

# Memoria Proyecto Fin de Ciclo JamBoree

Miguel Ángel Morcillo | Iván Gómez | Said El Mourabet

## Contenido

INTRODUCCIÓN .....	3
¿Cómo va a funcionar Jamboree y qué nos permitirá hacer? .....	3
Página de inicio o Index .....	4
Página principal o main .....	4
Foro .....	4
Audioteca .....	4
Estudio .....	5
¿A quién va dirigido? .....	5
PLANIFICACIÓN PREVIA.....	6
Roles.....	6
Metodología.....	6
¿Qué es scrum? .....	6
¿QUE ES JIRA Y COMO SE APLICA A SCRUM? .....	7
ANÁLISIS .....	11
Requisitos generales.....	11
¿Qué va a ser capaz de hacer Jamboree? .....	11
¿Qué necesita la aplicación para funcionar? .....	11
¿Qué va a usar la aplicación para ofrecer todo su potencial? .....	12
Casos de uso .....	13
DISEÑO .....	15
Arquitectura .....	15
TONE JS .....	16
Aspectos esenciales de tone.js.....	16
Audio en tone.js.....	16
Trato de datos en JamBoree estudio .....	18
Conceptos base .....	18
Creación de track desde cero .....	18
Carga de un track ya existente .....	19
Reproducción del sonido.....	20
Guardado del track generado .....	20
Trato de datos y flujo en audioteca y foro .....	21
Modelo de datos (E/R, tablas, etc.) .....	22
Interfaz gráfica .....	25
Representación gráfica del audio .....	25
Colores .....	29

Tipografía .....	29
Preprocesador LESS .....	29
Diferencias entre el diseño inicial y el diseño actual.....	29
DESPLIEGUE.....	30
Local.....	30
Cloud .....	31
Máquina virtual Ubuntu .....	31
Instancia de MySQL.....	31
Configuración de la máquina Ubuntu Server .....	32
Características futuras.....	33
Implementación de comentarios .....	33
Colaboración musical online .....	33
Capacidad de introducir canciones en álbumes.....	33
Perfil con avatar gráfico .....	34
Introducir nuevos sonidos y teclado .....	34
Exportación de temas en formato MP3/WAV.....	34
Exportación en formatos para software de Digital Audio Worsktation.....	34

## INTRODUCCIÓN

Jamboree (/ˌdʒambəˈri:/), del término inglés cuyo significado se traduce como “francachela”, o ‘juerga con diversión desmesurada’.

Es una aplicación web diseñada para el uso de todos aquellos músicos artistas que quieran crear y compartir música y también para aquellos usuarios que, sin necesidad de tener una formación ni conocimientos musicales avanzados, puedan revisar música de cualquier estilo y disfrutar de ella.

Como aplicación nos aporta un gran abanico de utilidades; ya sea utilizar el foro para escuchar música de cualquier tipo pudiendo filtrar por etiquetas, crear uno mismo sus propios temas con un estudio online propio e incluso poder replicar canciones de otros artistas en el estudio y modificarlas a tu estilo.

### ¿Cómo va a funcionar Jamboree y qué nos permitirá hacer?

Jamboree se trata de una página web que actúa como foro musical y estudio musical. Teniendo así una plataforma que permite a los usuarios trabajar únicamente en la página web.

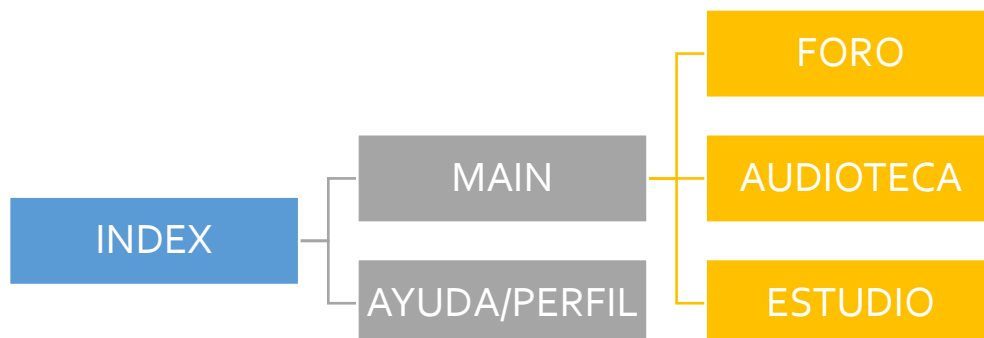
Esta página Web está estructurada inicialmente como una propia red social donde el usuario deberá iniciar sesión o registrarse si no lo está (**INDEX**).

Una vez éste consiga acceder a la página web tendrá la opción de elegir si va a escuchar música navegando en (**FORO**) o (**AUDIOTECA**), o si va a crear música (**ESTUDIO**).

- En el primer caso si el usuario únicamente quiere escuchar música entrará al foro para ver las creaciones de los diversos usuarios y teniendo éste la capacidad de elegir por etiquetas el estilo de música que desee escuchar.
- En el caso de que quiera crear música el usuario o editar alguna canción que tenga sin terminar podrá elegir la opción de ir a su audioteca y cargar algún archivo o ir al estudio y empezar de o.

De forma textual parece bastante complejo, pero de forma gráfica estos son los diversos caminos que puede optar el usuario:

De forma más detallada podríamos dividir la página web en las siguientes partes:



## Página de inicio o Index

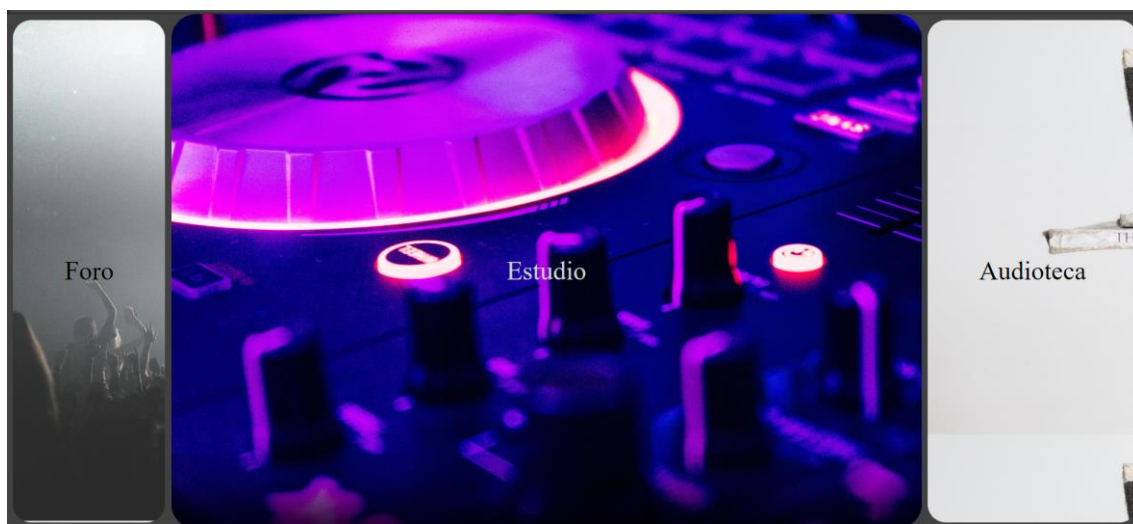
Consiste en una página de inicio de sesión como la de cualquier red social siendo familiar al usuario y aportando una primera impresión al usuario del potencial de Jamboree.

Contiene un video promocional que muestra el funcionamiento de la web para que un usuario nuevo pueda saber qué es antes de registrarse.

Una vez el usuario decida entrar tendrá que loguearse o registrarse dependiendo si es nuevo o no. En el caso de registrarse será redirigida a una página con un formulario para que pueda introducir sus datos e iniciar sesión.

## Página principal o main

La página principal o main es la página que se ve una vez se haya iniciado sesión. Esta está formada por 2 secciones en grande y uno de ellos subdividido en dos que nos permitirá elegir diversas opciones según lo que quiera el usuario. Gráficamente es:



### Foro

El Foro ofrecerá al usuario las últimas canciones creadas por la comunidad. Además, da la opción al usuario de filtrar mediante etiquetas el género musical que desee.

En un aside aparecerán los temas y artistas con más escuchas.

Cada entrada(tema) permite ser votada y añadida a tu audioteca personal para obtener una copia de ese tema y poder modificarlo en tu estudio.

### Audioteca

La audioteca ofrece todas las pistas guardadas del usuario, ya sean provenientes del foro o de su estudio propio.

Estas pistas pueden ser cargadas en el estudio para poder modificarlas o incluso subirlas al foro.

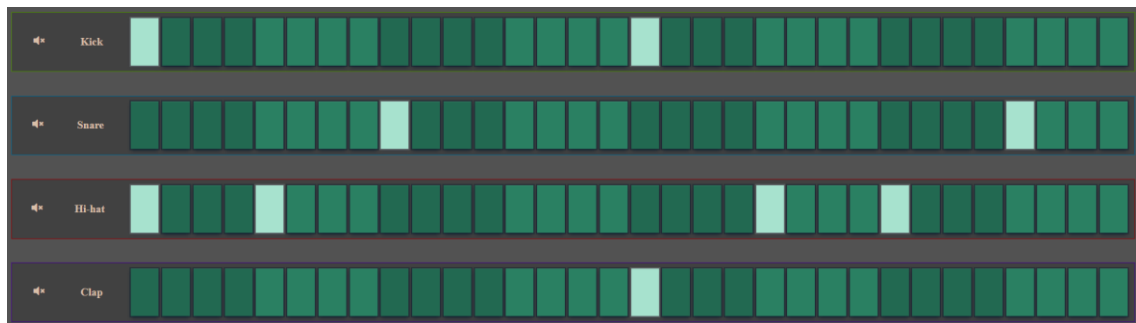
## Estudio

El estudio estará compuesto por un "channel rack" para poder crear diversos loop que luego se podrán subir al foro, guardar como archivo mp3 o guardarlo en la audioteca para modificarlo más tarde.

El channel rack se compone de 4 pistas de percusión: bombo, plato, caja y palmas.

Cada una de las pistas contiene 2 compases de 4/4. Así como 32 botones representando cada uno un tiempo, un beat. Al ser pulsado cada uno de estos botones indicará que ese beat de esa pista habrá de reproducirse en el loop.

Para controlar la reproducción se muestra un botón para reproducir y pausar, otro para limpiar los botones pulsados y empezar de cero, y un último botón para guardar el loop en la audioteca del usuario.



## ¿A quién va dirigido?

Jamboree no se trata de una página web dirigida a gente que necesite conocimientos amplios de la música si no a amantes de la música, artistas e incluso a productores musicales.

Simplemente con gustarte la música ya puedes sacar muy buen rendimiento de la página web puesto que el foro debe tener pistas suficientes para encontrar música nueva y nunca escuchada, siendo simples demos de artistas con un simple ordenador y una conexión a internet. Esto abre un mundo de posibilidades y reúne a un amplio grupo de público desde jóvenes que escuchan música, creadores profesionales de música, usuarios que simplemente desean compartir su música con amigos incluso managers y discográficas buscando nuevos talentos.

