud de los parámetros establecidos y en caso de aprobar, la configuración del dictado quedará dada de alta. Esta última pantalla no fue representada en el prototipo de baja fidelidad ya que se trata mayormente de tareas correspondientes al diseño de la interfaz y cómo se le presentará dicha información de la forma más ordenada posible al usuario.

4.2.1.3. Acceso a dictados - Docentes

Se vio cómo los estudiantes pueden reproducir los dictados y cómo los docentes dan de alta configuración de dictados. Ahora bien, es deseable que los docentes puedan escuchar dictados que la aplicación genera en base a sus configuraciones ya que es una forma de evaluar si realmente es lo que desean abordar en su clase. Por este motivo es que, en el menú "Dicts.", para los usuarios docentes, se los redirige a un flujo de uso (ver Figura 1.6 en Anexo 1) en la cual se tiene la posibilidad de escuchar un dictado.

La primer pantalla del flujo muestra los módulos y las configuraciones de dictado, análoga a la que tienen los estudiantes, excepto que al seleccionar una configuración de dictado se lo llevará a una pantalla en donde pueda ver un resumen de dicha configuración. Además de poder ver todos los parámetros establecidos se tiene un botón "Escuchar", el cual genera un dictado de forma aleatoria y llevará al usuario a la siguiente pantalla. Esta tiene un formato similar a la presentada en el flujo de uso del estudiante, con la diferencia que el dictado será siempre generado en el momento y no será almacenado ya que funciona a modo de poder escuchar un ejemplo de dictado. Además en el botón "Ver pentagrama", al igual que en el flujo del estudiante, se redirige a una pantalla que contiene el dictado graficado en un pentagrama, pero sin las opciones de auto calificarse.

4.2.2. Prototipo de alta fidelidad

En esta sección se aborda el prototipo de alta fidelidad, el cual fue diseñado con una arquitectura sólida y con las tecnologías en que se desarrolló la aplicación. Esto permitió realizarle incrementos hasta obtener como resultado la aplicación deseada. Además se abordaron, no solo temas de diseño, sino también cuestiones relacionadas a los flujos de uso, en concreto, cómo se resuelven ciertas casuísticas desde el lado visual.

4.2.2.1. Decisiones básicas del diseño

Existen decisiones de diseño, como puede ser la paleta de colores o aspectos relacionados a la tipografía, que lo ideal es que sean tomadas por personas expertas en el área. El contactar con alguien relacionado al tema quedará planteado como un aspecto deseable a mejorar ya que en el presente proyecto

no se tuvo la oportunidad de vincularse con personas de dicha índole, por lo que las decisiones fueron tomadas en base a autores conocedores del tema.

En cuanto al tipo de letra utilizada y el tamaño, se decidió utilizar las fuentes por defecto de los sistemas operativos en la cual corre la aplicación. Esto es, para los dispositivos con iOS la fuente de la aplicación es *San Francisco* mientras que para Android es *Roboto* (Kennedy, 2021). Si bien esto puede que no sea lo ideal, el utilizar las fuentes que vienen por defecto en los sistemas operativos de los dispositivos optimiza la lectura, haciendo que la aplicación se vea nativa (Nieskens, 2017). En base a cada una de las fuetes se elije el tamaño, ya que dos fuentes diferentes podrían requerir tamaños diferentes para optimizar la lectura y cumplir con el propósito de los diseñadores de cada sistema operativo. Es por esto que los tamaños de la fuente son tomados en base a las recomendaciones de diseño que da Apple[9] y Google[10]. Este último creó el sistema de diseño de *Material Design*, utilizado por los dispositivos con sistema operativo Android. Los tamaños de fuente más utilizados en la aplicación son representados en la siguiente tabla:

Elemento	Tamaño en iOS	Tamaño en Android
Título	22pt	24sp
Subtítulo	20pt	20sp
Cuerpo 1	17pt	16sp
Cuerpo 2	15pt	14sp

En cuanto a las medidas utilizadas, la conversión de puntos (pt) a píxeles escalados (sp) para iOS y Android respectivamente son uno a uno (*Material Design*).

Los tamaños de fuentes anteriores son utilizados en la mayoría de los textos de la aplicación, existiendo alguna excepción en casos particulares en donde, por cuestiones estéticas, se decidió tomar un valor no pertenecientes a esta tabla.

El otro aspecto de diseño que se aborda en la presente sección es la paleta de colores a utilizar, lo cual condiciona tanto el diseño de las pantallas como el logo de la aplicación. Esta decisión se basó principalmente en los aspectos que se quiere que la aplicación transmita a sus usuarios. Por un lado se sabe que

[9]https://developer.apple.com/design/human-interface-guidelines/ guías de diseño de interfaz ofrecida por Apple para sus dispositivos.

[10]https://material.io/design/guidelines-overview guías de diseño que ofrece Google a través de *Material Design*.

la principal temática del software es la música y por el otro se tiene que ésta es abordada desde el lado de la enseñanza. Estos dos conceptos se unen en la aplicación logrando automatizar la enseñanza de conceptos musicales a través de la generación de dictados musicales. Esto quiere decir que el entrenamiento auditivo de los estudiantes será bueno, siempre y cuando la aplicación sea un buen recurso, por lo que es necesario que ésta logre transmitir seguridad y confianza, tanto al docente como al estudiante, de que los ejercicios generados son de calidad. Esto se vincula fuertemente con que la aplicación pueda ser vista como una herramienta, la cual cuenta con las últimas tecnologías del sector. Estos aspectos ayuda a generar la seguridad necesaria en instituciones educativas y así lograr la incorporación de dicho software como una herramienta más para la educación.

Por lo mencionado, se quiere que el factor tecnológico y moderno sea algo presente en la aplicación. Esto lleva a estudiar los centros tecnológicos de mayor éxito en el mercado (no necesariamente relacionados con la música o enseñanza), de lo cual se obtiene que los cuatro colores más utilizados en compañías líderes en el sector tecnológico son: azul, blanco, negro y rojo. Dentro de ésta gama de colores, el azul transmite seguridad, confianza y conocimiento (BATHI, s.f.), aspectos que resulta interesante transmitir. Si bien se sabe que dicho color es utilizado por la gran mayoría de las empresas tecnológicas, la aplicación se distingue del resto de marcas por las diferentes tonalidades de éste color, presente en la Figura ??.

Cabe destacar que los elementos que se quiere que la aplicación transmita no estarán solamente abordados desde la selección de colores, sino que existen otros aspectos del diseño que cobran importancia.

A la paleta de colores presentada se le suman los colores utilizados para las alertas. Estas son utilizadas para informar al usuario cuando una operación es exitosa, cuando hay una advertencia o existe algún error o inconsistencia en la operación, representados con el color verde, amarillo y rojo[11], acompañado de iconos informativos.

4.2.2.2. Pantalla de inicio

En las siguientes secciones se presentan algunas pantallas del resultado final de la aplicación para abordar las principales decisiones de diseño que fueron

[11] Para ver en detalle los colores ver Anexo, la sección correspondiente a diseño, en donde se presentan los colores utilizados para las alertas junto con su código en hexadecimal.

Color	Importancia	Código
	Primario	#005f89
	Secundario	#0093b7
	Terciario	#63cdd7
	Cuarto	#a0c5cf
	Quinto	#e6eceb

Figura 4.1: Paleta de colores

tomadas a lo largo del desarrollo. Las imágenes presentadas son extraídas desde un iPhone 12 Pro Max aunque cabe aclarar que durante el desarrollo todas las pantallas fueron probadas en una gran variedad de tamaños de pantalla. De esta forma se asegura tener una buena coherencia ya que múltiples dispositivos estarán alojando la aplicación, y el hecho de que el software responda de forma correcta a diferentes tamaños de pantalla es una buena práctica a tener en cuenta al abordar el diseño (Gong y Tarasewich, 2004).

Al abrir la aplicación sin estar registrado se mostrará la pantalla de inicio en la cual aparecen varios elementos importantes (Figura 1.2 del Anexo 1)

En primer lugar, ocupando una gran parte de la pantalla, se tiene el logo de la aplicación el cual fue diseñado teniendo en cuenta la paleta de colores presentada anteriormente. En cuento a los elementos presentes en el logo, fueron seleccionados unos *sticks* (palitos de batería) como elemento musical, dando la sensación que golpean entre ellos. Este sonido está presente al comienzo de cada dictado generado por la aplicación, sirviendo para marcar los compases, lo cual puede ser visto como la preparación previa a escuchar un dictado musical. Esta es la razón por la que se eligió este elemento musical como logo, ya que se quiere generar la misma sensación de preparación al iniciar la aplicación y ver el logo.

Seguido de esto se tiene el login, en el cual se podrá iniciar sesión como

Figura 4.2: Pantalla de inicio

estudiante o docente. Ese *switch* que se tiene para poder iniciar con un rol u otro está presente dado que para un mismo email un usuario quisiera tener un perfil de estudiante y otro de docente. Esto es de utilidad para algunos casos en que los docentes sean además estudiantes o simplemente deseen tener un perfil con dicho rol para poder seguir profundizando en su entrenamiento auditivo.

Finalmente se tiene la opción de registrarse para los nuevos usuarios de la aplicación. Esto redirige al usuario a otra pantalla en la cual se le piden los datos necesarios, entre ellos si desea crear su perfil como estudiante, docente o ambos.

4.2.2.3. Reproducción de dictados

Las pantallas presentadas en la presente sección (Figura 4.3) son las correspondientes a las presentadas como prototipos de baja fidelidad, en el flujo de uso que tiene un usuario con rol de estudiante que desea reproducir un dictado.

Como se mencionó en los prototipos de baja fidelidad, en la primer pantalla se muestran los módulos y dentro de éstos las configuraciones de dictados, en base al curso que se tiene seleccionado. Para cada una de las configuraciones de dictados se agrega la descripción que le fue establecida. Esto le resultará de utilidad al estudiante para saber qué elementos musicales estará entrenando.

Una vez seleccionada una configuración de dictado se redirige hacia el listado de los dictados. Éstos son identificados con un número correspondiente al orden en que se generaron. Adicionalmente se presenta información de utilidad como es la clave en la que está el dictado, la tonalidad y en caso de que el estudiante haya realizado dicho dictado se mostrará la nota, del 1 al 10, que obtuvo en su último intento. En este punto, además, se utiliza una escala de colores compuesta por el verde, naranja y rojo, correspondiente de la nota más alta a la más baja y así permitirle al usuario identificar de forma rápida su desempeño. Además, estos colores fueron elegidos teniendo en cuenta el generar un buen contraste con el fondo de la pantalla, por tal motivo es que se descartó el uso del color amarillo.

Finalmente accediendo a un dictado se obtiene la pantalla en la que el estudiante llevará a cabo su entrenamiento con la reproducción del dictado (Figura 4.4).

Estas dos pantallas corresponden a la reproducción del dictado y al acceso a su solución respectivamente, las cuales fueron abordadas en el prototipo de

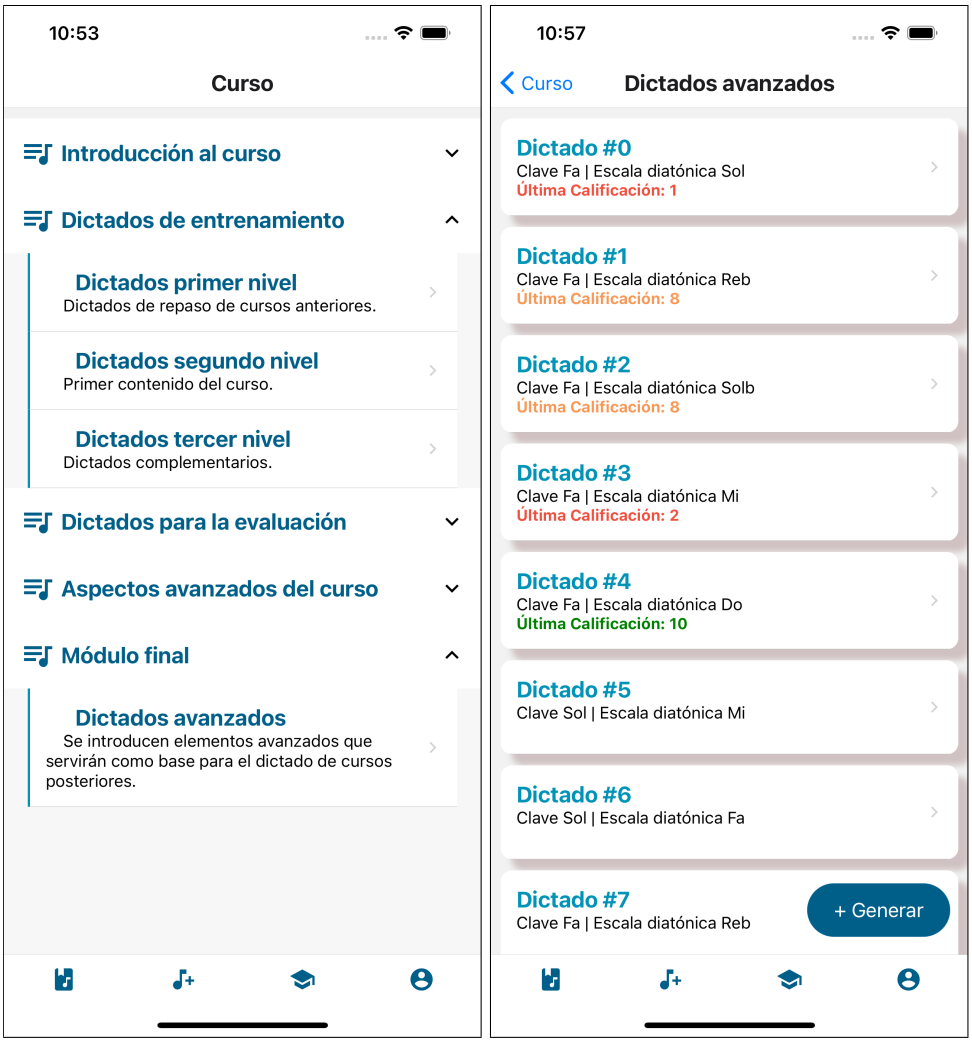

Figura 4.3: Reproducción de dictados

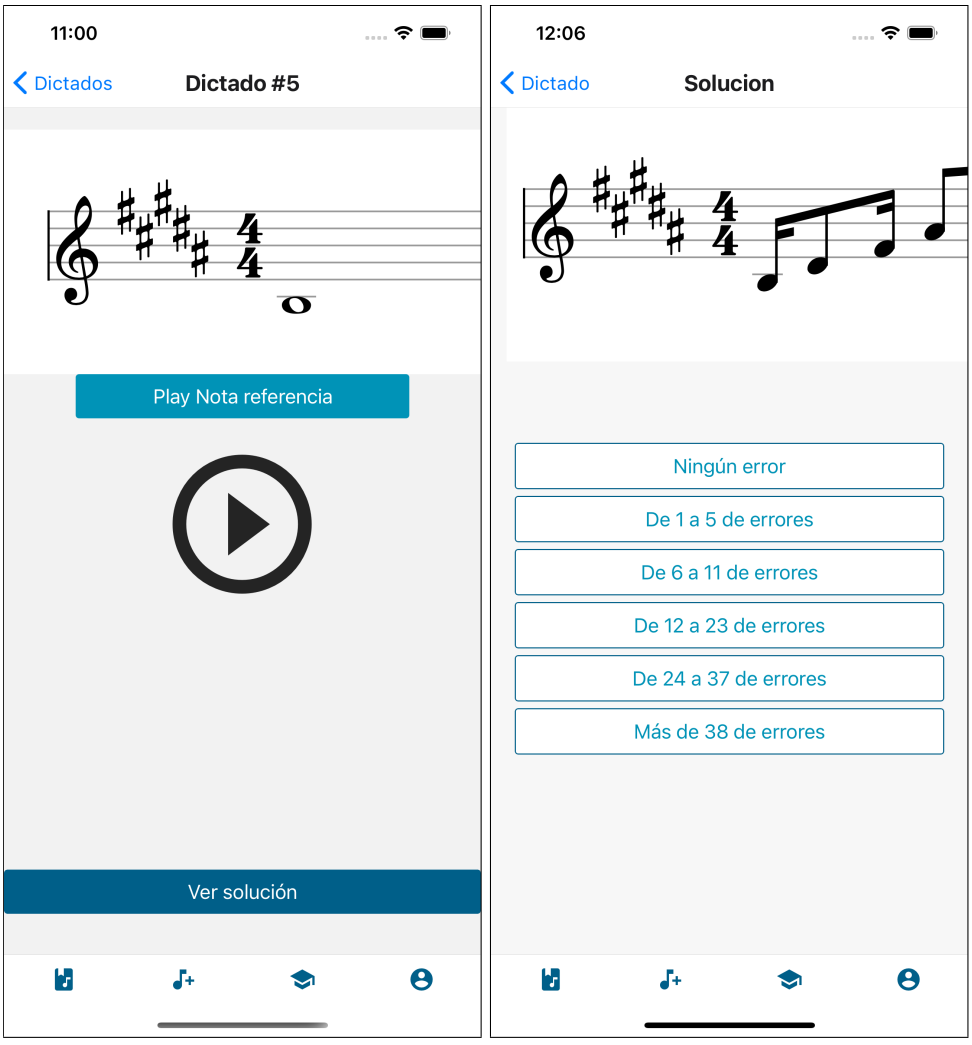

Figura 4.4: Reproducción y entrenamiento de dictados

baja fidelidad presentado anteriormente. En las pantallas presentadas se puede ver la presencia de los diferentes botones utilizados en la aplicación, utilizando cada uno de éstos para dar una prioridad diferente a su funcionalidad asociada. El botón primario, representado en el botón de "Ver solución", contiene el color primario de la aplicación y es utilizado para funcionalidades como redirigir a otra pantalla o confirmar algún evento. El botón secundario está presente en la misma pantalla, con un azul más claro, generalmente utilizado para funciones desplegadas en la misma pantalla como puede ser la apertura de un modal o, como es este caso, reproducir la nota de referencia. El botón terciario, presente en la segunda pantalla, se utiliza en funcionalidades de menor prioridad o cuando dicho botón está presente de forma reiterada en una pantalla, y así no saturar con la presencia de una gran cantidad de colores obscuros.

Cabe destacar que en la pantalla de la derecha, el rango de errores es calculado en función del largo del dictado, como fue explicado en secciones anteriores. Cada uno de estos botones (a excepción del primero) abre un modal en el cual el usuario podrá marcar si sus errores fueron mayormente errores relacionados al aspecto rítmico, melódico o ambos por igual. Este dato resultará de utilidad, tanto a estudiantes como docentes, para saber en que aspecto profundizar al momento de continuar con el entrenamiento auditivo.

4.2.2.4. Creación de dictados

Como ya se mencionó, el otro flujo principal consiste en crear configuraciones de dictados, a partir de las cuales la aplicación genera dictados aleatorios. Para esto se tiene un menú que tendrá una pantalla dedicada a la configuración de los parámetros musicales necesarios, además de establecer parámetros más generales (Figura 4.5) .

Estas dos pantallas contienen todos los conceptos musicales necesarios para para la configuración de un dictado, además de contener aspectos más generales, como puede ser el nombre o la descripción. Dichos conceptos fueron abordados en el prototipo de baja fidelidad, el propósito de presentar estas pantallas es poder tratar algunas decisiones de diseño tomadas.

El principal objetivo de diseño de esta funcionalidad consistió en poder agrupar las funcionalidades de agregar, editar y eliminar cada uno de los conceptos allí presentes en una sola pantalla. Esto se hizo para evitar redirigir al usuario a una nueva pantalla, en la cual tenga que realizar acciones y en caso

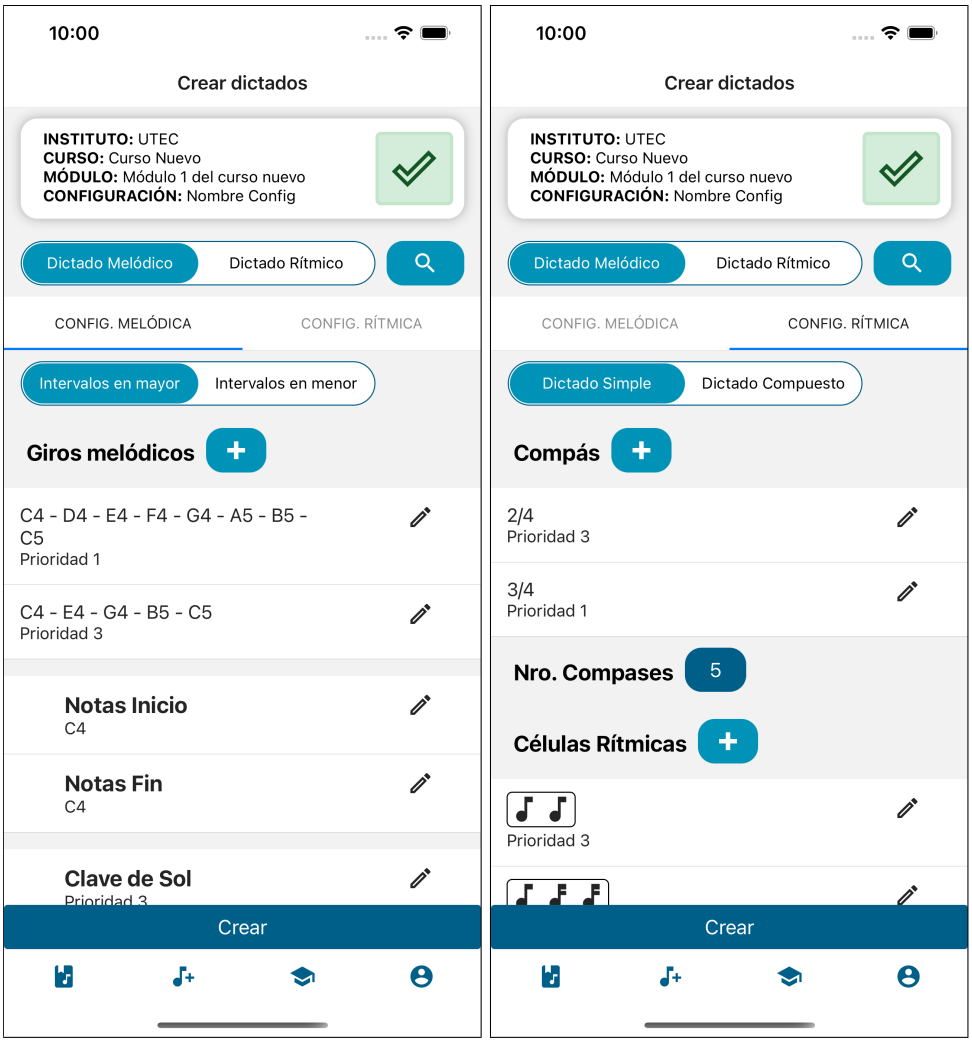

Figura 4.5: Creación de dictados

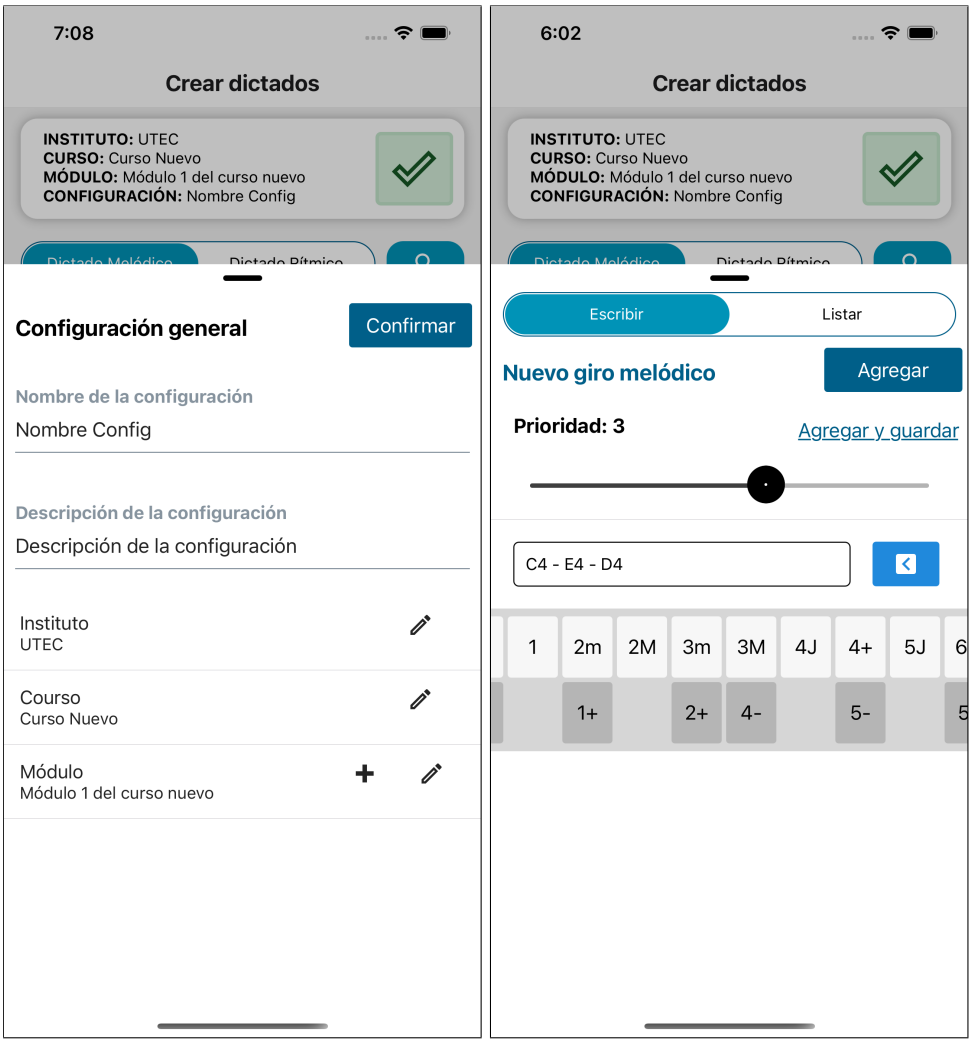

Figura 4.6: Botton sheet

de error volver hacia atrás en el flujo de navegación, ya que en este escenario el usuario podría perder el foco de atención de lo que realmente quiere configurar, agregando una mayor complejidad. Este objetivo se logró utilizando un elemento de diseño denominado *botton sheet* el cual consiste en un panel deslizante que aparece desde la parte inferior de la pantalla. Este aparecerá cada vez que se quiera agregar, editar o eliminar algún elemento de la configuración, dando la sensación al usuario de que siempre permanece en la misma pantalla y ante cualquier equivocación simplemente se cierra dicho elemento deslizando hacia abajo (Figura 4.6).

