Universidad Carlos III de Madrid

Grado en Ingeniería en Informática

Trabajo de Fin de Grado

Diseño e implementación de una aplicación de composición musical educativa basada en AngularJS 2 y Ionic 2

Estudiante: Marcel Bertagnini Pardo
Tutor: Alejandro Calderón Mateos

Índice de contenidos

Índice de Ilustraciones

Índice de tablas

Agradecimientos

Toda esta labor académica, que es hacer una carrera universitaria, culminada con las líneas de este documento, no hubiese sido nunca siquiera pensada de no ser por el enorme esfuerzo de mis padres. Tanto mi madre como mi padre me han infundido una capacidad de aprender, de progresar y en general de llegar más allá de donde creía que estaban mis límites. El amor por la curiosidad de aprender que poseo, nunca hubiese nacido en mí de no ser por el profundo amor de conocimiento que ha tenido mi padre y que me ha infundido durante toda la vida. La capacidad de progreso, que no he visto en ninguna otra persona, que mi madre ha tenido siempre, ha sido un pilar fundamental para seguir adelante.

También quiero mencionar a todo el cuerpo docente con el que he aprendido, no sólo de cuestiones técnicas, sino de cuestiones humanas. Todo este aprendizaje ha sido una enorme ayuda en mi progreso como profesional y también como persona. Es difícil cuantificar la cantidad de cambios positivos que ha sufrido mi personalidad por compartir ideas, proyectos y conocimientos con los diferentes docentes con los que he tenido el gran placer de conversar, aprender y tener distendidas charlas acerca de las novedades tecnológicas en nuestro ámbito profesional, esto me ha hecho querer aún más lo que hago.

Por otra parte, creo justo recordar en estas líneas a todos los compañeros con los que realicé la carrera. Todos ellos han sido unas grandísimas personas con las que me sentía realmente en el mismo barco. En ningún momento hubo sentimientos de competitividad sino por el contrario ganas de ayudarnos unos a los otros, sea con conocimientos técnicos o bien con consejos personales y visiones para el futuro. Entre todos los compañeros hemos conseguido superar las distintas dificultades que fueron surgiendo por el camino en una común unión realmente sorprendente. Muchos de mis compañeros universitarios han pasado a ser amigos y espero que, a pesar de que cada uno tome rumbos distintos, poder seguir disfrutando de la amistad y compañía, en la medida que se pueda, con todos ellos.

Quiero agradecer a todas las grandes personalidades que han creado y desarrollado el mundo computacional. Desde que por primera vez jugará a un videojuego y empezará a investigar acerca de cómo se desarrollan, conociendo el mundo de la informática, y los grandes artífices que la han hecho posible, he sentido una profunda admiración por todas estas personas. El desarrollo en este campo ha permitido transformar la sociedad, de una forma nunca antes vista en la historia, el nivel de progreso y técnica de hoy en día, no es comparable a ninguna otra época. Todo esto nos ha permitido realizar grandes proyectos y ha sido gracias a los distintos artífices que han desarrollado el mundo computacional, su labor no tendrá comparación en la historia.

Por último, y dado el carácter musical de este proyecto, me gustaría dar las gracias a los grandes músicos de la historia. La música, mi otra gran pasión, ha sido un apoyo incondicional a lo largo de mi vida y me ha aportado bastantes valores para entender el desarrollo de las personas. Poder desarrollar algo que pueda ayudar a acercar estos sentimientos que provoca en mí aprender de ella a otras personas me gratifica de una manera enorme.

Gracias.

Resumen

La idea de este proyecto de fin de grado surge de la pérdida de interés por el aprendizaje musical, sobre todo a edades tempranas. Actualmente hay bastantes aplicaciones para poder aprender de distintos temas musicales, sin embargo, son más complejas y no tienen ese carácter simple y sencillo para que la gente se pueda interesar a pesar de no saber nada a nivel musical.

Con este trabajo se pretende ofrecer un medio para aprender la creación de escalas de acordes, impartiendo teoría y de forma que se puedan hacer grabaciones de distintas progresiones, con lo que el usuario descubrirá una forma interesante de aprender a través de la composición. También podrá compartir sus composiciones y gestionarlas, todo ello desde un dispositivo móvil.

Abstract

The main motivation about developing this project, is to offer to users an application to learn some of the concepts of music composition theory. To make this application possible, the development is focused to present an attractive and easy-to-use app so even kids could use it. The presentation layer of the application, will be designed thinking in the interactive way people use mobile applications. This last point will be very important in the implementation process since the real instruments are mainly based in the concept of interactive and this application needs to give that concept too.

The other mainly point about this technologic solution is to offer an application that allows users to share their compositions with the theory learnt through the application. This concept will be very important during the development as almost the whole mobile applications currently make use of sharing concepts and social networks.

1. Introducción

A lo largo este apartado se explicarán distintos aspectos del proyecto y cuál será **la visión general de este documento**. Para ello, partiremos de la explicación de la necesidad de desarrollar este proyecto y a través del análisis de esta necesidad, identificar y establecer los objetivos que se pretenden alcanzar. Por último, se detallará la estructura del documento, de forma que se explique de forma más detallada cada apartado.

1.1. Motivación

Tras hacer un análisis del panorama del aprendizaje musical en la sociedad actual, se puede observar que hay un déficit de herramientas para poder obtener los conocimientos básicos de teoría musical para poder llegar a componer. Partiendo de este déficit, la mayoría de herramientas que hay en el mercado están más orientadas a un perfil con conocimientos musicales previos y no hay herramientas capaces de incentivar el aprendizaje, a través de la presentación de conceptos musicales de forma simple y amena, a los usuarios de un perfil, con ningún conocimiento previo o muy escaso. La mayoría de herramientas del mercado que se han analizado, o bien no tienen realmente un carácter didáctico, son bastante complejas para un perfil sin conocimiento previo o necesitan una inversión económica.

Tomando en cuenta todos estos problemas y su posterior análisis , el presente proyecto pretende presentar una herramienta que sea capaz de acercar un concepto básico en la composición musical como es el conocimiento de formar progresiones de acordes conociendo como están formadas las distintas escala de acordes y permitiendo al usuario que pueda generar sus propias composiciones en un ámbito personal y a la vez social, pudiendo publicar estas composiciones para que otros usuarios de la herramienta puedan escucharlas.

Por último, se quiere llegar al mayor número de usuarios intentando hacer una herramienta que pueda ser utilizada por la mayoría de personas intentando que el desarrollo pueda llevarse a cabo para varias plataformas.

1.2. Objetivos

Partiendo del análisis de los problemas de las herramientas que pretenden enseñar algunos aspectos de la teoría musical se han establecido una serie de objetivos que pudieran solucionar la mayor parte de estos problemas.

- La herramienta debe presentar una interfaz gráfica de forma fácil e intuitiva para el usuario, de forma que la curva de aprendizaje por parte del usuario de la herramienta sea los más corta posible.
- La presentación de teoría musical dentro de la herramienta debe ser muy visual, intentando evitar textos complejos.
- La solución irá dirigida a usuarios con pocos o ningunos conocimientos de teoría musical.
- Se evitará en lo posible que la solución tenga que requerir una inversión económica por parte del usuario.
- La herramienta deberá poder ser utilizada en diversas plataformas para llegar al máximo número de usuarios.

- Se deberá orientar el desarrollo de la herramienta para que tenga un carácter social, entre los distintos usuarios que la utilicen.

1.3. Estructura del documento

En este apartado se describirá cual será la estructura del este documento indicando, en términos generales, cuál es la información que contiene cada apartado. Se debe destacar que este proyecto se ha basado en la metodología *Métrica v3*, por lo que la estructura de los apartados es parecida, pero se usan solo los apartados de la metodología que correspondan al proyecto.

- **Capítulo 1 – Introducción:** primer apartado del documento. Identifica las características generales del contenido y la estructura que del mismo.

- **Capítulo 2 – Estado del arte y Estudio de mercado:** presentación del contexto de la solución. Se revisará una comparativa sobre las herramientas encontradas en el mercado, se identificarán sus características y se compararán estas características con las que se esperan de la solución deseada.

- **Capítulo 3 – Análisis:** en este apartado se revisarán las características necesarias de la solución que se necesita para posteriormente separar las partes y la responsabilidad de cada una.

- **Capítulo 4 – Diseño:** basado en el apartado anterior, se diseñará una solución por partes, que globalmente cubra el total de los requerimientos de la solución deseada.

- **Capítulo 5 – Implementación**: en este apartado se explicará cual ha sido el procedimiento que se ha llevado a cabo para implementar la solución deseada.

- **Capítulo 6 – Pruebas:** una vez terminado, el sistema debe ser probado para poder garantizar que su funcionamiento es lo que se había esperado de él.

- **Capítulo 7 – Planificación y presupuesto:** en este apartado se verá cuál ha sido la planificación del proyecto, indicando cuales son los plazos que se han estimado y los que se han cumplido. También se analizará el resultado del coste del proyecto y se explicarán un par de alternativas de monetización.

- **Capítulo 8 – Conclusión y trabajos futuros:** finalmente, se explicará aquello que no se ha podido llevar a cabo al finalizar el proyecto. Se explicarán cuáles son las opciones de mejora futura que se pueden implementar sobre lo que se ha desarrollado.

- **Apéndice I – Diagrama de Gantt:** carta Gantt de la planificación del proyecto en tamaño completo que permite facilitar su correcta impresión.

- **Bibliografía:** en este apartado se indicarán todas las referencias bibliográficas y fuentes que se han consultado en el desarrollo de este proyecto.

1.4. Definiciones y acrónimos

A continuación, se definirán aquellos términos, tecnicismos, siglas y abreviaciones que aparecen en el documento que, por su naturaleza, es difícil explicar por sí mismos. Cuando dichos conceptos aparezcan serán mostrados en cursiva (p.e.: *Framework*), de esta forma se hace denotar que la definición de dicho término se especifica en este apartado:

Ad hoc: expresión en Latin que significa "a medida".

Algoritmo: es un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite llevar a cabo una actividad.

API: interfaz de programación de aplicaciones, se trata un conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a ciertos servicios. Se comporta como una interfaz para comunicar sistemas.

API REST: *API* que funciona bajo el protocolo de transferencia de estado representacional (*REST*). Permite comunicar aplicaciones por medio del intercambio de datos vía *HTTP*.

Aplicaciones cliente servidor: tipo de aplicación que usa una arquitectura segmentada entre 2 elementos: la parte cliente se encuentra normalmente en un terminal mientras que la parte servidor se encuentra en un servidor. Conectando ambas partes por una red es posible hacer que ambas interactúen entre sí.

Base de dato, SGBD: Un sistema gestor de base de datos (SGBD) es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos. Una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Cache: estructura de datos que guarda temporalmente los datos recientes de los procesados.

Callback: función de retro llamada, permite recuperar datos de otras funciones de forma asíncrona.

Clave-valor: composición de 2 datos, una clave que identifica/tipifica el valor y el valor que es el dato en sí mismo.

Dirección IP: número que identifica, de manera lógica y jerárquica, a una Interfaz en red de un dispositivo que utiliza el protocolo TCP/IP

ES6: ECMAScript 6, ES6. Es una especificación del lenguaje de programación JavaScript.

Framework: es un conjunto de librerías estandarizadas o herramientas estandarizadas que permiten desarrollar un software.

HTTP (hypertext transfer protocol): protocolo de transferencia de hipertexto. Protocolo de comunicación de capa 5 orientado a transferir documentos por medio de una red de comunicaciones.

JavaScript: es un lenguaje de programación interpretado del lado del cliente.

JSON (Javascript Object Notation): Formato de texto ligero que se utiliza para el intercambio de datos.

Latencia: en términos prácticos puede entenderse como la suma de retardos temporales dentro de una red.

Link: enlace a una dirección URL.

Máquina virtual, MV, VM: software que simula a una computadora y puede ejecutar programas como si fuese una computadora real

Métrica v3: estándar de documentación de un proyecto de desarrollo de software que sirve para ejemplificar como se debe documentar.

Modelo SOLID: conjunto de 5 buenas prácticas en el desarrollo del software que permite desarrollar sistemas robustos, fáciles de mantener y escalables.

Navegador web: software que permite el acceso a Internet, interpretando la información de archivos y sitios web para que éstos puedan ser leídos.

Online: anglicismo que en ingles significa "en línea", indica que un recurso es accesible por medio de una red, como puede ser internet.

Parser: tipo de aplicación software que permite recuperar datos y transformar su formato, ajustándolo a una forma de salida.

Persistir, persistencia de datos: capacidad de un sistema de información de poder almacenar los datos de forma que, el sistema los preserve y pueda usarlos incluso después de una caída del sistema.

Protocolo TCP/IP: es un conjunto de protocolos que permite la comunicación por medio de redes. Se basa en direcciones IP y comunicaciones TCP/UDP.

Puerto de comunicación, Puerto TCP: un puerto de red es un interfaz de conexión asignado dentro de una dirección IP, permite establecer comunicaciones dentro del protocolo *TCP/IP*.

Red LAN: red de área local, se usa para comunicar localmente dispositivos en red.

Script: conjunto de instrucciones que se escriben en un fichero que se ejecutan secuencialmente por un intérprete.

Smartphone: teléfono inteligente, un dispositivo electrónico que tiene la capacidad de conectarse a internet y hacer peticiones HTTP GET, entre otras funciones.

Smartwatch: reloj inteligente, un dispositivo electrónico que tiene la capacidad de conectarse a internet y hacer peticiones HTTP GET, entre otras funciones.

Socket.io: *framework* escrito en *JavaScript* que permite establecer comunicaciones punto a punto basadas en *websockets*.

Testeable: del inglés "test" (prueba), hace referencia a la propiedad de un software, que puede ser probado.

Unix: es un sistema operativo portable, multitarea y multiusuario. Fue desarrollado por un grupo de empleados de los laboratorios Bell de AT&T. Sirve como estándar para otros sistemas operativos, entre ellos, *Linux*.

Websocket: es una tecnología que proporciona un canal de comunicación bidireccional y *full-dúplex* sobre un único ***puerto TCP***.

Sistema operativo: programa raíz desde el que se ejecutan los demás programas de un ordenador.

iOS: sistema operativo desarrollado por Apple para dispositivos portables como *smartphones* o *tablets*

Android: Sistema operativo desarrollado por Google basado en el kernel de Linux. Está pensado para ser implantado en dispositivos móviles, tablets, relojes inteligentes, televisores y automóviles.

MacOS: Sistema operativo desarrollado por Apple a partir de UNIX.

Stand alone: software que no necesita de conexión a una red para funcionar.

AngularJS: framework para el desarrollo de aplicaciones web, desarrollado por Google que permite la creación de aplicaciones del lado de cliente de una sola página (Single page application).

SPA (single page application): aplicación de única página, es una aplicación que se embebe entera en una sola página web.

Apache cordova: framework de desarrollo para aplicaciones móviles utilizando tecnologías web como HTML y JavaScript.

Ionic: SDK para el desarrollo de aplicaciones móviles híbridas.

SDK (software development kit): conjunto de herramientas que permiten la programación de software.

Swift: lenguaje de programación desarrollado por Apple para el desarrollo de aplicaciones para iOS y MacOs.

Android Studio: SDK para el desarrollo de aplicaciones Android.

MVC (modelo vista controlador): arquitectura de software compuesta de la interfaz gráfica (vista), la lógica de la aplicación (controlador) y los datos (modelo).

2. Estado del arte y Estudio de mercado

En este apartado se revisarán las distintas herramientas que ofrece el mercado del software como posibles soluciones. Se verá una introducción a cada uno de ellos y se revisarán sus características más importantes en una tabla comparativa que será usada más adelante para comparar las herramientas y ver si alguna de ellas sirve como solución a las necesidades explicadas en este documento.

Es importante destacar que para efectos de la metodología de trabajo *Métrica v3* este capítulo es formalmente conocido como Estudio de Viabilidad del Sistema o Estudio de Mercado. Por este motivo, en la planificación de este documento puede apreciarse que el presente capítulo ha sido nombrado como “Estado del arte y Estudio de Mercado”.

2.1. Generalización de las necesidades

Durante mis estudios de Grado en Ingeniería en Informática en la Universidad Carlos III de Madrid aprendí muchas herramientas que me permitirán desarrollarme profesionalmente en el futuro. Paralelamente a esto mantuve mis intereses en la música y en la composición musical y, dentro de ese mundo, no fui capaz de encontrar una aplicación de producción musical que se ajustara a mis necesidades teniendo que hacer un esfuerzo recurriendo a varias fuentes.

Cuando empecé a acercarme al tema de la composición musical, me di cuenta de que no era tarea sencilla, la teoría musical contiene una cantidad de conceptos que necesitan tiempo y esfuerzo para ser dominados. Al recordar esta perspectiva, me surgió la idea de presentar una herramienta que pudiese enseñar de una forma simple un concepto musical como es las progresiones de acordes en una escala. Como parte del aprendizaje constituye el ejercicio, me parece necesario poder disponer de una herramienta que me permita hacer mis propias composiciones de forma que pueda aprender mientras fomento mi creatividad. Por otra parte, también creo que la herramienta no debería restringirse a un determinado grupo de usuarios y debería evitar en lo posible la inversión económica por parte de los usuarios de forma que no encuentren impedimentos para poder disfrutar de la herramienta. Por último, me parece también importante que la solución pueda tener un desarrollo sencillo.

Basados en esto, se pueden definir los requerimientos de este proyecto como los que se mencionan a continuación:

1. La aplicación debe poder enseñar el concepto de composición musical a través de escalas de acordes y poder realizar composiciones con la teoría aprendida.
2. La aplicación debe ser portable a cualquier sistema operativo móvil.
3. La aplicación debe poder usarse con una sola mano.
4. La aplicación debe ser portable a una aplicación web.
5. El coste de las licencias de desarrollo y el mantenimiento de la aplicación no deben ser elevados
6. El uso de internet no debe ser obligatorio, siendo factible usar la aplicación en modo *stand alone*.
7. El tiempo de instalación de la aplicación no debe ser elevado, para ello la aplicación no debe tener un peso en megabytes excesivo.

Por lo que, basados en estas características revisaremos que puede ofertar el mercado que pueda cumplir nuestras necesidades.

2.2. Herramientas encontradas en el mercado

En el mercado de las aplicaciones de composición musical existen muchas alternativas, de las cuales se han seleccionado aquellas que más se ajustan a la solución deseada en función de los requerimientos especificados en el apartado anterior. Los detalles de estas soluciones de pueden ver a continuación.

