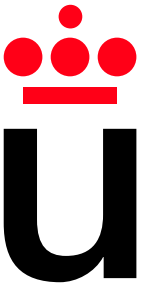


Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

Grado en Ingeniería Informática

Curso 2021-2022

Trabajo Fin de Grado

EARFIT: APLICACIÓN PARA ENTRENAMIENTO AUDITIVO MUSICAL

Autor: Alberto Gómez Cano
Tutor: Manuel Rubio Sánchez

Agradecimientos

Quiero agradecer este TFG a mi familia por siempre estar a mi lado aconsejándome en los momentos más difíciles de mi carrera.

A mi novia por aguantar mis frustraciones y animarme a seguir adelante.

A mi tutor académico Manuel Rubio Sánchez que me ha apoyado para realizar este trabajo.

Y finalmente, me gustaría agradecer a todos mis compañeros que han compartido esta carrera conmigo.

¡A todos, mil gracias!

Resumen (Redactar mejor)

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

Consistirá en una aplicación web basada Next.js y TypeScript y que será desplegada en Vercel. Utilizando Metodologías Ágiles y prácticas de DevOps para llevar a cabo la organización del proyecto.

La aplicación tiene el nombre de Earfit y se puede visitar en el siguiente enlace:

`https://earfit-alberttogoca.vercel.app/`

Palabras clave:

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

Índice de contenidos

Índice de figuras

Índice de códigos

1 Introducción

[1] En este capítulo daremos un poco de explicación sobre qué consiste el entrenamiento auditivo para que pueda comprender mejor el objetivo general y alcance del trabajo. Además, se realizará una pequeña descripción del proyecto y se comentará brevemente el estado del arte actual. Por último, se establecerán los objetivos y la estructura del documento.

1.1. Entrenamiento Auditivo

Los oídos son la herramienta más importante a la hora de hacer música. Pero si no se entrenan, nunca desarrollarán toda su potencia. Los músicos, productores y DJs se pueden beneficiar del entrenamiento de sus oídos. Puede resultar muy útil a la hora de mezclar música y componer canciones.

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

Muchas personas suponen que tener "oído musical" es tener la capacidad de identificar una nota al oírla. Tener oído musical es, también, ser capaz de escuchar y comprender música interiormente, sin que ésta este físicamente presente, igual que reflexionamos sobre palabras que hemos escuchado.

El entrenamiento auditivo es importante porque la escucha es una habilidad, al igual que tocar el piano. Por ejemplo, las melodías son simplemente series de

intervalos. Con el entrenamiento necesario para identificar los intervalos, puedes aprender a tocar una melodía de oído.

Para los productores de música, el entrenamiento auditivo sirve para identificar los rangos de frecuencias más rápidamente. El entrenamiento auditivo ayuda a encontrar las frecuencias que necesitas para conseguir los efectos buscados.

En conclusión, aprender entrenamiento auditivo te lleva al siguiente nivel como músico ya que te permite sacar canciones más rápido, con mayor precisión, improvisarás mejor, podrás llevar al instrumento las melodías que imaginas con mayor facilidad, y en general te permitirá ser mucho

1.1.1. Oído Absoluto

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

1.1.2. Oído Relativo

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

Las personas que disponen de oído relativo son capaces de:

- Denotar la distancia de una nota musical desde una nota de referencia establecida.
- Seguir la notación musical, esto permite cantar correctamente una melodía entonando cada nota de acuerdo a la distancia con la nota anterior.
- Seguir la notación musical, esto permite cantar correctamente una melodía entonando cada nota de acuerdo a la distancia con la nota anterior.

Los ejercicios más comunes de entrenamiento auditivo te ayudarán a desarrollar tu oído relativo.

1.2. Descripción del Proyecto

El presente Trabajo de Fin de Grado se centra en el diseño e implementación de una aplicación web con la finalidad de ayudar a músicos a desarrollar su oído musical mediante la realización de ejercicios de entrenamiento auditivo.

La aplicación se centrará en tres tipos diferentes de ejercicios divididos en la localización de notas, intervalos y escalas.

Para cada uno de los ejercicios se ha diseñado una página específica, que incluirá:

El propio ejercicio, que consistirá en un botón que reproducirá el sonido correspondiente, calculado aleatoriamente teniendo en cuenta las posibles respuestas y los botones correspondientes a las respuestas a elegir. La idea es que el usuario trate de adivinar el sonido que está sonando. Cuando pulse en una respuesta se evaluará si es correcta o incorrecta. Si es incorrecta el botón cambiará a color rojo y podrá seguir probando. Si es correcta el botón cambiará a verde un segundo, se reseterán los botones y se calculará un nuevo sonido. También existe un contador de racha que aparecerá cuando se den tres aciertos consecutivos y desaparecerá cuando se falle.

A su derecha aparecerán las opciones con las que podrás personalizar el ejercicio a tu gusto, añadiendo o quitando respuestas del ejercicio lo que incrementará o disminuirá la dificultad.

Además, en el ejercicio de notas se podrá cambiar la escala de la que se seleccionan las notas y en los ejercicios de intervalos y escalas se podrá elegir si las sucesión de notas será ascendente o descendente.

También todos los ejercicios incorporarán un piano que los usuarios podrán usar para tocar notas de referencia y ayudarse en la obtención de la respuesta correcta.

Aparte, se ha incorporado la posibilidad de poder cambiar el instrumento que suena y que será persistente para toda la aplicación.

Toda la aplicación se ha diseñado teniendo en mente que el diseño fuera simple, fácil de entender y limpio. Ha sido desarrollada utilizando Nextjs y Typescript que se explicarán más adelante en el apartado Implementación. Y se ha seguido un despliegue continuo de la aplicación en Vercel que también se explicará más adelante en Despliegue Continuo.

1.3. Estado del Arte

En la actualidad ya existen algunas aplicaciones para entrenar el oído como pueden ser:

ToneGym: `https://www.tonegym.co/`

ToneDear `https://tonedear.com/`

EarMaster: `https://www.earmaster.com/es/`

La mayoría de ellas está de acuerdo en qué el método más efectivo para progresar en el entrenamiento auditivo es el siguiente:

- Aumentar la frecuencia que se practica, no la duración. Esto se debe a que, después de haber pasado un tiempo practicando, el cerebro continúa pensando en ello y haciendo nuevas conexiones neuronales en segundo plano, incluso mientras se duerme (especialmente mientras se duerme). Por esta razón, se recomienda marcar tus ejercicios favoritos y hacerlos todos los días durante un tiempo determinado.
- Empezar de forma simple y aumentar gradualmente la dificultad. La práctica debe ser un desafío, pero no tanto como para sentirse abrumado.
- Realizar un seguimiento del progreso. Tener en un cuaderno, un archivo de texto o incluso una hoja de cálculo con el seguimiento del progreso. Esto permite saber con certeza si se está mejorando. Si puedes ver tu mejora, te alentará a continuar. También puede ayudar anotar cuándo se está estancado para poder encontrar la causa. Quizá no practicas con la suficiente frecuencia o aumentaste la dificultad demasiado rápido.
- Cantar escalas e intervalos. Todos los ejercicios de estos sitios implican identificar notas en lugar de generarlas, pero eso no significa que no debas cantar junto con ellas. Esto ayuda a internalizar los tonos. Es especialmente útil para el ejercicio de escalas. Intentar cantar hacia arriba y hacia abajo todas las escalas te ayudará a interiorizarlas.
- Transcribir música con un instrumento. Elegir tus canciones favoritas e intentar descubrir las notas con un instrumento es una buena práctica. Puedes comenzar con la melodía y luego intentar descifrar los acordes, o puedes empezar con los acordes y luego intentar descifrar la melodía. Practica en ambos sentidos.

