TRABAJO FIN DE GRADO

INGENIERÍA EN INGENIERIA INFORMÁTICA

Sintetizador virtual

Subtitulo del Proyecto

Autor

Miguel García Tenorio

Directores

Carlos Ureña Almagro

Aqui se puede incluir nombre y logo del Departamento responsable del proyecto

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Junio de 2021

Alternativamente, el logo de la UGR puede sustituirse / complementarse con uno específico del

Sintetizador Virtual

Subtítulo del proyecto.

Autor
Miguel García Tenorio

Directores
Carlos Ureña Almagro

Título del Proyecto: Subtítulo del proyecto

Nombre Apellido1 Apellido2 (alumno)

Palabras clave: palabra clave1, palabra clave2, palabra clave3, ...

Resumen

Poner aquí el resumen.

Project Title: Project Subtitle

First name, Family name (student)

Keywords: Keyword1, Keyword2, Keyword3, ...

Abstract

Write here the abstract in English.

Yo, **Miguel García Tenorio**, alumno de la titulación TITU- LACIÓN de la **Escuela Técnica Superior de Ingenierías Informáti- ca y de Telecomunicación de la Universidad de Granada**, con DNI 75576490P, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Miguel García Tenorio

Granada a X de mes de 201 .

D. **Nombre Apellido1 Apellido2 (tutor1)**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

D. **Nombre Apellido1 Apellido2 (tutor2)**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Título del proyecto, Subtítulo del proyecto***, ha sido realizado bajo su supervisión por **Nombre Apellido1 Apellido2 (alumno)**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 201 .

Los directores:

Nombre Apellido1 Apellido2 (tutor1) **Nombre Apellido1 Apellido2 (tutor2)**

Agradecimientos

Poner aquí agradecimientos...

Índice

Contenido

Sintetizador virtual 1
Subtitulo del Proyecto 1
Autor 1
Directores 1
Sintetizador Virtual 5
Subtítulo del proyecto. 5
Autor 5
Directores 5
Título del Proyecto: Subtítulo del proyecto 7
Resumen 7
Project Title: Project Subtitle 9
Abstract 9
Informan: 13
Los directores: 13
Índice 17
Capítulo 1: Introducción 18
Capítulo 2: Objetivos 19
Capítulo 3: Planificación y presupuesto 19
3.1. Planificación Inicial 19
3.2. Presupuesto 21
3.2.1. Justificación del presupuesto 21
Capítulo 4: Análisis 22
4.1. Metodología de desarrollo 22
4.2. Análisis del entorno 22
4.2.1. Análisis competitivo 22
Capítulo 5: Diseño 26
Capítulo 5: Implementación y pruebas 39
5.1. Tecnologías seleccionadas 39
5.1.1. Análisis de tecnologías 39
5.1.2. Conclusión del análisis de tecnologías 41
Capítulo 6: Conclusiones y vías futuras 44
Capítulo 7: Bibliografía final 45

Capítulo 1: Introducción

1.2. Glosario de términos

Antes de entrar en la materia del trabajo, es necesario saber lo que significan los siguientes conceptos:

- **DAW:** Acrónimo en inglés para Digital Audio Workstation, o en español, estación de trabajo de audio digital. Un DAW es un sistema software dedicado a la grabación y edición de audio que utiliza una interfaz de audio digital para realizar la conversión digital-analógica y viceversa, para que un hardware pueda reproducir el audio digital editado.
- **Amplitud:** Es la fluctuación o desplazamiento de una onda desde su valor medio. Aplicado al sonido, es la cantidad de fuerza o energía de un sonido, conocida como volumen. Su medida son los decibelios (dB).
- **Decibelios (DB):** Sistema de medida logarítmico empleado para comparar el nivel de proporción entre dos valores de presión sonora, voltaje de señal o potencia. En el procesamiento de audio digital se trabaja con el intervalo [-∞ db, 0db] que representan la atenuación en el nivel de la salida del amplificador.
- **Envolvente acústica:** Describe la evolución temporal de la amplitud de cualquier sonido a partir de cuatro parámetros: Attack, Decay, Sustain, Release.
- **Attack:** Es el tiempo de entrada. Lo que tarda en un sonido en alcanzar su amplitud máxima después de haber sido ejecutado el instrumento. No tiene porque ser lineal y se mide en mili segundos (ms).
- **Decay:** Es el tiempo que tarda el sonido en caer a una etapa de sostenimiento, después de haber alcanzado la amplitud máxima sin soltar la tecla pulsada o punto de inducción vibratoria que se pulso para ejecutar el instrumento. No tiene porque ser lineal y se mide en mili segundos (ms).
- **Sustain:** Tiempo durante el cual la amplitud del sonido se mantiene constante hasta que se deja de emitir vibración o se suelta la tecla o el punto de inducción vibratoria. Es lineal y se mide en mili segundos (ms).
- **Release:** Tiempo que tarda un sonido en perder toda su amplitud (volumen) después de despear la tecla o punto de inducción vibratoria. Es lineal y se mide en mili segundos (ms).

Capítulo 2: Objetivos

1. El sistema debe ser capaz de generar y reproducir señales de audio digitales.
2. El sistema debe simular el comportamiento de un sintetizador analógico de manera digital.
3. El sistema debe permitir a los usuarios diseñar sus propios sonidos.
4. El sistema debe permitir almacenar sonidos diseñados y cargarlos para su reproducción o modificación.
5. El sistema debe ser compatible con dispositivos MIDI.
6. El sistema debe permitir grabar piezas musicales.
7. El sistema debe de eliminar el retardo para garantizar la experiencia de tocar un instrumento de manera virtual.
8. El sistema debe de ser amigable y accesible para el usuario

Capítulo 3: Planificación y presupuesto

3.1. Planificación Inicial

Inicialmente se realiza una planificación en la que se opta por dividir el desarrollo del proyecto en 6 fases o iteraciones, las cuales se dividen en 2 Sprint de dos semanas, excepto la primera iteración y la última que contienen un Sprint, en la que se planifican 8 bloques principales que conllevarán una serie de tareas a realizar.
Se realiza esta división, debido a que se seguirá una metodología ágil de desarrollo, Scrum, que será expuesta en secciones posteriores (ver …).
El proyecto comienza la semana del 15/02/2021 y termina la semana del 5/07/2021, por lo que se planifican 20 semanas en total. La planificación consta de 8 bloques principales. La figura 3.1 muestra la temporización de la realización de cada uno de los bloques de acuerdo a esta división