GarageBand (1)**:** es un software propiedad de Apple Computer que permite producir música usando sintetizadores digitales y herramientas digitales-analógicas de distintos tipos. Con ello puede lograrse una producción musical de nivel medio, con ciertos matices profesionales.

GarageBand es mundialmente conocido por ser uno de los softwares de iniciación para novatos en el mundo de la producción digital, potenciando los conocimientos básicos de forma didáctica. La aplicación viene se serie en los ordenadores Apple, pero también es posible adquirirlo de forma gratuita en dispositivos con *Sistema Operativo IOS*.

Ilustración 1: logo de GarageBand.

La limitación de su licencia al uso exclusivo dentro de Sistemas Operativos de propiedad de Apple es su principal inconveniente, haciéndose obligatorio trabajar en entornos *IOS* o *MacOS* para su uso de forma legal.

Yousician (2)**:** es una aplicación diseñada con el fin de enseñar, práctica y didácticamente como tocar instrumentos musicales de distinta índole, tales como guitarra, piano, ukelele y batería. La aplicación funciona bajo entornos móviles (*Android* e *IOS*), pero también es posible instalarla en entornos de escritorio (MacOS, Windows y Linux).

La principal ventaja de **Yousician** es que permite aprender a tocar instrumentos musicales reales usando procesamiento digital de la señal analógica recibida desde el micrófono del dispositivo, permitiendo ejercitar y mejorar las habilidades del usuario como si se tratase de un profesor particular (3).

Ilustración 2: logo de Yousician.

El principal inconveniente de **Yousician** es que para poder usarlo es obligatorio tener un instrumento musical físicamente, ya que sin él la aplicación no es capaz de capturar la señal analógica y transformarla a digital. Por otro lado, su licencia gratuita limita su uso a una lección gratis cada 12 horas.

Guitar! Smule (4)**:** es un software que permite simular la interacción con una guitarra física por medio de la interfaz táctil de un dispositivo móvil. Tiene una similitud importante con el juego Guitar Hero, permitiendo tocar canciones disponibles de un menú y seleccionando el nivel de dificultad (a mayor nivel, mayor el número de acordes que aparecen).

La licencia gratuita de **Guitar!** permite acceder a ciertas canciones básicas. Para poder acceder a mas canciones es necesario contar con una licencia de pago o también es posible ver publicidad para desbloquear más canciones.

Ilustración 3: logo de Guitar! Smule.

La aplicación en si misma permite simular una guitarra real, pero no simula el tacto real de la misma ni permite componer canciones y no tiene un fin didáctico a nivel de teoría musical, por lo cual no cubre el principal requerimiento que se explicó en el apartado anterior.

Luego de revisar estas soluciones es necesario entender que no se ha encontrado ninguna herramienta en el mercado que cumpla con todos los requerimientos, de ahí nace la iniciativa de desarrollar este proyecto.

2.3. Comparativa entre las herramientas

Una vez completado el estudio de mercado del tema anterior se puede hacer una comparativa de las soluciones encontradas respecto a las necesidades mencionadas en el punto 2.1 y con ello se podrá ver si cumplen de forma total, parcial o nula dichos requerimientos.

Primero, se repasarán las características necesarias y luego el método de evaluación para finalmente hacer la comparativa entre estas herramientas:

1) **La aplicación debe poder componer música:** Dado que el fin mismo de la herramienta es aprender de composición musical, la aplicación debe permitir que los usuarios puedan practicar la teoría aprendida a través de realizar sus propias composiciones.

2) **La aplicación debe ser portable a cualquier sistema operativo:** la aplicación debe poder ser instalada en cualquiera de los siguientes sistemas operativos:

    - iOS
    - Android
    - MacOS
    - Linux

3) **La aplicación debe poder usarse con una sola mano:** dado que la simplicidad es un requerimiento importante se hace necesario que la interacción normal con la aplicación pueda hacerse usando una sola mano indistintamente si el usuario utiliza la izquierda o la derecha.

4) **La aplicación debe ser portable a una aplicación web:** debido a que se pretende llegar al máximo número de usuarios, debe existir la posibilidad de que la aplicación deba ser exportada a una aplicación web, por ello, se hace necesario este requerimiento.

5) **El coste de las licencias de desarrollo y el mantenimiento de la aplicación no deben ser elevados:** debemos recordar que esta aplicación tiene una finalidad educativa, por lo que se hace necesario que los costes de desarrollo y mantenimiento de la misma puedan ser llevados por un *startup* o un equipo de desarrollo de pocas personas.

6) **El uso de internet no debe ser obligatorio, siendo factible usar la aplicación en modo *stand alone*:** al tratarse de una aplicación educativa, es posible que ciertos centros educacionales o ciertas personas que la usen no cuenten con una conexión a internet permanente, por ello, es necesario que la aplicación no limite su funcionamiento al no disponer de una conexión a internet.

7) **El tiempo de instalación de la aplicación no debe ser elevado, para ello la aplicación no debe tener un peso en megabytes excesivo:** dado que la aplicación debe ser instalable en entornos móviles, el peso total de la aplicación se debe poder implantar en un entorno de esas características, que, por lo general, cuentan con poco espacio de almacenamiento.

Estos requerimientos pueden cumplirse, cumplirse parcialmente o no cumplirse, por lo que se usará la siguiente escala de evaluación:

- **SI:** el sistema cumple la métrica / se puede usar de esa forma.
- **PARCIAL:** el sistema cumple la métrica / se puede usar de esa forma, **pero con ciertas limitaciones** respecto a las características mencionadas.
- **NO:** el sistema no cumple la métrica / no es usable de esa forma.

Basados en esto y luego de haber hecho un análisis de las características de las herramientas según las especificaciones encontradas en las webs oficiales de los proveedores, la matriz de trazabilidad de las soluciones y las características deseadas es la siguiente:

Métrica / Herramienta	GarageBand	Yousician	Guitar! Smule
1	SI	NO	NO
2	NO	SI	NO
3	NO	NO	NO
4	NO	NO	NO
5	PARCIAL	SI	SI
6	SI	NO	NO
7	NO	SI	SI

Tabla 1: Comparativa entre la métrica seleccionada y las soluciones encontradas.

Como se puede ver en la matriz anterior, ninguna de las soluciones encontradas cumple con todos los requisitos y el más importante de ellos, el primero, solo se cumple para el caso de GarageBand, pero éste no cumple el segundo. Esto significa que **para poder solventar las necesidades de este proyecto se debe desarrollar una solución *Ad Hoc*** que permita cumplir los requerimientos previamente comentados, haciendo especial énfasis en la faceta didáctica y compositiva de la aplicación.

En el siguiente apartado veremos en profundidad que hemos hecho para mantener estas características positivas.

3. Análisis

3.1. Determinación de casos de uso

En los documentos basados en Métrica 3 (5) este apartado suele estar ubicado en el capítulo de diseño, sin embargo, en este documento extenderemos este apartado antes de los demás puesto que es esencial para comprender los siguientes puntos.

Esto se debe a que se está desarrollando un sistema que está orientado a cumplir con casos de uso concretos y por lo mismo, los casos de uso deben ser el foco principal del análisis de este proyecto.

La plantilla a usar para identificar los casos de uso es la siguiente:

Identificador	
Titulo	
Actores	
Objetivo	
Escenario	
Precondiciones	
Postcondiciones	

Tabla 2: Planilla estándar para identificación de casos de uso.

En la cual los apartados identifican la siguiente información:

• **Identificador:** permite diferenciar unívocamente al caso de uso mediante un código con la forma CU-NN, el cual se compone de:
  - o CU: indica que el identificador corresponde a un caso de uso.
  - o NN: indica el número unívoco del caso de uso.

• **Título:** indica el nombre del caso de uso.

• **Actores:** identifica el rol que juega cada usuario durante la interacción del caso de uso. Puede tomar tres valores distintos.
  - o **Usuario**: indica que el actor es un usuario (p.e.: un *Dealer*) es decir, personas físicas que interactúan con el sistema bajo petición.
  - o **Administrador**: indica que el actor una persona física con acceso privilegiado al sistema para configurar o realizar labores de mantenimiento.
  - o **Maquina**: indica que el actor es un sistema informático externo o un dispositivo, haciendo una petición.

• **Objetivo:** propósito de la interacción del caso de uso.

• **Escenario:** indica los pasos que el actor realiza para completar el objetivo.

• **Precondiciones:** identifica los requisitos que deben cumplirse para que se pueda generar la interacción.

• **Postcondiciones:** indica el estado en el que queda el sistema luego de la interacción.

CU-01	
Titulo	Crear usuario.
Actores	Usuario
Objetivo	Crear un usuario con cuyas credenciales se pueda acceder a las funcionalidades de la aplicación.
Escenario	El usuario descarga la aplicación. Tras esto la inicia, accede al formulario para poder crear un usuario de la aplicación, metiendo su nombre, nombre de usuario, email, contraseña y la verificación de su contraseña y crea su usuario.
Precondiciones	El usuario debe haber descargado previamente la aplicación desde donde corresponda según su plataforma.
Postcondiciones	El sistema crea un usuario con las credenciales propuestas por él mismo, con las que puede acceder a la aplicación.

Tabla 3: Caso de uso CU-01.

CU-02	
Titulo	Login.
Actores	Usuario
Objetivo	Acceder a la aplicación con las credenciales del usuario
Escenario	El usuario, tras registrarse en la aplicación puede acceder a ella a través del usuario y contraseña con el que creó su perfil.
Precondiciones	El usuario debe haberse registrado previamente en la aplicación.
Postcondiciones	El sistema permite al usuario tener acceso a las funcionalidades de la aplicación con sus datos personales.

Tabla 4: Caso de uso CU-02.

CU-03	
Titulo	Logout.
Actores	Usuario
Objetivo	El usuario sale de su perfil para volver a la pantalla en la que puede introducir sus credenciales.
Escenario	El usuario está en la aplicación utilizando sus funcionalidades con sus datos personales y sale de su perfil.
Precondiciones	El usuario debe haber hecho login con sus credenciales.
Postcondiciones	El sistema remite al usuario a la pantalla principal de login de la aplicación.

Tabla 5: Caso de uso CU-03.

CU-04	
Titulo	Recordar usuario.
Actores	Usuario
Objetivo	Recordar las credenciales con las que el usuario hace login.
Escenario	El usuario hace login y decide si quiere que sus credenciales se mantengan guardadas o tenga que introducirlas la próxima vez que acceda a la aplicación.
Precondiciones	Estar registrado en la aplicación.
Postcondiciones	El sistema mantiene las credenciales del usuario hasta que éste quiera que no se recuerden más.

Tabla 6: Caso de uso CU-04.

CU-05	
Titulo	Consultar información personal.
Actores	Usuario
Objetivo	El usuario puede acceder a su información personal en la pestaña de su perfil y consultarla.
Escenario	Al hacer login el usuario puede consultar la información con la que hizo su registro en la aplicación.
Precondiciones	El usuario deberá haber hecho login previamente.
Postcondiciones	El sistema muestra la información de registro del usuario.

Tabla 7: Caso de uso CU-05.

CU-06	
Titulo	Añadir foto.
Actores	Usuario.
Objetivo	El usuario añade una foto para su perfil.
Escenario	El usuario puede añadir en su información personal una imagen directamente desde la cámara de su dispositivo.
Precondiciones	El usuario debe realizar login con sus credenciales.
Postcondiciones	El sistema añade una imagen a los datos personales del usuario.

Tabla 8: Caso de uso CU-06.

CU-07	
Titulo	Consultar teoría musical.
Actores	Usuario
Objetivo	El usuario podrá realizar una consulta acerca de la teoría musical propuesta por la aplicación.
Escenario	El usuario accede a la pestaña de "composición" y en ella accede a un breve tutorial de teoría musical.
Precondiciones	El usuario deberá haber hecho el login previamente.
Postcondiciones	El sistema muestra al usuario un tutorial breve acerca de un concepto de teoría musical.

Tabla 9: Caso de uso CU-07.

CU-08	
Titulo	Tocar acordes de la escala de do mayor.
Actores	Usuario
Objetivo	El usuario podrá ir probando los distintos acordes que componen la escala de do mayor y jugar con ellos.
Escenario	El usuario ve el breve tutorial de conceptos de teoría musical y puede realizar interpretaciones con los acordes propuestos por la aplicación para entender mejor los conceptos musicales.
Precondiciones	El usuario debe haber consultado el tutorial de teoría musical previamente.
Postcondiciones	El sistema permite al usuario tocar los distintos acordes de la escala de do mayor.

Tabla 10: Caso de uso CU-08.

CU-09	
Titulo	Grabar progresión de acordes.
Actores	Usuario
Objetivo	El usuario puede grabar las progresiones de acordes que realice.
Escenario	El usuario, tras consultar la teoría musical y practicar progresiones de acordes con los botones que representan los acordes de la escala de do mayor, puede grabar las progresiones que desee.
Precondiciones	EL usuario debe haber consultado la teoría musical previamente.
Postcondiciones	La progresión de acordes que el usuario ha introducido queda grabada.

Tabla 11: Caso de uso CU-09.

CU-10	
Titulo	Escuchar progresión de acordes grabada.
Actores	Usuario
Objetivo	Se escucha la progresión de acordes que el usuario ha grabado.
Escenario	El usuario tras haber accedido a la pestaña de “composición” y haber grabado una progresión de acordes, puede escuchar esta progresión pulsando el botón para ello.
Precondiciones	Haber grabado previamente una progresión de acordes.
Postcondiciones	El sistema hace sonar la grabación que el usuario previamente ha grabado.

Tabla 12: Caso de uso CU-10.

CU-11	
Titulo	Añadir grabación a la lista personal de grabaciones.
Actores	Usuario
Objetivo	Añadir la grabación que el usuario ha creado a la lista de sus grabaciones personales.
Escenario	El usuario hace una grabación de la progresión de acordes que ha tocado y si quiere puede añadirla a la lista de sus grabaciones personales indicando el nombre de la nueva grabación
Precondiciones	El usuario deber haber hecho una grabación de una progresión de acordes previamente.
Postcondiciones	El sistema añade a la lista de grabaciones personales del usuario y en la lista de grabaciones de todos los usuarios, la nueva grabación de acordes realizada con el nombre indicado.

Tabla 13: Caso de uso CU-11.

CU-12	
Titulo	Consultar grabaciones de todos los usuarios.
Actores	Usuario
Objetivo	Se consultan todas las grabaciones hechas por los distintos usuarios de la aplicación del dispositivo.
Escenario	El usuario se posiciona en la pestaña en la que aparecen todas las grabaciones hechas por los usuarios de la aplicación y puede escuchar y detener todas estas grabaciones.
Precondiciones	El usuario debe hacer login previamente.
Postcondiciones	El sistema muestra la lista de todas las grabaciones que han realizado los distintos usuarios de la aplicación.

Tabla 14: Caso de uso CU-12.

CU-13	
Titulo	Consultar las grabaciones propias.
Actores	Usuario
Objetivo	Se puede escuchar las grabaciones que ha hecho el propio usuario de la aplicación.
Escenario	El usuario, tras haber realizado al menos una grabación, puede ver y escuchar en la pestaña de sus datos personales, las grabaciones que ha realizado previamente.
Precondiciones	El usuario debe al menos haber hecho una grabación y haberla agregado a la lista de sus grabaciones personales.
Postcondiciones	El sistema muestra las grabaciones que el usuario ha realizado previamente y permite escucharlas y detenerlas.

Tabla 15: Caso de uso CU-13.

3.2. Requisitos de sistema

Con la finalidad de formalizar las características mencionadas en el punto anterior, analizaremos todos los casos de uso para identificar cuáles son los requisitos que contiene cada uno de ellos, identificando principalmente dos tipos de requisitos:

Requisitos funcionales (RF): todos aquellos requisitos que indican una función del sistema, entendiendo por función todo aquello que el sistema es capaz de hacer.

Requisitos no funcionales (RNF): todos aquellos requisitos que indican propiedades o restricciones del sistema, entendiendo por propiedades: capacidades, aspectos de seguridad, etc.

Esta división es necesaria puesto que posteriormente los requisitos funcionales deben poder mapearse con subsistemas que permitan el desarrollo de estas funciones. Así mismo, los requisitos no funcionales indicaran las propiedades del sistema tales como: capacidad de almacenamiento, recursos, etc. Finalmente, las restricciones indicaran los límites del sistema tales como: tiempo de respuesta, máximo de peticiones, exigencias de seguridad, etc.
La plantilla a usar para identificar los requisitos es la siguiente:

Identificador	
Titulo	
Prioridad	
Necesidad	
Verificabilidad	
Estabilidad	
Descripción	
Req. relacionados	

Tabla 16: Plantilla estándar para especificación de requisitos.

En la cual, cada campo especificara la siguiente información:

• **Identificador:** será un código asociado a cada requisito que lo diferenciará unívocamente. La codificación se dará en la forma RX-NN, donde "X" cambia en función de:

- F: indica que es un requisito es de tipo funcional
- NF: indica que el requisito es de tipo no funcional, incluyendo tanto los de capacidad como los de restricción.
- NN: indica el numero correlativo del requisito, toma valores entre 01 y 99.

• **Titulo:** indica una breve descripción del requisito.

• **Prioridad:** especifica el nivel de prioridad del requisito, puede tomar valores:

- Alta: indica que el requisito es de implementación obligatoria, es esencial para el funcionamiento del sistema.
- Media: indica que el requisito se puede implementar, pero si no se implementa no es imprescindible para el funcionamiento del sistema.
- Baja: indica que la implementación del requisito en el sistema es opcional.

• **Necesidad:** indica la importancia que tiene el requisito para el cliente.

- Alta: el cliente ha indicado que el requisito debe ser satisfecho.
- Media: el cliente ha indicado que el requisito debe ser satisfecho como segunda prioridad, luego de satisfacer los de prioridad alta.
- Baja: el cliente ha indicado que el requisito es opcional.

• **Verificabilidad:** indica el grado en el que puede constatarse que el requisito está incluido en la solución.

- Alta: el requisito es claramente observable en el sistema final.
- Media: el requisito se puede intuir, pero no es claramente en el sistema final.
- Baja: el requisito no es observable en el sistema final.

• **Estabilidad:** indica el grado en el cual el requisito puede sufrir cambios durante el desarrollo del proyecto.