Todas ellas plantean diferentes ejercicios para reconocer notas, intervalos y escalas. La mayor diferencia que se encuentra en ellas es su diseño y su nivel de personalización de los ejercicios, que es donde vamos a enfocar este trabajo.

Dando no sólo una aplicación con ejercicios sino una herramienta que te dé la libertad de configurarla a tu gusto.

1.4. Objetivos

El objetivo principal del TFG es desarrollar una herramienta que permita ayudar a músicos a desarrollar su oído musical. Mediante el entrenamiento auditivo.

Subobjetivos:

- Desarrollar una interfaz interactiva
- Implementar diferentes tipos de ejercicios
- Incluir varios instrumentos

1.5. Estructura del documento

En este apartado se especificará, como su título indica, la estructura que plantea este breve trabajo o memoria. El objetivo de dicho punto es acercar al lector las diferentes partes que componen este proyecto, así como ayudarle en la comprensión de este.

Se espera que sea lo suficientemente clarificador y que permita una lectura comprensible, rápida y amena del trabajo que nos ocupa.

- En el Capítulo 1: Introducción, hemos realizado una pequeña puesta en contexto y explicación de los objetivos de este TFG. Además hemos explicado en qué consiste el Entrenamiento Auditivo y porqué es importante. A parte de una breve descripción del proyecto.

- En el Capítulo 2: Contenidos Principales, se explicará cómo ha sido el desarrollo de la aplicación.

  Empezando por Creación de Propuesta, donde exploraremos el concepto de Cómo nace la Idea aplicando Lean Startup, Design Thinking y diferentes métodos utilizados en esta fase, desde que surge la idea hasta el primer prototipo.

  Más tarde en Metodología de Trabajo se explicará la metodología que se ha seguido día a día a la hora de realizar la aplicación, hablaremos de Scrum y DevOps. Cómo se ha realizado la Integración Continua CI con Github y el Despliegue Continuo CD con Vercel.

Luego, en Desarrollo e Implementación se trataran las Tecnologías Empleadas cómo Next.js o TypeScript entre otras.

Se detallará el diseño y un análisis de cómo funciona la aplicación por dentro en el apartado Implementación y cómo se ha verificado su funcionamiento en Testing.

Finalmente, se mostrará el resultado final de todo este proceso.

- Por último, en el Capítulo 3: Conclusiones, se detallan las conclusiones derivadas del trabajo, lo qué he aprendido, y lo que queda por mejorar.

2

Contenidos principales

2.1. Necesidad, Hipótesis y Solución

¿Cómo nace la idea?

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

Junto con mi tutor Manuel Rubio, que es un apasionado de la música, intentamos dar forma a esta solución. Tuvimos varias reuniones en las que él, cómo músico, me explicaba las dificultades por las que pasan a la hora de entrenar el oído. Una vez tenidas claras sus necesidades era hora de crear una solución que aportase valor. Para ello decidí crear un producto mínimo viable (**MVP**) que cumpliese con las especificaciones requeridas usando **Lean Startup**, como método de aprendizaje validado y **Design Thinking**, como método de generación de ideas innovadoras.

Aplicando las bases de Lean Startup combinándolas con prácticas de Design Thinking **(Lean Design)**, como se explica a continuación, se consiguió desarrollar un MVP.

2.1.1. Lean Startup

Qué es

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

Se compone de 5 principios básicos:

1. Los emprendedores están en todas partes
2. El espíritu emprendedor es management
3. Aprendizaje validado
4. Crear medir aprender
5. Contabilidad de la innovación

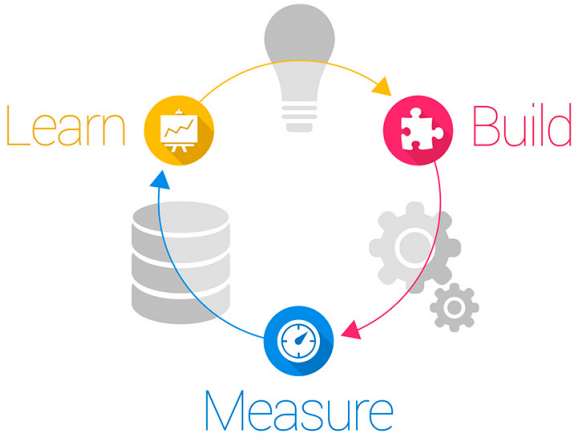
Porqué he decidido usarlo

El Lean Startup **sirve para** acortar los ciclos de desarrollo, medir el progreso y ganar feedback por parte de los usuarios. Esto se consigue porque se basa en el **aprendizaje validado**, la **experimentación científica** y la **iteración** en los lanzamientos del producto.

Aunque por su nombre pueda parecer que sólo esta enfocado al mundo Startup y crear una nueva empresa, en realidad es una herramienta imprescindible para la puesta en marcha de **soluciones software** como veremos a continuación.

Explicación Detallada

Para entender mejor el por qué de usar este método, explicaremos las **3 partes** de las que se compone y que hemos seguido, añadiendo prácticas de **Design Thinking** para la construcción de nuestra solución. Lo cual ayudará a entender el proceso que se ha seguido hasta llegar al resultado final:



1. VER

Comenzar

El objetivo es conocer qué quieren los potenciales usuarios.

Las startups se caracterizan por su constante incertidumbre. Así que en vez de hacer planes concretos apuntando a una sola dirección deben recoger feedback de sus potenciales clientes y hacer ajustes con la información que van recogiendo. Esto se llama **Circuito de feedback de información: crear, medir, aprender.**

Definir

Diseñar para crear un nuevo producto o servicio bajo condiciones de incertidumbre extrema. Un emprendedor no es sólo el fundador de una startup, también pueden ser managers de grande compañías creando nuevos negocios, o como es nuestro caso el lanzamiento de una nueva solución software.

Bajo estas condiciones de **incertidumbre** extrema las herramientas tradicionales de management no pueden funcionar bien. **Por eso necesitamos Lean Startup**.

Aprender

La función más importante de una startup es aprender qué quieren realmente los consumidores y que llevaría a un negocio sostenible. Para ello se necesita un proceso con disciplina de aprendizaje, lo que se llama **aprendizaje validado**.

Se trata de tener hipótesis testeables y diseñar experimentos para testearlas, luego analizar los datos para así aprender de ellos.

Experimentar

La metodología Lean Startup ve el hecho de crear una startup/producto como una ciencia. Crear una **hipótesis**, diseñar un **experimento** para **testear** esa hipótesis, llevar a cabo el experimento, reunir datos, reflexionar y ver si validan o rechazan la hipótesis.

Las hipótesis deberían girar entorno al problema más importante de una startup, cómo construir un negocio sostenible alrededor de tu visión. O en nuestro caso cómo construir una solución que realmente aporte valor.

Según Lean Startup la mejor forma es trackeando el comportamiento de gente real y no preguntar por opiniones ya que a veces no son capaces de verbalizar lo que quieren.

Por ello hay que pensar en el experimento más barato y rápido para validar la hipótesis, lanzarlo pronto dará más información del comprador antes de lanzar el producto real. Sobre si están interesados o no en lo que estás construyendo y te hace ver preocupaciones de estos que no tenias cuantificadas.

2. DIRIGIR

Qué tan rápido puedes desarrollar tus experimentos.

Primero estableces tu hipótesis, luego construyes tu **producto minimo viable (MVP)** para testear esa hipótesis. Luego llevas a cabo el experimento, lo más común es poner los usuarios delante del producto y recoger así información sobre su comportamiento. Recoges los datos y reflexionas sobre ellos para seguir adelante o cambiar de dirección.