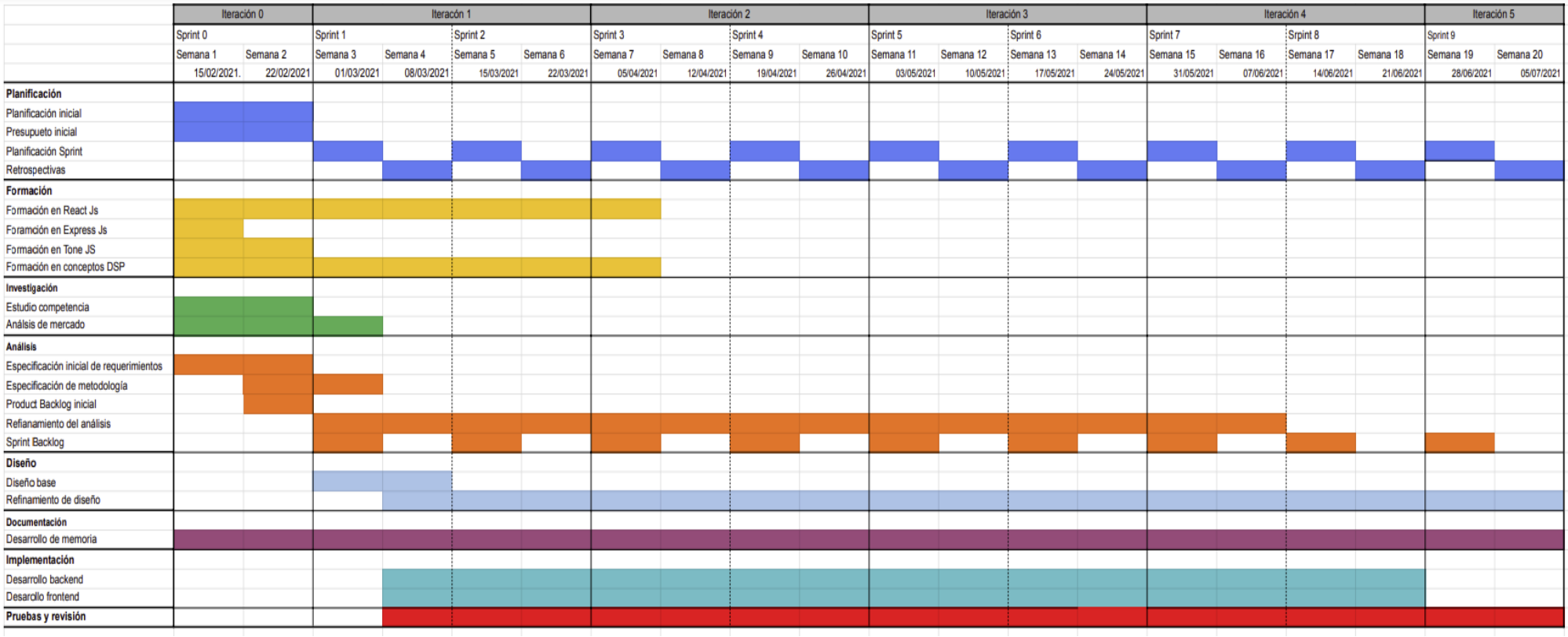


Figura 3.1. Diagrama Gantt planificación inicial

De acuerdo con esta planificación inicial se estiman las horas que será necesarias para la realización de cada uno de los bloques. De acuerdo con el Plan de Ordenación Docente del curso 2020/201 de la UGR 1 crédito ECTS se corresponde con 25 horas de trabajo, por lo que 12 créditos ECTS del Trabajo Final de Grado se corresponden con 300 horas de trabajo que hay que planificar. En la tabla 3.1 se muestra la correspondencia en horas para cada uno de los bloques planificados.

	HORAS	PORCENTAJE TOTAL
Planificación	22	7,3
Formación	21	7
Investigación	7	2,3
Análisis	40	13,3
Diseño	41	13,6
Documentación	32,5	10,83
Implementación	109	36,3
Revisión y Pruebas	27,5	9,17
TOTAL	300	100

Tabla 3.1. Horas planificadas

3.2. Presupuesto

Se estima que el precio del proyecto ronde los 5490€

Elementos	Coste	Total
Ordenador	780 €	780 €
Teclado MIDI	100 €	100 €
Auriculares estudio	60 €	60 €
Espacio de coworking (4 meses)	100 € / mes	400 €
Trabajo autónomo (4 meses)	13,5 € / hora	4050 €
Bibliografía	100 €	100 €
Total		5490 €

Tabla 3.2. Presupuesto del proyecto

3.2.1. Justificación del presupuesto

A continuación, se justifica cada elemento del presupuesto:

- **Ordenador:** Será necesario un ordenador de gama media/alta ya que se requiere de capacidad computacional para ejecutar el entorno de programación, lanzar el servidor para realizar pruebas y ejecutar el programa.
- **Teclado MIDI:** Se requiere un teclado MIDI para probar alguna de la

funcionalidad que ofrece el sistema. Basta con un teclado MIDI de gama media/baja para realizar las pruebas que rondan entorno a los 100 €

- **Auriculares de estudio:** Es preciso comprar unos auriculares de estudio de gama media para tener un sonido claro y limpio de los sonidos producidos por el software que se esta desarrollando. Este tipo de auriculares rondan los 60€ en el mercado.
- **Espacio de coworking:** Se supone que no se dispone de espacio de trabajo, por lo que se opta por alquilar una plaza en un espacio coworking en la que se dispondrá de electricidad, agua, calefacción, internet y sala de reuniones entre otros. Aproximadamente la tarifa ronda entre los 100 € mensuales.
- **Trabajo autónomo:** Como se planificaron 300 horas de trabajo (ver Tabla 3.1), y el salario de un Ingeniero informático Junior ronda los 13,5 € la hora y el proyecto tiene una duración de 4 meses, se estiman unos 4050 € de trabajo autónomo.
- **Bibliografía:** Serán necesarias fuentes que no se encuentran en Internet y no se encuentran en bibliotecas públicas (salvo en la de la UGR) que conllevarán un coste adicional.

Capítulo 4: Análisis

4.1. Metodología de desarrollo

4.1.1. Velocidad de Trabajo

Partimos de un equipo de desarrollo formado por 1 programador que van a dedicar un 100% de su trabajo al proyecto.
La duración de cada uno de los Sprints que vamos a realizar en el proyecto van a ser de 2 semanas.
La estimación del esfuerzo de cada una de las historias de usuario se ha expresado en días ideales de programación.
Se estima que un día ideal de programación se va a corresponder con 2 a 3 días reales de trabajo.
La duración de Sprint va a ser: 1 Sprint = 2 semanas = 12 Días reales
La Velocidad del equipo de desarrollo medido en punto de historia es: 1 Programador * 12= 12 días reales por iteración => de 10 a 12 PH por iteración.
Se ha decidido usar 12 Puntos de historia como la velocidad estimada del equipo.

4.2. Análisis del entorno

Antes de entrar en el análisis de los requerimientos del sistema o las historias de usuario, es necesario analizar el entorno para conocer cómo se comportan los usuarios al utilizar un sintetizador virtual y lograr un software más adaptado a los usuarios.

4.2.1. Análisis competitivo

En primer lugar, se realiza un análisis de la competencia para encontrar puntos fuertes y débiles de estos sistemas.
Antes de empezar con el análisis competitivo, hay que destacar que los softwares de síntesis digital más populares están diseñados para ser extensiones de programas

DAW, por lo que muchos de estos no se pueden utilizar sin un programa de este tipo, o bien se pueden utilizar, pero con una funcionalidad limitad. A pesar de esto, podemos encontrar otras opciones que no necesitan de un DAW para su funcionamiento.
De esta manera, podemos clasificar los sintetizadores en dos grandes bloques:

- Aquellos que necesitan de un DAW para su utilización.
- Aquellos que no necesitan un DAW para su utilización.