- Baja: el requisito es altamente afectable por los cambios que puede sufrir el proyecto.
- Media: el requisito puede varia de forma directamente proporcional a los cambios en el proyecto.
- Alta: el requisito es bastante estable, es muy difícil que sufra cambios.

• **Descripción:** explicación detallada de las características del requisito.

• **Requisitos relacionados:** indicara con que otros requisitos mantiene una relación directa, es útil para identificar qué requisitos se trazan con qué casos de uso.

A continuación se enumeran la lista de requisitos funcionales que debe disponer el sistema para su solución:

RF-01	
Titulo	Acceder a la pantalla de registro de usuario.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La solución debe tener un acceso a la pantalla en la que el usuario puede registrarse en la aplicación.
Req. relacionados	RF-02, RF-03

Tabla 17: Requisito funcional RF-01.

RF-02	
Titulo	Introducir credenciales para el registro de usuario.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	El sistema debe ofrecer al usuario campos de texto para que pueda introducir sus datos personales de registro.
Req. relacionados	RF-01, RF-03

Tabla 18: Requisito funcional RF-02.

RF-03	
Titulo	Registrar usuario.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación permite registrar al usuario, incluyendo su información en el sistema.
Req. relacionados	RF-01, RF-02

Tabla 19: Requisito funcional RF-03.

RF-04	
Titulo	Acceder a la pantalla de “log-in”
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación debe proporcionar un acceso a la pantalla de “log-in”.
Req. relacionados	RF-05, RF-06, RF-07, RF-08

Tabla 20: Requisito funcional RF-04.

RF-05	
Titulo	Introducir credenciales para el “log-in”.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación debe proporcionar al usuario campos de texto donde introducir sus credenciales de acceso.
Req. relacionados	RF-04, RF-06, RF-07, RF-08

Tabla 21: Requisito funcional RF-05.

RF-06	
Titulo	Validar las credenciales introducidas para realizar el “log-in”.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación debe validar las credenciales que el usuario introduce para tener acceso a la aplicación.
Req. relacionados	RF-04, RF-05, RF-07, RF-08

Tabla 22: Requisito funcional RF-06.

RF-07	
Titulo	Realizar “log-out”.
Prioridad	ALTA
Necesidad	MEDIA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación debe proveer un “log-out” para que el usuario salga de su sesión.
Req. relacionados	RF-04, RF-05, RF-06, RF-08.

Tabla 23: Requisito funcional RF-07.

RF-08	
Titulo	Tener la opción de recordar las credenciales para realizar el “log-in”
Prioridad	BAJA
Necesidad	BAJA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación debe proveer un elemento con el que el usuario pueda recordar sus credenciales de “log-in”.
Req. relacionados	RF-04, RF-05, RF-06, RF-07.

Tabla 24: Requisito funcional RF-08.

RF-09	
Titulo	Acceder a la pantalla de perfil con los datos de usuario.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación debe proveer un acceso a la pantalla en la que se encuentra la información personal del usuario.
Req. relacionados	RF-10, RF11.

Tabla 25: Requisito funcional RF-09.

RF-10	
Titulo	Añadir fotografía al perfil del usuario.
Prioridad	MEDIA
Necesidad	MEDIA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación debe permitir añadir una fotografía a la información de perfil del usuario.
Req. relacionados	RF-09, RF-11.

Tabla 26: Requisito funcional RF-10.

RF-11	
Titulo	Acceder a la cámara de fotos del dispositivo.
Prioridad	BAJA
Necesidad	MEDIA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación deber permitir acceder a la cámara de fotos del dispositivo con el fin de obtener una fotografía si el usuario lo desea.
Req. relacionados	RF-10, RF-11.

Tabla 27: Requisito funcional RF-11.

RF-12	
Titulo	Acceder a la pantalla de composición y teoría musical.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación tendrá un acceso para poder ir a la pantalla en la que el usuario puede consultar la teoría musical y componer.
Req. relacionados	RF-13, RF-14.

Tabla 28: Requisito funcional RF-12.

RF-13	
Titulo	Consultar teoría musical.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación debe permitir la consulta de la teoría musical desde la pantalla de composición.
Req. relacionados	RF-12, RF-14.

Tabla 29: Requisito funcional RF-13.

RF-14	
Titulo	Acceder a los acordes de la escala de Do Mayor.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	El sistema debe permitir acceder al usuario a los acordes de la escala de Do mayor.
Req. relacionados	RF-12, RF-13

Tabla 30: Requisito funcional RF-14.

RF-15	
Titulo	Reproducir los audios correspondientes a cada acorde de la escala de Do mayor.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación debe presentar un elemento que permita la reproducción de los audios correspondientes a los acordes de Do mayor al ser pulsados por el usuario.
Req. relacionados	RF-16, RF-17, RF-18, RF-19

Tabla 31: Requisito funcional RF-15.

RF-16	
Titulo	Hacer una grabación de las notas que se tocan.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación tendrá un elemento que permita la grabación de la progresión de notas que el usuario toque.
Req. relacionados	RF-15, RF-17, RF-18, RF-19

Tabla 32: Requisito funcional RF-16.

RF-17	
Titulo	Reproducir grabación.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	Tras realizar la grabación, la aplicación deberá proveer un elemento para poder reproducir la grabación.
Req. relacionados	RF-15, RF-16, RF-18, RF-19

Tabla 33: Requisito funcional RF-17.

RF-18	
Titulo	Poner un título a la grabación realizada.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	El sistema deberá dejar poner un nombre a la grabación realizada para que pueda ser identificada al guardarse.
Req. relacionados	RF-15, RF-16, RF-17, RF-19

Tabla 34: Requisito funcional RF-18.

RF-19	
Titulo	Añadir grabación a la lista de grabaciones guardadas.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación deberá permitir guardar las grabaciones realizadas para cada usuario.
Req. relacionados	RF-15, RF-16, RF-17, RF-18

Tabla 35: Requisito funcional RF-19.

RF-20	
Titulo	Acceder a la pantalla que muestra las grabaciones de todos los usuarios.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación deberá proporcionar un acceso a la pantalla en la que se muestran todas las grabaciones de todos los usuarios.
Req. relacionados	RF-21, RF-22.

Tabla 36: Requisito funcional RF-20.

RF-21	
Titulo	Reproducir las grabaciones que aparecen en la pantalla donde se muestran todas las grabaciones de los usuarios.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	En la lista de las grabaciones de todos los usuarios, la aplicación permitirá reproducir cada canción de la lista.
Req. relacionados	RF-20, RF-22.

Tabla 37: Requisito funcional RF-21.

RF-22	
Titulo	Detener las grabaciones que aparecen en la pantalla donde se muestran todas las grabaciones de los usuarios.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	En la lista de las grabaciones de todos los usuarios, la aplicación permitirá detener cada canción de la lista que se esté reproduciendo.
Req. relacionados	RF-02

Tabla 38: Requisito funcional RF-22.

Los siguientes requisitos constituyen la especificación de los requisitos no funcionales del sistema:

RNF-01	
Titulo	El sistema estará desarrollado para plataformas iOS y Android.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	MEDIA
Estabilidad	ALTA
Descripción	El desarrollo de la aplicación se deberá orientar a las plataformas móviles iOS y Android
Req. relacionados	RNF-8

Tabla 39: Requisito funcional RNF-01.

RNF-02	
Titulo	La aplicación estará en idioma español.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	Los distintos literales de la aplicación estarán en idioma español.
Req. relacionados	-

Tabla 40: Requisito funcional RNF-02.

RNF-03	
Titulo	El sistema guardará la información del usuario en el dispositivo.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	BAJA
Estabilidad	MEDIA
Descripción	Todos los datos que el usuario guarde, serán guardados únicamente en el dispositivo.
Req. relacionados	RNF-04

Tabla 41: Requisito funcional RNF-03.

RNF-04	
Titulo	Los datos del usuario no serán enviados fuera del dispositivo.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	MEDIA
Estabilidad	MEDIA
Descripción	La aplicación manejará los datos del usuario únicamente en el contexto del dispositivo.
Req. relacionados	RF-03

Tabla 42: Requisito funcional RNF-04.

RNF-05	
Titulo	La aplicación no compartirá datos con otras aplicaciones del dispositivo.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	MEDIA
Estabilidad	ALTA
Descripción	Los datos del usuario que maneje la aplicación estarán solo en el contexto de la aplicación.
Req. relacionados	RF-03

Tabla 43: Requisito funcional RNF-05.

RNF-06	
Titulo	Los dispositivos deben tener altavoz.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación hará necesitará hacer uso de dispositivos de salida de audio.
Req. relacionados	-

Tabla 44: Requisito funcional RNF-06.

RNF-07	
Titulo	Los dispositivos deberán tener cámara.
Prioridad	ALTA
Necesidad	MEDIA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	Para obtener imágenes, el dispositivo deberá tener un dispositivo de obtención de fotografías
Req. relacionados	-

Tabla 45: Requisito funcional RNF-07.

RNF-08	
Titulo	La aplicación será compatible con todas las versiones de iOS desde iOS 7.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación tendrá un correcto funcionamiento en sistemas operativos iOS a partir de la versión 7.
Req. relacionados	-

Tabla 46: Requisito funcional RNF-08.

RNF-09	
Titulo	La aplicación será compatible con todas las versiones de Android desde Android 4.4.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación tendrá un correcto funcionamiento en sistemas operativos Android a partir de la versión 4.4.
Req. relacionados	-

Tabla 47: Requisito funcional RNF-09.

RNF-10	
Titulo	La aplicación no necesitará conexión a internet para su utilización.
Prioridad	ALTA
Necesidad	MEDIA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	La aplicación funcionará de igual forma tanto con conexión a internet como sin ella.
Req. relacionados	-

Tabla 48: Requisito funcional RNF-10.

RNF-11	
Titulo	La aplicación tendrá plena disponibilidad para los usuarios.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	Los usuarios podrán utilizar la aplicación en cualquier momento y en cualquier lugar.
Req. relacionados	-

Tabla 49: Requisito funcional RNF-11.

RNF-12	
Titulo	La interfaz de usuario será hecha para aplicaciones móviles.
Prioridad	ALTA
Necesidad	ALTA
Verificabilidad	ALTA
Estabilidad	ALTA
Descripción	El desarrollo de la interfaz gráfica se ajustará a los cánones de desarrollo para aplicaciones móviles.
Req. relacionados	-

Tabla 50: Requisito funcional RNF-12.

3.3. Matriz de trazabilidad de requisitos funcionales y casos de uso

En este apartado revisaremos cual es la relación entre los casos de uso y los requisitos funcionales. Con ello, podremos revisar que todos los casos de uso se mapean a requisitos que, posteriormente, serán implementados en el **capítulo 4 – Implementación** de este documento.

En la siguiente matriz podemos revisar que las funciones especificadas en los casos de uso se relacionan con ciertos requisitos funcionales:

RF/CU	CU-01	CU-02	CU-03	CU-04	CU-05	CU-06	CU-07	CU-08	CU-09	CU-10	CU-11	CU-12	CU-13
RF-01	■												
RF-02	■												
RF-03	■												
RF-04		■											
RF-05		■											
RF-06		■											
RF-07			■										
RF-08				■									
RF-09					■								■
RF-10						■							
RF-11						■							
RF-12							■						
RF-13							■						
RF-14								■					
RF-15								■					
RF-16									■				
RF-17										■			
RF-18											■		
RF-19											■		
RF-20												■	
RF-21												■	
RF-22												■	

Tabla 51: Matriz de trazabilidad de requisitos funcionales y casos de uso.

Esta relación indica que todos los casos de uso se mapean con al menos un requisito funcional. En el caso de los casos de uso CU-01 y CU-02 podemos apreciar que existe una relación de 1 a 3, esto se debe a que los casos de uso implican más de un proceso y a su vez los requisitos deben cumplir el *principio de atomicidad*.

4. Diseño

En los siguientes apartados se detallarán y explicarán las distintas decisiones que se han tomado para el diseño de la solución propuesta, así como las tecnologías empleadas en la misma.

4.1. Diseño de la arquitectura del sistema

Dado que las soluciones basadas en arquitecturas para la realización de aplicaciones móviles emplean un modelo vista-controlador (MVC) (6), la solución propuesta tendrá este tipo de arquitectura:

Ilustración 4: imagen modelo-vista-controlador.

Modelo: esta capa se encarga de almacenar y manejar las peticiones de gestión de los datos que va a disponer la aplicación. En esta solución particular los datos almacenados estarán en el mismo dispositivo y serán los datos relacionados con el usuario. La petición de estos datos se realizará desde el controlador, por lo que la capa del modelo estará conectada con la capa del controlador.

Controlador: esta capa hará de intermediaria entre las capas de modelo y vista. Se encargará de realizar peticiones a la capa de almacenamiento (modelo) y de manipular y gestionar los datos obtenidos para que puedan ser utilizados por la capa "vista".

Vista: este elemento de la arquitectura es el responsable de obtener los datos que le brinda el controlador y ofrecerlos de manera visual al usuario. En este caso modifica la parte gráfica de la pantalla de la aplicación.

4.2. Elección de las tecnologías para el desarrollo de la solución deseada

Dado que el desarrollo de la solución propuesta corresponde a la realización de una aplicación móvil, la elección de las tecnologías de desarrollo se explicará bajo esta premisa, indicando en cada uno de ellas el motivo de su elección y que implica en el desarrollo del proyecto.

1. Ionic 2: este "framework" que combina el marco de desarrollo web "Angular" y "Apache cordova", para implementar las funciones nativas en los dispositivos móviles, permite la realización de aplicaciones hibridas para sistemas Android e iOS, además que su código, al ser angular, permite migrar la aplicación a un proyecto web fácilmente (7).
2. Angular: es un marco de desarrollo de ECMAScript que permite el desarrollo de aplicaciones web SPA ("single page application") que permiten que los componentes web ("CSS", "HTML" y "JavaScript") se descarguen una sola vez. Tiene una arquitectura modelo-vista-controlador, que permiten que el desarrollo y la realización de pruebas sea forma más simple (8).

3. Apache cordova: es un marco de desarrollo que permite utilizar las tecnologías web como HTML5, CSS3 y JavaScript para la realización de aplicaciones multiplataforma, sin necesidad de emplear el lenguaje nativo de cada plataforma (9).
4. NPM ("node package manager"): es un gestor de paquetes para marcos de desarrollo de JavaScript. Permite gestionar el uso de dependencias y librerías de terceros (10).

4.3. Determinación del entorno operacional

En este apartado definiremos cuáles serán los entornos operacionales que se utilizarán en el proyecto. Explicaremos detalladamente cual será la función de cada uno de ellos y explicaremos sus principales características.

Entorno de desarrollo: definiremos como entorno de desarrollo el entorno de trabajo local que permitirá desarrollar la implementación de las distintas capas del sistema, siendo un entorno de trabajo orientado al desarrollo, pruebas y posterior despliegue en el "entorno de producción"

En este entorno contamos con máquinas físicas y usaremos las mismas tecnologías que en el entorno de producción. Se enfocará al desarrollo y las pruebas de funcionamiento, pero siempre de cara al posterior despliegue del sistema en el entorno de producción en función de sus distintas fases.

En el entorno de desarrollo contaremos con las siguientes herramientas:

Especificaciones técnicas del equipo de desarrollo	
Sistema Operativo	Apple Macbook Pro Retina 13 - Mid 2017
Procesador	Intel Core I5 - 2,3ghz, 4 Núcleos - 8 Hilos
Memoria RAM	8GB DDR4L
Disco Duro	128GB SSD
Conectividad	Bluetooth, Wifi A/B/G/N/AC

IDE de desarrollo	
IDE de desarrollo	Visual Studio Code
Coste de la licencia	Licencia MIT, de uso gratuito.

Periféricos y material para el desarrollo	
Terminales de prueba	iPhone SE y bq Aquaris E4.5
Impresora	Impresora HP Laserjet 1018n
Monitor	Monitor Samsung SyncMaster EX1920
Acceso a internet	Vodafone - Fibra Óptica 50MB
Material de oficina	Rotuladores, pizarra blanca, folios, bolígrafos.

Tabla 52: Equipamiento físico del entorno de desarrollo.

Con estas herramientas podremos empezar a desarrollar utilizando un entorno virtual de trabajo que nos permita simular el entorno de producción. Dado que el equipo de desarrollo tiene un sistema operativo distinto al que usaremos en producción tendremos que adaptarlo a las necesidades del proyecto.

Entorno de producción: definiremos como entorno de producción el entorno en el cual se desplegará el desarrollo hecho y probado del entorno de desarrollo. De esta forma podremos desacoplar las funciones no terminadas, el desarrollo no probado y los errores que esto conlleva del entorno que se encuentra en estado "operativo".

Debemos tomar en cuenta que el entorno de producción genera un coste en función de las horas que esta *online*, por lo que se hace necesario contar con un mecanismo que permita apagar todo el entorno de producción en los momentos en que no se esté usando, disminuyendo así el consumo innecesario de recursos del proyecto.

Al tratarse de un entorno *IaaS*, el entorno de producción puede modificarse y administrarse dinámicamente, lo que permite adecuarse a las necesidades del proyecto de forma rápida y sencilla. Por este motivo no definiremos los elementos del mismo hasta el punto 5.2 del **capítulo 5 – Implementación e implantación**.

4.4. Diseño de la capa de persistencia

La persistencia de los datos, como se ha indicado antes, se hará en el propio dispositivo. Dada la simpleza y las opciones que nos permite ECMAScript6 de manejar objetos de tipo JSON (11), se ha escogido este formato para poder guardar los distintos datos en el dispositivo. La estructura de estos objetos será la siguiente:

- Para las canciones se tendrá la siguiente estructura a persistir:

```
{
        userName : "userName",
        songName: "songName",
        chords: [
                {
                        name: "name",
                        time: "time"
                },
                {
                        name: "name2",
                        time: "time2"
                }
        ]

}
```

Ilustración 5: Estructura canciones.

Donde el campo "userName" corresponderá al usuario que ha hecho la grabación, "songName" será el título de la canción y por último una lista de acordes que contendrán cada uno un nombre correspondiente al acorde y un tiempo que corresponde al tiempo de la grabación donde es tocado.

- Para los usuarios la estructura será la siguiente:

```
{
        userName: “userName”,
        name: “name”,
        email: “email”,
        password: “password”
}
```

Ilustración 6: Estructura usuarios.