Cuanto más rápido te muevas en este circuito, más rápido aprendes.

Saltar

Normalmente hay 2 tipos de hipótesis: hipótesis de creación de valor e hipótesis de crecimiento. Algunas hipotesis son más arriesgadas que otras. Unas tienen muchas probabilidades de ser ciertas y otras mucho menos. Las que tienen más riesgo son similares a las siguientes:

¿Tiene la gente el problema que crees que tienen?

¿Realmente quieren lo que estas ofreciendo?

¿Pagarían por ello?

Un marco para empezar una startup es por analogía si alguien ha tenido éxito con un modelo de negocio si realizamos el mismo modelo de negocio con un servicio distinto también debería tener éxito. Esta idea puede parecer perfecta

pero por otro lado esconde algunas presunciones sobre como funcionará el negocio. Para poder hacer una buena analogía hay que detallarla mucho más.

Hay muchas asunciones a tener en cuenta antes de construir tu prototipo por analogía. Lean Startup habla de una técnica utilizada en la fabricación japonesa llamada Genchi Genbutsu-"Go and See" que sería algo como ve y velo por ti mismo.

Una forma de hacerlo podría ser entrevistando a los potenciales clientes y una vez que estamos seguros de que el problema que queremos solventar existe, entonces viene el momento de construir un test.

Probar

¿Cuál es el **producto mínimo viable (MVP)** que puedes crear para obtener datos reales en tu hipótesis?

Este MVP no debe ser perfecto, lo que queremos es aprender lo más rápido posible. Añadir características a nuestro producto que todavía ni sabemos cómo va a ser afectado en el mercado es una perdida de tiempo.

Ejemplos de MVP: Landing Page con explicación de la app, Video con características principales y casos de uso, Mago de Oz: hacer creer al usuario que esta interactuando con la app o el utilizado aquí: el Conserje: empezando con un solo cliente y escalar.

Medir

Lean Startup establece que los informes deben entrar en las tres A: **Accionables**: que nos permitan conocer realmente cómo va una parte del modelo de negocio. **Accesibles**: que sean fácilmente interpretables para poder sacar conclusiones. **Auditables**: que puedan comprobarse por terceras persona, esto es importante para inversores.

Algunas buenas métricas pueden ser estas:

- Engagement
- Tiempo en el producto por usuario/por semana
- Porcentaje de usuarios que vuelven
- Crecimiento
- Factores virales
- Conversión en cada paso
- Nuevos usuarios ganados por semana
- Finanzas
- Coste por adquisición de nuevo cliente
- Valor por ciclo de vida de usuario

Las que mejor se adecuan a nuestro producto son: Engagement y Tiempo en

el producto por usuario.

También hemos usado Test A/B: enseñando dos versiones diferentes del producto al mismo tiempo para tomar decisiones acertadas.

Pivotar o Perseverar

Decidir si continuar con la dirección que habíamos tomado o cambiamos nuestras hipótesis esenciales sobre nuestro negocio. Dos señales son: nuestras métricas no son lo suficientemente buenas para conseguir los objetivos o los experimentos están llevando a tener menos progresos lo que significa que no se están teniendo buenas ideas.

3. ACELERAR

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

Crecer

Un crecimiento sostenible se basa en 4 elementos que deben coincidir:

- Publicidad
- Negocio repetitivo
- Efectos secundarios según la exposición o Status del producto
- Boca-Oreja

Un buen encaje entre producto y mercado ocurre cuando una startup encuentra una gran cantidad de consumidores que resuenan con su producto. Una buena idea sería usar técnicas de **Growth Hacking.**

Adaptar

Una startup debe estar en constante cambio adaptandose a los nuevos clientes que van llegando. Los Early Adopters, los primeros en utilizar el producto no serán muy exigentes con la calidad pero esto es así con los nuevos clientes que vendrán más tarde.

Ir demasiado rápido puede causar problemas. Para identificar estos problemas hay que preguntarse los 5 ¿por qué?:

¿Por qué ha sucedido el problema 'A'? A causa de 'B'

¿Por qué ha sucedido 'B'? Por 'C'

Y así sucesivamente hasta 5 ¿Por qué? y llegar a la raíz del problema. Ya que sino estaremos viendo una respuesta superficial.

Innovar

Se debe decidir si seguir con las necesidades de los clientes o innovar. En cuanto a la innovación, por un lado tenemos a la **Innovación Sostenida**, que se basaría en ir incrementando mejoras al producto ya existente. O bien la **Innovación Disruptiva**, que se basaría en crear nuevos productos rompedores.

Focalizar

En el pasado los hombres estaban primero y hoy en día lo primero son los sistemas. Las startups deben tener el foco en el grupo y no en el individuo. Deben evitar despilfarrar en cosas que no son las adecuadas. Focalizándose sólo en actividades que van a generar valor.

2.1.2. Design Thinking

Qué es

Es un método para generar ideas innovadoras que centra su eficacia en entender y dar solución a las necesidades reales de los usuarios. Proviene de la forma en la que trabajan los diseñadores de producto. De ahí su nombre, que en español se traduce de forma literal como "Pensamiento de Diseño.º "La forma en la que piensan los diseñadores".

Se empezó a desarrollar de forma teórica en la Universidad de Stanford en California (EEUU) a partir de los años 70, y su primera aplicabilidad con fines lucrativos como "Design Thinking"la llevó a cabo la consultoría de diseño IDEO, siendo hoy en día su principal precursora.

Porqué he decidido usarlo

Mientras que **Lean startup** surge como una metodología que permite impactar en el mercado con éxito, el **Design Thinking** busca el diseño de experiencias de alto valor, centradas en el usuario. Dado que Lean Startup no se centra en el usuario, lo ideal es coger lo mejor de ambas metodologías.

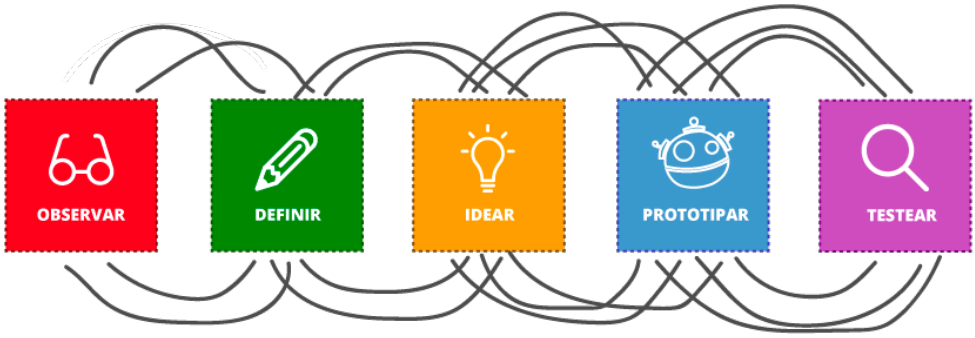
Añadir esta metodología al proceso de creación ayuda a conocer al cliente en profundidad y encontrar soluciones prácticas ante los problemas de las personas en un proceso ágil.

El proceso de innovar centrándonos en las personas. Partir desde las necesidades de los clientes, para generar productos o servicios que satisfagan sus

necesidades.

Etapas

El proceso de Design Thinking se compone de cinco etapas. No es lineal. En cualquier momento se puede ir hacia atrás o hacia delante si se cree oportuno, saltando incluso a etapas no consecutivas. Se comienza recolectando mucha información, generando una gran cantidad de contenido, que crecerá o disminuirá dependiendo de la fase en la que te encuentres.