De acuerdo con esta clasificación, para cada grupo, se analiza el sintetizador más popular:

- **Serum:**
    - Esta dentro del primer grupo (no necesita un DAW).
    - Diseñado por la empresa Xfer Records.
    - Implementado en C++ usando JUCE [1] como framework.
    - <u>Puntos fuertes</u>
        - En cuanto a síntesis digital, es de los mas sofisticados en el mercado.
        - Tiene dos osciladores.
        - Interfaz bastante sencilla e intuitiva.
        - Posibilidad de elegir distintos tipos de onda.
        - Ofrece la posibilidad de modificar la onda de audio de cada oscilador a nuestro antojo.
        - Sistema de modulación bastante versátil, permite arrastrar el componente de modulación a lo que se quiera modular e incluso cuenta con una matriz de modulación.
        - Permite guardar configuraciones de sonidos.
        - Posee un banco de configuraciones preestablecidas, es decir, de sonido ya diseñados que se pueden buscar y filtrar por categorías u otros criterios.
        - Cuenta con gran variedad de efectos que se pueden aplicar a la vez a un mismo sonido.
        - En la parte inferior de la interfaz hay una simulación de un piano que nos va indicando que nota estamos pulsando encada momento.
        - Posee gráficos que se sincronizan con el sonido producido.
        - Permite insertar muestras de sonido (simples) para su síntesis.
    - <u>Puntos débiles</u>
        - Solo se puede utilizar a través de un DAW, es decir, es un plugin
        - Tiene bastantes parámetros que se pueden modificar por lo que se requiere un conocimiento alto en síntesis digital.
        - Debido a que es un plugin sus funciones se limitan a crear sonidos y enviarlos al DAW.
        - Elevado coste, actualmente $189 USD.

- **Midi.city:**
    - Esta dentro del segundo grupo (no necesita un DAW).
    - Es un sintetizador en la nube, que puede usarse en cualquier dispositivo que disponga de un navegador web.

[1] JUCE: Framework basado en C++ para el desarrollo de Plugins para DAWS

- Constituye una comunidad, en la que puedes crearte una cuenta para participar en foros, votar para nuevas características, acceder al registro de cambios, reportar algún fallo etc.…
- Principalmente se basa en JavaScript usando ToneJS como Framework.
- <u>Puntos fuertes:</u>
  - Se puede utilizar en cualquier dispositivo con navegador web.
  - Es open source, su utilización no requiere comprar ninguna licencia.
  - Posee todas las octavas.
  - Posee gráficos que se sincronizan con el sonido producido
  - Los sonidos generados se pueden reproducir, con el teclado, pulsando con el ratón en la nota o conectando un teclado MIDI.
  - Cuenta con un metrónomo .
  - Es capaz de secuenciar archivos MIDI.
  - Podemos reproducir una secuencia de percusión a la vez que reproducimos los sonidos sintetizados.
  - Cuanta con un banco de sonidos bien clasificados.
- <u>Puntos débiles:</u>
  - No se pueden crear nuevos sonidos, los sonidos se limitan a los que vienen por defecto.
  - No se pueden grabar piezas musicales tocadas con los sonidos sintetizados.
  - No se pueden modificar los parámetros de los sonidos, la síntesis es invisible al usuario se realiza en un segundo plano.
  - En ocasiones presenta latencia.

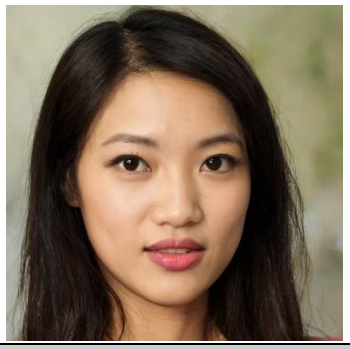
4.2.2. Personas y escenarios

Para identificar las metas y los puntos débiles de nuestro usuario objetivo, se crean dos personas que interaccionaran con Serum y Midi.city.

PERSONA 01		
Nombre	Javier Pérez	Foto
Edad	23	
Sexo	M	
Educación	Graduado en comunicación audiovisual. Master en producción musical	
Contexto de uso		
Cuándo	En horario laboral	
Dónde	En el estudio	

Dispositivo	Ordenador de sobremesa
Misión	
Objetivo	Quiere diseñar nuevos sonidos para sus producciones
Expectativas	Obtener sonidos de calidad y de manera sencilla
Motivación	
Urgencia	Lo necesita cuanto antes
Deseo	Quiere diseñar sonidos nuevos por el mismo
Actitud hacia la tecnología	
Esta siempre a la última, tiene bastante soltura con el uso de las tecnologías	

Tabla 4.X. Persona 01

PERSONA 02		
Nombre	Ava Miller	Foto 
Edad	27	
Sexo	10	
Educación	Graduado en Bellas Artes. Master en artes escénicas	
Contexto de uso		
Cuándo	En su tiempo libre	
Dónde	En casa	
Dispositivo	Portátil/ Móvil	
Misión		
Objetivo	Quiere poder tocar el piano con su nuevo teclado MIDI	

Expectativas	No gastar dinero en software adicional y poder grabar sus piezas
Motivación	
Urgencia	Está aprendiendo de manera autodidacta a tocar el piano y necesita empezar a tocar cuanto antes
Deseo	Poner en práctica las habilidades aprendidas al piano y experimentar con variedad de sonidos
Actitud hacia la tecnología	
Se desenvuelve, pero tampoco es una experta	

Tabla 4.X. Persona 02

A partir de estas dos personas se definen los siguientes escenarios

ESCENARIO 01		
Nombre persona	Javier Pérez	Foto
Objetivo persona	Diseñar nuevos sonidos para sus producciones	SERUM
Escenario		
Javier es un joven productor de Granada. Estudió el grado de Comunicación Audiovisual en la UGR y posteriormente hizo un master privado en producción musical, ya que desde pequeño ha sido un amante de la música y su sueño era llegar a ser alguien importante en el panorama musical. Paralelamente a sus estudios, Javier trabajaba como camarero de un pub para ganarse un sueldo y así ahorrar para su proyecto musical. Gracias a estos ahorros, recientemente ha montado un pequeño estudio en un local para comenzar su proyecto. Dentro de esta inversión incluyo Ableton Live y Serum para lograr más calidad en sus producciones ya que hasta ahora solo contaba con una guitarra y un teclado. Dispuesto a crear su primera pieza musical en su nueva etapa de productor profesional, abre el DAW Ableton Live y el sintetizador Serum. Su objetivo es lograr un sonido parecido al de una campana.		

Para ello enciende los dos osciladores, selecciona el tipo de onda que cree más conveniente para cada oscilador, establece los parámetros básicos de la envolvente y selecciona un tipo de filtro. Encuentra muchos parámetros que no sabe como funcionan por lo que no los toca, ya que él es productor musical y no es un experto en ingeniería de audio, pero tiene conocimientos medios. Para terminar añade reverb, delay y un compresor para hacer más interesante el sonido. Una vez terminado guarda el sonido como “Campana propia”. El resultado ha sido óptimo, pero no está convencido del todo, así que selecciona una de las campanas que vienen ya creadas por el sintetizador y la modifica un poco hasta conseguir el sonido que buscaba.

