UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería del Software

EGAW: Estación de trabajo de audio guiada por el oído
EGAW: Ear-guided audio workbench

Realizado por
Jaime Ezequiel Rodríguez Rodríguez

Tutorizado por
Javier Troya Castilla

Departamento
Lenguaje y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, junio de 2025

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
Graduado en Ingeniería del Software

EGAW: Estación de trabajo de audio guiada por el oído

EGAW: Ear-guided audio workbench

Realizado por
Jaime Ezequiel Rodríguez Rodríguez

Tutorizado por
Javier Troya Castilla

Departamento
Lenguaje y ciencias de la computacion

UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO DE 2025

Fecha defensa: julio de 2025

Resumen

El trabajo digital del audio conlleva múltiples fases en función del producto que se esté elaborando. En el desarrollo de la música se suele dividir en composición, grabación, mezcla y máster.

La mezcla busca combinar y balancear las pistas de audio individuales con el fin de alcanzar un resultado compacto y cohesivo. El proceso de máster parte del resultado de la mezcla, para dar lugar a un producto listo para la distribución y adaptado para poder reproducirse en una gran variedad de dispositivos sin experimentar distorsión en la escucha.

Estos procesos pueden resultar abstractos debido a las dificultades innatas del humano para comprender aquello que es imperceptible a la vista. A su vez, se pueden tornar extremadamente técnicos. A lo largo de los años, los profesionales han diseñado sistemas para visualizar valores cuantificables del sonido, como los decibelios o la imagen estéreo, entre otros.

La problemática surge cuando la exposición a estos medios visuales es tan alta que, de forma inconsciente, el audio pasa a un segundo plano. Los técnicos pueden acabar centrándose en conseguir un resultado cuantificable en lugar de plantear como cuestión principal si el producto está sonando adecuadamente.

El objetivo de este trabajo es proponer una alternativa a las herramientas de software existentes dedicadas al procesamiento del audio. Se plantea un diseño novedoso que tiene como filosofía principal que el oído nos guíe en todo momento durante la realización de estos procesos, y que lo visual sea algo puramente complementario y no el foco de nuestra atención. Esta política no solo orquestará el diseño del sistema, sino que también lo hará más accesible a nuevos usuarios de manera orgánica.

Palabras clave: Audio, Mezcla y Máster, Usabilidad, Oído

Abstract

Working with digital audio involves multiple phases, depending on the product being created. The process of music creation could be often divided into four stages: composition, recording, mixing and mastering.

Mixing involves combining and balancing individual audio tracks to achieve a cohesive and blended result. Mastering, on the other hand, starts with the mixed track and focuses on preparing the final product for distribution. It ensures the audio can be played effectively on different devices and in various environments, with no perceptible distortion for the listener.

This process can seem abstract due to the human tendency to struggle with concepts that aren't visually perceptible. Additionally, it can become highly technical. Over time, professionals have developed systems to visualize measurable aspects of the sound signal, such as decibel levels and stereo imaging.

Problems arise when users rely primarily on visual representations of audio, causing the sound itself to become an unconscious secondary concern. Audio engineers might focus on quantifiable results rather than asking themselves whether the result being produced sounds good.

The objective of this project is to offer an alternative to existing audio-focused software tools. The core philosophy of this new design is to prioritize auditory guidance throughout the mixing and mastering process, with visual feedback serving as a complementary aid. This policy will not only guide the system's design but also will make it more accessible to new users.

Keywords: Audio, Mixing and Mastering, Usability, Earing

Índice

1 Introducción

En este breve capítulo se realizará una introducción tanto al problema identificado como a la solución propuesta, que se desarrollarán a lo largo de este documento. En las próximas páginas desarrollaremos de manera introductoria la motivación del proyecto, el problema planteado, los objetivos propuestos y la estructura del propio documento.

1.1. Motivación

Hoy en día la industria musical representa una de las mayores fuentes de entretenimiento y cultura a nivel internacional, siendo considerada el quinto mercado a nivel mundial [1]. Es por lo mismo que para el desarrollo de las actividades necesarias en el sector, contamos con una gran variedad de profesionales implicados en los proyectos artísticos. El abanico va desde administrativos y gestores encargados de manejar presupuestos y la economía relativa a la industria, hasta los propios artistas centrados en la composición o interpretación de la música en sí, pasando por directores de marketing encargados de crear estrategias personalizadas de acuerdo a la intencionalidad del artista. En toda esta gran variedad de perfiles involucrados en proyectos musicales encontramos a los ingenieros de sonido o ingenieros de mezcla; estos se encuentran en un punto algo difuso entre ser técnicos y a la vez creativos en una misma tarea. Son los encargados de mezclar adecuadamente todas las pistas que forman una composición musical para conseguir que el producto pase de ser una demo a conformar una pista de audio que cohesione todos los elementos adecuadamente.

El proceso de mezcla tradicionalmente se hacía de manera analógica, haciendo uso de las reconocibles mesas de mezcla y procesando la señal de manera orgánica mediante sistemas como compresores o reverberadores, siendo estos complejos aparatos electrónicos dedicados a estas tareas en específico [2]. Actualmente, en una era de absoluta dominancia

de los sistemas digitales, este proceso también se ha digitalizado. A día de hoy se recurre principalmente a entornos software llamados DAW (*Digital Audio Workstation*) [3]. Estas herramientas nos permiten procesar la señal de audio de manera digital.

Las herramientas existentes son muy variadas, algunas de ellas se centran más en la parte compositiva y otras están enfocadas en mayor medida a facilitar los procesos de mezcla. De igual forma, cuando exploramos el mercado existente, encontramos que estas herramientas tienden a sobrecargar visualmente al usuario con medidores, parametrizaciones o formas de onda; fomentando de manera implícita a los usuarios a tomar decisiones basadas en lo visual más que en lo auditivo. Este enfoque generalizado puede deberse a nuestra forma de entender la interacción con los sistemas digitales, donde la mayor parte de la información se obtiene visualmente. Esta dependencia puede afectar negativamente al juicio auditivo, acomodando a los usuarios a patrones visuales.

Por ello, la motivación principal de este proyecto consiste en proporcionar al mercado un sistema software diseñado teniendo en cuenta las problemáticas expuestas. Este sistema no se plantea como un programa aislado, sino como un conjunto de productos software enfocados todos a través de un prisma que revoca la inundación de los sistemas de audio digitales con estímulos visuales.

1.2. Planteamiento del problema

El proceso de mezcla constituye una parte fundamental en la postproducción de composiciones musicales [4]. Este tratamiento del audio busca proveer a la pieza como conjunto una cohesión auditiva de todos los elementos que la forman [5]. La mezcla se caracteriza por plantearse como un proceso técnico y subjetivo al mismo tiempo. En este procesamiento se trabajará con factores que requieren que el usuario profundice en aspectos del sonido como son volumen, frecuencias, dinámica o posicionamiento de los elementos; a su vez, cómo han de tratarse estos elementos en favor de la propuesta artística resulta algo subjetivo, de forma que el criterio del técnico quedará proyectado sobre el producto final.

La sobreestimulación visual que encontramos en las herramientas software actuales puede guiar de manera errónea a resultados no satisfactorios, o desviar el componente subjetivo del proceso de mezcla hacia un enfoque puramente técnico, donde enmarcar el resultado audible en unos parámetros establecidos puede dañar el producto final. En los

sistemas actuales prevalece un enfoque sistemático y paramétrico a la hora de trabajar el audio, sobreexponiendo al usuario a un nivel de detalle e información que en la contraparte analógica no estaría disponible ni sería necesaria para realizar las tareas adecuadamente.

Como hemos planteado, esta casuística puede llegar a dañar de forma inconsciente el resultado audible, desviando la atención de los usuarios hacia soluciones numéricas y enmarcadas en cuantificaciones sobre productos sensoriales y de tal calibre de subjetividad como es el arte. Este problema no solo repercute en el desarrollo de la fase de mezcla en sí, sino que también afecta a la forma en la que se aprende y en la que nuevos usuarios interactúan con los sistemas. Futuros técnicos en formación o cualquier otro usuario que desee aprender sobre el proceso de mezcla puede verse inclinado a seguir ciertas reglas numéricas de manera sistemática sin plantearse en cada decisión si realmente es necesario o no en base al objetivo establecido. Esto se propicia, en parte, por la facilidad y predisposición de los sistemas a cuantificar todo parámetro posible alrededor del audio.

La solución propuesta a esta problemática plantea un enfoque mucho más natural, despejando de los estímulos sensoriales del usuario todo aquello que no sea puramente necesario para interactuar adecuadamente con el equipo digital. El proyecto software a desarrollar no solo se idea como un único programa, sino como un entorno virtual en el que este proceso técnico puede ser elaborado de principio a fin. Esta solución dará lugar a un conjunto de elementos software predispuestos a facilitar un entorno natural para la mezcla de audio donde se ponga en valor la toma de decisiones y el criterio subjetivo de los técnicos en sintonía a la adecuación necesaria basada en la propuesta artística que se esté trabajando.

1.3. Objetivos

El objetivo principal del proyecto es construir un entorno completo para el desarrollo de las tareas de mezcla centrado en dar al usuario una experiencia guiada por el oído, donde se fomentarán las decisiones tomadas en base a la escucha y no a la retroalimentación visual. Este enfoque pretende alinear el software con el producto final que se produce, ya que, en el desarrollo de composiciones auditivas, el sentido que debe prevalecer en la toma de decisiones ha de ser el oído.

El objetivo primario es desarrollar un sistema base sobre el que los usuarios puedan

trabajar el audio. Este sistema se orientará como un programa principal en el que cargar y colocar en el tiempo pistas de audio para, a posteriori, organizarlas y enrutarlas en distintos canales; creando así una base sólida sobre la que cimentar todo el proyecto. A partir de ahí se desarrollarán las diferentes herramientas para el procesamiento digital de la señal de acuerdo a la filosofía establecida.

En nuestro sistema base encontramos tres entornos pensados de acuerdo a distintas tareas a realizar durante el proceso de mezcla.

En primer lugar, la pantalla que emula una mesa de mezcla, en ella situamos un canal principal que hace de bus donde la señal recibida representa la salida de audio final perceptible por el usuario. En este entorno dispondremos de tantos canales como deseemos, pudiéndolos clasificar como canal de audio, bus de mezcla o bus de procesamiento. Todos ellos tienen un propósito individual; como desarrolladores, nuestro objetivo es diferenciar claramente las herramientas para que todas ellas sean igual de valiosas para el usuario. El canal de audio servirá como lector y reproductor de archivos, permitiendo a los usuarios cargar ficheros de audio para su procesamiento. El bus de mezcla tiene como propósito principal agrupar señales, es decir, transformar N entradas de audio en una sola salida. El bus de procesamiento se presenta como un canal de una sola entrada y una salida, en el que poder alojar herramientas de procesamiento de audio y que estas actúen sobre la señal recibida para producir el sonido deseado a través de estos efectos. Esta estancia de nuestro entorno nos permitirá organizar el procesamiento y enrutado de la señal de acuerdo a las decisiones del usuario.

A continuación, encontramos el entorno temporal, donde podemos ajustar los canales de audio en el tiempo para que se reproduzcan en el momento deseado, visualizando además el flujo del tiempo. El usuario tendrá la posibilidad de manejar la duración del audio, pudiendo ajustar el momento de inicio o de fin dentro de su propio fichero de audio.

Finalmente, la representación visual mostrada en la tercera pantalla será una sala de escucha. Se trata de una pantalla vacía donde se incentiva al usuario a simplemente escuchar atentamente el estado de su proyecto, permitiéndole redirigir el foco de atención hacia la percepción auditiva.

Además, a lo largo del proyecto se plantea el desarrollo de pequeños productos derivados que se traten de procesadores de la señal, como pueden ser ecualizadores, compresores

o reverberadores, entre otros. Este sistema de efectos complementa la base software establecida, configurando un entorno para la mezcla de audio al completo y ofreciendo al usuario la experiencia deseada.

Todos los objetivos establecidos se complementan confeccionando el producto deseado final, que se trata de este entorno virtual completamente enfocado en ofrecer una experiencia guiada por el oído a la hora de la elaboración de procesos técnicos en el audio, como la mezcla.

1.4. Estructura del documento

El documento se compone de ocho capítulos, teniendo en cuenta este primer capítulo de introducción. Los capítulos se estructurarán en el siguiente orden:

- *Capítulo 1. Introducción*

  Este capítulo inicial realiza una breve descripción del contexto del proyecto, del problema que se pretende resolver y de cuáles son los objetivos perseguidos con la realización de este.

- *Capítulo 2. Estudio del estado del arte*

  En este capitulo se realizará un análisis sobre la situación actual desde diferentes puntos de vista con respecto al problema planteado. Se revisarán estudios sobre la percepción auditiva, se hará una revisión de la posición artística de acuerdo al marco técnico en la música y ,por último, se explorarán las alternativas software propuestas en el mercado.

- *Capítulo 3. Tecnologías Estudiadas*

  Este capítulo revisa las tecnologías estudiadas para el desarrollo del proyecto. En ella vemos desde el prisma técnico las herramientas evaluadas para el desarrollo del proyecto. Especificaremos qué tecnologías han sido seleccionadas y por qué descartamos el uso de otras.

- *Capítulo 4. Metodologías*

  El capítulo muestra metodologías extendidas para desarrollar proyectos software.

Las metodologías mencionadas en la sección de la memoria conforman la base de la metodología aplicada para el desarrollo.

- *Capítulo 5. Objetivos y Requisitos*

  En este capítulo hacemos una clara disposición de los objetivos a los que pretendemos llegar en el proyecto. También, ejemplificaremos la forma en la que se han tomado los requisitos, tanto funcionales como no funcionales, en forma de historias de usuario.

- *Capítulo 6. Desarrollo del proyecto*

  Siendo el más extenso, este capítulo profundiza en todos los aspectos del desarrollo del proyecto. Quedan descritos los aspectos más técnicos, como son los sistemas internos ideados, la experiencia de usuario que brinda cada pantalla disponible, la linea de "plugins" ideada para el sistema, la construcción del instalador y las pruebas de calidad realizadas.

- *Capítulo 7. Conclusiones y Líneas Futuras*

  Capítulo que concluye la memoria de este trabajo. Encontramos las conclusiones finales del proyecto realizado y planteamos las líneas futuras para continuar el desarrollo en un futuro.

2

Estudio del estado del arte

A lo largo de este capítulo realizaremos un análisis de los avances científicos que tratan cómo afecta la visión a la percepción auditiva. También estudiaremos cómo el proceso de mezcla tiene una componente subjetiva y artística pese a la tecnicidad implícita del proceso. Por último, repasaremos algunas de las alternativas de software más populares en el mundo del procesamiento de audio.

2.1. Percepción auditiva alterada por la vista

La percepción humana es un fenómeno intrínsecamente complejo que se fundamenta en la integración de la información proveniente de múltiples sentidos. Esta mezcla sensorial es clave para construir una representación coherente y unificada de nuestro entorno, permitiendo la interacción natural con el mismo [6]. En todo proceso sensorial encontramos un elemento clave: la atención, una función que dirige nuestros recursos hacia estímulos específicos, facilitando la percepción al filtrar la información irrelevante.

Un concepto angular para este estudio es comprender que la atención no se trata de un recurso ilimitado; su distribución entre los distintos sentidos es inherentemente dinámica. Cuando un sentido, como la vista, demanda una mayor porción de estos recursos, la disponibilidad para otros sentidos, como el oído, se reduce de manera natural [7]. Esta competencia por los recursos atencionales subraya la naturaleza finita de la atención y su impacto directo en la percepción multisensorial. Un ejemplo cotidiano con el que muchos se podrán identificar es la realización de tareas con música en segundo plano; de manera natural, cuando focalizamos nuestra atención en tareas complejas, desatendemos el oído y, cuando volvemos a ser conscientes de ello, de repente, el disco ha pasado de canción.

La investigación neurocientífica ha demostrado de forma consistente que la visión es el sentido predominante para los seres humanos que no presentan dificultades sensoriales. Este fenómeno se ilustra mediante el Efecto Colavita, un hallazgo robusto en neurociencia que indica cómo los estímulos visuales tienden a ser priorizados sobre los estímulos auditivos que se presentan simultáneamente, lo que a menudo conduce a la ignorancia de la información auditiva. Los experimentos originales de Francis B. Colavita en 1974 revelaron que los participantes respondían predominantemente al estímulo visual en pruebas bimodales, incluso cuando un estímulo auditivo estaba presente [8].

En entornos complejos y visualmente ricos, como un estudio de mezcla de audio, la soberanía visual innata puede llevar a una subestimación inconsciente de la información auditiva, incluso cuando esta es el foco objetivo de la tarea. Si el cerebro tiene un sesgo inherente a dirigir la atención hacia la visión, entonces en un entorno donde la interfaz de una Estación de Trabajo de Audio Digital (DAW) presenta una gran cantidad de estímulos visuales [9] , esta tendencia natural del cerebro actuará, desviando la atención del ingeniero de sonido de forma inconsciente y afectando la calidad de su juicio auditivo.

En todos los procesos mentales relacionados con la mezcla no solo interviene la distribución de la atención, sino que también entra en juego el factor psicológico y los sesgos. Los sesgos cognitivos son atajos mentales adheridos a la cognición humana que pueden influir significativamente en la percepción del sonido y en las decisiones de mezcla, especialmente bajo la influencia de estímulos visuales. Los sesgos son patrones sistemáticos que afectan a la racionalidad en el juicio. Se describen como atajos mentales o errores en el procesamiento de la información por parte del cerebro que conducen a conclusiones incorrectas y a una visión distorsionada de la realidad [10]. Estos sesgos son inherentes a la cognición humana y afectan a todas las personas, independientemente de su inteligencia o preparación; nadie escapa a ellos.

En el ámbito de la mezcla de audio, la escucha crítica es la habilidad más fundamental [11]. Este proceso implica un análisis auditivo activo y profundo, donde se evalúan y comprenden elementos sonoros como el timbre, la altura, el volumen, la espacialidad y sus complejas interacciones. La capacidad de percibir la música de manera objetiva y con un alto nivel de detalle es crucial para tomar decisiones sobre el procesamiento de las señales de audio, lo que se traduce directamente en la calidad final del producto.

Las interfaces gráficas de usuario (GUI) de los DAWs y los plugins, aunque diseñadas para facilitar el trabajo y proporcionar retroalimentación visual, pueden introducir sesgos perceptuales significativos. La estética, el diseño visual y la organización de una interfaz pueden crear una sensación que influye en cómo un ingeniero percibe el sonido. Por ejemplo, una interfaz con un diseño "vintage" podría sugerir calidez sonora, mientras que una futurista podría evocar una percepción de limpieza. Esto se relaciona con el efecto halo, donde la apariencia visual de un producto influye en la percepción de su usabilidad y calidad general [12]. El desorden visual o la simplicidad de la interfaz también pueden impactar la percepción del sonido; un entorno menos saturado visualmente podría llevar a una percepción de sonido más claro, simplemente porque la experiencia visual es menos distractora.

Los medidores de audio como los VU, RMS, True Peak, Loudness y los analizadores de espectro [13] son herramientas visuales cruciales que proporcionan retroalimentación sobre el nivel, el rango dinámico y el contenido de frecuencia de una señal de audio. Sin embargo, confiar únicamente en la retroalimentación visual es un error común; el oído debe ser el juez final. Los medidores de sonoridad, por ejemplo, utilizan la tecnología para simular la percepción auditiva humana, pero no reemplazan la escucha crítica para las decisiones artísticas y relacionales en la mezcla. Los medidores realizan "juicios de valor" técnicos, mientras que el oído toma "decisiones artísticas". Un error común es confiar demasiado en las representaciones visuales del ecualizador, que pueden ser engañosas y llevar a una mezcla sobreprocesada o antinatural [14]. Se recomienda utilizar los analizadores de espectro para identificar problemas potenciales o para comparar la mezcla con pistas de referencia, pero siempre validando con la escucha crítica. Las herramientas visuales como los medidores y analizadores de espectro son valiosos apoyos en el flujo de trabajo, pero deben usarse para confirmar lo que se escucha, no para dictar las decisiones.

Para contrarrestar la dominancia visual y desarrollar una escucha más precisa y objetiva en la mezcla de audio, los ingenieros pueden implementar diversas estrategias prácticas basadas en la evidencia. La práctica de la "mezcla a ciegas" [15] implica reducir o eliminar intencionalmente la retroalimentación visual durante ciertas fases del proceso de mezcla para forzar una mayor dependencia de la percepción auditiva. Esta técnica ofrece beneficios significativos: aumenta la objetividad en las decisiones de mezcla al eliminar el sesgo

visual y las expectativas previas. Al suprimir la estimulación visual, la mente puede enfocarse profundamente en las sutilezas auditivas, ya que el sentido visual puede desviar la atención. Además, ayuda a discernir si un ajuste o un cambio de equipo realmente mejora el sonido o si la percepción está influenciada por la visualización o el efecto placebo.

A su vez se requiere del ingeniero de sonido cierto nivel de entrenamiento, y ser capaz de trabajar el oído para percibir lo preciso según la situación. El entrenamiento auditivo es un proceso deliberado para desarrollar y refinar la capacidad de escuchar, distinguir e identificar los diferentes elementos que componen un sonido [16]. Esta práctica es fundamental para la producción de audio y puede mejorar drásticamente las habilidades de mezcla y masterización, a menudo más que cualquier actualización de equipo. El entrenamiento auditivo es una inversión en la neuroplasticidad del cerebro, permitiendo una reconfiguración de las vías neuronales para priorizar la información auditiva, reduciendo la dependencia de las muletas visuales. Al igual que un músculo se fortalece con el ejercicio, el oído se vuelve más preciso y sensible con un entrenamiento constante. Algunas de las áreas en las que entrenar el sentido del oído son: la identificación de frecuencias, la dinámica acústica, el espacio estéreo y la detección de la distorsión y artefactos.

2.2. La mezcla como proceso subjetivo y la relación de los artistas compositores con el ámbito técnico

Los ingenieros de sonido, a menudo, se encuentran en la búsqueda del balance ideal entre la parte técnica y la parte artística de su trabajo, especialmente cuando el proyecto encargado trata de una propuesta artística. Multitud de ingenieros destacan cómo han de transformarse y unificar el conocimiento técnico del área de trabajo con la interpretación subjetiva.

Rob Mayzes destaca que en el fondo es su componente subjetiva la que ha de predominar, ya que su objetivo es satisfacer la parte artística del proceso y dar lugar a música atractiva para el público [17]. Medios como *Rocky Mountain College of Art* enfocan la mezcla como un acto de equilibrio donde las herramientas técnicas son utilizadas para esculpir el sonido [18]. A su vez, la relevancia de estos profesionales es tal que son considerados por muchos como “el ingrediente secreto”, capaces de transformar grabaciones e

ideas en crudo en piezas uniformes y pulidas [19]. La componente subjetiva de la mezcla puede entenderse como la forma en la que los ingenieros de sonido interpretan la visión del artista para construir un paisaje sonoro que mantenga el sentimiento y la emoción original.

Esta forma de pensar, donde lo subjetivo es el fin de los medios técnicos, se trata de algo común entre los profesionales. El galardonado Dave Pensado, quien ha trabajado con artistas como The Black Eyed Peas, Beyoncé o Whitney Houston, entre otros, se presenta en una entrevista para ProSoundWeb [20] bajo la frase: *"I'm not selling my engineering, I'm selling my taste"*, lo que se podría traducir como "No vendo mi conocimiento de ingeniero, vendo mi gusto". Esta frase resume perfectamente las expectativas sobre el trabajo a realizar por los ingenieros de mezcla y la realidad del valor que aportan estos profesionales. A lo largo de la entrevista se le pregunta si no le inquieta revelar sus técnicas de mezcla; Dave reitera su forma de pensar, y es que, pese a tener encargados que aprenden directa y personalmente de él, ninguno de ellos suena como él. Explica que esto es debido a que cada uno tiene su gusto musical particular que actúa como una lente que proyecta el carácter personal del técnico sobre la mezcla a la hora de trabajar. Dave no solo reconoce la importancia del gusto individual, sino que, además, incentiva a desarrollarlo por delante del ámbito técnico; citándole: *"If you have two hours available, the best use of your time is to listen to as many records as possible instead of just learning techniques"*, traducido, viene a explicar que es mucho más interesante como profesional centrarse en la escucha y en aprender auditivamente en lugar de simplemente aprender técnicas sin un contexto.

Podemos confeccionar que los ingenieros de mezcla son el soporte técnico encargados de traducir a través de su visión la amalgama de la creación artística en una imagen sonora estructurada. Esto es algo que refuerza Josh Rogosin para el medio Mic the Snare [21], ingeniero de sonido principal del formato Tiny Desk de NPR [22]. En esta entrevista, Josh deja claro que la idea es establecer un portal mediante el cual los artistas puedan compartir su historia al mundo. El objetivo de Tiny Desk es establecer un medio personal y cercano donde los artistas aportan las composiciones y la interpretación, y los ingenieros usan todo su conocimiento técnico para cumplir con el fundamento que da lugar al formato, que es brindar a los oyentes una interpretación natural y orgánica de la música. Como vemos, pese a la complejidad técnica envuelta en este proceso el valor real que hace único

el proyecto no deja de ser el prisma a través del cual se visualiza la obra.

Ahora bien, ¿por qué si se trata de una labor tan subjetiva existe personal dedicado a esta misma y no la realizan los propios artistas? Esta cuestión es natural, y es que tras todo lo explicado previamente parece evidente que los artistas podrían dedicarse a este proceso manteniendo a la perfección su visión creativa. La respuesta es compleja, ya que cada proceso artístico es único; puede darse el caso en el que un mismo artista sea capaz de trabajar todos los procesos creativos sin la necesidad de delegar en terceros, aunque lo común y recomendado suele ser que el artista se dedique a la composición y confíe en un perfil técnico para el proceso de mezcla [23]. Esto viene dado por la amplia diferencia que existe entre el conocimiento necesario para la composición musical y aquel necesario para la mezcla. A modo general, la composición requiere del manejo de conceptos musicales como la sonoridad, harmonía o conocimiento sobre estructuración de composiciones; mientras tanto, la mezcla necesita de la especialización en ideas como rangos frecuenciales, dinámica acústica o espacio estéreo. Estos conocimientos involucran áreas muy alejadas dentro del conocimiento sobre el sonido, para los profesionales suele resultar imposible especializarse en ambos campos y mantener las exigencias de la industria en sus tareas.