## PLANIFICACIÓN PREVIA

### Roles

- Iván Gómez:
  - Modelado de las bases de datos
  - Diseño de headers en todas las páginas
- Said El-Mourabet
  - Creación de base de datos
  - Diseño de interfaces en index
  - Gestión de usuarios
- Miguel Ángel Morcillo
  - Diseño y desarrollo del estudio
  - Diseño de representación visual del sonido en audioteca y foro
  - Diseño de interfaz de página "main"
  - Desarrollo de parte servidora.

### Metodología

La metodología que se ha seguido para la construcción de Jamboree ha sido la metodología Scrum. Siendo ésta una metodología ágil que permite al equipo organizarse y trabajar de una forma sencilla.

#### ¿Qué es scrum?

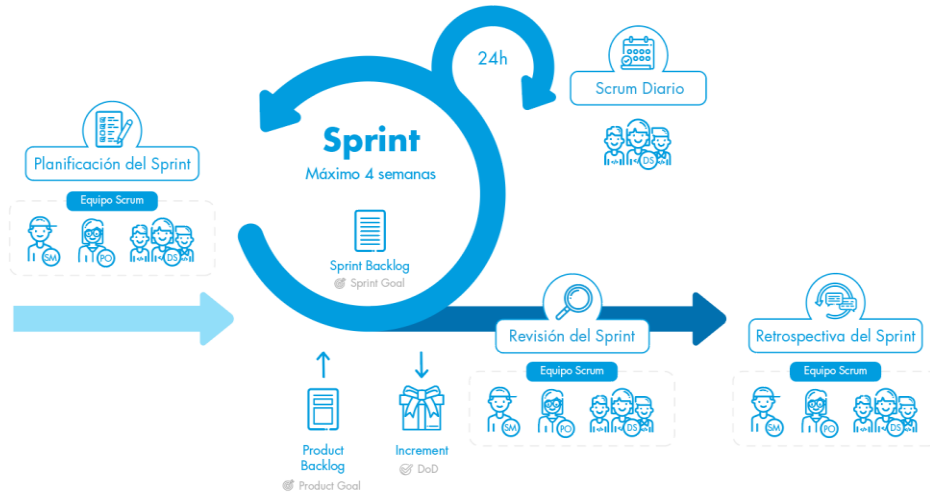
Scrum es una metodología ágil que se utiliza para minimizar la necesidad de reuniones no definidas en Scrum y establecer un equilibrio que permita al equipo fomentar la comunicación y colaboración reduciendo el tiempo en reuniones extensas.

Para conseguir esto todas las tareas tienen una caja de tiempo en cada sprint y una vez se inicia el sprint este tiene una duración fija. Puede durar cada sprint entre dos o cuatro semanas teniendo una reunión diaria de 10 min como mucho y una reunión para valorar el sprint y otra para adoptar las tareas que van a afrontar en el siguiente sprint.

En nuestro caso, la duración de cada sprint es de dos semanas, coincidiendo con cada reunión quincenal con nuestro tutor del proyecto. Además, sabiendo que cada uno de los tres integrantes tenemos horarios y circunstancias diferentes debido a las prácticas en el centro de

trabajo, la reunión diaria la hemos agilizado haciendo tanto reuniones virtuales en Discord como por texto en grupo de Whatsapp.

Al terminar cada reunión quincenal con el tutor del proyecto, nos hemos reunido para repasar las tareas que quedaran pendientes del anterior sprint, planificar el nuevo sprint y definir el objetivo para esas dos semanas siguientes.

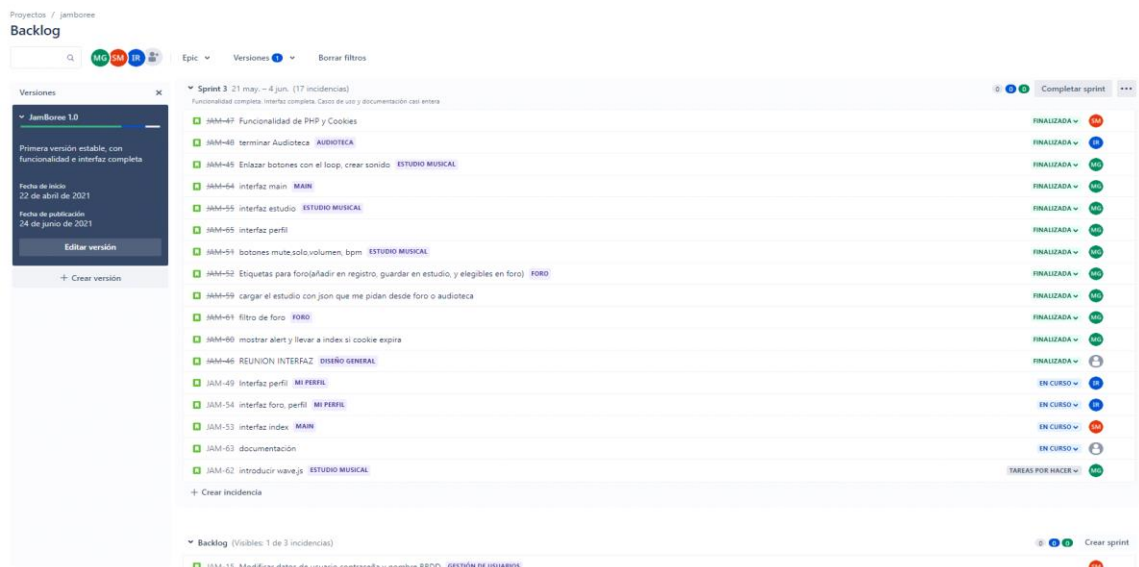


Existen diversas apps que permiten aplicar Scrum como es el caso de MeisterTask, Clarizen o Wrike. En nuestro caso hemos usado Jira, de Atlassian, que al igual que las anteriores es una herramienta para ayudar a equipos de todo tipo a gestionar el trabajo, y que además en su versión gratuita nos aporta todo lo necesario para trabajar con ella.

### ¿QUE ES JIRA Y COMO SE APLICA A SCRUM?

Con Jira hemos aplicado el método scrum dividiendo los sprint en cada dos semanas de tal forma que coincida con las tutorías para definir los objetivos con el tutor.

Antes de empezar a crear la web app se realiza un backlog para saber que tareas van a tener que realizarse durante el desarrollo de Jamboree. De esta forma se organizan las tareas y se crean mini tareas para poder trabajar de forma eficiente. Además, estas tareas no son fijas, si no que irán evolucionando y surgiendo otras tareas a medida que transcurra el sprint.





Para clasificar las tareas creamos varios Epic de las categorías principales, siendo estos:

- Gestión de usuarios
- Estudio musical
- Foro
- Perfil
- Ayuda
- Index
- Diseño general.

De tal forma podemos observar cuáles son las partes de la página que más trabajo requieren y cuales estan más desarrolladas.

Epic	ABR	MAYO	JUN
	Sprint 1		
> JAM-8 Gestión de usuarios			
> JAM-6 Estudio musical			
> JAM-5 Foro			
> JAM-7 Mi perfil			
JAM-34 Ayuda			
> JAM-37 Index			
> JAM-38 Diseño general			

Para medir la evolución de la funcionalidad de la aplicación creamos 3 diferentes versiones o releases, en la que la 1.0 será la aplicación con funcionalidad básica e interfaz básica, la 2.0 funcionalidades añadidas, la 3.0 funcionalidad completa y diseño de interfaz completo y la 4.0 retoques estéticos mínimos, documentación y perfección de Jamboree.

### Sprint 1.0

En el sprint de la primera versión gran parte de las tareas tenían como fin crear una estructura simple de la página para poder introducir pequeñas funcionalidades que permiten trabajar con la página y navegar sobre ella. Como por ejemplo el manejo de usuarios y la creación de la estructura del main, index, foro, estudio y audioteca.

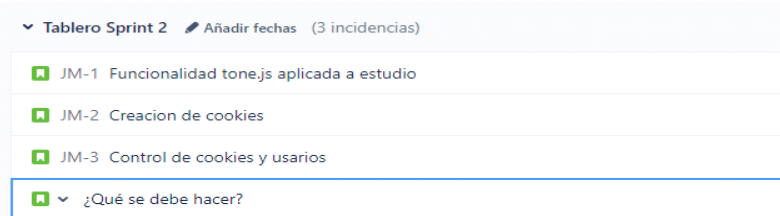
T	Clave	Resumen	Responsable	Informador	Pv	Estado	Resolución	Creada	Actualizada
	JAM-44	Auth para login	Said El Mourabet	Miguel Ángel Morcillo Gutiero	↑	FINALIZADA	Listo	02/may/21	13/may/21
	JAM-42	Estructura Foro	IVAN GOMEZ RODRIGUEZ	IVAN GOMEZ RODRIGUEZ	↑	FINALIZADA	Listo	26/abr/21	11/may/21
	JAM-36	Estructura main.html	Said El Mourabet	Said El Mourabet	↑	FINALIZADA	Listo	26/abr/21	03/jun/21
	JAM-35	Estructura index.html	Said El Mourabet	Said El Mourabet	↑	FINALIZADA	Listo	26/abr/21	11/may/21
	JAM-33	Como guardar canciones con la API?	Miguel Ángel Morcillo Gutiero	Miguel Ángel Morcillo Gutiero	↑	FINALIZADA	Listo	24/abr/21	02/may/21
	JAM-32	Estructura estudio.html	Miguel Ángel Morcillo Gutiero	Miguel Ángel Morcillo Gutiero	↑	FINALIZADA	Listo	24/abr/21	11/may/21
	JAM-20	Enlaces redes sociales	Said El Mourabet	IVAN GOMEZ RODRIGUEZ	↑	FINALIZADA	Listo	22/abr/21	20/may/21
	JAM-19	Logo	Said El Mourabet	IVAN GOMEZ RODRIGUEZ	↑	FINALIZADA	Listo	22/abr/21	20/may/21

### Sprint 2.0

En el segundo sprint se trabaja gran parte de la funcionalidad del estudio utilizando y aplicando tone.js para poder realizar canciones y tener nuestra principal herramienta de trabajo lista y que pueda usarse en toda la web.

La tarea JM-1 se dividió en subtarefas dada la complejidad de la librería Tone JS, teniéndose que seguir un recorrido de estudio y ensayo y error hasta lograr implementar la funcionalidad deseada con esta herramienta.

También se estructura la base de datos y se realiza un control de las cookies.