Tabla 4.X. Escenario 01

ESCENARIO 02		
Nombre persona	Ava Miller	Foto
Objetivo persona	Tocar el piano, probar sonidos diferentes y grabar piezas musicales	
Escenario		
Ava es chica de 27 nacida en Londres, que trabaja para una compañía española de macro-festivales que incluye espectáculos escénicos en sus festivales realizados por todo el mundo. Debido a la situación de pandemia, la empresa no puede realizar su actividad por lo que Ava se encuentra en un ERTE. Ava es una persona inquieta y curiosa y debido a esta situación ha necesitado buscarse un entretenimiento, el piano. Ava ha comenzado a aprender a tocar el piano de manera autodidacta a través de un teclado MIDI que tenía su hermano. Este tipo de teclados requieren de un software para que se escuche “algo”. Existen muchas alternativas de pago, pero Ava no quiere gastarse dinero ya que la situación económica no es favorable y además necesita algo que pueda usarlo en varios ordenadores ya que no siempre usa el mismo. Investigando por la red, encontró Midi.city. Para poner a prueba lo que esta aprendiendo, conectó su teclado al ordenador y comenzó a tocar notas. Se dio cuenta que había una biblioteca de sonidos, así que eligió el piano. Para su sorpresa, a la vez que tocaba podía establecer una percusión que le iba acompañando mientras tocaba, además de poder establecer la velocidad a la que iba, había un metrónomo. Por desgracia, su experiencia no fue satisfactoria al completo, ya que no encontró ningún botón donde poder grabar lo que iba tocando.		

Tabla 4.X. Escenario 02

4.2.3. Aspectos a tratar

A partir del análisis competitivo y de la creación y propuesta de personas y escenarios, se establecen los siguientes aspectos que deberían tratarse a la hora de definir la funcionalidad de nuestro sistema ya que vana a aportar un valor añadido al producto final:

Interesante/relevante	Criticas
• La portabilidad del software es una ventaja competitiva • Existencia de gráficos que acompañen al audio • Uno o más osciladores • Uno o más efectos • Existencia de filtros • Posibilidad de usar un teclado MIDI para reproducir los sonidos generados • Biblioteca de sonidos ya diseñados para poder usar alguno y/o modificarlos • Opción para guardar aquellos sonidos creados o modificados • Existencia de un metrónomo • Posibilidad de importar sonidos para realizar una síntesis a partir de estos • Grabación de las piezas musicales tocadas • Están disponibles todas las octavas	• Demasiados parámetros hacen compleja la interfaz y el diseño de sonido • Las interfaces no son accesibles y no se adaptan al usuario • El coste de estos programas es muy elevado, no existen muchas opciones open source • Muchos requieren de otros programas de pago para funcionar, hay usuarios que no quieren esto. • Problemas frecuentes relacionados con la latencia

Tabla 4.X. Aspectos a tratar

4.3. Requisitos del sistema

A continuación, se especifican los requisitos que debe cumplir el sistema:

4.3.1. Requisitos Funcionales

RF-1. Gestión de las “Fuentes”. El sistema será capaz de generar señales de audio mediante la síntesis digital mediante el uso de osciladores.

RF-1.1. Uso de uno o dos osciladores. El usuario podrá encender y apagar cualquiera de los dos osciladores, permitiendo que los dos estén encendidos simultáneamente.

RF- 1.1.1. Elección del tipo de onda. Para cada oscilador se podrá elegir entre una onda con forma de seno, triangulo, cuadrado, o diente de sierra.

RF- 1.1.2. Los osciladores generarán todas las frecuencias correspondientes a todas las octavas de un piano.

RF-1.2. Polifonía. El sistema será capaz de generar polifonía, es decir, será capaz de generar la señal de audio correspondiente a un acorde o a una nota por separado.

RF-1.3. Suma de señales. El audio generado por las fuentes será la suma de las señales de audio generadas por las fuentes.

RF-2. Gestión de los “Modificadores”. El sistema podrá alterar la señal de audio que las fuentes generan.

RF-2.1. Aplicación de la envolvente. Para cada uno de los osciladores existirá una envolvente de la amplitud de audio que controlará la evolución temporal de la seña generada.

RF-2.2. Uso de reverberación (Reverb). El usuario podrá aplicar un efecto de Reverberación sobre la señal de audio generada por las fuentes.

RF-2.2.1. Encendido y apagado. Se podrá encender y apagar la reverberación en cada momento.

RF-2.3. Uso de Delay. El usuario podrá aplicar un efecto de Delay sobre la señal de audio generada por las fuentes.

RF-2.3.1. Encendido y apagado. Se podrá encender y apagar el Delay en cada momento.

RF-2.4. Uso de compresor. El usuario podrá aplicar un efecto de Compresor que controle la dinámica sobre la señal de audio generada por las fuentes.

RF-2.4.1. Encendido y apagado. Se podrá encender y apagar el compresor en cada momento.

RF-2.4. Uso de compresor. El usuario podrá aplicar un efecto de Compresor que controle la dinámica sobre la señal de audio generada por las fuentes.

RF-2.4.1. Encendido y apagado. Se podrá encender y apagar el compresor en cada momento.

RF-2.5. Uso de filtros. El usuario podrá aplicar filtros de frecuencia sobre la señal de audio generada por las fuentes.

RF-2.5.1. El usurario podrá aplicar un filtro pasa bajos (Low Pass), es decir un filtro que filtra las frecuencias aguadas

RF-2.5.2. El usurario podrá aplicar un filtro pasa altos (High Pass), es decir un filtro que filtra las frecuencias aguadas

RF-2.4. Uso de ecualizador de 3 bandas. El usuario podrá aplicar ecualización de 3 bandas sobre la señal de audio generada por las fuentes.

RF-2.4.1. Encendido y apagado. Se podrá encender y apagar el ecualizador en cada momento.

RF-3. Gestión de los “Controladores”. El sistema permitirá al usuario controlar los modificadores y las fuentes

RF-3.1. Control de la envolvente. Para cada envolvente se podrán modificar los valores de Attack, Decay, Sustain, Release.

RF-3.1.1. Modificar el tiempo de ataque (Attack). El sistema permitirá modificar tiempo de ataque de una envolvente, el tiempo que tarda la señal en alcanzar su máximo valor de amplitud. Este valor se podrá modificar entre los 0 y los 2 ms.

RF-3.1.2. Modificar el tiempo de caída (Decay). El sistema permitirá modificar tiempo de caída de una envolvente, el tiempo que tarda la señal en caer a un valor sostenido después de superar la fase de ataque. Este valor se podrá modificar entre los 0 y los 2 ms.
RF-3.1.3. Modificar el tiempo de sostenimiento (Sustain). El sistema permitirá modificar tiempo de sostenimiento de una envolvente, el tiempo que la señal se mantiene constante después de superar la fase de caída. Este valor se podrá modificar entre los 0 y los 1 ms.
RF-3.1.4. Modificar el tiempo de relajación (Release). El sistema permitirá modificar tiempo de relajación de una envolvente, el tiempo que la señal tarda en perder toda su amplitud. Este valor se podrá modificar entre los 0 y los 5 ms.