Donde el “userName” es el nombre del usuario, “name” es el nombre del usuario, “email” es el email del usuario y por último el “password” que se guarda de forma encriptada.

4.5. Diseño de la interfaz gráfica

En este apartado se hará un análisis y se mostrará la interfaz gráfica correspondiente a la solución propuesta. Se hará énfasis en las buenas prácticas de maquetación para dispositivos móviles y en los cánones de desarrollo para aplicaciones multiplataforma (12).

Cada pantalla será respuesta de los diferentes casos de uso, previamente analizados y definidos, y del desarrollo de aplicaciones para dispositivos móviles.

1. "Log-in": la pantalla tendrá dos campos de texto para introducir nombre de usuario "username" y la contraseña "password". Por otra parte tendrá una opción seleccionable para que el usuario pueda recordar las credenciales de los campos de texto. Por último, contendrá dos botones, uno para el realizar la validación de los datos y acceder a la aplicación y otro para acceder a la pantalla de registro. La estructura será la siguiente:

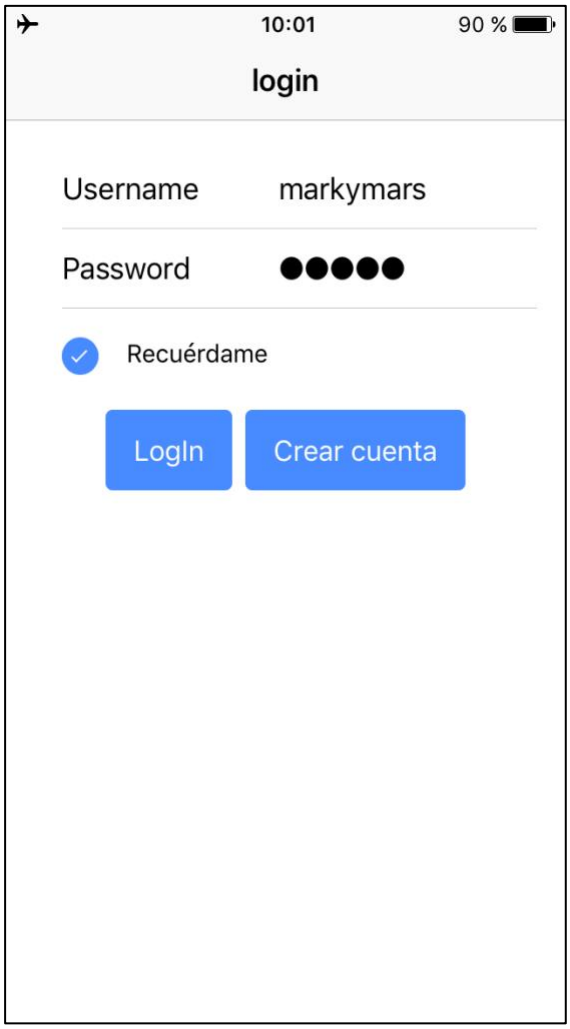


Ilustración 7: pantalla de log-in.

2. Registro: en esta pantalla se mostrarán cinco campos de texto correspondientes a los datos de registro del usuario (nombre de usuario, nombre, Email, password y la validación del password). Por otra parte contendrá un botón que permitirá el registro del usuario. La estructura es la siguiente:

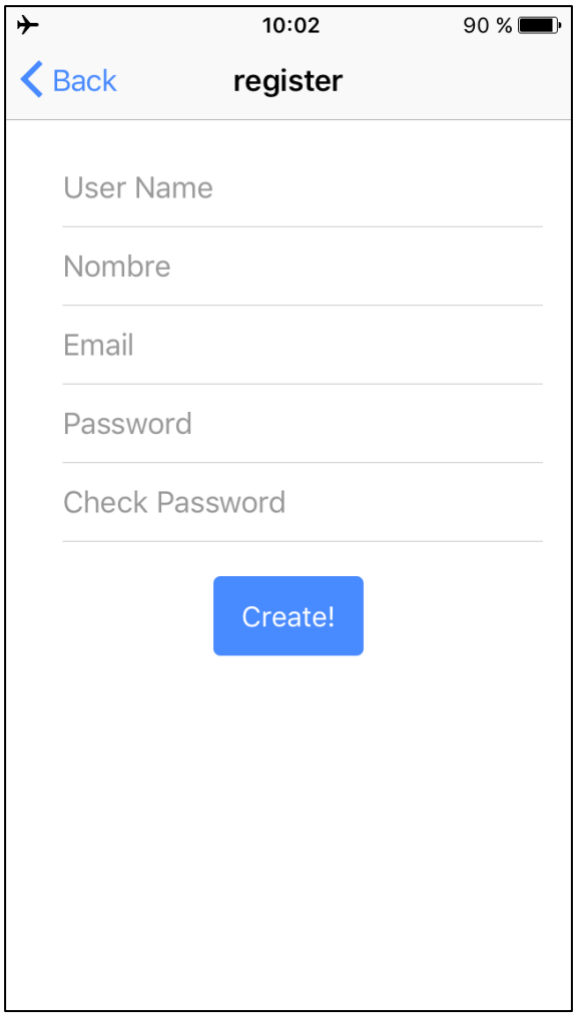


Ilustración 8: pantalla de registro.

3. “Home”: La pantalla de perfil o “home” contendrá la información de perfil del usuario que previamente se registró (nombre de usuario, nombre y e-mail). Por otra parte contendrá una lista con las grabaciones que el usuario haya guardado. Cada grabación de la lista contendrá el título de la misma junto con dos botones para reproducir y detener la grabación. Por último, tendrá un campo en el que se pueda añadir una imagen de usuario. La estructura será la siguiente:

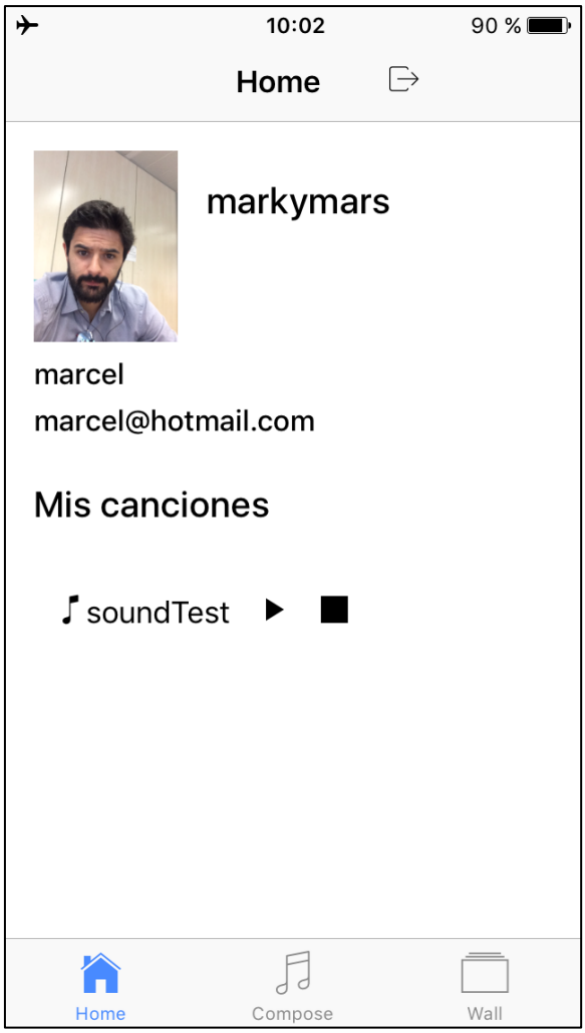


Ilustración 9: pantalla “home”.

4. Teoría musical: la pantalla que mostrará la teoría musical se compondrá de una serie de “slides” o deslizables, en los que el usuario podrá ir navegando entre ellos.

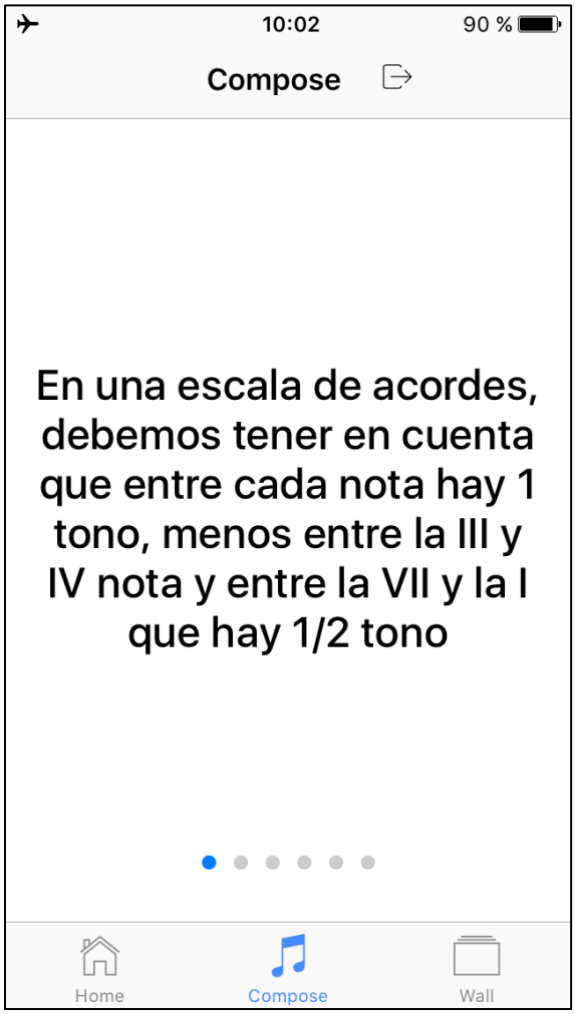


Ilustración 10: pantalla de teoría 1.

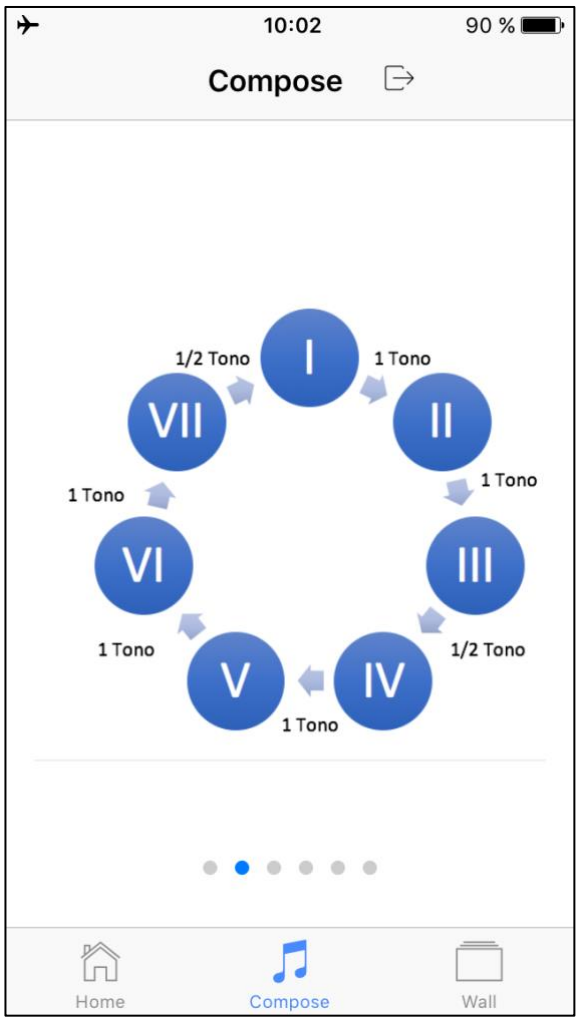


Ilustración 11: pantalla de teoría 2.

5. Composición: en la pantalla de composición, se mostrarán los distintos acordes de la escala de Do mayor para poder reproducirlos, un elemento que permita la grabación y otro que permita reproducir la grabación. Por último tendrá un botón con el que se pueda agregar la grabación a la lista de grabaciones guardadas. La estructura será la siguiente:

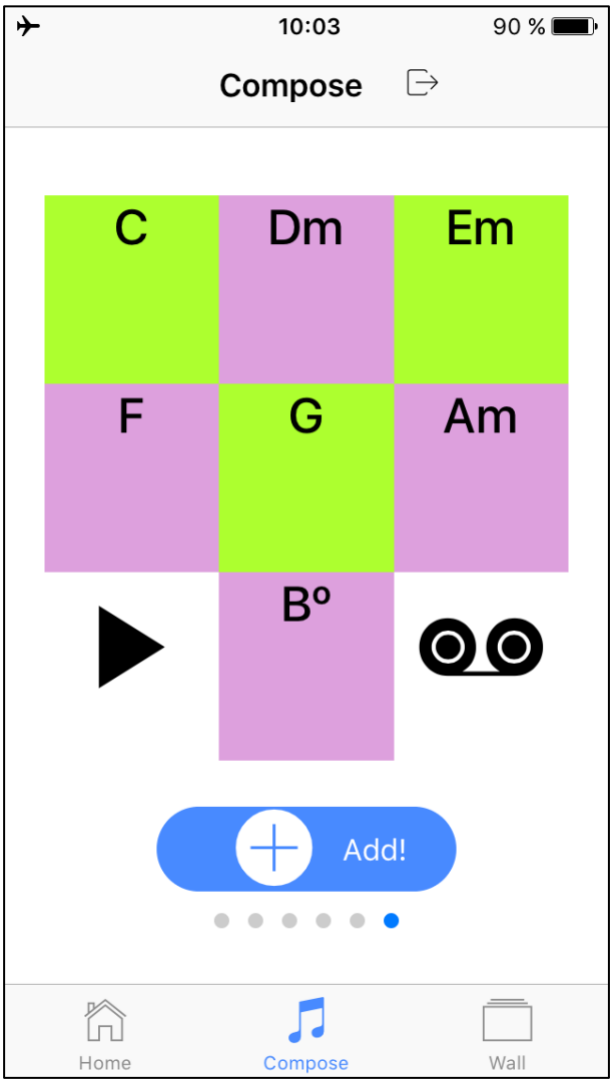


Ilustración 12: pantalla de acordes.

6. “Wall”: en esta pantalla se mostrarán todas las grabaciones que los distintos usuarios de la aplicación, en el mismo dispositivo, hayan guardado. Cada elemento de la lista tendrá una estructura de “carta” que se compondrá de la foto del usuario, el título de la grabación y un par de botones para reproducir o detener la grabación.

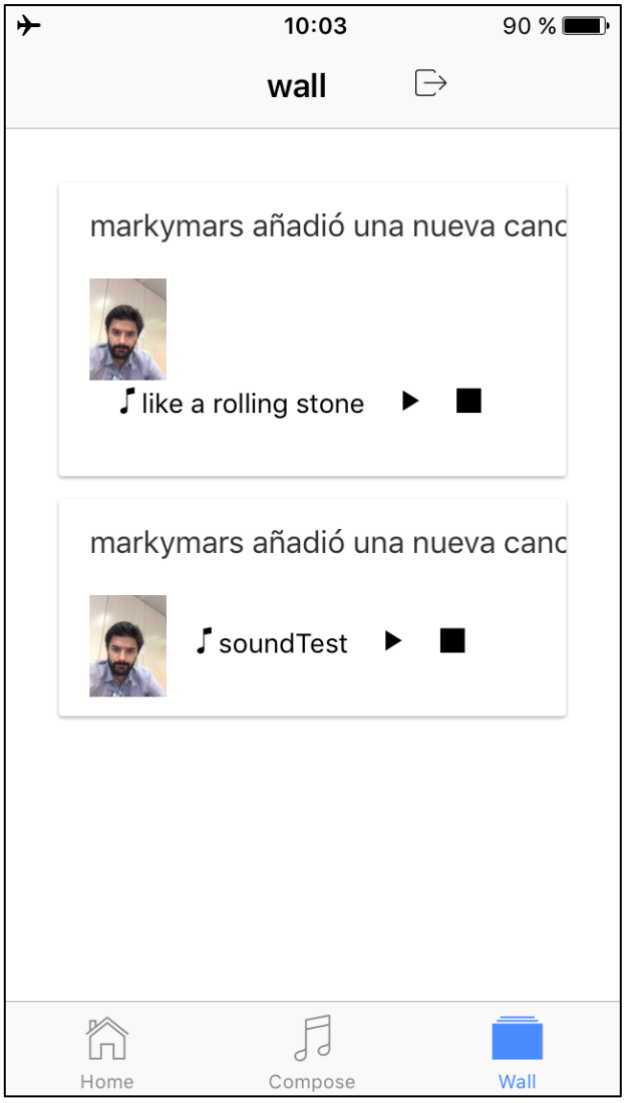


Ilustración 13: pantalla de “Wall”.

5. Implementación e implantación

A lo largo de este punto se detallará la instalación y configuración del entorno de desarrollo con el que se va a llevar a cabo la realización de la implementación de la solución propuesta.

5.1. Instalación de un manejador de paquetes

Para realizar la instalación de las diversas dependencias que va a necesitar el proyecto, se ha decidido utilizar el NPM(“node package manager”). Para su instalación solo basta con instalar NodeJS en nuestra versión, la “v6.10.3” (13) con el que también se instalará el NPM en la versión 3.10.10 (14).

5.2. Instalación de cada entorno de desarrollo

Para llevar a cabo la implementación de la solución se utilizan principalmente dos IDE’s(Integrated Development Environment): el Visual Studio Code (15) por una parte, para el desarrollo del proyecto y el Xcode (16) para realizar las “build” para iOS.

- Visual Studio code: la instalación de esta herramienta, se podrá realizar directamente desde la página web para Visual Studio. Por otra parte siguiendo las buenas prácticas para proyectos basados en ECMAScript 6 escrito con TypeScript, instalaremos algunos “plugins” para Visual Studio Code como el ESLint (17), programa que permite analizar el código en tiempo de desarrollo para avisar de posibles errores.

- Xcode: para la instalación de este IDE, desde el MacOs vamos a la App Store e instalamos la aplicación.