A lo largo del proceso se irá afinando ese contenido hasta desembocar en una solución que cumpla con los objetivos. Y seguramente, incluso los supere.

Empatiza: El proceso de Design Thinking comienza con una profunda comprensión de las necesidades de los usuarios implicados en la solución que estemos desarrollando, y también de su entorno. Debemos ser capaces de ponernos en la piel de dichas personas para ser capaces de generar soluciones consecuentes con sus realidades.

Algunas técnicas: **Entrevistas, Inmersión Cognitiva, Focus Group.**

Define: Durante la etapa de Definición, debemos cribar la información recopilada durante la fase de Empatía y quedarnos con lo que realmente aporta valor y nos lleva al alcance de nuevas perspectivas interesantes. Identificaremos problemas cuyas soluciones serán clave para la obtención de un resultado innovador.

Algunas técnicas: **Mapa de Empatía, User Personas, Listas de Problemas.**

Idea: La etapa de Ideación tiene como objetivo la generación de un sinfín de opciones. No debemos quedarnos con la primera idea que se nos ocurra. En esta fase, las actividades favorecen el pensamiento expansivo y debemos eliminar los juicios de valor. A veces, las ideas más estrambóticas son las que generan soluciones visionarias.

Algunas técnicas: **Brainstorming, Product Box, MindMap.**

Prototipa: En la etapa de Prototipado volvemos las ideas realidad. Construir prototipos hace las ideas palpables y nos ayuda a visualizar las posibles soluciones, poniendo de manifiesto elementos que debemos mejorar o refinar antes de llegar al resultado final.

Algunas técnicas: **Sketches, WireFrames, Prototipo, StoryBoard.**

Testea: Durante la fase de Testeo, probaremos nuestros prototipos con los usuarios implicados en la solución que estemos desarrollando. Esta fase es crucial, y nos ayudará a identificar mejoras significativas, fallos a resolver, posibles carencias. Durante esta fase evolucionaremos nuestra idea hasta convertirla en la solución que estábamos buscando.

Algunas técnicas: **Pruebas de Usuario, Test de Usabilidad.**

2.1.3. MVP: Diseñando la Propuesta de Valor

Como se ha explicado la metodología **Lean Startup** se basa en el Circuito de feedback de información: **crear, medir, aprender**. Lo que lleva al proceso de **Aprendizaje validado.**

Para crear nuestra propuesta de valor se ha decidido complementar con prácticas de **Design Thinking**: Empatiza, Define, Idea, Prototipa y Testea.

La mezcla de estas dos metodologías se conoce cómo **Lean Design** aunque actualmente está en desarrollo. Para entender mejor donde encajan estas dos metodologías disponemos de este gráfico.

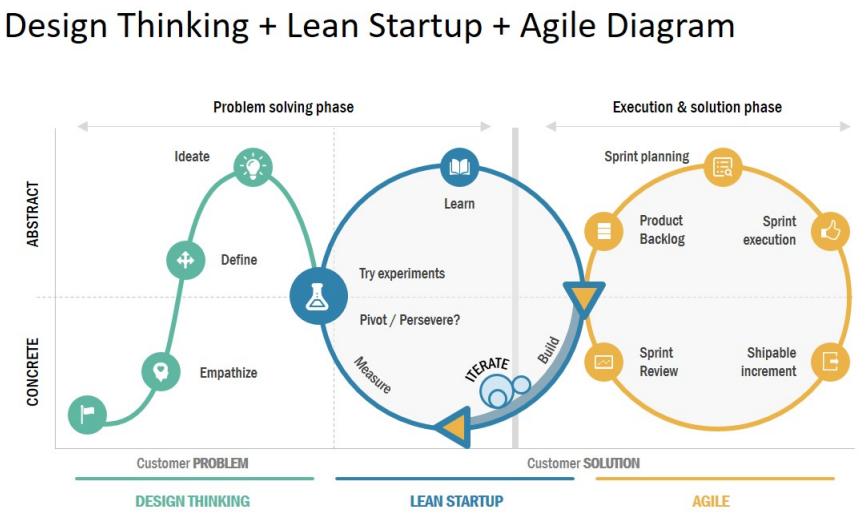


Figura 2.1: Diagrama de Lean Design

Con todo esto conseguimos un proceso que parte de la nada, se centra en el usuario, gestiona el caos, integra el error y es iterable.

Crear una **hipótesis**, diseñar un **experimento** para **testear** esa hipótesis, llevar a cabo el experimento, reunir datos, reflexionar y ver si validan o rechazan la hipótesis. Comenzamos:

CREAR

Lo primero de todo es definir nuestra **visión**. ¿Cuáles son nuestros objetivos?

El objetivo es conocer qué quieren los potenciales usuarios. Pero nos encontramos bajo unas condiciones de incertidumbre extrema. Nuestra tarea es diseñar para crear un nuevo producto o servicio bajo estas condiciones de incertidumbre extrema.

Es aquí donde necesitamos aplicar **Design Thinking:**

EMPATIZAR

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

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

Tenemos varias reuniones por Teams en las que, cómo músico, me explica las

dificultades por las que pasan a la hora de entrenar el oído y la importancia que el **entrenamiento auditivo** tiene.