RF-3.2. Control del efecto de Reverberación. Para el efecto de reverberación se podrá modificar el Decay, Predelay, HiCut, LowCut.

RF-3.2.1. Modificar Predelay. El sistema permitirá modificar el valor de predelay de la reverberación, es decir el tiempo a partir del cual el efecto de reverberación empieza actuar. Este valor se podrá modificar en proporción a la duración de la señal de audio que producen los osciladores, por lo que esta comprendido entre 0% y 100%.
RF-3.2.2. Modificar Decay. El sistema permitirá modificar el valor de Decay de la reverberación, es decir el tiempo de duración del efecto.
RF-3.2.3. Modificar HiCut. El sistema permitirá modificar el valor de la frecuencia donde se comienza a filtrar la reverberación. Este valor estará dentro del intervalo de 20k-0hz.
RF-3.2.4. Modificar LowCut. El sistema permitirá modificar el valor de la frecuencia donde se comienza a filtrar la reverberación. Este valor estará dentro del intervalo de 0-20khz.

RF-3.3. Control del efecto de Delay. Para el efecto de Delay se podrá modificar el FeedBack.

RF-3.3.1. Modificar FeedBack. El sistema permitirá modificar la cantidad

de FeedBack de efecto Delay, es decir, durante cuánto tiempo tendrá lugar el Delay. Este valor está comprendido entre el 0% y el 100%

RF-3.4. Control del Compresor. Para el compresor se podrá modificar el Ratio, Threshold, Attack, Release.

RF-3.4.1. Modificar Ratio. El sistema permitirá modificar el valor de ratio, es decir, como se va a comprimir la onda. Este valor está comprendido entre 1 y 20 Uds.
RF-3.4.2. Modificar Threshold. El sistema permitirá modificar el valor de Threshold, es decir, a que altura en el eje y de la amplitud de la onda se va a colocar el ratio. Este valor está comprendido entre -100 y 0 dB.
RF-3.4.2. Modificar Ataque (Attack). El sistema permitirá modificar el valor de ataque, es decir, cuando empieza a actuar el compresor. Este valor está comprendido entre 0 y 1 ms.
RF-3.4.3. Modificar Release. El sistema permitirá modificar el valor de reléase para el compresor, es decir, durante cuánto tiempo actúa el compresor antes de volver a actuar (antes de volver a la fase de ataque). Este valor está comprendido entre 0 y 1 ms.

RF-3.5. El sistema permitirá modificar la cantidad de efecto que se aplica para cada uno de ellos, es decir, valor de dry/wet. Este valor es proporcional al efecto y está comprendido entre el 0% y el 100%.

RF-3.6. El sistema permitirá alterar el valor de volumen maestro del programa. Este valor oscilará entre los -100 y 0 dB.

RF-3.7. Modificar las bandas del ecualizador. Cada banda aceptará valores entre -100 y 0 dB.

RF-3.8. Control oscilador. Para cada oscilador se podrá modificar el Paneo y el volumen.

RF-3.8.1. Modificar el paneo, se podrá establecer cuanto sonido sale por el canal L y cuanto sale por el canal R. Este valor va del -100 a 100 Uds.
RF-3.8.2. Modificar volumen, se podrá se podrá ajustar la cantidad de sonido que se quiera (el volumen del sonido que está produciendo). Este valor está comprendido en el intervalo de -100db-0dB.

RF-4. Gestión de gráficos. El programa pintará de manera gráfica, siguiendo un determinado algoritmo, el espectro frecuencial de la onda que producen los osciladores

RF-5. Gestión de almacenamiento. El sistema gestionará todo lo relacionado con lo referente a los sonidos y los usuarios de este.

RF-5.1. Guardar un sonido creado o modificado. Se guardarán en base de datos todos los valores que producen la señal de audio final. Además, se podrá asignar un nombre y una categoría, valoración

RF-5.2. Cargar un sonido guardado en base de datos

RF-5.3. Buscar un sonido previamente guardado.

RF-5.3.1. Aplicar filtro por categoría o valoración en la búsqueda.

RF-6. Gestión “MIDI”. El sistema será capaz de procesar información MIDI o de control.

RF-6.1. El sistema será capaz de comunicarse con un controlador MIDI de manera que se pueda utilizar este con las señales de audio producidas.

RF-6.2. Cuando se pulsa una tecla del teclado, si hay algún oscilador encendido, se produce un sonido correspondiente a una nota, simulando un teclado MIDI. Será posible tocar hasta dos octavas desde el teclado.

RF-6.3. El sistema podrá leer y procesar archivos MIDI

RF-7. El sistema permitirá grabar los sonidos producidos y descargarlos en mp3.
RF-8. El sistema permitirá a un usuario registrarse.
RF-9. El sistema permitirá a un usuario hacer login y logout.
RF-10. Se podrá encender y apagar el metrónomo y ajustar la velocidad de este

4.3.2. Requisitos no funcionales

Interfaz

RN-1. Cuando se modifique cualquier valor se debe de mostrar en la interfaz simultáneamente el valor que se es está modificando

RN-2. La interfaz debe ser sencilla, accesible e intuitiva

RN-3. Cuando se modifica un sonido aparecerá un asterisco al lado de su nombre para indicar que se ha modificado y se puede guardar

RN-4. La interfaz debe mostrar un piano

Rendimiento

RN-5. El sistema deberá ocuparse de la sincronización entre gráficos y audio, la latencia no puede ser superior a 1 segundo

RN-6. La latencia entre que se pulsa una tecla del teclado del ordenador, del controlador MIDI o se clica en una tecla de la interfaz, hasta que se reproduce el sonido correspondiente no puede exceder los 700ms.

Disponibilidad

RN-7. El sistema debe poder utilizarse en cualquier dispositivo que disponga de un navegador web

RN-8. El sistema debe estar siempre en ejecución.

Implementación

RN-9. Se deben de usar lenguajes de programación que puedan ser interpretados por navegadores web

Seguridad

RN-10. Solo podrán acceder al sistema los usuarios dados de alta
RN-11. Cada usuario tendrá un espacio limitado para almacenar sus sonidos

Usabilidad

RN-12. Siempre se cargará un sonido por defecto cuando se entra en el sistema

Físicos

RN-13. El sistema requiere de un servidor para almacenar la base de datos y ejecutar tanto el backend como el forntend.

4.4. Product Backlog

A partir de la fase de análisis en la que se especificaron los requerimientos del sistema se realiza un listado priorizado de las historias de usuario que se llevarán a acabo durante le desarrollo del sistema.