Encontrar el equipo de trabajo adecuado es una tarea clave para los artistas, especialmente cuando se trata de tareas que repercuten directamente al producto final, como es la mezcla. Hay casos en los que los artistas no se ven conformes con los resultados, ya que no comparten la misma visión que sus ingenieros, ejemplo de ello es el conjunto SwaggerBoyz que en Grimey TV [24] relataban que tras la mezcla y el master escuchaban el producto con demasiada claridad con respecto a lo que querían proyectar a través de un sonido más sucio. Es por ello que los artistas no pueden desentenderse al completo de estos procesos y han de ser cuidadosos sobre cómo trabajan su producto y la forma de comunicarse con el área técnica del desarrollo. El glosario de tecnicismos es extenso pero se recomienda a la componente creativa que se familiarice con ciertos conceptos para mejorar la comunicación en este trabajo en equipo [25]. Un caso de éxito es Rosalía con su disco Motomami, la artista maneja conceptos de postproducción y transmite su visión minimalista del disco que explica en la entrevista de Jaime Altozano [26] al ingeniero de mezcla Manny Marroquin encargado del proyecto.

2.3. Alternativas software existentes

A lo largo de este apartado del capítulo realizaremos una recopilación de los elementos principales de las principales herramientas software para el desarrollo de tareas relacionadas con el audio. Expondremos su funcionamiento y alguna de sus características en referencia a cómo se comunica al usuario la información necesaria para trabajar.

2.3.1. FL Studio

FL Studio, anteriormente conocido como FruityLoops, es un Entorno de Producción de Audio Digital (DAW) desarrollado por la compañía belga Image-Line [27]. Desde su lanzamiento, ha evolucionado hasta convertirse en una plataforma integral y ampliamente reconocida en la industria musical, utilizada por un espectro diverso de productores, desde aficionados hasta profesionales, en géneros que abarcan desde la música electrónica o el hip-hop hasta composiciones orquestales.

Su reputación se basa en un flujo de trabajo intuitivo y altamente visual, junto con una notable versatilidad que le permite abordar diversas tareas de producción musical. Entre sus funcionalidades clave se incluyen la grabación y edición de audio, la mezcla y masterización, así como la programación de baterías y melodías mediante la creación de patrones y secuencias.

La interfaz de usuario de FL Studio (ver Figura 1) se caracteriza por su modularidad y por la interconexión de sus ventanas principales, lo que facilita un flujo de trabajo dinámico. Los componentes esenciales que conforman esta interfaz son una barra de herramientas y un navegador de archivos como componentes básicos; por otra parte, como editores reales de audio encontramos la lista de reproducción, el visor de canales y el mezclador. La barra de herramientas proporciona acceso a las funciones generales del software, incluyendo herramientas de reproducción y pausa, control del metrónomo y diversas herramientas de edición global del proyecto. El navegador se trata de una herramienta fundamental para la gestión de archivos; permite explorar y organizar eficientemente archivos de audio, instrumentos virtuales, efectos y los propios proyectos del usuario [28].

La lista de reproducción (ver Figura 2) actúa como el lienzo donde los elementos musicales se ensamblan para formar la estructura completa de una canción. Permite arrastrar

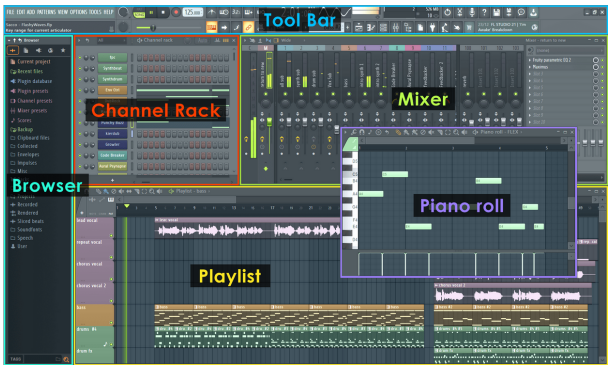


Figura 1: Interfaz principal de FL Studio, extraída del manual web oficial.

y manejar patrones MIDI y ficheros de audio, además de automatizaciones, para construir arreglos en el audio. Ofrece herramientas directas para manipular el audio, como funciones de recorte, estiramiento temporal y ajustes de volumen.

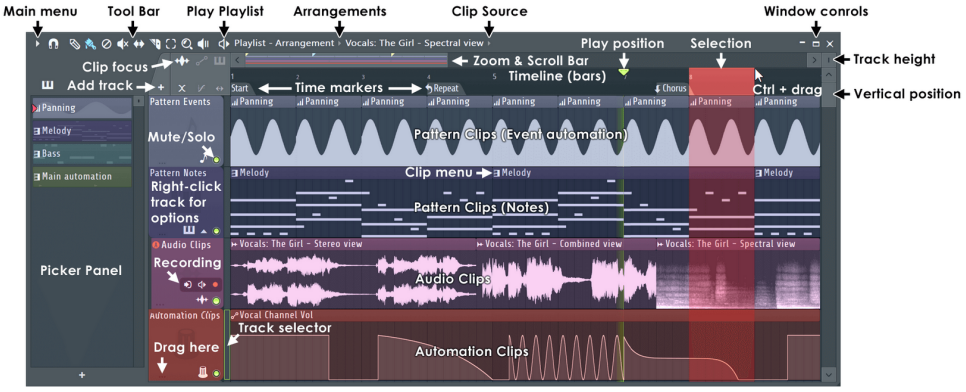


Figura 2: Lista de reproducción de FL Studio, extraída del manual web oficial.

El visor de canales es esencial para la construcción rítmica y melódica dentro de FL Studio. Cada fila en el visor de canales representa un canal que puede alojar un instrumento virtual, un archivo de audio o una automatización.

La consola de mezcla central de FL Studio (ver Figura 3) es crucial para dar forma al sonido final. Cada pista en el mezclador puede recibir la señal de uno o varios canales internos o de entradas de audio externas. Permite ajustar con precisión los niveles de volumen y la posición estéreo de cada elemento sonoro. Se pueden insertar efectos de audio directamente en cada pista, o utilizar envíos para aplicar efectos compartidos en varias pistas. Ofrece opciones avanzadas de manejo y envío de audio, permitiendo la creación de subgrupos y cadenas de procesamiento complejas. La pista “Master” es la

salida final de audio, donde se aplican los últimos procesos para optimizar el sonido antes de la exportación.

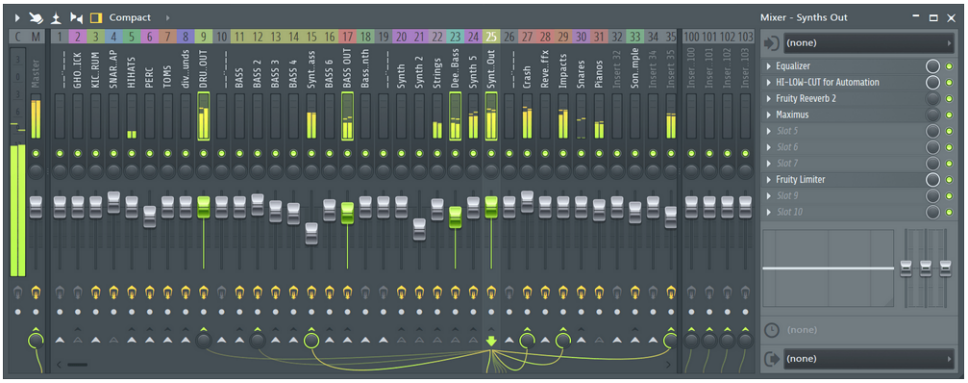


Figura 3: Consola de mezcla de FL Studio, extraída del manual web oficial.

FL Studio se erige como una solución integral y robusta para la producción musical contemporánea. Su diseño de interfaz intuitivo, combinado con un conjunto completo de funcionalidades para la secuenciación, mezcla, edición y masterización, lo posiciona como una herramienta accesible tanto para quienes se inician en la producción como para profesionales experimentados.

2.3.2. Ableton Live

Ableton Live constituye un DAW desarrollado por la compañía alemana Ableton [29]. Se distingue de sus alternativas por su enfoque innovador en la actuación en vivo y su flujo de trabajo no lineal, lo que lo hace excepcionalmente popular entre productores de música electrónica, DJs y artistas que integran la producción y la actuación en sus prácticas.

A diferencia de los DAWs más tradicionales que se centran principalmente en una línea de tiempo lineal, Ableton Live ofrece dos vistas principales que facilitan tanto la composición detallada como la improvisación y el performance en tiempo real. Sus funcionalidades abarcan desde la grabación y edición de audio, la programación MIDI, la mezcla y masterización, hasta la creación y manipulación de bucles en tiempo real. Las vistas principales a destacar son la vista de sesión y la de arreglo [30].

La vista de sesión (ver Figura 4) es la característica más distintiva de Ableton Live y su principal diferenciador. Se presenta como una cuadrícula de pistas verticales y escenas horizontales. Cada columna representa una pista de audio o MIDI, donde se pueden cargar

instrumentos o efectos. Dentro de cada pista, se encuentran las celdas donde se alojan los bucles de audio o patrones MIDI. La principal fortaleza de la vista de sesión es la capacidad de lanzar clips individualmente o en grupos (escenas) en cualquier orden y en cualquier momento, permitiendo improvisar y experimentar.

Figura 4: Vista de sesión de Ableton Live.

La vista de arreglo (ver Figura 5) ofrece una visión lineal y cronológica del proyecto, similar a la de la mayoría de los DAWs. Es el espacio donde se ensambla la estructura final y secuencial de una canción. Los clips de audio y MIDI se organizan a lo largo de una línea de tiempo horizontal, permitiendo una edición precisa de su posición, duración y automatización.

Ableton Live representa una evolución significativa en el diseño de DAWs, ofreciendo una solución de producción musical que es tanto un estudio de grabación completo como un potente instrumento para la actuación en vivo. Su distintiva vista de sesión fomenta la creatividad a través de la experimentación no lineal y la improvisación, mientras que la vista de arreglo proporciona las herramientas necesarias para estructurar y refinar una composición final. La riqueza de sus instrumentos y efectos nativos, combinada con su flexibilidad para integrar herramientas de terceros, consolida a Ableton Live como una herramienta indispensable para un amplio espectro de artistas y productores contemporáneos.

Figura 5: Vista de arreglo de Ableton Live.

2.3.3. Reaper

Reaper es un entorno de producción de audio digital desarrollado por Cockos Inc [31]. Se distingue notablemente en el mercado por su modelo de licencia asequible, su naturaleza altamente personalizable, su eficiencia en el uso de recursos y su extrema flexibilidad. A diferencia de otros DAWs que a menudo se especializan en un flujo de trabajo o género particular, Reaper se presenta como un patio virtual, permitiendo a los usuarios configurar el software para adaptarse a casi cualquier necesidad de producción de audio, desde la grabación y edición de audio de alta fidelidad hasta la mezcla, masterización, diseño de sonido y producción MIDI.

Su filosofía de diseño se centra en la versatilidad y la extensibilidad, lo que lo convierte en una herramienta potente tanto para principiantes como para profesionales que buscan un DAW adaptable a sus métodos de trabajo específicos, sin las limitaciones impuestas por otros sistemas propietarios [32].

La interfaz de Reaper (ver Figura 6) es, por defecto, funcional y sin adornos, pero su verdadera potencia reside en su capacidad de personalización casi ilimitada. Los usuarios pueden modificar colores, diseños, menús, atajos de teclado y barras de herramientas para crear un entorno de trabajo que se adapte perfectamente a sus preferencias. Conformando la interfaz, encontramos múltiples componentes, comenzando por el menú principal y

herramientas, contenido en la parte superior de la interfaz, que proporcionan acceso rápido a funciones clave, herramientas de transporte y acciones definidas por el usuario. A su vez, contamos con el panel de control de pistas, situado a la izquierda de la ventana principal, que muestra las pistas del proyecto. La vista de arreglo representa la línea de tiempo principal donde los elementos de audio y MIDI se organizan, editan y mezclan para construir la composición completa. Por último, el panel de mezcla, similar a una consola de mezcla física, muestra los controladores de volumen, controles de imagen estéreo, envíos, retornos e inserciones de efectos para todas las pistas, proporcionando un control detallado sobre la mezcla.

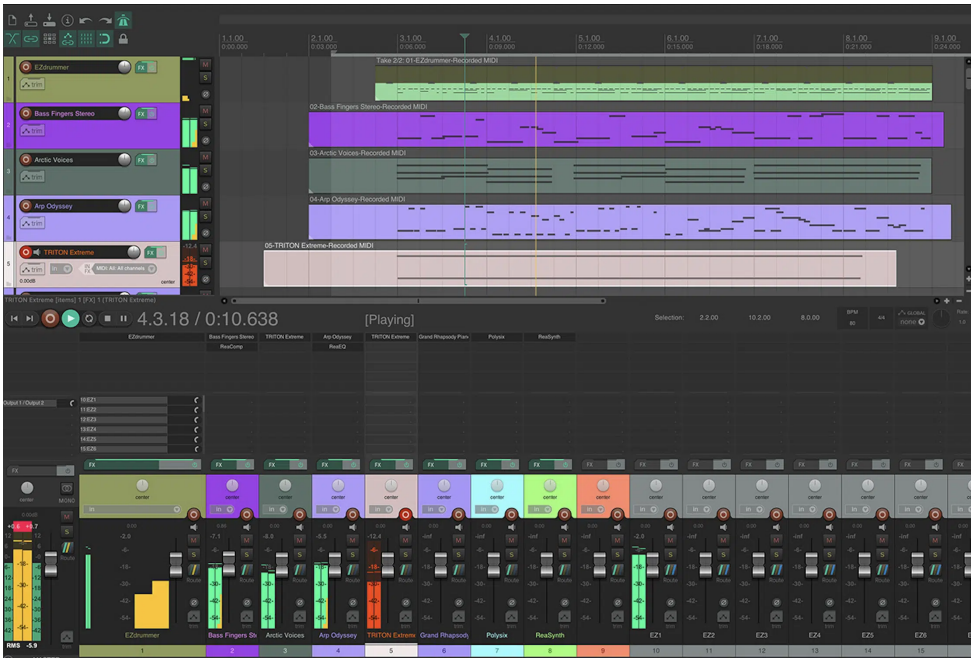


Figura 6: Vista personalizada de Reaper.

En Reaper, el concepto de pista es extraordinariamente flexible. Una sola pista puede manejar audio, MIDI, albergar instrumentos virtuales, actuar como bus de envío, o incluso como carpeta para organizar otras pistas. Esta versatilidad simplifica el flujo de trabajo y elimina la necesidad de gestionar diferentes tipos de pistas.

La vista de arreglo es el espacio principal para la composición lineal, donde los ítems de audio y MIDI se organizan y editan a lo largo de la línea de tiempo. Permite manejar herramientas de edición potentes que habilitan la manipulación de elementos de forma granular, incluyendo recorte, división, fundidos, estiramiento temporal y ajuste de tono.

El panel de mezcla proporciona una interfaz de consola de mezcla completa para controlar los niveles, la imagen dimensional y los efectos de todas las pistas. Una de las características más aclamadas de Reaper es su sistema de envío de audio planteado como "cualquier cosa a cualquier cosa". Esto permite enviar audio y MIDI entre cualquier pista o punto del proyecto, facilitando la creación de complejas configuraciones de bus, submezclas y cadenas de efectos.

Reaper se posiciona como un DAW excepcionalmente poderoso y flexible que, aunque puede tener una curva de aprendizaje inicial debido a su vasta personalización, recompensa al usuario con un control sin precedentes sobre su entorno de producción. Su modelo de licencia justo, su eficiencia operativa y su comunidad activa que contribuye de manera constante, lo convierten en una opción atractiva para aquellos que valoran la adaptabilidad y la capacidad de construir un flujo de trabajo a medida. Reaper no es solo un DAW; es una plataforma de audio que se adapta a las necesidades exactas de cada productor.

2.3.4. Logic Pro

Logic Pro es un DAW desarrollado por Apple Inc [33]. Exclusivo para el ecosistema macOS, se ha establecido como una herramienta robusta y completa para la producción musical profesional, la composición, la mezcla y la postproducción de audio. Su popularidad radica en una combinación de potentes herramientas de nivel profesional, una interfaz de usuario intuitiva que evoluciona con las tendencias de diseño de Apple, y una extensa biblioteca de sonidos e instrumentos de alta calidad incluidos de serie.

Logic Pro es ampliamente utilizado en estudios de grabación, por compositores de bandas sonoras, productores de música electrónica y artistas que buscan un flujo de trabajo fluido y una integración profunda con otros dispositivos y aplicaciones de Apple. Se distingue por ofrecer una solución todo en uno que abarca desde la grabación de audio multicanal hasta la masterización final, con un énfasis en la creatividad y la eficiencia.

La interfaz de Logic Pro (ver Figura 7) es conocida por su diseño limpio, moderno y funcional, que equilibra la complejidad de sus herramientas con una disposición clara. Se organiza principalmente alrededor de la ventana principal que integra múltiples paneles. El menú de control, en la parte superior, ofrece acceso a las funciones generales del software, controles de tiempo, información de tempo y clave, y botones personalizables para acceso

rápido a herramientas y vistas. También contamos con el área de pistas, el corazón de la composición lineal, aquí se organizan las pistas que se disponen a lo largo de una línea de tiempo horizontal. Finalmente, el mezclador, que se puede abrir en una ventana separada o acoplar en la parte inferior de la ventana principal, mostrando todos los canales del proyecto con sus controladores de volumen, controles de imagen estéreo, inserciones de efectos y envíos

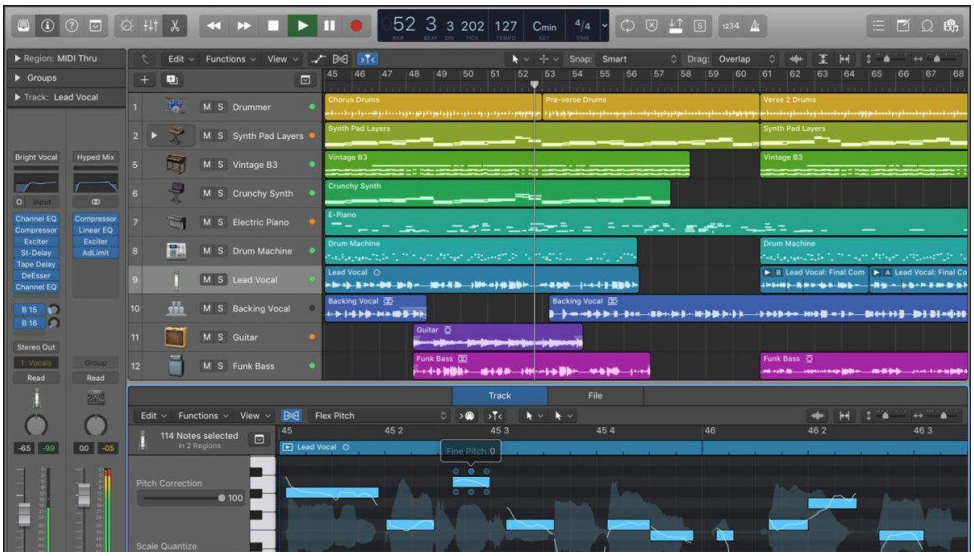


Figura 7: Vista de Logic Pro.

El área de pistas es donde se construye el arreglo musical de forma lineal. Logic Pro ofrece varios tipos de pistas para diferentes propósitos: pistas de audio, para grabar y editar audio en vivo o importar archivos de audio; pistas de instrumento, para cargar y reproducir instrumentos virtuales a través de MIDI; pistas auxiliares, para procesar audio de múltiples pistas a través de buses de envío; y los grupos y pistas de carpeta, útiles para organizar y controlar grandes proyectos de mezcla.

El mezclador de Logic Pro (ver Figura 8) es una consola virtual completa, que permite controlar cada aspecto de la mezcla. Cada pista cuenta con un canal correspondiente en el mezclador, con controles de volumen, imagen, solo, mute, envío/retorno y una ranura para hasta 15 inserciones de efectos. El sistema de envío/retorno de la señal permite enviar una parte de la señal de una pista a un bus auxiliar donde se aplica un efecto compartido, optimizando el uso de recursos. Esta vista también permite agrupar pistas para un control colectivo.

Figura 8: Mezclador de Logic Pro.

Logic Pro es una solución de producción musical formidable que ofrece un equilibrio excepcional entre funcionalidad profesional y una interfaz de usuario accesible. Su vasta colección de instrumentos y efectos de alta calidad y sus herramientas de edición intuitivas lo convierten en una opción atractiva para compositores, productores e ingenieros de audio que operan dentro del ecosistema de Apple. Logic Pro no solo permite la creación musical desde cero, sino que también proporciona las herramientas necesarias para llevar un proyecto a una calidad de producción final.

2.3.5. Avid Pro Tools

Pro Tools es un entorno de producción de audio digital desarrollado por Avid Technology [34]. Desde su origen, se ha establecido como el estándar de la industria para la grabación, edición, mezcla y masterización de audio profesional, así como para la postproducción de audio para cine y televisión. Su predominio en estudios de grabación de alto nivel, instalaciones de postproducción y emisoras se debe a su precisión en la edición, su robusto motor de audio, su escalabilidad con hardware dedicado y su fiabilidad en entornos de producción críticos.

Pro Tools es sinónimo de eficiencia y calidad en el manejo de grandes proyectos con numerosas pistas, ofreciendo un control granular sobre cada aspecto del audio. Si bien es utilizado en la producción musical de todos los géneros, su fortaleza histórica reside en la grabación multipista, la edición detallada y la mezcla para grandes producciones.

La interfaz de Pro Tools (ver Figura 9) es funcional y está diseñada para la eficiencia en flujos de trabajo profesionales. Se organiza principalmente alrededor de dos ventanas clave que se complementan mutuamente, junto con paneles de controles adicionales. El primero de estos paneles es la ventana de edición; constituye la ventana principal y lineal donde se organizan cronológicamente las pistas; da forma al espacio de trabajo central para la grabación, edición y arreglo del proyecto. Por otro lado, la ventana de mezcla, que representa una consola de mezcla virtual, muestra todos los canales de manera similar a un mezclador de hardware, crucial para la mezcla y el procesamiento del audio.

Figura 9: Vista de la interfaz de Pro Tools.

La ventana de edición es el espacio de trabajo donde se manipulan los fundamentos del proyecto: la grabación, la edición y el arreglo lineal. Pro Tools ofrece varios tipos de pistas optimizadas para diferentes tareas: pistas de audio, pistas MIDI y pistas auxiliares para submezclas o buses de audio. Los “clips” son los bloques fundamentales de audio o MIDI en las pistas editables para estructurar el proyecto completo.

La ventana de mezcla proporciona una vista completa de la consola de mezcla, crucial para balancear y procesar todos los elementos sonoros. Cada pista tiene su propio canal en el mezclador, con controladores de volumen, controles de imagen, botones de solo/mute, medidores de nivel y ranuras para inserciones de efectos y envíos.

Pro Tools se mantiene como la piedra angular en la producción de audio profesional, especialmente en estudios de grabación, postproducción de cine/TV y broadcast. Su énfasis en la precisión de la edición, la robustez de su motor de audio y su escalabilidad

a través de soluciones de hardware dedicadas lo convierten en la elección preferida para proyectos de alta envergadura y calidad crítica. Aunque su interfaz puede parecer menos "creativa" para la composición inicial en comparación con otros DAWs, su eficiencia en la edición, mezcla y masterización lo consolidan como una herramienta indispensable para los profesionales que buscan un control absoluto y la máxima fiabilidad en cada etapa del proceso de producción de audio.

3

Tecnologías Estudiadas

En este capítulo se presentan las tecnologías consideradas y estudiadas para el desarrollo de este proyecto. Se enumerarán las diferentes tecnologías junto con una breve introducción y el estudio o razonamiento que valora la necesidad de incluir o no cierta herramienta en el trabajo desarrollado.

3.1. C++

Se trata de un lenguaje de programación de propósito general desarrollado como una extensión del lenguaje C [35]. Su principal característica es que combina la programación estructurada con la programación orientada a objetos, lo que permite desarrollar software de manera más modular, reutilizable y escalable.

Es ampliamente utilizado en el desarrollo de sistemas operativos, videojuegos, aplicaciones de alto rendimiento, software embebido y más, gracias a su eficiencia, velocidad y control sobre los recursos del sistema. El lenguaje permite a los programadores manejar directamente la memoria, lo que lo hace eficaz a la vez que exigente en la gestión de errores. Además de las características del lenguaje C, C++ introduce conceptos como clases, herencia, polimorfismo, encapsulamiento y plantillas, lo que lo convierte en una herramienta flexible y robusta para desarrollar aplicaciones complejas.

La elección de C++ para el desarrollo de un DAW no es casual. Este lenguaje de programación de propósito general se ha consolidado como la columna vertebral de aplicaciones de alto rendimiento y baja latencia, características intrínsecas al procesamiento de audio en tiempo real. Este tipo de herramientas exigen una gestión eficiente de los

recursos computacionales y una respuesta instantánea a las interacciones del usuario, ámbitos donde C++ brilla por su capacidad para optimizar el rendimiento.

En el sector del software musical, desde sintetizadores y efectos hasta entornos complejos y motores de audio, C++ es la opción preferida por su poder y flexibilidad. La combinación con entornos como JUCE, específicamente diseñados para el desarrollo de aplicaciones de audio y sistemas multiplataforma, potencia aún más sus capacidades.

3.2. JUCE

JUCE (Jule's Utility Class Extensions) es un entorno de desarrollo en C++ ampliamente utilizado para crear aplicaciones multimedia, especialmente software de audio [36]. Se ha convertido en una herramienta clave en la industria del desarrollo de plugins de audio, estaciones de trabajo de audio digitales, sintetizadores y otras aplicaciones musicales.

JUCE proporciona una amplia colección de clases y herramientas que simplifican el desarrollo de interfaces gráficas, procesamiento de audio en tiempo real, manejo de archivos y muchas otras funcionalidades, todo desde un solo entorno multiplataforma. Esto significa que una aplicación escrita con JUCE puede compilarse fácilmente para sistemas Windows, macOS, Linux, iOS y Android sin tener que reescribir el código base.