En estas dos imágenes se puede ver un claro ejemplo del uso de los *bottom sheet*. El primero se despliega al presionar en el panel superior correspondiente a la configuración general, en donde se seleccionará el instituto, curso y módulo

al que pertenecerá la configuración de dictado, además asignarle un nombre y una descripción. Cabe destacar que en dicha pantalla se permite dar de alta nuevos módulos para el curso seleccionado ya que es en este flujo de uso en donde se podría desear agregar un nuevo módulo.

En la segunda pantalla se muestra el mismo panel deslizante, el es utilizado cuando se desea agregar nuevos giros melódicos. En este punto se pueden distinguir tres secciones. La primera consiste en un *switch* en el cual se elige si se quiere escribir un nuevo giro melódico o seleccionar uno de la lista brindada por el sistema. En ambos casos se le asigna una prioridad en la escala de 1 a 5 utilizando un *slider*, elemento gráfico que permite al usuario deslizar el circulo indicador sobre la linea, estableciendo un valor dentro de un rango determinado. En último lugar, si se decidió escribir el giro melódico, se tiene un teclado y un recuadro en donde irán apareciendo lo que el usuario escriba. El teclado fue diseñado en conjunto con estudiantes avanzados de la carrera de música, en donde cada tecla representa un intervalo los cuales dependerán si en la configuración melódica se eligió los intervalos en menor o mayor. Para más información sobre que nota le corresponde a cada intervalo, dependiendo si es menor o mayor, ver Anexo correspondiente a la sección diseño.

Una vez escrito el giro melódico se puede seleccionar "Agregar" el cual añade dicho elemento a la lista de giros melódicos de la configuración del dictado. Al presionar "Agregar y guardar", además de cumplir la función anterior, dicho giro melódico será guardado en el sistema. Esto hará que al elegir la opción de listar los giros melódicos se mostrará dicho elemento. Esto se puede ver en la Figura 4.7

En esta lista de giros melódicos se puede ver que existe un botón para ir seleccionando todos los que se desean agregar, además de ir asignándoles una prioridad.

Todos los elementos vistos en esta última sección son los que están presentes en el resto de los *bottom sheet*. Por ejemplo, todos los elementos que se deban establecer notas musicales, como son notas de inicio, notas de fin y nota de referencia estará presente el teclado de intervalos. En elementos donde se tenga que seleccionar de una lista dada, como ser las tonalidades, los compáses y células rítmicas estará presente el botón a modo *check*, para ir seleccionando los elementos a agregar, además de estar presente el *slider* para asignarle una prioridad a dicho elemento, siempre en el rango de 1 a 5. Los elementos de clave de sol y fa solo contendrán el *slider* de tal forma que si hay una clave

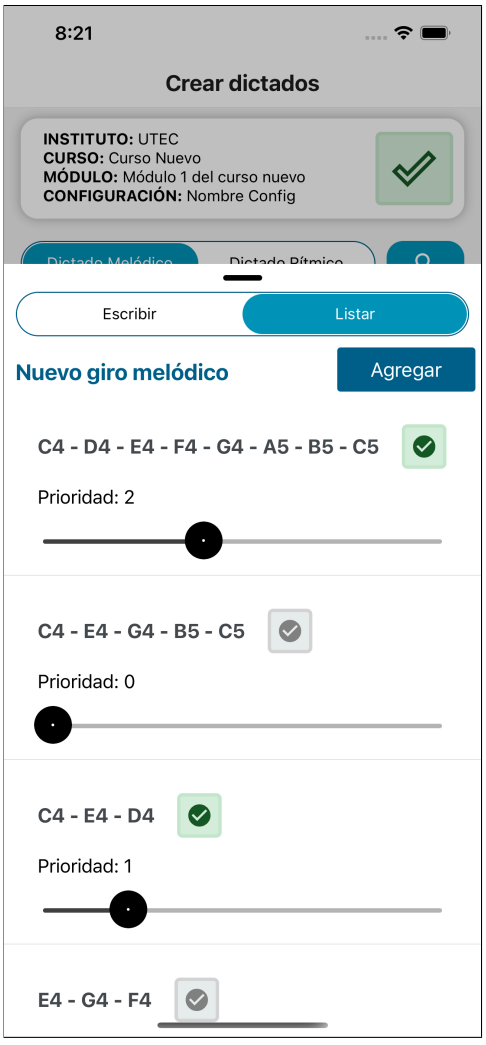

Figura 4.7: Agregar elemento

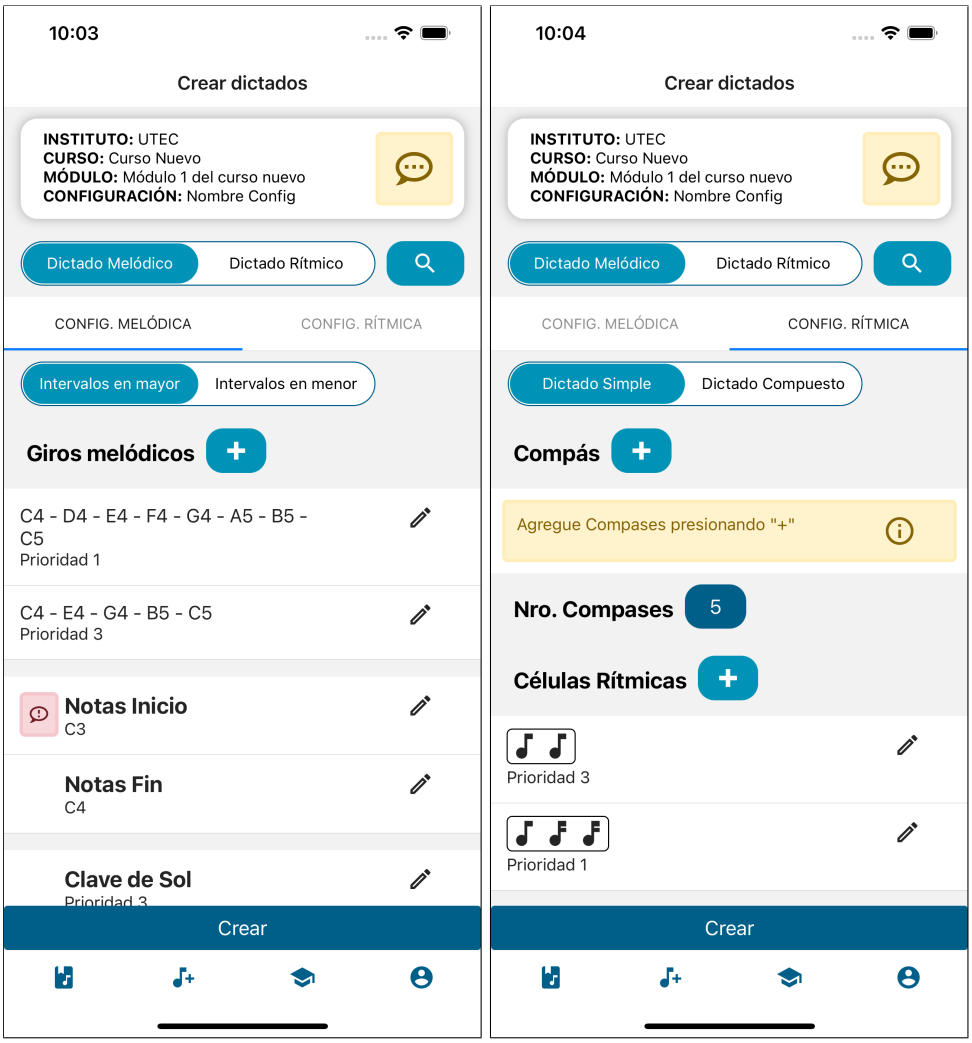

Figura 4.8: Control de inconsistencias

que no se desea simplemente se le asigna una prioridad con valor cero.

Dada la gran cantidad de elementos a configurarse y lo complejo que puede resultar para el usuario tener que pensar en las prioridades o incluso escribir giros melódicos, es que el sistema implementa un control de inconsistencias. Esto se implementa teniendo en cuenta el principio de la prevención y manejo de errores en sistemas móviles, el cual tiene que ser informado al usuario de alguna forma gráfica (Gong y Tarasewich, 2004). A modo de abordar la forma en que son manejadas las inconsistencias se presentan las imágenes (Figura 4.8) correspondientes a la configuración de dictados.

Contrastando dichas pantallas con las presentadas anteriormente, se puede ver dos advertencias (mostradas en color amarillo) correspondientes a la configuración general y a la configuración rítmica, en la sección de compases. Estas

advertencias están visibles cuando existe información faltante y debe ser completada por el usuario. Por otro lado, dentro de la configuración melódica se puede identificar un error en las notas de inicio (mostradas en color rojo). Esos son mostrados cuando existe un dato mal ingresado que no es consistente con el resto de la configuración. En estos casos, al presionar sobre dicho indicador el sistema desplegará una ventana conteniendo un texto informativo de cual es la inconsistencia particular y posibles soluciones para corregirla. Los datos mal ingresados que se controlan son los siguientes:

- No pueden existir notas de inicio o notas de fin las cuales no pertenezcan a un giro melódico, ya que será imposible comenzar o terminar en dicha nota.
- Al menos debe existir una clave (Sol o Fa) con prioridad mayor a cero.
- Al menos debe existir una tonalidad con prioridad mayor a cero.

Cabe destacar que en la escala de colores de advertencia, el color verde es utilizado para brindarle al usuario cuando un dato o algún paso dentro de la configuración es realizado de forma correcta (como puede verse en pantallas anteriores al completar la configuración general), dando así un *feedback* positivo.

Adicionalmente se implementa otro control correspondiente a dichas inconsistencias y es que, al darle a "Crear"teniendo una advertencia o un error, la aplicación mostrará un cartel informativo como el de la Figura 4.9, para que dicha inconsistencia sea revisada.

Finalmente, si todos los parámetros son consistentes, al presionar "Crear"se redirige a la pantalla que se muestra en la Figura 4.10 como último paso del flujo:

Este paso es de valor para el usuario ya que tiene la posibilidad de revisar todos los parámetros configurados de forma resumida. Esto además contempla la posibilidad de revertir una acción, por ejemplo al momento de establecer la configuración de dictados, concepto que es una buena práctica al momento de diseñar una aplicación (Gong y Tarasewich, 2004). Sobre el final de dicha pantalla se tiene el botón "Finalizar.[el] cual creará dicha configuración y la dejará guardada para el curso establecido. Al momento que el usuario ejecuta esta acción, el sistema corre de fondo un proceso el cual, con la configuración establecida, intenta crear un dictado de forma aleatoria. En caso de que lo logre

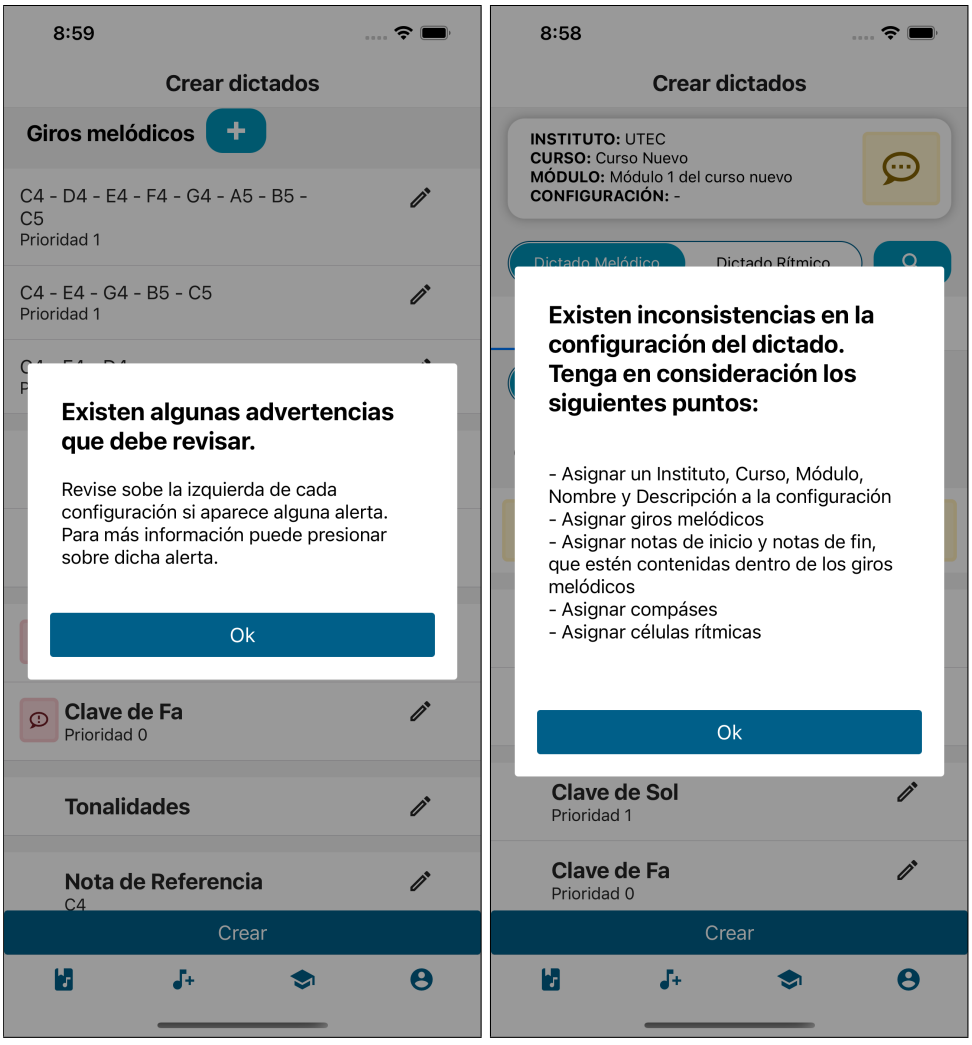

Figura 4.9: Cartel informativo

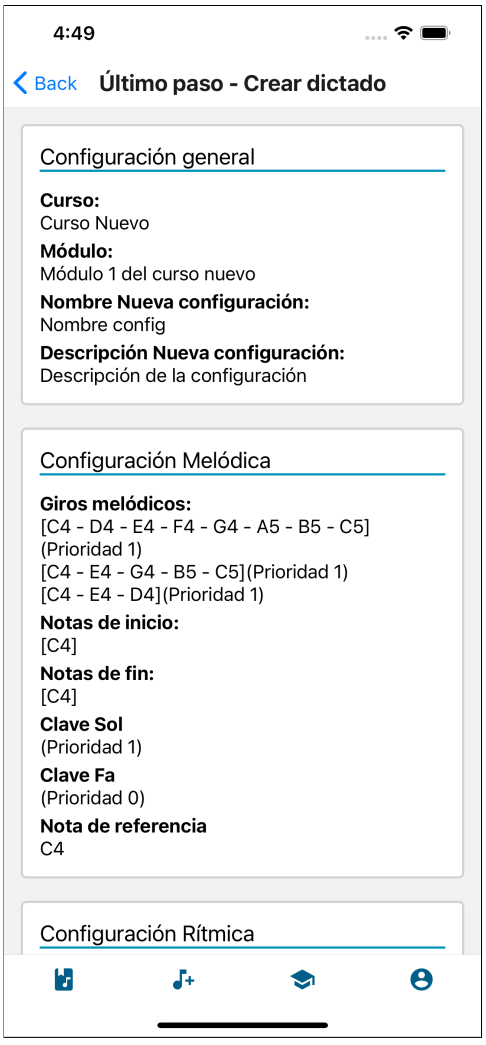

Figura 4.10: Pantalla de resumen del dictado

el flujo termina con éxito, en caso contrario, suponiendo que tras varios intentos, no se logre crear un dictado, la aplicación mostrará un cartel al usuario informando de lo sucedido, junto con algunos consejos a seguir para tener una configuración de dictado exitosa. Esto corresponde a un caso borde que muy rara vez sucede y generalmente se da porque no existe una secuencia de notas que comience y termine en las notas de inicio y fin establecidas, utilizando los giros melódicos dados, en un largo que es determinado por las células rítmicas y compáses configurados. Este es el último paso que ejecuta el sistema para implementar un control correcto control de inconsistencias, asegurando que los dictados serán generados de forma correcta.

Dado que la funcionalidad de crear nuevas configuraciones de dictados está enfocada principalmente hacia los usuarios con rol de docente, se analizó la realidad y contexto en el que éstos completarían dicho flujo. De esto se deduce que la mayor parte del tiempo, un docente creará configuraciones de dictados pertenecientes a un mismo curso, incluso dentro de un mismo módulo. Es por esto que, al finalizar dicho flujo, se guardará en el *storage* del dispositivo móvil el último curso y módulo al que se agregó una configuración de dictado. Así, la próxima vez que se quiera dar de alta una nueva, el usuario ya tendrá preseleccionado dichos datos.

4.2.2.5. Calificaciones y perfil

Por último, en la barra de navegación se tienen las calificaciones y el perfil, correspondientes al tercer y cuarto icono respectivamente. En la primera, si se accede desde un perfil de estudiante, se muestran los cursos a los que se está inscripto junto con la calificación promedio de todos los dictados realizados dentro de este. A su vez, accediendo a un curso se lleva a una pantalla en la que se listan los módulos que contiene, y dentro de cada uno de éstos se listan las configuraciones de dictado, cada uno mostrando la nota promedio obtenida para el usuario. Esta información es de utilidad al momento de evaluar su desempeño y saber en qué tipo de dictados debe continuar entrenándose. Análogamente, para un usuario docente, la pantalla de calificaciones muestra todos los cursos que dicta el usuario, junto con la nota promedio de todos los estudiantes que se han entrenado con dictados dentro de ese curso. Además se listan los módulos y configuraciones de dictados dentro de cada curso, mostrando también el promedio de calificaciones para cada uno. Desde el lado del

docente la idea no es evaluar el desempeño de cada estudiante por separado mediante el uso de la aplicación, sino el tener un *feedback* del desempeño promedio de sus estudiantes y así saber cómo seguir con el dictado del curso.

Por otro lado, para el menú de perfil se decidió dejarlo con lo mínimo y necesario para un correcto uso, dándose solamente la opción para cerrar sesión. En este menú se tiene pensado incluir otras funcionalidades como trabajo a futuro, tema que será abordado en secciones posteriores.

Capítulo 5

Desarrollo

5.1. Metodología de desarrollo

El presente trabajo consiste en un proyecto interdisciplinario, lo que implica tener una constante interacción entre personas pertenecientes a distintas disciplinas, desde músicos, docentes y desarrolladores de software. Esto lleva a que los requerimientos y funcionalidades se irán especificando y clarificando a medida que se consolide un mínimo vocabulario común. Además los requerimientos irán surgiendo y cambiando a medida que se vaya profundizando en ciertos temas, lo cual lo hace un escenario adecuado para el uso de metodologías ágiles de desarrollo. Dichos métodos han sido exitosos en desarrollos de sistemas a medida dentro de una organización y en los que se identifica una participación activa en el proceso de desarrollo por parte del cliente. También resulta adecuado que no existan regulaciones externas que puedan llegar a afectar el desarrollo del software (Sommerville, 2011). Al ser un proyecto interdisciplinario ambas partes estarán sumamente comprometidas, siendo partícipes del desarrollo del software. Además, al interactuar directamente con docentes de institutos de música, esto quita casi en su totalidad cualquier tipo de regulación externa. Todas estas características hace al proyecto un buen candidato a resultar exitoso al ser desarrollado con el uso de metodologías ágiles.

Por otro lado, el uso de las metodologías ágiles traen consigo una serie de principios beneficiosos para el proyecto. La entrega incremental resulta de gran valor ya que al entregar funcionalidades en cada incremento permite ir validando lo desarrollado con personas conocedoras del tema. Además los métodos ágiles permiten el poder adaptarse fácilmente a los cambios, algo fundamental

cuando no se tiene total conocimiento sobre los temas principales del software a desarrollar. Como principio también se menciona que la metodología de desarrollo debe estar orientado a las personas y no a los procesos. Esto quiere decir que debe permitirse trabajar a las personas del equipo sin procesos rigurosos establecidos. Finalmente se menciona como principio el mantener la simplicidad, tanto en el software como en el proceso de desarrollo, lo cual permitirá no agregarle complejidad a este proyecto interdisciplinario.

Si bien en el manifiesto ágil[12] existen doce principios en dicha metodología, fueron mencionado unos pocos debido a que éstos se considera los que mayor impacto tienen dado las características del proyecto.

5.1.1. Marco utilizado

Mientras que una metodología indica qué principios se debe seguir en un proyecto, un marco de desarrollo especifica cómo seguir dichos principios. Gabriel Ledesma en una de sus charlas[13] define a Scrum como *"...un marco de trabajo simple para gestionar proyectos, donde las personas involucradas en el proyecto utilizan soluciones adaptativas para resolver problemas complejos, lo hacen de forma creativa y productiva, entregando frecuentemente valor de negocio al cliente"*.