La prioridad está expresada entre 1 y 5, siendo 1 las historias más prioritarias y 5 las menos prioritarias. A continuación, se muestra el Product Backlog ordenado por prioridad

ID	Nombre de la historia	Prioridad
1	Como usuario registrado quiero utilizar uno o dos osciladores	1
2	Como usuario registrado quiero elegir el tipo de onda de un oscilador	1
3	Como usuario registrado quiero ajustar el volumen de los osciladores por separado	1
4	Como usuario registrado quiero modificar la envolvente de un oscilador 1	1
5	Como usuario registrado quiero tocar notas clicando en las teclas que aparecen en la pantalla	1
6	Como usuario registrado quiero tocar notas desde el teclado del ordenador	1
7	Como usuario registrado quiero tocar un acorde	1
8	Como usuario registrado quiero aplicar un efecto Reverb	2
9	Como usuario registrado quiero modificar el efecto Reverb	2
10	Como usuario registrado quiero aplicar un efecto Delay	2
11	Como usuario registrado quiero modificar el efecto Delay	2
12	Como usuario registrado quiero aplicar un efecto Reverb	2
13	Como usuario registrado quiero controlar el volumen del programa	2
14	Como usuario registrado quiero aplicar un filtro	3
15	Como usuario registrado quiero modificar el filtro	3
16	Como usuario registrado quiero aplicar un compresor	3
17	Como usuario registrado quiero modificar el compresor	3

18	Como usuario registrado quiero buscar y cargar un sonido	3
19	Como usuario registrado quiero utilizar mi teclado MIDI con el programa	4
20	Como usuario registrado quiero utilizar el metrónomo	4
21	Como usuario registrado quiero utilizar mi teclado MIDI con el programa	4
22	Como usuario registrado quiero guardar un sonido	4
23	Como usuario registrado quiero reproducir un archivo MIDI	4
24	Como usuario registrado quiero ver el espectro del sonido que estoy produciendo	5
25	Como usuario registrado quiero hacer login	5
26	Como usuario registrado quiero hacer logout	5
27	Como usuario no registrado quiero registrarme	5
28	Como administrador quiero añadir sonidos a la biblioteca por defecto	5
29	Como administrador quiero eliminar sonidos de la biblioteca por defecto	5
30	Como administrador quiero hacer login	5
31	Como administrador quiero hacer logout	5

4.5. Sprints Backlogs

A continuación, se dividen las historias de usuario del Product Backlog en los distintos Sprints. Para cada historia se realiza una estimación de los puntos de estas en función del esfuerzo que se estima para la realización de esa historia. Los puntos de historia son valores pertenecientes a la serie de Fibonacci.

4.5.1. Sprint 1

ID	Nombre de la historia	Prioridad	Puntos de historia
1	Como usuario registrado quiero utilizar uno o dos osciladores	1	5
2	Como usuario registrado quiero elegir el tipo de onda de un oscilador	1	1
3	Como usuario registrado quiero ajustar el volumen de los osciladores por separado	1	1
4	Como usuario registrado quiero modificar la envolvente de un oscilador	1	8

Identificador: HU.1	Como usuario registrado quiero utilizar uno o dos osciladores
Descripción: Un usuario registrado en el sistema puede utilizar uno o dos osciladores que generen una señal digital de audio, de manera que pueda encender y apagar los osciladores que quiera en cada momento, así como también pueda utilizar los dos a la vez.	
Estimación: 5	**Prioridad**: 1
Pruebas de aceptación:	

• Encender un oscilador y obtener un sonido • Encender los dos osciladores a la vez y obtener un sonido como suma de los dos osciladores • Apagar los dos osciladores y no obtener ningún sonido
Tareas: • Instalar NodeJS, ReactJS y ToneJS • Crear la infraestructura inicial del forntend en React • Crear dos osciladores • Crear una vista para los osciladores • Crear un botón de encendido y apagado en cada oscilador
Observaciones: Se elegirá una onda por defecto para los osciladores y se creará un botón que reproduzca una nota por defecto para realizar las pruebas.

Identificador: HU.2	Como usuario registrado quiero elegir el tipo de onda de un oscilador
Descripción: Un usuario registrado en el sistema puede elegir el tipo de onda de un oscilador entre una onda de tipo seno, seno, triangulo, cuadrado, o diente de sierra.	
Estimación: 1	**Prioridad**: 1
Pruebas de aceptación: • Elegir una onda seno en un oscilador y obtener un sonido senoidal. • Elegir una onda triangular en un oscilador y obtener un sonido triangular. • Elegir una onda cuadrada en un oscilador y obtener un sonido cuadrangular. • Elegir una onda diente de sierra en un oscilador y obtener un sonido de diente de sierra. • Si tengo los dos osciladores encendidos y elijo una onda distinta en cada uno el sonido que obtengo es la mezcla de los dos osciladores.	
Tareas: • Crear en cada oscilador un selector para el tipo de onda • Diseñar algoritmo para la mezcla de dos ondas distintas	
Observaciones:	

Identificador: HU.3	Como usuario registrado quiero ajustar el volumen de los osciladores por separado
Descripción: Un usuario registrado en el sistema puede ajustar el volumen de salida de cada oscilador entre -100dB y 0, es decir, puede ajustar el nivel de amplitud producido por cada oscilador.	
Estimación: 1	**Prioridad**: 1
Pruebas de aceptación: • Si asigno un valor de volumen a un oscilador superior a 0dB automáticamente se convierte en 0dB. • Si asigno un valor de volumen a un oscilador inferior a -100dB automáticamente se convierte en -100dB. • Si incremento los dB se incrementa el volumen, al contrario, si decremento • Si el volumen está a -100dB no se emite sonido.	
Tareas: • Crear en la vista de cada oscilador un Knob para el volumen • Diseñar algoritmo para controlar el volumen de los osciladores	
Observaciones:	

Identificador: HU.3	Como usuario registrado quiero modificar la envolvente de un oscilador
Descripción: Un usuario registrado en el sistema puede modificar la envolvente de un oscilador, es decir, puede modificar los valores Attack, Decay, Sustain, Release.	
Estimación: 8	**Prioridad**: 1
Pruebas de aceptación: • Si asigno un valor de Attack entre 0 y 2ms el inicio del sonido producido por un oscilador se ve modificado • Si asigno un valor de Decay entre 0 y 2ms el tiempo de caída producido por un oscilador se ve modificado. • Si asigno un valor de Decay entre 0 y 1ms el tiempo de sustain producido por un oscilador se ve modificado. • Si asigno un valor de Decay entre 0 y 5ms el tiempo de Release producido por un oscilador se ve modificado.	
Tareas: • Crear una envolvente por defecto. • Crear el algoritmo que para aplicar la envolvente. • Crear en la vista de cada oscilador un Knob para el Attack,Decay,Sustain,Release.	

Observaciones: La interfaz no permitirá a los usuarios introducir valores inferiores a los establecidos en las pruebas de aceptación.

4.5.2. Sprint 2

ID	Nombre de la historia	Prioridad	Puntos de historia
5	Como usuario registrado quiero tocar notas clicando en las teclas que aparecen en la pantalla	1	2
6	Como usuario registrado quiero tocar notas desde el teclado del ordenador	1	3
7	Como usuario registrado quiero tocar un acorde	1	5
8	Como usuario registrado quiero aplicar un efecto Reverb	2	3

Identificador: HU.5	Como usuario registrado quiero tocar notas clicando en las teclas que aparecen en la pantalla
Descripción: Un usuario registrado en el sistema puede tocar notas musicales clicando en el piano de la pantalla en sus respectivas notas. Solo podrá tocar por pantalla dos octavas.	
Estimación: 2	**Prioridad**: 1
Pruebas de aceptación: • Si clico en una tecla del piano de la pantalla suena una nota. • Si clico la una tecla del piano suena la nota correspondiente a la del piano. Por ejemplo, si toco Do suena Do. • Si clico en dos notas distintas se aprecia que son notas distintas. • Si cambio de tamaño la ventana el piano se ajusta a esta.	
Tareas: • Implementar método para tocar notas distintas. • Implementar un componente para la vista del piano. • Asociar a cada nota del piano una nota musical.	
Observaciones: Solo se mostrará por pantalla dos octavas del piano y un botón para subir o bajar de octava.	