Una de sus principales ventajas es su integración con estándares de audio como VST o AU, lo que permite a los desarrolladores crear plugins profesionales para una amplia gama de entornos musicales. JUCE es muy valorado por su eficiencia, flexibilidad y comunidad activa.

De cara al desarrollo de un proyecto masivo como puede ser una estación de trabajo de audio, JUCE se trata de un entorno avalado por la industria; es utilizado abiertamente por desarrolladores, como el equipo de Tracktion [37] y tiene un gran soporte a través de su comunidad; reflejo de esto es su relevancia en la Audio Developers Conference [38].

El uso de JUCE nos permite enfocarnos en los desafíos a alto nivel planteados en este proyecto en lugar de tener que lidiar con las complejidades nativas de cada sistema operativo o construyendo componentes de la interfaz desde cero. JUCE no solo acelera el desarrollo, sino que también eleva la calidad y la robustez del producto final.

3.3. Tracktion Engine

Tracktion Engine es una biblioteca de código abierto escrita en C++ que permite a los desarrolladores construir rápidamente aplicaciones de audio digital avanzadas [37], como estaciones de trabajo digitales, secuenciadores, editores de audio y más. Este motor fue creado por la misma empresa detrás de JUCE, y está diseñado para integrarse perfectamente con él.

A diferencia de JUCE, que es un entorno generalista para multimedia, Tracktion Engine se centra exclusivamente en la lógica de alto nivel para la creación y gestión de proyectos de audio. Proporciona herramientas para manejar pistas, clips, efectos, automatización, mezcla, reproducción y grabación, lo que ahorra años de desarrollo al construir una aplicación profesional desde cero.

Al estar basado en C++, Tracktion Engine ofrece un rendimiento óptimo para el procesamiento de audio en tiempo real y es compatible con múltiples plataformas. Gracias a su arquitectura modular y bien documentada, es ideal tanto para empresas como para desarrolladores independientes que buscan crear productos musicales innovadores.

Debido a las necesidades del proyecto y la curva de aprendizaje a la que nos enfrentamos a la hora de desarrollar el trabajo desde cero, se tomó la decisión de no utilizar Tracktion Engine. A largo plazo supone una adición muy valiosa para agilizar el desarrollo del proyecto, pero en el limitado tiempo de desarrollo se consideraba excesivo, puesto que implica aprender dos entornos completos en el ámbito del audio, sobre el que nunca antes habíamos trabajado.

3.4. Drivers de audio

Durante este subapartado estudiaremos sobre los principales controladores software de audio disponibles, en nuestro caso nos enfocaremos en el sistema operativo Windows, ya que será el entorno para el cual desarrollemos nuestro proyecto.

Nos referimos a estos controladores como APIs [39], ya que es la manera más común de referirse a interfaces de programación de aplicaciones. Son en esencia los puntos de entrada y salida que habilitan los sistemas hardware para comunicarse con ellos mediante el software.

3.4.1. Direct Sound

DirectSound [40] es una API de Microsoft que forma parte del conjunto DirectX [41], diseñada para facilitar la reproducción y captura de audio en sistemas Windows. Fue una de las primeras soluciones de audio de alto rendimiento disponibles para desarrolladores de juegos y aplicaciones multimedia en la plataforma Windows. Proporciona acceso de bajo nivel a las capacidades de reproducción de sonido del hardware, incluyendo efectos 3D, mezclado de múltiples fuentes de audio, posicionamiento espacial y control sobre búfers de sonido.

Sin embargo, con la llegada de Windows Vista, Microsoft dejó de actualizar activamente DirectSound, en favor de tecnologías más modernas como WASAPI [42]. A pesar de esto, DirectSound aún se utiliza en aplicaciones heredadas y sigue siendo compatible con versiones actuales de Windows para mantener la retrocompatibilidad.

3.4.2. WASAPI

WASAPI (Windows Audio Session API) es una interfaz de programación de aplicaciones desarrollada por Microsoft [42]. Forma parte del conjunto de APIs de audio de bajo nivel implementadas de forma nativa en Windows, y proporciona un acceso directo y eficiente al hardware de audio del sistema.

WASAPI permite a las aplicaciones reproducir y capturar audio con alta fidelidad, baja latencia y control preciso sobre las sesiones de audio. Ofrece dos modos principales de operación:

- Modo compartido: el audio de múltiples aplicaciones se mezcla automáticamente antes de ser enviado al hardware.
- Modo exclusivo: la aplicación toma control total del dispositivo de audio, evitando cualquier procesamiento adicional del sistema, lo que es ideal para tareas donde la latencia y la calidad del audio son críticas, como en producción musical o grabación profesional.

WASAPI se utiliza ampliamente en software profesional de audio, videojuegos, reproductores multimedia avanzados y cualquier aplicación que requiera un manejo preciso del

sonido en entornos Windows.

3.4.3. Asio

ASIO (Audio Stream Input/Output) [43] es una interfaz de audio desarrollada por Steinberg [44] que permite a las aplicaciones de audio comunicarse directamente con el hardware de sonido, evitando las capas de procesamiento del sistema operativo. Su principal objetivo es ofrecer baja latencia, alta fidelidad y sincronización precisa, lo que lo hace ideal para aplicaciones de audio profesional, como estaciones de trabajo digital, grabación en tiempo real y procesamiento de efectos.

A diferencia de APIs como WASAPI o DirectSound, que suelen pasar por el mezclador de audio del sistema operativo, ASIO opera en modo exclusivo, enviando y recibiendo audio directamente del dispositivo. Esto reduce significativamente la latencia y elimina cualquier procesamiento intermedio que pueda afectar la calidad del audio. ASIO es compatible principalmente con Windows, y requiere drivers específicos proporcionados por los fabricantes de interfaces de audio.

3.5. Formatos de plugin

A lo largo de este apartado estudiaremos diferentes formatos y estándares a la hora de exportar y trabajar con plugins como son moduladores de la señal o generadores de audio. Estos formatos son maneras de encapsular los programas software de forma que los entornos dedicados al audio puedan utilizarlos, así como Windows es capaz de interpretar y ejecutar las funcionalidades de un archivo ".exe", los controladores de audio están estandarizados para trabajar con ciertos formatos.

3.5.1. VST

VST (Virtual Studio Technology) [45] es un estándar de plugins de audio desarrollado por Steinberg [44] que permite integrar efectos de audio y sintetizadores virtuales en estaciones de trabajo de audio digital y otros entornos de producción musical. Introducido en 1996, VST revolucionó la industria del audio digital al permitir que software y plugins trabajen juntos de forma modular, como si fueran equipos físicos en un estudio

de grabación.

El estándar VST es compatible con múltiples plataformas, aunque su soporte principal ha sido en Windows y macOS. Su versión más actualizada es VST3 [46] , que ofrece mejoras significativas. Es por el avance tecnológico de la industria que Steinberg limita la licenciación necesaria para poder implementar el formato VST original y otras versiones anteriores en el software; de esta forma, incitan a los desarrolladores a usar VST3 como el nuevo estándar [47].

3.5.2. VST3

VST3 es la tercera y más reciente versión del estándar "Virtual Studio Technology" [46], desarrollado por Steinberg [44]. Fue lanzado en 2008 como una evolución tecnológica, con el objetivo de ofrecer mayor flexibilidad, eficiencia y capacidad para adaptarse a los flujos de trabajo modernos en la producción de audio.

Algunas de sus cualidades diferenciadoras son [48]:

- Procesamiento basado en eventos: el plugin solo procesa audio cuando hay datos presentes, lo que mejora el rendimiento y reduce el consumo de CPU.

- Entradas/salidas dinámicas: permite cambiar las configuraciones de canales de entrada y salida de forma flexible según las necesidades del host.

- Soporte avanzado para MIDI y controladores: en lugar de depender de un canal MIDI único, VST3 maneja eventos de control (como automatización y expresión) de manera más detallada y estructurada.

- Automatización de parámetros mejorada: VST3 permite una integración más precisa y rica de la automatización entre el plugin y el host.

Pese a que muchas de estas características no tienen por qué ser utilizadas específicamente en nuestro proyecto, es de valor conocerlas en caso de desarrollar funcionalidades relacionadas con estas cualidades. Conocer estas propiedades es vital para tomar decisiones sobre qué herramienta usar en cada situación.

3.5.3. AU

Audio Units (AU) [49] es el formato de plugin de audio desarrollado por Apple [50] para sus sistemas operativos macOS e iOS. Forma parte del entorno "Core Audio" [51], este es la base del sistema de audio en las plataformas de Apple. Al igual que VST, AU permite a los desarrolladores crear instrumentos virtuales, efectos de audio y procesadores de señal que pueden integrarse en estaciones de trabajo de audio digital como Logic Pro, GarageBand [52], u otras compatibles con el formato AU.

Aunque AU es exclusivo de las plataformas de Apple, es considerado un estándar esencial para cualquier desarrollador de plugins que desee distribuir productos en el ecosistema macOS/iOS. Muchas herramientas de desarrollo, como JUCE, permiten exportar plugins en formato AU junto con VST3 y otros, desde un único código base.

En nuestro caso, dado que desarrollaremos el proyecto orientado a Windows no haremos uso de esta tecnología. Es valioso conocerla y situarla en el campo del desarrollo de software relacionado con el audio, pero, de manera particular, no será utilizada en este trabajo.

3.6. Inno Setup

Inno Setup es una herramienta gratuita y de código abierto utilizada para la creación de instaladores en sistemas Windows [53]. Destacar la posibilidad de empaquetar múltiples archivos, crear accesos directos, registrar librerías, modificar el registro de Windows y definir tareas que se ejecutan antes o después de la instalación. Además, gracias a su sistema de escritura basado en Pascal Scripts [54], permite personalizar de forma avanzada el comportamiento del instalador, adaptándose a diferentes necesidades del despliegue.

En este proyecto se ha empleado para generar un instalador mediante un archivo personalizado que permite distribuir la aplicación de forma sencilla y profesional.

4

Metodologías

Durante este capítulo profundizaremos en las metodologías de trabajo empleadas para el desarrollo del proyecto. Se explicarán las bases sobre las que cimentamos nuestra forma de trabajar y concretaremos cómo se han hecho ciertas modificaciones para poder llevar a la práctica estos sistemas en el proyecto.

4.1. Metodologías ágiles

Las metodologías ágiles son un conjunto de enfoques para la gestión y desarrollo de proyectos, especialmente en el ámbito del software, que se centran en la flexibilidad, la colaboración constante y la entrega incremental de valor [55]. Surgieron como una alternativa a las metodologías tradicionales, que seguían modelos más rígidos y secuenciales, como el modelo en cascada.

El enfoque ágil se basa en la idea de dividir un proyecto en pequeñas partes funcionales, que pueden desarrollarse, probarse y entregarse de forma iterativa. Esto permite detectar errores, aplicar mejoras y adaptarse a cambios en los requisitos de manera más rápida y eficiente. La prioridad se centra en ofrecer un producto funcional lo antes posible, e ir mejorándolo progresivamente a lo largo del desarrollo.

El marco conceptual de las metodologías ágiles está recogido en el Manifiesto Ágil (Agile Manifesto) [56], publicado en 2001, que establece cuatro valores fundamentales:

- Individuos e interacciones sobre procesos y herramientas.
- Software funcionando sobre documentación extensiva.
- Colaboración con el cliente sobre negociación de contratos.
- Respuesta ante el cambio sobre seguir un plan.

Además, promueve principios como la comunicación continua entre los miembros del equipo, la entrega frecuente de versiones funcionales del producto y la capacidad de adaptación a nuevos requerimientos incluso en fases avanzadas del desarrollo.

Existen diversos marcos de trabajo que implementan estos principios ágiles, siendo Scrum [57], Kanban [58] y Extreme Programming [59] algunos de los más utilizados. Aunque cada uno tiene sus particularidades, todos comparten la misma filosofía: fomentar la eficiencia, la transparencia, y una mejora continua a lo largo del proyecto.

Las metodologías ágiles ofrecen una forma dinámica y adaptable de gestionar proyectos, centrada en la entrega de valor constante, la comunicación activa y la capacidad de adaptación frente a los cambios.

4.2. Scrum

Scrum es un marco de trabajo dentro de las metodologías ágiles, diseñado para gestionar y desarrollar productos de forma iterativa e incremental [57]. Se utiliza principalmente en proyectos de desarrollo de software.

La base de Scrum es dividir el trabajo en ciclos cortos y repetitivos llamados "sprints", que suelen durar entre una y cuatro semanas, facilitando así un seguimiento más claro del progreso y una respuesta rápida ante posibles cambios o mejoras. Al inicio de cada sprint se realiza una planificación para definir qué tareas se abordarán, y al finalizar, se hace una revisión para evaluar lo conseguido y una retrospectiva para identificar mejoras en el proceso.

Entre sus principios destacan la transparencia, la inspección continua del progreso y la capacidad de adaptación a cambios. Gracias a su estructura y filosofía, Scrum permite entregar valor de forma temprana y constante, mejorar la comunicación dentro del equipo y adaptarse rápidamente a nuevas necesidades o imprevistos.

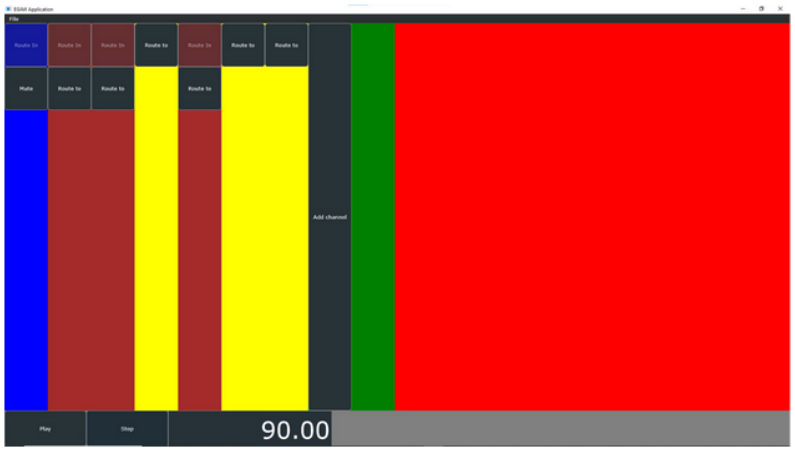
4.3. Metodologías seguidas en este proyecto

A lo largo del proyecto hemos seguido los principios de las metodologías ágiles haciendo una adaptación del sistema Scrum dado que en este caso se trata de un trabajo individual. La idea detrás de esta adaptación es conservar intacta la esencia de Scrum en

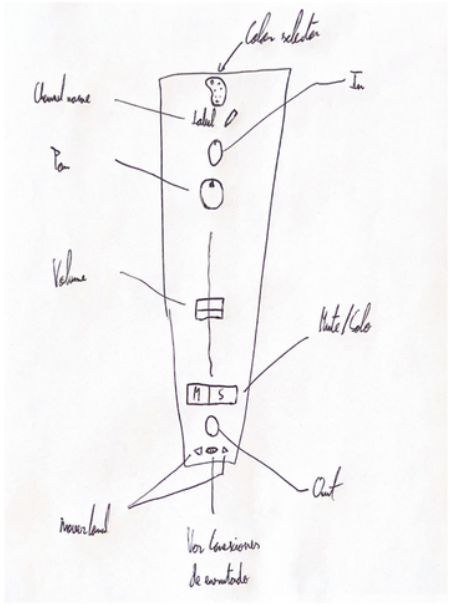
cuanto a organización temporal y flexibilidad ante el cambio, a la vez que se preservan funcionalidades como las ceremoniosas reuniones diarias, pero de manera individual.

26-02-2025

Se ha conseguido que la interfaz de contención actúe adecuadamente y ya comenzamos a hacer uso de un primer sistema de enrutado básico donde podemos enrutar entre canales. Hasta ahora la interfaz se ve así:



También se ha planteado un diseño inicial de cómo debe quedar la interfaz del canal



El color selector y el channel name es algo que se retrasará para dar prioridad a otros objetivos.

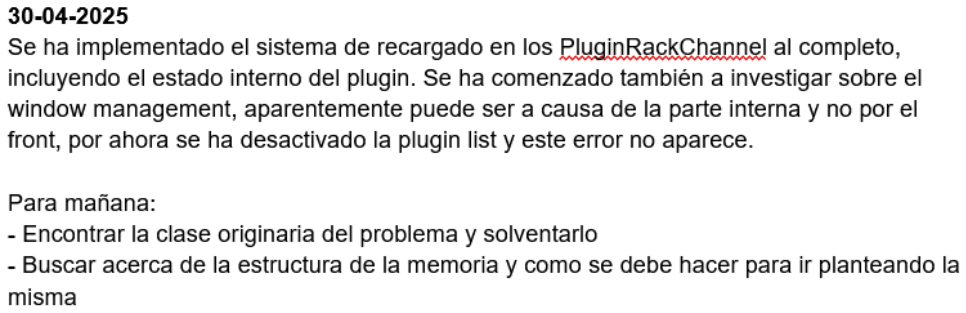
Figura 10: Diario informal de trabajo.

En este caso, el cliente que toma decisiones sobre qué se necesita y qué priorizar era yo mismo, por lo que pude adaptar los cambios de enfoque o decisiones de diseño a lo largo del desarrollo con fluidez. El desarrollo ha seguido un formato de sprints de distintas

longitudes de acuerdo a los hitos a obtener en cada periodo; en este caso, se establecían hitos realistas y alcanzables para plantear sprints que durasen entre una y tres semanas, variando según la magnitud del logro.

Durante el desarrollo de cada sprint contaba con un diario de trabajo utilizado para suplir la ausencia de reuniones diarias (ver figuras 10 y 11). En lugar de celebrar las reuniones que plantea Scrum, en este diario anotaba al final del día los avances, bloqueos, ideas y tareas para hacer; de esta forma no solo se documenta el avance del proyecto, sino que también, psicológicamente, permite ser consciente sobre el trabajo realizado y mantener la organización sobre las tareas a realizar a muy corto plazo. Este sistema facilita la constancia al tratarse de un proyecto individual y facilita la percepción sobre el trabajo realizado y a realizar.

La planificación, la constancia y la adaptabilidad en la priorización de tareas son motivos clave del desarrollo realizado. Pese a las posibles dificultades de mantener el ritmo siendo un proyecto individual, el sistema planteado con metas determinadas para ciertas fechas facilita el continuo desarrollo del trabajo y la perseverancia.


30-04-2025
Se ha implementado el sistema de recargado en los PluginRackChannel al completo, incluyendo el estado interno del plugin. Se ha comenzado también a investigar sobre el window management, aparentemente puede ser a causa de la parte interna y no por el front, por ahora se ha desactivado la plugin list y este error no aparece.

Para mañana:
- Encontrar la clase originaria del problema y solventarlo
- Buscar acerca de la estructura de la memoria y como se debe hacer para ir planteando la misma

Figura 11: Diario informal de trabajo.

5

Objetivos y Requisitos

A lo largo de este capítulo presentaremos el proyecto que propone una solución para las problemáticas expuestas y estableceremos de manera clara los objetivos fijados para el proyecto. Además, recogeremos algunos de los requisitos del proyecto clasificando entre requisitos funcionales y no funcionales.

5.1. Introducción al entorno planteado e ideas principales

El proyecto se presenta como un entorno virtual en el que poder trabajar el audio, enfocándose en tareas como la mezcla. En particular, se trata de construir un sistema completo que promueva y facilite a los usuarios trabajar escuchando de manera activa.

La escucha activa es fundamental para un trabajo adecuado del audio, y es este principio el que vertebra todo el desarrollo. A partir de este, construimos todas las ideas de diseño que conforman el producto. Algunos de los conceptos que forman el núcleo principal de ideas de desarrollo son:

- La percepción auditiva sobre la vista, es decir, toda aquella interacción que se pueda realizar con el sistema mediante el oído prevalecerá sobre su implementación visual.
- Proporcionar un espacio libre de la sobrestimulación visual. Este principio busca brindar simplicidad a la interfaz de usuario, reduciendo siempre que sea posible la información visible en pantalla.
- Adaptabilidad a los requisitos técnicos del usuario. Esto significa que el técnico tendrá la posibilidad de consultar valores paramétricos si su flujo de trabajo lo requiere,

aunque alineándose con los anteriores, el sistema no fomentará la parametrización de manera inherente.

Estos son algunos de los principios ideológicos que cimientan las bases del diseño del producto y, alrededor de ellos, se construyen todas las ideas particulares que afectan a cada uno de los componentes de nuestra interfaz.

5.2. Objetivos propuestos

Desde un inicio se dejan claros ciertos objetivos tangibles para el proyecto, no solo a nivel de diseño como se propone anteriormente, sino también en cuanto a funcionalidades y arquitectura del software. Estos objetivos son realistas y, pese a las posibles complicaciones, son alcanzables en el tiempo dado para el desarrollo del trabajo.

Destacar que desde un primer momento, a la hora de establecer los objetivos del proyecto, se ha tenido en cuenta la complejidad y magnitud que conlleva hacer una pieza de software así. Una de las charlas más enriquecedoras a la hora de entender la monstruosidad de proyectos similares es la de David Rowland, CTO de Audio Squadron [60], en la *"Audio Developers Conference"* de 2023 [61]. Esta charla de tan solo tres cuartos de hora expone las expectativas sobre lo que debe poder hacer una estación de trabajo de audio digital, sus costes y cómo se tienden a implementar algunas de estas funcionalidades. En unos minutos desglosa el coste económico y temporal que conlleva crear una herramienta así; centrándonos en la parte temporal que nos concierne, concluye que una herramienta multiusos como se entiende hoy día supone, para ser desarrollada por una sola persona, más de diez años de trabajo. Nuestro proyecto, al centrarse en un área en concreto de la producción musical, podrá prescindir de algunas de estas funcionalidades, pero la magnitud del mismo no deja de ser algo destacable y a tener en cuenta.

Una vez se plantea la complejidad y el tamaño del proyecto, se decide qué arquitectura seguirá. En nuestro caso, al tratarse de una aplicación de escritorio, no encontramos las posibles dificultades de plantear un proyecto web y toda la infraestructura que este puede acarrear. A la hora de orquestar la estructura de nuestro software, seguiremos el patrón modelo, vista y controlador [62], aunque con ciertas modificaciones. En nuestro caso, la vista y los controladores se funden en un mismo plano. Debido a la estructura

de componentes planteada por JUCE [63] y cómo se manejan los hilos en el sistema, los elementos visuales también se diseñarán para manejar la interacción con el usuario; ejemplos nativos de la librería son el botón [64] o el deslizador [65], entre otros. Por su parte, el modelo se tratará de una implementación de la clase "ValueTree" [66], diseñada para ser la estructura de datos principal de la librería, siendo esta consistente y rápida en sus accesos. El modelo no solo cubrirá en nuestro enfoque las estructuras de datos, sino que también consideramos como parte del modelo la estructura interna necesaria para manejar el audio mediante su hebra correspondiente. Todo el trabajo técnico mencionado quedará explicado a lo largo de este capítulo.

Finalmente marcamos varios objetivos tangibles, desde hitos a más alto nivel hasta metas concretas sobre implementaciones. A alto nivel conceptual, los objetivos que buscamos alcanzar en nuestro software son:

- Un sistema perenne en la interfaz que permita controlar la reproducción del audio y movernos a través de las diferentes vistas
- El manejo de proyectos permitiendo la carga y el guardado de los mismos, además de su exportación a ficheros de audio.
- Contar con una pantalla dedicada exclusivamente a la audición donde el usuario pueda poner en práctica la escucha activa sin distracciones visuales.
- Llevar a cabo un sistema de posicionamiento temporal donde poder manejar los archivos de audio y colocarlos en el tiempo.
- Conseguir una abstracción de una mesa de mezcla analógica con canales funcionales y especializados en diferentes funciones.

5.3. Requisitos

A lo largo del desarrollo para implementar los hitos establecidos se profundizará capa a capa hasta llegar a pequeños objetivos muy concretos. Puesto que seguimos metodologías ágiles, no planteamos toda la estructura de requisitos desde un principio, sino que moldeamos unos primeros requisitos y, conforme la implementación avanza, desarrollamos

requisitos de mayor concreción. Todo ello queda reflejado mediante historias de usuario que muestran desde los requisitos que buscamos como usuarios de la aplicación hasta algunos requisitos técnicos que facilitarán nuestra labor como desarrolladores.

Podemos dividir los requisitos como funcionales, los cuales especifican qué debe hacer el sistema, o no funcionales, que definen cómo debe conformarse el sistema.

Algunas de las historias de usuario que capturan requisitos funcionales son las siguientes:

Sistema de enrutamiento
Como usuario **Quiero** disponer de un sistema de enrutamiento de entrada y salida en mis canales **Para** poder controlar el flujo de mis señales de audio y tener un seguimiento del procesamiento que estas siguen.

Manejador de señales
Como desarrollador software **Quiero** establecer manejadores de señales centralizados y únicos **Para** poder interconectar acciones entre componentes sin generar dependencias entre los mismos.

Metodo para silenciar canales
Como usuario **Quiero** poder silenciar un canal **Para** decidir si escuchar o no la salida sin tener que afectar a las configuraciones del canal.

Ajuste de longitud
Como usuario **Quiero** controlar la longitud de los bloques de audio **Para** eliminar fragmentos indeseados del inicio o final de los archivos de audio.

Para algunos de los requisitos no funcionales encontramos las siguientes historias de usuario:

Coherencia estética

Como usuario

Quiero que los botones y controladores dedicados a una misma función mantengan la misma linea estética

Para identificar de manera intuitiva las funciones de estos controladores.

Escalabilidad del sistema

Como desarrollador software

Quiero que los componentes estén desacoplados y sean perfectamente modulares

Para asegurar la escalabilidad del sistema.

Estándar de guardado

Como desarrollador

Quiero establecer un formato estandarizado para los archivos de guardado

Para mantener la cohesión entre este tipo de ficheros.

6

Desarrollo del proyecto

En este capítulo trataremos en profundidad la solución desarrollada. Nos sumergimos en los sistemas técnicos clave para la solución, recorreremos todas las interfaces integradas en el producto, trataremos las herramientas externas implementadas en forma de "plugin", veremos el funcionamiento del instalador y, por último, las pruebas de calidad relacionadas con el desarrollo realizado.