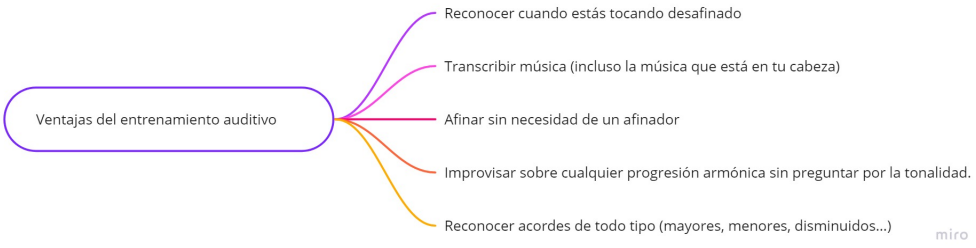


Figura 2.2: Ventajas del entrenamiento auditivo

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

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

DEFINIR

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

IDEAR

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

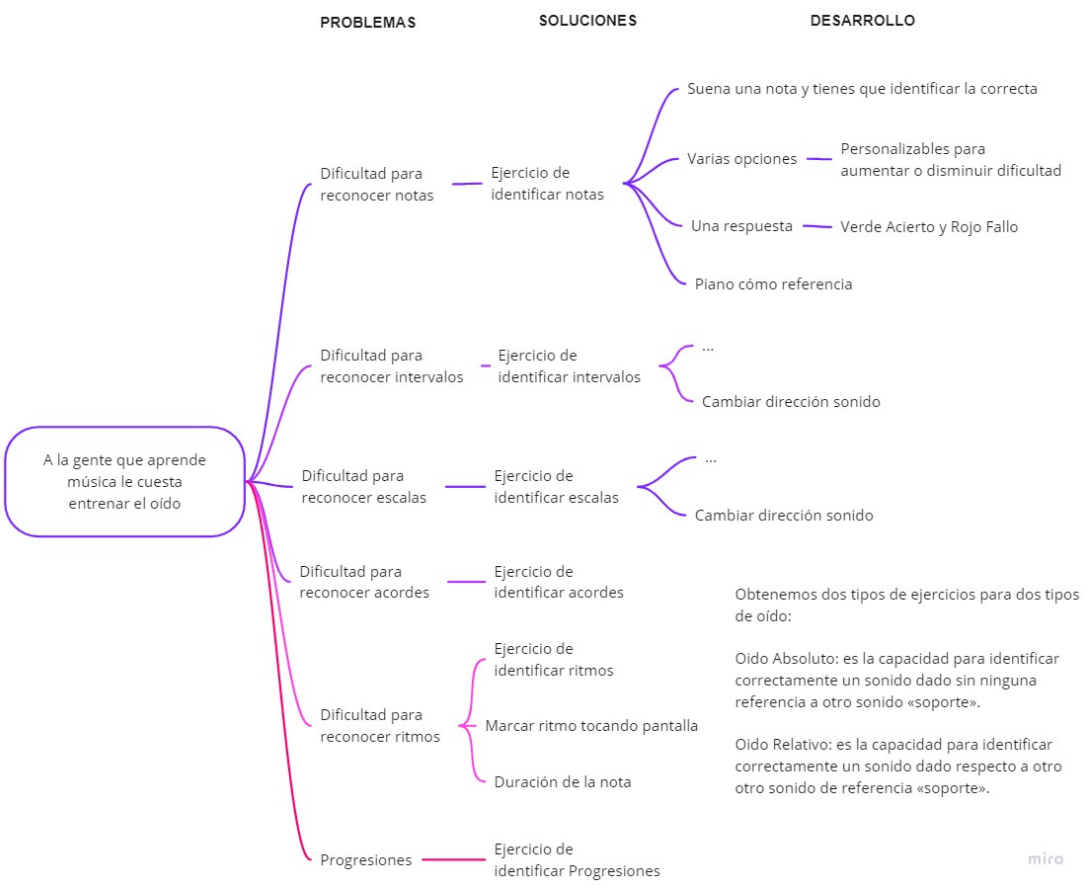


Figura 2.3: MindMap Parte 1

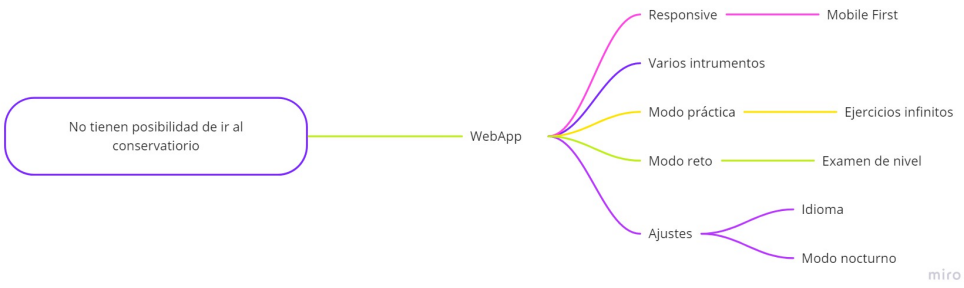


Figura 2.4: MindMap Parte 2

Después de varias iteraciones concluimos que una posible forma de solucionar estos problemas es mediante una WebApp con ejercicios de Entrenamiento Auditivo.

PROTOTIPAR

Es ahora cuando toca llevar a cabo nuestra solución para testear esta hipótesis.

Pero antes que nada debemos tener en cuenta nuestras limitaciones: poco tiempo para el desarrollo, aprender conceptos musicales, vamos a usar tecnologías que no conocemos. Y hacernos la pregunta: ¿Cuál es el **producto mínimo viable** (MVP) que puedo crear para obtener resultados?

Para ello, la técnica **MoSCoW** nos permitirá establecer las prioridades del proyecto. La cuál consiste en definir lo siguiente sobre la aplicación:

Must have (Debe tener): Características absolutamente críticas para el nuevo proyecto, sin ellas el proyecto es un fracaso.

- Modo Práctica
- Ejercicio Notas
- Ejercicio Intervalos
- Ejercicio Escalas

Should have (Debería incluir): Aspectos del proyecto críticos también, pero no imprescindibles.

- Opciones personalizables
- Varios instrumentos
- Piano
- Responsive

Could have (Podría incluir): Iniciativas que estaría bien tenerlas, ya que añadirían valor al proyecto, pero no son críticas.

- Ejercicio Acordes
- Ejercicio Ritmos
- Ejercicio Progresiones

Won't have (No se van a hacer): Características que no se merecen la inversión y no aportan ningún beneficio en este momento y se podrían considerar más tarde.

- Modo reto
- Modo nocturno
- Varios Idiomas

Este MVP no debe ser perfecto, lo que queremos es aprender lo más rápido posible. Añadir características a nuestro producto que todavía ni sabemos cómo va a ser aceptado es una perdida de tiempo. Por lo que vamos a centrarnos en desarrollar prioritariamente lo que debe tener y más tarde lo que debería incluir.

Una vez establecidas las prioridades del proyecto pasamos a visualizarlas en forma de Sketches en papel, y más tarde realizamos los Wireframes teniendo en cuenta la filosofía de diseño Mobile First. Tratamos de realizar un diseño simple

pero efectivo basado en tres columnas.