Identificador: HU.6	Como usuario registrado quiero tocar notas desde el teclado del ordenador
Descripción: Un usuario registrado en el sistema puede tocar notas musicales pulsando en las teclas del ordenador, de manera que el teclado del ordenador simulara el de un piano. El usuario podrá tocar dos octavas simultáneamente.	
Estimación: 3	**Prioridad**: 1
Pruebas de aceptación: • Cuando toco una tecla suena una nota. • Si toco una tecla que no se corresponde con ninguna nota no se emite ningún sonido. • Cuando toco una tecla se indica por pantalla que nota estoy tocando. • Si toco simultáneamente una tecla del teclado y la de la interfaz prevalecerá la del teclado.	
Tareas: • Implementar método para hacer coincidir teclas con notas. • Implementar método para que se refleje en pantalla que nota/as estoy tocando al pulsar las teclas. • Gestionar la simultaneidad de eventos entre el piano de la interfaz y las teclas tocadas.	
Observaciones: Cuando el usuario toca una tecla del teclado y se corresponde con una nota se indica por interfaz que nota/as está tocando.	

Identificador: HU.7	Como usuario registrado quiero tocar un acorde
Descripción: Un usuario registrado en el sistema puede tocar un acorde, es decir, podrá tocar un conjunto de notas de manera simultánea y que suene a la vez.	
Estimación: 5	**Prioridad**: 1
Pruebas de aceptación: • Cuando toco una tecla suena una nota. • Cuando toco varias notas suena el acorde correspondiente, es decir, la suma de las notas individuales por separado de manera simultánea.	
Tareas: • Implementar método para gestionar la polifonía.	
Observaciones: Cuando el usuario toca una tecla del teclado y se corresponde con una nota se indica por interfaz que nota/as está tocando.	

Identificador: HU.8	Como usuario registrado quiero aplicar un efecto Reverb
Descripción: Un usuario registrado en el sistema puede aplicar un efecto de Reverb o reverberación a los sonidos producidos por los osciladores.	
Estimación: 3	**Prioridad**: 2
Pruebas de aceptación: • Cuando toco una tecla suena el sonido producido por la suma de los osciladores más e Reverb. • Si no tengo encendido el efecto de Reverb el sonido suena limpio.	
Tareas: • Implementar método para aplicar Reverb al conjunto de los dos osciladores. • Establecer un Reverb por defecto. • Implementar un componente para los efectos en la vista. • Implementar un componente dentro del componentes de los efectos para el Reverb. • Implementar un botón para encender y apagar el Reverb.	
Observaciones:	

Capítulo 5: Diseño

Capítulo 6: Implementación y pruebas

Una vez realizado el diseño del sistema software, se procede a explicar con detalle el proceso que comprende la implementación del mismo

6.1. Estudio de las tecnologías

6.1.1. Análisis de las diferentes tecnologías

Para llevar a cabo la implementación del sistema software que se requiere, se necesita una tecnología que nos ofrezca lo siguiente:

- Soluciones para desarrollar un sistema de tiempo real (multiprocesamiento, temporizadores, semáforos, señales de tiempo real, entrada/salida síncrona y asíncrona etc..)
- Métodos y variables para comunicarnos con el sistema de audio del

computador
- Métodos eficientes para el procesamiento de señales digitales (DSP)
- Documentación para el tratamiento de audio
- Facilidades para implementar una interfaz de usuario
- Facilidades para implementar gráficos complejos

Múltiples lenguajes de programación como Java o Python, ofrecen parcialmente soluciones a los puntos comentados, pero debido a la complejidad del procesamiento de señales digitales es preciso utilizar Frameworks y librerías nativas que simplifiquen esta tarea.
Existen dos lenguajes de programación que nos proporcionan lo expuesto en este punto:

- C++
- JavaScript

A continuación, se describen las ventajas e inconvenientes de estos lenguajes

6.1.1.1. C++

Ventajas

- Dispone de librerías para trabajar con el tiempo, como chrono.
- Ofrece soluciones bastante eficientes para el multiprocesamiento y la sincronización.
- Dispone de librerías para comunicarnos con el sistema de audio del computador.
- Dispone de librerías DSP, como IIPP de Intel.
- La gran mayoría de sistemas software para el procesamiento de audio están implementados en C++ o se basan en este.
- Existen librerías para el procesamiento de gráficos, como Open GL .
- Existen múltiples Frameworks para el desarrollo de interfaces.
- Permite aprovechar al máximo los recursos del ordenador.

Desventajas

- No es portable en nuestro sistema a realizar, necesitaremos realizar una versión para cada SO ya que el sonido se trata de distinta forma en cada uno de estos.
- Los Frameworks existentes no dan soporte a la solución que se busca, ya que están muy limitados a la realización de plugins para DAWS existentes. Con estos Frameworks solo podremos usar nuestro sistema software dentro de un DAW y no podremos realizar una interfaz desde cero.
- Si se usa esta tecnología, el resultado final será de bajo nivel por lo que la dificultad y el tiempo para realizarlo se eleva con creces.
- Debido a la ausencia de Frameworks específicos, se requiere demasiado conocimiento en cuestiones matemáticas y físicas para el tratamiento de las señales de audio digital.

6.1.1.2. JavaScript

Ventajas

- Portable, si se utiliza esta tecnología el software estará disponible para cualquier dispositivo con un navegador.
- Existe una API para este lenguaje llamada Web audio API, destinada a la creación de sonido digital en el ámbito del navegador, la cual se ajusta bastante bien al sistema a desarrollar, ya que ofrece soluciones a alto nivel de síntesis digital que no requerirían de un excesivo conocimiento en cuestiones matemáticas y físicas. Además, da soporte a la comunicación con el sistema de audio del computador.
- Existen Frameworks para trabajar con gráficos como Three js .
- Existen Frameworks que proporcionan herramientas para el desarrollo de una aplicación en la nube como React.
- En general, hay una alta disponibilidad de librerías.

Desventajas

- Las capacidades de tiempo real son más limitadas, los callbacks no están sincronizadas con precisión. La Web Audio API soluciona parcialmente este problema
- El ámbito de ejecución, se limita al navegador, por lo que los recursos estarán limitados a este.
- Poco software de audio utilizando este lenguaje.
- Da problemas para programas no triviales, al no ser fuertemente tipado.