6.1. Sistemas clave del producto desarrollado

En esta parte del capítulo se expondrán con claridad ciertos sistemas clave para el desarrollo del proyecto. En este subapartado estudiaremos los mecanismos que vertebran el funcionamiento interno de la aplicación software. Destacar que muchos de los conceptos aquí expuestos se complementan con el resto de subapartados, y su comprensión se facilita a lo largo de la lectura completa.

A la hora de comprender el funcionamiento del audio y cómo hace el entorno JUCE para trabajar los diferentes mecanismos planteados, es necesario comprender algunos conceptos que cimentan cómo funciona el sistema. Dentro de este entorno encontramos dos hilos principales, el hilo dedicado a los mensajes e interfaz (*"MessageThread"*) y el hilo dedicado al procesamiento de audio en tiempo real (*"AudioThread"*) [67]. El entorno está planteado para que estos hilos corran de manera independiente y se puedan ejecutar funciones preestablecidas por el sistema desde cada hilo. Por último, añadir que no es necesario ceñirse únicamente a estos dos elementos; podremos crear hilos a nuestro antojo en el mismo marco de trabajo.

6.1.1. Sistema de manejo de audio y enrutado de la señal

El sistema construido para manejar la señal de audio confecciona un entramado complejo diseñado de acuerdo a las bases de JUCE. Este es elemental para el proyecto, puesto que si no realizamos un procesamiento del audio adecuado, el resto del software carece de sentido.

La salida del audio en tiempo real recae sobre el ya mencionado *"AudioThread"*, este hilo se lanzará desde nuestro componente principal que extiende la clase *AudioAppComponent* (ver figura 12) [68]. Esta clase cuenta con tres métodos esenciales de implementar por cualquier componente que vaya a manejar algún procesamiento de audio (*AudioSource* [69]), estos métodos son:

- `prepareToPlay(int samplesPerBlockExpected, double sampleRate)`: esta funcionalidad se especifica para que todo metodo que cuente con un procesamiento de audio, cuente a su vez con un método que prepare el audio para su procesamiento. Esta función siempre se ejecutará de forma previa a la reproducción del audio. Tomamos por parámetros los valores *"samplesPerBlockExpected"* y *"sampleRate"*; el primero de ellos también se puede conocer como *"bufferSize"* y maneja el número de muestras que necesita el sistema para interpretar una señal de audio [70]. En segundo lugar, el *"sampleRate"*, que es el número de muestras que recoge el sistema por segundo real [71].

- `releaseResources()`: esta función se acciona una vez se destruye o desconecta el componente encargado de procesar audio simplemente libera los recursos necesarios.

- `getNextAudioBlock(const AudioSourceChannelInfo& bufferToFill)`: este es el método principal entorno al cual el resto de funcionalidades están diseñadas. Esta función se encarga de tomar el búfer de audio que está manejando la aplicación y procesarlo según se indique.

Nuestros sistemas no solo heredan de la clase interna del entorno *AudioSource* [69] ya mencionada, sino que lo hacen de *PositionableAudioSource* [72] que nos da los medios necesarios para poder manejar, además, la posición desde la cual se reproduce el audio.

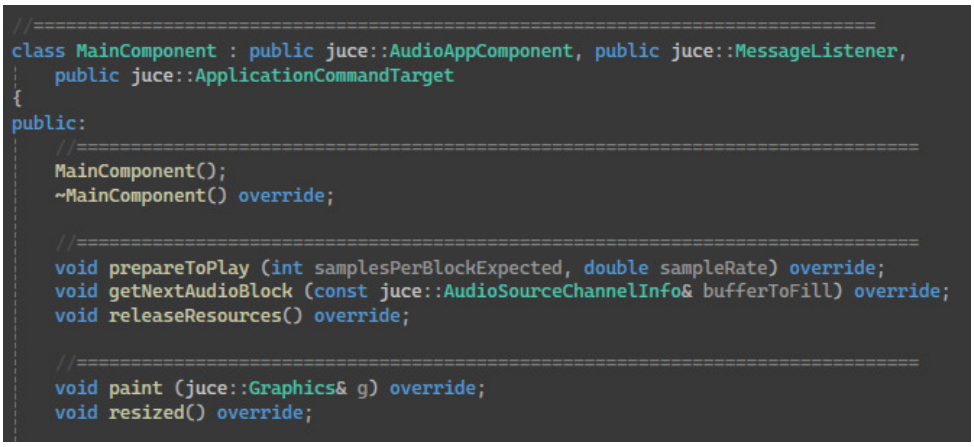


Figura 12: Pequeña muestra de código básico del componente principal.

El esqueleto encargado de manejar el audio se compone de múltiples elementos interconectados, partiendo de nuestro *AudioSystemBus* como eje principal. A partir de esta clase principal que maneja todos los eventos relacionados con el audio y los transmite de forma directa al componente principal de nuestra aplicación, encontramos diferentes elementos que desarrollamos para enriquecer el sistema. Estas clases pueden visualizarse, en formato simplificado, en el diagrama de clases propuesto (ver figura 13).

Dentro de este entramado de clases podemos diferenciar dos rasgos; por un lado, las clases y relaciones elementales para el funcionamiento básico del audio, y de manera aditiva, las clases componentes. Estas clases triviales para el funcionamiento del audio son el *MainComponent* ya mencionado, el *AudioSystemBus* en el que profundizaremos y la existencia de un *MixBusChannel* particular llamado *masterBusChannel*. El resto de sistemas se complementan y estructuran toda la arquitectura para el manejo del audio según las necesidades del usuario final.

La clase *AudioSystemBus* constituye el manejador principal que orquesta todo el trabajo del audio. Almacena los componentes dispuestos al usuario y los interconecta según sea necesario. Esto lo hace almacenando en listas todos los componentes desplegados por el usuario, encargándose el *AudioSystemBus* de manejar su creación, eliminación y enrutamiento. Cuenta además con un componente *MixBusChannel* permanente; este canal lo utilizará como salida, es decir, nuestro sistema principal delega el procesamiento del audio en el llamado *masterBusChannel*, y todo canal que quiera enviar su señal hacia el exterior tendrá que llegar de una forma u otra hasta el *masterBusChannel*. Este sistema emula el

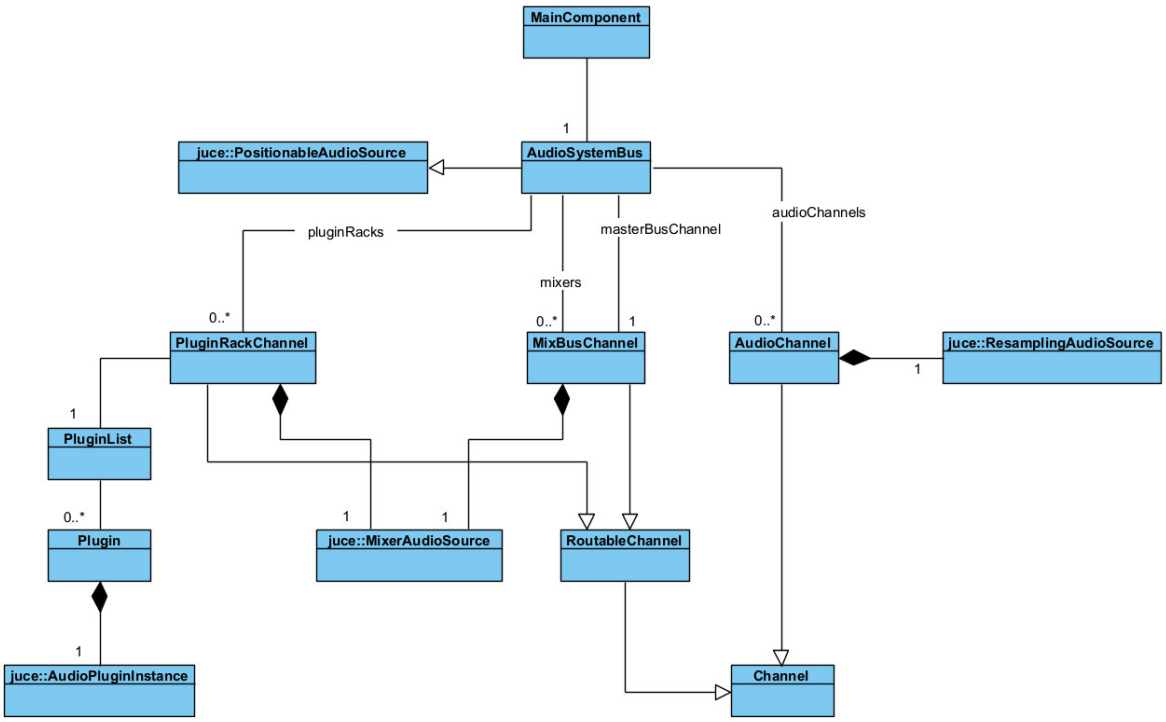


Figura 13: Diagrama de clases simplificado del sistema de audio implementado.

funcionamiento de las mesas de mezcla reales donde encontramos un bus principal al que llega la señal que queremos unificar como final.

A continuación, profundizaremos en cómo se estructuran las clases relacionadas directamente con los componentes que invoca el usuario en función de sus necesidades. Nuestras clases *PluginRackChannel*, *MixBusChannel* y *AudioChannel* se fundamentan en el patrón "Decorator" [73] para simplificar el trabajo del audio. Esto significa que, en esencia, *PluginRackChannel* y *MixBusChannel* se tratan de un gran decorador que adapta a nuestro sistema el funcionamiento base de la clase original *AudioChannel*, utilizada para recoger múltiples señales de audio y unificarlas en una sola. A su vez, *AudioChannel* se construye para envolver la clase *ResamplingAudioSource* [74]. Todo esto teniendo en cuenta que se trabaja una estructura interna dirigida por *Channel* y *RoutableChannel* que crean una cohesión en ciertos datos que han de manejar todos los canales y cómo se clasifican (revisar la figura 13 para facilitar la comprensión). Viendo cada una de estas clases en profundidad, encontramos:

- *Channel*: representa la clase base de la cual hereda todo componente del tipo *channel*, esta clase aporta los métodos necesarios para la gestión del identificador único del canal, el manejo conceptual de la posición numerica en la que se encuentra dentro del mezclador y además almacenar los identificadores de los canales a los que se está enrutando nuestro canal.

- *RoutableChannel*: es una extensión conceptual de la clase anterior que además nos permite identificar que el canal puede recibir señal de algún otro.

- *AudioChannel*: esta clase representa un canal de audio. Refleja un sistema en el cual introducimos una señal de audio para su procesamiento. Es vital para el desarrollo del trabajo puesto que simboliza nuestra fuente de audio. Un *AudioChannel* nos permite cargar un archivo de audio, modificar su longitud y reproducirlo adecuadamente.

- *MixBusChannel*: es un sistema ideado para unificar señales de audio. Un bus de mezcla sirve esencialmente para recoger señales de audio y unificarlas un una sola salida. Este proceso se hace delegando en la clase nativa de JUCE *MixerAudioSource*.

- *PluginRackChannel*: este canal es una herramienta angular para el trabajo del audio en procesos como la mezcla; nos permite recoger una señal de entrada y procesarla a través de los efectos que desee añadir el usuario. Esto se implementa haciendo uso de la *PluginList* a la cual delegamos el procesamiento de la señal, a su vez, esta lista únicamente se encarga de procesar iterativamente la señal por los "plugins" creados en el orden establecido por el usuario. Nuestra clase *Plugin* carga y maneja internamente el "plugin" apoyándose en la clase nativa *AudioPluginInstance* haciendo el procesamiento real de la señal y no solo un redireccionamiento estructural.

La complejidad estructural que supone un sistema de audio es acorde a la amplia variedad de herramientas que presentamos al usuario. Es nuestro deber como desarrolladores estructurarlo adecuadamente para asegurar la interoperabilidad de los componentes y su escalabilidad.

6.1.2. El uso de la estructura de datos "ValueTree"

El sistema software necesita una estructuración de los datos para mantener la cohesión durante el trabajo. Nuestra estructura de datos elegida será el *"ValueTree"* [75] nativo del entorno JUCE; esta estructura almacenará la información necesaria para comunicar los sistemas de audio, la interfaz y el resto de elementos que componen el producto.

Esta herramienta proporcionada por la librería escogida es considerada por sus creadores como la manera ideal de trabajar los datos dentro del entorno JUCE [76]. Es una estructura basada en los conocidos árboles [77], que es capaz de almacenar nodos y acceder a ellos, adjuntar nodos hijo o atribuir propiedades a los propios nodos; la flexibilidad en la forma de trabajar y la cantidad de operaciones disponibles facilita el uso de esta estructura.

La manera de desplazarse por la estructura y guardar referencias a la misma es muy cómoda. Lo esencial es guardar una referencia al nodo raíz para poder manejar siempre el árbol completo, aunque también se podría recuperar desde cualquier nodo hijo. A partir de ahí, podemos acceder a cualquier hijo que esté conectado a nuestra posición mediante su identificador. Un nodo no puede construirse sin identificador, en tal caso, el entorno genera un nodo invalidado.

Además, para cada nodo podemos establecer propiedades. Estas variables son valores genéricos que almacena la estructura de datos; pueden ser tanto numéricos como booleanos o cadenas de caracteres. Para agregar un nodo al árbol principal hemos de construirlo independientemente y, posteriormente, asociarlo como hijo de otro nodo; esto ya nos adelanta que podremos establecer estructuras estándar asociadas a un componente y agregarlas de manera ordenada a nuestro árbol [78].

Un patrón de comportamiento esencial para potenciar el uso de esta estructura de datos es el patrón del observador [79], también conocido dentro de JUCE como "listeners" u oyentes [80]. Este patrón establece una manera en la que se diseña el comportamiento de los componentes frente a los cambios. La idea principal es que existe un actuador y uno o varios oyentes; estos oyentes funcionan bajo demanda del actuador, es decir, cuando este componente realiza un cambio, notifica a los oyentes y estos reaccionarán de acuerdo con el comportamiento programado. Todos los componentes pueden ser completamente

independientes, simplemente necesitamos un mecanismo a través del cual vinculemos al oyente con el actuador. A su vez, el comportamiento de respuesta de los oyentes es único; cada oyente puede realizar una función diferente frente al mismo evento.

Implementamos este patrón haciendo que los componentes relacionados con los parámetros y diferentes elementos estructurales sean oyentes de los nodos que nos interesen. Un ejemplo muy sencillo de ello es el funcionamiento del botón de zoom aplicado a la lista de reproducción. Cuando accionamos este botón, el componente cambia el valor existente en nuestra estructura de datos; la estructura de datos se encarga de notificar a sus oyentes, este componente recoge la notificación como oyente y, finalmente, actúa rescalando la interfaz con la cantidad de zoom deseada (ver figura 14).

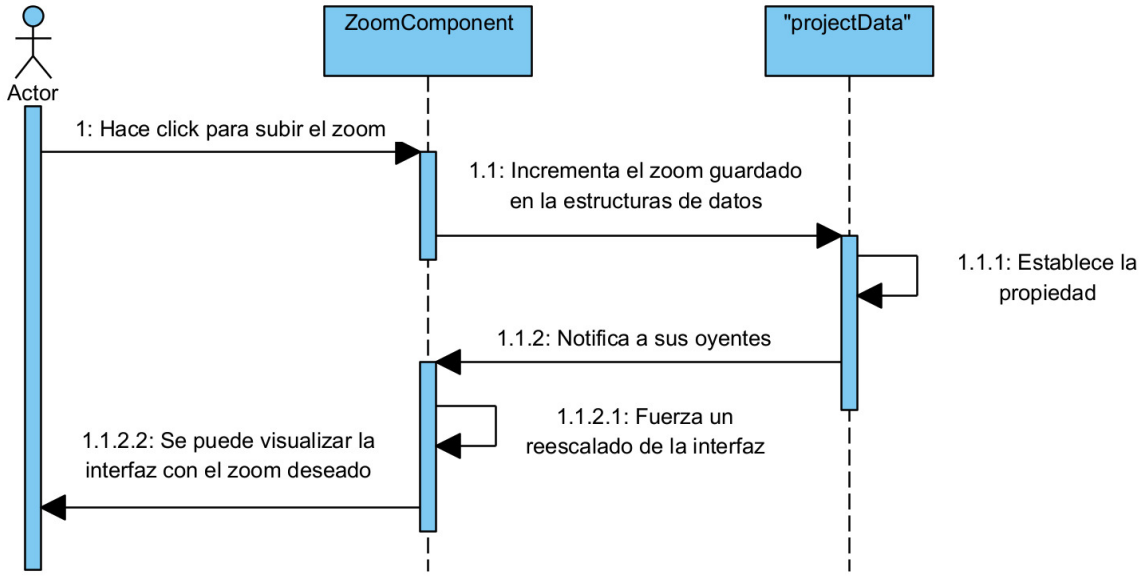


Figura 14: Sistema de oyentes aplicado sobre el componente de zoom.

En nuestra implementación de la estructura de datos, construimos un árbol principal que llamaremos “projectData”, este será el modelo que almacene el estado del proyecto en el que estamos trabajando. La forma básica de “projectData” contiene el nodo principal del proyecto, un nodo encargado de mantener el orden en el que se disponen los canales y el nodo asociado al canal máster (ver figura 15). Para visualizar los estados y estructuras existentes en nuestro modelo de datos, lo representaremos mediante XML, como se realiza de manera nativa para exportar archivos de guardado.

Para conservar la cohesión en nuestra estructura de datos, respetamos este patrón de

```
<AudioSystemBus View="MIXER" Zoom="10.0" bpm="90.00" saveFilePath="C:\Users\admin\Desktop\EGAW\BasicSave.egaw"
                projectName="BasicSave">
  <channelOrder>
    <6addfbbecc1a77f50000000000000000/>
  </channelOrder>
  <6addfbbecc1a77f50000000000000000 Type="MixBusChannel" Gain="0.0" Pan="0.0" Mute="0" Solo="0" Colour=""
                                    Name="Master Bus" mixerPosition="0">
    <channelsRoutedTo/>
  </6addfbbecc1a77f50000000000000000>
</AudioSystemBus>
```

Figura 15: Estructura del árbol de datos básico de nuestro sistema.

construcción; todo canal tendrá su propio nodo hijo de la raíz representado mediante el identificador único y, además, se añadirá un nodo hijo de "channelOrder" que identifique este canal. Debido a la base establecida por la clase *Channel* explicada en el subapartado anterior, todo nodo asociado a un canal tendrá como identificador el identificador único del canal y contará con un hijo llamado "channelsRoutedTo"; el funcionamiento de este hijo es similar a "channelOrder", almacenamos como hijos de "channelsRoutedTo" los identificadores de los canales hacia los que enviamos la señal.

Los canales que siguen esta estructura al pie de la letra son el canal de mezcla y el canal de audio (ver figura 16). Todos los canales quedan clasificados por el parámetro "Type", de forma que podamos distinguir rápidamente el tipo de elemento que es. El canal de mezcla, siendo el más simple de ellos, almacena parámetros esenciales para la edición del audio permitida por este sistema. El canal de audio añade a su estructura ciertos datos elementales para manejar los archivos de audio y su posición en el tiempo. Este sistema guarda la ubicación del archivo, la posición en la que debe iniciar su reproducción dentro del proyecto junto con los valores relacionados con la duración del archivo de audio y posibles modificaciones de inicio y fin realizadas por el usuario. Estas funcionalidades mencionadas en la estructura de datos quedarán explicadas en profundidad en apartados posteriores del capítulo.

La estructura más compleja en nuestro árbol la compone el canal de "plugins". Este elemento mantiene los elementos básicos asociados a cualquier canal; además, cuenta con un hijo dedicado a guardar los parámetros relacionados con los "plugins". Este hijo, a su vez, tiene asociado un elemento que guarda el orden en el que se trabajan los "plugins" en la cadena de procesamiento y, adicionalmente, almacena un nodo hijo para cada hueco preestablecido para un "plugin". Este nodo asociado al hueco del "plugin" se

```
<AudioSystemBus View="MIXER" Zoom="10.0" bpm="90.00" saveFilePath="C:\TFG\RouteTestBASIC.egaw"
                projectName="RouteTestBASIC">
  <channelOrder>
    <6addfbbecc1a77f50000000000000000/>
    <1b87229bf15c438984879489779f8f9f/>
    <dde0779304b34cb5b2548944eefc0027/>
  </channelOrder>
  <6addfbbecc1a77f50000000000000000 Type="MixBusChannel" Gain="0.0" Pan="0.0" Mute="0" Solo="0" Colour=""
                                    Name="Master Bus" mixerPosition="0">
    <channelsRoutedTo/>
  </6addfbbecc1a77f50000000000000000>
  <1b87229bf15c438984879489779f8f9f Type="MixBusChannel" Gain="0.0" Pan="0.0" Mute="0" Solo="0" Colour="ff3c5c4c"
                                    mixerPosition="1" Name="Mix Bus 1">
    <channelsRoutedTo>
      <6addfbbecc1a77f50000000000000000/>
    </channelsRoutedTo>
  </1b87229bf15c438984879489779f8f9f>
  <dde0779304b34cb5b2548944eefc0027 Type="AudioChannel" Gain="0.0" Pan="0.0" Mute="0" Solo="0" StartTime="0.0"
                                    Colour="ff787878" FilePath="C:\TFG\Hip Hop Classic Drum Break 84 BPM [raf87geuIgk].mp3"
                                    Name="Hip Hop Classic Drum Break 84 BPM [raf87geuIgk]" AudioFileStart="0.0"
                                    AudioFileEnd="11.52" AudioFileLength="11.52" mixerPosition="2">
    <channelsRoutedTo>
      <6addfbbecc1a77f50000000000000000/>
      <1b87229bf15c438984879489779f8f9f/>
    </channelsRoutedTo>
  </dde0779304b34cb5b2548944eefc0027>
</AudioSystemBus>
```

Figura 16: Estructura del árbol de datos con un canal de mezcla y uno de audio en nuestro sistema.

preocupa primeramente sobre si el hueco está ocupado por una instancia y, en caso de estarlo, almacena su descripción y su estado. Ambos valores se obtienen directamente de la instancia del "plugin" y nos proporcionan la información necesaria para instanciar en cualquier momento el programa software y el estado de sus parámetros, respectivamente (ver figura 17).

6.1.3. Funcionamiento de los manejadores de señales unificados

En la implementación de la herramienta software, una pieza clave para facilitar el desarrollo y la comunicación entre los componentes ha sido el uso de manejadores de señales. En esencia, estos manejadores son construcciones desarrolladas siguiendo el patrón "Singleton" [81]. Esta estructura establece que solo pueda existir en ejecución una única instancia de cada manejador; de esta forma, podemos solicitar la instancia desde cualquier pieza de código y nos aseguramos de que sea un manejador transversal único.

De igual manera que con los árboles preestablecidos por JUCE, aplicaremos el sistema de oyentes planteado en el patrón observador [79]. Los manejadores de señales almacenan los oyentes que han de notificar; adicionalmente, cuentan con una cola de señales para asegurar que el orden de ejecución no se ve interrumpido por algún otro componente. Por ejemplo, en caso de que una señal dispare un proceso que requiera enviar una segunda

```
<5c1b19da34ba437bb6a9b2c569848c5f Type="PluginRackChannel" Mute="0" Solo="0" Colour="ff8b008b"
                                  Name="Plugin Rack 1" mixerPosition="1">
  <plugins>
    <pluginOrder>
      <4fad55d35134477f907e2ba01aec1c74/>
      <62764e9d64e745fda0a17cf7b7751ee5/>
      <7a4875cbffc446d495f0863ac1f2599c/>
      <bce49e747ee248cd8623009da53a4ea1/>
      <48bcbac3f3be49a0a5eb855390103c3d/>
    </pluginOrder>
    <48bcbac3f3be49a0a5eb855390103c3d isOccupied="1" Name="Plugin" isBypassed="0" pluginPosition="5">
      <pluginDescription descriptionXML="&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;
      <pluginStateData stateData="835.VMjLgny....O+fWarAhckI2bo8la8HRLt.iHfTlai8FYo41Y8HRUTYTK3HxO9.BOVMEl
:SdPM2bZsRatbEU4o0UC4hKt3hKt3hKt3hKt3hKt3hYt3hKt3hKt3hKt3hKt3hKt3hKt3hKt3hK77RREQVZzMzatQmbuwFakImO77hl
    </48bcbac3f3be49a0a5eb855390103c3d>
    <4fad55d35134477f907e2ba01aec1c74 isOccupied="0" Name="Plugin" isBypassed="0" pluginPosition="1"/>
    <62764e9d64e745fda0a17cf7b7751ee5 isOccupied="1" Name="Plugin" isBypassed="0" pluginPosition="2">
      <pluginDescription descriptionXML="&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;
      <pluginStateData stateData="1517.VMjLgPdA...O+fWarAhckI2bo8la8HRLt.iHfTlai8FYo41Y8HRUTYTK3HxO9.BOVMI
t3hKt3hKt3hKt3hKt3hKt3hKt3hKtvyKIMzasA2atUlaz4COIUDYoQ2Pu4Fcx8FarUlb9TSLx3xLuMVZPYjbRMzPDQjQjQzMEYmLlII
3cVZtMEcgQWY9.."/>
    </62764e9d64e745fda0a17cf7b7751ee5>
    <7a4875cbffc446d495f0863ac1f2599c isOccupied="0" Name="Plugin" isBypassed="0" pluginPosition="3"/>
    <bce49e747ee248cd8623009da53a4ea1 isOccupied="0" Name="Plugin" isBypassed="0" pluginPosition="4"/>
  </plugins>
  <channelsRoutedTo>
    <6addfbbecc1a77f50000000000000000/>
  </channelsRoutedTo>
</5c1b19da34ba437bb6a9b2c569848c5f>
```

Figura 17: Estructura ejemplo de un canal de “plugins” de nuestro sistema.

señal, el manejador se asegura de que la primera señal llegue a todos sus oyentes y, entonces, da paso a actualizar esta segunda señal.

Adicionalmente, podemos usar estas estructuras universales para almacenar información relevante para la señal. La idea detrás de esta flexibilidad es para los casos en los que ciertas señales requieran de información actual para que los oyentes reaccionen. Ejemplo de ello es el caso del manejador de la vista de los enrutamientos de un canal; este elemento controla si la interfaz debe o no mostrar visualmente la forma en la que se distribuye la señal de un canal. Para distinguir el canal que ha de mostrar, el manejador almacena el identificador del canal que ha accionado la señal; de esta forma, la pieza de la interfaz encargada de mostrar los enrutamientos es capaz de obtener a qué canal ha de hacer referencia (ver figura 18).