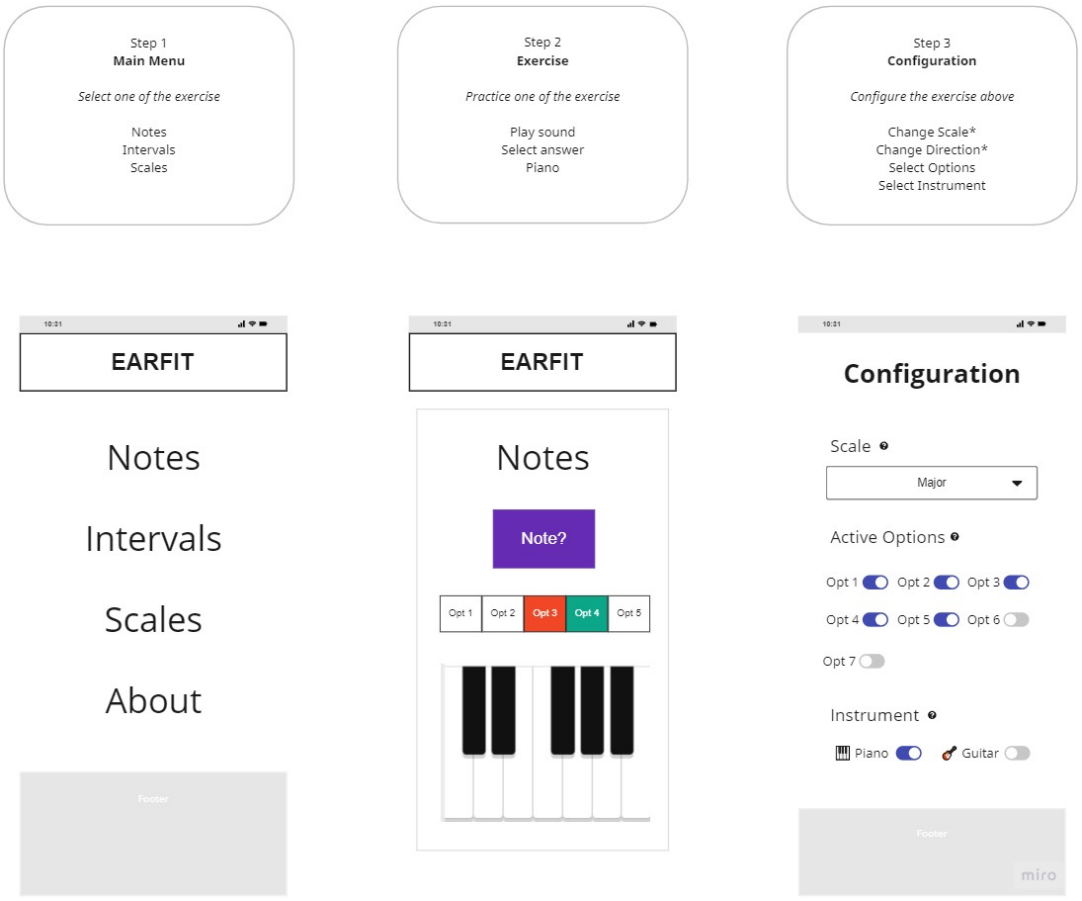


Figura 2.5: Prototipo Smartphone

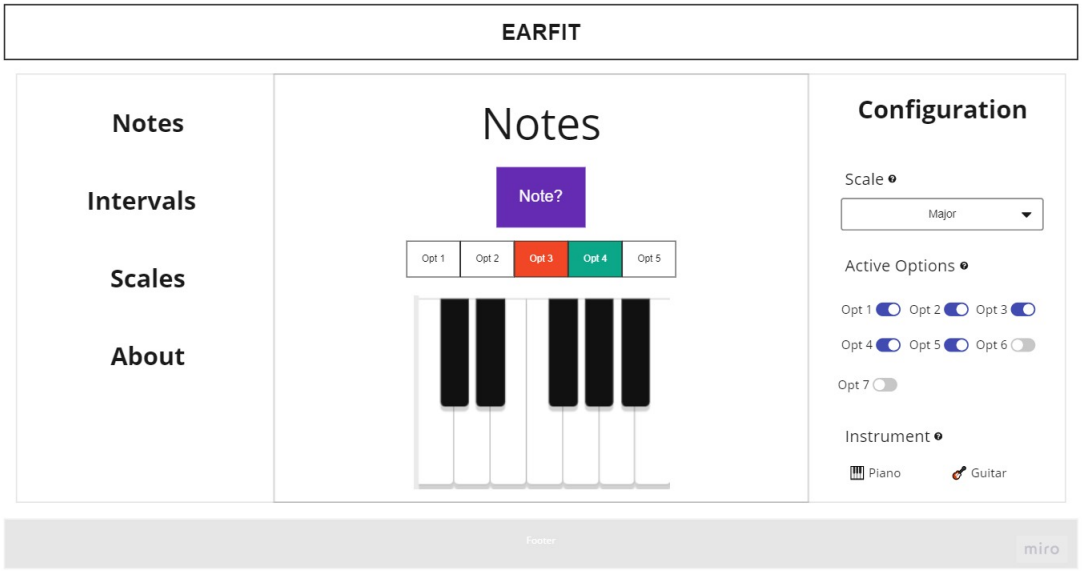


Figura 2.6: Prototipo PC

La ventaja principal del uso de **prototipos en UX o Experiencia de Usuario** es que, ayudan a plasmar visualmente los objetivos y como se alinean con las expectativas y necesidades de los usuarios y si son satisfechas a través del producto diseñado.

Modificar un prototipo, tiene un coste menor que modificar el producto digital una vez subido a producción, por lo que supone también una ventaja en el ahorro de costes dentro del proceso de desarrollo.

Ayuda a que el cliente ya tenga una visión sobre parte del proyecto y que nos pueda proporcionar su feedback desde las fases tempranas.

TESTEAR

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

Después de haber testeado los primeros diseños con el usuario, haber hecho correcciones y haber validado el último prototipo que hemos visto anteriormente, pasamos a crearlo usando **metodología Agile**.

En el siguiente capítulo se explicará esta metodología de trabajo seguida para implementar la solución. Y después de este los detalles técnicos de su desarrollo. Pero, por ahora seguimos con como fue el proceso de creación general.

MEDIR

Es hora de llevar a cabo el experimento, lo más común es poner los usuarios delante del producto y recoger así información sobre su comportamiento. Recoges los datos y reflexionas sobre ellos para **pivotar**, es decir, seguir adelante o cambiar de dirección.

Lo importante es qué tan rápido puedes desarrollar tus experimentos para hacer evolucionar la aplicación.

Los test fueron poco a poco en un proceso iterativo hasta llegar a completar los diseños del prototipo, generando siempre una versión entregable. Siguiendo la técnica del **Conserje**: empezando con un sólo usuario (Manuel Rubio) y una vez habiendo desarrollado la aplicación lo suficiente escalar y hacer pruebas con más usuarios.

La idea es ir implementando y testeando poco a poco la aplicación, empezando por los **Must haves** e ir añadiendo los **Should haves** paulatinamente y hacer ajustes según el feedback del usuario. Para dar respuesta a las siguientes preguntas:

¿Tiene la gente el problema que crees que tienen? ¿Realmente quieren lo que estas ofreciendo?

Para ello, cada vez que se desarrollaba una **nueva versión** de la aplicación. Se testeaba en una reunión con el tutor recogiendo **feedback** y haciendo los ajustes pertinentes. Más tarde se empezó a testear también con conocidos, en concreto dos estudiantes de conservatorio y una persona que está empezando a tocar. Es cuando se empezaron a aplicar algunas **métricas**:

Las que mejor se adecuaron a nuestro producto son: **Engagement** y **Tiempo en el producto** por usuario. También se tuvo en cuenta la **retención**: el usuario lo usa nuevamente, y la **referencia**: el usuario comparte el producto con sus amigos.

Además se utilizaron **test A/B**: enseñando dos versiones diferentes del producto al mismo tiempo para tomar decisiones acertadas. Aprovechando las características que nos ofrecía **Vercel** como veremos en su apartado.

Aclarar que aparte de estos test que **miden el éxito** de la aplicación se realizaron pruebas también relacionadas al funcionamiento cómo se verá más adelante en su apartado **Testing** en Desarrollo e Implementación.

APRENDER

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

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

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

En un principio no se incluían todos los **intervalos existentes**, sólo los de notas naturales, un test con el tutor nos hizo darnos cuenta de que faltaban las notas alteradas. Lo que llevo también a una investigación para corregir el nombre de estos, ya que descubrimos que dos intervalos con diferente nombre pueden sonar igual. Esto es debido a que los intervalos se nombran no sólo por la distancia de sus notas sino también por cómo están escritas en el pentagrama. Por ejemplo: entre Do y Dos hay 1 semitono y entre Do y Reb también hay 1 semitono, la misma distancia, suenan igual, pero son intervalos distintos.

También nos dió el feedback de que estaría bien añadir un **sonido al acertar**. Aprovechando el sonido de la misma respuesta para que así los usuarios puedan interiorizarla.

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

En un principio se mostraban todas las opciones disponibles, lo que hacía el ejercicio muy difícil. Por lo que dejando como predeterminado tres opciones se ajustaba la **dificultad** inicial. Además descubrimos de que resultaba engorroso una vez señaladas varias opciones volver a seleccionarlas por lo que añadimos un botón que seleccionase y deseleccionase todas de golpe.

También descubrimos que había un problema de **compatibilidad** con iphone que hacía inservible la aplicación el cual pudimos corregir rápido. Así cómo poder ajustar el **volumen** de los sonidos ya que al principio no sonaba lo suficientemente alto.

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

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

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

CONCLUSIÓN

Una vez pasado por todo este proceso iterativo, el resultado es un MVP que poder sacar al mercado. Y que se debe seguir mejorando.

Los **Early Adopters**, los primeros en utilizar el producto no serán muy exigentes con la calidad pero esto es así con los nuevos clientes que vendrán más tarde.

Todo se basa en un proceso de **Innovación Sostenida**, que se basaría en ir incrementando mejoras al producto ya existente.

Para ver capturas de la aplicación tras todo este proceso puedes ir al apartado **Resultado Final**

2.2. Metodología de Trabajo

¿Cómo gestionamos el desarrollo?