### Sprint 3.0

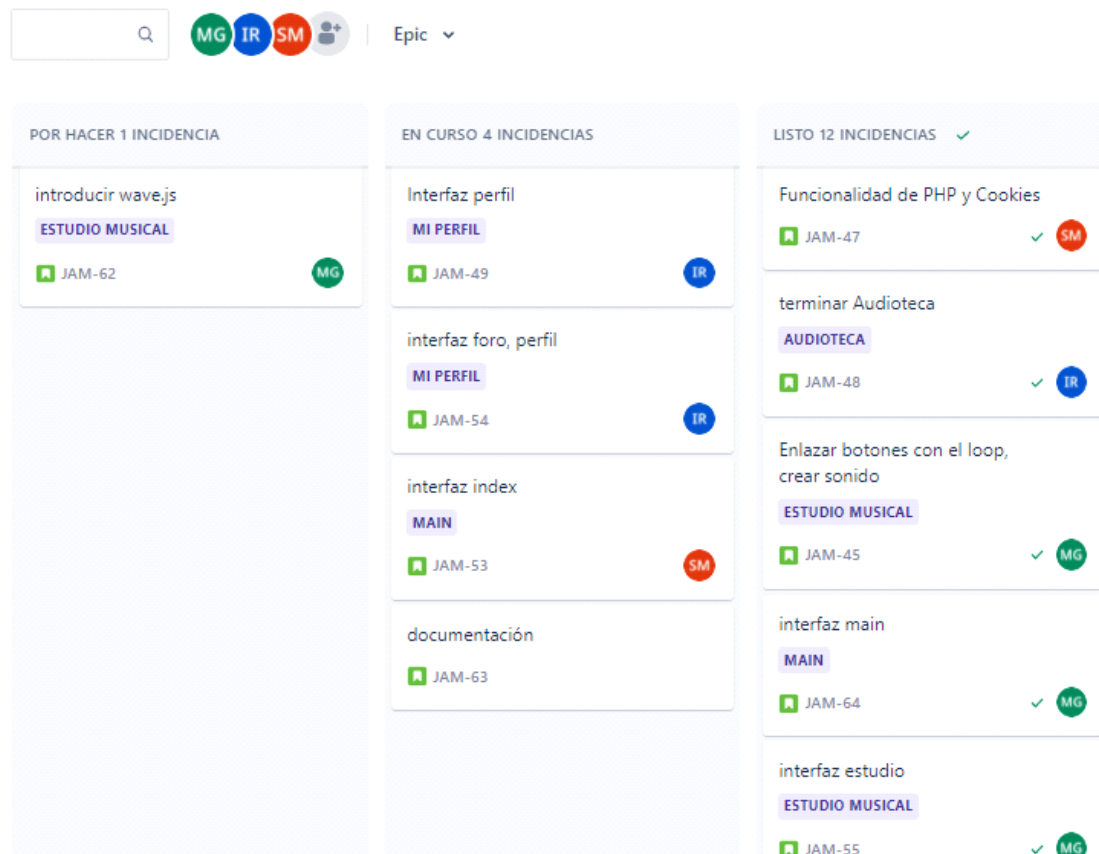
En el tercer sprint se ha realizado gran parte de la documentación y se ha pulido completamente el flujo de la página y sus datos de forma que la funcionalidad de la página se ha completado.

El diseño de la interfaz general es parte importante del Sprint, sin llegar a tener que estar finalizado aún, pero avanzando gran parte el aspecto final.

Se intentó introducir una librería nueva: Wave JS. Para implementar gráficos de las ondas de audio, pero su alta complejidad y la imposibilidad de exportar el audio que se crea en el loop terminó por hacer que buscáramos una solución alternativa más manual (explicada en pág .

### Sprint 3

Funcionalidad completa. Interfaz completa. Casos de uso y documentación casi entera

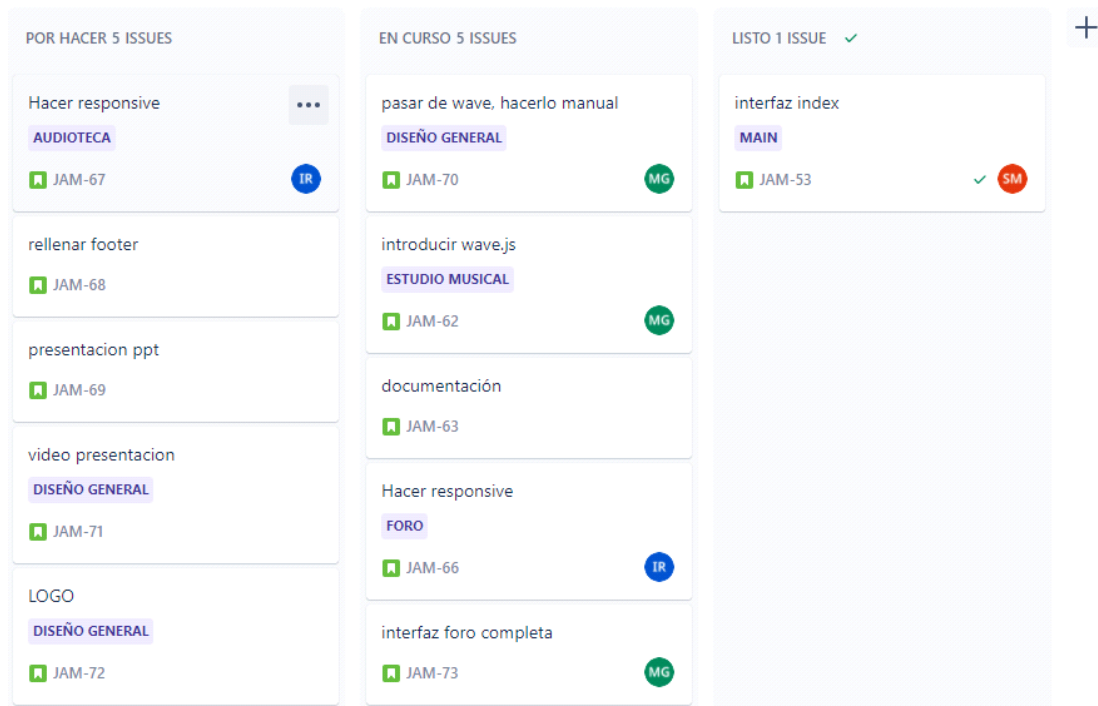


### Sprint 4.0

El cuarto sprint tuvo como objetivo finalizar la interfaz gráfica de toda la aplicación, incluyendo por fin los aspectos responsive y ultimación de detalles visuales.

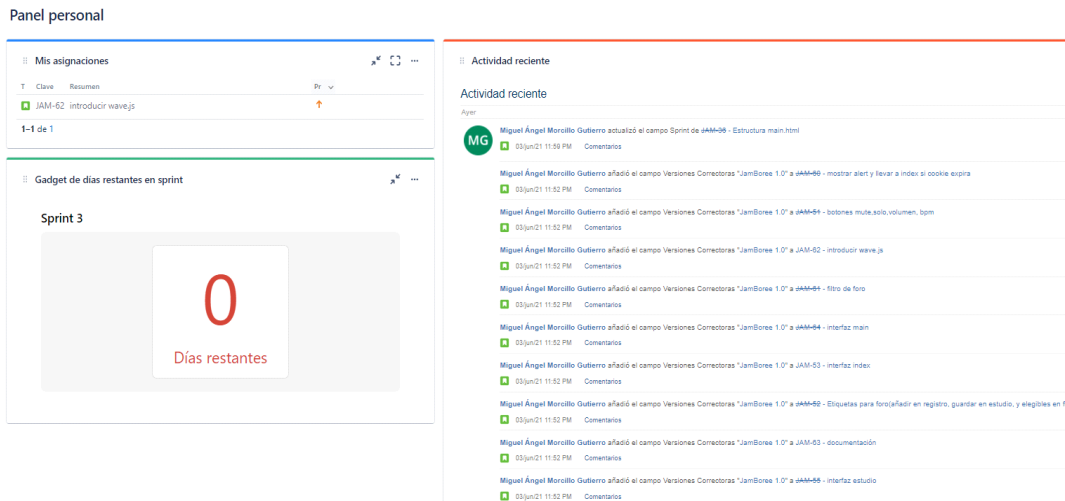
Respecto a la funcionalidad, sirvió como repaso general antes del despliegue y entrega del proyecto, teniéndose que pulir pequeños errores de flujo, errores gramaticales o de tamaños en cabeceras y footers de cada página.

Por último, y con todo lo demás terminado, en este Sprint terminamos la documentación restante.



Otra de las características que permite Jira es observar el avance del proyecto y las tareas que se estén llevando a cabo por todos los miembros.

Respecto al control del trabajo personal, JIRA es muy útil por la sencillez con la que muestra el trabajo que tiene cada UNO pendiente. Esto se observa en el panel personal de la siguiente imagen, donde se puede ver las asignaciones pendientes, días restantes de ese Sprint y la actividad reciente de los miembros del proyecto.



## ANÁLISIS

### Requisitos generales

#### ¿Qué va a ser capaz de hacer Jamboree?

La aplicación deberá ser capaz de permitir a cualquier usuario registrado la creación de loops musicales en el estudio. Estos loops en esta primera versión serán de sonidos percusivos, en un futuro se irán implementando otros sonidos.

Además de la creación de loops, la aplicación ofrecerá un lugar donde guardar las creaciones del usuario; una audioteca. En ella el usuario deberá poder editar esos temas, borrarlos y subirlos o quitarlos del foro.

El aspecto social de la aplicación es importante, por ello Jamboree debe ofrecer la visualización de temas creados por los demás usuarios, y de poder subir los temas propios para su valoración de la comunidad. Para ello, dispondrá de un "foro" musical. En él se podrán filtrar los temas por categoría musical y se podrán editar los temas en el estudio de cada usuario.

La aplicación también dispondrá un apartado de ayuda. En estudio esta ayuda será visual, dando una plantilla gráfica de qué significa cada botón en esa página. En las demás páginas habrá acceso a una ayuda más genérica, que explicará lo que hace la aplicación y cada página en específico (ayuda.html)

Por último, se podrá visualizar los datos del usuario en una página de perfil y, además, desde esta se podrá cambiar la contraseña del mismo.

#### ¿Qué necesita la aplicación para funcionar?

De parte del usuario:

- Ordenador, móvil o tablet con conexión a Internet.
- Navegador de escritorio:

- Chrome 14
- Edge
- Firefox 23
- Opera 15/22
- Safari 6
- Navegador móvil (posibles latencias en móviles de gama baja/media):
  - Chrome 28
  - Edge
  - Firefox Mobile 25
  - Firefox OS 1.2
  - Safari 6

### ¿Qué va a usar la aplicación para ofrecer todo su potencial?

#### Parte servidora

La parte servidora está diseñada en PHP, por lo que se necesitará de un servidor con motor capaz de procesar peticiones PHP y devolver su resultado de forma dinámica en el navegador.

En caso de hacer el despliegue en local bastará con el motor de XAMPP. Por otro lado, en caso de hacer despliegue en un servidor cloud como Google Cloud Platform, se necesitará instalar php en la máquina servidora y configurarlo en el servidor de aplicaciones (nginx o apache)




#### Base de datos

La base de datos en local será la mySQL de XAMPP que, aunque limitada, nos bastará para poder mostrar la aplicación.

En Google Cloud Platform hemos hecho uso del entorno gratuito que ofrecen durante 90 días y, por ende, hemos creado una máquina dedicada al servidor, corriendo un mySQL 8.0 con las siguientes características:

#### Configuración

CPU virtuales	Memoria	Almacenamiento de SSD
2	7.5 GB	10 GB

-  La versión de la base de datos es MySQL 8.0
-  El aumento automático de almacenamiento está habilitado
-  Las copias de seguridad automatizadas están habilitadas

#### Parte front-end

Para la parte gráfica de la aplicación usaremos:

- jQuery: librería de JavaScript que facilita la programación en general y nos ahorra iteraciones y líneas de código.