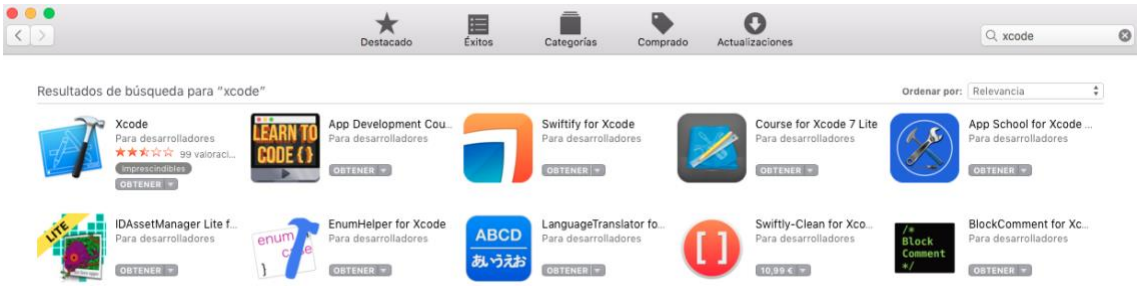


Ilustración 14: App Store.

5.3. Instalación de dependencias

La instalación de dependencias que necesitaremos para el proyecto, se hará como se ha comentado previamente, a través del gestor de paquetes NPM. Todas las dependencias se podrán ver en el archivo “package.json” del proyecto.

Cada instalación del proyecto se realizará situándonos en la carpeta del mismo ejecutando desde terminal el siguiente comando:

npm install <<nombre de la dependencia>> -dev

La directiva “-dev” hace referencia a la instalación de la dependencia directamente en el proyecto y no globalmente en el sistema. Todas las dependencias instaladas se guardarán en la carpeta “node_modules” y para instalar todas las que estén especificadas en el “package.json” solamente habría que introducir el comando.

```
npm install
```

La lista de todas las dependencias que se van a utilizar para el desarrollo de este proyecto es la siguiente:

Ilustración 15: dependencias del proyecto.

5.4. Implementación

Cada desarrollo que se vaya haciendo del código se irá subiendo a un repositorio en la nube con la herramienta para el control de código Bitucket (18). Cada desarrollo se hará siguiendo las buenas prácticas para este tipo de control de código, es decir, se hará una rama en la herramienta por cada funcionalidad del código para posteriormente pasar a una rama de integración de funcionalidades en donde se mezclarán los desarrollos de cada rama (19). En esta rama se harán pruebas de forma que la aplicación quede lista para pasarse a la rama master, cuyo contenido será el que se compilará para producción. La dirección del repositorio donde se almacenará el código es la siguiente:

https://MaqubeX@bitbucket.org/MaqubeX/tfg_marcel_bertagnini.git

Por otra parte, la arquitectura del esqueleto de la aplicación corresponderá a las estructuras propias de aplicaciones que contienen elementos “componentizados”, es decir, una serie de elementos de código agrupados por funcionalidad y cada grupo siendo independiente el uno del otro (20).

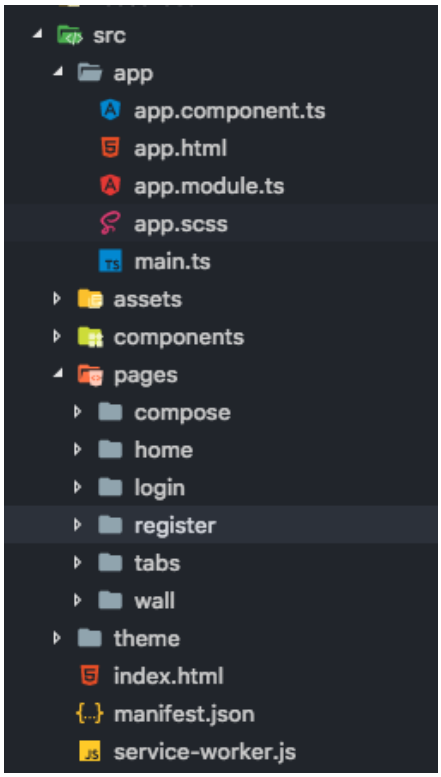


Ilustración 16: esqueleto del proyecto.

En este caso se agruparán las funcionalidades por cada capa de la aplicación, es decir, por cada página de la aplicación. Cada agrupación será independiente la una de la otra y todas estarán contenida en el módulo “app” que es el módulo madre de este proyecto.

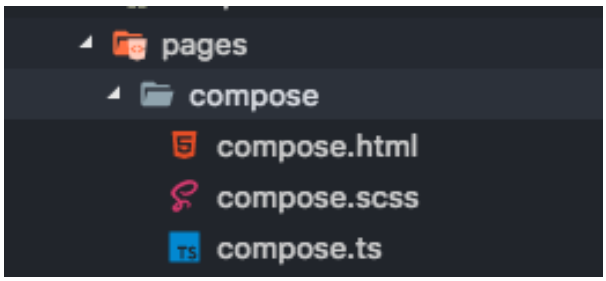


Ilustración 17: esqueleto de componente.

Cada agrupación de funcionalidad será delimitada por cada pantalla de la aplicación, cuyos elementos se hallarán en una carpeta que contendrán los tres archivos fundamentales de cada pantalla:

- Html(HyperText Markup Language): cada archivo con la extensión ".html” contendrá el código de marcado de la pantalla (dado que es un archivo “.html” corresponderá a una página web).

- Scss(Sassy Cascading Stylesheets): estos archivos con extensión ".scss" contendrán los estilos gráficos que utilizarán los elemento definidos el ".html".

- Ts(TypeScript): Contiene el código que contiene la lógica de los elementos definidos en el ".html". En estos archivos se importarán todas las librerías necesarias para utilizar las distintas funciones de la pantalla. También, en este archivo, se definirá que el conjunto de los elementos de esta pantalla, constituirán un "componente".

```
@Component({
  selector: 'page-compose',
  templateUrl: 'compose.html'
})
```

Ilustración 18: definición de componente.

5.5. Implantación

El proceso de implantación para definir el entorno de producción de la solución propuesto se centrará en dos vías paralelas que difieren en el sistema operativo en el que se realice la puesta en producción: por una parte, se tendrá que generar una edición de la aplicación para distribución para iOS de forma que pueda ser publicada en la "App Store" y por otra parte una edición de distribución para el sistema operativo Android para publicarse en el "Google Play".

1. **Implantación para iOS:** para poder integrar una aplicación en la "App Store" necesitamos una cuenta en el sistema de "iTunnes Connect" (21). Este sistema permite a los desarrolladores gestionar la distribución de las aplicaciones en la "App Store" como distribuir también versiones beta de alguna app. El procedimiento de subir la aplicación consta de varios pasos:

    a. Se crea una archivo ".ipa" para poder añadirlo a la plataforma "iTunes connect". Para ello, se accede a Xcode y en el menú "product" se elige la opción "Destination" y luego se elige la opción "Generic iOS Device".

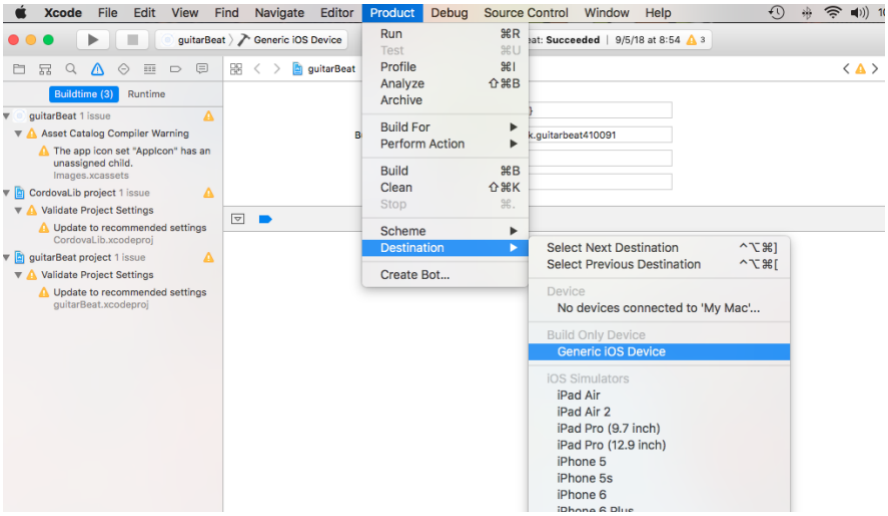


Ilustración 19: generación de ".ipa" 1.

Luego se activa la opción “Archive” dentro del menú “Product” para posteriormente exportar el archivo binario como un “.ipa” firmado por el usuario desarrollador de Apple.

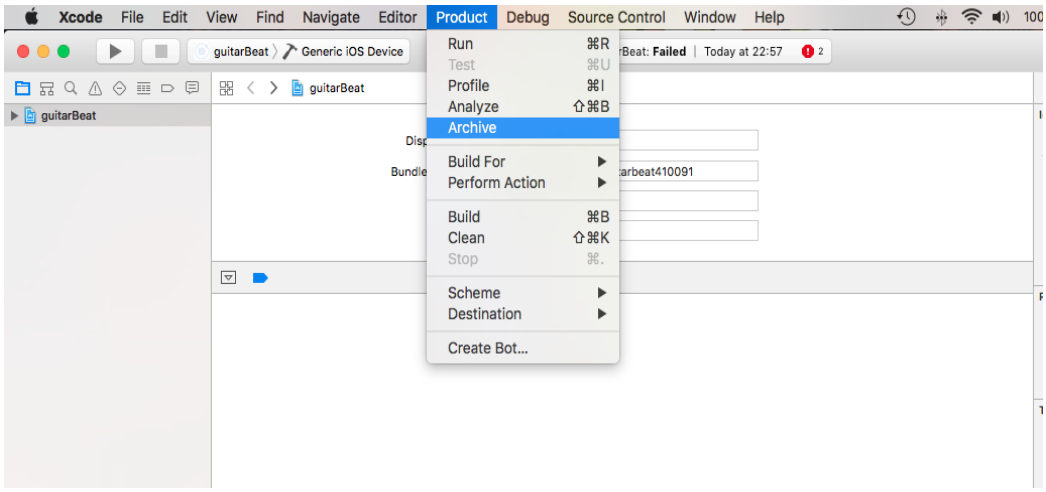


Ilustración 20: generación de “.ipa” 2.

b. Se accede a la plataforma “iTunnes Connect” con el usuario desarrollador de Apple y se selecciona “Apps” como contenido.

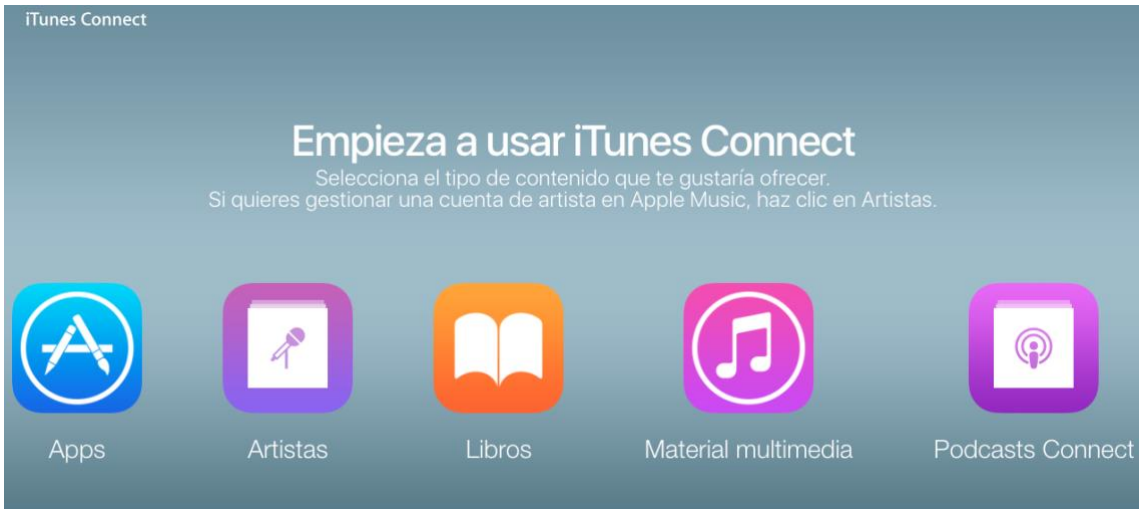


Ilustración 21: iTunes Connect.

c. Nos dirigimos al apartado de mis aplicaciones, seleccionamos la aplicación y le damos a la opción de “Submit for Review”.
d. Una vez pasada la revisión le damos a “Submit” para subir la aplicación a la “App Store” (22).

El proceso de publicación de la app queda resumido en el siguiente esquema:

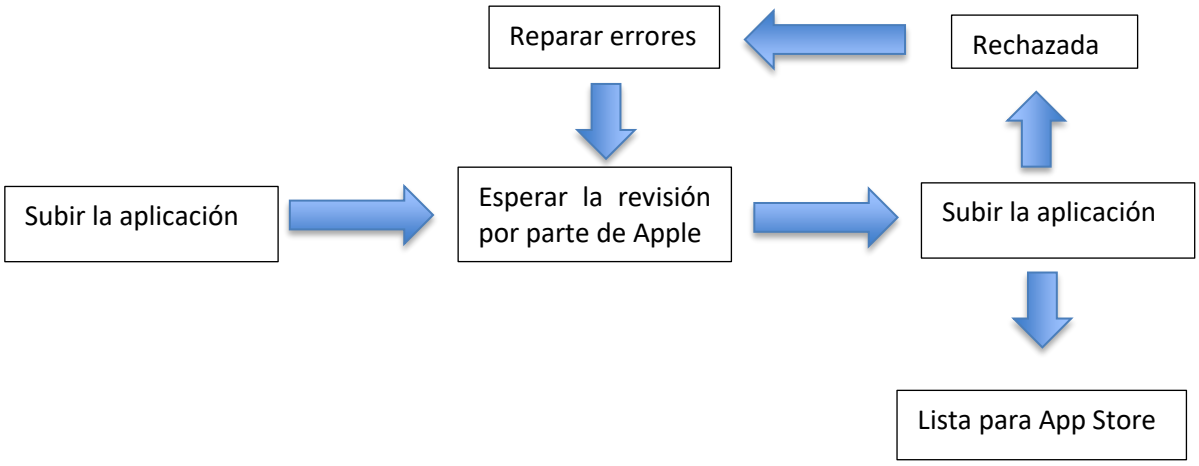


Ilustración 22: ciclo de publicación de apps en iOS.

2. **Implantación para Android** (23): En primer lugar se generará una archivo ".apk", a partir del proyecto. Para ello hay que añadir la plataforma Android al proyecto con el siguiente comando:

```
Ionic add platform android
```

Una vez creada la plataforma para Android se generará la ".apk" a través del siguiente comando:

```
ionic build Android
```

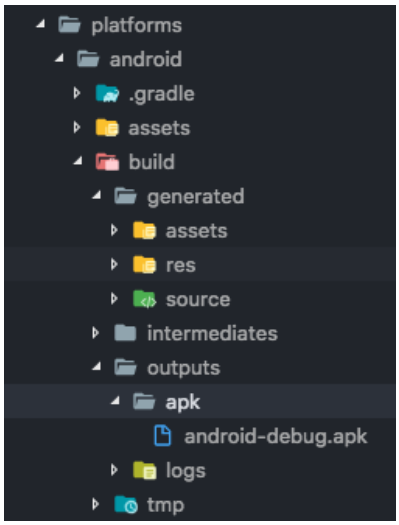


Ilustración 23: apk de Android.

Esto generará un ".apk" sin firmar. Para generar una clave privada para poder firmar el ".apk", se utilizará el siguiente comando:

```
keytool -genkey -v -keystore key-android.keystore -alias marcel -keyalg RSA -keysize 2048 -validity 10000
```

Tras ejecutar este comando, se obtendrá la clave para poder firmar el ".apk". Para ello utilizamos el siguiente comando:

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore key-android.keystore
android-debug.apk marcel
```

Con la ".apk" firmada, se accede a la "Google Play Developer Console" con las credenciales de desarrollador Android, se añade la nueva aplicación que se ha generado, seleccionando el menú "Tus aplicaciones" añadimos la ".apk" y subimos la aplicación.

6. Pruebas

6.1. Especificación del plan de pruebas

El plan de pruebas pretende verificar que el sistema cumple con las necesidades de este proyecto, por ello debemos relacionar cada prueba con aquellos requisitos funcionales que pretende contrastar.

El diseño del plan de pruebas atenderá al siguiente tipo de pruebas, habituales en proyectos similares:

- **Pruebas unitarias**: testearán el funcionamiento de cada componente del código verificando su correcto funcionamiento de forma aislada. La tecnología que se utilizará para llevarlas a cabo será el framework de pruebas "karma-jasmine" (24).
- **Pruebas de integración:** verificarán que los componentes de la aplicación funcionan correctamente de manera conjunta.
- **Pruebas de fin a fin ("end to end"):** estas pruebas verificarán que la solución cumple con los requisitos que se han definido de forma satisfactoria. La tecnología que se utilizará para llevarlas a cabo será el framework de pruebas "Selenium" (25).

Para cumplir esta función debemos definir una plantilla que nos permita registrar la prueba:

Identificador	
Objetivos	
Necesidades	
Entradas	
Secuencia	
Salida	
Requisitos relacionados	

Tabla 53: Plantilla de pruebas.

Donde los elementos de la plantilla son los siguientes:

- Identificador: permite diferenciar unívocamente la prueba. Tendrá el formato PY-NN, donde el formato indica lo siguiente:
  Y: indica el tipo de prueba:
  - U: prueba unitaria.
  - I: prueba de integración
  - E2E: prueba "end to end"

  NN: indica el número correlativo de la prueba que es unívoco.
- Objetivos: indica la finalidad de la prueba, el fin de la realización de la misma.
- Necesidades: prerrequisitos que son necesarios para la realización de la prueba.
- Entradas: valores de entrada necesarios para la realización de la prueba.

- Secuencia: pasos necesarios para que la prueba pueda realizarse. No aplica para las pruebas unitarias.
- Salidas: valores de salida o resultado de la realización de la prueba.
- Requisitos relacionados: requisitos que se corresponden con la prueba y que son contrastados al realizar la misma.

Finalmente, las pruebas que integran el plan de pruebas son las siguientes:

PU-01	
Objetivo	Comprobar que se obtiene el nombre de usuario al introducirlo.
Necesidades	El componente debe estar operativo.
Entradas	Nombre del usuario.
Salida	Cadena de caracteres.
Requisitos relacionados	RF-04, RF-05, RF-06

Tabla 54: Prueba PS-01.

PU-02	
Objetivo	Comprobar que se obtiene la contraseña al ser introducida.
Necesidades	El componente debe estar operativo.
Entradas	Contraseña del usuario.
Salida	Cadena de caracteres.
Requisitos relacionados	RF-04, RF-05, RF-06

Tabla 55: Prueba PS-01.

PU-03	
Objetivo	Comprobar que al introducir al contraseñas no se pinta su texto en claro.
Necesidades	El componente debe estar operativo.
Entradas	Contraseña del usuario.
Salida	La contraseña con puntos en lugar de en claro.
Requisitos relacionados	RF-04, RF-05, RF-06

Tabla 56: Prueba PS-01.

PU-04	
Objetivo	Comprobar el check “recuérdame”
Necesidades	El componente debe estar operativo.
Entradas	Gesto de seleccionar el check.
Salida	True.
Requisitos relacionados	RF-04, RF-08

Tabla 57: Prueba PS-01.

PU-05	
Objetivo	Comprobar nombre en el registro.
Necesidades	El componente debe estar operativo.
Entradas	Nombre del usuario.
Salida	Cadena de caracteres correspondiente al nombre del usuario.
Requisitos relacionados	RF-01, RF-02, RF-03

Tabla 58: Prueba PS-01.

PU-06	
Objetivo	Comprobar el email en el registro.
Necesidades	El componente debe estar operativo.
Entradas	Email del usuario.
Salida	Cadena de caracteres correspondiente al email del usuario.
Requisitos relacionados	RF-01, RF-02, RF-03

Tabla 59: Prueba PS-01.

PU-07	
Objetivo	Comprobar foto de perfil.
Necesidades	El componente debe estar operativo.
Entradas	Foto de perfil.
Salida	Al realizar o elegir la imagen el sistema obtiene una imagen válida.
Requisitos relacionados	RF-09, RF-10, RF-11

Tabla 60: Prueba PS-01.

PU-08	
Objetivo	Comprobar el botón de grabar.
Necesidades	El componente debe estar operativo.
Entradas	Gesto de pulsar el botón.
Salida	Al pulsar el botón se hace la llamada al método “record()”.
Requisitos relacionados	RF-12, RF-16

Tabla 61: Prueba PS-01.

PU-09	
Objetivo	Comprobar los botones correspondientes a los acordes.
Necesidades	El componente debe estar operativo.
Entradas	Gesto de pulsar los botones correspondientes a los acordes.
Salida	Al pulsar el botón se hace la llamada al método “playChord()”.
Requisitos relacionados	RF-12, RF-13, RF-14, RF-15

Tabla 62: Prueba PS-01.

PU-10	
Objetivo	Comprobar añadir grabación.
Necesidades	El componente debe estar operativo.
Entradas	Gesto de pulsar el botón de añadir
Salida	Al pulsar el botón se hace la llamada al método “showPrompt()” que generará un pop-up con la información necesaria para añadir la grabación.
Requisitos relacionados	RF-12, RF-14, RF-16

Tabla 63: Prueba PS-01.

PU-11	
Objetivo	Comprobar grabación.
Necesidades	El componente debe estar operativo.
Entradas	Se debe haber hecho una grabación.
Salida	Se hace una llamada al método “playSong()” que reproducirá el audio previamente grabado.
Requisitos relacionados	RF-12, RF-16, RF-17, RF-19

Tabla 64: Prueba PS-01.

PU-12	
Objetivo	Comprobar el título de la grabación.
Necesidades	El componente debe estar operativo.
Entradas	Introducir el nombre de la grabación.
Salida	Se obtiene una cadena de caracteres correspondiente al título de la grabación.
Requisitos relacionados	RF-12, RF-18, RF-19

Tabla 65: Prueba PS-01.

PU-13	
Objetivo	Comprobar reproducción de canciones en el “muro”.
Necesidades	El componente debe estar operativo.
Entradas	Pulsar el botón de reproducción en la página de “Wall”.
Salida	Se llama al método “play(chords)” que se encarga de reproducir las grabaciones en la página de “Wall”.
Requisitos relacionados	RF-20, RF21

Tabla 66: Prueba PS-01.

PU-14	
Objetivo	Comprobar pausa de canciones en el “muro”.
Necesidades	El componente debe estar operativo.
Entradas	Pulsar el botón de reproducción en la página de “Wall”.
Salida	Se llama al método “stop()” que se encarga de pausar las grabaciones en la página de “Wall”.
Requisitos relacionados	RF-20, RF-21, RF-22

Tabla 67: Prueba PS-01.

PI-01	
Objetivo	Comprobar que al acceder a la página de “Wall”, aparecen todas las canciones de todos los usuarios.
Necesidades	La aplicación debe estar completamente operativa para los test.
Entradas	Ninguna.
Secuencia	Realizamos log-in con un usuario de prueba. Accedemos a la página de “Wall”. Comprobamos que están todas las canciones de cada usuario que se han grabado.
Salida	Están todas las grabaciones que se han realizado.
Requisitos relacionados	RF-04, RF-20

Tabla 68: Prueba PS-01.

PI-02	
Objetivos	Comprobar que al acceder a la página de “home” aparecen todas las canciones del usuario.
Necesidades	La aplicación debe estar completamente operativa para los test.
Entradas	Ninguna.
Secuencia	Realizamos log-in con un usuario de prueba. Accedemos a la página de “Home”. Comprobamos que están todas las canciones del usuario que se han grabado.
Salida	Están todas las grabaciones que el usuario a grabado y guardado.
Requisitos relacionados	RF-04, RF-09

Tabla 69: Prueba PS-01.

PI-03	
Objetivos	Comprobar que el usuario de la página de “home” corresponde al mismo usuario que el introducido en la pantalla de “log-in”.
Necesidades	La aplicación debe estar completamente operativa para los test.
Entradas	Ninguna.
Secuencia	Realizamos log-in con un usuario de prueba. Accedemos a la página de “Home”. Comprobamos que los datos del usuario de prueba corresponden con el usuario introducido para el log-in.
Salida	Ninguna.
Requisitos relacionados	RF-03,RF-04, RF-09

Tabla 70: Prueba PS-01.

PI-04	
Objetivos	Comprobar que al añadir una nueva grabación, ésta se añade también a la lista de canciones que tiene el usuario en su pantalla de “home”.
Necesidades	La aplicación debe estar completamente operativa para los test.
Entradas	Ninguna.
Secuencia	Se realiza el log-in con un usuario de prueba. Se accede a la página de “compose”. Se comprueba la teoría musical. Se realiza una grabación. Se guarda la grabación. Se accede a la pantalla de “home”. Se comprueba que la nueva grabación ha sido añadida a las grabaciones del usuario.
Salida	Ninguna.
Requisitos relacionados	RF-04, RF-09, RF-12, RF-13, RF14, RF-16, RF-19

Tabla 71: Prueba PS-01.

PI-05	
Objetivos	Comprobar que al añadir una canción, ésta se añade en la pantalla de “Wall”.
Necesidades	La aplicación debe estar completamente operativa para los test.
Entradas	Ninguna.
Secuencia	Se realiza el log-in con un usuario de prueba. Se accede a la página de “compose”. Se comprueba la teoría musical. Se realiza una grabación. Se guarda la grabación. Se accede a la pantalla de “Wall”. Se comprueba que la nueva grabación añadida se encuentra en la lista de grabaciones del “Wall”.
Salida	Ninguna.
Requisitos relacionados	RF-04, RF-12, RF-13, RF14, RF-16, RF-19, RF-20

Tabla 72: Prueba PS-01.

PI-06	
Objetivos	Comprobar que al registrar un usuario, éste puede acceder al log-in con sus credenciales.
Necesidades	La aplicación debe estar completamente operativa para los test.
Entradas	Ninguna.
Secuencia	Se accede al registro de usuario. Se introducen las credenciales del usuario. Se registra al usuario. Accedemos a la pantalla de log-in. Se introducen las credenciales del usuario previamente registrado. Accedemos al log-in.
Salida	Ninguna.
Requisitos relacionados	RF-01, RF-02, RF-03, RF-04, RF-05, RF-06

Tabla 73: Prueba PS-01.

PE2E-01	
Objetivos	Realizar una grabación y hacer log-out.
Necesidades	La aplicación debe estar completamente operativa para los test.
Entradas	Ninguna.
Secuencia	Se accede al registro de usuario. Se introducen las credenciales del usuario. Se registra al usuario. Accedemos a la pantalla de log-in. Se introducen las credenciales del usuario previamente registrado. Accedemos al log-in. Se accede a la página de “compose”. Se comprueba la teoría musical. Se realiza una grabación. Se guarda la grabación. Se hace log-out.
Salida	Ninguna.
Requisitos relacionados	RF-01, RF-02, RF-03, RF-04, RF-05, RF-06,RF-07, RF-12, RF-13, RF14, RF-16, RF-18, RF-19,

Tabla 74: Prueba PS-01.

6.2. Matriz de trazabilidad de requisitos y pruebas

En este apartado revisaremos el mapeo entre las pruebas del punto anterior y los requisitos funcionales del proyecto. Identificando cual es la relación entre ellos.

Como podemos ver a continuación, cada prueba se mapea con al menos un requisito funcional. Al mismo tiempo, todos los requisitos funcionales son probados al menos una vez.

Pruebas / RF	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22
PU-01				■	■	■																
PU-02				■	■	■																
PU-03				■	■	■																
PU-04				■				■														
PU-05	■	■	■																			
PU-06	■	■	■																			
PU-07									■	■	■											
PU-08												■				■						
PU-09												■	■	■	■							
PU-10												■		■		■						
PU-11												■				■	■		■			
PU-12												■						■	■			
PU-13																				■	■	
PU-14																				■	■	■
PI-01				■																■		
PI-02				■					■													
PI-03			■	■					■													
PI-04				■					■			■	■	■		■			■			
PI-05				■								■	■	■		■			■	■		
PI-06	■	■	■	■	■	■																
PE2E-01	■	■	■	■	■	■	■					■	■	■		■		■	■			

Tabla 75: Matriz de trazabilidad entre pruebas de sistema y requisitos funcionales.

6.3. Resultado de la ejecución de las pruebas de sistema

Siguiendo con el plan de pruebas, debemos mostrar los resultados de la ejecución del plan de pruebas sobre el entorno de pre-producción del sistema. Con esto pretendemos demostrar que los requisitos funcionales se cumplen y que las funciones esenciales definidas en el punto 2.3 de este documento se satisfacen.

Definiremos el resultado de la prueba como el resultado obtenido luego de seguir los pasos que se definen en el plan de pruebas en el entorno operacional del sistema, pudiéndose obtener 2 resultados distintos:

- **OK**: el sistema pasa la prueba, la funcionalidad implementada responde correctamente y la salida es satisfactoria.

- **ERROR**: el sistema no pasa la prueba, la funcionalidad implementada no responde correctamente y la salida es errónea.

Con ello, revisaremos los resultados obtenidos luego de la ejecución de las pruebas de sistema.

Prueba ejecutada	Resultado	Observaciones
PU-01	OK	Se obtiene una cadena de caracteres.
PU-02	OK	Se obtiene una cadena de caracteres.
PU-03	OK	No aparece la contraseña en claro.
PU-04	OK	Devuelve “true”.
PU-05	OK	Se obtiene una cadena de caracteres.
PU-06	OK	Se obtiene una cadena de caracteres.
PU-07	OK	Se obtiene una imagen válida.
PU-08	OK	Llama al método correspondiente.
PU-09	OK	Llama al método correspondiente.
PU-10	OK	Llama al método correspondiente.
PU-11	OK	Llama al método correspondiente.
PU-12	OK	Se obtiene una cadena de caracteres.
PU-13	OK	Llama al método correspondiente.
PU-14	OK	Llama al método correspondiente.
PI-01	OK	Se realiza la comprobación correctamente.
PI-02	OK	Se realiza la comprobación correctamente.
PI-03	OK	Se realiza la comprobación correctamente.
PI-04	OK	Se realiza la comprobación correctamente.
PI-05	OK	Se realiza la comprobación correctamente.
PI-06	OK	Se realiza la comprobación correctamente.
PE2E-01	OK	La secuencia de operaciones se realiza de forma correcta.

Tabla 76: Matriz de resultado de las pruebas del sistema.

7. Planificación y presupuesto

En este apartado revisaremos la perspectiva temporal y económica de este proyecto. La idea es poder exponer el cálculo detallado de los costes y beneficios obtenidos tomando en cuenta la programación estimada de las fases del mismo.

7.1. Planificación

La programación de este proyecto se estimó en un total de 310hrs, de las cuáles he descontado un 10% previsto para ajustes finales y margen de error de tiempo, dejando un total de 279hrs disponibles para el desarrollo completo.

Calculando una carga de 10hrs de trabajo más 5hrs de documentación a la semana, podemos calcular una carga total de trabajo de **15hrs a la semana**, por lo cual, podemos adjudicar una duración total del proyecto de **20 semanas y 10hrs.**

Por otro lado, la fecha de inicio de este proyecto será el día lunes 25 de septiembre del año 2017. Basándonos en esto se ha estimado que la fecha de finalización del proyecto será el día 14 de Febrero del año 2018.

Las fases que se desarrollarán a lo largo de este proyecto son las que se muestran a continuación:

- **Documentación**
- **Concepto del proyecto**
  - Esquematización de que se quiere desarrollar
  - Identificación de las necesidades del proyecto
- **Estudio de mercado**
  - Estudio de las posibles soluciones
  - Comparativa de las soluciones
- **Análisis**
  - Análisis de los casos de usos
  - Análisis de los requisitos
  - Análisis de la arquitectura general
- **Diseño**
  - Diseño de la arquitectura completa
  - Diseño de las capas
- **Implementación**
  - Implementación de la *API REST*
  - Implementación de los contenedores
- **Implantación**
  - Configuración del entorno operacional
  - Implantación en *AWS*
- **Evaluación**
  - Plan de pruebas
  - Ejecución del plan de pruebas

Viendo la siguiente carta Gantt, podemos hacer un seguimiento cronológico de lo ocurrido en el desarrollo del proyecto:

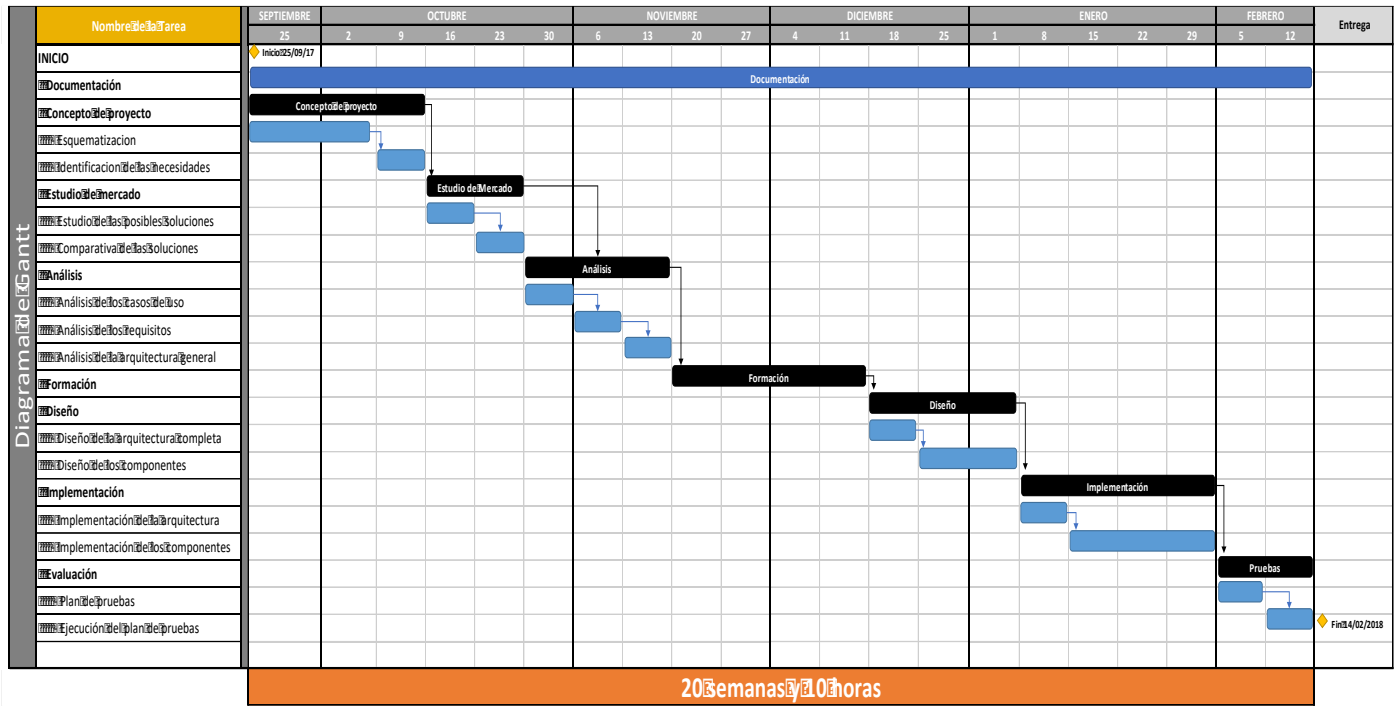


Ilustración 24: Diagrama de Gantt de la planificación.

En el diagrama de Gantt podemos visualizar cual es la planificación que se ha seguido en el proyecto. Para visualizar mayores detalles, en el **Capítulo Apéndices – Apéndice I: Detalles de la planificación, diagrama de Gantt** es posible ver una vista horizontal de la ilustración.

7.2. Presupuesto del proyecto

Basados en lo antes visto, podemos tener una idea del coste en horas hombre del proyecto, pero debemos traducir ese resultado a números. Para ello debemos identificar los costes asociados y calcular la suma total de ellos, luego debemos añadir el margen de riesgo y finalmente el margen de beneficio, esto nos entregara el precio final del proyecto sin impuestos.

Debemos tomar en cuenta que el desglose real de los costes asociados se divide en tres grupos:

- **Costes del personal**: corresponde al coste de las horas hombre que se necesita para desarrollar el proyecto.

- **Costes de los equipos:** corresponde al coste asociado a las herramientas y elementos necesarios para desarrollar el proyecto.

• **Costes indirectos:** corresponde a los costes asociados al entorno donde se desarrolla el proyecto, costes relacionados con los gastos de facturas comunes a este proyecto y a otros que se hagan en paralelo.

Luego de esta aclaración, revisaremos a cuánto ascienden estos costes

Coste del personal:

Salarios	
Salario Bruto Anual	25.000,00 €
Salario Bruto Mensual	2.083,33 €

Cuota patronal mensual		
Seguridad Social	23,60%	491,67 €
Desempleo	5,50%	114,58 €
Formación Profesional	0,60%	12,50 €
Fogasa	0,20%	4,17 €
Total, cuota patronal		622,92 €

Planificación del proyecto	
Duración del proyecto en horas	310
Duración del proyecto en semanas	20,7
Horas del trabajo por semana	15
Horas de trabajo al mes	60

Coste total del personal	
Coste mensual del trabajador	2.706,25 €
Coste de la hora trabajada	45,10 €
Total de Coste de Horas Hombre del proyecto	**13.982,29 €**

Tabla 77: Coste de personal.

Coste de los equipos:

Coste de los equipos	
Licencia de Visual Studio Code	0,00 €
Licencia de desarrollador de Apple	100,00 €
Licencia de desarrollado de Android	25,00€
Total, de Coste los equipos del proyecto	**125,00 €**

Tabla 78: coste de equipos.

Costes indirectos:

Equipos de desarrollo y documentación	
Apple Macbook Pro 13' 2017	1.244,29 €
Monitor Samsung SyncMaster EX1920	134,47 €
Impresora HP Laserjet 1018n	147,92 €
iPhone SE	345,00€
Bq Aquaris E4.5	109,00€

Amortización de equipos de desarrollo y documentación				