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

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

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

2.2.1. DevOps

El término DevOps, que es una combinación de los términos ingleses **development** (desarrollo) y **operations** (operaciones), designa la unión de personas, procesos y tecnología para ofrecer valor a los clientes de forma constante.

DevOps describe los enfoques para agilizar los procesos con los que una idea

(como una nueva función de software, una solicitud de mejora o una corrección de errores) pasa del desarrollo a la implementación, en un entorno de producción en que puede generar valor para el usuario.

La creación de registros de trabajo pendiente, el seguimiento de los errores, la administración del desarrollo de **software ágil** con **Scrum**, el uso de paneles **Kanban** y la visualización del progreso son algunas de las formas en las que los equipos de DevOps planean con agilidad y visibilidad.

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

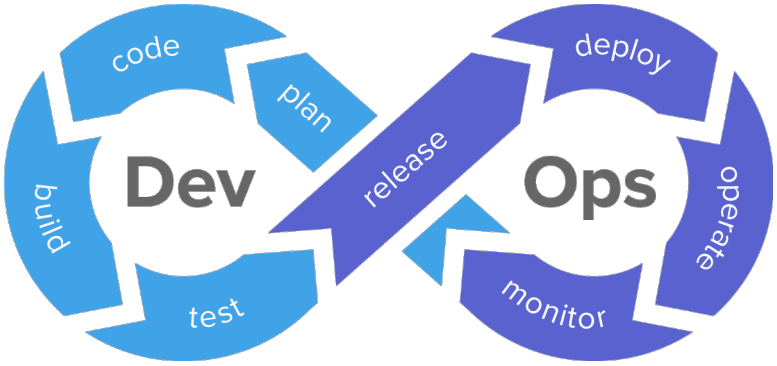


Figura 2.7: DevOps

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

El primero de los 12 principios del Manifiesto Ágil es el siguiente: "Satisfacer a los clientes mediante la distribución de software continua y oportuna". Este es el motivo por el que es importante aplicar prácticas de DevOps, cómo la integración continua y el despliegue continuo.

Integración Continua (CI)

La **integración continua** es una práctica de DevOps mediante la cual los desarrolladores combinan los cambios en el código en un repositorio central de forma periódica. Para implantar **integración continua** solemos definir un "pipeline", un conjunto de etapas, de fases por las que va pasando el software y que se automatizan.

En nuestro caso usamos **Github**, que es un servicio basado en la nube, cómo herramienta de control de versiones **Git**. Y utilizamos **Gitflow*,***

que es un modelo alternativo de creación de ramas en Git en el que se utilizan ramas de función y varias ramas principales.

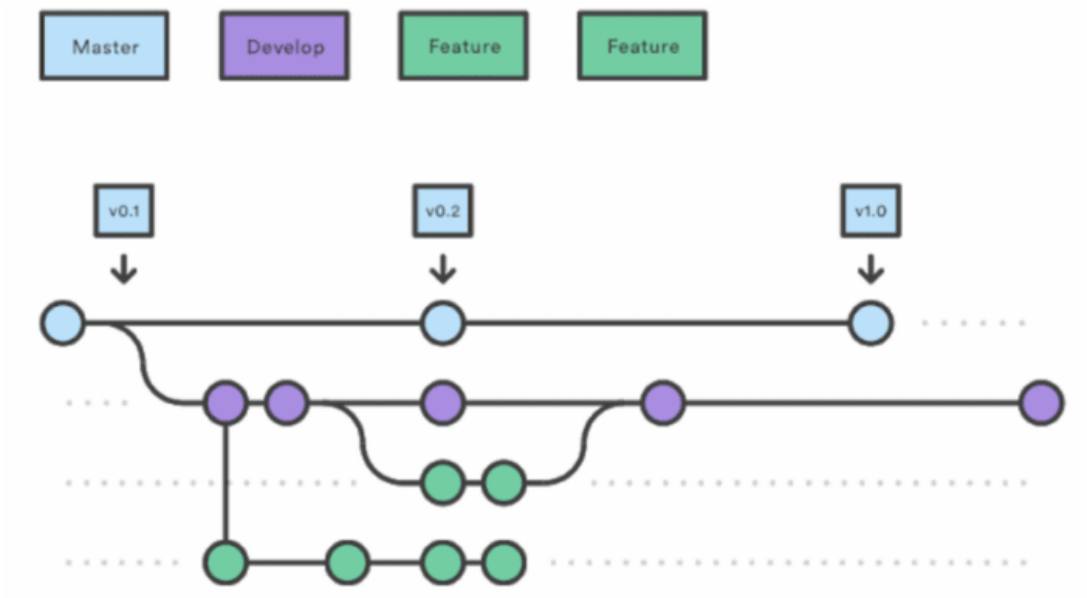


Figura 2.8: Gitflow

Su funcionamiento se basa en dos ramas principales: la rama de producción y la rama de desarrollo. De la rama de desarrollo se van sacando nuevas ramas de funcionalidades y uniendo a está a medida que se completan. Un vez una nueva versión está lista se une la rama de desarrollo con la de producción generando una nueva release.

Despliegue Continuo (CD)

El **despliegue continuo** es una estrategia de DevOps en la que los cambios de código de una aplicación se publican automáticamente.

Para entregar funcionalidades de software de forma frecuente a través de la automatización de despliegues utilizamos **GitHub Actions** el cuál nos permite automatizar, personalizar y ejecutar estos flujos de trabajo directamente desde nuestro repositorio.

El flujo consistía en que cada vez que se actualizaba el repositorio mediante una subida de código, éste automáticamente desplegaba las nuevas actualizaciones en **Vercel,** que ****es una plataforma en la nube que permite a los desarrolladores alojar sitios web y servicios web que escalan automáticamente y no requieren supervisión.

2.2.2. Scrum

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

Con Scrum, un producto se basa en una serie de iteraciones llamadas **sprints** que dividen proyectos grandes y complejos en porciones minúsculas. Priorizadas por el beneficio que aportan al receptor del producto.

1. Un Product Owner ordena el trabajo de un problema complejo en un Product Backlog.

2. El Equipo Scrum convierte una selección del trabajo en un Incremento de valor durante un Sprint.

3. El Equipo Scrum y sus partes interesadas inspeccionan los resultados y se ajustan para el próximo Sprint.

4. *Repetir*

Scrum no es un proceso o una técnica para desarrollar/construir productos, realmente es un marco de trabajo donde podemos emplear un conjunto de diferentes procesos y técnicas, siendo muy fácil de implantar y muy popular en el desarrollo software por los resultados rápidos que consigue.

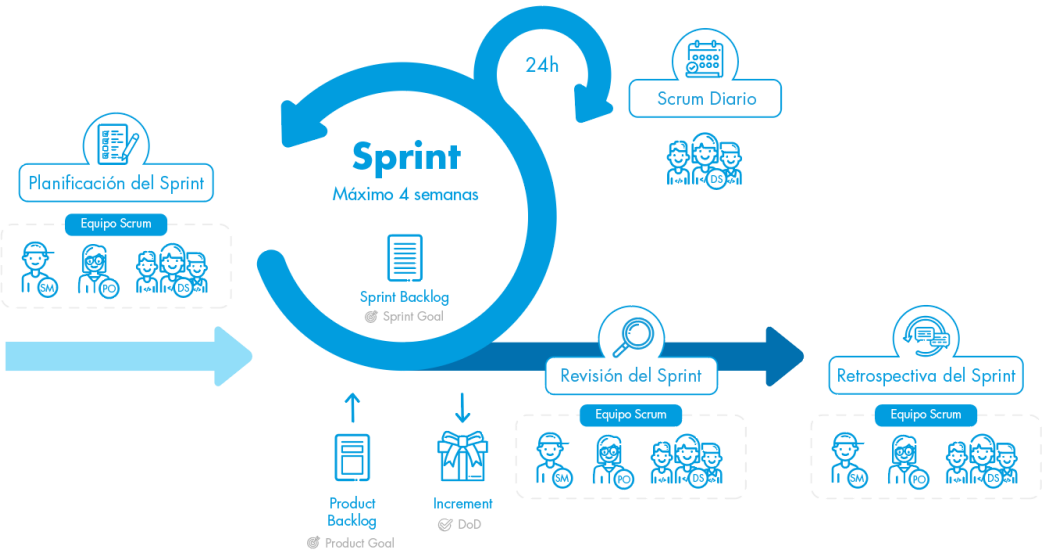


Figura 2.9: Scrum

Implementar este marco de trabajo nos permite **obtener resultados pronto**, reduciendo el **Time to Market**, es decir, a tener lo antes posible en el mercado nuestro producto o una característica de nuestro producto, aumentando la **satisfacción del cliente**. Donde los requisitos son cambiantes o poco definidos, **aportando tolerancia al cambio.**