- Tone JS: framework que se basa en la API Web Audio, desarrollada por Mozilla e implícita en las últimas versiones de motores en navegadores.
- LESS: preprocesador CSS que nos ahorra líneas de código y permite el uso de funciones y variables.
- Carousel.js : plugin de jQuery para la creación de carruseles de imágenes de forma rápida y responsive.

## Casos de uso

### Interacción en el Caso de Uso CU01

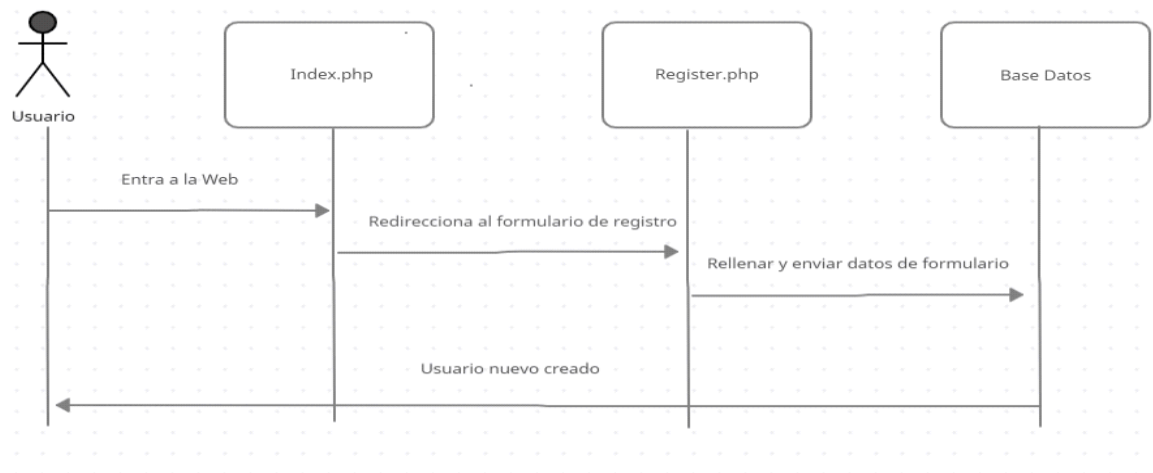


Figura 1: Diagrama de interacción para el Caso de Uso CU01

En el diagrama de la Figura 1 se muestra el flujo necesario para poder llevar a cabo el caso de uso "crear un usuario nuevo en la página web".

Para ello en primer lugar el usuario deberá entrar a la página web y elegir la opción de registrarse puesto que no tiene un usuario creado. Una vez la elija se le redireccionará a un formulario donde tendrá que poner sus datos y enviarlo.

Finalmente, estos datos se guardarán en la base de datos y tendrá acceso a todas las funciones de Jamboree.

### Interacción en el Caso de Uso CU02

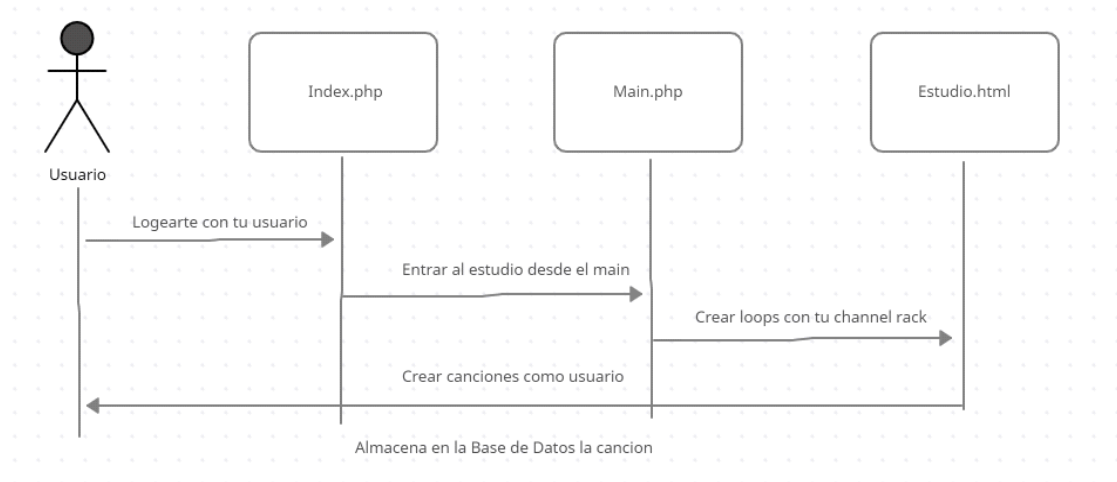


Figura 2: Diagrama de interacción para el Caso de Uso CU02

En el diagrama de la Figura 2 se muestra el flujo necesario para poder llevar a cabo el caso de uso: crear una canción nueva en la página web. Para ello en primer lugar el usuario deberá entrar a la página web e iniciar sesión para ir al apartado de main.php y elegir la opción del estudio. Una vez se encuentre este en el estudio podrá crear su loop y guardar la canción eligiendo un nombre para ella y así poder almacenarse en la BBDD de Jamboree.

### Interacción en el Caso de Uso CU03

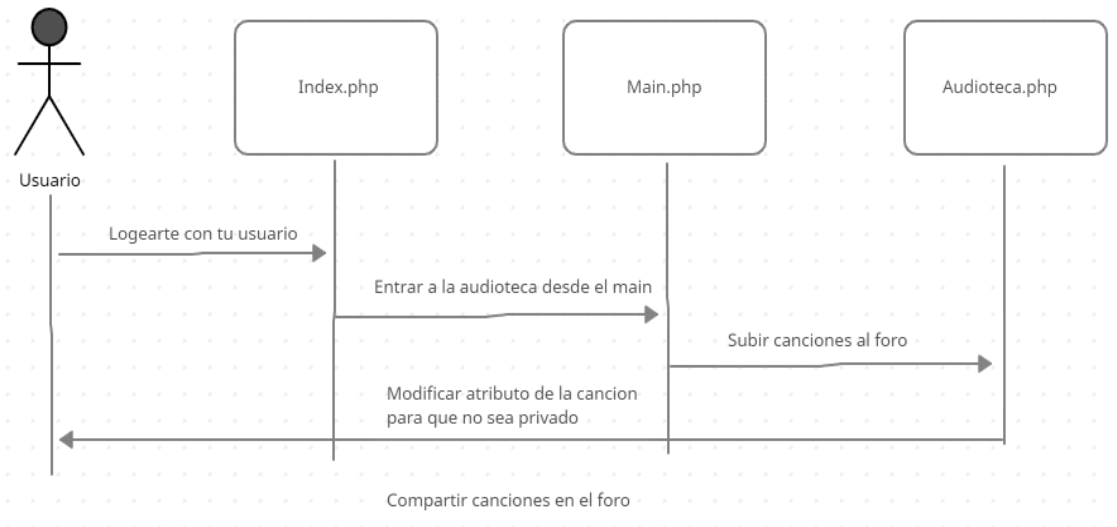


Figura 3: Diagrama de interacción para el Caso de Uso CU03

En el diagrama de la Figura 3 se muestra el flujo necesario para poder llevar a cabo el caso de uso: compartir una canción en el foro. Para ello en primer lugar el usuario deberá entrar a la página web e iniciar sesión para ir al apartado de main.php y elegir la opción de audioteca. Una vez se encuentre este en la audioteca podrá hacer que la canción deje de ser privada y compartirla con el resto de usuarios del foro en caso contrario esta solo le aparecerá a él.

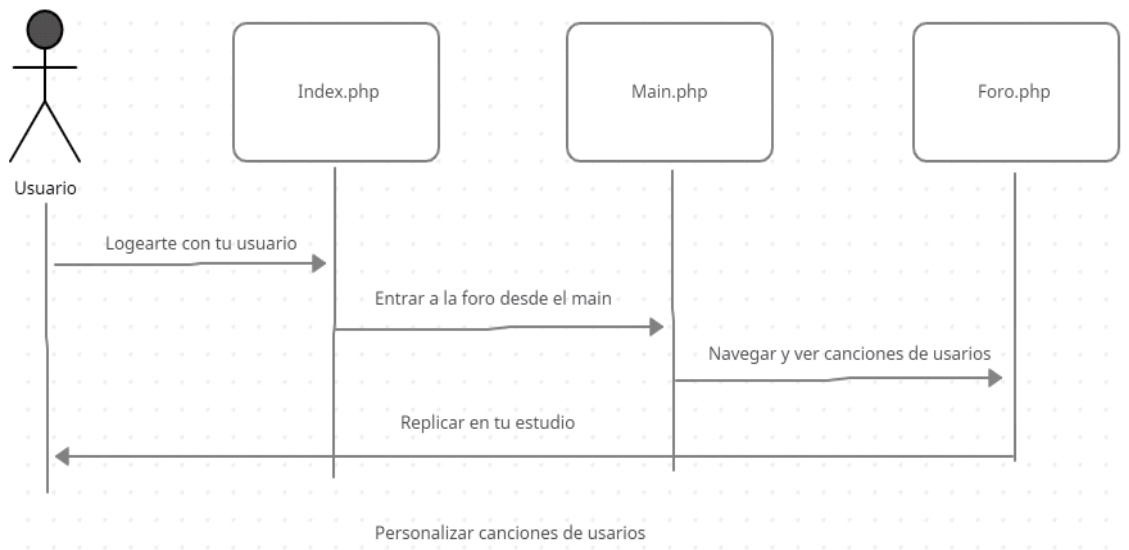
*Interacción en el Caso de Uso CUO4*

Figura 4: Diagrama de interacción para el Caso de Uso CUO4

En el diagrama de la Figura 4 se muestra el flujo necesario para poder llevar a cabo el caso de uso: personalizar canciones de usuarios desde foro. Para ello en primer lugar el usuario deberá entrar a la página web e iniciar sesión para ir al apartado de main.php y elegir la opción de foro. Una vez se encuentre este en el foro podrá navegar y escuchar diversos y loop y poder replicarlos en su estudio para hacer un nuevo sonido y almacenarlo.

## DISEÑO

### Arquitectura

La arquitectura que hemos utilizado en nuestra aplicación es cliente-servidor. Esta tecnología consiste en el procesamiento de la información desde una computadora central a la cual múltiples clientes, envían peticiones.

Para el desarrollo de nuestra aplicación hemos utilizado las siguientes herramientas:

- Visual Studio Code: Visual Studio, entorno de desarrollo en el cuál hemos realizado todo el código de la aplicación, soporta distintos lenguajes de programación y que permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno compatible con la plataforma.
- MySQL: en local usamos mySQL Workbench y en Cloud la consola de Google. En cada uno hemos podido crear nuestra base de datos con toda la información, es un sistema relacional de gestión de bases de datos que nos permite almacenar y acceder a los datos.

Como mostramos en el subtema de [requisitos](#) una de las tecnologías que es necesaria para la funcionalidad de JamBoree es la que ofrece Tone JS, dada su complejidad y el desconocimiento que teníamos de ella previo proyecto le dedicaremos un apartado dedicado a continuación.



## TONE JS

Antes de hablar de este framework pondremos en situación los pasos que hemos dado hasta llegar a él.