6.1.2. Conclusión del análisis de tecnologías

Sopesadas las ventajas y desventajas de las tecnologías analizadas en el punto anterior, se llegan a las siguientes conclusiones:

1. Pese a que C++, ofrece más capacidades de tiempo real que JavaScript, la Web Audio API soluciona parcialmente el problema de la sincronización y el tiempo real de manera sencilla por lo que se podría llegar a alcanzar el grado de sincronización requerido.
2. El grado de dificultad en la implementación de la interfaz en JavaScript es más bajo
3. La ventaja de la total e inmediata disponibilidad de JavaScript aporta un gran valor añadido al producto final.
4. La Web Audio API ofrece muchas soluciones a un nivel más alto que C++, permitiendo que la implementación de este sea más asequible
5. Debido a la alta disponibilidad de librerías para JavaScript, podremos encontrar soporte y soluciones para gran variedad de los requisitos del programa.

A partir de estas primeras conclusiones, se llega a la conclusión final:

Se elige JavaScript como lenguaje de programación para la implementación del sintetizador, utilizando como entorno de ejecución **Nodejs.** Además se utilizan la

Web Audio API, para trabajar con las señales de audio, y los frameworks **React** para el Frontend y **Express** para el Backend.

6.2. Tecnologías empleadas

6.2.1. Lenguajes de programación

6.2.2. Entorno de Ejecución

6.2.3. Frameworks

6.2.3. Control de versiones

6.3. Instalación y ejecución

6.3.1. Instalación

Dado que el sistema está pensado para ser desplegado en un servidor se explicará el proceso de instalación para Windows y Linux. Antes de comenzar con la explicación es necesario que el computador en el que se está instalando el sistema software tenga acceso a Internet, ya que se requiere su uso durante este proceso.

Lo pasos de instalación del sistema software serán los siguientes:

1. **Instalar la última versión de Nodejs:**
    a. Si trabajamos con Windows:
        i. Descargamos el Instalador para Windows (.zip) desde el sitio web[2] de Nodejs
        ii. Descomprimimos el .zip y ejecutamos el archivo .exe
        iii. Una vez ejecutamos se nos abrirá el instalador de Windows y pulsamos en Siguiente.
        iv. Abrimos una terminal y ejecutamos *node -v* para comprobar si se ha instalado correctamente
    b. Si trabajamos con Linux ejecutamos por terminal lo siguiente:
        i. *sudo apt update*
        ii. *sudo apt install nodejs*
        iii. *nodejs -v* para comprobar que se ha realizado correctamente

2. **Instalar el gestor de paquetes npm de Nodejs:**
    a. Si trabajamos con Windows
        i. El gestor de paquetes se instaló junto con la instalación de Nodejs al ejecutar el .exe

[2] Web Nodejs: https://nodejs.org/es/download/

b. Si trabajamos con Linux ejecutaremos por terminal lo siguiente:
    i. *sudo apt install npm*

3. **Instalar Mongo DB:**
    a. Si trabajamos con Windows:
        i. Descargar la versión para Windows (.zip) desde el sitio web de Mongo DB
        ii. Descomprimimos el .zip y ejecutamos el archivo .exe
        iii. Una vez ejecutamos se nos abrirá el instalador de Windows y pulsamos en Siguiente.
        iv. Abrimos una terminal y ejecutamos *db.version()* para ver si se ha realizado correctamente la instalación
    b. Si trabajamos con Linux
        i. *sudo apt update*
        ii. *sudo apt install -y mongodb*

4. **Clonar el repositorio de Github con el código fuente**
    a. Tanto en Windows como en Linux ejecutaremos en una terminal
        i. git clone https://github.com/migueg/Sintetizador-Virtual-TFG-UGR.git

5. **Instalar todas las dependencias:**
    a. Tanto en Windows como en Linux :
        i. Nos situamos en la carpeta Frontend del repositorio clonado desde terminal y ejecutamos *npm install*
        ii. Nos situamos en la carpeta Backend del repositorio clonado desde terminal y ejecutamos *npm install*

Nota: El gestor de dependencias npm de Nodejs, instalará automáticamente en el servidor todas las dependencias que aparecen en el package.json que durante la etapa de desarrollo fueron añadidas a este.

6.3.2. Ejecución

6.4. Implementación

A continuación, se va a proceder a una explicación al detalle del proceso de implementación del sistema en cuestión.
A rasgos generales, la implementación se ha dividido en dos grandes partes o módulos: Backend y Frontend, siendo este último el más completo, laborioso y de más difícil desarrollo.
Por consiguiente, esta sección se divide en esas dos partes comentadas.

6.4.1. Frontend

El Frontend es la parte de nuestro sistema que se ejecutará directamente en el navegador del cliente. A grandes rasgos proporciona la interfaz, que el usuario utilizará para interaccionar con el sistema software y una serie de clases y métodos que harán cumplir los objetivos del sistema.

En la introducción a la sección de implementación se ha comentado que la parte

correspondiente al Frontend es la más completa y laboriosa. Gran parte del procesamiento del sistema se lleva a cabo en el Frontend, es decir, gran parte del sistema se ejecutará en el navegador del cliente.

Para cumplir con los objetivos del sistema se necesita buscar la inmediatez, la mayor sincronía posible y evitar los retardos, ya que la experiencia de tocar un instrumento o en este caso un sintetizador virtual se vería afectada con creces. Además, como el sistema se trata de un sistema que genera y reproduce señales de audio se necesita trabajar directamente con el contexto de audio del dispositivo del cliente, es decir, los métodos para reproducir señales de audio tienen que ejecutarse en su dispositivo.
Es por esta razón por la que la mayor parte del procesamiento se realiza en el Frontend, a diferencia de las arquitecturas cliente servidor clásicas que se realizan en el Backend.
Al trasladar la mayor parte del procesamiento al Frontend, se reduce al mínimo cualquier retardo que se pudiera producir en una llamada al servidor de Backend y se realiza una comunicación directa con el dispositivo del cliente que es requerido para reproducir las señales de audio generadas.

A continuación, se explica con detalle todas estas cuestiones comentadas.

6.4.1.1. Estructura

6.4.1.2. Sintetizador

Para lograr la generación y reproducción de señales de audio digital necesitamos emular el comportamiento de un sintetizador analógico. Como se mencionó en la introducción (ver Capitulo 1), hay varios tipos de síntesis de audio. En este sistema se va a implementar una **síntesis aditiva,** de manera que la señal de audio resultante será la suma de todas las señales generadas.

La señal que el usuario percibirá cuando utilice el sistema será el resultado de sumar:

- Dos osciladores, que contendrán a su vez un número determinado de voces, es decir, de frecuencias de oscilación (ver sección sadfd).
- Efectos, que modificarán la señal producida por los osciladores.

Para representar el sintetizador en el sistema se utiliza la **clase Synth.** Esta clase constituirá una clase fachada que comunicará los elementos del controlador con la vista. Además, es Singleton, ya que solo habrá una única instancia de esta en todo el sistema.

Cada vez que el usuario ejecuta interaccione con la vista, está llamará a un método de la clase Synth.

6.4.1.3. Generación de audio

6.4.1.4. Reproducción de audio sin MIDI

Capítulo 7: Conclusiones y vías

futuras

Capítulo 8: Bibliografía final

https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/ipp.html

Ingeniería de sonido: conceptos, fundamentos y casos prácticos
Zafra, Julián
RA-MA Editorial

https://xferrecords.com/products/serum -SERUM

https://midi.city/ Midi.city

https://tonejs.github.io/ - Tonejs

https://es.reactjs.org/ react

https://expressjs.com/es/ express