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

Para visualizar el trabajo utilizamos **Scrum Board**. Esto nos permite separar en pequeñas tareas cada Product Backlog Item y minimizar los riesgos de entrega. Realizando una construcción iterativa incremental donde se integra al final de cada sprint para validar el resultado.

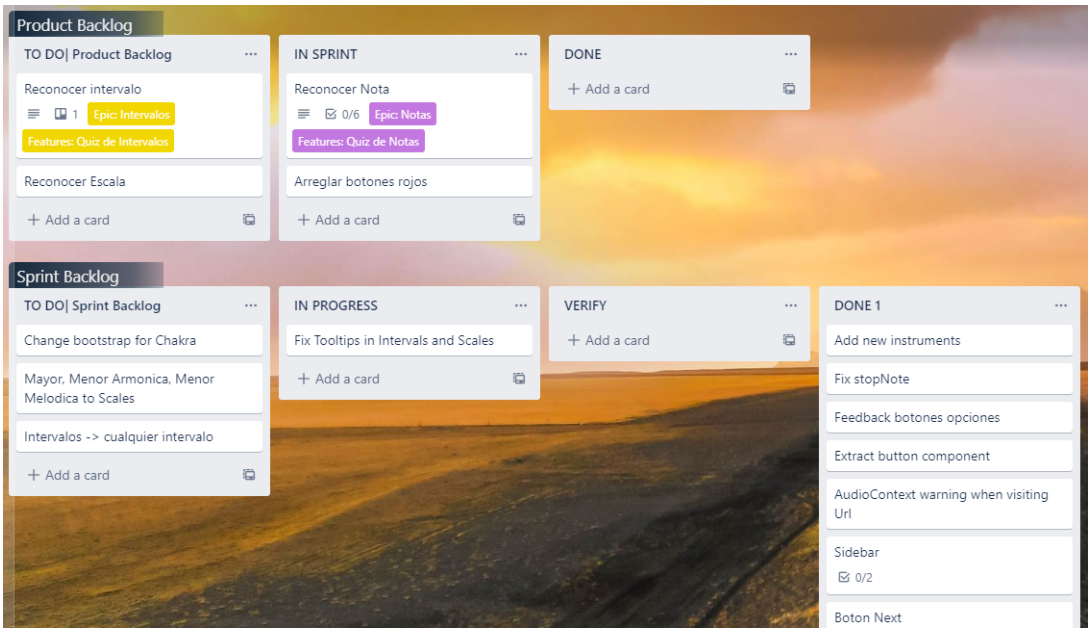


Figura 2.10: Kanban

Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Lo que nos permite aprovechar el cambio y mejorar continuamente.

2.3. Desarrollo e Implementación

¿Cómo implementamos la solución?

La aplicación fue implementada siguiendo unas guías de estándares de calidad de código y un moderno stack tecnológico. A continuación se encuentra una deta-

llada descripción informática incluyendo especificación, diseño, implementación y pruebas.

2.3.1. Stack Tecnológico

Después de una larga investigación y comparación de tecnologías teniendo en mente las necesidades del proyecto. Se llegó a la conclusión de que estás herramientas, frameworks y lenguajes eran los necesarios para implementar esta aplicación.

VSCode

Visual Studio Code es un editor de código redefinido y optimizado para crear y depurar aplicaciones web y en la nube modernas. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

Gracias a su integración de Git es sencillo revisar las diferencias, preparar los archivos y realizar confirmaciones directamente desde el editor antes de subirlo al repositorio en la nube.

Además cuenta con una biblioteca de extensiones, las cuales nos da un sin número de opciones para ser más eficiente a la hora de estar programando. Desde extensiones de otros lenguajes de programación como diferentes herramientas para visualizar o estructurar el código de manera más eficiente.

Las extensiones más importantes para mejorar la experiencia de desarrollo y buenas prácticas de este proyecto fueron las que se explican a continuación. Configurar estas herramientas será una inversión que haremos una vez y sus beneficios los notaremos durante todo el proyecto

ESLint

ESLint es una herramienta de análisis de código para identificar patrones problemáticos que se puede instalar como extensión en VSCode.

Su función es analizar el código de nuestra aplicación, detectar problemas por medio de patrones y si está a su alcance resolverlos. Cuenta con unas reglas por defecto pero en este caso lo vamos a configurar con una guía de estilo JavaScript llamada **StandardJS**. Lo que nos permitirá:

- Corregir errores de sintaxis
- Corregir código poco intuitivo o difícil de mantener

- Evitar el uso de "malas practicas"
- Hacer uso de un estilo de código consistente.

ESLint está diseñado para ser flexible y configurable por lo que añadimos ejecutar ESLint como parte del proceso de integración continua.

Prettier

Prettier es un formateador de código que puede instalarse como extensión en VSCode. Aplica un estilo consistente analizando el código y formateándolo con sus propias reglas que toman en cuenta la longitud máxima de línea, ajustando el código cuando es necesario.

Para ello, analiza el código y lo **da formato** cada vez que se guarda el archivo. Su objetivo es acabar con los debates sobre el estilo del código.

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

Next.js

TypeScript

Node.js

2.3.2. Implementación

Code Guidelines

Análisis y Diseño

Librerias Node.js

2.3.3. Testing

Vercel Metricas

Pruebas Unitarias

Evaluación de la Interfaz

2.4. Resultado Final

Explicacion y capturas

3

Conclusiones

En este capítulo se detallan las conclusiones derivadas del TFG y la propuesta de posibles trabajos futuros.

Las citas del texto Autor [1], Autor [2], Autor [3], Autor [4] y Autor [5] deben ir referenciadas en la bibliografia.

Qué he aprendidos Qué debo mejorar

Bibliografía

[1] M. Giaquinta and S. Hildebrandt, *Calculus of variations II.* Springer Science and Business Media, 2013, vol. 311.

[2] S. Fortune and C. J. Van Wyk, "Efficient exact arithmetic for computational geometry," in *Proceedings of the Ninth Annual Symposium on Computational Geometry*, 1993, pp. 163–172.

[3] S. Fortune, "Voronoi diagrams and delaunay triangulations," *Computing in Euclidean geometry*, pp. 225–265, 1995.

[4] J. C. Mitchell, "Social networks," *Annual review of anthropology*, vol. 3, no. 1, pp. 279–299, 1974.

[5] C. B. Morrey Jr, *Multiple integrals in the calculus of variations.* Springer Science and Business Media, 2009.

Apéndices

A

Conceptos Musicales

A.1. Notas

Sección del apéndice

A.2. Intervalos

A.3. Escalas