Sabíamos que los últimos motores de los navegadores más importantes estaban implementando la utilización de aspectos multimedia, por ello empezamos a documentarnos sobre cómo utilizar nuevas etiquetas como <audio> o cómo importar audios MIDI u OGG. Pero en seguida vimos que estas herramientas no eran suficientes para lo que queríamos, que era: creación de música en el propio navegador.

Siguiendo con la investigación nos topamos con la existencia de [Web Audio Api](#), una API que por fin permitía la creación y (extensa) modificación de audio online. El problema fue que, con nuestro tiempo limitado y lo extensa y difícil que era la documentación de esta API, no nos iba a dar tiempo a implementarla. Así que seguimos con la investigación y llegamos finalmente al framework Tone.js, el cuál parecía resolver nuestros problemas ya que se basa en la API de Web Audio, ofreciendo toda la potencia que ésta nos daba, pero aportando una capa de abstracción muy necesaria para ahorrarnos tiempo y dolores de cabeza.

### Aspectos esenciales de tone.js

Para empezar a trabajar con audio en tone.js deberemos importar el código que lo conforma en una etiqueta script:

```
<!-- link para Tone.js -->
<script src="https://tonejs.github.io/build/Tone.js"></script>
```

Ahora, para iniciar el trabajo con sonido debemos crear un [Tone.Transport](#), que hace las veces de "cronometrador" principal en el navegador. Se puede empezar, parar, pausar y repetir las veces que se quiera, pero siempre controlando que el usuario inicie la acción. Ya que las políticas de Google no permiten a una página empezar un audio sin la acción específica del usuario para ello.

Como en Jamboree existe la opción de dar en "play" varias veces, ya sea en temas distintos o en el mismo tema, lo que haremos será controlar que el transporte no este iniciado y, si lo está, simplemente reiniciarlo para evitar duplicidades y acoples en el sonido:

```
if (Tone.context.state !== 'running') {
  Tone.context.resume();
} else {
  Tone.Transport.start()
}
```

A éste Transport se le asociarán diferentes eventos y funciones que estarán sincronizadas con el tiempo del Transport.

### Audio en tone.js

Con tone.js podemos crear de o una amplia diversidad de sonidos en formato MIDI, modificando su frecuencia, tono, pudiendo añadir filtros, osciladores... Pero en Jamboree decidimos no crear sonidos MIDI por la razón de querer ofrecer un sonido más realista. Por ello, en vez de crear sonidos MIDI lo que hicimos fue precargar en buffer 4 sonidos reales en formato WAV y usar tone.js para controlar su sonido y secuenciarlos.

```
const baseSeq = new Tone.ToneAudioBuffers({
  urls: {
    kick: "/Kicks/Kick 01.wav",
    hat: "/Hats/Hat 01.wav",
    snare: "/Snares/Snare (1).wav",
    clap: "/Claps/Clap 01.wav",
  },
  baseUrl: "../../media",
})
```

Con [ToneAudioBuffers](#) podemos estructurar todas las fuentes de los sonidos a usar en un solo objeto. Ahora necesitaremos un reproductor para cada uno de esos sonidos, por lo que creamos un new Player por cada uno de los 4 sonidos que vamos a tener, uno por pista del estudio. A cada Player introduciremos el sonido mediante el atributo buffer. Como se indica a continuación:

```
//get del buffer los sonidos
let k = baseSeq.get("kick");
let h = baseSeq.get("hat");
let s = baseSeq.get("snare");
let c = baseSeq.get("clap");

//crear player por cada sonido
const players = [
  new Tone.Player(),
  new Tone.Player(),
  new Tone.Player(),
  new Tone.Player()
]
players.forEach(player => player.toDestination())
//cargamos en cada player su sonido
players[0].buffer = k
players[1].buffer = s
players[2].buffer = h
players[3].buffer = c
```

Como se ve, a cada Player también se le aplica el método toDestination(). Este lo que hace es conectar ese objeto de Tone a la destinación principal del equipo. Es decir, en un ordenador o en un móvil o los altavoces, o los auriculares si se estuvieran usando. Grosso modo es el equivalente a conectar un instrumento musical a un amplificador.

```
function play() {
  // loaded para asegurar que se carguen en bufer todos los sonidos
  Tone.loaded().then(() => {
    Tone.Transport.scheduleRepeat(function (time) {
      Tone.Draw.schedule(function () {
        }, "32n");
    }, "32n");
  });
}
```

Como hemos dicho, es necesaria la acción del usuario para activar los eventos relacionados al audio, por ello, cada vez que se pulse al play se activará la función play(). Que, a su vez, activará tres funciones relacionadas, como se ven en la imagen anterior, a saber:

1. Tone.loaded().then(...): esta promesa garantiza que se ejecutarán las funciones siguientes siempre y cuando se carguen completamente cada Player previo en buffer.
2. Tone.Transport.scheduleRepeat(...): esta función iterativa se repite cada tiempo que estimemos con el segundo parámetro, en nuestro caso, como el loop es de 32 beats, ese tiempo será 32n.
3. Tone.Draw.schedule(...): en cada iteracion de scheduleRepeat es recomendable usar la function draw.schecule para poder aociar eventos gráficos y de modificación en el

DOM para que se sincronicen con el tiempo del Transport, por ello como segundo parámetro toma la variable "time", recogida del cronómetro de Tone.Transport.

Estas funciones correlacionadas responden a una máxima vital en el uso del audio y el tiempo en JavaScript, y es: que los callbacks en JavaScript no son para nada síncronos, es por ello tan importante la precisión que aporta el parámetro time recogido en el callback para sincronizar todas las acciones.

Por último, dentro de la función Draw llamaremos cada vez que se lea un "beat" como marcado a la función start, pasando como parámetro 0.01 segundos, para evitar latencias a la hora de reproducir cada sonido.

```
player.start("+0.01")
```

Ya hemos visto como crear sonido con la librería Tone.js. Ahora, ¿cómo aplicamos estos conceptos a la aplicación Jamboree? Los siguientes puntos explican el paso a paso del trato de datos en JamBoree.

## Trato de datos en JamBoree estudio

### Conceptos base

Para poder leer los beats marcados en la interfaz y sincronizarlos con el audio lo que hemos usado es un objeto llamado "track". En él hay cuatro objetos conformados cada uno por un doble array de:

```
var tracks = {  
  0: [Array(32).fill(0), "kick"],  
  1: [Array(32).fill(0), "snare"],  
  2: [Array(32).fill(0), "hat"],  
  3: [Array(32).fill(0), "clap"]  
};
```

- Un array bidimensional de 0 y 1, donde 0 es un beat sin marcar y 1 es un beat marcado. Las dimensiones son de 32x4.
- El nombre del track en cuestión, ya sea "kick", "snare", "hihat" o "clap".

Inicialmente los arrays se rellenan a 0, y dependiendo si el usuario está creando una canción nueva o si viene de audioteca o foro, este array se irá modificando con las acciones en el DOM, a saber:

### Creación de track desde cero

Cada vez que se haga click en cualquier botón de beat se activa el evento con la función leerMarked(), que marca esa posición del array como 1:

```
function leerMarked() {
  for (let i = 0; i < 4; i++) {
    for (let j = 1; j < 33; j++) {
      `${tracks[i][1]} .beats .beat.${j}`.hasClass("marked") ?
      tracks[i][0][j - 1] = 1 :
      tracks[i][0][j - 1] = 0
    }
  }
}
```

### Carga de un track ya existente

Se comprueba que exista id en la URL, eso significaría que se hizo una llamada a estudio.html con el id pasado como parámetro GET, y entonces, deberíamos precargar un "track" ya existente con ese ID en la base de datos, en la imagen siguiente se ve la comprobación de la existencia de la variable GET:

```
//recoger id si existe en url, eso significaria que hay que preecargar ese track
const urlParams = new URLSearchParams(window.location.search);
if (urlParams.has('id_song')) {
  var id_song = urlParams.get("id_song")
  console.log(id_song);
  //get json track from db
  getTrack()
  getBPM()
} else console.log("No precargar");
```

En caso de existir "id\_song" se llamará a la función getTrack(), que lo que hace es recoger mediante fetch el track en JSON guardado en la base de datos con ese id recogido en la URL de estudio.html. Al recibir el track se parsea y se iguala el track a este objeto recibido, así tendríamos un track precargado con los 1 donde suenen las pistas y o donde no:

```
function getTrack() {
  var params = {
    method: 'POST',
    headers: {
      'Content-Type': 'application/x-www-form-urlencoded;charset=UTF-8'
    },
    body: "option=get&id_song=" + id_song
  }
  fetch("../server.php", params)
    .then(function (respuesta) {
      return respuesta.text()
    }).then(function (datos) {
      let obj = JSON.parse(datos)
      for (let i = 0; i < 4; i++) {
        tracks[i][0] = obj[i][0]
      }
      leerTrackJSON()
    })
}
```

Esta petición a server.php se recoge de la siguiente manera:

```
elseif ($_POST['option'] == "get") {
    $id_song = $_POST['id_song'];
    $consulta = "SELECT * FROM canciones where Id_Cancion='$id_song'";
    $resultado = mysqli_query($conexion, $consulta);

    while ($row = mysqli_fetch_array($resultado, MYSQLI_ASSOC)) {
        echo $row['track'];
    }
    $conexion->close();
}
```

Se hace una consulta a la base de datos con el id pasado como parámetro y se devuelve el track guardado, para parsearlo en el cliente.

### Reproducción del sonido

Con esa modificación leída desde el DOM o desde la base de datos, cada vez que demos en el botón de reproducir se llamará a la función play(), que iterará del modo que vimos previamente con scheduleRepeat, y que además leerá en cada iteración cada una de las 4 pistas.

Para la lectura de cada pista se crea una variable index y se va aumentando en cada iteración de scheduleRepeat hasta 32(número de beats), en cuyo caso volverá a inicializarse a 0. Esta variable index se repetirá por cada una de las 4 pistas en un bucle controlado de 4 iteraciones.

En caso de encontrar un 1 en la posición track[nombrePista][posicionBeat] reproducirá el Player correspondiente (el parámetro de "+0.01" es para evitar latencias, que el sonido no intente reproducirse antes de que el tiempo del Transport haya llegado a ese punto):

```
var index = 0

function play() {
    // loaded para asegurar que se carguen en bufer todos los sonidos
    Tone.loaded().then(() => {
        Tone.Transport.scheduleRepeat(function (time) {
            Tone.Draw.schedule(function () {
                for (let i = 0; i < 4; i++) {
                    let player = players[i]
                    if (tracks[i][0][index] == 1) {
                        //reproduzco ese sonido
                        player.start("+0.01")
                    }
                }
                if (index < 31) {
                    index++
                } else {
                    index = 0
                }
            }, time)
        }, "32n");
    })
}
```

### Guardado del track generado

En caso de que el usuario clique en el botón de guardar se pedirá en un formulario nombre y etiqueta de la canción, y se llamará a la base de datos con un fetch con los siguientes parámetros:

Se enviará la opción guardar, el id, el track a guardar ya hecho JSON, el nombre y etiquetas elegidos y los bpm recogidos del DOM.