Scrum es un marco el cual brinda una gran cantidad de reglas además de tener en cuenta una gran cantidad de roles diferentes, lo cual resulta imposible incorporarlo en su totalidad dado el contexto y las personas participantes del proyecto. Aún así existen muchos conceptos los cuales fueron incluidos de forma total o parcial.

5.1.1.1. Planeación

Al comienzo del proyecto existe la etapa de planeación del bosquejo y diseño arquitectónico (Sommerville, 2011), la cual consistió en establecer los objetivos del proyecto. Esto consistió en definir las funcionalidades básicas de la aplicación, quienes iban a ser los usuarios y cómo éstos la iban a utilizar. Teniendo esto en consideración se definió la arquitectura a utilizarse, evaluando pros y contras.

[12] https://agilemanifesto.org

[13] https://open.fing.edu.uy/courses/cis/7 Charla expuesta en la clase de Introducción a la Ingeniería del Software en la facultad de ingeniería en 2019.

5.1.1.2. Scrum Team

Existe el equipo de Scrum el cual estaría compuesto por el equipo de desarrollo, el dueño de producto y el *scrum master*. Debido al número reducido de integrantes encargados del desarrollo de la aplicación se decidió no tener roles definidos y, si bien por momentos existían tareas más orientadas a la definición del producto o al desarrollo, los integrantes contaron con la capacidad de poder asumir una u otra tarea durante cualquier etapa del proyecto.

5.1.1.3. Artefactos

En primer lugar existe el *product backlog* el cual consiste en un conjunto compuesto por todas las historias de usuario que fueron creadas en base a los requerimientos, y en segundo lugar se deriva el *sprint backlog*, el cual lo forman aquellas historias de usuario que serán implementadas en un sprint dado. El *sprint backlog* era definido en una reunión junto con personas del área musical además de ser priorizados. Posteriormente, los integrantes encargados del desarrollo desglosaban dichas historias de usuario en un conjunto de tareas.

En el momento que las historias de usuario, las cuales habían sido definidas en conjunto, se finalizaban, es que se tenía un incremento. Esto se refiere a tener software funcionando, el cual era presentado a personas del área musical para poder ir validando estos avances sobre el producto para saber que una funcionalidad había sido cubierta.

5.1.1.4. Sprint

Los sprints son ciclos de ejecución de corta duración en el cual el objetivo es conseguir un incremento, lo que se traduce a agregar valor a el producto que se esta construyendo. Si bien el marco de scrum establece que los sprints tengan una duración fija, dado el contexto del presente proyecto, la duración de estos varió en función de la etapa del proyecto. Al comienzo, al tratarse de un proyecto interdisciplinario, los sprint tuvieron una duración de una semana, en la cual se realizaban pruebas de concepto, las cuales se validaban con las personas conocedoras del tema. Cabe destacar que este desarrollo fue realizado en las tecnologías que luego utilizaría la aplicación (concretamente la API) para que el mismo pueda ser reutilizado más avanzado el proyecto. Una segunda etapa del proyecto dio comienzo una vez los integrantes, encargados del desarrollo del software, adquirieron un conocimiento suficiente en el tema. En este

punto los sprints pasaron a tener una duración fija de dos semanas, teniendo la posibilidad de presentar un mayor incremento en el valor del producto al final de cada uno de estos ciclos.

Independientemente de la duración de los sprints, estos siempre están compuestos por cuatro etapas: valoración, selección, desarrollo y revisión (Sommerville, 2011). La primera consiste en una reunión de todo el equipo, en la cual se revisaba el *product backlog* y se definía qué era lo prioritario desarrollar, identificando posibles riesgos además de definirse nuevos requerimientos o tareas. En la misma reunión se abordaba el tema de la selección, es decir, identificar en conjunto qué funcionalidades se iban a implementar en el transcurso de dicho sprint. La etapa de desarrollo involucra a los integrantes encargados del desarrollo de la aplicación. Previo al inicio de esta etapa, dichos integrantes se reunían para desglosar las historias de usuario a abordar en un conjunto de tareas para luego dar comienzo efectivamente a esta etapa. Finalmente, al dar por cerrado el sprint se reunía todo el equipo para presentar los avances, revisar detalles y, por cuestiones de tiempo de los integrantes, ya se daba comienzo al nuevo sprint.

Durante la etapa de desarrollo estaba presente las *daily scrum*. El marco de scrum establece que éstas son reuniones diarias de un máximo de 15 minutos en las cuales se abordan tres temas: que hizo el integrante ayer, que hará hoy en vista de lograr los objetivos del sprint y si existe algún posible impedimento. Dichas reuniones eran realizadas de forma informal y en el caso de que existiera algún tema a tratarse en conjunto se organizaba un encuentro para el tema en concreto.

5.1.1.5. Cierre del proyecto

El cierre del proyecto se dio una vez la aplicación fue publicada en la tienda, tanto para dispositivos Android como iOS, además de la publicación, tanto de la base de datos como de la API. Esto estuvo acompañado de un primer uso efectivo por parte de estudiantes avanzados de la carrera de música, quienes la utilizaron para configurar un curso introductorio. De esta manera en la aplicación queda con datos pre-cargados y pronto para que pueda ser utilizado.

5.2. Modelado de la realidad

Modelar la realidad de forma gráfica es una herramienta que permite analizar en mayor profundidad qué aspectos tiene que tener en cuenta el software. Además un modelado gráfico (por ejemplo un modelo entidad relación) es una representación abstracta de la realidad que permite analizarla en un lenguaje no demasiado técnico.

Un modelo entidad relación (MER) consiste en representar de forma gráfica las entidades de la realidad y la relación que existen ellas, teniendo cada elemento un conjunto de atributos asociados. Si bien el modelado entidad relación es utilizado como una herramienta básica para el modelado del diseño de la base de datos (Badia, 2004), es un recurso que puede ser utilizado para definir en conjunto con el cliente el modelado de la realidad. En particular, dicho modelo fue presentado a docentes de música, siendo de gran utilidad al momento de definir temas relacionados a la gestión de instituciones, cursos, estudiantes y docentes.

En la Figura 5.1 se presenta el modelado entidad relación, sirviendo como apoyo para explicar cómo es modelada la realidad.

Como restricciones no estructurales al diagrama presentado se tiene:

- Un usuario no puede cursar y dictar una misma materia
- No puede haber docentes o estudiantes que dicten o cursen un curso personal
- Un usuario sólo puede crear configuraciones de dictado la cual pertenezca a un módulo dentro de un curso el cual:
  - El usuario es docente y dicta dicho curso
  - El usuario es dueño del curso personal

La gestión pilar de dictado de cursos nace en la entidad Instituto, la cual va a representar las diferentes instituciones educativas. Cada una de éstas va a contener varios cursos, al mismo tiempo que cada curso va a estar compuesto por un conjunto de módulos. La idea de éstos es que sirvan para organizar un curso en diferentes temas o elementos musicales a entrenarse. Los módulos contendrán diferentes configuraciones de dictados las cuales serán creadas y asignadas a un módulo por un usuario de la aplicación. Los usuarios con rol docente podrán dar de alta configuraciones de dictado dentro de módulos que pertenezcan a: cursos que son dictados por dichos usuarios o a su curso

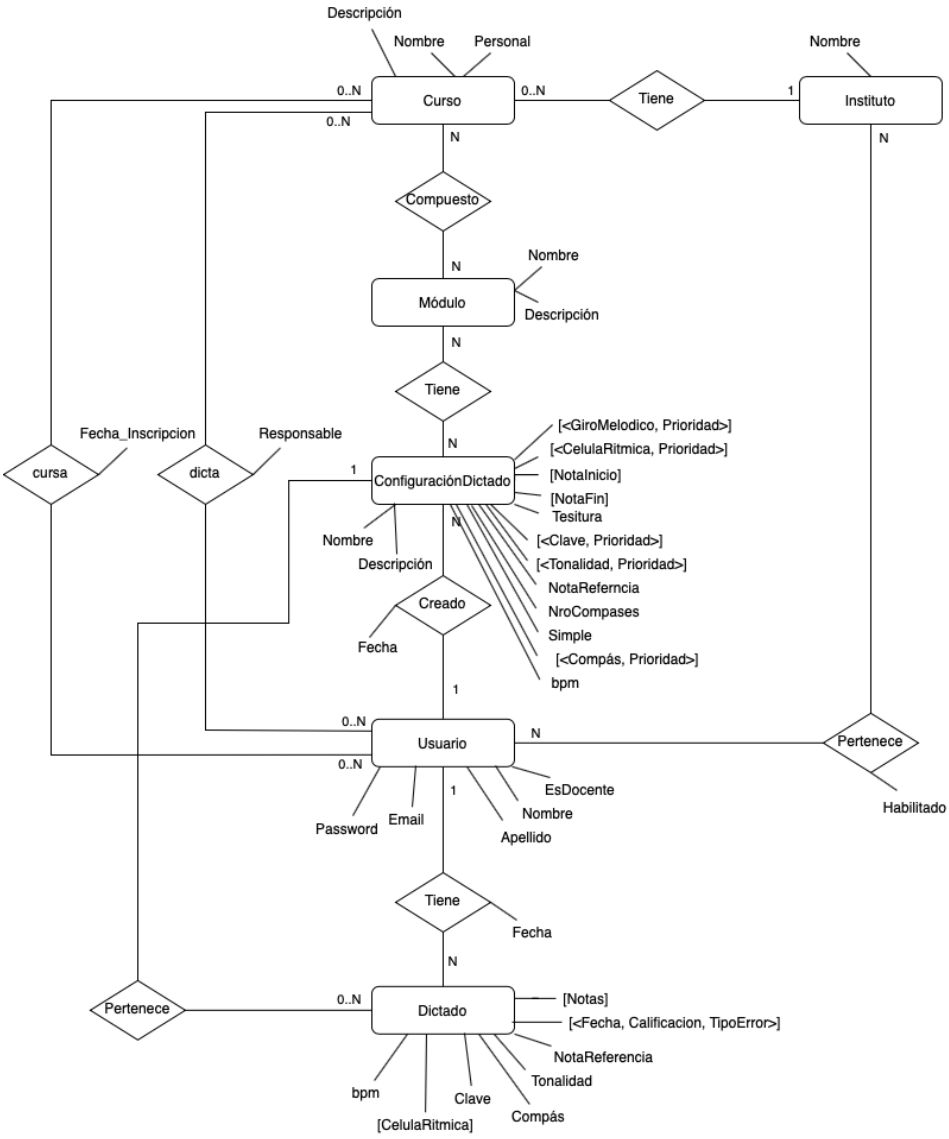

Figura 5.1: Modelo entidad relación de la aplicación ADA.

personal propio. Por otro lado los estudiantes tienen la posibilidad de dar de alta configuraciones de dictados siempre y cuando pertenezcan a algún módulo dentro del curso personal de dicho usuario.

Para profundizar los últimos conceptos mencionados, los cursos definidos como personal tienen la característica que sólo van a poder ser accedidos y modificados por su propietario. Esto resulta de utilidad para los estudiantes ya que ahí es donde podrán guardar sus configuraciones que den de alta para poder entrenarse ellos mismos. Por otro lado, un docentes tiene la posibilidad de crear configuraciones de dictado dentro de módulos pertenecientes a su curso personal y una vez esté conforme con el material configurado podrá copiarlos a un curso público, para que efectivamente los estudiantes, que estén inscriptos a dicho curso, tengan la opción de entrenarse a partir de éstas. Un punto importante que resulta de utilidad es que las configuraciones de dictado que pertenezcan a un curso personal tendrán la opción de ser modificadas. No es así la de los cursos públicos, ya que en base a las configuraciones de dictados que haya en dicho curso pueden haber estudiantes que hayan generado dictados, y si dicha configuración es modificada, los dictados quedarían incongruentes con la configuración asociada.

Con respecto a las configuraciones de dictados cabe aclarar que pueden ser creadas tanto de cero o a partir de una copia de una configuración ya existente (cada usuario tendrá la posibilidad de acceder a cualquier configuración dada de alta en el sistema y a partir de éstas crearse una propia).

Como se ilustra en el diagrama de la Figura 5.1, las configuraciones de dictados cuentan con un gran conjunto de atributos ya que es a partir de éstos que, cuando un estudiante acceda, se generarán dictados de forma aleatoria. A continuación se van a explicar brevemente cada una de éstas.

- `[<GiroMelodico, Prioridad>], [<CelulaRitmica, Prioridad>], [<Clave, Prioridad>], [<Tonalidad, Prioridad>], [<Compás, Prioridad>]`

  Todos éstos atributos consisten en un conjunto de tuplas, donde el primer componente de la tupla es un elemento musical concreto al cual le corresponde una cierta prioridad. Esto es, cada giro melódico que se establezca en la configuración de dictado se va a guardar con una prioridad seleccionada por el usuario, de esta forma, los que tengan asociados una prioridad mayor tendrán probabilidad de aparecer en el dictado ge-

nerado. De la misma forma funciona para las células rítmicas, claves, tonalidades y compáses.

- `[NotaInicio], [NotaFin]`
  Ambos elementos están formados por un conjunto de notas musicales, las cuales indicarán en las posibles notas que pueda iniciar y terminar el dictado respectivamente. Cabe destacar que cada una de las notas deben estar en al menos un giro melódico ya que de lo contrario no habría forma de partir o de finalizar en dicha nota.
- `Tesitura`
  Parámetro que indica la tesitura tanto para la clave de Sol como para la clave de Fa.
- `NotaReferencia`
  Esta va a ser la nota de referencia que el estudiante pueda escuchar previo a reproducir el dictado. Dicha nota es establecida en la configuración del dictado y luego, al generar el dictado, es transformada en función de la tonalidad en que se genere el dictado.
- `NroCompases`
  Este atributo establecerá el largo del dictado marcando el número de compases que contendrá.
- `Simple`
  Atributo que marcara si el dictado generado será simple o compuesto. Cabe aclarar que este parámetro determinará qué tipo de compás y células rítmicas se podrán establecer en la configuración del dictado ya que los compáses y células rítmicas son simples o compuestos.
- `bpm`
  El la duración en bpm que se le establece a la figura musical negra y en base a esta es que se determinará la duración del resto.

Cada dictado va a ser generado para un usuario estudiante, y estarán contenidos dentro de una configuración de dictado. Gran parte de los atributos de los dictados mostrados en el diagrama son utilizados por la aplicación para generar el contenido multimedia en tiempo real y poder ser reproducido. De esta forma en lugar de persistir el archivo de música, se almacenan los datos necesarios para generarlo. Con el conjunto de notas, células rítmicas, la clave, el compás y la tonalidad se generarán los dictados. A esto se le suma el parámetro de bpm el cual determinará la duración de cada una de las figuras

musicales que suene. Como dato extra para el usuario que desee entrenarse se tiene la nota de referencia, información que es accedida previa al dictado. Finalmente cada dictado tendrá una combinación de fecha, calificación y tipo de error, las cuales tomarán valor una vez que el usuario haya realizado al menos un intento de dicho ejercicio y los mismos corresponden a la fecha en que lo realizó, la calificación obtenida y los errores más frecuentes que tuvo (errores rítmicos o melódicos). Además se registrará la fecha en que cada usuario generó un dictado.

Por último se tiene una relación entre usuario e instituto, la cual estará presente solamente para usuarios docentes e indicará a qué institución educativa pertenece. Esta relación tendrá un campo que indicará si dicha relación está verificada o no, ya que no se debe tener la posibilidad que cualquier usuario ingresado en el sistema como docente se relacione con un instituto educativo sin estarlo realmente. Es por esto que existirá un usuario administrador el cual verificará la veracidad de dichas relaciones.

5.3. Etapas del proyecto

El desarrollo del proyecto tuvo una duración de ocho meses, en el cual se trabajó en diferentes etapas. Cada una de estas, a su vez, atravesó un conjunto de tareas con el objetivo de cerrar la etapa que la contiene. En el Anexo 1, correspondiente al presente capítulo se muestra un diagrama de Gantt, con el objetivo de exponer el tiempo dedicado a cada una de las etapas, desglosadas por tarea.

5.4. Arquitectura

El diseño de la arquitectura fue establecida al comienzo del proyecto, teniendo en cuenta el proyecto a abordarse junto con sus principales características. Esto consistió en definir cómo se organiza el sistema identificando los principales componentes estructurales y la forma de comunicación entre ellos.

5.4.1. Arquitectura orientada a servicios (SOA)

Un servicio es una representación estándar de cierto recurso, como ser un procesador especializado en cierto tipo de tarea o acceso a información almace-

nada, con el fin de que puedan ser utilizados por otros softwares. La principal característica es que proveer el servicio es independiente de la aplicación que lo consuma.

La arquitectura orientada a servicios se diseña con el fin de permitir reutilizar elementos gracias a las interfaces que ofrecen los servicios, los cuales se comunican a través de la red con un lenguaje en común. Esta forma de desarrollo permite tener sistemas distribuidos en la que cada uno de los componentes son servicios independientes que ejecutan en computadoras diferentes, cada una de estas teniendo recursos diferentes según la tarea que deban desempeñar (Sommerville, 2011). Esto es una característica que se busca tener en el proyecto dado que permitiría, en un futuro, adaptar la solución obtenida a diferentes plataformas, como puede ser un navegador web. Incluso facilitaría el hecho de poder migrar la aplicación hacia otras tecnologías en el caso de que las necesidades futuras lo requieran.

Alguna de las características de este tipo de arquitecturas que interesan tener presentes para el proyecto actual son las siguientes:

- Escalabilidad y flexibilidad: El permitir implementar diferentes servicios en diferentes lenguajes permite abarcar un mayor número de funcionalidades.
- Ágil: La posibilidad de reutilizar los servicios agiliza y simplifica el proceso de desarrollo.
- Mantenimiento sencillo: Gracias a la independencia entre los servicios, estos pueden ser modificados o incluso reemplazados, sin afectar al resto.

5.4.2. Arquitectura de tres niveles

La forma en que se implementa es en una arquitectura de tres niveles, donde cada uno de estos implementa diferentes servicios. Esta arquitectura organiza la aplicación en tres niveles: el nivel de presentación (cliente), el de procesamiento de datos (servidor de aplicaciones) y nivel de datos (sistema de gestión de base de datos) (Education, 2020).

Nivel de presentación

Este nivel corresponde a la interfaz de usuario y es donde el usuario interactúa con la aplicación. Es en este punto en donde se le muestra la información que el usuario solicita, además de recopilar información que este provee. Es-

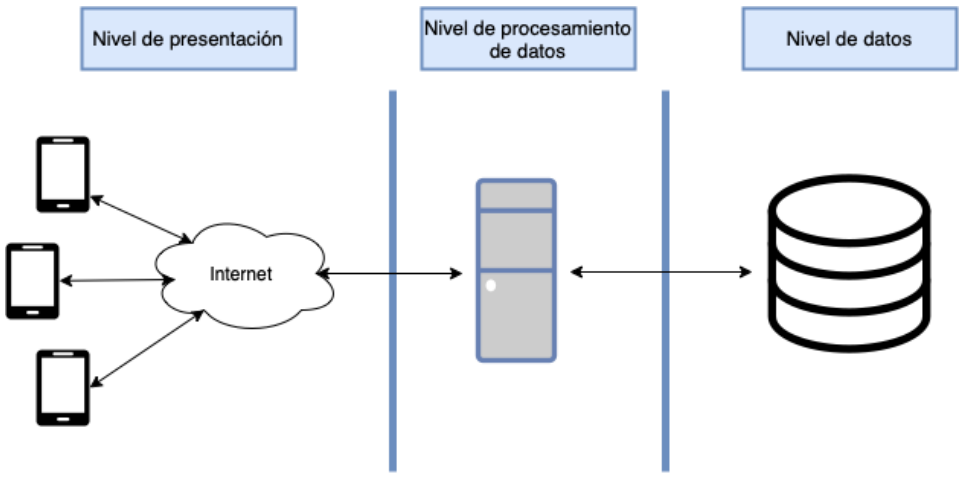

Figura 5.2: Niveles de la arquitectura

te nivel es ejecutado en los diferentes dispositivos móviles en donde corre la aplicación.

Nivel de procesamiento de datos

Este nivel también es conocido como nivel lógico o nivel medio, ya que es donde se procesa la información recopilada en el nivel superior para ser procesada, y en caso de corresponder, es enviada al nivel inferior. Es en esta capa en donde está presente la lógica de negocio.

Nivel de datos

Este nivel es el acceso a los datos, tanto lectura como escritura y es conocida como el nivel de base de datos.

La comunicación entre estos tres niveles está representada en el diagrama de la Figura 5.2

5.5. Tecnologías utilizadas

Para construir un sonftware de calidad, en este caso una aplicación para dispositivos móviles, se requiere tener una buena estructura establecida. Ésta la compone, por un lado, la arquitectura del sistema y por otro el stack tecnológico (o conjunto de tecnologías) con la que es implementada. Si bien, dentro del stack tecnológico pueden entrar un amplio espectro de herramientas utilizadas para el desarrollo, en esta sección se abordaran las tecnologías con las que se desarrolló la aplicación desde tres lugares bien distinguidos: *frontend*, *backend* y la base de datos.

5.5.1. Frontend

El frontend es la parte del software a la que un usuario puede acceder directamente, entrando dentro de esta categoría, todas las tecnologías que corren en el dispositivo móvil al momento de levantar la aplicación y que se encargan de la interactividad con el usuario (Chapaval, 2021).

Hay una gran variedad de tecnologías para elegir cuando se trata de desarrollar una nueva aplicación. Sin embargo, antes de elegir una, es necesario saber que el desarrollo móvil se divide en dos enfoques: desarrollo nativo y desarrollo multiplataforma. Cada uno de estos trae consigo una serie de ventajas y desventajas, las cuales serán analizadas, teniendo en cuenta el desarrollo del presente proyecto.

5.5.1.1. Desarrollo nativo

El enfoque del desarrollo nativo para crear aplicaciones móviles implica elegir una tecnología que sea nativa del dispositivo, más concretamente del sistema operativo que tenga. Los sistemas operativos a los que se apunta son iOS y Android debido a que son los dos sistemas operativos que dominan el mercado actual de dispositivos móviles.

Si se quiere desarrollar una aplicación nativa para dispositivos iOS, esto significa que tendrá que desarrollarse usando lenguajes como *Swift* u *Objective-C*. Si, por el contrario, se desea crear una aplicación nativa para dispositivos Android, se tendrá que implementar utilizando *Java* o *Kotlin*.

Esto trae una gran desventaja y es que si se quiere desplegar una aplicación para ambos sistemas operativos se debe realizar dos implementaciones diferentes, cada una de éstas en el lenguaje nativo correspondiente a cada sistema operativo. A esto se le suma que tanto el mantenimiento como el incorporar nuevas funcionalidades deberá hacerse a dos proyectos separados.

Sin embargo, este enfoque sigue siendo muy popular debido a que aporta una serie de ventajas:

Mejor presentación:

La primera y más notable ventaja de las aplicaciones nativas es que se comunican directamente con el sistema operativo de la plataforma, lo que significa que tienen acceso directo a sus APIs y otras funciones. Esto hace que este tipo de aplicaciones sean más rápidas y eficaces. También permite que las aplicaciones accedan directamente a ciertas funciones particulares de cada

dispositivo como el GPS o las cámaras de los teléfonos, lo que, una vez más, mejora el rendimiento.