Concepto	amort. Anual	amort. Semanal	semanas	total
Apple Macbook Pro 13' 2017	622,14 €	11,52 €	14,4	165,91
Monitor Samsung SyncMaster EX1920	67,24 €	1,25 €	14,4	17,93
Impresora HP Laserjet 1018n	73,96 €	1,37 €	14,4	19,72
iPhone SE	172,50€	3,20€	14,4	46,08
Bq Aquaris E4.5	90,08€	1,67€	14,4	24,05
total				**273,69**

Otros Costes	
Alquiler del local (Luz y agua incluidos)	630,00 €
consumibles de ofimática	70,00 €
total	**700,00 €**

Otros Costes	
Total de Costes indirectos	**973,69 €**

Tabla 79: costes indirectos.

Para poder determinar el precio final del proyecto debemos sumar los costes, aplicar el margen de riesgo, el margen de beneficio y los impuestos correspondientes.

Precio final:

Total de Coste de Horas Hombre del proyecto	**13.982,29 €**
Total de Coste los equipos del proyecto	**125,00 €**
Total de Costes indirectos	**973,69 €**
COSTE TOTAL	**15.080,98 €**

Subtotal		**15.080,98 €**
Margen de riesgo	10%	1.508,01 €
Margen de beneficio	17%	2.563,76 €
Coste del proyecto sin IVA		19.152,76 €
IVA	21%	4.022,08 €
Coste total del proyecto IVA incluido		**23.174,84 €**

Tabla 80: coste final.

El precio final del proyecto es de **veintitrés mil ciento setenta y cuatro euros con ochenta y cuatro céntimos.** Este precio incluye un margen de riesgo de un 10%, sobre él se incluye el 17% de margen de beneficio con lo cual se obtiene el precio sin impuestos, finalmente se suma el 21% de impuesto al valor añadido.

7.3. Monetización

Dado que uno de los requisitos de la aplicación consiste en que su descarga es de carácter gratuito, la monetización de la misma se centrará en la inclusión de publicidad en la aplicación.

Para este propósito, se ha elegido la herramienta “AdMob” (26) que permite, una vez registrados, generar códigos publicitarios que se pueden incluir en la aplicación de manera customizada . Esta herramienta permite incluir publicidad en cualquier plataforma, es decir, no distingue entre plataforma iOS o Android. Esta publicidad que se incluye en la aplicación generará beneficios económicos dependiendo el número de interacciones con el “banner” publicitario incluido.

8. Conclusión y trabajos futuros

A lo largo de este apartado se irán mostrando todos los aspectos que se han concluido tras la realización de este proyecto. Se analizarán las proyecciones que habían en la concepción de la idea y se compararán con los hitos conseguidos y como la planificación del proyecto también tuvo un impacto en el alcance del mismo. De este último punto, partirán las posibles concepciones de mejoras de la aplicación, la forma en la que podemos mejorarla, darle valor agregado y mejorar su alcance.

8.1. Conclusiones

Este proyecto es una respuesta a mi gran interés por las tecnologías web y móviles por lo que las conclusiones de este apartado se basarán en como ha cambiado, este proyecto, mi concepción de los desarrollos de este tipo. También comentaré las impresiones que me surgen, una vez realizado el proyecto, con respecto al futuro del desarrollo web, del desarrollo móvil y de cómo se integran ambos desarrollos aportando ventajas por una parte y desventajas por otra. En primer lugar comentaré mis experiencias y pensamientos personales respecto al proyecto y posteriormente se hará un análisis más objetivo respecto al desarrollo del mismo.

Conclusiones personales:

La idea de este proyecto no surgió de la espontaneidad, siempre he tenido un profundo interés y curiosidad por el desarrollo de aplicaciones móviles y aplicaciones web, y por otra parte la música ha sido mi gran pasión a lo largo de mi vida. Actualmente la libertad de desarrollo en este tipo de plataformas, la cantidad de “frameworks” que hay, que nacen y que evolucionan es inmensa. Esta constante evolución en este campo me parece fascinante, ya que permite materializar cualquier idea, desarrollarla y presentarla en forma de aplicación y prueba de ello es este proyecto.

Por otra parte, a pesar del gran abanico de posibilidades para desarrollar este tipo de proyectos, me he dado cuenta que escoger la tecnología o “framework“ más apropiado es bastante complicado. También me he dado cuenta que la curva de aprendizaje de estos “frameworks” no es nada sencilla. Después de haber hecho desarrollos con Angular más Ionic, React Native o VueJS más Ionic me he dado cuenta que llegar a dominar estas herramientas requiere bastante tiempo y aprendizaje de conceptos.

Los aspectos que desconocía y me parecieron de gran interés personal se encuentran en las últimas etapas del proyecto, además de coincidir con la etapa en la que me encuentro actualmente en el proyecto donde trabajo. Esto aspectos son la preparación para subir la aplicación a producción y que sea visible para un público amplio. Este aspecto me parece muy interesante ya que hay que tener muchas variables en cuenta y me he dado cuenta de la gran importancia que tiene ya que, por ejemplo, la misma compañía de Apple invierte gran parte de su desarrollo en ofrecer herramientas de prueba de pre-producción como por ejemplo TestFlight además de analizar ellos mismos cada aplicación que se quiera subir a su propia tienda de aplicaciones. Por otra parte, me resultó de gran interés las herramientas de monetización que se disponen en caso de querer un tipo de negocio de una aplicación gratuita. Nunca llegué a pensar que hubiesen herramientas con tantas opciones de configuración de publicidad para que puedan ser añadidas de forma tan sencilla a los proyectos de los desarrolladores. De estas herramientas, me gustaría destacar a “AdMob”, la cual estuve investigando y me pareció bastante útil para añadir publicidad.

Por último me gustaría hablar acerca de mis perspectivas personales acerca del futuro de este tipo de desarrollos. En primer lugar quiero partir del gran momento actual de este ecosistema que constituyen todas las herramientas para el desarrollo de aplicaciones. La gran variedad de "frameworks" que están disponibles permitiendo a los desarrolladores elegir según el lenguaje de programación, el tipo de programación o incluso por el tipo de empresa que desarrolla estas herramientas no tiene precedentes. En este aspecto y en un futuro no muy lejano, pienso que el gran abanico de herramientas puede ser contraproducente en algunos sentidos para los desarrolladores. Al producirse constantemente nuevas herramientas, se hace cada vez más complicado poder especializarse en una en concreto, por lo que veo necesario en un futuro que surja una herramienta más estandarizada para los desarrolladores. También me parece muy importante en el futuro, que los desarrolladores tengan una formación completa en el desarrollo de las aplicaciones, es decir, que puedan ser lo que se conoce como "full stack developers", desarrolladores que entienden tanto de las APIS("Application Programming Interface") de front-end como de back-end, conocimientos en integración continua y despliegue automático, herramientas de control de versiones y que tengan una visión concreta de los modelos de negocio que hay detrás de todos estos desarrollos.

Como última conclusión y partiendo de este último punto, mi objetivo personal es llegar a ser capaz de conocer todos los puntos del desarrollo y objetivo de una aplicación, de forma que pueda traducir una solución tecnológica concreta en negocio.

Conclusiones acerca del producto:

En este punto me gustaría hacer un análisis de los distintos objetivos marcados en la concepción y planificación del proyecto haciendo hincapié en si se vieron satisfechos o no. La solución tecnológica ofrecida en este proyecto cumple con su objetivo último de ser una aplicación diseñada para poder instruir, de manera práctica, sencilla y didáctica, conceptos de teoría musical, a la vez de ofrecer libertad de composición. A partir de este objetivo último, veremos los demás objetivos que se propusieron durante la concepción de la aplicación:

- ✓ La aplicación debe poder enseñar el concepto de composición musical a través de escalas de acordes y poder realizar composiciones con la teoría aprendida.
- ✓ La aplicación debe ser portable a cualquier sistema operativo.
- ✓ La aplicación debe poder usarse con una sola mano.
- ✓ La aplicación debe ser portable a una aplicación web.
- ✓ El coste de las licencias de desarrollo y el mantenimiento de la aplicación no deben ser elevados
- ✓ El uso de internet no debe ser obligatorio, siendo factible usar la aplicación en modo *stand alone*.
- ✓ El tiempo de instalación de la aplicación no debe ser elevado, para ello la aplicación no debe tener un peso en megabytes excesivo.

Conclusiones acerca del proceso de desarrollo:

Desde un primer momento la estimación de la realización del proyecto, teniendo en cuenta los objetivos definidos, tenía en cuenta buscar el tiempo correspondiente al valor en tiempo de la asignatura de **trabajo de fin de grado con una carga de trabajo equivalente a 12 créditos ECTS**. Esta limitación influyó en gran medida el alcance del proyecto en cada una de sus fases.

En primer lugar se quiso buscar unos objetivos realistas según el tiempo planificado para la realización del proyecto, por ello se tuvo que hacer una análisis bastante profundo en las etapas más tempranas del proyecto. Este análisis tuvo sus frutos ya que los objetivos propuestos se mantuvieron siempre en la planificación establecida.

La parte más crítica de la planificación residió en el tiempo de manejar la tecnología de forma fluida. El aprendizaje de Angular y las herramientas de Cordova, conlleva el dominio de varios conceptos difíciles de asimilar y que requieren práctica para poder utilizarlos con soltura, por lo que se decidió optar por tener un periodo de formación. Viendo atrás este consideración, se fue bastante acertado tomar este tiempo ya que compensó el tiempo de las fases de diseño e implementación.

8.2. Mejoras y trabajos futuros

A lo largo de este apartado se detallarán posibles evolutivos y nuevas funcionalidades que fueron pensados a lo largo del desarrollo de este proyecto y que por cuestión de la limitación de tiempo del desarrollo del proyecto no pudieron llegar a ser implementados.

- **Tener una sección de video-tutoriales:** La presentación de teoría musical ha sido presentada a través de *slides* que a pesar de ser una solución rápida e interesante, sería más didáctica, amena, directa y efectiva si se añade una sección presentando esta misma teoría a través de distintos video-tutoriales y organizándolos a través de distintas categorías, de forma que el usuario pudiese decidir el tema específico en el que profundizar su aprendizaje.

- **Ofrecer una parte de la aplicación como *stand alone* y otra con conexión a internet para poder subir composiciones y que las vean otros usuarios:** Otro de los puntos interesantes a desarrollar en futuros trabajos es, mantener la usabilidad de la aplicación actual sin requerir conexión a internet, pero también poder ofrecer a los usuarios una forma de poder compartir sus grabaciones con otros usuarios distintos a los del dispositivo, lo que requeriría que una parte de la aplicación necesitara una conexión a internet. Esta nueva funcionalidad tendría que implementarse teniendo en cuenta dos desarrollos:

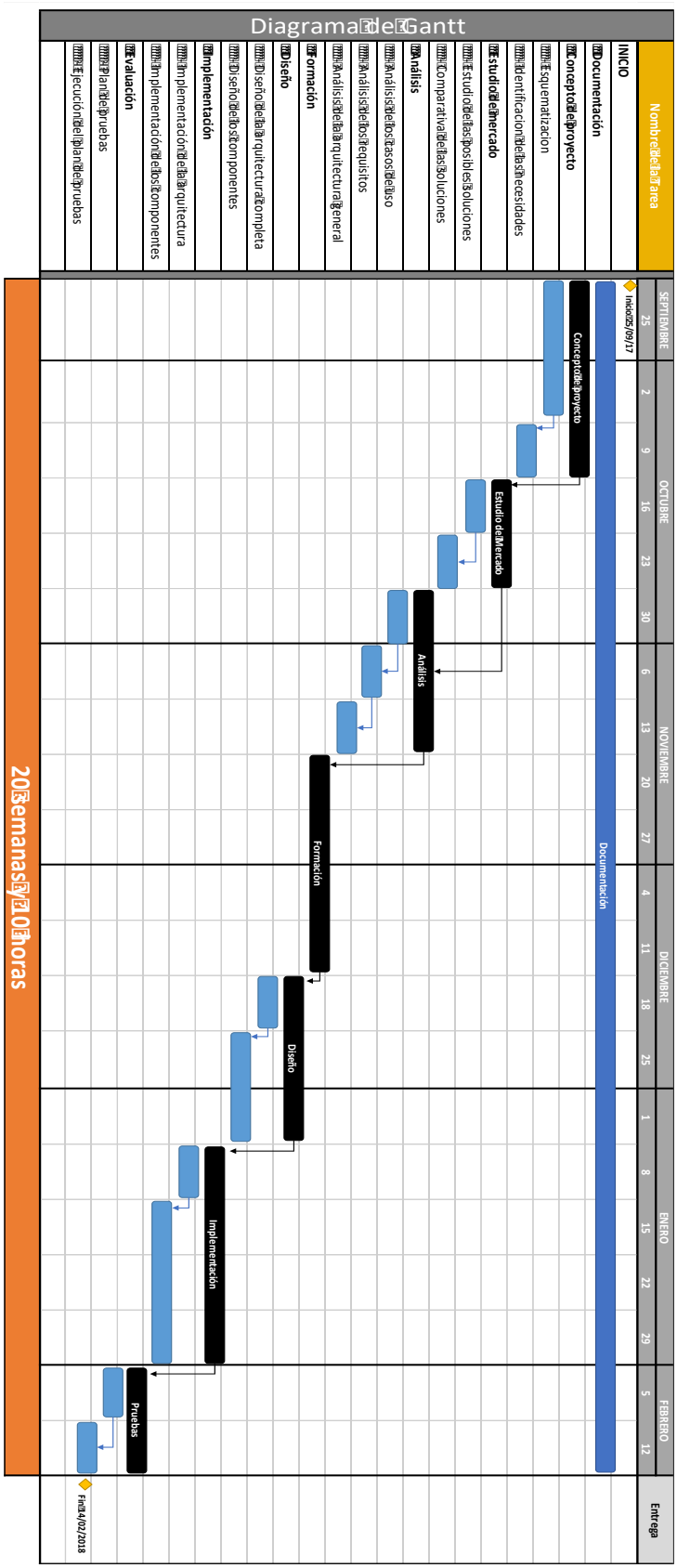
    - **Aplicar un nuevo framework:** dado que se necesitaría una base de datos externa al dispositivo y un acceso a la misma. Para ello, una opción interesante sería aplicar el *framework* MEAN(MongoDB, ExpressJS, Angular, NodeJS) realizando una API REST ofreciendo servicios para la gestión de usuarios y sus datos.
    - **Seguridad:** al tener que exponer datos a la red, habría que hacer un desarrollo para gestionar la confidencialidad de los datos, sobre todo con el cifrado de las credenciales del usuario, de forma que se pueda garantizar al usuario la seguridad de sus datos.

- **Añadir más instrumentos:** sería interesante que el usuario pudiese elegir distintos instrumentos a la hora de reproducir los acordes de la escala musical, ofreciendo un matiz más dinámico a la funcionalidad de la grabación.

- **Poder cambiar de escala musical:** Por último, una ampliación en la funcionalidad de la grabación pudiendo elegir entre distintas escalas musicales, hubiese potenciado el concepto didáctico y lúdico de la aplicación.

9. Apéndices

9.1. Gantt en horizontal



9.2. Resumen del proyecto en inglés

Introduction

Nowadays there are a lots of different mobile applications and the development of this applications and web applications continues raising. In the context of learning, specifically, music theory concepts, there are not enough tools to get the basic concepts of music composition for people without any prior knowledge and there is any tools to have an easy-learning approach. In fact, there isn't any application developed specifically for kids for learning music theory.

Motivation

Starting at this point and making an analysis of the requirements that needs an implementation to solve this problems, this project wants to present a tool capable to make an approach of the basis of music composition theory through the knowledge of chords progressions, showing how every chords scale is formed and letting users to make their owns compositions with the possibility of sharing they with some other users of the application so they can be heard by every user of the app.

Finally, the project development wants to arrive to a mayor number of users, being available in many platforms.

Goals

The solution implementation needs to have a number of specific goals to cover all of the requirements to solve the problems of music theory learning. This goals are:

- The tool must implement a graphic interface easy and intuitive to the user. The estimate time to learn the use of the app must be as shorter as possible.
- The music theory concepts in the application must be shown very easy, avoiding complex texts and explanations.
- The final user of the app, correspond to an user with little or any knowledge of music composition.
- The monetization of the developed solution must avoid as much as possible any economic invest of the user.
- The tool must be developed for many platforms to reach the maximum number of final users.
- The conception of the application must be thought under de basis of social application development, letting users to share they compositions.

Setting this list of goals the application development implementation must cover the whole list of requirements to solve the problems mentioned before.

Document structure

In this section, the structure of the whole document will be describe explaining each section. This project is based on "Métrica v3" methodology. The list of sections of the document structure are:

- **Chapter 1 - Introduction**: first section of the document. Identifies the main characteristics and structure of the document content.

- **Chapter 2 - State of the art**: presentation of the solution context. It will show a comparative between other tools found on the market and another comparative between these tools and the expected solution.

- **Chapter 3 - Analysis**: in this section, the characteristics of the expected solution will be exposed in order to divide them attending to their responsibilities and explaining every user case of interaction with the solution.

- **Chapter 4 - Design:** starting from the Analysis, the design of the solution will be exposed, so that every requirement could be covered.

- **Chapter 5 - Implementation:** in this section, the process of the implementation of the expected solution will be described.

- **Chapter 6 - Tests:** once the solution is finished, it needs to be tested in order to grant that every requirement is covered.

- **Chapter 7 - Planning and economic estimation:** in this section the planning of the project will be describe, exposing every estimate deadline and which of them had been covered. The result of the economic estimation will be analyzed and some monetization alternatives will be explained.

- **Chapter 8 - Conclusion and future works:** finally, some conclusions will be exposed in this section and the work that could not be carried out. A list work that could be implemented to improve the development of the project will be describe too.

- **Chapter 9 – Attached:** in this section some documents related to the main document will be attached like the Gantt diagram or the summary of the document of the development of the project explained in English.

- **Bibliography:** this sections contains all the bibliographic references and all the sources that have been consulted for the success of the development and implementation of the solution.

State of the art

Starting from my own personal experience trying to learn some concepts of the music theory to reach the knowledge compose my own music, I found this learning very difficult, and the tools I found they helped just a bit. Focusing on this problem, I start to think about a technologic solution to present a new tool to simplify the exposition of this concepts of music composition. And not just that: I thought about that not only would be capable of present music theory in an easy way, also to present a tool which allows the user compose without any instrument for practicing the music composition concepts.

Having in mind these principles of what a tool for composition-learning must be, a list of basic aims could be determined:

1. The tool must be capable to teach concepts of music composition through the use of chords scales and must let users to make their own compositions with the learnt knowledge.
2. The tool must be portable to any mobile operating system.
3. The application must be manageable with one hand.
4. The application must be portable to a web application.
5. The costs of the development licenses and the maintenance of the application must be not expensive.
6. Internet use must not be necessary, the use of the application must be possible in an stand-alone mode.
7. Installation time of the application must be as shorter as possible as the size of the tool.

Defining these characteristics, the analysis of the applications that could cover these requirements and are available by the current market could be specified.