En la parte servidora, esta acción se recoge del siguiente modo:

```
if ($_POST['option'] == "guardar") {
    $content = $_POST['track'];
    $id_User = $_POST['id_User'];
    $song_name = $_POST['song_name'];
    $tag = $_POST['tag'];
    $bpm = $_POST['bpm'];
    $consulta = "INSERT INTO canciones(id_Cancion,track,etiquetas,username,nombre,publica,escuchas,bpm)
VALUES (null,'$content','$tag','$id_User','$song_name','0',null,'$bpm')";
    mysqli_query($conexion, $consulta);
    $conexion->close();
}
```

Con los datos enviados como parámetros se hace una consulta de inserción en la base de datos para introducir esta nueva canción con todos los datos necesarios. En la base de datos se recogerá del siguiente modo:

[illegible]

## Trato de datos y flujo en audioteca y foro

La audioteca y foro difieren del estudio en sentido de que en estas páginas lo principal no es crear nuevos temas, si no únicamente poder cargar y reproducir el audio de los temas desplegados. Para ello, y por reducir código, lo que hemos utilizado es una característica de jQuery llamada `getScript()`:

```
$.getScript("../functions.js", function () {
    console.log("script cargado");
    loop()
});
```

Este método lo que hace es llamar a rutas de otros archivos de código para poder usar los métodos y variables de ese código en la misma página, es similar a un include de php.

Así, tanto en foro como en audioteca nada más cargar la página se llamará al método `loop()` de `functions.js`, que contiene esencialmente el `Tone.Transport.scheduleRepeat` que vimos [previamente](#).

Ya con el loop en funcionamiento, cada vez que se pulse en un botón de reproducir de foro o audioteca se llamará a las funciones `cargaTrack()` y `play()` de `functions.is`:

```
$(".play").click(function (ev) {
  track = $(this).prev().html()
  let bpm = $(this).prev().prev().html()
  let id = $(this).next().html()
  cargaTrack(track, bpm)
  play(id)
})
```

cargaTrack recibe como parámetros el track en JSON y los bpm recogidos del tema en cuestión:

```
function cargaTrack(t, bpm) {
  index = 0
  let obj = JSON.parse(t)
  for (let i = 0; i < 4; i++) {
    |   tracks[i][0] = obj[i][0]
  }

  $("#tempo").attr("value", bpm)
  Tone.Transport.bpm.value = bpm
}
```

Play recibe como parámetro el id del tema para poder encontrar ese track en la base de datos desde functions.js:

```
function play(id) {
  t_id = parseInt(id)
  if (Tone.context.state !== 'running') {
    |   Tone.context.resume();
  } else {
    |   Tone.Transport.start()
  }
}

function stop() {
  Tone.Transport.stop()
  console.log("pausado");
}
```

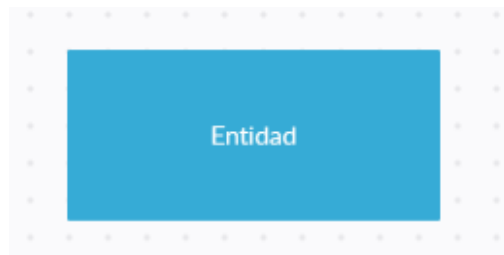
### Modelo de datos (E/R, tablas, etc.)

En este punto vamos a explicar qué es un diagrama entidad-relación, por qué elementos está formado y las tablas que hemos estimado necesarias para que nuestra página web funcione correctamente.

A través del diagrama entidad-relación podemos representar las entidades de una base de datos que presentan relevancia del sistema, junto con sus propiedades e interrelaciones. Sobre este modelo destacan los siguientes elementos:

**-Entidad:** Es una unidad de la base de datos que representa a un objeto, persona, empresa... etc., sobre el que se desea almacenar información y está compuesto de atributos, aquellos que nos permiten almacenar los datos que completan la definición del objeto.

Una entidad puede ser un objeto con existencia física como: una persona, un animal, una casa, etc. (Entidad concreta) o un objeto con existencia conceptual como: un nombre, una asignatura de clases, un puesto de trabajo, etc. (Entidad abstracta). Uno o un conjunto de los atributos que componen la entidad, nunca va a poder repetirse. A este atributo se le conoce como clave primaria o clave de la entidad. En cambio, existen atributos que contienen valores que coinciden con la clave primaria de otra tabla y que se utilizan para unir tablas. A este atributo se le conoce como clave foránea.



**-Relación:** Es el pilar fundamental en la construcción de bases de datos relacionales, ya que nos permite establecer las concordancias, asignaciones y relación entre las entidades o tablas, así mismo permite garantizarnos la integridad referencial de los datos, por tanto, la ventaja que tienen es que nos permiten evitar la duplicidad de registros en la base de datos.



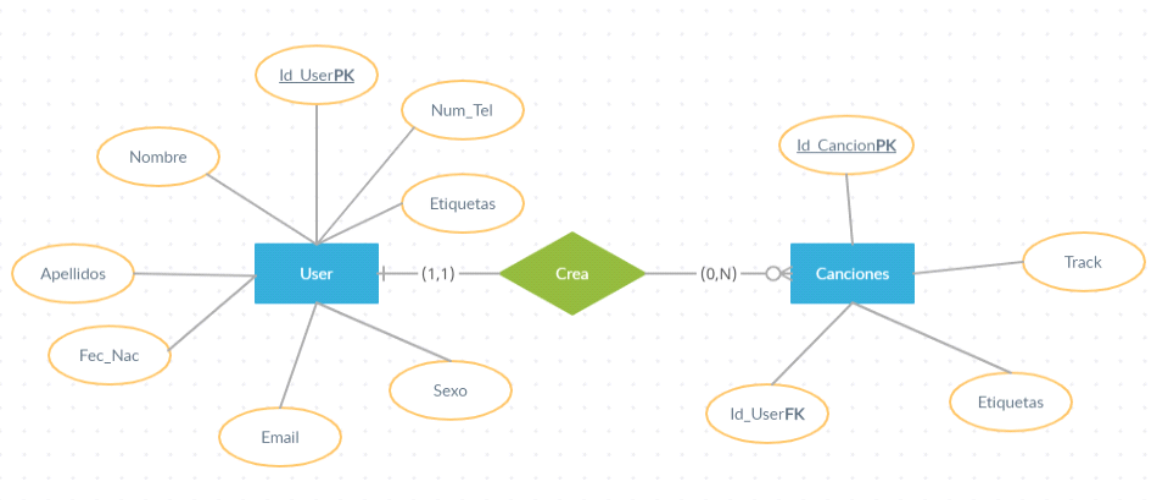
Según el número de entidades relacionadas (cardinalidad), se pueden definir principalmente tres tipos de relaciones (Relación Uno a Uno (1:1), Uno a Muchos (1: n), Muchos a Muchos(n:m)). A continuación, se puede observar un pequeño ejemplo de Relación Uno a Uno:

- Relaciones Uno a Uno (1:1). Una entidad A está asociada a lo más con una entidad B, y una entidad B a lo más con una entidad A. Ejemplo: "Ser jefe de" es una relación 1:1 entre las entidades empleado y departamento.



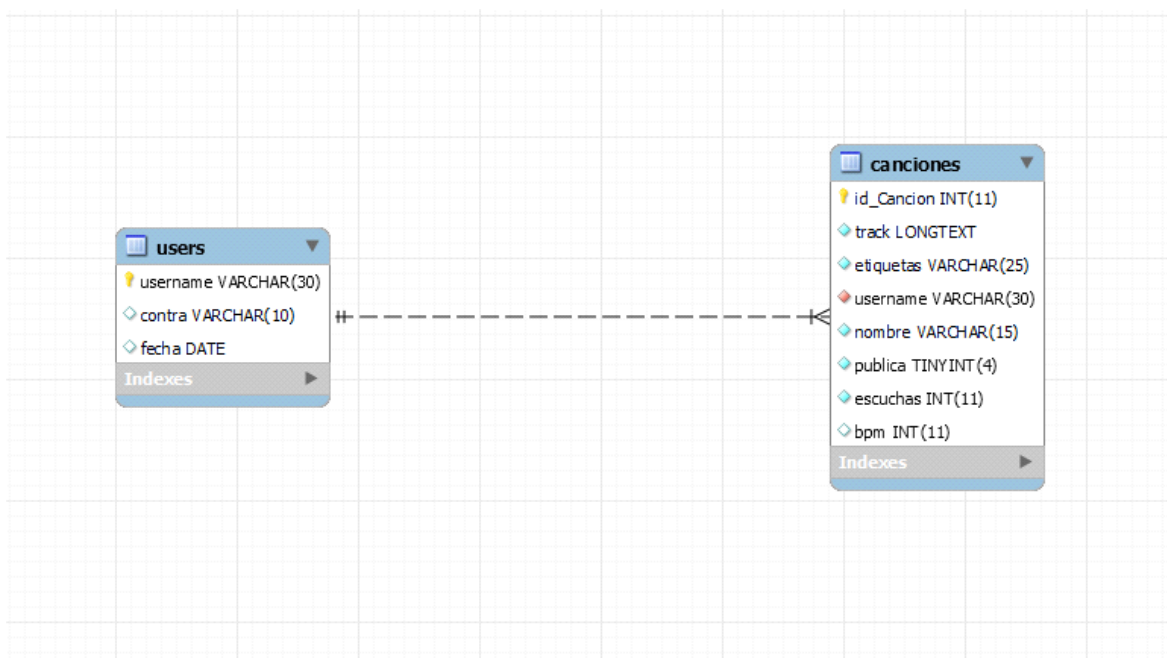


La página web de Jamboree se basa en el siguiente diagrama entidad-relación que presenta las cardinalidades (0, n) y (1,1):



En este diagrama podemos observar las dos entidades relevantes que componen nuestra base de datos. Se tratan de User y Canciones. Se puede diferenciar claramente como es User la entidad principal, ya que la entidad Canciones posee el atributo Id\_User que depende de la clave primaria de la entidad User. La entidad User posee como atributos su número identificador, nombre, apellidos, fecha de nacimiento, email, sexo, número de teléfono y etiquetas, siendo el primero clave de la entidad. Al igual que en la anterior entidad, la entidad Canciones también posee como atributo su número identificador, además de otros atributos como track, etiquetas y el atributo Id\_User que hace referencia a la clave primaria de la entidad User.

Una vez realizado el diagrama entidad-relación diseñamos el modelo lógico-relacional que es la conversión del modelo entidad/relación. Con este modelo se pueden ver las tablas con sus campos y los tipos de relaciones. Sería el diagrama final y el más detallado de la base de datos.