Este diseño se usa en múltiples secciones del proyecto. Es de utilidad en cuestiones generales, como es el caso de “SignalManagerUI”, utilizado para establecer una serie de señales básicas que accionan comportamientos esenciales del sistema software. También se puede aplicar en casos particulares como máquinas de estados, usando las señales para cambiar de estado. Ejemplo de ello es la clase “RoutingActionStateManager” que se encarga de manejar el estado en el que se encuentra la acción de establecer una ruta entre

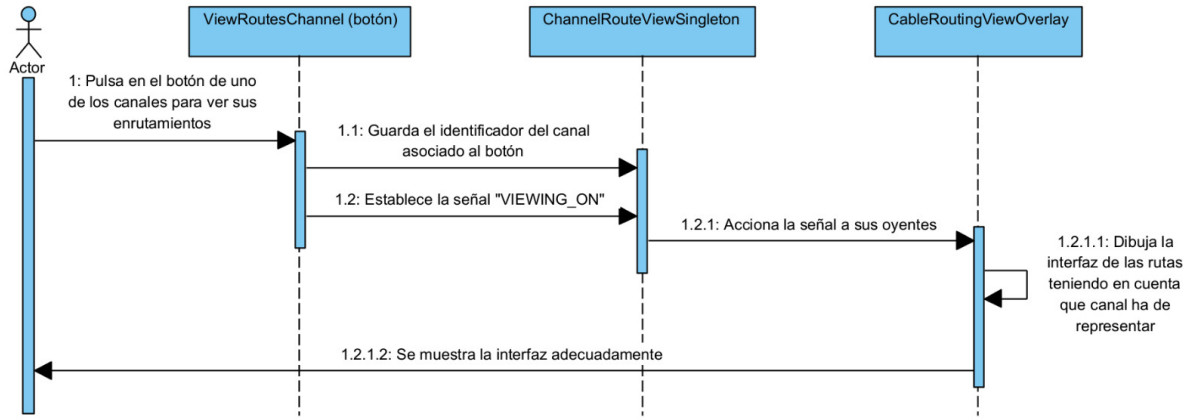


Figura 18: Diagrama de secuencia del sistema para mostrar la interfaz de rutas de un canal.

dos canales (ver figura 19).

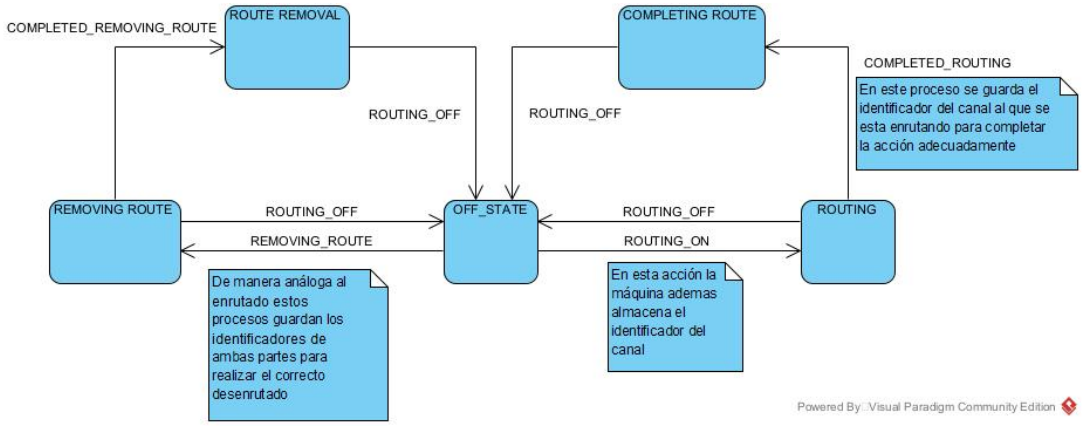


Figura 19: Máquina de estados del sistema de enrutado.

6.1.4. Desarrollo de la exportación de audio

Una funcionalidad clave para el desarrollo de una herramienta enfocada en el trabajo del audio es la capacidad de exportar el resultado a un fichero reproducible. Este sistema se basa en capturar la salida que está direccionando el usuario hacia el canal máster previamente explicado.

El sistema se acciona mediante el uso de un menú superior (ver figura 55) pulsando *Render*. La interacción lanza una señal que recoge nuestra aplicación para mostrarle al

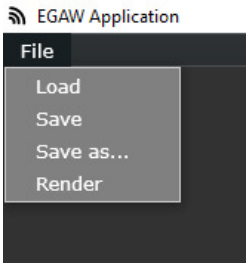


Figura 20: Menú de ficheros proporcionado por la interfaz del sistema.

usuario un menú de sus directorios donde puede seleccionar la ubicación y el nombre del archivo que obtendrá como resultado. A continuación, el usuario recibirá información visual sobre el porcentaje del proceso completado y un mensaje final complementando que se ha finalizado el proceso (ver figuras 21 y 22).

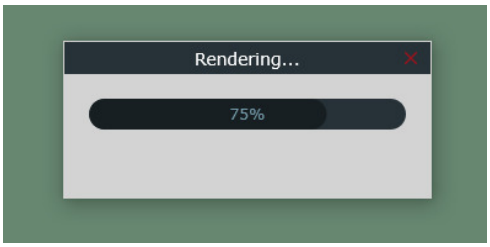


Figura 21: Captura del sistema mientras se realiza el proceso de exportación.

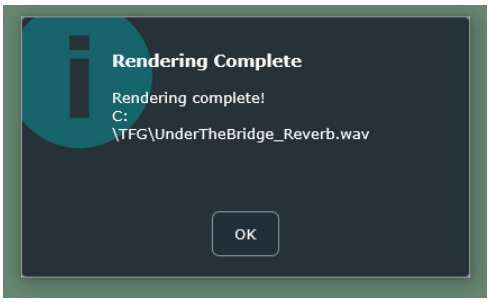


Figura 22: Captura del sistema una vez ha terminado la exportación.

Este proceso se ejecuta gracias a un hilo programado exclusivamente para transformar el proyecto actual en este nuevo archivo resultante. Este proceso configura un objeto de escritura nativo del entorno *AudioFormatWritter* [82] y, utilizando un búfer configurado para simular el búfer utilizado por el hilo principal de audio, recorre el proyecto utilizando el *AudioSystemBus* escribiendo los bloques leídos en el archivo deseado. Este proceso se

lleva a cabo de manera aislada, asegurando la integridad del software en el manejo de hebras.

6.1.5. Implementación del guardado y cargado de proyectos

La capacidad de guardar y cargar proyectos es esencial para obtener un programa realmente funcional. Esto nos permite organizarnos como usuarios, distribuyendo el trabajo por sesiones y nos da la oportunidad de retomar un proyecto siempre que lo necesitemos. Constituye otra de las funcionalidades esenciales para cualquier producto software centrado en trabajar en proyectos. Esta funcionalidad se presenta al usuario mediante el menú de ficheros superior (ver figura 55).

El sistema de archivos se basa en la estructura vertebradora construida con el uso de la clase *ValueTree*. Esta clase de forma nativa nos permite transformar nuestra estructura de datos al formato XML [83], que es un lenguaje de marcado estructurado; de igual forma, podemos construir un nuevo *ValueTree* a partir de una estructura XML. Esta estructuración nos permite crear nuestros propios ficheros de guardado que acomodaremos bajo el formato propio del sistema ".egaw".

Al inicio de nuestra aplicación inicializamos nuestra clase encargada de este proceso de manera independiente llamada *"ProjectFileManager"* que tiene acceso a un *ValueTree* dedicado llamado "FileRestoreProjectData" y a nuestro "projectData" principal, el primero dedicado al proceso de carga y el segundo al de guardado.

El proceso de guardado abre una ventana donde el usuario escoge la ubicación donde guardar el proyecto y su nombre. Una vez escogido, el sistema toma los valores de la estructura principal "projectData" y almacena todo de manera acorde en el archivo del usuario. Adicionalmente, si el usuario no había guardado previamente o cargado de un archivo, nuestra clase almacena temporalmente la dirección del archivo para facilitar un guardado rápido durante la sesión.

El cargado de un proyecto consta de un proceso algo más complejo. Una vez el usuario escoge el archivo de proyecto del cual cargar el estado en el programa, los datos del archivo se vuelcan sobre nuestra estructura "FileRestoreProjectData". Cuando esta estructura de datos está cargada, se acciona un proceso en el *AudioSystemBus* que limpia nuestra estructura "projectData" y se encarga de reconstruir el sistema pieza a pieza como indica

"FileRestoreProjectData". La manera en la que actúa el sistema de montaje es similar a seguir un libro de instrucciones; en este caso, *AudioSystemBus* lee las instrucciones por cada canal proveniente del fichero y lo reconstruye y coloca de acuerdo a los parámetros dados por el fichero de carga. De manera consecuente, cada vez que coloca un canal, nuestro "projectData" reacciona como si fuesen acciones del usuario; de esta forma, la interfaz se predispone para reconstruirse automáticamente, ya que esta responde a los cambios en nuestra estructura de datos principal. Finalmente, se establece una señal para que los componentes de la interfaz se recompongan basándose en "projectData".Una vez finaliza este proceso, tenemos cargado en memoria nuestro proyecto tal y como quedó indicado por el fichero guardado (ver diagrama de la figura 23).

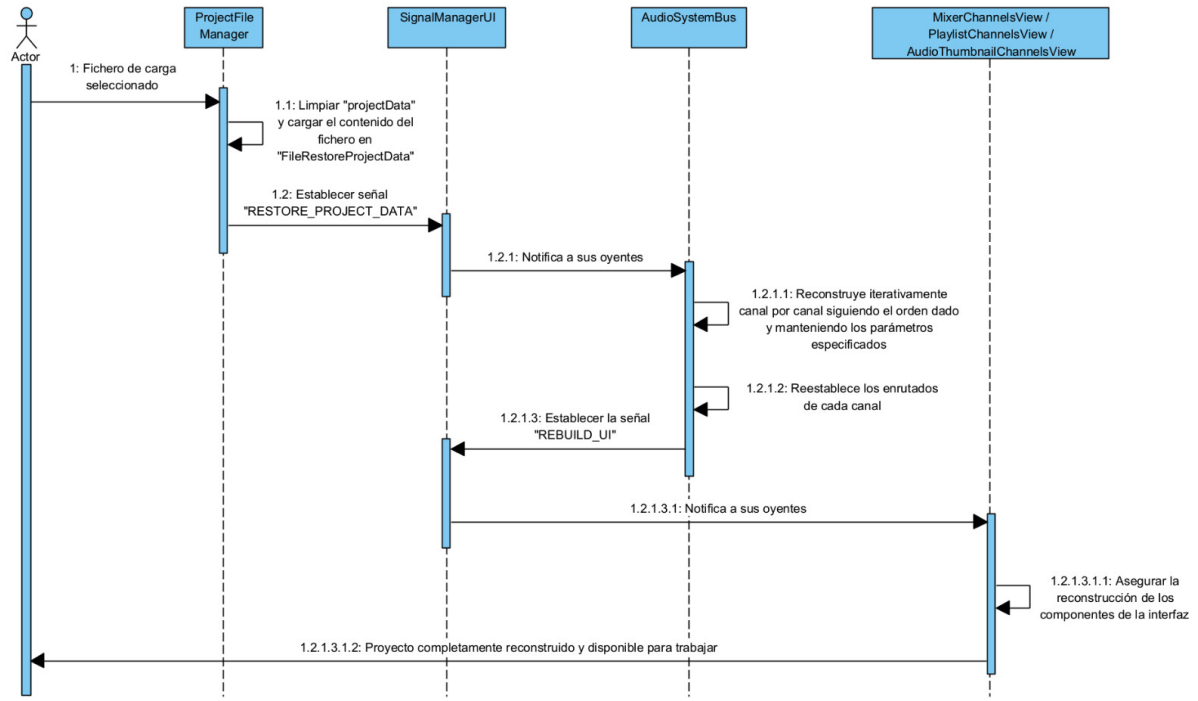


Figura 23: Diagrama de secuencia del sistema de carga de proyectos.

6.1.6. Menú de control principal

Este menú representa de forma estática un elemento global para la interfaz. Esta herramienta nos sirve para controlar aspectos del proyecto o de la ventana en la que estemos trabajando de manera cómoda e intuitiva.

La interfaz se compone de una pareja de botones que representan “Play / Pause” y “Stop” que son funcionalidades relacionadas con la reproducción del audio, permitiendo reproducir, pausar y rebobinar al inicio la reproducción del audio. También podemos visualizar el tiempo de reproducción, obteniendo consciencia de que el sistema está corriendo adecuadamente y dando retroalimentación esencial sobre el funcionamiento del programa. Además, encontramos un controlador que maneja la vista en la que nos encontramos. Por último, dejamos un espacio libre en esta interfaz para adecuar nuevas funcionalidades en función de la vista o de las necesidades futuras del sistema (ver figura 24). Por ejemplo, la interfaz de la lista de reproducción nos da la opción de controlar el zoom con el que vemos los bloques de audio (ver figura 25).

Figura 24: Sección de la interfaz dedicada al menú de control principal básico.

Figura 25: Menú de control principal alterado por las funcionalidades adicionales de la vista.

El sistema de control del audio funciona transmitiendo señales mediante nuestra clase *SignalManagerUI*. Por otro lado, la visualización de la pantalla se controla haciendo uso del sistema de propiedades de “projectData”; esta facilidad nos permite no solo acomodarnos al sistema de oyentes de nuestro *ValueTree*, sino que también nos permite guardar esto como un estado en los archivos de guardado.

6.2. Mezclador de canales

A lo largo de esta sección exploraremos desde el punto de vista de diseño y composición las funcionalidades que presta el mezclador, de qué se trata y cómo interactúa el usuario con él en diferentes modos.

6.2.1. Conceptos esenciales del mezclador

Un mezclador, en nuestro caso, representa una mesa de mezclas analógica. En esencia, una mesa de mezclas es un dispositivo electrónico que permite al usuario recopilar todas sus grabaciones para manejar su señal y obtener una única salida final que mezcle las pistas deseadas [84]. Una mesa de mezclas puede resultar abrumadora cuando no sabemos desglosar sus componentes; a primera vista, vemos una amalgama de botones, controladores y puertos sin aparente sentido (ver figura 26).

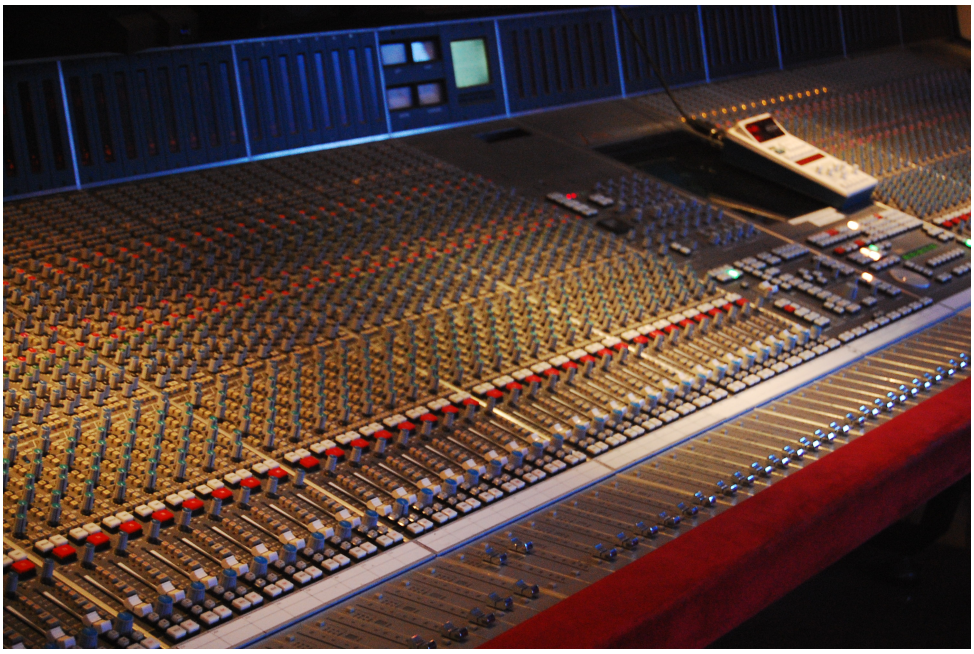


Figura 26: Mesa de mezclas real en un estudio de grabación.

Entender que estas herramientas pueden ser estudiadas por canales hace mucho más fácil su comprensión. Virtualmente, acostumbramos a crear y destruir canales de audio dinámicamente, mientras que en el mundo real no podemos añadir y eliminar estos elementos a un dispositivo hardware a nuestro antojo. Cada franja vertical de elementos representa un canal unitario, esto hace más sencillo el problema de entender una mesa de mezclas analógica. La comprensión del funcionamiento individual de un canal nos proporciona una perspectiva desde la cual podemos desglosar la pieza hardware por canales, simplificando la complejidad del sistema. Cabe destacar el flujo de la señal estandarizado, es decir, las mesas de mezclas reciben la señal desde la parte superior de un canal y la procesan verticalmente de arriba hacia abajo (ver figura 27).

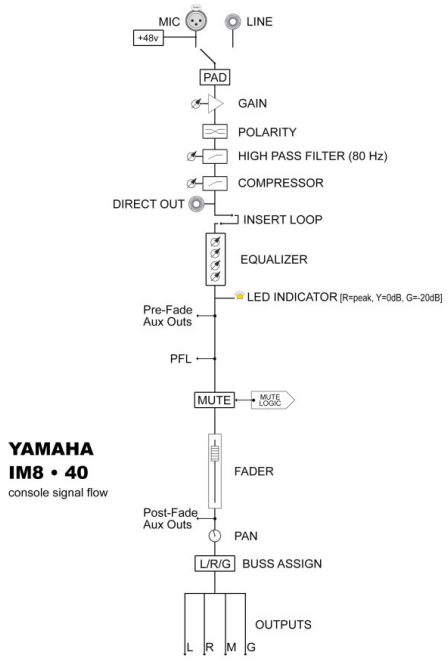


Figura 27: Diagrama del flujo de la señal de la consola de mezclas Yamaha IM8-40.

En nuestro sistema software haremos una representación de una mesa de mezclas adaptándola al entorno virtual y acomodando los elementos para facilitar la comprensión de la misma. En nuestro caso, al no depender de recursos físicos limitados, establecemos la posibilidad de crear y deshacer canales en todo momento, conservaremos el flujo de la señal estandarizado, construiremos un sistema de enrutado basado en conexiones de cables virtuales y diferenciamos tres tipos de canales según su funcionalidad: canal de audio, canal de mezcla y canal de efectos.

La interfaz completa incluye un visor donde poder acceder a los canales de audio habilitados por el usuario y el canal maestro predefinido por la arquitectura del sistema (ver figura 28). De forma adicional, contamos con un botón que nos habilita como usuarios finales para poder añadir canales. Este botón despliega un menú vertical donde podemos seleccionar el tipo de canal que queremos incluir en nuestro mezclador.

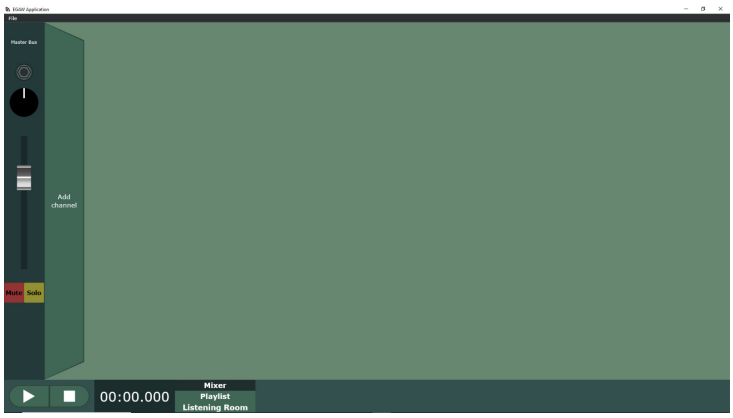


Figura 28: Interfaz base del mezclador.

6.2.2. Componentes genéricos

A lo largo de esta sección explicaremos individualmente los componentes genéricos que forman parte de los canales. Haremos una revisión de los componentes en orden verticalmente de acuerdo a la distribución estandarizada dentro del sistema para los mismos. Para visualizarlos, nos basamos en un diagrama que parte de un canal de mezcla, puesto que este es uno de los más completos y comparte elementos con otros canales (ver figura 57).

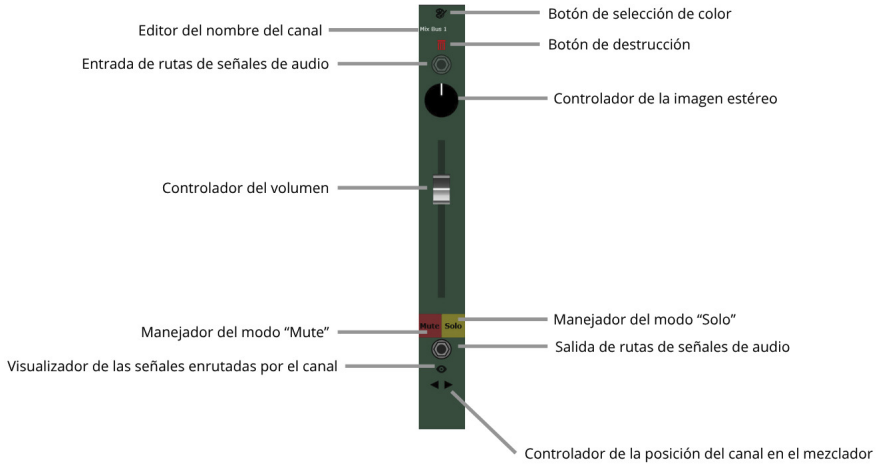


Figura 29: Diagrama con los componentes del canal de mezcla señalizados.

El primer bloque de componentes son controles básicos sobre la personalización y utilidades del canal. El primero de los componentes es selector de color del canal; este permite

cambiar el color con el que visualizamos el elemento completo. Este botón despliega una ventana donde el usuario puede seleccionar, haciendo uso de una paleta de colores, el color de fondo del canal (ver figura 58). Justo debajo encontramos el nombre designado para el canal; este también se puede modificar a nuestro antojo haciendo doble clic y editando el texto. Cerrando este bloque, encontramos el botón de destrucción del canal; como medida de seguridad, despliega un cuadro de texto de confirmación previo a la destrucción total del canal (ver figura 31).

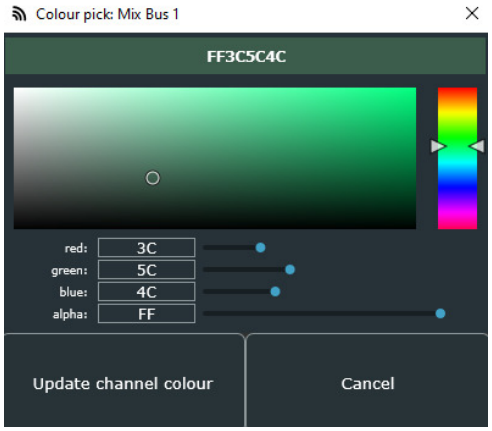


Figura 30: Selector de color desplegado para elegir el color base del canal.

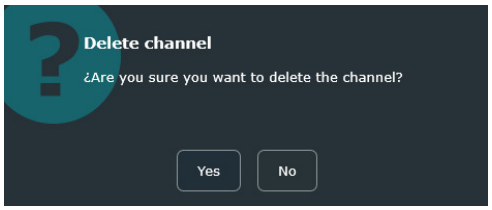


Figura 31: Diálogo mostrado para confirmar la eliminación de un canal.

A continuación encontramos los componentes relacionados con el manejo de señales y procesamiento básico. El primero de los componentes representa el receptor de señales, este se activa únicamente cuando el sistema de enrutado está activo y este componente puede ser destino de una decisión del usuario. A continuación, encontramos el controlador de señal estéreo que define si la señal se ha de balancear hacia el emisor izquierdo o derecho. Seguido, se encuentra el controlador de volumen que maneja en decibelios si hay que aumentar o disminuir el volumen de la señal original. También, dentro de este bloque encontramos el accionador de enrutados, simboliza la salida del canal y, haciendo

clic en él, nos permite iniciar la gestión de la creación o eliminación de envíos de la señal producida por el canal. En penúltimo lugar, encontramos la pareja de botones de "Mute" y "Solo". La funcionalidad "Mute" nos permite silenciar por completo la salida de audio producida. Mientras tanto, la funcionalidad de "Solo" orquesta la salida para que exclusivamente reciba las señales de audio que el usuario seleccione particularmente, en caso de que ningún canal esté predispuesto a actuar en modo "Solo" esta funcionalidad no tiene efecto alguno en el mezclador. Finalmente, dentro del bloque encontramos el botón encargado de la visualización de las rutas del propio canal.

Para cerrar los elementos que componen la interfaz de un canal, encontramos el manejador de posición. Este componente controla otra funcionalidad básica para cualquier elemento del mezclador; mediante él, el usuario puede mover la ubicación del canal. Esto permite a los usuarios acomodar los elementos visuales de acuerdo a su manera personal de trabajar.

6.2.3. Canal de audio

Nuestro sistema de mezclas se basa en los canales de audio, son los bloques esenciales que dan sentido al resto de estructuras, sin una señal de audio no tiene sentido una infraestructura dedicada a procesar señales. Estos elementos software cargan un archivo de audio que supone la entrada de audio. Mantienen una interfaz de audio simplificada habilitando las funciones generales de un canal y conservando los controladores de imagen estéreo y volumen (ver figura 32).

Destacar de este componente el sistema diseñado para poder enviar de forma múltiple la señal hacia varios receptores, o más bien consumidores. Los canales de mezcla actúan como consumidores bajo demanda, es decir, si un canal de audio se conecta a uno de mezcla, de forma interna lo que sucede es que el de mezcla le pide al de audio los bloques con la señal que necesita. Esta estructura ocasionaba colisiones en tiempo real debido a que si el canal de audio quedaba conectado a dos mezcladores simultáneamente, estos colisionaban, ya que ambos querían el mismo bloque al mismo tiempo. La solución fue hacer un sistema de guardado de los bloques; esto significa que el primer consumidor que acceda al método del canal de audio forzará al canal a construir el bloque leyendo del archivo. Sin embargo, cuando un segundo consumidor aparezca, el canal de audio

Figura 32: Interfaz que representa un canal de audio.