Tools available on the current market

On the current market, there are lots of applications that could help in music theory learning. The applications chosen are the alternatives that are more suitable for the characteristics described before.

- **GarageBand:** it is a software property of Apple Computer for music composition. This application has a great amount of digital synthesizers and analogic and digital tools to reach a music production in a semi-professional mode.

  The disadvantage of this software is its limited license and the impossibility to work in others operating systems than Apple Computer systems.

- **Yousician**: it is an applications designed for the learning of how to play some music instruments like guitars, piano, ukulele or drums. This application runs on different operating systems like mobile systems and desktop systems.

  The disadvantage of this software is that users must have a real instrument to practice the concepts that this software expose and its free license limits every lesson every 12 hours.

- **Guitar! Smule:** it is a software that allows emulate the interaction with a guitar through the touch screen of the mobile device. Users can choose from a list of different songs to play in different levels.

  The two main disadvantages of this software are that in the free version of the app, the list of songs are limited to a basic list of songs. The other one, is related in the fact that this app do not explain any composition theory.

After the analysis of these solutions, the conclusions are that there is not any application in the current market to fulfill the requirements. From this point, the idea to develop this project arises.

Comparison between tools

Taking the seven desired characteristics in the development of the solution, a comparison with the solutions described before is needed.

After an analysis of the tools presented, gathering information about which requirements are covered by these applications, the matrix between solutions and the characteristics is:

Characteristic	GarageBand	Yousician	Guitar! Smule
1	YES	NO	NO
2	NO	YES	NO
3	NO	NO	NO
4	NO	NO	NO
5	PARTIAL	YES	YES
6	SI	NO	NO
7	NO	YES	YES

As it can be seen none of the solutions analyzed fulfill all requirements and what it is more important, just the GarageBand tool fulfill the first and more important requirement but this tool do not cover the others requirements.

Starting from this last conclusion, it is necessary to develop a project to cover all requirements, making a focus on the didactic concept of the solution.

Analysis

Along this section, the user cases will be defined and once the user cases are defined, the functional requirements and nonfunctional requirements will be defined too.

User cases

Every user case will be defined attending to:

- **The actors that will interact**: these actor could be the user, the admin or the system.
- **Goal**: the aim of the user case.
- **Settings:** expose every step the actor makes to fulfill the user case.
- **Preconditions:** all the requirements the user case need to generate successfully the interaction.
- **Postconditions:** final state after the interaction.

System requirements

Once the user cases are defined, the next step is to define the system requirements that are generated of each user case. There are two kinds of system requirements:

- **Functional requirements**: every requirement that expose a function (everything the system is capable to do) of the system.

- **Nonfunctional requirements**: every requirement that expose properties(security, capacities etc.) or constraints to the system.

This categorization is needed because after of the analysis process, the functional requirements have to generate systems to develop these functions. Nonfunctional requirements will expose properties of the system like, storage, resources, etc. Finally, the constraints will expose the limits of the system like security, response times, maximum requests etc.

The definition of each requirement will be make attending to the next aspects:

- **Priority**: Could be high, medium or low.
- **Necessity**: expose the importance of the requirement to the client. Could be high, medium or low.
- **Verifiability**: indicates the level to find the requirement on the final solution.
- **Stability**: indicates the level which the requirement can have modifications during the development of the project.
- **Description**: describe the characteristics of the project.

Design

This section will explain every decision made related to the design of the project. The technologies used will be explained too.

Architecture design of the system

For the architecture of the system, the best suitable architecture will be de MVC architecture (Model-View-controller architecture). The next diagram will show how every element of the architecture will be correlated each other.

Technologies used for the development of the solution

This subsection will present every technology presented in the solution.

- **Ionic 2**: this framework combines the web framework Angular with Apache Cordova, to implement native functions for mobile applications. This framework allows to make hybrid solutions for Android and iOS and makes very easy to export this solution to a web project.
- **Angular**: is a ECMAScript framework that makes possible the development of SPA(single page applications).
- **Apache Cordova**: is a framework that use web technologies for the development of cross platforms applications, without the necessity of native language of each platform.
- **NPM(node package manager)**: is a manager for ECMAScript frameworks. It manage the dependencies for every project.

Design of the persistence layer and graphic layer

The persistence of the data will be handled by the device. The structure of every element to persist will respond to a JSON structure.

The design of the graphic layer will use the ionic 2 tools for developing graphic interfaces for the development of mobile applications.

Implementation

Along this section, the installation and configuration of each environment of development will be explained:

- Installation of the package manager: for this purpose, the technology chosen is NPM. For its installation, only the installation of NodeJS is required.

- Installation of every environment of development: for the installation of Visual Studio Code it is possible to get the installer from its official web page. For the installation of de XCode tool, it is possible to install it from the App Store.

- Dependencies installation: the installation of every dependency of the ionic 2 project of the project is possible through NPM. The list of dependencies to install is:

Implantation

The implantation will be in two different ways: one for iOS platforms and another for Android platforms.

- iOS: the release for production will be made through iTunes Connect.
- Android: the release for production will be made through Google Play Developer Console.

Tests

Every test made for the project will correspond to a three different contexts of the project. Because of that, there will be three different kinds of test that will correspond to unit testing, integration testing and “end to end” tests:

- **Unit tests**: this tests will test the functionality of every component of the code, verifying the correct functionality in an isolated way. The technology for this test will be the karma-jasmine framework.

- **Integration tests**: the setup of this tests will verify that the components of the application will work correctly together.

- **End to end tests**: this kind of tests will verify that the solution fulfill every user case and requirement defined. The technology to make this test will be Selenium.

To define each test, the format will attend to some concepts:

- Goals: determinates the goal of the test.
- Necessities: indicates the pre-requirements for the tests.
- Inputs: values of inputs.
- Outputs: values of the outputs.
- Requirements: list of the requirements defined and correlated to the test.

Planning and economic estimation

This section will determine the planning and schedule for the development and the economic estimation of the project. For the planning, every stage of the project is presented in the Gantt diagram included in the attached section of this document.

Planning

The total duration for this project was estimated in 310 hours with a 10% as an error margin, so the total duration for the development is 279 hours.

Setting that every week the burden of work will be of 10 hours plus 5 hours of documentation (15 hours per week), the total duration of the project is 20 week and ten hours. Starting from the 25 of September of 2017, the deadline of the project will be de 14 of February of 2018 .

Economic estimation

The economic estimation will correspond of three kinds of costs: staff costs, equipment costs and indirect costs:

Staff costs:

Salaries	
Annual gros salary	25.000,00 €
Monthly gross salary	2.083,33 €

Monthly employers share		
Social Security	23,60%	491,67 €
Unemployment	5,50%	114,58 €
Professional training	0,60%	12,50 €
Fogasa	0,20%	4,17 €
Total, employer share		622,92 €

Planning of the project	
Project duration (hours)	310
Project duration (weeks)	20,7
Hours worked per week	15
Hours worked per month	60

Staff costs	
Monthly cost of employer	2.706,25 €
Costs of every hour worked	45,10 €
STAFF COSTS	**13.982,29 €**

Equipment costs:

Equipment costs	
Visual Studio Code license	0,00 €
Apple developer license	100,00 €
Android developer license	25,00€
EQUIPMENT COSTS	**125,00 €**

Indirect costs:

Development equipment and documentation	
Apple Macbook Pro 13' 2017	1.244,29 €
Samsung SyncMaster EX1920 display	134,47 €
HP Laserjet 1018n printer	147,92 €
iPhone SE	345,00€
Bq Aquaris E4.5	109,00€

Amortization				
Concept	amort. Annual	amort. weekly	weeks	total
Apple Macbook Pro 13' 2017	622,14 €	11,52 €	14,4	165,91
Samsung SyncMaster EX1920 display	67,24 €	1,25 €	14,4	17,93
HP Laserjet 1018n printer	73,96 €	1,37 €	14,4	19,72
iPhone SE	172,50€	3,20€	14,4	46,08
Bq Aquaris E4.5	90,08€	1,67€	14,4	24,05
total				**273,69**

Other costs	
Local rental (includes electricity and water)	630,00 €
Ofimatic suite	70,00 €
total	**700,00 €**

Other costs	
Total de Costes indirectos	**973,69 €**

Project total cost	**23.174,84 €**

Conclusion and future works

Along this section, I will describe my conclusions about the development of this project. I will analyze the early projections of the concept of the solution and I will compare it with the goals achieve and how the planning of the project had a great impact on the range of the solution.

Personal conclusions

The idea of this project did not come randomly, I always had a deep curiosity about mobile and web applications. On the other hand, music have been my great hobby along my life. Currently, the development environment for this kind of projects and the amount of frameworks is huge. The continuous evolution in this field, keep my interest because in let to develop any idea in an app form.

This great amount of technologies, has also, some disadvantages. One of them, is that is very difficult to choose the best technology to develop an idea and some of these technologies had difficult concepts that developers must learn.

Finally my general personal conclusion is that, nowadays, to be in touch of the latest technologies in this field is fundamental.

Product conclusions

In this point, I am very happy to have reach all the goals I planned before. It was hard to define realistic goals in order to reach them with a deadline, but with the background given to me by the different knowledge of the university I succeeded on plan this project.

Development process conclusions

From the beginning the estimation of the project was based on the time corresponding to the subject of this final degree project that is estimated on 12 ECTS. This limitation has a great impact on every progress of each stage of the project. One of the most important points of the development process was, to be familiar with the Angular framework, so the training stage, in order to learn this technology, was fundamental.

Future work

There were some features I would like to add to this project but due to the deadline limit o could not implement. This features are:

- A section of video-tutorials.
- Make another part of the application with the possibility to have internet and share the compositions with others users.
- Add more instruments.
- The possibility to change the chords scale.

Bibliografía

1. **GarageBand.** https://www.apple.com/es/ios/garageband/. *https://www.apple.com/es/ios/garageband/.* [Online] [Cited: 06 10, 2018.] https://www.apple.com/es/ios/garageband/.
2. **Yousician.** https://yousician.com/. *https://yousician.com/.* [Online] [Cited: 06 10, 2018.] https://yousician.com/.
3. —. https://www.youtube.com/watch?v=xFnB0uK7PcQ&feature=youtu.be. *https://www.youtube.com/watch?v=xFnB0uK7PcQ&feature=youtu.be.* [Online] [Cited: 06 10, 2018.] https://www.youtube.com/watch?v=xFnB0uK7PcQ&feature=youtu.be.
4. **Smule.** https://itunes.apple.com/us/app/guitar!-by-smule/id632353530?mt=8&app=itunes. *https://itunes.apple.com/us/app/guitar!-by-smule/id632353530?mt=8&app=itunes.* [Online] [Cited: 06 10, 2018.] https://itunes.apple.com/us/app/guitar!-by-smule/id632353530?mt=8&app=itunes.
5. **Electrónica, portal de Administración.** https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html#.Wx2yjtOFMWo. *https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html#.Wx2yjtOFMWo.* [Online] [Cited: 06 10, 2018.] https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html#.Wx2yjtOFMWo.
6. **Mozilla, Developer.** https://developer.mozilla.org/en-US/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture. *https://developer.mozilla.org/en-US/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture.* [Online] [Cited: 06 10, 2018.] https://developer.mozilla.org/en-US/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture.
7. **framework, Ionic.** https://ionicframework.com/docs/intro/installation/. *https://ionicframework.com/docs/intro/installation/.* [Online] [Cited: 06 10, 2018.] https://ionicframework.com/docs/intro/installation/.
8. **Angular.io.** https://angular.io/docs. *https://angular.io/docs.* [Online] [Cited: 06 10, 2018.] https://angular.io/docs.
9. **cordova.apache.org.** https://cordova.apache.org/docs/en/latest/guide/overview/index.html. *https://cordova.apache.org/docs/en/latest/guide/overview/index.html.* [Online] [Cited: 06 10, 2018.] https://cordova.apache.org/docs/en/latest/guide/overview/index.html.
10. **npmjs.com.** https://docs.npmjs.com/getting-started/what-is-npm. *https://docs.npmjs.com/getting-started/what-is-npm.* [Online] [Cited: 06 10, 2018.] https://docs.npmjs.com/getting-started/what-is-npm.
11. **es6-features.org.** http://es6-features.org/#ObjectPropertyAssignment. *http://es6-features.org/#ObjectPropertyAssignment.* [Online] [Cited: 06 10, 2018.] http://es6-features.org/#ObjectPropertyAssignment.
12. **theblog.adobe.com.** https://theblog.adobe.com/mobile-design-best-practices/?origref=https%3A%2F%2Fwww.google.es%2F. *https://theblog.adobe.com/mobile-design-best-practices/?origref=https%3A%2F%2Fwww.google.es%2F.* [Online] [Cited: 06 10,

2018.] https://theblog.adobe.com/mobile-design-best-practices/?origref=https%3A%2F%2Fwww.google.es%2F.
13. **nodejs.org.** https://nodejs.org/en/blog/release/v6.10.3/. *https://nodejs.org/en/blog/release/v6.10.3/.* [Online] [Cited: 06 10, 2018.] https://nodejs.org/en/blog/release/v6.10.3/.
14. **libraries.io.** https://libraries.io/npm/npm/3.10.10. *https://libraries.io/npm/npm/3.10.10.* [Online] [Cited: 06 10, 2018.] https://libraries.io/npm/npm/3.10.10.
15. **code.visualstudio.com.** https://code.visualstudio.com/docs. *https://code.visualstudio.com/docs.* [Online] [Cited: 06 10, 2018.] https://code.visualstudio.com/docs.
16. **developer.apple.com.** https://developer.apple.com/xcode/ide/. *https://developer.apple.com/xcode/ide/.* [Online] [Cited: 06 10, 2018.] https://developer.apple.com/xcode/ide/.
17. **eslint.org.** https://eslint.org/docs/user-guide/getting-started. *https://eslint.org/docs/user-guide/getting-started.* [Online] [Cited: 06 10, 2018.] https://eslint.org/docs/user-guide/getting-started.
18. **bitbucket.org.** https://bitbucket.org/product. *https://bitbucket.org/product.* [Online] [Cited: 06 10, 2018.] https://bitbucket.org/product.
19. **git-scm.com.** https://git-scm.com/book/en/v2/Git-Branching-Branching-Workflows. *https://git-scm.com/book/en/v2/Git-Branching-Branching-Workflows.* [Online] [Cited: 06 10, 2018.] https://git-scm.com/book/en/v2/Git-Branching-Branching-Workflows.
20. **angular.io.** https://angular.io/guide/styleguide. *https://angular.io/guide/styleguide.* [Online] [Cited: 06 10, 2018.] https://angular.io/guide/styleguide.
21. **itunesconnect.apple.com.** https://itunesconnect.apple.com/. *https://itunesconnect.apple.com/.* [Online] [Cited: 06 10, 2018.] https://itunesconnect.apple.com/.
22. **docs.microsoft.com.** https://docs.microsoft.com/es-es/xamarin/ios/deploy-test/app-distribution/app-store-distribution/itunesconnect. *https://docs.microsoft.com/es-es/xamarin/ios/deploy-test/app-distribution/app-store-distribution/itunesconnect.* [Online] [Cited: 06 10, 2018.] https://docs.microsoft.com/es-es/xamarin/ios/deploy-test/app-distribution/app-store-distribution/itunesconnect.
23. **ionicframework.com.** https://ionicframework.com/docs/v1/guide/publishing.html. *https://ionicframework.com/docs/v1/guide/publishing.html.* [Online] [Cited: 06 10, 2018.] https://ionicframework.com/docs/v1/guide/publishing.html.
24. **jasmine.github.io.** https://jasmine.github.io/setup/nodejs.html. *https://jasmine.github.io/setup/nodejs.html.* [Online] [Cited: 06 10, 2018.] https://jasmine.github.io/setup/nodejs.html.
25. **seleniumhq.org.** https://www.seleniumhq.org/docs/. *https://www.seleniumhq.org/docs/.* [Online] [Cited: 06 10, 2018.] https://www.seleniumhq.org/docs/.
26. **google.es.** https://www.google.es/admob/. *https://www.google.es/admob/.* [Online] [Cited: 06 10, 2018.] https://www.google.es/admob/.

- Fin del documento -