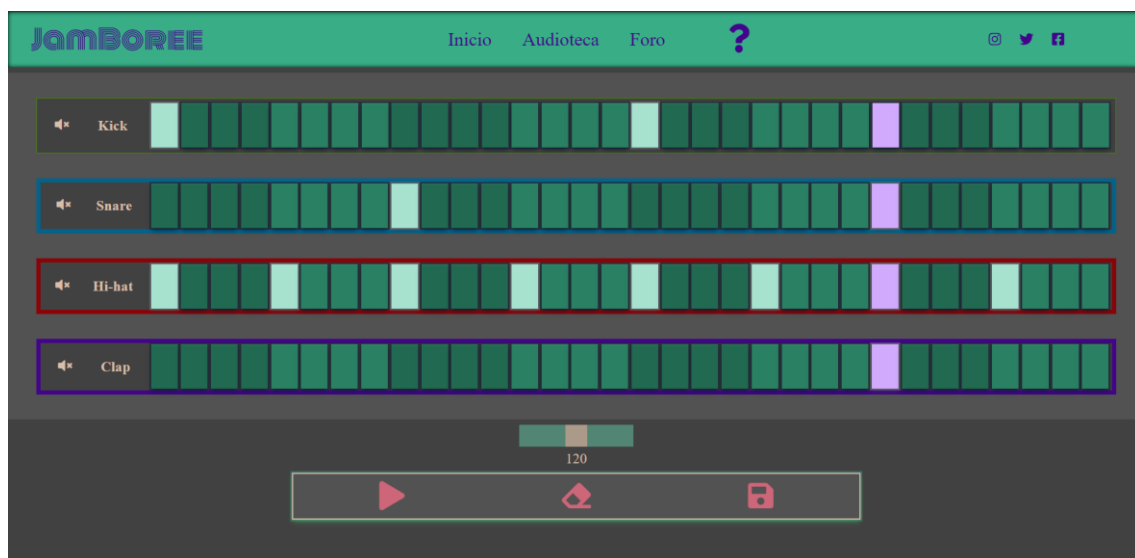


## Interfaz gráfica

### Representación gráfica del audio

En un primer momento la idea fue de usar otra librería de JavaScript para poder mostrar de manera gráfica el sonido. Pensamos en la librería Wave.js pero después de un estudio de sus capacidades vimos que no sería posible usar esa u otras librerías porque en nuestro caso, el audio no era un archivo mp3 o wav ya generado, si no que se basa en un sistema JSON propio de JamBoree. Por lo que decidimos hacer también manualmente esta representación gráfica usando tanto la facilidad de cambiar el DOM que ofrece jQuery como el pintado y borrado del elemento canvas.

### En estudio



Cada uno de los botones que representan un beat tiene asociado un evento que, al pulsarse intercambiará la clase "marked", que tendrá asociado unas características LESS para cambiar el color de ese botón:

```
/* evento a cada beat para pulsarlo */
$(".beat").click(function (ev) {
  ev.preventDefault()
  $(this).toggleClass("marked")
  leerMarked()
})
```

Al dar en el icono de "?" se mostrará un popup full page en el que se explica lo que significa cada elemento:



A la hora de ver por donde pasa el sonido existen diferentes visualizaciones:

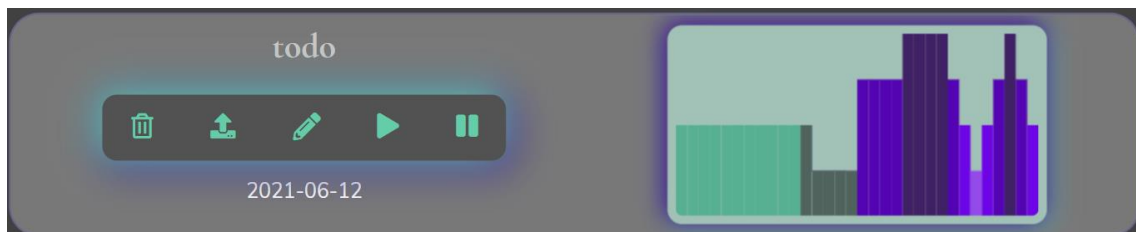
- Cada negra, o cada 8 beats, se animan los botones de control emulando una vibración de bombo, esto es con la clase "drummed".
- Para seguir el punto exacto de los 32 beats por donde está reproduciéndose el sonido se marca esa columna entera de un color diferente, esto es con la clase "timeline".
- Siempre que una de las 4 pistas suene, se remarca su borde de un color diferente, con la clase "drummed".

```

Tone.Draw.schedule(function () {
  for (let i = 0; i < 4; i++) {
    //mostrar animacion cada negra
    if (index == 1 || index == 9 || index == 17 || index == 25) {
      $('`.controles`).addClass("drummed")
    } else {
      $('`.controles`).removeClass("drummed")
    }
    //mostrar por donde va el sonido
    if (index == 32) {
      $('`.${tracks[i][1]} .beats .beat.${index}`').addClass("timeline")
    } else if (index == 0) {
      $('`.${tracks[i][1]} .beats .beat.32`).removeClass("timeline")
      $('`.${tracks[i][1]} .beats .beat.${index+1}`').toggleClass("timeline")
    } else {
      $('`.${tracks[i][1]} .beats .beat.${index+1}`').toggleClass("timeline")
      $('`.${tracks[i][1]} .beats .beat.${index}`').toggleClass("timeline")
    }
    let player = players[i]
    if (tracks[i][0][index] == 1) {
      //al sonar una track marco esa fila entera para visualizacion
      $('`.${tracks[i][1]}`).addClass("drummed")
      //reproduzco ese sonido
      player.start("+0.01")
    } else {
      $('`.${tracks[i][1]}`).removeClass("drummed")
    }
  }
}

```

*En audioteca/foro*



Tanto en foro como audioteca cada tema va acompañado de un canvas que se carga con la previsualización de los sonidos que hay en ese track. Ese dibujado se hace al cargar la página de la siguiente manera:

```

$("canvas").each(function () {
  let id = $(this).attr("id")
  let track = JSON.parse($(this).prev().html())
  console.log(track);
  for (let i = 0; i < 4; i++) {
    for (let j = 0; j <= 31; j++) {
      if (track[i][0][j] == 1) {
        draw(i, j, id)
      }
    }
  }
});
//@param i=0,1,2,3 ; j= 0-31
function draw(i, j, id) {
  //var colores = ["#6D5FF5", "#DE6262", "#F5CA5F", "#6DEB5B"]
  var colores = ["#412166", "#5504B3", "#6E05E6", "#954BEA"]
  var canvas = document.getElementById(`${id}`)
  var ctx = canvas.getContext("2d");

  /* quiero q se dibuje la 32ª parte que corresponda
  | si index es 4 que se dibuja la 4 línea de 32.
  beatwidth es el ancho del canvas / cada hueco de los beats */
  let beatWidth = canvas.width / 32
  let x = beatWidth * j
  let beatHeight = canvas.height / 4
  //let y = beatHeight * i+canvas.height
  let y = (beatHeight * i)

  /* quiero q se dibuje la track que corresponda
  | si i es 1 que se dibuje la barra de snare.
  beatwidth es el ancho del canvas / cada hueco de los beats */
  ctx.fillStyle = colores[i];
  ctx.fillRect(x, y, beatWidth, canvas.height);
}

```

Por cada canvas se recoge el track que contiene y se recorre su array en busca de 1 o 0, en caso de que una posición tenga un 1 se llama a la función draw.

La función draw recibe como parámetros el id del canvas en el DOM, el beat y la pista. Con ello pinta en el canvas correspondiente una barra por beat pulsado del modo siguiente:

- Se calcula el ancho de cada barra como una 32ª parte del ancho del canvas. En caso de que el beat sea el 5º, se posicionara el fillRect en  $32/\text{canvas} * \text{número de beat}$ .
- Se calcula el alto de cada barra como una 4ª parte del alto total del canvas. Así, se pintará cada una de esas 4 porciones con cada pista sonando, si es una se pintará  $\frac{1}{4}$ , si son todas se pintarán  $\frac{4}{4}$ . Para diferenciar cada pista también se cambia el color según la pista que se haya pasado como parámetro.

Cuando se pulse en el botón de reproducir se llamara a la función draw() pero del archivo functions.js. Este lo que hará será repintar por donde vaya el timeline de un color diferente al morado original, repintando con tonos verdes para saber por dónde va el track reproduciéndose.

## Colores

Para los colores de la aplicación nos basaremos en fondos oscuros, emulando los Digital Audio Workstation (DAW) más conocidos como FL Studio o Cubase, para facilitar al usuario poder estar largo tiempo sin cansar su vista y poder centrarse en la creación musical.

No obstante, definimos dos gamas de colores para representar la marca y dar un toque de color y contraste. Estas gamas son las abajo representadas, y se usarán para el logo, botones, recuadros...



## Tipografía

La palabra "JamBoree" en las páginas de estudio, foro, audioteca, perfil y ayuda se definirá con la tipografía "Monoton" y "serif" como *safe font*.

El texto de contenido ira en una tipografía sin serifa "Nunito" y "sans-serif" como *safe font*.

Las cabeceras de las diferentes cajas en la interfaz, como las de los temas en foro y audioteca o las pistas en estudio se dispondrán en tipografía Cormorant (serif para *safe Font*).

## Preprocesador LESS

Al dividir el trabajo entre 3 miembros nos encontramos con la necesidad de poder usar un procesador gráfico que pueda brindar el uso de variables y funciones que vayan a ser iguales en todas las páginas que componen la aplicación. Por ello, y además por el ahorro en líneas de código que aporta, usaremos el preprocesador LESS.

En él definiremos las variables de @texto, @título con las tipografías mencionadas previamente. También definiremos clases como .sombreado, .borde... Para poder ser usadas en las diversas páginas.

## Diferencias entre el diseño inicial y el diseño actual

Se puede observar gráficamente esa diferencia entre el archivo jamboree-mockup.pdf de la carpeta docs y el diseño final de la aplicación. Desgranando cada componente las diferencias son:

## Index

En un principio pensamos que la disposición de su estructura sería mejor en dos columnas (vídeo en una columna y el carrusel de artistas en la otra), pero al final lo estructuramos todo en una misma columna para que al entrar en la aplicación el usuario se fije en el vídeo en el cual se explica brevemente las funcionalidades de Jamboree).

## Main

La idea que teníamos sobre la estructura de Main era poner 3 botones (el de foro que ocupase la mitad del main y los botones de audioteca y estudio que ocupasen la otra mitad). Finalmente hemos puesto que cada botón tenga su respectiva imagen y que ocupen lo mismo cada uno de ellos, excepto cuando se hace hover sobre uno de ellos. En ese caso ese botón ocupa 2/3 del main.

## Foro

En foro las diferencias que existen con respecto al diseño inicial es que hemos implementado un filtro por etiquetas que busca aquellas canciones que tengan la misma etiqueta que tú has indicado.

Otra diferencia es que al dar al play en una canción podemos observar el transcurso de la melodía.

## Audioteca

Al igual que en foro, podemos observar el transcurso de la melodía. Además, hemos creado un botón que permite al usuario poder subir su canción al foro o al revés, poder quitarla del foro.

## Estudio

En estudio hemos implementado una barra deslizadora que nos permite aumentar o disminuir la velocidad en la que se reproduce la música. También hemos metido un icono que permite al usuario mutear o desmutear cualquier track. Por último, en la cabecera hay una interrogación que si la pulsas te sale un pequeño esquema el cuál, te da unas pequeñas instrucciones sobre las funcionalidades de estudio.

## Mi perfil

En mi perfil, además de poder cerrar sesión, le damos la opción al usuario de poder cambiar su contraseña en cualquier momento que desee.

## DESPLIEGUE

Para ambos tipos de despliegue (desarrollo en local y producción en cloud) tendremos el código fuente en nuestro repositorio de GitHub: <https://github.com/proyecto-jovellanos/jamboree>

Por lo que habrá que tomar en cuenta el entorno en el que se despliegue en cada caso, pero siempre tomando como referencia el código actualizado del repositorio.