únicamente le enviará una copia del bloque ya procesado. De esta forma evitamos los errores audibles en el acceso simultáneo a un recurso único.

6.2.4. Canal de mezcla

Un canal de mezcla funciona como un unificador de señales. Esto es especialmente útil a la hora de gestionar el manejo del audio por el usuario; además, permite reducir los recursos utilizados. Por ejemplo, si necesitamos que tres grabaciones distintas relacionadas a una misma pista de guitarra se procesen de igual manera, podemos primeramente enviarlas a un canal de mezcla y entonces procesar esta señal unificada; de esta forma, en lugar de generar 3 procesamientos idénticos, únicamente tenemos que configurar uno. La interfaz cuenta con todos los elementos básicos que conforman nuestros canales software, siendo esta muy similar a la del canal de audio, pero añadiendo el componente que simboliza una entrada analógica, donde poder enviar las señales para unificar (ver figura 33).

6.2.5. Canal de efectos

El canal de efectos o también reconocible como un “rack de plugins” es el canal que más se diferencia del resto de herramientas disponibles. Su funcionalidad es alojar los efectos para procesar una señal única de audio. El canal cuenta con cinco huecos

Figura 33: Interfaz que representa un canal de emzcla.

para alojar herramientas software en formato “VST3”, estos efectos pueden ser nativos del entorno software o también pueden incluirse herramientas de terceros. El elemento software representa virtualmente los bloques diseñados para contener procesadores de la señal analógicos; estos artefactos están diseñados y estandarizados para poder montarles componentes electrónicos modulares dedicados al procesamiento de la señal auditiva (ver figura 34).

Figura 34: Sistema físico para la organización de procesadores de señal analógicos.

La virtualización de estos sistemas físicos supone la democratización de las herramientas para la mezcla; estos artefactos suelen ser costosos debido a la sofisticada electrónica

y necesitan una infraestructura previa para poder utilizarlos adecuadamente. La contraparte digital de estas herramientas puede encontrarse en múltiples formatos y precios económicos; esto rompe una de las barreras de entrada a la ingeniería de mezcla.

La interfaz de este componente sigue los patrones de diseño básicos establecidos en nuestra interfaz, incorporando los medios necesarios para hacer manejo de los efectos. En el canal vemos el número de huecos disponibles para insertar efectos (ver figura 35). Una vez el usuario ha seleccionado algún efecto, esta interfaz cambia (ver figura 36), permitiendo al usuario acceder a la herramienta software en una ventana aislada, manejar el orden en el que se aplican los efectos, activar una funcionalidad de "bypass" o eliminar el efecto. La funcionalidad de "bypass" puede resultar la menos intuitiva de todas; este control acciona que la señal sobrepase el efecto sin que este se aplique, es decir, mientras esté activa, el efecto no afectará a la señal producida.

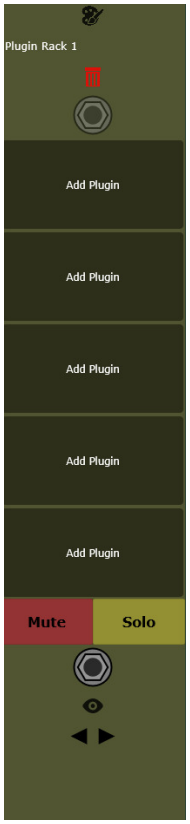


Figura 35: Interfaz que representa un canal de efectos.

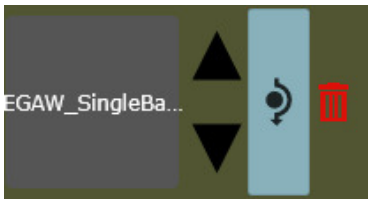


Figura 36: Interfaz mostrada cuando un hueco es ocupado por un modulador de la señal.

6.2.6. Sistema de enrutado

El sistema establecido para el envío de señales se asemeja visualmente a las conexiones realizadas utilizando cables en los dispositivos analógicos. Como vimos anteriormente, la responsabilidad técnica de recoger señales recae sobre la clase nativa de JUCE, *MixerAudioSource*.

A nivel funcional, contamos con los elementos de salida y llegada y un botón de visualización en nuestros componentes para interactuar con el sistema de enrutado. Las funciones de salida y llegada accionan la máquina de estados explicada previamente (ver figura 19). Adicionalmente, disparan la interfaz que visualiza las rutas del canal deseado y una interfaz invisible que captura los clics, de forma que si el usuario quiere cancelar la acción simplemente tiene que hacer un clic fuera de cualquier elemento de recepción de señal y el sistema aborta el enrutamiento. La forma de distinguir sobre si el usuario va a crear o eliminar una ruta de la señal queda representada visualmente en la interfaz; el usuario podrá escoger entre cada acción alternando entre clic izquierdo, para crear, y clic derecho, para eliminar.

Por último, incluimos dentro de nuestro sistema funcional de enrutado la posibilidad de visualizar las conexiones de un canal. Utilizando el botón de control dedicado a ello, podemos desplegar una metáfora visual que simboliza mediante cables hacia qué canales se está enviando la señal producida por el canal (ver figura 37).

6.3. Lista de reproducción

En este apartado del capítulo trataremos de aclarar qué es la lista de reproducción, cómo se compone su interfaz y destacar funcionalidades que presta este sistema al usuario.

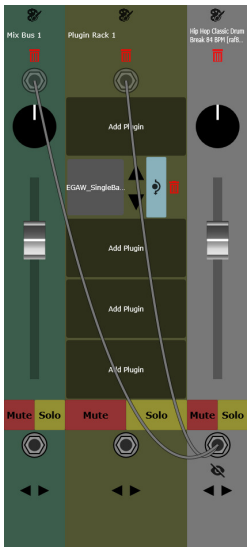


Figura 37: Interfaz que muestra las rutas establecidas por el canal de audio seleccionado.

6.3.1. Funcionalidad básica y distribución visual

Este apartado visual se dedica a manejar las pistas de audio en el tiempo y ubicarlas en dicha dimensión. Permite al usuario visualizar y modificar la forma en la que se colocan temporalmente las pistas de audio asociadas a los canales de audio. No solo nos permite editar la línea temporal del proyecto, sino que también nos permite movernos libremente por ella, facilitando la edición del audio.

La interfaz se compone de tres elementos principales (ver figura 38). La zona dedicada al control de los canales de audio, donde el usuario podrá ajustar ciertos parámetros de acuerdo al canal análogo del mezclador. El área de edición temporal donde el usuario podrá ubicar los archivos de audio en forma de bloques y realizar operaciones con ellos. Por último, el control del tiempo donde contamos con una línea temporal que marca el progreso del audio y la cual nos servirá para desplazarnos en esta dimensión.

Además, en esta sección de la interfaz encontramos una herramienta de zoom que nos permite ajustar la interfaz según nuestras necesidades. En caso de necesitarlo, el usuario podrá ampliar o reducir la interfaz relacionada con el tiempo para tener una visión más local o global del proyecto en función de la situación (ver figura 39).

Figura 38: Diagrama con las diferentes secciones de la lista de reproducción.

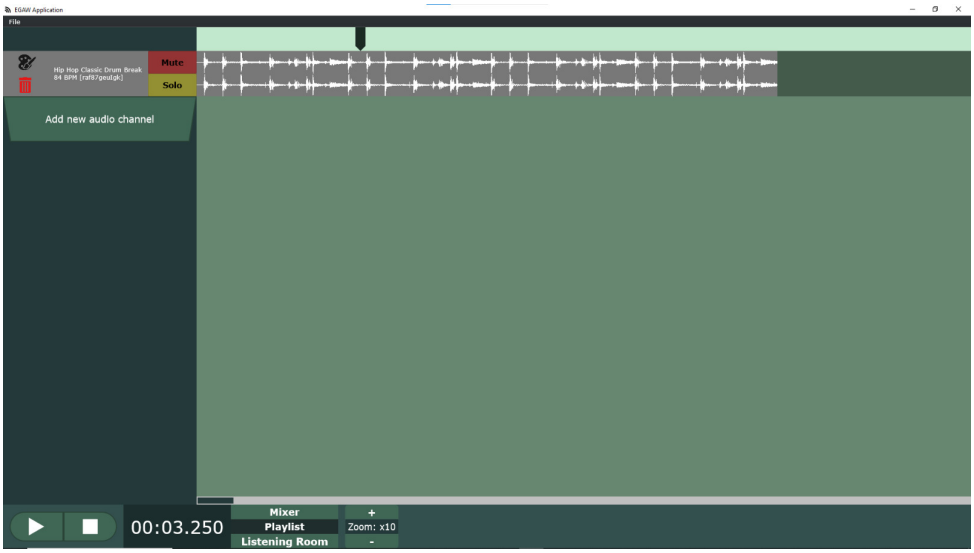


Figura 39: Interfaz de la lista de reproducción con el zoom al máximo.

6.3.2. Canales de audio

Esta subsección de la interfaz simplemente se encarga de representar para esta vista los medios con los que estamos trabajando, de forma que sean fácilmente asociables al canal correspondiente del mezclador. Representamos los canales de audio y ofrecemos opciones de control de manera similar a como lo hacemos en la interfaz previamente mencionada; de esta manera, el usuario puede identificar claramente a qué elemento de la composición está afectando. Además, mantenemos una cohesión entre los elementos de control de forma que todas las funcionalidades disponibles en esta representación del

canal son equiparables de forma visual a las funciones que nos ofrece su contraparte visual del mezclador.

Adicionalmente, contamos con un botón similar al situado a la derecha del mezclador, pero para añadir canales de audio. De esta forma, el usuario puede añadir rápidamente los elementos clave de esta interfaz, que son los canales de audio.

6.3.3. Sistemas de bloques de audio

El sistema de bloques establecido está diseñado para poder posicionar y manejar el aspecto temporal de una composición musical. Cada canal tiene su propio raíl sobre el que queda fijado el bloque que representa el archivo de audio. Estos bloques son de tamaño proporcional al tiempo que ocupan. Como usuarios, podemos desplegar un menú que nos permite activar si queremos ver la onda de audio del bloque o ajustar su longitud (ver figura 61). Por defecto, la onda de audio del canal de audio quedará oculta; esto es debido a que puede ser un elemento que desvíe nuestra atención de la respuesta auditiva que obtenemos del canal.

Figura 40: Menú desplegado para manejar parámetros de los bloques de audio.

La movilidad de los bloques es completamente horizontal, permitiendo a los usuarios ajustar el instante en el que desean que comience la reproducción del archivo. Los ajustes de longitud de estos elementos de audio nos permiten ajustar el instante del fichero original en el que comienza y termina el bloque; el sistema tiene en cuenta la longitud completa del archivo y permite a los usuarios que el fragmento reproducido en el proyecto inicie en un punto más avanzado al original o que termine su reproducción de manera prematura. Este control se despliega usando el menú previamente presentado y se muestra en forma de ventana en la que podemos ajustar estos parámetros (ver figura 62).

Figura 41: Control del tiempo de inicio y fin del bloque de audio.

6.3.4. Control del tiempo

El control de la posición temporal se hace mediante nuestra barra superior; ubica el instante de reproducción mediante un puntero, que forma una metáfora de la aguja de un tocadiscos [85]. Esta aguja se desplaza visualmente cuando el sistema está reproduciendo el contenido. De manera intuitiva, podemos mover este marcador de posición, ya sea arrastrándolo o haciendo clic en el instante al que deseemos enviar la aguja.

De manera interna, este simple sistema orquesta la reproducción de los elementos dispuestos en el proyecto. Disponemos de una clase del tipo "Singleton" llamada *GlobalPlayhead* que maneja de manera autónoma el número de muestras que está leyendo el sistema y lo relaciona utilizando una simple función matemática (1) que aprovecha la propiedad inherente del "Sample Rate" puesto que es el número de bloques que lee el sistema por segundo.

$$\text{Instante de reproducción} = \frac{\text{Numero de bloques leídos hasta ahora}}{\text{Sample rate}} \tag{1}$$

La clase *GlobalPlayhead* almacena la posición en número de bloques leídos del controlador de tiempo. A partir de este dato, los canales de audio consultan esta posición accediendo mediante la instancia única y reproducen en sincronía el bloque correspondiente al instante marcado por la aguja.

6.4. Plano de audición

Esta sección de la interfaz es la más simple de todas y es el factor que le da el valor a la misma. En esencia, es una pantalla vacía donde el usuario no tiene retroalimentación ni estímulos visuales más allá de ver el tiempo correr en el visor (ver figura 42).

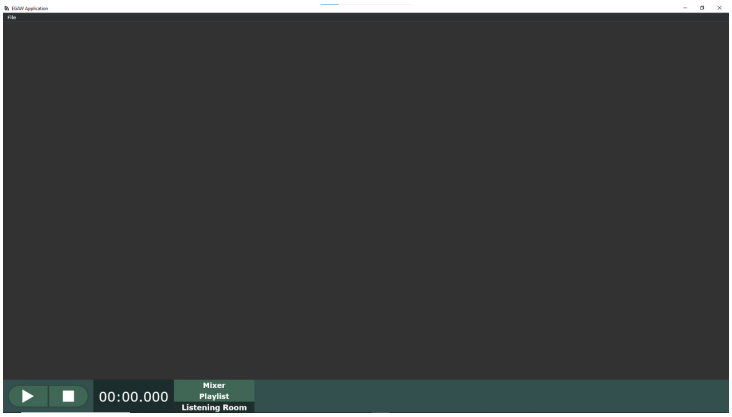


Figura 42: Interfaz dedicada al plano de audición proporcionada en el sistema.

Esta falta de recursos visuales es completamente intencional. La idea es que el usuario disponga de un entorno en el que, sin necesidad de salir del sistema virtual, pueda centrarse por completo en la respuesta sonora del proyecto y no desviarse inintencionadamente por culpa de estímulos visuales.

6.5. Linea de plugins

En este apartado del capítulo trataremos los plugins. Son complementos software que aportan valor al sistema principal, permitiendo al usuario procesar la señal aplicando efectos en función del resultado deseado.

6.5.1. Concepto principal de la linea de plugins

Como desarrolladores del producto, buscamos construir un ecosistema de plugins completo; a través del cual los usuarios de la aplicación puedan trabajar la señal por completo sin necesidad de acudir a software de terceros. Esta idea no solo trata de afianzar al usuario a nuestra línea de productos, sino que, además, pretende seguir cultivando la

metodología guiada por el oído, es decir, estos elementos software mantienen la misma filosofía principal que el entorno de trabajo principal.

Estas extensiones del software principal vendrán de forma nativa implementadas con la instalación del producto, lo que las hace fácilmente accesibles a los usuarios, ya que las podrán encontrar preinstaladas junto con el sistema. Los elementos construidos hasta la fecha son un ecualizador y un reverberador; este grupo de plugins nativos puede ofrecer una clara línea de futuro con la consecuente construcción de muchos otros efectos.

6.5.2. Plugin de ecualización

Este efecto, también llamado "SingleBandEQ" dentro del entorno, es un modulador que afecta a la frecuencia de la señal. Esto se utiliza para moldear la señal frecuencialmente, pudiendo seleccionar el punto sobre el que actuar en el espectro de frecuencias, normalmente entre veinte y veinte mil hercios, seleccionando una función que aplicar a la frecuencia y, en caso de que esta lo permita, ajustar el volumen con el que se aplica la función [86].

Las funciones aplicables dentro de un ecualizador son muchas, el nuestro cuenta con las más comunes. Contamos con funciones de corte como son las conocidas como "pass" o "cut" que nos permiten seleccionar una frecuencia concreta y realizar un corte en ella según nuestro antojo, por ejemplo, un "high cut" en los mil hercios hace que todo lo que sobrepase esa frecuencia quede silenciado. También permitimos el uso de funciones con un parámetro de volumen como son las tipo "shelf" o "peak", un "peak" por ejemplo, nos permite acceder a una frecuencia concreta y aumentar o disminuir su volumen [87].

La interfaz, siendo simple, contiene todos los elementos necesarios para trabajar un ecualizador de una sola función. En ella encontramos un controlador vertical que representa el volumen de la función en caso de necesitarse, un controlador horizontal de la frecuencia a la que debe actuar la función y, por último, un selector de función donde escoger el efecto que queremos aplicar sobre la señal de sonido (ver figura 63).

La interfaz de este sistema no solo mantiene una coherencia visual con los elementos del sistema principal, sino que, además, mantiene la filosofía basada en la audición. El producto oculta todo parámetro de modo nativo de forma que impulsa a que el usuario critique el resultado mediante el oído; de igual manera, el usuario puede acceder a estos

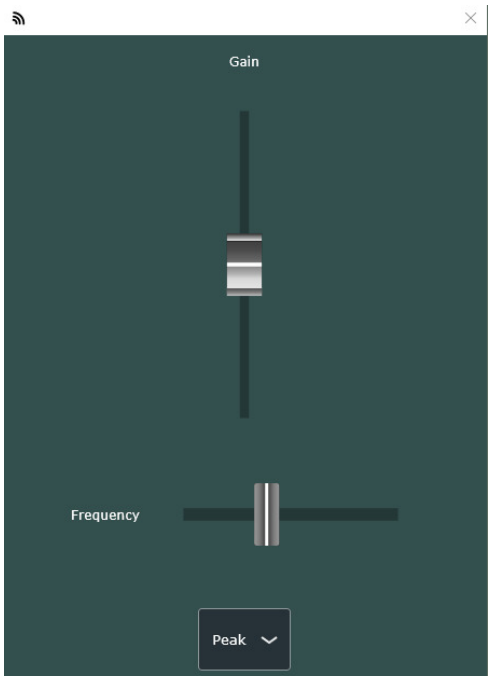


Figura 43: Interfaz del plugin de ecualización diseñado.

valores, aunque no será el objetivo del diseño.

6.5.3. Plugin de reverberación

Este elemento, nombrado dentro del entorno como “SimpleVerb”, es un modulador que simula la reverberación física del sonido [88]. El uso de este efecto depende, como en la mayoría de estos, del resultado que queramos obtener; por ejemplo, es común usarlo en voces de música pop para agregar presencia, voluminosidad y aportar la sensación de divinidad [89].

Este efecto cuenta con una interfaz sencilla en la que podemos manejar la mezcla de señales que deseamos emplear, es decir, qué cantidad de señal “seca” (sin efectos) y “mojada” (aplicando el efecto) queremos de salida. Además, podemos ajustar parámetros que afectan a la reverberación producida. Encontramos el “damping”, que refleja la dureza de la habitación virtual, afectando al comportamiento de las frecuencias agudas. El “room size” o tamaño de habitación, que afecta a la duración de la reverberación. Por último, el “width” o ancho, que afecta a la forma en la que el efecto ensancha en el espacio estéreo la señal [90] (ver figura 64).

De igual forma que el efecto anterior, este plugin mantiene la guía ideológica sobre la

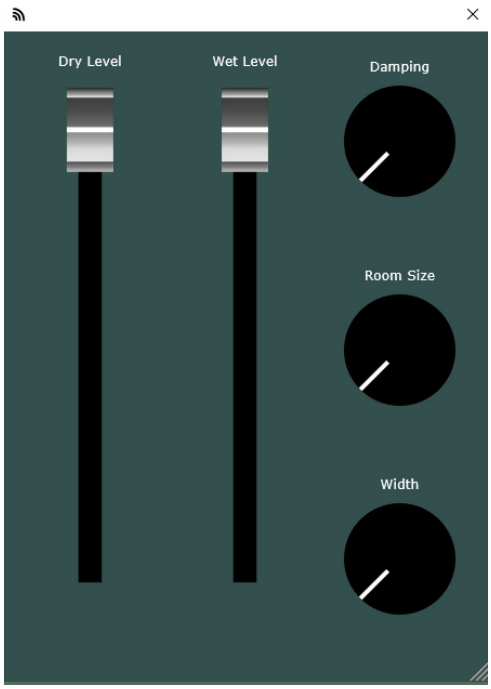


Figura 44: Interfaz del plugin de reverberación diseñado.

que hemos construido nuestro sistema. En este elemento se fomenta el uso crítico del oído en la mezcla y deja la parametrización en un segundo plano; será accesible, pero no parte principal de la interfaz.

6.6. Instalador

El proyecto cuenta adicionalmente con un instalador desarrollado con la herramienta Inno Setup [53]. Esta herramienta es utilizada para construir instaladores en formato ejecutable de Windows. El resultado es un instalador sencillo que contiene el ejecutable compilado del producto software junto con una carpeta que contiene los "plugins" desarrollados que forman parte del entorno virtual (ver figura 45).

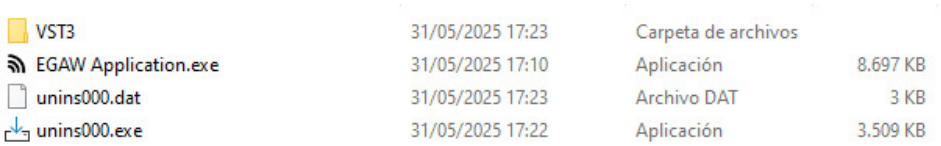


Figura 45: Resultado de la instalación del software utilizando el ejecutable desarrollado.

6.7. Testing y pruebas con usuarios

A lo largo de esta sección nos centraremos en las pruebas relacionadas con el proyecto. Debido al enfoque sensorial del proyecto, estas pruebas no podrán restringirse exclusivamente a pruebas técnicas.

6.7.1. Pruebas software

Al inicio del desarrollo se plantea el uso de pruebas unitarias para asegurar la integridad del código. A medida que el proyecto comienza a producirse, se toma la decisión de paralizar la producción de este tipo de pruebas. Esta decisión viene incentivada principalmente por el coste temporal en comparación con el beneficio que aportan a un proyecto unipersonal. El ambiente ideal para la creación de estas pruebas es que las estructure personal ajeno al código que se pone a prueba; en nuestro caso, esto es imposible.

Frente a esta decisión, suplimos la falta de pruebas escritas explícitamente en código con pruebas manuales sobre las funcionalidades técnicas del software. Durante el desarrollo, todas las funciones se han probado exhaustivamente, llevándolas al límite y experimentando con las diferentes situaciones posibles. La idea de estas pruebas no es probar individualmente los elementos, sino poder probar los sistemas y su funcionamiento conjunto; para agilizar el proceso de pruebas, hacemos uso del sistema de guardado y cargado implementado.

6.7.2. Pruebas realizadas con usuarios

En el desarrollo del software hemos realizado pruebas de usabilidad a nivel de experiencia de usuario para comprobar las interacciones de los usuarios y cómo se comportan diferentes actores ante tareas específicas del programa. La metodología es simple, se graba la pantalla y la cámara del dispositivo para poder evaluar a posteriori el comportamiento de los usuarios de forma detenida, se establece una serie de acciones u objetivos a lograr mediante el software y se deja libertad al usuario para que interactúe naturalmente con el software (ver figura 46) [91].

Las pruebas han sido realmente útiles para comprobar los comportamientos de usuarios que nunca antes habían utilizado el sistema. Destacar que la complejidad y tecnicidad del

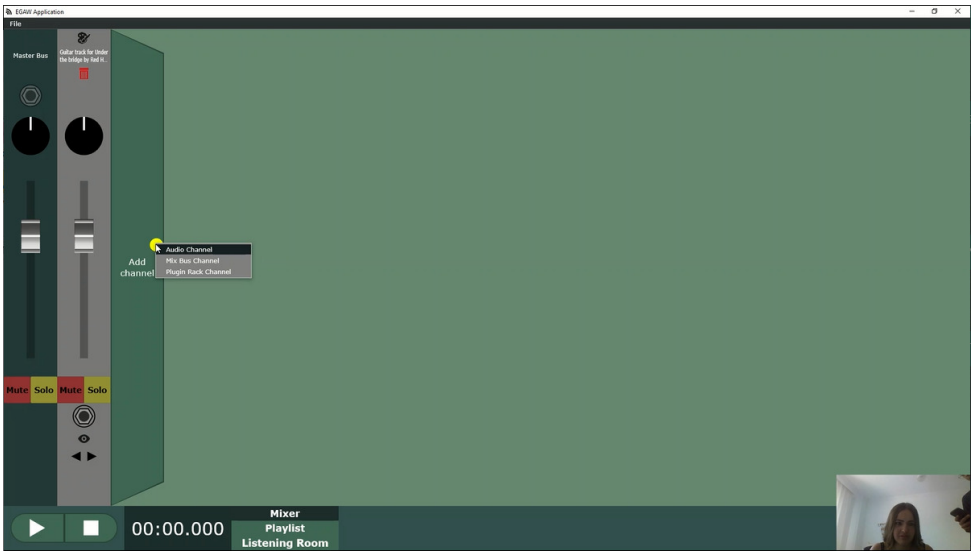


Figura 46: Prueba de usuario realizada.

ámbito de la aplicación, en muchas ocasiones, podían ser causa de confusión por la falta de familiaridad con conceptos de audio. Esta carencia técnica se tuvo que suplir en algunas pruebas deconstruyendo las instrucciones, es decir, en lugar de pedir como tarea "crea dos canales y enruta uno en otro", se desglosaba pidiendo que accediera a cierta vista, que se creasen dos tipos concretos de canal y que enviase la señal de uno a otro.

7

Conclusiones y Líneas Futuras

En este capítulo concluimos el proyecto reflexionando sobre el valor que aporta y planteando posibles líneas futuras de desarrollo en el producto alcanzado.

7.1. Conclusiones

El desarrollo de este proyecto ha supuesto un reto en múltiples ámbitos. El primero de ellos era investigar y entender al completo cómo psicológicamente estamos predispuestos a enfocar nuestra atención en los estímulos visuales. A continuación, nos encontramos con un segundo gran reto, introducirnos e interiorizar todos los conceptos adquiridos sobre el mundo del desarrollo de software dedicado al audio; pese a tener una base previa sobre el aspecto técnico del audio, el desarrollo de software dedicado a él sigue siendo una tarea muy compleja. Sumergirse en JUCE como librería significa aprender un entorno completamente único. Construido sobre C++ y apoyado en la orientación a objetos del lenguaje, el marco de JUCE abarca todo lo necesario para construir un sistema completo, por ello es necesario abarcar el manejo de las interfaces, el uso y control de hilos, o entender estructuras de datos complejas para alcanzar el producto construido. No solo nos hemos enfrentado a pericias concretas de la temática desarrollada, sino que también hemos logrado diseñar una estructura global del software sostenible, haciendo uso de patrones de diseño y estructuración del código, evitando el acoplamiento y las dependencias entre funcionalidades.