Más rápido de adaptarse:

Las aplicaciones nativas, al estar desarrolladas en un leguaje propio del sistema operativo, se adaptan mucho más rápido a los cambios en sus respectivas plataformas. Si bien puede que luego de un tiempo una solución multiplataforma se ponga al día con una actualización del sistema operativo, los desarrolladores que trabajan con idiomas nativos pueden adaptarse a estos cambios mucho más rápidamente, haciendo que la corrección de errores o incorporación de funcionalidades que nacen de cambios en los sistemas operativos, sea casi inmediata.

Más intuitivas:

Las aplicaciones nativas utilizan los componentes de la interfaz de usuario de una plataforma directamente y, por definición, están diseñadas concretamente para dicha plataforma. Esto facilita el seguir las pautas de diseño de cada plataforma haciendo que la interfaz de usuario de las aplicaciones se vean como una extensión de su sistema operativo. Esto le ofrece al usuario una mejor experiencia, siendo incluso más intuitiva en ciertos aspectos.

5.5.1.2. Desarrollo multiplataforma

Durante algún tiempo, el desarrollo nativo fue prácticamente la única forma viable de hacer las cosas, y los equipos de desarrollo móvil se especializaron cada vez más en sus respectivos lenguajes, sin embargo, esto tuvo un alto costo. A medida que evolucionó la industria móvil, quedó claro que producir aplicaciones para una sola plataforma sería menos eficiente tanto en costos de producción como en tiempo.

Las soluciones multiplataforma permite codificar una sola aplicación y que esta se pueda ejecutar en múltiples plataformas. Esto significa escribir desarrollar y dar mantenimiento a un sólo código y que la aplicación funcione tanto para Android como para iOS. Esto da como resultado una serie de ventajas:

Rápido desarrollo

La principal ventaja del desarrollo multiplataforma consiste en reducir los tiempos de desarrollo, debido a que los desarrolladores del software deberán enfocarse en la construcción de un solo proyecto. Este punto resulta interesante tenerlo presente para el proyecto actual ya que permite cubrir un mayor número

de requerimientos en un tiempo establecido.

Fácil administración

El desarrollo nativo dirigido a múltiples plataformas se traduce en múltiples aplicaciones. A medida que estas aplicaciones comienzan a crecer, se agregan nuevas funcionalidades. Como resultado, poder mantener y administrar estas versiones diferentes se vuelve cada vez más difícil. Si no se dedica una cantidad decente de recursos a la administración de versiones, es probable que algunas aplicaciones se queden atrás, lo que generará usuarios descontentos. Con el desarrollo multiplataforma, por otro lado, solo hay una base de código en la que trabajar, por lo que la administración de versiones se vuelve mucho más fluida.

Ideal para MVPS

Las soluciones multiplataforma son un buen recurso si se desea comenzar por producir un MVP (mínimo producto viable). Esto consiste en una versión de un producto con la cantidad mínima de características, con las cuales se cubren los principales flujos de uso. El poder optimizar los tiempos de desarrollo permite construir un MVP de mejor calidad, permitiendo, además, tener tiempo disponible para realizar testing.

5.5.1.3. Selección de tecnología - Frontend

A partir del análisis realizado se opta por un desarrollo multiplataforma ya que, en primer lugar, el realizar una aplicación que funcione tanto en iOS como en Android es un requisito necesario. Por otro lado, al ser un proyecto, en el cual se tiene un tiempo acotado y un equipo de desarrollo reducido, resulta de interés poder optimizar los tiempos de desarrollo y así poder abarcar una amplia gama de funcionalidades. Además, dada la complejidad de algunas funcionalidades se establece como objetivo construir un MVP, teniendo en cuenta las funcionalidades básicas tanto del docente como del estudiante.

Dado que la aplicación desarrollada es de código abierto con el objetivo que pueda ser mantenida y logre evolucionar adaptándose a las necesidades de sus usuarios, se investigaron los frameworks más populares en la comunidad. Esto con el objetivo de que en un futuro, la tecnología en la que está desarrollada la aplicación no sea un impedimento a la hora de incorporar nuevas funcionalidades por personas interesadas. Esto llevó a considerar dos frameworks desarrollado por gigantes tecnológicos: por un lado *Flutter* desarrollado

por Google y *React-native* desarrollado por Facebook.

Flutter

Es un framework de código abierto utilizado para el desarrollo de aplicaciones para dispositivos móviles compiladas de forma nativa.

Las aplicaciones de Flutter se desarrollan con el lenguaje de programación Dart, siendo el motor escrito principalmente en C++. Esta tecnología permite interactuar con los SDK específicos de cada plataforma, como son los proporcionados por los sistemas operativos iOS y Android. Esto permite acceder a un conjunto de herramientas que dejan disponibles los desarrolladores de los sistemas operativos.

Como principal ventaja, tomando en consideración el presente proyecto, se destaca su rendimiento. Este es muy similar a las que ofrecen las aplicaciones desarrolladas en lenguaje nativo, siendo un factor de suma importancia para brindar una experiencia de uso de buena calidad. A esto se le suma que, al estar escrito en Dart ofrece compilación *Just-in-Time* (JIT) lo cual mejora el flujo del trabajo para el lado del desarrollador. Esto facilita la recarga activa, lo cual significa que todos los cambios realizados en el código se ven reflejados de forma automática en la interfaz, sin la necesidad que el desarrollador tenga que compilar nuevamente.

Como desventaja considerable se tiene que es un marco con poca antigüedad, pudiendo hacerlo inmaduro en aspectos como las bibliotecas disponibles. Esto trae como consecuencia que no exista una comunidad tan grande.

React Native

Es una biblioteca de JavaScript que, al igual que Flutter, es utilizada para el desarrollo de aplicaciones para sistemas iOS y Android. Los principios operativos son muy similares a los de ReactJs, librería de gran popularidad utilizada para el desarrollo web.

Tomando en cuenta las características del proyecto, una ventaja interesante es la arquitectura modular de esta biblioteca. Esto permite la reutilización de módulos, ayudado también a mantener una estructura limpia al momento que el proyecto crece. Por otro lado, React Native es una tecnología más madura que, si bien tiene solamente dos años más de antigüedad (fecha de lanzamiento 2015, contra 2017 en Flutter), tiene una comunidad mayor. Esto hace que exista una gran cantidad de librerías construidas por los desarrolladores, lo cual permite tener disponible una gran cantidad de recursos al momento de desarrollar una solución.

Las características mencionadas anteriormente (Simões, 2019, Skuza et al. 2021) son de las que mayor relevancia tienen al momento de abordar el proyecto actual, de lo cual se desprende que ambas opciones son buenas alternativas al momento de abordar un proyecto.

Un punto a considerar al momento de elegir una tecnología es la curva de aprendizaje, lo cual se tiene que React Native tiene una curva de aprendizaje menor que Flutter. A este punto se le suma que el equipo de desarrollo encargado de llevar adelante la construcción de la aplicación tiene experiencia previa en lenguajes como lo es ReactJs, siendo mucho más fácil la transición hacia React Native.

Esto desencadena en la elección de React Native como tecnología de desarrollo para el Frontend. Además se puede ver que, si bien Flutter tiene un desempeño mas cercano al que tienen las aplicaciones nativas, el rendimiento que tiene React Native es muy bueno.

5.5.2. Backend

El backend es la pieza del sistema que centraliza todas las peticiones realizadas por la aplicación proveniente de diferentes dispositivos móviles, permitiendo, además, el acceso a la base de datos.

Existen varios factores a tener en cuenta para la elección de la tecnología para el backend. Dadas las características del proyecto, se tomaron en cuenta factores como el desempeño, el tiempo de respuesta, la escalabilidad, la disponibilidad de librerías y la comunidad de las tecnologías. Estos atributos son los que se intenta cubrir en mayor medida dado que se busca construir una base lo suficientemente sólida para que la aplicación escale. Y es justamente por este motivo por el cual se tomará en consideración los lenguajes más populares y utilizados con mayor frecuencia en los software actuales, ya que muchos de estos soportan un numero gigante de usuarios, además de implementar funcionalidades complejas.

Existen tecnologías como PHP o Golang, las cuales, si bien pueden ser muy potentes, no cuentan con un gran número de librerías o frameworks. Esto es un punto crucial debido a que los nuevos requerimientos que se quieran incorporar al proyecto podrán exigir recurrir a recursos dispuestos por la comunidad. Por otro lado el framework .NET Core fue tenido en cuenta ya que presenta una gran comunidad además de destacarse por su buena performance (inVerita,

2020). A pesar de las buenas características, lo cual lo hubiese hecho un recurso interesante, esta tecnología fue descartada dada la curva de aprendizaje que presenta dicho framework en base a los conocimientos previos del equipo.

Finalmente, luego de analizar un gran abanico de posibilidades se decidió analizar en mayor profundidad las tecnologías NodeJs y Flask, basadas en los lenguajes JavaScript y Python respectivamente.

5.5.2.1. Flask

Flask[14] es un *micro-framework* desarrollado con el objetivo de simplificar la creación de APIs bajo el patrón MVC. El término *micro-framework* hace referencia a que al iniciar un proyecto con esta herramienta se instalan las dependencias básicas y necesarias para la construcción de una API funcional. Esto lo hace una buena opción para el desarrollo de prototipos y proyectos ágiles. Al ser un *micro-framework* lo hace tener una buena velocidad y desempeño, debido a su sencillez.

Por otro lado, una vez el proyecto en Flask comienza a requerir nuevas funcionalidades, estas pueden ser adquiridas mediante el uso de plugins. Esto permite abarcar un amplio abanico de funcionalidades, permitiendo mantener el proyecto ligero y fácil de mantener (Muñoz, 2017).

5.5.2.2. NodeJs

NodeJs[15] es un entorno en tiempo de ejecución multiplataforma que ejecuta en la capa del servidor. Esta herramienta ejecuta en el motor JavaScript V8 desarrollado por Google, destacándose principalmente por su alta velocidad y rendimiento de ejecución.

Esta herramienta está diseñada para trabajar en base a eventos de entrada y salida asíncronas. Esto implica una gran ventaja al momento de evaluar la escalabilidad de la aplicación, ya que el uso de hilos permite que diferentes procesos en ejecución no se bloqueen entre sí.

Como última característica que interesa analizar de esta tecnología es su procesamiento en un solo hilo con bucle de eventos. Este mecanismo de eventos ayuda al servidor a responder de forma óptima y hace que sea altamente escalable en comparación con otros servidores tradicionales que crean hilos

[14]https://flask.palletsprojects.com/en/2.0.x/ documentación oficial de Flask.
[15]https://nodejs.org/en/about/ documentación oficial de NodeJs.

limitados para poder manejar las diferentes solicitudes que le llegan (Khare, 2021).

5.5.2.3. Selección de tecnología - Backend

Como primer punto a destacar se tiene la arquitectura de ambas tecnologías. En este aspecto, NodeJs se destaca por su manejo de operaciones asíncronas, es decir, cuando se es necesario leer o escribir datos, ya sea en la red, base de datos o sistema de archivos, en lugar de bloquear el hilo de ejecución, se reanudarán las operaciones cuando la operación termine, haciendo que no se desperdicien ciclos de CPU esperando. Con respecto a este punto, Python no está diseñado para soportar operaciones asíncronas y manejo de eventos de forma nativa. Si bien el framework de Flask permite que se le instalen extensiones para soportar este tipo de operaciones, requiere un esfuerzo extra (Romanyuk, 2020). Este análisis permite ver que NodeJs tiene un mejor desempeño cuando se trata de manejar un gran número de conexiones concurrentes, como puede llegar a suceder al momento que la aplicación es utilizada en varios dispositivos móviles de forma simultánea. Además, al manejar eventos no bloqueantes permite tener un tiempo de respuesta mejor ya que la aplicación estará haciendo lecturas y escrituras tanto en la base de datos como en sistemas de archivos al momento de generar el contenido multimedia.

En cuanto a la comunidad, tanto Python como NodeJs tienen una gran comunidad muy activa lo que implica que se pueda acceder a una gran cantidad de recursos. En este punto cabe destacar que, si bien Python tiene una gran comunidad detrás, el framework de Flas la tiene en menor medida lo que podría dificultar acceder a foros de discusión para ciertos temas concretos.

Finalmente, teniendo en cuenta los perfiles de las personas involucradas en el proyecto, si bien ambos lenguajes resultan conocidos, se tiene una mayor desenvoltura en el lenguaje de JavaScript, siendo, además, el lenguaje utilizado en el frontend. Esto hace tomar la decisión de desarrollar la API en NodeJs.

Cabe aclarar que, si bien se eligió desarrollar el backend en NodeJs, Python cuenta con características de gran valor ya que es un lenguaje orientado a ciertas áreas de la inteligencia artificial y análisis de datos (Khare, 2021). Si bien no se tienen requerimientos con estas características, podría resultar interesante tener dicho recurso disponible por futuras funcionalidades que se quieran incorporar. Esta cuestión está contemplado desde el lado de la arquitectura

orientada a servicios, ya que, si bien NodeJs es utilizado para las funcionalidades principales, en un futuro puede desarrollarse un servicio particular implementado en Pyhton (o cualquier otro lenguaje) y que sea consumido por la aplicación.

5.5.3. Base de datos

Una base de datos no solo permite agrupar y almacenar los datos recaudados por la aplicación, sino también facilita el compartir los datos entre diferentes usuarios. La correcta elección de la base de datos hará posible, no solo gestionar esa información de forma adecuada, sino también hacerlo de una forma óptima.

Es por esto que la primera decisión consiste en elegir entre el paradigma de base de datos relacional o no relacional. Esto se hace teniendo en cuenta las características del proyecto y los tipos de operaciones que más demanda la aplicación.

5.5.3.1. Base de datos relacional

Las base de datos relacionales consiste en una colección de elementos de datos los cuales contienen relaciones entre ellos. Estos elementos son representados como tablas, es decir, cada fila representa un conjunto de valores relacionados a un objeto o entidad de la realidad modelada, siendo cada columna un determinado tipo de dato. Estos datos son administrados mediante un lenguaje de consulta estructurado (SQL[16]). Si bien existen un conjunto de extensiones de dicho lenguaje, como ser *T-SQL* ofrecido por Microsoft o *PL/SQL* de Oracle, la base de éstas radica en SQL.

Toda la información en una base de datos relacional está organizada en partes más chicas, cada una de éstas pudiendo tener identificadores únicos. Esto permite establecer relaciones entre los datos y de esta forma tener una forma sencilla de acceder a ellos de una forma estructurada.

5.5.3.2. Base de datos no relacional

Las bases de datos no relaciones consisten en un sistema de almacenamiento de datos, caracterizado por no usar lenguaje SQL. Además, las bases de datos

[16] *Structured Query Language* por sus siglas en inglés.

no relacionales (o NoSQL) estás diseñadas para soportar esquemas flexibles, lo cual se obtiene mediante la flexibilización de restricciones de coherencia, algo que no está presente en las bases de datos relacionales. Esto se debe a una de las principales diferencias con una base de datos relaciones, y es que los datos no son almacenados en tablas, sino que incluyen diferentes tipos de almacenes, como ser almacén de columnas, de documentos, de clave valor, de gráficos, entre otros.

Las bases de datos no relacionales surgieron por las crecientes necesidades de muchas organizaciones que no son cubiertas por una base de datos relacional, o que, si bien son cubiertas, lo hacen a un alto costo en cuanto a hardware y tiempo. Alguna de estas necesidades son:

- Manejar diferentes estructuras de datos sin incrementar altamente la complejidad
- Escalabilidad para poder soportar una gran cantidad de datos y consultas de forma concurrente
- Manejar de forma óptima grandes volúmenes de datos provenientes de otras fuentes de información
- Brindar un tiempo de respuesta adecuado ante el aumento de consultas que reciba el sistema
- Desde el punto de vista de la gestión permite adecuarse al desarrollo ágil con entregas incrementales, gracias a la flexibilidad al modelar las estructuras

Estas necesidades son cubiertas por una base de datos NoSQL (Mesa, 2018), en particular se enumeran las que resulta interesante tener presente para el proyecto.

5.5.3.3. ¿SQL o NOSQL?

Se considera que tanto las bases de datos SQL como las NoSQL son adecuadas, ya que, de una forma u otra, se pueden obtener resultados aceptables. Lo que interesa es poder optimizar los puntos fuertes del paradigma elegido en base a la realidad del presente proyecto. Es por esto que en esta sección se abordarán las ventajas y desventajas.

En primer lugar, debido al largo tiempo que llevan en el mercado las bases de datos relacionales, existen una gran variedad de herramientas con mayor

soporte para gestionarlas. Esto también implica que haya una mayor cantidad de profesionales con conocimientos más profundos en este paradigma, lo que facilitaría, en un futuro, incorporar nuevos miembros al equipo de desarrollo o incluso, al ser una aplicación de código abierto, que personas externas desarrollen una versión propia a partir del resultado obtenido.

La atomicidad de las operaciones en las bases de datos relacionales es un punto fuerte, ya que esto habilita el poder revertir operaciones utilizando funciones de *rollback*. Si bien es un punto interesante, la mayor lógica de la aplicación se encuentra en el lado de la API y no en consultas a la base de datos. Es decir, no existen operaciones complejas en la base de datos la cual sea necesario revertir. Si bien esto se presenta como un punto fuerte, debido a la atomicidad de las operaciones se ve afectado el rendimiento de forma negativa.

Finalmente, las bases de datos relacionales, al ser un lenguaje de consulta estructurado, exige tener una estructura ya establecida. Esto resulta adecuado en proyectos en donde se tiene conocimiento en cuanto a los requerimientos del software a desarrollarse, ya que conocer la estructura de los datos almacenados resulta de utilidad al momento del desarrollo. El presente trabajo consiste en un proyecto formado por un equipo interdisciplinario en el cual se estableció como requisito que sea adaptable a cambios. Incluso, al requerirse que la aplicación, luego de liberada, cuente con una base necesaria para que se adapte a las necesidades que surjan, es posible que el modelado cambie, tanto en el transcurso del proyecto como a futuro.

Abordando ahora características de las bases de datos NoSQL, se tiene que una de las principales ventajas es que permite una alta escalabilidad, en concreto, se destacan por su escalabilidad horizontal. En este punto cabe aclarar que la escalabilidad vertical consiste en ampliar los recursos de hardware en que se almacena la base de datos y así tener un mejor rendimiento, característica que tienen ambos paradigmas de bases de datos. Por otro lado se tiene la escalabilidad horizontal, la cual consiste en tener diversos servidores, conocidos como nodos, trabajando como un todo. De esta forma se divide la demanda de trabajo entre los nodos que conforman la red se servidores. Este tipo de escalabilidad, técnicamente puede tener un crecimiento ilimitado, ya que consiste agregar nodos a una red, diferente a la escalabilidad vertical, la cual está limitada por el hardware del momento. Cabe destacar que, si bien las bases de datos NoSQL se caracterizan por su escalabilidad horizontal, también permiten escalar verticalmente aumentando los recursos de hardware.

Como un punto deseable se tiene que las bases de datos NoSQL permiten realizar ciertos cambios, como pueden ser cambios de esquemas, sin tener que parar la base de datos. Esto resulta de utilidad dado el requerimiento de poder contemplado nuevas funcionalidades una vez esté en uso la aplicación.

Existen ciertas desventajas en las bases de datos relacionales, la más destacable es la falta de estandarización. Esto significa que no existe un criterio definido entre los motores que utilizan este tipo de base de datos, el cual varía según el tipo de base de datos utilizada. A esto se le suma que muchas bases de datos NoSQL no suelen tener una interfaz gráfica tan intuitiva, o incluso deben ser manejadas por consola. Si bien esto puede limitar, en un futuro, la incorporación de nuevos colaboradores a la aplicación, para futuras funcionalidades que surjan, se considera que para el desarrollo del proyecto el equipo cuenta con los conocimientos básicos para poder cubrir esta desventaja.

De estas características mencionadas (PandoraFMS, 2015) se destaca la escalabilidad y la flexibilidad como principales atributos a optimizarse para el proyecto. Existe un último punto deseable a optimizar: el rendimiento. Cabe destacar que el sistema cuenta un un alto grado de operaciones de lectura a la base de datos, teniendo un moderado o bajo uso de operaciones de inserción o modificación. En cuanto a este punto, las bases de datos NoSQL se destacan por tener un mejor rendimiento en operaciones de lectura para grande cantidades de datos. Existe una variedad de artículos de investigación (Choi, 2014, Pandey, 2020, Hammood y Saran, 2016) los cuales comparan el rendimiento de diferentes bases de datos relacionales y no relacionales con *Yahoo Cloud Serving Benchmark*[17]. Esta es una herramienta de código abierto que permite evaluar el rendimiento de sistemas, particularmente de diferentes bases de datos. Esto lo hace definiendo diferentes cargas de trabajo enfocada en operaciones CRUD (operaciones de inserción, lectura, actualización y eliminación), es decir, simula una cantidad elevada de operaciones de forma simultánea y se analiza el tiempo de respuesta. En general se observa un mejor rendimiento de las bases de datos NoSQL en lo que corresponde a flujos de trabajo en donde se ejecutan mayormente operaciones de lectura. Si bien los resultados que presenta dicha herramienta varían dependiendo del ambiente en que se ejecute y que bases de datos en concreto se pongan a prueba, se entiende que dichos resultados son representativos y extrapolables al contexto del proyecto.

El conjunto de características mencionadas hace que se opte por una base

[17]https://github.com/brianfrankcooper/YCSB acceso al repositorio de la herramienta.

de datos no relacional ya que se entiende que podría optimizar ciertos puntos de interés para el proyecto.

5.5.3.4. Elección de BDNR

La elección en cuanto a la base de datos no relacional se baso en el artículo *Choosing the right NoSQL database for the job: a quality attribute evaluation* (Laurenço, 2015), presentado como material de estudio complementario en la materia Bases de Datos No Relacionales Edición 2021. En dicho material se estudian las siguientes bases de datos no relacionales: *Aerospike, Cassandra, Couchbase, CouchDB, HBase, MongoDB y Voldemort*. Si bien se sabe que existen otras posibilidades, se entiende que sí abarca las más populares y de mayor éxito, por lo que se considera un universo de opciones aceptables a tener en cuenta.

El artículo mencionado aborda diferentes atributos de calidad, de los cuales se extrajo los que más importancia tenían dado el contexto del proyecto y así seleccionar la base de datos no relacional que más los optimizara.

Previo al análisis de dichos atributos cabe aclarar que, dada la realidad planteada en el proyecto, las operaciones de consultas será algo a optimizar. Esto es así ya que las funcionalidades principales de la aplicación (generación y reproducción de dictados) van a requerir estar haciendo lecturas constantemente a la base de datos para leer los datos que forman los dictados y así poder generar dicho contenido multimedia y reproducirlo. En cuanto a la escritura, si bien son operaciones presentes en la aplicación y es necesario su correcto funcionamiento, no es un atributo a optimizar ya que serán las menos frecuentes.

Aclarado el punto anterior, la performance como atributo de calidad es de las principales características en las que se hizo foco. Dentro de éste se encuentra el desempeño para la escritura y para la lectura, siendo, por lo anteriormente mencionado, la performance en la lectura el principal atributo a optimizar.

Por otro lado esta la fiabilidad, esto trata sobre la probabilidad de que el sistema opere sin fallas durante un periodo de tiempo dado. Esto resulta un aspecto importante ya que se debe contemplar el crecimiento de la aplicación y que pueda soportar que muchos usuarios la utilicen simultáneamente.

La escalabilidad trata sobre la habilidad de un sistema de manejar una

carga de trabajo mayor. Es decir, la capacidad que tiene la base de datos de soportar las consultas (ya sea lectura o escritura) cuando el negocio demanda más recursos. Las Bases de Datos No Relacionales, si bien han sido creadas con el objetivo de mejorar este aspecto, es un atributo que, a priori, interesa mejorar con la elección de la base de datos.