### Local

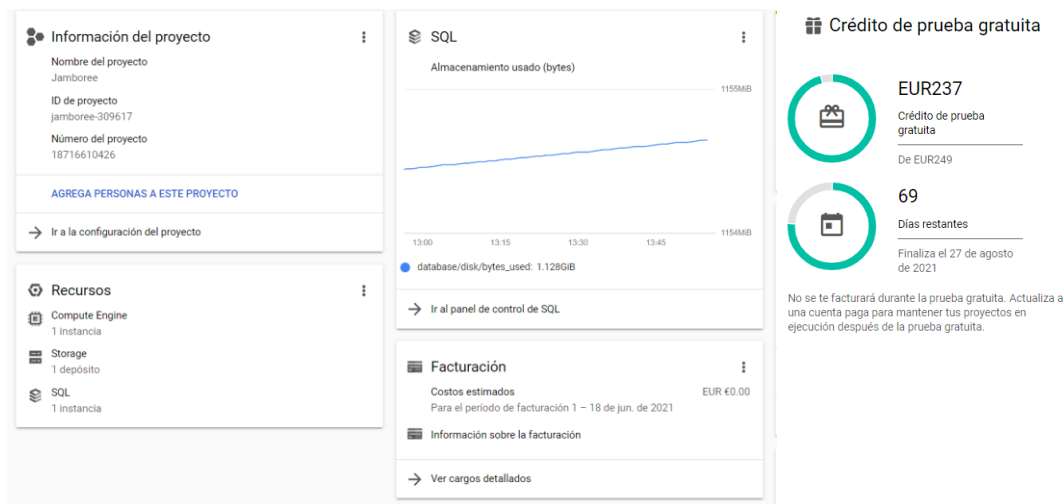
Para el despliegue de la aplicación orientado a su presentación y pruebas haremos el despliegue en local.

Esto es, simplemente usando XAMPP y mySQL Workbench para controlar la base de datos de una manera más completa que con phpMyAdmin.

Copiaremos la carpeta del repositorio de GitHub en C:\xampp\htdocs\ y ya podremos acceder a <http://localhost/jamboree> para ver nuestra página.

## Cloud

El despliegue en la nube será en la plataforma de Google Cloud, que ofrece 90 días y 249€ para probar la plataforma. En GCP (Google Cloud Platform) hemos creado un proyecto "jamboree", en el cual incluiremos las máquinas y recursos a usar, a saber: instancia ("compute engine") de Ubuntu Server, instancia de MySQL y un depósito de almacenamiento.



## Máquina virtual Ubuntu

Con IP 35.195.112.246. Sus características son las siguientes:

<input type="checkbox"/>	Estado	Nombre ↑	Zona	Recomendaciones	En uso por	IP interna	IP externa	Conectar
<input checked="" type="checkbox"/>	✓	jamboree	europa-west1-b			10.132.0.4 (nic0)	35.195.112.246	SSH

<b>Tipo de máquina</b>	<b>Hora de creación</b>	<b>Disco de arranque</b>	<b>Imagen</b>	<b>Tamaño (GB)</b>
e2-medium (2 CPU virtuales, 4 GB de memoria)	2 jun. 2021 10:44:08	jamboree	ubuntu-pro-2004-focal-v20210423	10

## Instancia de MySQL

Es una máquina dedicada exclusivamente a la base de datos. Con MySQL 8.0, 10GB de almacenamiento (dinámico) y 7,5 GB de memoria.

ID de instancia	Tipo	Dirección IP pública	Dirección IP privada	Nombre de la conexión con la instancia	Alta disponibilidad	Ubicación	Almacenamiento usado
jamboree	MySQL 8.0	35.241.128.167		jamboree-309617:europa-west1:jamboree	AGREGAR	europa-west1-d	1 GB de 10 GB

Para conectarnos a esta instancia de un modo seguro, autorizamos solamente la red de la máquina virtual que sirve la app. Así, incluimos su IP de la siguiente manera:

☒ IP pública  
 Autoriza una red o usa el [proxy de Cloud SQL](#) para conectarte a esta instancia. [Más información](#)

Redes autorizadas

jamboree (35.195.112.246) ✓

[AGREGAR RED](#)



## Configuración de la máquina Ubuntu Server

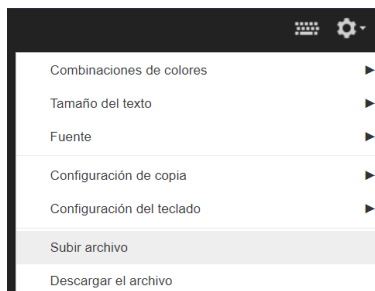
Conectarse a la máquina virtual es tan sencillo como apretar el botón "SSH" desde la consola de nuestro GCP:



Esta acción nos abrirá en una pestaña de navegador la consola de nuestra instancia Ubuntu. En ella instalaremos nginx, php y zip.

```
miguelmorci@jamboree:~$ sudo apt-get install nginx php zip
```

Una vez descargados todos los paquetes, subiremos a la máquina el archivo .zip del directorio que contiene todos los archivos del site jamboree, lo descomprimiremos en /home y copiaremos a /var/www/:



```
miguelmorci@jamboree:~$ unzip org-jamboree.zip
```

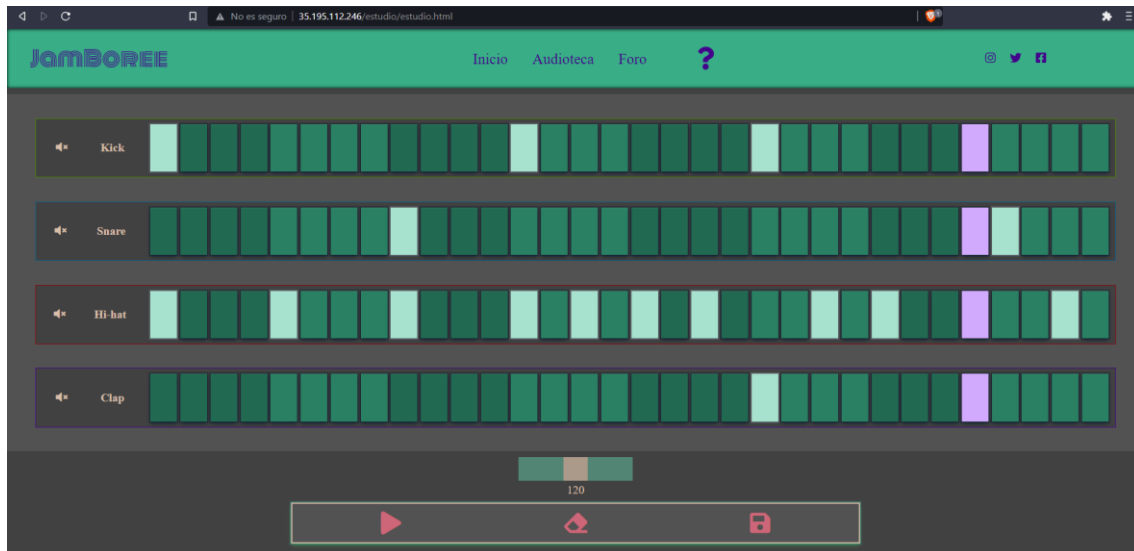
```
cp -r org-jamboree/ /var/www/
```

Acto seguido, creamos el archivo de configuración "jamboree.conf" en /etc/nginx/sites-enabled/ y creamos un symlink a /etc/nginx/sites-available

Ese fichero lo editaremos de modo que pueda servir páginas php y que las rutas de archivos estáticos funcionen:

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    # SSL configuration
    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;
    # include snippets/snakeoil.conf;
    root /var/www/org-jamboree/src;
    index index.php index.html index.htm index.nginx-debian.html;
    location /media {
        root /var/www/org-jamboree;
    }
    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
    }
}
```

Finalmente, reiniciamos nginx para que los cambios surtan efecto y ya podremos acceder a la dirección IP desde cualquier parte para acceder a la aplicación JamBoree.



### Características futuras

Jamboree representa una pequeña demo de lo que podría llegar a ser con el tiempo, puesto que se podrían implementar futuras funcionalidades que aportarán a Jamboree un carácter más fuerte como aplicación. Algunas de estas posibles futuras funcionalidades son:

#### Implementación de comentarios

Fue una de las principales ideas a implementar dentro del proyecto de Jamboree, pero por cuestiones de tiempo y de complejidad no se ha llevado a cabo.

La idea es que el usuario sea capaz de comentar en el foro los temas de otros artistas y poder establecer una conversación directa con los demás miembros de la comunidad.

Esta funcionalidad recalca también la otra cara de Jamboree que, aparte de ser una app musical, también conforma una red social para poder interactuar con otros usuarios.

#### Colaboración musical online

La colaboración online entre artistas consiste en la capacidad de dos o más artistas de poder compartir un proyecto en común y poder modificarlo de forma simultánea como si fuese un archivo de Drive o un juego online.

Esta funcionalidad no viene dada en casi ninguna app de producción musical y, a la vez, gran porcentaje de las canciones de la industria son colaboraciones. Por lo cual esta feature facilitaría mucho la producción a los artistas y abriría un gran mundo de posibilidades.

#### Capacidad de introducir canciones en álbumes

Por ahora el foro de Jamboree muestra solamente los temas de los artistas y no existe ninguna opción de poder agrupar las canciones en álbumes, por lo cual sería una gran característica a introducir en la app para organización del perfil del usuario.

Esto permitiría a los usuarios poder sacar álbumes o simplemente agrupar diversas canciones en un álbum y poder disfrutar de sus propias playlists.

### Perfil con avatar gráfico

En el apartado del perfil se podría implementar una funcionalidad bastante dinámica y original que sería la capacidad de representar un avatar que todo el mundo pudiera ver y reconocer.

Se le podrían añadir un sin fin de herramientas para poder personalizarlo como accesorios de moda, un logo musical personalizado, tatuajes, corte de pelo...

### Introducir nuevos sonidos y teclado

La parte del estudio permite a día de hoy sólo la utilización de 4 sonidos y no se puede cambiar el tono, pero en un futuro se podrá modificar el tono y poder tener instrumentos adicionales como un piano o una guitarra.

Para esto se introduciría un teclado que se asociará a cada instrumento y se le podrá cambiar el tono dependiendo de la tecla que se presione.

### Exportación de temas en formato MP3/WAV

Otra de las características que se implementarán será la capacidad de poder descargar en un formato mp3 o wav tus propios sonidos y canciones, de tal forma que se pudiese guardar en tu dispositivo y poder escucharla de forma offline.

### Exportación en formatos para software de Digital Audio Workstation

Por último, reconocemos la capacidad que tiene Jamboree como "bloc de notas" musical, en el que los usuarios podrán esbozar ideas rápidamente gracias a la nula instalación que requiere su uso y a que se podrá usar en el móvil. Es por ello que una de las características futuras será la de importar estos esbozos en formatos reconocidos por las grandes aplicaciones de escritorio que son usadas en el mundo de la producción musical, como Cubase o FL Studio, donde el artista podrá continuar y masterizar las ideas que empezó en la aplicación Jamboree.