La existencia de retos y dificultades trae de forma inherente el aprendizaje y desarrollo de los conocimientos. Con el trabajo desarrollado hemos adquirido y afianzado ideas que

nos forman como Ingenieros del Software y nos capacitan para su desarrollo, habiendo sido capaces de construir un producto completo y autosuficiente con un objetivo claro. Diseñar estructuralmente el funcionamiento de los mecanismos internos y orquestar la modularidad de la interfaz en un proyecto tan amplio aportan un gran valor al aprendizaje adquirido.

A raíz de este proyecto hemos obtenido una base sólida y funcional sobre la que construir todo un producto comercial dedicado al audio. El valor distintivo que hace único al proyecto es el entendimiento de la cognición humana y el ajuste realizado sobre la forma en la que interactuamos con el sistema de acuerdo al objetivo perseguido por el usuario. Este trabajo no solo aporta la experiencia adquirida durante el desarrollo, sino que, además, deja consigo un sistema software completamente disponible para el uso en la mezcla de proyectos de audio.

7.2. Líneas Futuras

Este proyecto cuenta con la posibilidad de convertirse en el futuro en una herramienta masiva. Hay múltiples caminos en los que expandir los horizontes del producto. Los avances a largo plazo sobre la base construida pueden dar lugar a un sistema capaz de posicionarse entre las herramientas más populares en la industria del audio, distinguiéndose siempre por la filosofía establecida alrededor de la audición.

A corto plazo las necesidades del software son claras. Ajustes para facilitar la usabilidad en más ámbitos, si cabe, por ejemplo, habilitar un sistema de arrastre para hacer más intuitivo e interactivo el recorte del audio. Disponer de un manejador de acciones que facilite al usuario el poder deshacer sus acciones siguiendo un historial ordenado. Desarrollar un sistema de recuperación de archivos en la carga de proyectos; actualmente, al cargar un proyecto, si un archivo de audio no es encontrado, el sistema simplemente se cierra; lo ideal sería disponer de un menú donde el usuario pueda decidir si reubicar el archivo o descartar el canal para la adecuada carga. Estos son solo algunos de los objetivos a corto plazo disponibles para el proyecto. Por la metodología en la que se ha desarrollado el sistema, los objetivos cercanos pueden definirse con mayor exactitud, mientras que los objetivos a largo plazo se establecen como grandes hitos a los que aspiramos y, conforme nos acercamos, desglosamos en forma de requisitos concretos.

Otro de los objetivos interesantes es disponer de la integración adecuada con elementos hardware. Facilitar el uso de tarjetas de sonido o controladores virtuales para la mezcla son objetivos claros en el desarrollo de una aplicación que pretende competir en el mercado con alternativas que han sido capaces de desarrollar incluso hardware dedicado para sus propias herramientas software.

En el desarrollo software encontramos además un camino que puede dar lugar a una gran cantidad de novedades, y este es la elaboración de un entorno completo de plugins. Construir herramientas complementarias puede dar lugar a miles de proyectos; desde las adiciones más elementales como compresores, saturadores o demoradores de la señal, hasta sistemas que integren el uso de la inteligencia artificial para trabajar la señal, ya sea limpiándola de artefactos o reconstruyendo grabaciones dañadas. Las posibilidades son verdaderamente infinitas puesto que siempre podremos idear nuevas maneras de interactuar con el sistema y nuevos procesamientos que facilitar a los usuarios. La clave de esta generación de nuevos productos software se encuentra en mantener una cohesión estilística e interiorizar en el desarrollo la filosofía orientada al oído.

Referencias

[1] BBVA. *La industria musical el 5o mercado mundial*. 2024. URL: https://www.bbva.ch/blog/inspiracion/la-industria-musical-el-5o-mercado-mundial.html (visitado 20-05-2025).

[2] SchoolTraining. *Analógico y digital. Dos métodos y dos épocas*. 2022. URL: https://schooltraining.es/noticias/analogico-y-digital-dos-metodos-y-dos-epocas (visitado 20-05-2025).

[3] Steinberg. *¿Qué es un DAW y para qué sirve?* URL: https://www.steinberg.net/es/tutorials/what-is-a-daw/ (visitado 20-05-2025).

[4] José A. Medina. *La mezcla: ideas fundamentales*. 2008. URL: https://www.hispasonic.com/tutoriales/mezcla-ideas-fundamentales/2419 (visitado 20-05-2025).

[5] Amanda Medeiros. *Masterización y Mezcla: ¿Cuál es La Diferencia?* 2021. URL: https://moises.ai/es/blog/consejos/masterizacion-y-mezcla-diferencia/ (visitado 20-05-2025).

[6] Lorin Lachs. *8.10: Percepción multimodal*. 2022. URL: https://espanol.libretexts.org/Ciencias_Sociales/Psicologia/Libro%3A_Psicolog%C3%ADa_(Noba)/Chapter_8%3A_Sensaci%C3%B3n_y_Percepci%C3%B3n/8.10%3A_Percepci%C3%B3n_Multimodal (visitado 21-05-2025).

[7] Arisha N. Rahman Cristopher W. Robinson Andrew M. Hawthorn. "Developmental differences in filtering auditory and visual distractors during visual selective attention". En: *Frontiers in Psychology* (2018).

[8] Wikipedia. *Colavita visual dominance effect*. 2021. URL: https://en.wikipedia.org/wiki/Colavita_visual_dominance_effect (visitado 21-05-2025).

[9] Paul Théberge. *Making music as if the platform depended on it: Issues of genre, practice, automation and design*. 2025. URL: https://www.hf.uio.no/imv/english/research/news-and-events/events/Research-Forum/2025/making-music-as-if-the-platform-depended-on-it-iss.html (visitado 30-05-2025).

[10] Alex A. *Introducción a los sesgos cognitivos: ¿Qué son y por qué importan?* 2024. URL: https://www.aproximadamente.com/introduccion-a-los-sesgos-cognitivos:-que-son-y-por-que-importan/ (visitado 21-05-2025).

[11] Renato Zamora. *La escucha crítica y el entrenamiento auditivo: Claves para un resultado perfecto por Renato Zamora*. 2024. URL: https://www.genelec.lat/-/blog/la-escucha-critica-y-el-entrenamiento-auditivo-claves-para-un-resultado-perfecto-por-renato-zamora (visitado 21-05-2025).

[12] Jakub Staron. *The Halo Effect in UI design*. 2024. URL: https://flyingbisons.com/blog/the-halo-effect-in-ui-design (visitado 21-05-2025).

[13] MixingMonster. *Audio Metering in Mixing and Mastering*. 2025. URL: https://mixingmonster.com/audio-metering/ (visitado 21-05-2025).

[14] Mixing Monster. *Common EQ mistakes in mixing and how to avoid them 2025*. 2025. URL: https://mixingmonster.com/common-eq-mistakes-in-mixing/ (visitado 21-05-2025).

[15] Cristiano Sadun. *Blind testing in practice*. 2021. URL: https://www.theaudioblog.org/post/blind-testing-in-practice (visitado 21-05-2025).

[16] Dídac. *Entrenamiento Auditivo para una Escucha Crítica*. 2025. URL: https://www.masteringbox.com/es/learn/entrenamiento-auditivo-para-ingenieros-de-sonido (visitado 21-05-2025).

[17] Rob Mayzes. *Engineers vs Artists*. 2024. URL: https://mastering.com/engineers-vs-artists/ (visitado 22-05-2025).

[18] Eddy E. *The Art of Mixing and Mastering: Tips for achieving a Professional sound*. 2024. URL: https://www.rmcad.edu/blog/the-art-of-mixing-and-mastering-tips-for-achieving-a-professional-sound/ (visitado 22-05-2025).

[19] Dave Hagen. *The Art and Science of Sound: What is an Audio Engineer?* 2024. URL: https://darkhorseinstitute.com/the-art-and-science-of-sound-what-is-an-audio-engineer/#:~:text=Audio%20engineers%20are%20the%20wizards,and%20hopefully%20emotional%20final%20product. (visitado 22-05-2025).

[20] Matthew Weiss. *In the Studio: an interview with mix engineer Dave Pensado*. 2012. URL: https://www.prosoundweb.com/in-the-studio-an-interview-with-mix-engineer-dave-pensado/ (visitado 22-05-2025).

[21] Mic the Snare. *What everyone misses about tiny desk concerts*. 2024. URL: https://www.youtube.com/watch?v=2NCWckYqh4w (visitado 22-05-2025).

[22] NPR. *Tiny Desk*. 2025. URL: https://www.npr.org/series/tiny-desk-concerts/ (visitado 22-05-2025).

[23] Shachar Gilad. *4 razones por las que deberías contratar a un ingeniero de mezclas profesional*. 2016. URL: https://musicodiy.cdbaby.com/4-razones-por-las-que-deberias-contratar-a-un-ingeniero-de-mezclas-profesional/ (visitado 22-05-2025).

[24] GRIMEY TV. *Entrevista SWAGGERBOYS | Grimey TV 4x26*. 2025. URL: https://www.youtube.com/watch?v=-zy7IU9c_NM (visitado 22-05-2025).

[25] *Glossary of Terms*. URL: https://support.engineears.com/en/knowledge-base/glossary-of-terms#:~:text=Audio%20engineering%20glossary%20provides%20definitions,by%20mixing%20and%20mastering%20engineers. (visitado 22-05-2025).

[26] Jaime Altozano. *Entrevista completa con Rosalía: Motomami por dentro*. 2022. URL: https://www.youtube.com/watch?v=JSGlJ00w8Zw (visitado 22-05-2025).

[27] Image-Line. *FL Studio*. 2025. URL: https://www.image-line.com/fl-studio/ (visitado 21-05-2025).

[28] *FL Studio User Interfaces*. URL: https://www.image-line.com/fl-studio-learning/fl-studio-online-manual/.

[29] Ableton. *Ableton Live*. 2025. URL: https://www.ableton.com/ (visitado 23-05-2025).

[30] Ableton. *Manual Ableton Live*. 2025. URL: https://www.scribbr.com/citation/generator/folders/1kG8sX8bgRTC4vT36pAzaT/lists/3NMqd1dzeR65cALPZdmtne/ (visitado 23-05-2025).

[31] Cockos. *Reaper*. 2025. URL: https://www.reaper.fm/ (visitado 23-05-2025).

[32] Cockos. *Manual Reaper*. 2025. URL: https://dlz.reaper.fm/userguide/ReaperUserGuide737c.pdf (visitado 23-05-2025).

[33] Apple. *Logic Pro*. 2025. URL: https://www.apple.com/es/logic-pro/ (visitado 23-05-2025).

[34] Avid. *Pro Tools*. 2025. URL: https://www.avid.com/pro-tools (visitado 23-05-2025).

[35] *C++*. 2025. URL: https://es.wikipedia.org/wiki/C%2B%2B (visitado 26-05-2025).

[36] *JUCE*. 2023. URL: https://juce.com/ (visitado 26-05-2025).

[37] *Tracktion*. 2025. URL: https://www.tracktion.com/ (visitado 26-05-2025).

[38] *The Audio Developer Conference*. 2025. URL: https://audio.dev/ (visitado 26-05-2025).

[39] Michael Goodwin. *¿Qué es una API (interfaz de programación de aplicaciones)?* 2025. URL: https://www.ibm.com/es-es/think/topics/api (visitado 26-05-2025).

[40] *DirectSound*. 2025. URL: https://es.wikipedia.org/wiki/DirectSound (visitado 26-05-2025).

[41] *DirectX*. 2025. URL: https://es.wikipedia.org/wiki/DirectX (visitado 26-05-2025).

[42] *WASAPI*. 2023. URL: https://learn.microsoft.com/es-es/windows/win32/coreaudio/wasapi (visitado 26-05-2025).

[43] *ASIO*. 2025. URL: https://es.wikipedia.org/wiki/Audio_Stream_Input/Output (visitado 26-05-2025).

[44] *Steinberg*. 2025. URL: https://www.steinberg.net/es/ (visitado 26-05-2025).

[45] *VST*. 2025. URL: https://es.wikipedia.org/wiki/Virtual_Studio_Technology (visitado 26-05-2025).

[46] *VST3*. 2025. URL: https://www.martinic.com/es/blog/vst3-audio-plugin-standard (visitado 26-05-2025).

[47] *How can I get the SDK for VST Legacy?* 2023. URL: https://forum.juce.com/t/how-can-i-get-the-sdk-for-vst-legacy/57662 (visitado 26-05-2025).

[48] *Main benefits of VST 3*. 2025. URL: https://steinbergmedia.github.io/vst3_dev_portal/pages/Main+benefits+of+VST+3/Index.html (visitado 26-05-2025).

[49] *AU*. 2025. URL: https://es.wikipedia.org/wiki/Audio_Units (visitado 26-05-2025).

[50] *Apple*. 2025. URL: https://www.apple.com/es/ (visitado 26-05-2025).

[51] *Core Audio*. 2025. URL: https://es.wikipedia.org/wiki/Core_Audio (visitado 26-05-2025).

[52] *GarageBand*. 2025. URL: https://www.apple.com/es/mac/garageband/ (visitado 26-05-2025).

[53] Jordan Russell. *Inno Setup*. 2025. URL: https://jrsoftware.org/isinfo.php (visitado 27-05-2025).

[54] *Pascal Script*. 2025. URL: https://es.wikipedia.org/wiki/Pascal_Script (visitado 27-05-2025).

[55] Sandra Garrido Sotomayor. *Metodologías ágiles: ¿Qué son y cuáles son más utilizadas?* 2024. URL: https://www.iebschool.com/hub/que-son-metodologias-agiles-agile-scrum/ (visitado 27-05-2025).

[56] *Manifiesto por el Desarrollo Ágil de Software*. 2001. URL: https://agilemanifesto.org/iso/es/manifesto.html (visitado 27-05-2025).

[57] *Scrum (desarrollo de software)*. 2025. URL: https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software) (visitado 27-05-2025).

[58] *Metodología Kanban | Kanban Tool*. 2025. URL: https://kanbantool.com/es/metodologia-kanban (visitado 27-05-2025).

[59] *Extreme Programming*. 2025. URL: https://en.wikipedia.org/wiki/Extreme_programming (visitado 27-05-2025).

[60] *Audio Squadron*. 2025. URL: https://www.audiosquadron.com/ (visitado 28-05-2025).

[61] ADC - Audio Developer Conference. *Why you shouldn't write a DAW - David Rowland - ADC23*. 2024. URL: https://www.youtube.com/watch?v=GMlnh6_9aTc (visitado 28-05-2025).

[62] *MVC - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN*. 2023. URL: https://developer.mozilla.org/es/docs/Glossary/MVC (visitado 28-05-2025).

[63] *JUCE: Component Class reference*. 2025. URL: https://docs.juce.com/master/classComponent.html (visitado 28-05-2025).

[64] *JUCE: Button Class reference*. 2025. URL: https://docs.juce.com/master/classButton.html (visitado 28-05-2025).

[65] *JUCE: Slider Class reference*. 2025. URL: https://docs.juce.com/master/classSlider.html (visitado 28-05-2025).

[66] *JUCE: ValueTree Class reference*. 2025. URL: https://docs.juce.com/master/classValueTree.html (visitado 28-05-2025).

[67] *Need some understanding on the message thread and async calls*. 2024. URL: https://forum.juce.com/t/need-some-understanding-on-the-message-thread-and-async-calls/61573 (visitado 02-06-2025).

[68] *JUCE: AudioAppComponent Class Reference*. 2025. URL: https://docs.juce.com/master/classAudioAppComponent.html (visitado 02-06-2025).

[69] *JUCE: AudioSource Class Reference*. 2025. URL: https://docs.juce.com/master/classAudioSource.html (visitado 02-06-2025).

[70] *Which Buffer Size Setting Should I Use in My DAW?* 2022. URL: https://www.sweetwater.com/sweetcare/articles/which-buffer-size-setting-should-i-use-in-my-daw/ (visitado 02-06-2025).

[71] Tobias K. Tobias. *What is sample rate?* 2023. URL: https://www.lewitt-audio.com/blog/what-sample-rate (visitado 02-06-2025).

[72] *JUCE: PositionableAudioSource Class Reference*. 2025. URL: https://docs.juce.com/master/classPositionableAudioSource.html (visitado 02-06-2025).

[73] Alexander Shvets. *Decorator*. 2025. URL: https://refactoring.guru/design-patterns/decorator (visitado 02-06-2025).

[74] *JUCE: ResamplingAudioSource Class Reference*. 2025. URL: https://docs.juce.com/master/classResamplingAudioSource.html (visitado 02-06-2025).

[75] *JUCE: ValueTree Class Reference*. 2025. URL: https://docs.juce.com/master/classValueTree.html (visitado 03-06-2025).

[76] *David Rowland - Using JUCE value trees and modern C++ to build large scale applications (ADC'17)*. 2017. URL: https://www.youtube.com/watch?v=3IaMjH5lBEY (visitado 03-06-2025).

[77] *Tree (abstract data type)*. 2025. URL: https://en.wikipedia.org/wiki/Tree_(abstract_data_type) (visitado 03-06-2025).

[78] *The ValueTree class - JUCE Tutorial - JUCE*. 2025. URL: https://juce.com/tutorials/tutorial_value_tree/ (visitado 03-06-2025).

[79] Alexander Shvets. *Observer*. 2025. URL: https://refactoring.guru/design-patterns/observer (visitado 03-06-2025).

[80] *JUCE: ValueTree::Listener Class Reference*. 2025. URL: https://docs.juce.com/master/classValueTree_1_1Listener.html (visitado 03-06-2025).

[81] Alexander Shvets. *Singleton*. 2025. URL: https://refactoring.guru/design-patterns/singleton (visitado 03-06-2025).

[82] *JUCE: AudioFormatWritter Class Reference*. 2025. URL: https://docs.juce.com/master/classAudioFormatWriter.html (visitado 02-06-2025).

[83] Adobe. *Archivo XML: qué es y cómo abrir un archivo XML*. 2025. URL: https://www.adobe.com/es/acrobat/resources/document-files/text-files/xml-file.html (visitado 02-06-2025).

[84] Adrián Rodriguez. *Mesa de mezclas: descubre todo lo que debes saber sobre ella*. 2024. URL: https://35mm.es/mesa-mezclas-descubre-todo-debes-saber/ (visitado 04-06-2025).

[85] *Tocadiscos*. 2025. URL: https://es.wikipedia.org/wiki/Tocadiscos (visitado 05-06-2025).

[86] Armada Music. *EQ Explained - The Basics*. 2022. URL: https://www.armadamusic.com/university/music-production-articles/eq-explained-the-basics (visitado 05-06-2025).

[87] Icon Radio. *What Are the Different Types of EQ and Filters?* 2024. URL: https://www.iconcollective.edu/types-of-eq (visitado 05-06-2025).

[88] *Reverberación*. 2025. URL: https://es.wikipedia.org/wiki/Reverberaci%C3%B3n (visitado 05-06-2025).

[89] Jaime Altozano. *La Verdad sobre la Música POP | Jaime Altozano (ft. Ter)*. 2018. URL: https://www.youtube.com/watch?v=ySa67e0jKNA (visitado 05-06-2025).

[90] Joe Crow. *Reverb Parameters Explained!* 2020. URL: https://www.joecrowtheaudiopro.com/2020/03/17/reverb-parameters-explained/ (visitado 05-06-2025).

[91] Bisiesto Estudio. *¿Qué es un test de usuarios en UX?* 2023. URL: https://bisiesto.es/que-es-un-test-de-usuarios-en-ux/ (visitado 06-06-2025).

[92] *Download - JUCE*. 2025. URL: https://juce.com/download/ (visitado 09-06-2025).

[93] *IDE de Visual Studio 2022: herramienta de programación para desarrolladores de software*. 2024. URL: https://visualstudio.microsoft.com/es/vs/ (visitado 09-06-2025).

Apéndice A
Manual de Instalación

Este apéndice explica de manera sencilla cómo instalar la aplicación. Distinguimos entre una instalación simplificada para usuarios y una instalación para desarrolladores; esto se debe a que las herramientas necesarias para desarrollar suponen un gran costo en almacenamiento del dispositivo y esto no tendría sentido para un usuario común.

Para acceder al código públicamente, consulta el siguiente repositorio de GitHub: https://github.com/jaiRodRod/EGAW

A.1. Instalación para usuarios

Los medios dispuestos para la instalación simplificada para usuarios están diseñados para el sistema operativo Windows; esto se debe a que el instalador es un archivo ejecutable del tipo ".exe".

Para instalar el software únicamente tendremos que ejecutar el instalador llamado "EGAW Installer", incluido con el código fuente. Una vez lanzado el instalador, simplemente seguimos las instrucciones de instalación y acabaremos teniendo disponible el producto diseñado para los usuarios (ver figura 47).

A.2. Instalación para desarrolladores

Esta instalación, al igual que la anterior, está diseñada para usuarios del sistema operativo Windows. Sin embargo, el proyecto puede ser desplegado para el desarrollo, instalando de la misma forma JUCE y el editor de código correspondiente.

Para la instalación dedicada a seguir desarrollando la aplicación, primero, descargaremos la biblioteca de JUCE desde la página web oficial [92]. Una vez descargado, obtendremos un archivo comprimido con una carpeta con el nombre de la biblioteca. Descomprimiremos esta carpeta en la ruta deseada. Acto seguido, comprobaremos que

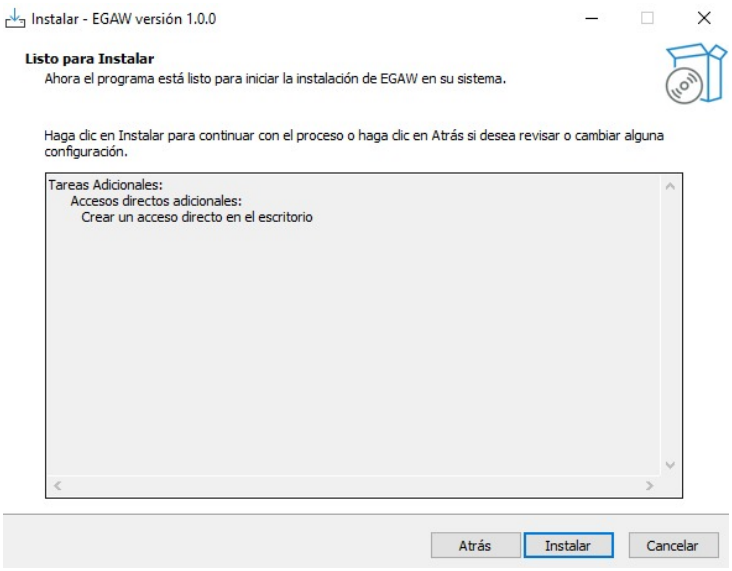


Figura 47: Herramienta diseñada para facilitar la instalación a los usuarios.

podemos lanzar la aplicación ejecutable "Projucer", esta aplicación sirve como gestor de proyectos en el entorno JUCE y se encarga de sincronizar todos los componentes necesarios para desarrollar y compilar el proyecto (ver figura 48).

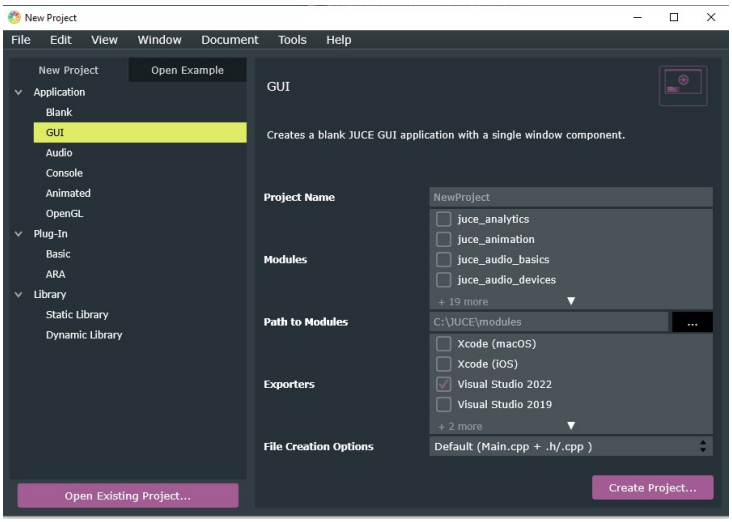


Figura 48: Vista al abrir "Projucer" por primera vez.

Utilizaremos como editor de código Visual Studio 2022 debido a su soporte para el desarrollo de C++ incorporado y la compatibilidad con JUCE. Accedemos a la web oficial del editor de código para descargarlo; en caso de no contar con licencia, podremos descargar la versión comunitaria [93]. Una vez lanzado el instalador, seleccionaremos instalar

el paquete de desarrollo de aplicaciones C++ llamado “Desarrollo para el escritorio con C++” (ver figura 49).

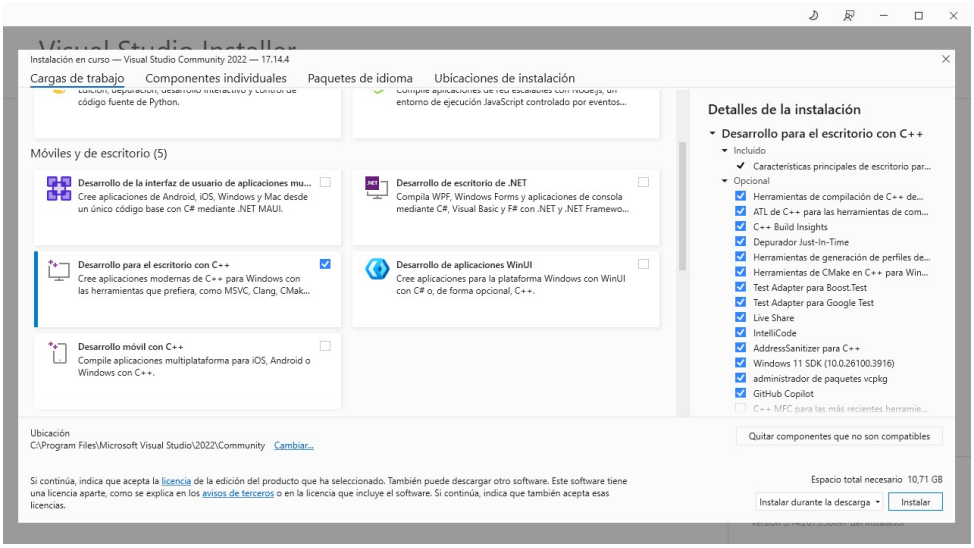


Figura 49: Fase de descarga de Visual Studio 2022.