Existe además los atributos de disponibilidad y consistencias, dos elementos opuestos ya que si se aumenta el grado de disponibilidad disminuye la garantía de consistencia. La disponibilidad consiste en que un nodo en funcionamiento debe retornar siempre una respuesta razonable en un período de tiempo razonable. Por otro lado la consistencia garantiza que la lectura de datos retornará la escritura más reciente, para un registro dado. Eso quiere decir que siempre que se haga alguna modificación en algún dato, dicho cambio debe reflejarse en todos los nodos de la base de datos, garantizando que siempre se accede a la información correcta desde cualquiera de dichos nodos. Si bien ambos elementos serían deseables en la aplicación, se considera que con una disponibilidad razonable, la aplicación no se vería afectada en términos de usabilidad (el software no depende de que la base de datos le brinde ciertos datos en un tiempo determinado, como sí lo sería una aplicación de streaming o similares). Por otro lado sí sería interesante optimizar la consistencia ya que, si bien las configuraciones de dictados una vez hayan sido dadas de alta no serán modificadas, sí existe el escenario en donde se creen nuevas en el transcurso del curso. Además, considerando posibles funcionalidades extras, si la aplicación tomase un rumbo en el cual contemplase la toma de evaluaciones auditivas, sería un aspecto importante que la habilitación de cierta configuración de dictado sea accedida por todos los usuarios estudiantes, por lo que es necesario que se vea reflejado en todos los nodos de la base de datos.

Los atributos de calidad mencionados son algunos de los estudiados en el paper de Laurenço, 2015. El resto de los atributos allí abordados, si bien su optimización podrían mejorar ciertos aspectos de la aplicación, no son factores decisivos. En la figura 5.3 se presentan los resultados obtenidos en dicho estudio:

En base a los atributos que se quieren optimizar se puede ver que bases de datos como Aerospike, Cassandra, Couchbase y MongoDB podrían resultar interesantes. Se puede ver que tanto Aerospike como Coushbase presentan un punto débil en el atributo correspondiente a la fiabilidad. Esto podría ser en cierto grado preocupante ya que si se desea contemplar un crecimiento en el

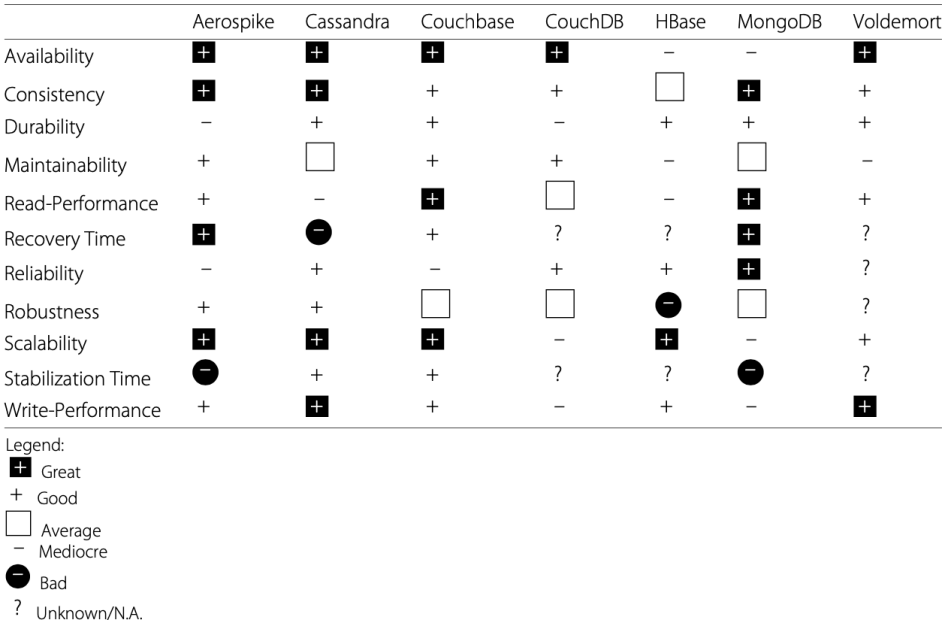

Figura 5.3: Resultados de los estudios. Imagen obtenida del artículo *Choosing the right NoSQL database for the job: a quality attribute evaluation* (Laurenço, 2015).

número de usuarios de la aplicación, con una alta demanda de consultas a la base de datos por largos períodos de tiempo sería más probable que el software fallase.

Por otro lado, en cuanto a la escalabilidad, MongoDB presenta una deficiencia en este atributo. Si bien esto es de suma importancia, las bases de datos NoSQL se han desarrollado para escenarios donde la escalabilidad es un factor muy importante. Esto se contempla agregando otros servidores, conocidos como nodos, los cuales trabajan de forma conjunta como un todo. Por lo tanto se considera que este aspecto podría ser sencillamente solucionado ampliando el número de nodos en caso de ser requerido.

Finalmente Cassandra presenta un desempeño inferior en las operaciones de lectura, que las bases de datos anteriormente mencionadas, factor considerablemente importante dada la lógica y tipo de operaciones más demandadas de la aplicación. Si bien las bases de datos no relacionales tienen un mejor desempeño que las relacionales en este aspecto se considera que este podría ser un factor decisivo en la elección de la base de datos.

Para poder saber si realmente las operaciones de lectura difieren mucho en cuanto a desempeño entre una base de datos y otra es que se recurrió a un segundo estudio el cual aborda este detalle en mayor profundidad. En el

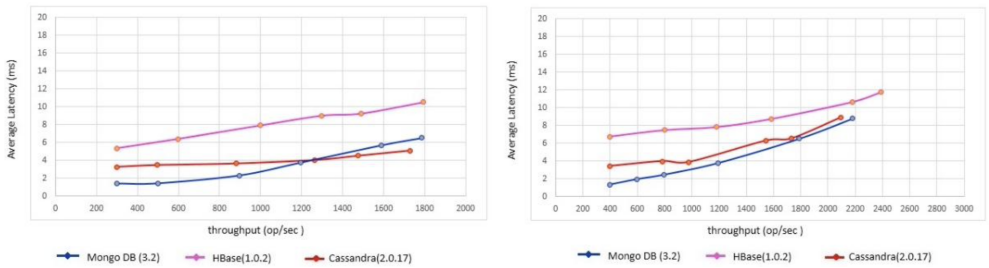

Figura 5.4: Pruebas de rendimiento. Imágenes obtenidas del artículo *A Comparison Of NoSQL Database Systems: A Study On MongoDB, Apache Hbase, And Apache Cassandra* (Hammood y Saran, 2016).

paper "A Comparison Of NoSQL Database Systems: A Study On MongoDB, Apache Hbase, And Apache Cassandra" (Hammood y Saran, 2016), interesa abordar los estudios que los autores realizaron sobre las bases de datos no relacionales, concretamente analizando las operaciones de lectura en MongoDB y Cassandra. Las pruebas fueron realizados con el software de Yahoo Cloud Serving Benchmark (YCSB) que, si bien fueron ejecutadas en un ambiente de test específico (con un cierto hardware el cual podría no ser el mismo que se tiene en el servidor donde se encuentra la aplicación), se considera que los resultados obtenidos son extrapolables a otros ambientes.

Se hará foco en las pruebas donde la herramienta simula cargas de trabajo grandes, en operaciones enfocadas a la lectura. Particularmente se analizarán los siguientes dos resultados obtenidos: Operaciones 100 % de lectura y Operaciones mayormente de lectura, imagen de la izquierda y derecha respectivamente, haciendo foco en las bases de datos de MongoDB y Cassandra.

Se puede ver (figura 5.4) que los resultados en las gráficas (Hammood y Saran, 2016) muestra la latency en función del throughput, es decir, el tiempo (en milisegundos) que demora en devolver los datos consultados en función de la cantidad de consultas (operaciones por segundo).

Se puede observar que con operaciones solamente de lectura, MongoDB tiene un mejor desempeño en un throughput moderado, pero cuando este crece Cassandra tiene una mejor latencia. En la segunda imagen muestra los resultados a partir de un 95 % de operaciones de lectura y el resto operaciones de actualización. En este escenario se ve un mejor desempeño de MongoDB en su latencia tanto en un throughput moderado y alto. Esto se debe a que mongo soporta el almacenamiento en memoria mapeada en caché (Hammood y Saran, 2016).

El software desarrollado es más similar a la segunda situación analizada, ya que varias funcionalidades de la aplicación implican operaciones de inserción y actualización, siendo estas últimas las menos frecuentes. Este resultado inclina la balanza hacia la base de datos MongoDB. A esto se le suma también que los integrantes encargados del desarrollo de la aplicación se encuentran familiarizados con dicha tecnología. Esto es un motivo no menor, ya que la curva de aprendizaje para poder utilizar MongoDB será mucho menor que para la utilización de Cassandra u otra base de datos no relacional. Finalmente con lo presentado en la presente sección se decidió utilizar MongoDB como base de datos de la aplicación.

5.5.4. Herramientas relacionadas con la música

En la presente sección se abordarán las herramientas y tecnologías utilizadas concretamente con los temas relacionados a la música, esto es, generación archivos multimedia que contengan los dictados, graficado en los pentagramas y algunas configuraciones establecidas.

En primer lugar, para la generación de los dictados se utiliza *MidiWriterJS*, librería de *JavaScript* que proporciona una API para la generación de archivos MIDI. Esta sigla corresponden a la abreviatura en inglés de *Musical Instrument Digital Interface* y los archivos con dicha extensión almacenan datos que contienen una serie de instrucciones que se puede traducir a sonido. Estas instrucciones son mensajes que indican a un cierto instrumento emulado cuales son las notas musicales, su duración, la fuerza de toque y las modulaciones de los parámetros de los sonidos, información necesaria para la generación de sonidos[18].

Dicha librería tiene una configuración muy amplia para así poder controlar la configuración de los dictados rítmicos y melódicos generados. Claro esta que de los principales parámetros a configurar se encuentran las figuras musicales y las notas asociadas a cada una de estas. Relacionado a esto se establece también la duración en bpm para cada una de las figuras. Cabe recordar que en la configuración de dictados que se da de alta en la aplicación, se define un rango de bpm (para la figura negra) y en base a dicha duración es que se le asigna un cierto valor en bpm al resto de las figuras del dictado. Sea `bpm_base`

[18]Información obtenida del sitio oficial de *MIDI Manufacturers Association* de California https://www.midi.org/

el valor en bpm para la nota negra, el valor que se le asigna a cada una de las figuras se ilustra en la siguiente tabla:

Nota	**Valor en bpm**
Redonda	`bpm_base*4`
Blanca	`bpm_base*2`
Blanca con puntillo	`bpm_base*2 + bpm_base`
Blanca con doble puntillo	`bpm_base*2 + bpm_base + bpm_base/2`
Negra	`bpm_base`
Negra con puntillo	`bpm_base + bpm_base/2`
Negra con doble puntillo	`bpm_base + bpm_base/2 + bpm_base/4`
Corchea	`bpm_base/2`
Corchea con puntillo	`bpm_base/2 + bpm_base/4`
Corchea con doble puntillo	`bpm_base/2 + bpm_base/4 + bpm_base/8`
Semicorchea	`bpm_base/4`
Fusa	`bpm_base/8`
Semifusa	`bpm_base/16`

Finalmente como último parámetro configurado se encuentra el instrumento en el que se van a reproducir la melodía. Junto con docentes del instituto de música se llego a la decisión de tomar como instrumento por defecto el piano de cola acústico (el cual esta incluido como instrumento en la librería de *Midi Writer js*) ya que el mismo no agrega grandes complejidades al dictado. Para futuras versiones de la aplicación se deja contemplado el poder contemplar diferentes instrumentos musicales en caso de que resulte de valor para el entrenamiento auditivo. Además de los parámetros mencionados, existen otros parámetros como son la velocidad de reproducción, el volumen del sonido o incluso el poder agregar una demora al comienzo de cada nota, los cuales se dejaron fijos, ya que no es algo que interese dejarlo parametrizable.

La librería *Midi Writer js* es utilizada del lado del *backend*, más concretamente es integrada en *node js*, tecnología basada en el lenguaje de *javascript*, lo que permite la integración con dicha librería. Como se mencionó anteriormente, esta herramienta genera archivos MIDI, los mismos son transformados a formato MP3 y a partir de estos se genera una URI que es enviada a la aplicación para que sea reproducida. Para generar los archivos MP3 a partir de archivos MIDI se utilizan dos paquetes instalados a nivel del servidor: *TiMidity++* y *FFmpeg*. El primero se utiliza para generar audio digital a partir

de archivos MIDI para ser procesados en tiempo real[19]. Una vez es procesado se utiliza el paquete *FFmpeg*[20] el cual permite manejar archivos multimedia para grabar, convertir a otros formatos y hacer streaming ya sea audio o video. Ambos paquetes funcionan en conjunto de la siguiente manera: *TiMidity++* genera audio de forma digital (sin llegar a reproducirlo realmente) a partir del dictado que fue generado en un archivo MIDI. Posteriormente se captura dicha salida de audio digital con el paquete de *FFmpeg* el cual lo procesa y lo guarda en un archivo MP3. Finalmente con el módulo de *File System*[21] de *node js* permite leer dicho archivo multimedia y enviarlo a la aplicación en formato URI para ser reproducido. Cabe aclarar que dichos archivos generados en el servidor son eliminados una vez son utilizados ya que de lo contrario llegaría a ocupar mucho espacio de disco en donde se encuentra alojada la API. Además los archivos son generados con el identificador de cada usuario para que de esta manera no haya conflicto si dos usuarios desean reproducir un archivo al mismo tiempo.

Una vez el dictado es reproducido, el siguiente paso será que el estudiante pueda acceder a la solución graficada en un pentagrama, para lo cual se introduce una nueva serie de herramientas. *VexFlow*[22] es una API *open-source* la cual permite, mediante el ingreso de ciertos parámetros musicales, el renderizado de un pentagrama. Dicha API contiene un "pequeño lenguaje"llamado *EasyScore*, esto es una forma sencilla de escribir elementos musicales que luego sean traducidas a los elementos que *VexFlow* necesita para escribir en un pentagrama. Cabe aclarar que dicha herramienta genera el pentagrama para ser renderizado en un navegador web, por tal motivo es que se utiliza *WebView*[23] para poder desplegarlo en la aplicación móvil.

5.6. Implementación

Esta sección abarca los algoritmos desarrollados y la forma en que éstos resuelven los problemas presentes en las funcionalidades principales de la apli-

[19]http://timidity.sourceforge.net sitio oficial de *TiMidity++*.

[20]https://www.ffmpeg.org sitio oficial de *FFmpeg*.

[21]Módulo que permite la interacción con los archivos del directorio. Documentación obtenida de https://nodejs.org/api/fs.html.

[22]https://www.vexflow.com sitio web de la herramienta en donde ejemplifica su uso además de poder acceder a su repositorio en *github*.

[23]https://github.com/react-native-webview/react-native-webview, repositorio del paquete donde se encuentra disponible para plataformas, tanto para móvil como para escritorio.

cación. Dado que los problemas a solucionar tienen su complejidad, la implementación fue una de las etapas a las que más tiempo se le dedicó, no solo para lograr un correcto manejo de los elementos musicales, sino para lograr una ejecución óptima dado los recursos disponibles. Particularmente la generación de los dictados aleatorios fue de las funcionalidades que más análisis acarreó debido a, no solo la dificultad natural de generar elementos aleatorios, sino también la de manejar y representar elementos pertenecientes a otra disciplina, en este caso la música.

Toda la generación de dictados se hace del lado del servidor ya que este podría contar con mayores recursos y así obtener mejores resultados de rendimiento, además de que facilita el acceso a la base de datos para almacenar los resultados. En primer lugar, al igual que establecer una configuración de dictados en la aplicación, la generación aleatoria se divide en dos etapas: la generación de elementos rítmicos y la generación de elementos melódicos. En anexos, en la sección correspondiente al capítulo de desarrollo puede verse un pseudocódigo el cual permite profundizar aún más en cualquier detalle que se aborde en las siguientes secciones.

5.6.1. Generación de elementos rítmicos

La generación de los elementos rítmicos es el primer paso en la generación de dictados ya que en base a los resultados de esta etapa es que se generan los elementos melódicos.

Como entrada de esta primera etapa se tienen las células rítmicas, los compáses, el número de compases y si el dictado será simple o compuesto, teniendo, los dos primeros elementos, una probabilidad asociada. Sabiendo estos datos, el primer paso consiste en obtener un conjunto de células rítmicas que correspondan a un compás, y para que estas sean válidas deben tenerse en cuenta los valores de las figuras y contrastarlos con el valor de cada pulso. En este punto cabe mencionar que a partir del compás se tiene que el numerador indica el número de pulsos dentro de un compás y el denominador el valor de cada pulso.

Sabiendo esto es que se irán tomando células rítmicas de forma aleatoria (teniendo en cuenta la probabilidad asociada a cada una) y a partir de este se calcula lo siguiente: para el conjunto de células rítmicas obtenidas hasta el momento, se suma el valor de cada figura perteneciente a cada célula rítmica

y se multiplica por el denominador. Esto dará como resultado un número que tiene la siguiente interpretación: si es menor que el valor del numerador es que se debe agregar otra célula rítmica de forma aleatoria al conjunto y se vuelve a llamar a la función de forma recursiva para volver a hacer dicho cálculo; si es igual al valor del numerador quiere decir que dicho conjunto de células rítmicas corresponde a un compás lo cual hace que el conjunto generado hasta el momento forme parte del dictado final; si el número es mayor al numerador quiere decir que los valores de dicho conjunto son más grandes que el valor que tiene que tener un compás por lo que lo generado es inválido. En este último caso se procede a quitar la última célula rítmica obtenida y volver a llamar a la función de forma recursiva y así obtener de forma aleatoria otra célula rítmica diferente, repitiendo este paso hasta obtener un conjunto válido. Cabe destacar que este procedimiento es posible ya que se estipuló que el sistema tendrá células rítmicas que siempre completarán de forma exacta uno o más pulsos y nunca una fracción de los mismos. A continuación se presenta el algoritmo:

```
FUNCIÓN  GenerarDictadoRitmico
    MIENTRAS falten compases en el dictado:
        // A partir del conjunto de células rítmicas, se
        // seleccionan de forma aleatoria (teniendo en
        // cuenta sus prioridades) hasta completar un compás
        FigurasCompas = ObtenerFigurasParaUnCompas
        Dictado.INSERTAR(FigurasCompas)
    FIN MIENTRAS
FIN FUNCIÓN
```

El procedimiento explicado anteriormente dará como resultado un conjunto de células rítmicas correspondientes a un compás, por lo que dichos pasos se repiten de forma iterativa hasta completar la cantidad de compáses indicada. De esta forma se obtiene un conjunto de células rítmicas, lo que se traduce a un conjunto de figuras musicales a partir de lo cual da inicio a la siguiente etapa en la generación de dictados.

5.6.2. Generación de elementos melódicos

Esta segunda etapa consiste en asignarle una nota a cada figura obtenida en el paso previo, siguiendo las reglas de la configuración del dictado para que el resultado sea coherente. Los datos que son utilizados son los siguiente: giros melódicos, la tesitura (la cual no es establecida por el usuario, sino que es un parámetro establecido en el sistema), notas de inicio, notas de fin, claves, tonalidades, nota de referencia y cantidad de figuras, siendo esta última un dato obtenido de la generación de elementos rítmicos.

Como pasos iniciales se obtienen, de forma totalmente aleatoria, las notas de inicio y de fin. Obtener la clave en que será generado el dictado se hace de forma aleatoria pero teniendo en cuenta la probabilidad asociada a cada una de las claves, y con este último resultado es que se obtiene la tesitura del dictado. Con los datos anteriores se obtiene la tonalidad, nuevamente de forma aleatoria y teniendo en cuenta la prioridad de cada una, y en base a este último resultado se procede a ejecutar dos acciones: por un lado transformar los giros melódicos a la tonalidad seleccionada y por el otro, a este último resultado se lo adapta a la tesitura, modificando las alturas de los giros melódicos según corresponda. En caso de que los giros melódicos establecidos no sean capaces de adaptarse a la tesitura, se repetirán los pasos, con otra tonalidad diferente, hasta obtener giros melódicos que estén dentro del rango de notas establecido por la tesitura.

A partir de lo obtenido se procede a transformar la nota de referencia, la nota de inicio y la nota de fin a dicha tonalidad, y éstos dos últimos elementos adaptarlos a la tesitura. Con estos resultados, y teniendo la nota de inicio como primer nota del dictado, quedan definidos todos los elementos que utiliza el dictado. Los tres pasos siguientes consisten en seleccionar de forma adecuada el resto de las notas musicales y así asignarle cada una de éstas a las figuras obtenidas en la etapa anterior.

Paso 1: Agregar al dictado las notas faltantes

Dado que fue una inquietud planteada por personas del área de la música, se decidió que en los dictados es deseable que contengan todas las notas establecidas en los giros melódicos, por lo que el primer paso consiste en poder cubrir dicho requerimiento. Esto se hace, en primer lugar, comparando los giros melódicos con el dictado obtenido hasta el momento y así obtener las notas faltantes. A partir de este conjunto se elige una nota de la siguiente forma:

Se selecciona una nota de forma aleatoria de dicho conjunto, teniendo mayor probabilidad de ser elegida las notas que se encuentran a mayor distancia de la última nota del dictado obtenido hasta el momento. Entiéndase por distancia la cantidad de notas que deben ser agregadas al dictado para construir un "camino" entre dos notas, en base a los giros melódicos. Esto se hace con el objetivo de lograr cubrir todas las notas que aparecen en los giros melódicos.

Teniendo la nota faltante elegida, se procede a agregar al dictado todas las notas necesarias hasta llegar a dicha nota, para lo cual se sigue el siguiente procedimiento: se irá agregando de a una nota, y para cada nota incorporada se calculará el camino más corto hasta la nota en que debe finalizar el dictado. Si dicho camino hasta la nota de fin tiene el largo exacto que debe tener el dictado se completa la lista de notas y termina el flujo. En caso contrario se agrega otra nota y se repite el proceso hasta llegar al paso de agregar la nota faltante elegida.

De esta forma el paso 1 es repetido hasta que se obtiene un conjunto de notas válido para el dictado, en dicho caso finaliza el flujo, o hasta que se hayan incluido todas las notas pertenecientes a los giros melódicos. En este punto puede suceder que, si se configuró un dictado de muy pocos compáses o con un número muy grande de giros melódicos, y además no se logró llegar a la nota de fin, existe la posibilidad de que el dictado obtenido hasta el momento sea más largo que lo estipulado. Si esto sucede se vuelve a llamar a la función, utilizando la recursividad, tantas veces como sea necesario hasta obtener un resultado correcto o hasta llegar a un límite de llamadas recursivas. Este número de llamadas recursivas máximas fue establecido en 20 ya que, por un lado, es una casuística que es muy difícil que suceda y puede ser solucionado si los elementos que fueron seleccionados de forma aleatoria se eligen en diferente orden. Por otro lado, se considera que, al ser una función que no consume grandes cantidades de recursos del dispositivo, se puede tener la libertad apelar a la recursividad esa cantidad de veces.

Paso 2: Completar el dictado

La mayoría de las configuraciones de dictados deberían llegar a este paso ya que, si finalizaron en el paso anterior se debe a una configuración muy pobre por parte del usuario en variedad de elementos musicales, como pueden ser en giros melódicos, largo en compáses o células rítmicas. Al entrar en este paso del flujo se tiene un dictado parcial el cual tiene un largo menor al deseado, y además debe cumplir con que debe finalizar en la nota de fin. Teniendo estos

datos, el principal objetivo será priorizar el hecho de utilizar los giros melódicos que hayan sido configurados para aparecer con una mayor probabilidad.

Para controlar de forma adecuada la correctitud del dictado generado, la primera función a ejecutar consiste en verificar si, a partir de la última nota del dictado construido hasta el momento, se puede obtener el camino más corto del largo establecido, que pueda terminar en la nota de fin. En caso afirmativo, se agrega dicho camino al dictado y termina el flujo. En caso contrario, se busca agregar una nota musical en base a los siguientes criterios: En función de la última nota del dictado, se toman todas las notas que pueden ir a continuación, tomando como regla a seguir los giros melódicos. De este conjunto de notas posibles, se elije una, teniendo una mayor probabilidad las notas que se obtuvieron de aplicar un giro melódico con mayor prioridad.

Una vez se agrega una nota se vuelve a repetir el paso 2 hasta que el dictado sea del largo deseado.

Paso 3: Corregir el dictado

Al entrar en este paso se tiene un dictado el cual es del largo deseado pero puede que no finalice en la nota que se requiere. Si este es el caso, se procede a ejecutar una función recursiva la cual modificará dicho dictado, corrigiendo desde la última nota hasta la primera. Este procedimiento consiste en lo siguiente: Si el dictado no termina en la nota de fin se le quita la última nota y a partir de la nueva nota de fin, la cual es ubicada como raíz, se construye un árbol de altura uno, siendo los hijos de los nodos las posibles transiciones de una nota a otra, en base a los giros melódicos de la configuración del dictado. A partir del resultado obtenido, se busca la nota de fin en las hojas del árbol, y en caso de encontrarse, el camino de dicha hoja hasta la raíz se agrega al dictado, obteniendo así el resultado deseado.

En caso de no encontrarse la nota de fin dentro de las hojas del árbol, se repite el paso 3 de forma recursiva, obteniendo, en el paso n de la recursividad, un árbol de altura n. De esta forma, no solo se asegura que el dictado termine en la nota deseada, sino que, al concatenar al dictado el camino desde la raíz hasta la hoja del árbol, se mantiene el largo del dictado.

Cabe destacar que, si bien en pasos anteriores siempre se controlaba si en algún punto del dictado existía un camino con la menor distancia posible hasta la nota de fin, la diferencia con el paso 3 es que en éste se busca cualquier tipo de camino posible. Además el paso 3 es reiterado un número fijo de veces, el cual fue establecido en 12, ya que por cada recursión, el árbol construido crecerá su

altura en un nivel. Esto sumado a que cada nodo puede eventualmente tener varios hijos, hace que el número límite de cantidad de recursión no aumente ya que el al considerar un número mayor se estaría construyendo una estructura arborescente a la cual se le estaría dedicando una gran cantidad de recursos del dispositivo.

Paso 4: Volver a ejecutar

En caso de no haber obtenido el resultado esperado, se vuelve a ejecutar el paso 1 comenzando nuevamente. Esto hará que se genere un dictado con una nueva lista de notas desde sus comienzos, lo cual lleva a generar un resultado diferente al obtenido. Nuevamente estas iteraciones se reiteran un número fijo de veces, establecido en 20, ya que no es un valor extremadamente grande para saturar los recursos del dispositivo móvil calculando las estructuras que maneja el algoritmo, y por otro lado es un número lo suficientemente alto para poder cubrir un amplio espectro de diferentes dictados.

A modo de resumir lo explicado anteriormente, a continuación se presenta un breve pseudocódigo con las funcionalidades principales:

```
FUNCIÓN  GenerarDictadoMelódico
    // Se selecciona una tonalidad de forma aleatoria
    // (teniendo en cuenta sus prioridades asociadas) y en base
    // a ésta se trasponen los giros melódicos, nota de referencia,
    // notas de inicio y fin.
    Tonalidad = ObtenerTonalidad

    // PASO 1: agregar al dictado notas faltantes
    MIENTRAS hayan notas en giros melódicos que no estén en el dictado:
        // Se elije una nota que no esté en el dictado y se agrega el
        // camino de notas que va desde la última nota del dictado a
        // la nota faltante, en función a los giros melódicos
        Nota = ElegirNotaFaltante
        Notas = NotasHaciaNotaFaltante
        Dictado.INSERTAR(Notas)
    FIN MIENTRAS

    // PASO 2: Completar el dictado
```

```
    MIENTRAS dictado menor al largo estipulado
        // Se elige la nota teniendo en cuenta la última nota del
        // dictado y los giros melódicos (con sus probabilidades
        // asociadas)
        Nota = ElegirNota
        Dictado.INSERTAR(Nota)
    FIN MIENTRAS

    // PASO 3: Corregir el dictado
    DictadoParcial = Dictado
    i = 0
    MIENTRAS dictado no termine en nota fin
        DictadoParcial = QuitarUltimaNota (DictadoParcial)
        i = i + 1
        // Construyo todos los caminos posibles desde la última
        // nota de DictadoParcial de largo i, teniendo en cuenta
        // los giros melódicos. Si existe un camino el cual
        // termine en la nota de fin deseada, retorna el camino.
        Camino = CaminosPosibles (DictadoParcial, i)

        SI (Camino no es vacío)
            DictadoParcial.INSERTAR(Camino)
            RETORNO DictadoParcial
        FIN SI
    FIN MIENTRAS

    // PASO 4: llamar de forma recursiva a la función en caso
    // no obtener un dictado válido
    GenerarDictadoMelódico
FIN FUNCIÓN
```

5.6.3. Fin de la generación aleatoria del dictado

Una vez obtenidos los elementos rítmicos y melódicos, se guardan en la base de datos, para que luego, cuando un usuario estudiante desea reproducir

el dictado, la aplicación pueda leer los datos correspondientes al dictado y generar en tiempo real el archivo .midi y .mp3 para la reproducción en el dispositivo.

En el servidor donde se encuentra publicada la API es donde se obtienen tanto la lista de figuras musicales como las notas correspondientes a cada una de las figuras. Con estos datos sumado al valor en *bpm* de cada figura es que se está en condiciones de generar el contenido multimedia para ser enviado en formato de URI y así ser reproducido por el dispositivo móvil.

5.7. Desplegando la aplicación

En esta sección se abordará los diferentes servicios y recursos que fueron utilizados para lograr dejar disponible la aplicación en diferentes para los diferentes sistemas operativos de los dispositivos móviles.

5.7.1. Servidor de base de datos

La base de datos fue publicada utilizando los servicios de MongoDb Atlas[24] y realizando las configuraciones pertinentes.

Este servidor fue configurado para aceptar peticiones provenientes de cualquier dirección IP dado que, el recurso utilizado para publicar la API (tema bordado en la siguiente sección) no provee una dirección IP estática. Si bien esto podría verse como una falla en la seguridad, se creó un usuario administrador de tal forma que dichas credenciales deben estar presentes al establecer la cadena de conexión hacia la base de datos. Además, dicho servidor fue configurado para restringir el acceso mediante autenticación. Esto hace que la seguridad para acceder a la base de datos esté en gran medida cubierta.

Los servicios contratados en el servidor son los básicos, los cuales cuentan con las siguientes características:

- Versión de MongoDB 4.4
- Capacidad de almacenamiento de 512 MB.
- Cuenta con recursos compartidos, esto es, tanto la RAM de 2 GB como la CPU utilizada para ejecutar las consultas son recursos que son compartidos por otros usuarios de MongoDB Atlas.

[24]https://www.mongodb.com/cloud/atlas/lp/try2 sitio oficial de MongoDb Atlas.

- El proveedor de servicios en la nube utilizado para el almacenamiento es Amazon Web Services, alojado en la región de Northern Virginia, en América del Norte.
- No cuenta con sistema de Backup.
- Maneja el sistema de encartación de datos por defecto.

5.7.2. Servidor de la aplicación

La API desarrollada fue desplegada en Heroku[25], plataforma en la nube que permite alojar una aplicación y tenerla disponible bajo una cierta URL que dicho servicio provee.

Esta plataforma fue contratada con los servicios básicos por lo que los recursos que tiene disponibles para atender las solicitudes enviadas por la aplicación son limitados. Alguna de estas características pueden llegar a limitar el desempeño de la aplicación, siendo una de las más destacables el no contar siempre con la aplicación corriendo. Esto quiere decir que luego de 30 minutos de inactividad el servicio se apaga, lo cual implica que luego de este período de inactividad, al realizarle una petición el servicio debe levantarse para poder procesar la petición. Esto se traduce como una demora en la respuesta.

Si bien este servicio contratado está pensado para ser utilizado para MVPs y proyectos personales, fue de gran utilidad para poder publicar la aplicación.

Heroku fue utilizado junto con Docker[26]. Esta herramienta permite empaquetar software dentro de contenedores, con el objetivo de incluir todo lo necesario para que un determinado software ejecute, teniendo disponible todas las bibliotecas y herramientas que necesita. Similar al funcionamiento de una maquina virtual, los contenedores virtualizan los recursos necesarios en un servidor.

La configuración de Docker utilizada consistió en utilizar como capa base la imagen de Ubuntu 20.04, sobre la cual se instalaron los recursos necesarios como ser NodeJs y herramientas utilizadas para la generación de contenido musical, como son timidity y ffmpeg. Esto quiere decir que la aplicación desplegada en Heroku corre sobre el sistema operativo Ubuntu, de forma virtualizada, teniendo a disposición los recursos mencionados.

[25]https://www.heroku.com sitio oficial de la plataforma.
[26]https://www.docker.com sitio oficial de la herramienta.

5.7.3. Aplicación móvil publicada

La aplicación fue publicada en las tiendas de Google Play[27] y App Store[28]. Para esto fueron creados los perfiles como desarrolladores correspondientes a cada plataforma, con su respectiva licencia.

Una vez realizada las configuraciones pertinentes y habiendo sido aprobada por los encargados de ambas tiendas, la aplicación quedo disponible para su uso en sistemas operativos Android e iOS.

5.8. Etapa experimental

Las pruebas realizadas al software es una etapa del ciclo del desarrollo muy importante para poder disminuir la aparición de defectos en el producto final.

Las pruebas funcionales comprueban que las funcionalidades se comporten según lo esperado. Esto está asociado con la lógica de negocio, es decir, que los resultados obtenidos sean correctos y coherentes desde el punto de vista musical. Dentro de este tipo, las que más estuvieron presentes fueron las pruebas unitarias. Estas eran ejecutadas por los desarrolladores una vez se cerraba una funcionalidad concreta, teniendo el objetivo de asegurar que dicha unidad del código, dentro de un componente dado, brinde los resultados adecuados. Por otro lado, una vez se cerraba un flujo de uso, eran ejecutadas pruebas de integración, en donde se prueban diferentes funcionalidades del software como un grupo. Cabe destacar que dichas pruebas no fueron automatizadas principalmente por dos motivos: el primero es debido a que el equipo de desarrollo debería haber invertido tiempo en especializarse en herramientas dedicadas al objetivo. Esta inversión podría haber sido rentable en proyectos de mayor duración, con un mayor numero de funcionalidades. En segundo lugar, muchas funcionalidades, para ser probadas, incluye un alto grado de conocimiento de la lógica de negocio, por lo que el automatizar las pruebas resultaría complejo.

Las pruebas de aceptación por parte del usuario se hicieron en dos etapas: en la mayor parte del proyecto consistieron en reuniones virtuales en donde se mostraba la aplicación y el funcionamiento de lo desarrollado hasta el momen-

[27] https://play.google.com/store/apps/details?id=prueba.apk link para descargar la aplicación ADA Entrenamiento auditivo para dispositivos Android.

[28] https://apps.apple.com/us/app/ada-entrenamiento-auditivo/id1592608904?ign-mpt=uo%3D2 link para descargar la aplicación ADA Entrenamiento auditivo para dispositivos iOS.

to. En la etapa final del proyecto, junto con la liberación de la aplicación, se les dio a los usuarios perfiles de prueba para que pudieran probar los flujos de uso principales de la aplicación. A través de dicha actividad se obtuvo un feedback de gran valor con el cual se dejaron planteadas las siguientes actualizaciones del sistema.

Las pruebas no funcionales consisten en probar la aplicación bajo ciertas circunstancias y así corroborar bajo que aspectos podría fallar. Estas pruebas, generalmente incluyen pruebas de carga para el rendimiento, pruebas de fiabilidad, escalabilidad, usabilidad, entre otros. Este tipo de pruebas no fueron consideradas en su totalidad ya que, si bien con las tecnologías utilizadas o el diseño construido se contemplan aspectos de escalabilidad o usabilidad, estas pueden resultar más complejas de diseñar. Ejemplos como la escalabilidad podría requeriría del uso de herramientas externas, o en el caso de la usabilidad, requerir a personas especializadas en el tema, que dedicaran una parte importante de su tiempo en evaluar la aplicación.

Capítulo 6

Evaluación de la aplicación

El desarrollo del proyecto fue realizado de forma modular lo cual permitió evaluar y validar las funcionalidades a medida que éstas se iban cerrando. Esto se llevó a cabo con la participación docentes y estudiantes avanzados de la carrera de musica, quienes participaron de forma activa proponiendo ideas y posibles mejoras que consideraban adecuadas, desde el lado musical. Esto permitió llegar con un producto robusto y de gran valor al lanzamiento de la primera versión.

Junto con el lanzamiento de la aplicación se tuvo un período de evaluación, en la cual se probó en diferentes dispositivos móviles. El presente capítulo se centra en dicho período y los resultados obtenidos de estos primeros usos, en un ambiente real de producción.

6.1. Evaluación del equipo de desarrollo

Esta evaluación se centró principalmente en aspectos técnicos, además de verificar el funcionamiento de los flujos principales de la aplicación.

Como primer aspecto técnico se detectó una demora en las respuestas de la aplicación. Esto es debido a que, tanto la API como la base de datos se encuentran publicadas en servidores gratuitos, lo cual hace que se tengan pocos recursos disponibles. Esto desencadenó en que, si la pantalla de carga no es mostrada luego de enviar una solicitud al servidor, el usuario pueda presionar de forma reiterada un botón, enviando varias solicitudes. Si bien la pantalla de carga es utilizada en las llamadas asíncronas de la aplicación para poder darle al usuario un feedback inmediato, existen lugares en los que no fue incorpora-

da. Este inconveniente no fue detectado de forma temprana debido a que las pruebas fueron realizadas de forma local, haciendo que las respuestas sean casi inmediatas, siendo imperceptible dicho caso. A pesar de esto, la aplicación no causó ningún error a causa de este inconveniente.

En cuanto a la evaluación de los flujos principales de la aplicación, se detectó un error al ser utilizada en dispositivos con sistema operativo iOS. Este consiste en que la aplicación no fue capaz de reproducir sonidos, solamente para este tipo de móviles. Este inconveniente fue detectado de forma temprana luego de la primer publicación, el cual se convirtió en un punto de alta prioridad a atacar. La forma de abordarlo consistió en suprimir el uso del Expo[29] como framework de la aplicación, ya que éste presenta inconvenientes al acceder a ciertas funcionalidades en dispositivos iOS. Este fue uno de los mayores cambios que se realizó posterior al primer lanzamiento, ya que se debió crear nuevamente el proyecto en React Native (sin el uso de ningún framework) lo que trajo como consecuencia el tener que evaluar nuevamente la gran mayoría de los componentes utilizados, y en muchos casos arreglar detalles visuales.

Si bien los problemas detectados tuvieron que ser abordados de forma inmediata, en esta etapa no fueron detectados problemas mayores por parte del equipo de desarrollo.

6.2. Evaluación por parte del usuario

Una vez la aplicación se publicó, fue utilizada por estudiantes avanzados de la carrera de música con el objetivo de configurar un curso inicial, orientado a estudiantes que deban enfrentarse a las pruebas de admisión. En esta etapa se encontró un error de funcionalidad de la aplicación, que consistía lo siguiente: al querer configurar los bpm de las figuras, estas no permitían todos los valores necesarios. Esto fue abordado de forma inmediata, siendo solucionado para la siguiente actualización.

Por otro lado se observaron diferentes inconvenientes en el entendimiento de algunos de los flujos de uso generales. Principalmente surgieron dudas con respecto a cómo son administrados los cursos y si éstos son accedidos por cualquier usuario. Además, dado que un mismo curso fue configurado por dos usuarios en esta etapa, resultó confuso si ambos podían asociarle configuracio-

[29]Framework que disponibiliza un conjunto de herramientas para el desarrollo de aplicaciones en React Native. Documentación disponible en: https://docs.expo.dev.

nes de dictados. De esto se puede observar que podría ser de utilidad incluir cierto tipo de información útil con respecto a los cursos, como puede ser los profesores que lo dictan y resaltar cuando un curso es público o personal.

En cuanto a los flujos de uso principales no se destacaron mayores inconvenientes, más allá del error en los bpm encontrado. Esto es un punto a destacar, dado que establecer una configuración de dictado es uno de los flujos principales y de mayor dificultad. Cabe destacar que a dichos usuarios se les presentó parte de las funcionalidades en instancias anteriores, por lo que contaban con cierta noción del uso de la aplicación.

Para poder lograr resultados más contundentes en cuanto a la evaluación de la aplicación por parte de los usuarios será necesario ponerla en práctica en algún curso piloto, en el cual se utilice a ADA como herramienta complementaria al curso. Así se podrá evaluar, en primer lugar, si resulta de utilidad para la formación del estudiante, y en segundo lugar poder evaluar la usabilidad de la herramienta. A pesar de que esto podría resultar de mucha utilidad para poder adaptar la herramienta a las necesidades y capacidades de los usuarios, de la evaluación realizada se observa que la aplicación resultó de utilidad al momento de configurar un curso inicial.

Capítulo 7

Conclusiones

El presente proyecto dio como resultado la aplicación **ADA - Entrenamiento Auditivo** de código abierto, la cual fue evaluada por estudiantes avanzados de la carrera de música, tanto en dispositivos Android como iOS. Esto permitió corroborar que la misma resulta ser funcional y de utilidad tanto para configurar nuevos cursos como para acceder a su contenido.

El resultado obtenido puede ser visto como un aporte hacia la educación musical ya que, en contraste con la investigación de trabajos relacionados, no existe un software que esté orientado a la enseñanza y además brindando los resultados obtenidos. A continuación se muestra una tabla en la cual se comparan trabajos relacionados, abordados en el Capítulo 2 de Marco referencial, con la aplicación desarrollada, haciendo énfasis en las distinciones.

Live music programming in Haskell	**ADA**
Orientado a un público con conocimientos en el lenguaje de programación Haskell.	Orientada a ser utilizada en el ámbito de la enseñanza, enfocada a estudiantes y docentes de música.
No se puede enfatizar en la generación de ciertos elementos musicales	La configuración de los dictados es realizada por los docentes, quienes indican qué elementos musicales desean enfatizar.
Rhytmic Dictator	**ADA**
Los diferentes temas se dictan en un orden pre-establecido sin posibilidad de personalizar los temas abordados en cada dictado.	Enfocado a la enseñanza de la música, en donde el docente indica el camino del aprendizaje.
No se adapta a las necesidades del usuario, ya que debe pasar los primeros niveles para poder acceder a nuevos ejercicios.	Los estudiantes inscriptos a un curso pueden acceder a ejercicios de cualquier módulo, permitiéndole al usuario enfocarse en sus necesidades.
Teoría.com	**ADA**
Se destaca por incluir una variedad muy amplia de configuración de elementos musicales a entrenar.	Si bien la configuración de elementos musicales no es tan amplia, se abarca lo necesario para cursos iniciales de música.
No es una aplicación orientada al entrenamiento auditivo de forma aislada, sin brindarse un seguimiento.	Esta orientada a la educación, contando con la presencia de roles docentes para brindar una guía en el camino del entrenamiento auditivo.
No es una aplicación de código abierto.	Es de código abierto.

Estas comparaciones muestran las diferencias con las aplicaciones ya existentes en el mercado, enfatizando el rumbo que se le dio a la aplicación, en base a las problemáticas que se querían resolver.

De estas aplicaciones se puede ver que ninguno cumple con la totalidad

de los requerimientos relevados. Adicionalmente, de los trabajos relacionados abordados se puede ver que todos son de uso gratuito, a excepción de *Quizlet*, la cual tiene una versión de pago que permite mejorar las funcionalidades relacionadas al los ejercicios que son brindados. Además, solamente las aplicaciones de *Live music programming in Haskell* y *Thyrmic Dictator* son de código abierto. Esto hace que la aplicación de ADA tenga un gran valor sobre el resto de aplicaciones similares.

En base a los requerimientos establecidos por parte de todo el equipo interdisciplinario, se observa que se tuvo una muy buena cobertura, quedando algunos fuera del alcance del proyecto. Esto último fue debido a la complejidad tanto en desarrollo como en testing que acarrean los requerimientos no contemplados. La funcionalidad de mayor importancia que fue dejada para una versión futura es la configuración y generación de dictados armónicos, la cual está basada en una gran cantidad de conceptos ya utilizados para los dictados rítmicos y melódicos. Esto hace que la aplicación cuente con una base lo suficientemente sólida para poder incorporar dicho requerimiento, permitiendo la reutilización de funcionalidades ya testeadas.

En lo que respecta al diseño de la aplicación se considera que se cumplieron los objetivos generales, ya que los flujos de uso resultaron ser intuitivos para los estudiantes de música. Esto es acompañado por un diseño de interfaz que, si bien puede ser mejorado ya que no fue el principal foco de atención del proyecto, cumple con una estética coherente en todas las pantallas de la aplicación. Dicho maquetado cumple con un diseño *responsive* para una gran variedad de pantallas de celulares, a diferencia de las pantallas en dispositivos tablet, en donde, para pantallas muy grandes, los componentes no siempre acompañan en ancho de esta. Si bien esto es un punto a mejorar, el principal foco estuvo en el diseño para celulares, lo cual se entiende que, de todas formas, los resultados obtenidos fueron aceptables.

En cuanto a la interacción que tienen los usuarios con la aplicación se observa buenos tiempos de respuesta al momento de la generación de dictados, lugar donde se encuentra la lógica más compleja de la aplicación. A pesar de éstos buenos resultados, la fluidez de la aplicación, por momentos, se ve afectado debido a que tanto la API como la base de datos se encuentran publicadas en servidores gratuitos. Esto hace que los recursos que se tienen disponibles sean bajos, perjudicando así el tiempo de respuesta de la aplicación, en momentos puntuales.

Durante el transcurso del proyecto surgieron, de forma reiterada, ciertos cambios o adaptaciones en funcionalidades de la aplicación. Ante estos, se observó una rápida respuesta por parte del equipo de desarrollo debido a que las necesidades fueron detectadas de forma temprana, gracias a las prácticas de las metodologías ágiles. En segundo lugar, algunos de estos cambios provocaron modificaciones en el modelado de los datos, con lo que se comprobó la buena flexibilidad de MongoDB al momento de incorporar y modificar estructuras.

En cuanto a los resultados brindados por la aplicación se observa que, tanto los dictados como las soluciones brindadas son coherentes y responden a las reglas establecidas en las configuraciones de dictado. Por otro lado, comparando diferentes resultados, se observa un buen manejo de la aleatoriedad, lo que hace que los dictados generados por la aplicación sean variados. Este punto resuelve el problema que manifestaron algunos estudiantes de música, quienes expresaban que existen plataformas, orientadas al entrenamiento auditivo, que resultan repetitivas y poco variadas.

En cuanto a los objetivos del proyecto y los problemas que se planteó atacar se obtuvieron muy buenos resultados. En primer lugar, la aplicación genera dictados de calidad por lo que es un buen recurso a utilizarse en el dictado de cursos de música. Esto permite que el entrenamiento auditivo se pueda realizar de forma independiente por los alumnos (en base al curso configurado previamente por el docente) y así dedicar las clases a abordar temas en los que se requiera necesariamente la interacción entre el estudiante y docente. Esto tiene el objetivo de funcionar como un complemento y no con el fin de sustituir los entrenamientos auditivos en las clases.

Por otro lado, la aplicación fue utilizada por estudiantes avanzados de música con el objetivo de configurar un curso inicial el cual sirva para los estudiantes que deseen ingresar a carreras relacionadas con la música y deban enfrentarse a una prueba de admisión. Así la aplicación funcionará como un complemento extra para que dichos estudiantes adquieran la formación musical básica que deben tener para el ingreso. A estas configuraciones se les puede agregar una mayor dificultad, incluyendo otro tipo de elementos melódicos, lo cual servirá a estudiantes con mayores conocimientos a seguir perfeccionando su oído musical.

La aplicación liberada cuenta con las características necesarias para que los flujos principales de creación y generación de dictados sean utilizados tanto por docentes como por estudiantes. Si bien, como primera versión de la aplicación

se tiene pendiente realizar actualizaciones y correcciones menores, cuenta con un desarrollo sólido como base, sobre la cual poder agregar y construir nuevas funcionalidades. Esto hace considerar a la aplicación como un aporte a la educación musical, tanto formal como informal.

Capítulo 8

Trabajo a futuro

La aplicación para dispositivos móviles que se tuvo como resultado del presente proyecto cuenta con las funcionalidades básicas para el entrenamiento auditivo por parte de estudiantes. Sin embargo, existen requerimientos y nuevas funcionalidades que surgieron en el transcurso del trabajo, las cuales quedaron fuera del alcance del proyecto. Esto queda planteado como trabajo a futuro, para ser incorporado en la aplicación en versiones posteriores, generando como resultado una aplicación mucho más potente y completa, aportando un mayor valor al dictado de los cursos de música. Esto es posible gracias a que la versión de la aplicación liberada en el presente proyecto cuenta con una base lo suficientemente sólida y flexible como para permitir la incorporación de nuevas funcionalidades. Esto es debido a que la lógica con la que se generan los dictados rítmicos y melódicos genera buenos resultados, a partir de los cuales se planea expandir la plataforma.

8.1. Pendientes

En primer lugar se abordarán los pendientes que fueron planteados para el proyecto, que por temas de tiempos y complejidad no fueron incluidos.

En cuanto a los requerimientos no funcionales, los requerimientos *Req. 005* y *Req. 006* no fueron cubiertos. De estos, el que debe atacarse con mayor prioridad es el primero (se requiere que los datos que maneja la aplicación no sean accedidos por personas externas), ya que al no contar con un método de autenticación por *token*, las funcionalidades que deja disponible la API no están restringidas solamente a los usuarios del sistema. En segundo lugar, el reque-

rimiento 006 (incorporar la conexión con otras plataformas educativas) no fue abarcado, pero sería algo deseable tener, ya que facilitaría la administración de instituciones educativas, docentes y estudiantes. De toas formas, la aplicación cuenta con una arquitectura y tecnologías que son adecuadas para incorporar la comunicación con sistemas externos.

En cuanto a la generación de dictados, los requerimientos *Req. 016* y *Req. 020* no fueron liberados en la versión actual de la aplicación. En primer lugar, tanto la generación de dictados armónicos como el incluir un parámetro *humaizer* no fueron incluidos debido a la complejidad y tiempo tanto de desarrollo como de testing que acarrean. Por un lado, el poder generar dictados armónicos es uno de los requerimiento de los que más valor agregaría a la aplicación ya que este tipo de ejercicios es muy utilizado en diversos cursos de música, además de ser el tipo de dictados que más atractivos resultan a los usuarios, permitiendo llegar a un público de carácter autodidacta. El incorporar un parámetro *humanizer* permitiría generar dictados, tanto rítmicos, melódicos o armónicos, más reales. Es decir, el agregar cierto tipo de imperfecciones en cuanto a los tiempos, da como resultado un dictado más cercano a un dictado reproducido por un docente.

Los requerimientos *Req. 024* y *Req. 025*, pertenecientes al manejo y gestión de la aplicación fueron parcialmente contemplados. El primero de estos consiste en incorporar una guía integrada en la aplicación para que el usuario pueda consultar sobre las funcionalidades. Esto, si bien no está incorporado, existen indicadores que validan la correctitud de los datos ingresados, el cual incluye una pequeña guía sobre como solucionar las inconsistencias en caso de que existan. En segundo lugar, el requerimiento *Req. 025* consiste en poder acceder al detalle de las configuraciones ya existentes en el sistema y poder tomarlas como modelo para crear las propias. Con respecto a este punto, las configuraciones se almacenan en la base de datos, junto con ciertos metadatos para facilitar la búsqueda, la cual debe ser implementada en las pantallas de creación de configuraciones de dictados.

Finalmente, en cuanto a los requerimientos relevados, quedan planteados como trabajo a futuro el *Req. 030* y *Req. 031* los cuales están estrechamente relacionados a la creación de un perfil con rol administrador. Estos requerimientos son de los principales puntos a atacar dentro de las funcionalidades relacionadas a la administración de la aplicación. Con respecto a este punto se debe analizar el contexto en que funcionará la aplicación para saber quie-

nes tendrán el rol de administrador, ya que, dependiendo si serán los propios docentes o alguien ajeno a la aplicación, dicho rol estará incluido dentro de la aplicación o se implementará mediante el uso de un portal web.

8.2. Mejoras

Existen ciertas funcionalidades que surgieron al evaluar la versión final de la aplicación y otras que, si bien surgieron en el transcurso del proyecto, no fueron agregadas como requerimientos necesario para la primer liberación del producto, y fueron agregadas como funcionalidades deseables a incorporar, las cuales son enumeradas a continuación.

- Como usuario quiero poder inhabilitar las configuraciones de dictado que haya dado de alta, para que no se sigan generando dictados a partir de la misma.
  Debido a que no es conveniente eliminar de forma permanente una configuración de un dictado, ya que puede existir estudiantes que hayan generad dictados y entrenado en base a estos, se tendrá la opción de dejarla inhabilitada. Esto se refiere a que cualquier estudiante que acceda no podrá generar nuevos dictados.
- Como usuario docente quiero que la aplicación genere dictados a partir de una configuración que di de alta para poder evaluar la correctitud de los dictados generados, previo a dejarlo habilitado para los estudiantes.
  Esto funcionará de forma similar a la generación de dictados para los estudiantes, con la diferencia que no se hace con el objetivo de evaluación, es decir, no se presentarán las pantallas de autoevaluación Además los dictados generados serán descartados, ya que cumplen la finalidad de ser escuchados solamente a modo de ejemplo.
- Como usuario estudiante quiero poder girar el celular en la pantalla de soluciones para que el pentagrama sea visualizado de forma horizontal y facilite su revisión.
  Esta funcionalidad surgió a raíz de presentar la aplicación a futuros usuarios, a quienes les resulta intuitiva esta forma de operar.
- Como usuario estudiante quiero poder escribir los dictados en la aplicación y que ésta evalúe el desempeño de forma automática para no tener que utilizar una hoja pentagramada y evaluar la correctitud del ejercicio

de forma manual.

Para esta funcionalidad deberá investigarse la forma más sencilla e intuitiva de escribir dictados. Para esto se investigarán aplicaciones actuales, como es la aplicación *iWriter* la cual lleva un largo tiempo en el rubro y soluciona problemas similares.

- Como usuario de la aplicación quiero que las calificaciones se muestren con un formato gráfico para facilitar y agilizar su entendimiento.

  Esto consiste en generar gráficas del desempeño de los estudiantes en los cursos y así, tanto el estudiante como el docente, serán capaces de visualizar de forma más amigable la evolución de desempeño. Por otro lado, esto hará más atractiva la aplicación para los usuarios, ya que, al presentar un análisis de sus datos, brindará un servicio más personal.
- Como usuario quiero poder modificar mis datos personales para poder tener el control de mi perfil de usuario.

  Esta funcionalidad consistirá en que el usuario sea capaz de modificar sus datos personales, no solo por cuestiones de gusto sino también por cuestiones de seguridad, como puede ser la modificación de contraseñas.
- Como usuario quiero poder recuperar mi contraseña en caso de ser olvidada para no perder el progreso en dicho perfil.

  Esta funcionalidad es de gran utilidad y será implementada con un envío de mail, ya que el mismo funciona como identificador del perfil de usuario.

Las funcionalidades mencionadas funcionarán a modo de agregar valor a la aplicación, complementando las funcionalidades ya existentes.

8.3. Pruebas de rendimiento

Finalmente queda planteado como trabajo a futuro realizar pruebas de rendimiento a la aplicación las cuales se enfocarán en evaluar la respuesta y funcionamiento del sistema bajo ciertas condiciones. Si bien fueron realizadas pruebas similares previo a la liberación de la aplicación, las mismas se acotaron a evaluar el correcto funcionamiento para una cantidad de usuarios y datos acotados.

Por un lado se realizarán pruebas de estrés, las cuales servirán para medir cual es la carga máxima que puede soportar el sistema. Esto está relacionado con los recursos del servidor (como ancho de banda, memoria, entre otros),

con lo cual se podrá estimar un número aproximado de usuarios que soporta el servidor que está actualmente en uso.

Por otro lado las pruebas de resistencia serán ejecutadas para medir la capacidad que tiene el sistema para afrontar situaciones en las que se demanda un recurso específico del sistema, como puede ser número de conexiones a la base de datos.

Por último se tendrán las pruebas de concurrencia las cuales simulan llamadas a una funcionalidad de la API de forma simultánea, evaluando su desempeño bajo este contexto.

Estas pruebas son las que se consideran de mayor importancia realizar ya que esto dará como resultado una estimación aproximada del tamaño y alcance que puede llegar a tener la aplicación, con los recursos que actualmente tiene asignados. En caso de surgir la necesidad del aumento de recursos, estas pruebas deberán volver a ejecutarse, así estaremos asegurando un correcto desempeño de la aplicación, sabiendo bajo qué circunstancias se está al límite de su capacidad.

8.4. Evaluación

Una etapa de vital importancia para la mejora del producto será poner la aplicación en funcionamiento, dentro del contexto del dictado de un curso. Esto no solo ayudará a evaluar qué tan útil resulta ésta, como herramienta de apoyo para la formación del estudiante, sino también evaluar la usabilidad de las funcionalidades. Esto implicaría que, ante la posibilidad de resultar complejo su uso, se opte por reajustar cómo son abordadas ciertas funcionalidades, para lograr un mejor entendimiento de los conceptos manejados, o, en caso de ser necesario, brindar una capacitación, a modo presencial o incluida en la aplicación a modo de manual integrado en la herramienta.

Bibliografía

Badia, A. (2004). Entity-Relationship Modeling Revisited. *ACM SIGMOD Record*.

BATHI. (s.f.). El color de los logos de tecnología [consultado el 07-10-2021]. *https://99designs.es/logo-design/business-of-color/technology*.

Chapaval, N. (2021). Qué es Frontend y Backend: diferencias y características - Platzi [consultado el 10-11-2021]. *https://platzi.com/blog/que-es-frontend-y-backend/*.

Choi, C. J. (2014). A STUDY AND COMPARISON OF NOSQL DATABASES.

Education, I. C. (2020). Arquitectura de tres niveles [consultado el 12-11-2021]. *https://www.ibm.com/es-es/cloud/learn/three-tier-architecture*.

Gong, J. y Tarasewich, P. (2004). GUIDELINES FOR HANDHELD MOBILE DEVICE INTERFACE DESIGN. *College of Computer and Information Science*.

Hammood, A. y Saran, M. (2016). A Comparison Of NoSQL Database Systems: A Study On MongoDB, Apache Hbase, And Apache Cassandra. *International Conference on Computer Science and Engineering*.

inVerita. (2020). .NET Core vs Node.js: What Should You Choose? [consultado el 12-11-2021]. *https://medium.com/swlh/net-core-vs-node-js-what-should-you-choose-3d7a96cd40*.

Kennedy, E. D. (2021). Font Sizes in UI Design: The Complete Guide [consultado el 07-10-2021]. *https://learnui.design/blog/ultimate-guide-font-sizes-ui-design.html*.

Khare, M. (2021). Node.js vs Python: Elige la Mejor Tecnología para Tu Aplicación Web [consultado el 12-11-2021]. *https://kinsta.com/es/blog/nodejs-vs-python/*.

Lamprecht, E. (2021). The Difference Between UX and UI Design – A Beginner's Guide [consul-

tado el 25-09-2021]. *https://careerfoundry.com/en/blog/ux-design/the-difference-between-ux-and-ui-design-a-laymans-guide/*.

Laurenço, J. R. (2015). Choosing the right NoSQL database for the job: a quality attribute evaluation. *Journal of Big Data*, *2*(18).

Mesa, L. F. (2018). Los beneficios de las bases de datos NoSQL [consultado el 31-10-2021]. *https://www.pragma.com.co/blog/los-beneficios-de-las-bases-de-datos-nosql*.

Moreno, M. J. (2021). ¿Cómo leen los usuarios el contenido en línea? [consultado el 03-10-2021]. *https://contenttu.com/blog/marketing-de-contenidos/como-leen-los-usuarios-el-contenido-en-linea*.

Muñoz, J. D. (2017). Qué es Flask [consultado el 12-11-2021]. *https://openwebinars.net/blog/que-es-flask/*.

Nieskens, R. (2017). How to use fonts in your hybrid mobile app [consultado el 07-10-2021]. *https://www.kabisa.nl/tech/how-to-use-fonts-in-your-hybrid-mobile-app/*.

Pandey, R. (2020). Performance Benchmarking and Comparison of Cloud-Based Databases MongoDB (NoSQL) Vs MySQL (Relational) using YCSB.

PandoraFMS. (2015). NoSQL vs SQL; principales diferencias y cuándo elegir cada una de ellas [consultado el 31-10-2021]. *https://pandorafms.com/blog/es/nosql-vs-sql-diferencias-y-cuando-elegir-cada-una/*.

Pesek, M., Suhadolnik, L., Šavli, P. y Marolt, M. (2020). THE RHYTHMIC DICTATOR: DOES GAMIFICATION OF RHYTHM DICTATION EXERCISES HELP? *Conference: International Society for Music Information Retrieval*.

Pinto, R. (2018). 4 simple steps to designing a strong user experience [consultado el 25-09-2021]. *https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/strong-audience-design/*.

Ripalda, C. G. y Garrido, A. (2020). Framework based on Gestalt principles to design mobile interfaces for a better user experience. *Indoamérica University, Mechatronics and Interactive Systems Research Center*.

Romanyuk, O. (2020). NodeJS vs Python: How to Choose the Best Technology to Develop Your Web App's Back End [consultado el 12-11-2021]. *https://www.freecodecamp.org/news/nodejs-vs-python-choosing-the-best-technology-to-develop-back-end-of-your-web-app/*.

Sagredo, M. S. C. (2020). La Educación Musical y su evolución histórica desde comienzos del siglo XX.

Simões, C. (2019). React Native vs Flutter. ¿Cuál es mejor para mi producto? [consultado el 11-11-2021]. *https://www.itdo.com/blog/react-native-vs-flutter-cual-es-mejor-para-mi-producto/.*

Skuza, B., Mroczkowska, A. y Włodarczyk, D. (2021). Flutter vs. React Native – What to Choose in 2021? [consultado el 11-11-2021]. *https://www.thedroidsonroids.com/blog/flutter-vs-react-native-what-to-choose-in-2021.*

Sommerville, I. (2011). *SOFTWARE ENGINEERING* (9.ª ed.). Addison-Wesley.

Thielemann, H. (2013). Live music programming in Haskell. *Halle, Germany.*

ANEXOS

Anexo 1

Desarrollo

1.1. Etapas del proyecto

En la Figura 1.1 se muestra un diagrama de Gantt, con el objetivo de exponer el tiempo de dedicación de cada una de las etapas, desglosado en tareas. El color de cada barra representa tareas pertenecientes a cada una de las etapas. Al mismo tiempo se representa con una flecha entre barras las tareas que dependen de que otra sea finalizada para ser iniciada. Además la escala del tiempo está medida en semanas, es decir, la unidad mínima de cada una de las barras representa una semana.

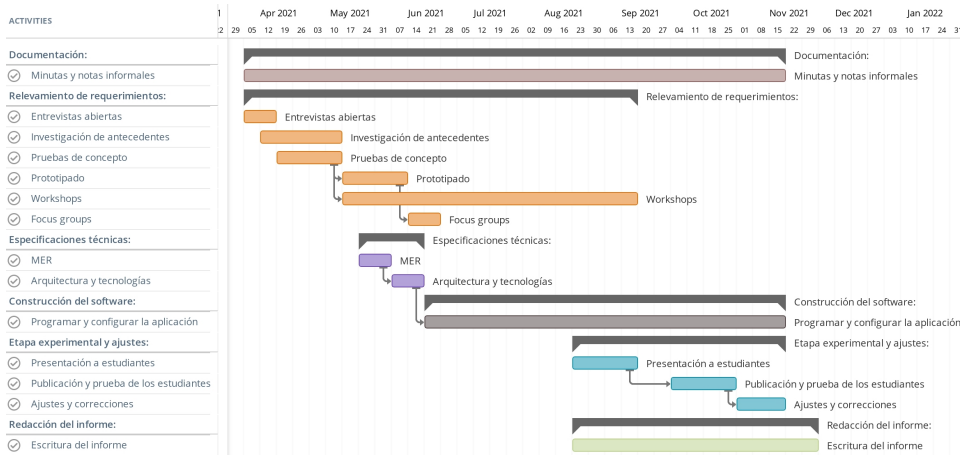

Figura 1.1: Diagrama de Gantt

1.1.1. Documentación

La tarea de tomar notas informales y minutas está presente a lo largo de todo el proyecto ya que las reuniones con docentes y estudiantes de música eran algo recurrente y de las mismas siempre surgen temas que deben ser documentados. Esto abarca desde anotar conceptos musicales, documentar requerimientos o registrar opiniones y comentarios acerca de algún aspecto particular de la aplicación. La información obtenida en esta etapa es utilizada en el resto de tareas del proyecto.

1.1.2. Relevamiento de requerimientos

Si bien esta etapa fue abordada en el capítulo correspondiente al tema, el diagrama muestra las diferentes tareas de forma gráfica, mostrando la duración de cada una e incluso la dependencia con otras tareas.

Se puede ver que las entrevistas abiertas fueron utilizadas para comenzar la etapa de relevamiento de requerimientos ya que, al ser la música un área compleja e incluso desconocida para personas del palo de la ingeniería, es adecuada para empaparse en el tema.

La investigación de antecedentes y pruebas de concepto fueron realizadas en paralelo ya que la primera no requiere de la participación de personas conocedoras del tema y en cierta medida ayudan a seguir profundizando en aspectos musicales. Al mismo tiempo las pruebas de conceptos fueron realizadas a modo de bajar a tierra el conocimiento adquirido, esto es, poder representar en código conceptos musicales como los giros melódicos o las células rítmicas.

Las tareas anteriores, en particular las pruebas de concepto, permitió tener claro conceptos fundamentales para el uso de la aplicación, lo cual habilitó a seguir con las tareas de prototipado y workshops. La primera consistió principalmente en analizar el flujo de uso al momento de configurar un dictado ya que esta funcionalidad es compleja (tanto para desarrollarla como posteriormente para ser usada en la aplicación) y además debe manejar muchos conceptos y términos musicales. En paralelo se comenzaron con los workshops ya que los avances que se realizaban en los prototipos eran presentados a personas cercanas a la música en reuniones al estilo workshops. En estas se definían requerimientos de forma conjunta y se discutían los avances presentados. Esta tarea estuvo presente por gran parte del proyecto ya que luego de que dio comienzo el desarrollo de la aplicación, las funcionalidades que eran terminadas

se analizaban en conjunto y de estas solían surgir nuevos requerimientos. Cabe destacar que al ser un proyecto interdisciplinario, el mismo es gestionado con el uso de metodologías ágiles, permitiendo incorporar nuevos requerimientos que surjan a lo largo del proyecto al software en construcción.

Volviendo a la tarea del prototipado, una vez fue finalizada habilitó relevar requerimientos mediante el uso de focus groups. En estas actividades se utilizó el prototipo construido para presentarselo a estudiantes avanzados de la carrera de música y así discutir los flujos de uso tanto para los estudiantes como docentes.

1.1.3. Especificaciones técnicas

Una vez estaba avanzada la etapa de relevamiento de requerimientos y se tuvo una base de conocimiento lo suficientemente sólida en el tema, se diseñaron las entidades y relaciones que manejaría la aplicación. De esta tarea dependen ciertos aspectos de la siguiente tarea: definir la arquitectura y tecnologías a utilizarse. Si bien ciertos aspectos de la arquitectura fueron definidos en base a otros factores, para decidir la base de datos a utilizar se debe, primero, realizar un profundo análisis de la realidad.

1.1.4. Construcción del software

Una vez se tuvo definida la arquitectura y tecnologías a utilizarse, se dio comienzo a una de las etapas más duraderas del proyecto: comenzar con el desarrollo de la aplicación propiamente dicha. En el diagrama presentado se presenta una sola tarea para esta etapa, la cual engloba todas las subtareas que corresponden al desarrollo de todas las funcionalidades de la aplicación, incluyendo además las configuraciones y comunicación entre los diferentes sistemas. Esto es así ya que el desglosar cada una de éstas tareas entorpecería el análisis de las etapas.

1.1.5. Etapa experimental y ajustes

Esta etapa incluye tareas que involucran a personas cercanas a la música, particularmente se trabajó con estudiantes avanzados de la carrera de música. Al momento de presentarles la versión publicada de la aplicación, el objetivo fue que lograsen usarla tanto para configurar un curso inicial como para acceder a

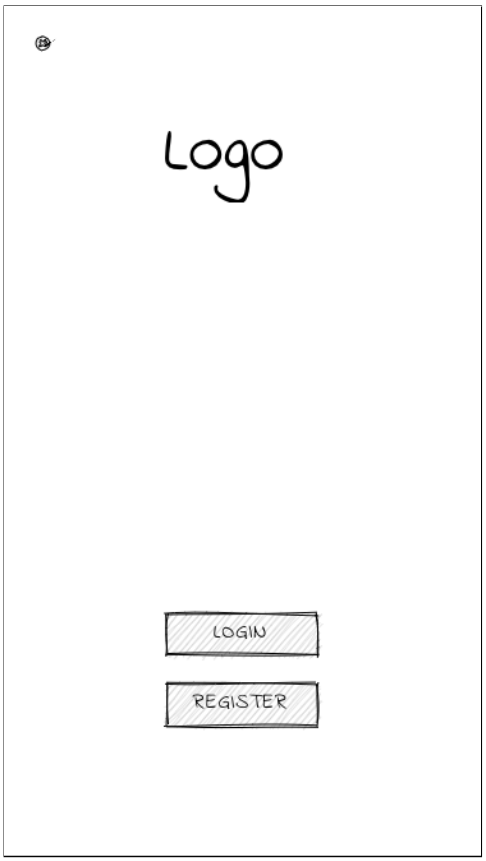

Figura 1.2: Pantalla de inicio

los dictados generados. A partir de este primer encuentro se les preguntó sobre su uso para abordar tanto temas funcionales como de sensaciones al utilizarla, a modo de poder tener opiniones provenientes de posibles futuros usuarios.

1.1.6. Redacción del informe

El informe comenzó a escribirse en paralelo a otras etapas, como ser la etapa experimental y construcción del software ya que se contaba con una gran cantidad de información relevada a lo largo del transcurso del proyecto.

1.2. Figuras Prototipo de baja fidelidad

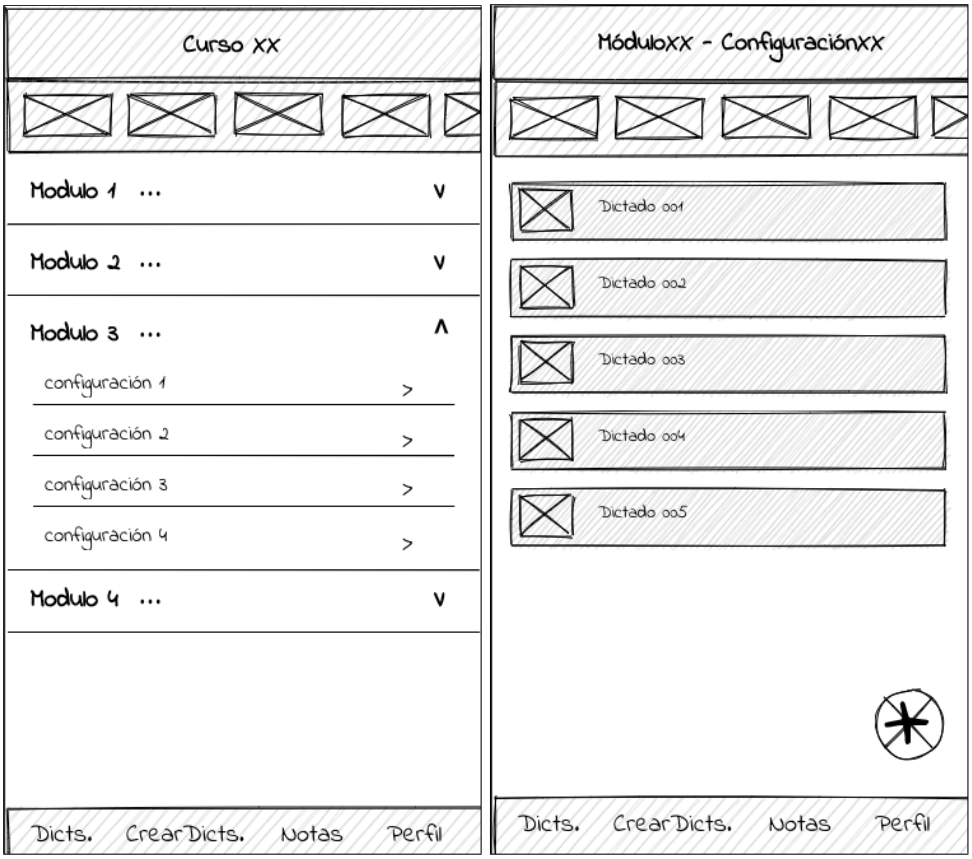

Figura 1.3: Pantallas de cursos,modulos y dictados

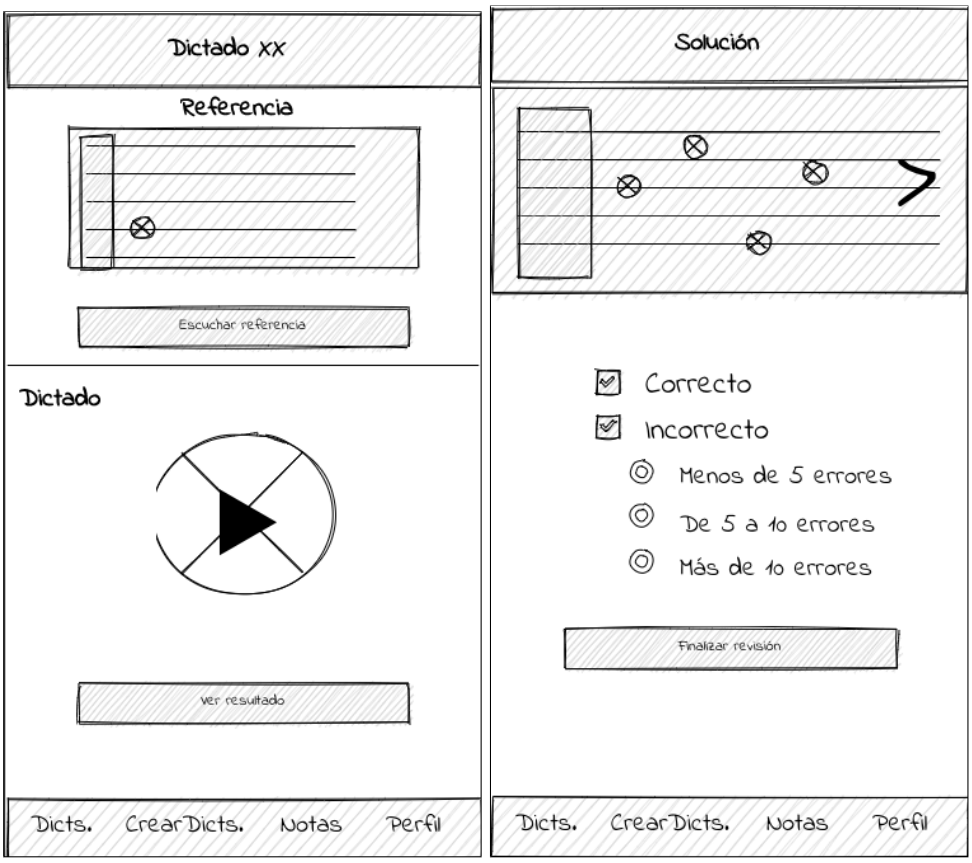

Figura 1.4: Pantallas de dictados y soluciones.

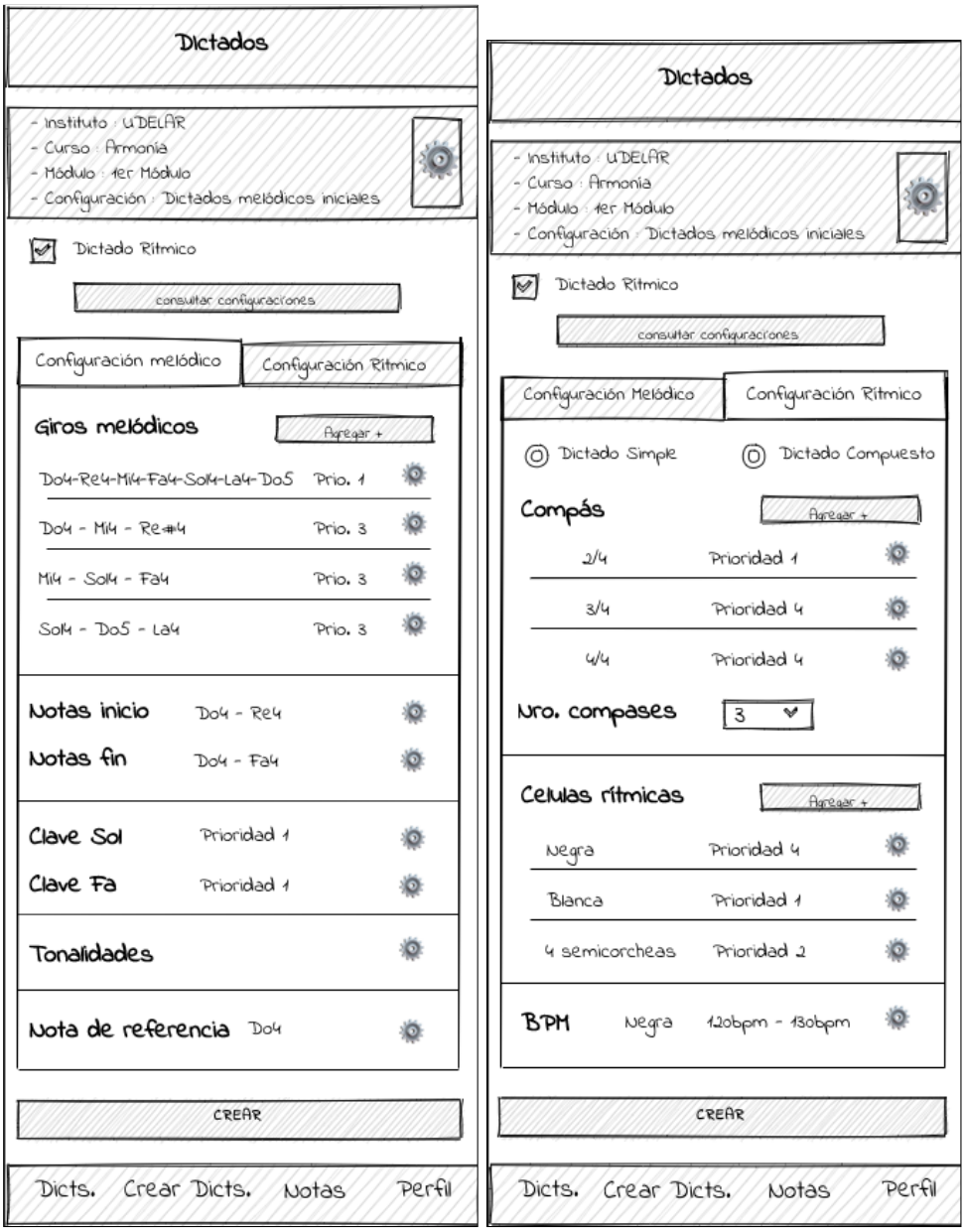

Figura 1.5: Creación de dictados

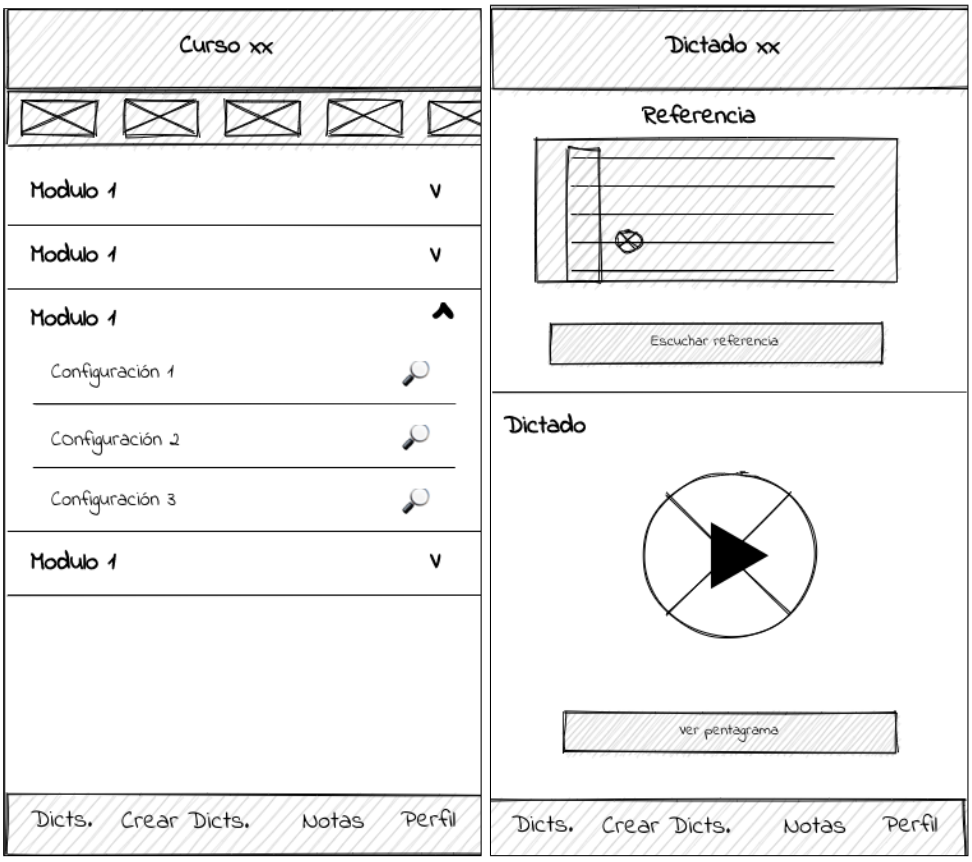

Figura 1.6: Escuchar dictados

1.3. Generación de dictados

Para profundizar en cómo se implementó la generación aleatoria de los dictados se presenta un pseudocódigo de cómo se realizó:

```
function GenerarDictado() {
    DictadoRitmico = GenerarDictadoRitmico (
        CelulasRitmicas,
        NroCompases,
        Compas,
        Simple
    )
    DictadoMelodico = GenerarDictadoMelodico (
        GirosMelodicos,
        Tesitura,
        NotasInicio,
        NotasFin,
        Claves,
        DictadoRitmico.Largo,
        Tonalidades,
        NotaReferencia
    )
}

function GenerarDictadoRitmico (
    CelulasRitmicas,
    NroCompases,
    Compas,
    Simple
) {
    function ObtenerFigsValidas (
        CelulasRitmicas,
        Numerador,
        Denominador,
        DictadoParcial
    ) {
```

```
    // El Denominador indica el valor de cada pulso.
    // La función devuelve cuántos pulsos ocupa el DictadoParcial
    // en función del valor de cada pulso
    Valor = SumarValores(DictadoParcial, Denominador)

    // El Numerador indica el número de pulsos de cada compás
    if (Valor == Numerador) return DictadoParcial
    if (Valor > Numerador) return FAIL

    Celula = ObtenerElementoPrioridad(CelulasRitmicas)
    DictadoParcial.push(Celula)

    Dictado = ObtenerFigsValidas (
        CelulasRitmicas,
        Numerador,
        Denominador,
        DictadoParcial
    )

    if (Dictado.EsCorrecto) return Dictado

    QuitarElemento(CelulasRitmicas, Dictado.UltimoElemento())
    if (CelulasRitmicas.Largo == 0) return FAIL

    QuitarUltimoElemento(Dictado)
    Celula = ObtenerElementoPrioridad(CelulasRitmicas)
    Dictado.push(Celula)

    return ObtenerFigsValidas (
        CelulasRitmicas,
        Numerador,
        Denominador,
        Dictado
    )
}
```

```
    Numerador, Denominador = ObtenerNumeradorDenominador(Compas)
    Dictado = [];
    i = 0
    while (i < NroCompases) {
        CelulasRitmicasAux = CelulasRitmicas
        do {
            DictadoParcial = ObtenerElementoPrioridad(CelulasRitmicasAux)

            // Obtener figuras válidas correspondientes a un compás
            FigsCompas = ObtenerFigsValidas (
                CelulasRitmicas,
                Numerador,
                Denominador,
                [DictadoParcial]
            )

            // Si no se llega a obtener un conjunto válido de figuras se
            // vuelve a intentar comenzando
            // en una célula rítmica diferente
            if(FigsCompas == FAIL)
                QuitarElemento(CelulasRitmicasAux, DictadoParcial)
        } while (FigsCompas == FAIL || CelulasRitmicasAux.Largo > 0)

        if (FigsCompas == FAIL) return ERROR

        Dictado.push(FigsCompas)
        i++
    }

    return Dictado
}

function GenerarDictadoMelodico (
    GirosMelodicos,
    Tesituras,
    NotasInicio,
```

```
    NotasFin,
    Claves,
    CantidadFiguras,
    Tonalidades,
    NotaReferencia
){
    NotaInicio = ObtenerElementoAleatorio(NotasInicio)
    NotaFin = ObtenerElementoAleatorio(NotasFin)
    Clave = ObtenerElementoPrioridad(Claves)
    Tesitura = ObtenerTesituraParaClave (
        Tesituras,
        Clave
    )

    do {
        Tonalidad = ObtenerElementoPrioridad(Tonalidades)

        GirosMelodicosTransformados = TransformarNotasATonlaidad (
            GirosMelodicos,
            Tonalidad
        )

        // Modifica las alturas de los giros melódicos para que
        // estén dentro de la tesitura
        // En caso de no poder devuelve vacío
        GirosMelodicosTransformados = AdaptarATesitura(
            GirosMelodicosTransformados,
            Tesitura
        )

        if (GirosMelodicoTransformados.Largo == 0)
            QuitarElemento( Tonalidades, Tonalidad)
    } while (GirosMelodicosTransformados.Largo == 0 || Tonalidades.Largo > 0)

    if (Tonalidades.Largo == 0) return ERROR
```

```
    // Se transforman a la tonalidad y se adapta a la tesitura
    NotaReferenciaTransformada = TransformarATonalidadYAdaptarATesitura(
        NotaReferencia,
        Tonalidad,
        Tesitura
    )
    NotaInicioTransformada = TransformarATonalidadYAdaptarATesitura(
        NotaInicio,
        Tonalidad,
        Tesitura
    )
    NotaFinTransformada = TransformarATonalidadYAdaptarATesitura(
        NotaFin,
        Tonalidad,
        Tesitura
    )

    Dictado = GenerarDictadoNotas(
        GirosMelodicosTransformados,
        NotaInicioTransformada,
        NotaFinTransformada,
        CantidadFiguras,
        GirosMelodicosTransformados,
        0
    )

    if (Dictado.Largo == 0) return ERROR

    return [Dictado, Clave, Tonalidad, NotaReferenciaTransformada]
}

function GenerarDictadoNotas(
    GirosMelodicos,
    NotaInicio,
    NotaFin,
    LargoDictado,
```

```
    NotasObligatorias,
    NroRecursion
){
    Dictado = [NotaInicio]
    NotasFaltantes = ObtenerNotasFaltantes(
        NotasObligatorias,
        Dictado
    )

    // PASO 1: agregar al dictado notas faltantes
    do{
        // Devuelve una nota perteneciente a NotasFaltantes
        // teniendo mayor prioridad las notas que estén a mayor
        // distancia de Dictado.UltimoElemento, siendo la
        // distancia la cantidad de notas que hay entre una nota
        // hasta la otra nota tomando como transición los GirosMelodicos
        NotaDestino = ElegirNotaFaltante(
            NotasFaltantes,
            GirosMelodicos,
            Dictado.UltimoElemento
        )

        // Se maneja una estructura de árbol en una función recursiva
        // para encontrar el camino  más corto desde
        // Dictado.UltimoElemento hasta NotaDestino
        DictadoParcial = FinalizarDictado(
            Arbol = [{
                Nota: Dictado.UltimoElemento,
                Padre: null
            }],
            NotaDestino,
            NroRecursionFinalizarDictado = 0,
            GirosMelodicos
        )

        // Además de concatenar Dictado con DictadoParcial,
```

```
    // a medida que va concatenando cada nota, se fija si
    // existe un camino de largo mínimo el cual, de una nota
    // del dictado termine en la NotaFin en el largo deseado
    Dictado = ConcatenarDictado(
        Dictado,
        DictadoParcial,
        LargoDictado,
        NotaFin,
        GirosMelodicos
    )

    NotasFaltantes = ObtenerNotasFaltantes(
        NotasObligatorias,
        Dictado
    )
} while(NotasFaltantes.Largo > 0 && Dictado.Largo < LargoDictado)

if (Dictado.Largo > LargoDictado)
    if (NroRecursion < LIMITE_NUMERO_RECURSION)
        return GenerarDictadoNotas(
            GirosMelodicos,
            NotaInicio,
            NotaFin,
            LargoDictado,
            NotasObligatorias,
            NroRecursion++
        )
    else
        return ERROR

// PASO 2: Completar el dictado
while(Dictado.Largo < LargoDictado) {
    // Se fija si puede terminar el dictado en NotaFin
    // en el largo deseado, en caso afirmativo
    // devuelve el dictado
    DictadoTerminado = TerminarDictado(
```

```
        Dictado,
        LargoDictado,
        GirosMelodicos,
        NotaFin
    )

    if (DictadoTerminado.Largo != 0){
        Dictado = DictadoTerminado
    } else {
        // Devuelve una nota válida para el último elemento
        // del dictado, teniendo en cuenta los GirosMelodicos
        // y su respectiva prioridad
        NuevaNota = ObtenerNotaDeGiroMelodicoConPrioridad(
            Dictado.UltimoElemento,
            GirosMelodicos
        )
        Dictado.push(NuevaNota)
    }
}

// PASO 3: Corregir el dictado

// Función recursiva que hace que el dictado finalice
// en nota fin. Va quitando a partir de la úlitma nota
// del dictado hacia atrás, y cada vez que quita una
// construye una estructura de árbol para fijarse si
// existe un camino para que el dictado termine en NotaFin
// en el largo deseado.
Dictado = FinalizarEnNota(
    Dictado,
    NotaFin,
    GirosMelodicos
)

// PASO 4: llamar de forma recursiva a la función en caso
// de no obtener el resultado
```

```
    if (Dictado.Largo == 0)
        if (NroRecursion < LIMITE_NUMERO_RECURSION)
            return GenerarDictadoNotas(
                GirosMelodicos,
                NotaInicio,
                NotaFin,
                LargoDictado,
                NotasObligatorias,
                NroRecursion++
            )
        else
            return ERROR

    return Dictado
}
```

Anexo 2

Diseño

2.1. Prototipo de alta fidelidad

En cuanto a los colores para brindar mensajes informativos (tanto de casos de éxito, de error o simplemente informativos) fueron utilizados los colores de la siguiente tabla:

Color alertas exitosas

Texto

Código color borde: `#c3e6cb`
Código color texto: `#155724`
Código color fondo: `#d4edda`

Color alertas de advertencia

Texto

Código color borde: `#ffeeba`
Código color texto: `#856404`
Código color fondo: `#fff3cd`

Color alertas de error

Texto

Código color borde: `#f5c6cb`
Código color texto: `#721c24`
Código color fondo: `#f8d7da`

2.2. Creación de dictado

Para establecer los giros melódicos, personas del área musical recomendaron que éstos fueran escritos como intervalos, es decir, en la pantalla se les presenta un teclado, en donde cada tecla corresponde a un intervalo. Ese intervalo se traduce a una nota, dependiendo si el intervalo es calculado en menor o mayor.

A continuación se establecen la correspondencia entre los intervalos y las notas musicales.

2.2.1. Tabla intervalos menor

Intervalo	Nota	Intervalo	Nota	Intervalo	Nota
1	A2	1	A3	1	A4
1+	A#2	1+	A#3	9m	Bb4
2m	Bb2	2m	Bb3	9M	B4
2M	B2	2M	B3	9+	A#4
2+	B#2	2+	B#3	10m	C5
3m	C3	3m	C4	10+	B#5
3M	C#3	3M	C#4	10M	C#5
4-	Db3	4-	Db4	11-	Db5
4J	D3	4J	D4	11J	D5
4+	D#3	4+	D#4	11+	D#5
5-	Eb3	5-	Eb4	12-	Eb5
5J	E3	5J	E4	12J	E5
5+	E#3	5+	E#4	13m	F5
6m	F3	6m	F4	13+	E#5
6M	F#3	6M	F#4	14m	G5
7-	Gb3	7-	Gb4	14M	G#5
7m	G3	7m	G4	15-	Ab5
7M	G#3	7M	G#4	15J	A5
1-	Ab3	1-	Ab4		

2.2.2. Tabla intervalos mayor

Intervalo	Nota	Intervalo	Nota	Intervalo	Nota
1	C3	1	C4	8J	C5
1+	C#3	1+	C#4	7+	B#4
2m	Db3	2m	Db4	8+	C#5
2M	D3	2M	D4	9m	Db5
2+	D#3	2+	D#4	9M	D5
3m	Eb3	3m	Eb4	9+	D#5
3M	E3	3M	E4	10m	Eb5
4-	Fb3	4-	Fb4	10M	E5
4J	F3	4J	F4	11-	Fb5
4+	F#3	4+	F#4	11J	F5
5-	Gb3	5-	Gb4	11+	F#5
5J	G3	5J	G4	12-	Gb5
5+	G#3	5+	G#4	12J	G5
6m	Ab3	6m	Ab4	12+	G#5
6M	A3	6M	A4	13m	Ab5
6+	A#3	6+	A#4	13M	A5
7m	Bb3	7m	Bb4	13+	A#5
7M	B3	7M	B4	14m	Bb5
1-	Cb4	8-	Cb5	14M	B5
				15-	Cb6
				14+	B#5
				15J	C6