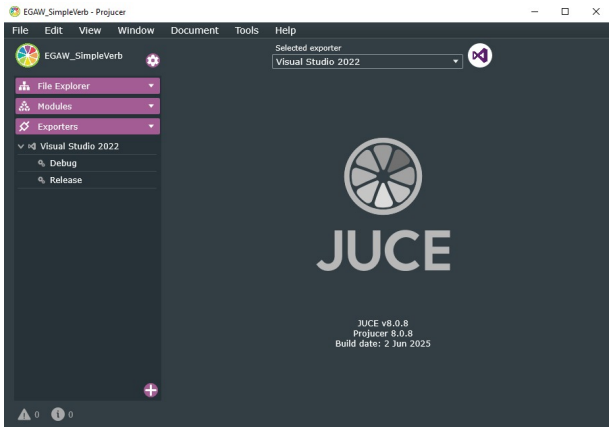


Figura 50: Proyecto de JUCE abierto desde “Projucer”.

Una vez tengamos instalado todo el entorno para el desarrollo de software, descargamos el código fuente de la aplicación. Este código estará disponible en la correspondiente entrega del trabajo. Una vez descomprimido en el directorio deseado el archivo que contiene el código fuente del proyecto, procedemos a abrir “Projucer”. Dentro del gestor, podremos abrir cualquiera de los proyectos disponibles, la aplicación principal o los plugins (ver figura 50). Una vez abierto, tendremos que ubicar las librerías y módulos de JUCE (ver figura 51). Una vez configurado, podremos abrir Visual Studio 2022 desde

la aplicación, dando paso a un editor configurado para compilar adecuadamente nuestro proyecto (ver figura 52).

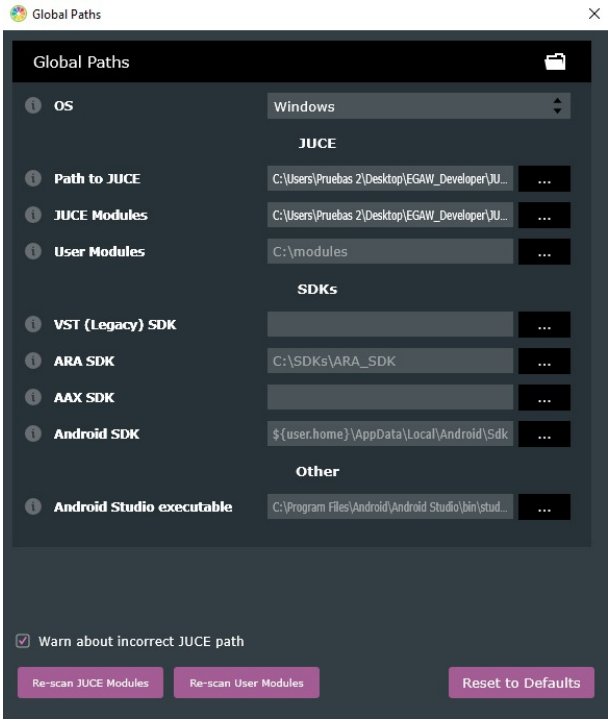


Figura 51: Configurador dentro de “Projucer” para asignar los directorios de las librerías.

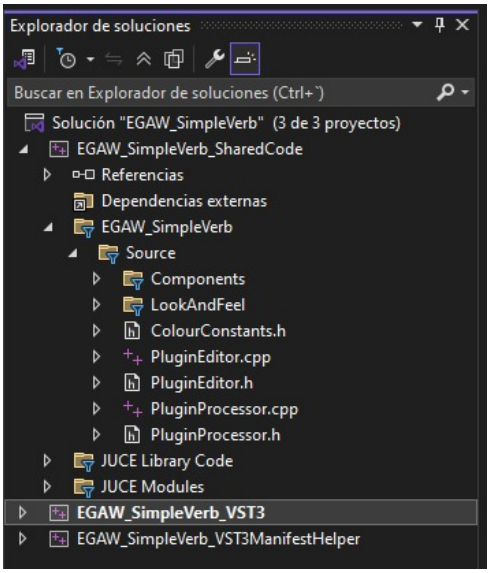


Figura 52: Proyecto desplegado adecuadamente en Visual Studio 2022.

En caso de querer recompilar el instalador simplemente tendremos que abrir el archivo ubicado en la carpeta “InstallerScript” con formato “.iss” usando InnoSetup [53]. Para

actualizar el contenido del software contenido en el instalador podremos sustituir directamente los elementos compilados que se encuentran en el mismo directorio (ver figura 53).

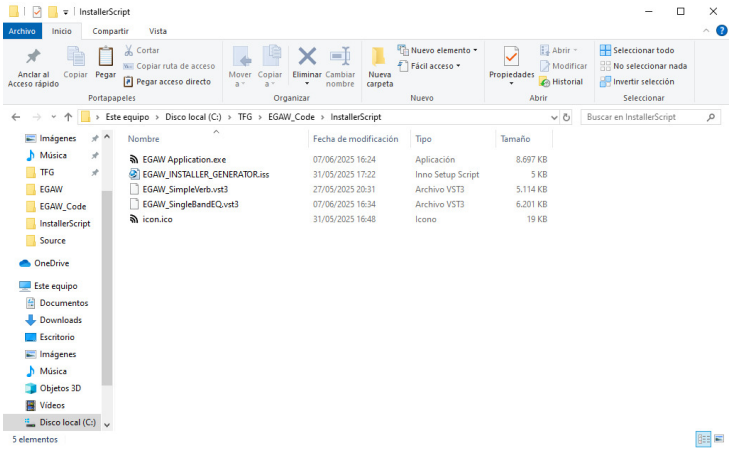


Figura 53: Carpeta incluida en los archivos para poder generar el instalador.

Apéndice B
Manual de Usuario

Este apéndice desarrolla un manual para los usuarios del sistema. En este manual explicamos de manera sencilla las funcionalidades y cómo se disponen en la interfaz.

B.1. Controlador universal

En la parte inferior de la interfaz encontraremos en todo momento un menú disponible para controlar la reproducción del proyecto y para movernos entre las diferentes interfaces. Además, este menú se adapta a la vista, añadiendo un componente de zoom cuando nos encontramos en la lista de reproducción.

Figura 54: Diagrama de componentes del controlador universal.

En la figura 54 encontramos todos los componentes marcados en la interfaz, quedan explicados de la siguiente forma:

- "Play/Pause": este elemento controla el estado del reproductor, reproduce y pausa el proyecto.
- "Stop": Utilizado para pausar por completo el proyecto y rebobinarlo hasta el inicio del mismo.
- Visor de tiempo: este componente nos muestra en todo momento el instante de reproducción en el que nos encontramos.

- Manejador de vista: nos permite ubicar en qué vista nos encontramos y cambiar la interfaz de trabajo a nuestro antojo.

- Controlador de zoom: siendo un componente exclusivo de la lista de reproducción, permite al usuario ajustar el zoom con el que se visualizan los bloques de audio.

Contamos con atajos de teclado para estas funciones. La barra espaciadora nos permitirá controlar "Play/Pause" y utilizar la combinación control y barra espaciadora acciona la función "Stop". Las teclas de funcionalidad "F1", "F2" y "F3" nos permiten movernos entre el mezclador, la lista de reproducción y la pantalla de audición, respectivamente. Por último, combinar control junto con las teclas "más" o "menos" accionará subir o bajar el zoom.

B.2. Menú de ficheros

Este menú se encontrará siempre en la parte superior de la pantalla, en él visualizamos cuatro opciones distintas (ver figura 55), siendo sus funcionalidades las siguientes:

- "Load": permite al usuario cargar un proyecto guardado previamente.

- "Save file": guarda el proyecto actual. Si el proyecto no ha sido cargado de un guardado anterior o no ha sido guardado previamente, la acción requerirá al usuario a darle nombre y ubicación al archivo de guardado.

- "Save file as...": guardará el proyecto actual bajo el nombre y en la ubicación que el usuario desee.

- "Render": comienza el proceso de exportación del proyecto actual a un archivo de audio único.

Encontramos algunos atajos relacionados con este menú. "Control" y "L" para cargar un proyecto. "Control" y "S" para guardar un proyecto. "Control", "Shift" y "S" para guardar como el usuario desee. "Control" y "R" para accionar la exportación del proyecto.

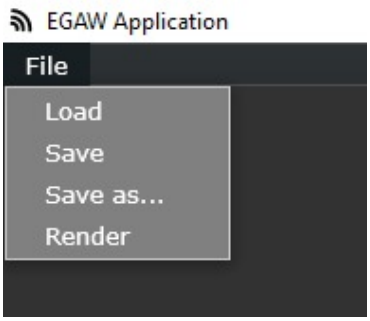


Figura 55: Menú desplegado para la gestión de archivos.

B.3. Mezclador

El mezclador permite al usuario acceder a los canales y realizar la mezcla simulando una mesa analógica. Esta interfaz se descompone en piezas y entender esto será clave en el entendimiento del manual.

B.3.1. Tipos de canales

Los canales son el contenedor a más alto nivel dentro del mezclador. Encontramos cuatro canales, siendo uno de ellos único e insustituible (ver figura 56). Cada canal se diseña con un propósito para la mezcla. Los canales trabajan el procesamiento verticalmente, es decir, la señal se procesa de arriba hacia abajo por cada módulo. La explicación de los tipos de canales es la siguiente:

- Máster: este canal siempre se encontrará a la izquierda de la interfaz y representa la salida maestra del programa. Toda señal que llegue hasta este canal será procesada adecuadamente para que forme parte de la salida audible.
- Canal de audio: aquí podremos cargar archivos de audio. Será la entrada de audio desde fuera del entorno para los proyectos.
- Canal de mezcla: permite unificar múltiples señales en una sola. Este canal permite recoger las salidas de audio proporcionadas por varios canales y unificarlas en una sola salida.
- Canal de “plugins”: este canal es el más especial de todos en el procesamiento de la señal. Nos permitirá alojar sistemas externos para trabajar el audio. Consta de

cinco huecos donde poder incluir procesadores que contarán con una interfaz propia dentro del canal. Particularmente este canal únicamente acoge una sola señal.

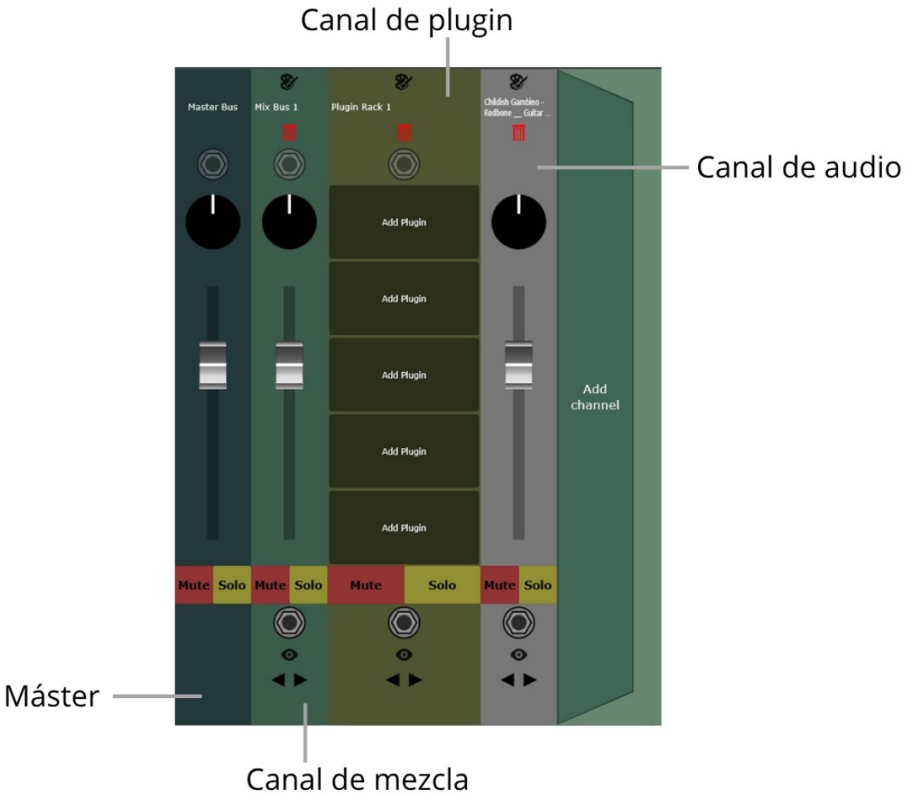


Figura 56: Diagrama del mezclador con los diferentes tipos de canales.

Los atajos relacionados con la creación de los canales son: “Alt” y “A” para añadir un canal de audio, “Alt” y “M” para crear un canal de mezcla, “Alt” y “P” para dar lugar a un nuevo canal de “plugins”.

B.3.2. Controladores y acciones

Los canales alojan controladores que nos permiten accionar los sistemas referidos a cada canal. Tendremos una gran variedad de componentes (ver figura 57). Las funcionalidades que manejan cada controlador son:

- Selector de color: este botón lanza el menú de selección de colores (ver figura 58). Este menú permite seleccionar el color del que queramos que se vean los canales.

- Editor de nombre del canal: nos permite designar un nombre reconocible a cada canal, facilitando el flujo de trabajo.
- Botón de destrucción: borra la instancia del canal deseado y todas sus posibles conexiones.
- Entrada de la señal: este control permanece inactivo mientras no se haya inicializado una acción relacionada con el enrutamiento. Nos permitirá cerrar este tipo de acciones, es decir, una vez comenzamos una acción relacionada con el envío de la señal entre canales, la terminaremos en este elemento del canal deseado.
- Controlador de la imagen estéreo: establece el balance de la señal, permitiendo darle más peso a la izquierda o a la derecha. Haciendo doble clic reestableceremos el valor central.
- Manejador de volumen: controla el volumen con el que se reproduce la señal. Haciendo doble clic podremos reestablecer el valor inicial.
- "Mute": este botón acciona un estado de silenciado en el que la señal emitida por el canal es nula.
- "Solo": añade el canal al grupo de "Solo" activo. Esta funcionalidad es grupal; mientras que ningún canal tenga este estado activo no pasará nada, una vez algún canal entre en este modo las señales "Solo" tienen el control del mezclador, en tal caso, solo los canales con el estado activo emiten la señal.
- Salida de la señal: representando una salida analógica de la señal, permitirá iniciar las acciones de enrutado que tendrán fin en la entrada de la señal de otro canal. Usando clic izquierdo iniciamos un enrutado y usando clic derecho comenzamos el proceso de eliminación de una ruta.
- Visor de las rutas: permitirá activar un estado de visualización en el que podremos ver las rutas del canal seleccionado.
- Controlador de posición: acciona el movimiento del canal, de esta forma el usuario podrá posicionarlo donde desee de acuerdo a su flujo de trabajo personal.

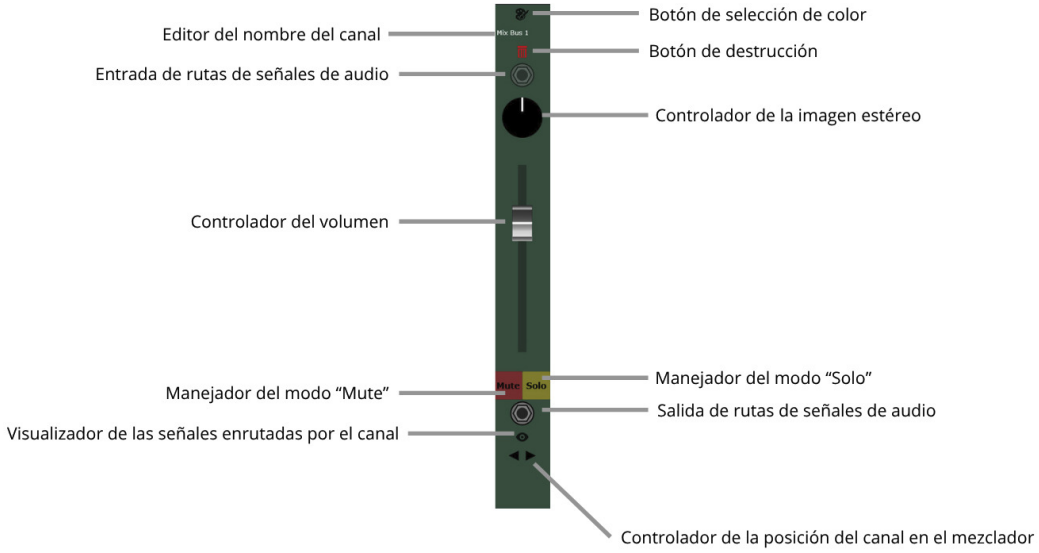


Figura 57: Diagrama de un canal de mezcla con los componentes marcados.

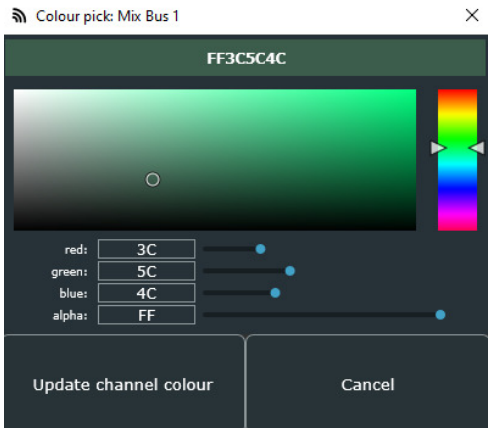


Figura 58: Selector del color del canal desplegado.

Además, contamos con la interfaz propia que nos permite manejar algunas funcionalidades básicas sobre los "plugins" (ver figura 59). Las funcionalidades asociadas a estos controles son:

- Acceso al "plugin": abrirá en una ventana individual el "plugin" para poder editarlo de forma sencilla.
- Control de posición: afecta a la posición vertical del efecto. Esto también repercutirá en el orden en el que se procesa la señal, siendo el efecto más alto el primero en tratar el sonido y el más bajo el último.

- "Bypass": este botón simboliza el estado "Bypass". Mientras esté activo la señal pasará por el bloque sin ser procesada por el efecto.
- Botón de destrucción: servirá al usuario para eliminar el efecto del hueco que ocupa.

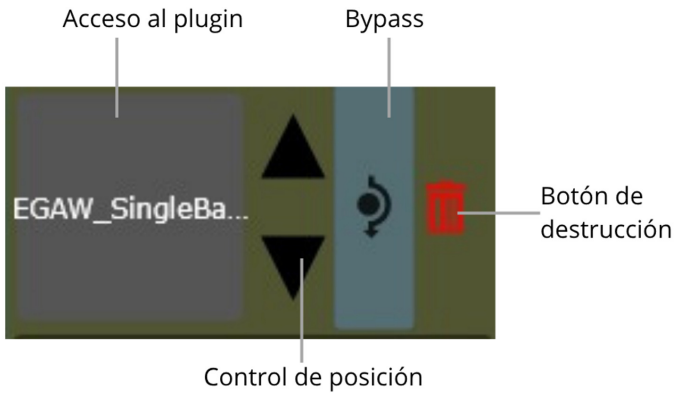


Figura 59: Diagrama del controlador desplegado cuando se añade un plugin.

B.4. Pantalla de audición

Esta interfaz es la más sencilla de todas puesto que realmente no cuenta con ninguna interacción. La vista está ideada para eso. Esta sección del sistema está dedicada a que, como usuario, preste atención al proyecto trabajado y no se vea distraído por estímulos visuales.

B.5. Lista de reproducción

Esta vista será útil para organizar las pistas de audio en el tiempo. Únicamente contaremos con los elementos relacionados con los canales de audio. Podemos dividir la vista en controles del canal, sección de bloques de audio y controlador de tiempo (ver figura 60).

B.5.1. Controladores del canal

Este controlador de canal funciona sincronizado con el mezclador, es decir, las acciones aplicadas en uno se manifestarán también en el otro. En esta versión del canal

Figura 60: Diagrama de las secciones de la lista de reproducción.

tenemos controles reducidos con las mismas simbologías y funcionalidades, por lo que, para profundizar en estos aspectos, podrá consultar la sección previa relacionada con los controladores del mezclador.

B.5.2. Funcionalidades del sistema de bloques de audio

Un bloque de audio en la interfaz representa el archivo de audio dispuesto en el tiempo. Contamos con varias acciones sobre estos bloques. La acción más sencilla se trata de mover el audio en el espacio temporal del proyecto; se acciona arrastrando el bloque horizontalmente. Podemos acceder a un conjunto de acciones mediante el clic derecho sobre el clip de audio (ver figura 61). En este menú contamos con las siguientes posibilidades:

Figura 61: Menú desplegado sobre las acciones del bloque de audio.

- Mostrar onda de audio: permite visualizar la onda de audio. Por defecto, esta opción no estará activa.
- Ajustar longitud del audio: permite desplegar un menú donde poder reajustar el instante de inicio y de fin del archivo de audio en segundos (ver figura 62).

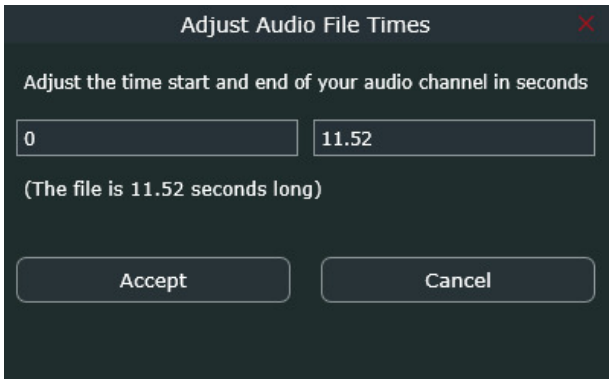


Figura 62: Interfaz del menú de ajuste de la longitud de un bloque de audio.

B.5.3. Manejo del tiempo

La barra controladora posicionada en la parte superior de la interfaz nos permite visualizar y controlar el instante de reproducción del proyecto. Podremos visualmente percibir el momento en el que nos encontramos y manejarlo, ya sea arrastrando la aguja que marca el instante de tiempo o haciendo doble clic en el instante deseado.

B.6. Plugins

Junto con el software principal explicado previamente vendrán instaladas algunas herramientas para modular la señal de audio. Estas herramientas son desplegables utilizando el canal de "plugins".

B.6.1. SingleBandEQ

Esta herramienta es un ecualizador de una sola banda. Esto significa que solo se puede aplicar una función a la señal mediante este "plugin". Encontramos las siguientes funcionalidades en la interfaz (ver figura 63):

- Manejo de volumen: controla el volumen aplicado en caso de que la función incorpore este parámetro para el funcionamiento.
- Controlador de frecuencia: permite definir la frecuencia en la que se aplica la función seleccionada.
- Selector de función aplicada: nos permite seleccionar la función que actuará sobre la señal.

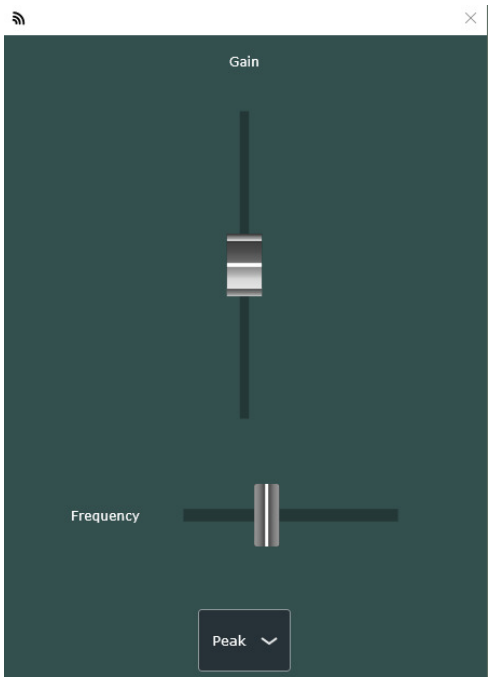


Figura 63: Interfaz del plugin “SingleBandEQ”.

B.6.2. SimpleVerb

Esta herramienta aplica el efecto de reverberación a la señal. Este efecto está basado en la reverberación física del sonido. Las funcionalidades que encontramos en la interfaz son (ver figura 64):

- Nivel de “Dry”: ajusta la cantidad de señal “seca” o sin efecto que saldrá del procesador.
- Nivel de “Wet”: permite seleccionar la cantidad de señal afectada por el efecto que saldrá del procesador.

- “Damping”: este parámetro refleja la dureza de las paredes de una habitación, afecta a las frecuencias agudas procesadas.
- “Room size”: simulando el efecto físico, el tamaño de la habitación afecta a la duración de la reverberación.
- “Width”: controla el ensanchamiento estéreo provocado por el efecto.

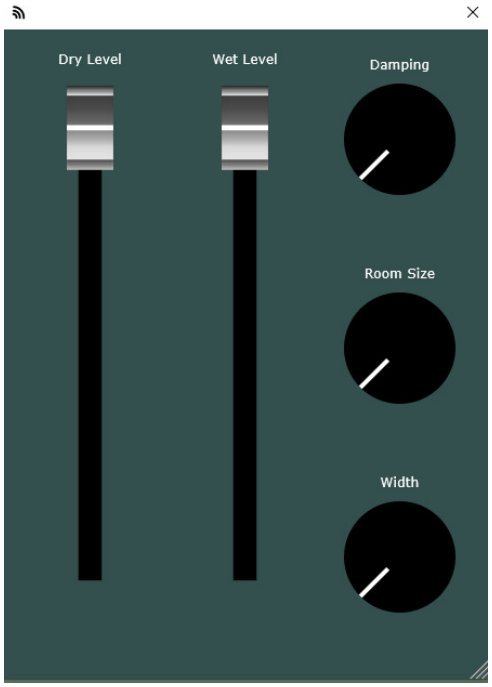


Figura 64: Interfaz del plugin “SimpleVerb”.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA