UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Desarrollo de una aplicación web para el entrenamiento del oído musical

Trabajo Fin de Grado

Grado en Tecnología Digital y Multimedia

AUTOR/A: Cholbi Escrivá, Pere

Tutor/a: Ferrer Contreras, Miguel

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

Universitat Politècnica de València

Escuela técnica Superior de Ingeniería de Telecomunicación

Desarrollo de una aplicación web para el entrenamiento del oído musical

Trabajo fin de grado

Grado en Tecnología Digital y Multimedia

AUTOR/A: Pere Cholbi Escrivà

TUTOR/A: Miguel Ferrer Contreras

CURSO ACADÉMICO: 2022/2023

RESUMEN

Título: Desarrollo de una aplicación web para el entrenamiento del oído musical.

Resumen: El dictado musical consiste en escribir los sonidos de una melodía mientras esta está sonando. Realizar esta tarea implica tener un oído entrenado para poder detectar la diferencia de tonalidad entre las distintas notas de una escala musical. Es decir, se debe saber determinar el intervalo musical existente entre dos sonidos.

En este trabajo se propone una aplicación Web que sirve de herramienta para poder entrenar nuestro oído a la par que resulta entretenida como si de un juego se tratara. Dicha aplicación hace uso de la API WebAudio y del lenguaje de diseño gráfico CSS. La aplicación web permitirá elegir entre ejercicios de diferente dificultad, desde determinar los intervalos musicales que están sonando hasta reconocer acordes musicales. La aplicación implementará un sintetizador de teclado de piano que el usuario podrá usar para comprobar su agilidad auditiva o repetir las notas reproducidas en los ejercicios. La aplicación determinará el acierto del usuario en la detección de los sonidos y emitirá puntuaciones que valoran la destreza de cada usuario en esta materia y servirán para determinar su progreso.

Por tanto, se proponer el desarrollo de una aplicación con una finalidad didáctica que puede servir de ayuda, sobre todo, a los estudiantes de música que necesiten desarrollar y practicar su habilidad para diferenciar y detectar sonidos musicales.

La temática de este TFG surge por el interés del alumno propuesto para la realización del mismo tanto en la música como en el empleo de tecnologías Web.

RESUM

Títol: Desenvolupament d'una aplicació web per a l'entrenament de l'oïda musical.

Resum: El dictat musical consisteix en escriure els sons d'una melodia mentre esta està sonant. Realitzar aquesta tasca implica tindre la capacitat per poder detectar la diferencia de tonalitat entre les diferents notes d'una escala musical. Es a dir, es deu saber determinar l'interval musical existent entre dos sons.

En aquest treball es proposa una aplicació Web que es te la funció de ser una ferramenta per poder entrenar la nostra orella a la vegada que resulta entretinguda com si d'un joc es tractara. L'aplicació fa us de la API WebAudio i del llenguatge de disseny gràfic CSS. L'aplicació web permetrà triar entre exercicis de diferent dificultat, des de determinar els intervals musicals que estan sonant fins reconèixer acords musicals. L'aplicació implementarà un sintetitzador de teclat de piano que l'usuari podrà utilitzar per comprovar la seua agilitat auditiva o repetir les notes reproduïdes en els exercicis. L'aplicació determinarà si les respostes de l'usuari en la detecció dels sons i emetrà puntuacions que valoren la destresa de cada usuari en esta matèria i que serviran per determinar el seu progrés.

Per tant, es proposa el desenvolupament d'una aplicació amb una finalitat didàctica que pot servir d'ajuda, sobre tot, als estudiants de música que necessiten desenvolupar i practicar la seua habilitat per diferenciar i detectar sons musicals.

La temàtica d'aquest TFG sorgeix per l'interès de l'alumne proposat per la realització del mateix tant en la música com en l'ús de tecnologies Web.

ABSTRACT

Title: Development of a web application for musical ear training.

Summary: Musical dictation consists of writing the sounds of a melody while it is playing. Performing this task involves having our ear trained to be able to detect the difference in tonality between the different notes of a musical scale. This means that you must know how to determine the musical interval between two sounds.

This paper proposes a Web application that serves as a tool to train our ear while being entertaining as if it was a game. This application makes use of the WebAudio API and the CSS graphic design language. The web application will allow you to choose between exercises of different difficulty, from determining the musical intervals that are playing to recognizing musical chords. The app will implement a piano keyboard synthesizer that the user can use to check their auditory agility or repeat the notes played in the exercises. The application will determine the success of the user in the detection of sounds and will issue scores that assess the skill of each user in this matter and will serve to determine their progress.

Therefore, we propose the development of an application with a didactic purpose that can help, above all, music students who need to develop and practice their ability to differentiate and detect musical sounds.

The theme of this TFG arises from the interest of the student proposed for the realization of the same both in music and in the use of Web technologies.

ÍNDICE

1. INTRODUCCIÓN

Desde poco después de la aparición de los primeros lenguajes de programación existe el desarrollo web, el cual nos permite crear las diferentes páginas que aparecerán cuando introduzcamos una determinada URL en el buscador. Este despegó gracias a la aparición de PHP en 1995 y la de Javascript en 1998. Con más de dos décadas a sus espaldas, estas tecnologías han experimentado grandes avances y hoy en día permiten aplicaciones mucho más complejas que las desarrolladas a finales de los noventa o a principios de los dos mil, hace ya más de veinte años.

Figura 1: logo de PHP

Figura 2: logo de Javascript

Hoy en día se pueden construir gran cantidad de páginas web con amplias funcionalidades desde el navegador, sin necesidad de instalar ningún tipo de Software adicional en nuestro equipo. Estas aplicaciones pueden ser muy variadas, pudiendo ser muy diferentes entre sí

visualmente y estar basadas en diferentes tecnologías como puedan ser gráficos 3D, el uso de los inputs del usuario, páginas web convencionales con texto HTML o el tratamiento y uso de video o de audio.

Precisamente, el enfoque del trabajo final de grado propuesto se centra en esta última categoría y aprovecha HTML, CSS y Javascript junto con la Web Audio API (interfaz de programación de aplicaciones). Esta API brinda la capacidad de generar, modificar y reproducir sonidos directamente en el navegador, cosa que permite crear experiencias auditivas interactivas. Por tanto, en este Trabajo Final de Grado se desarrolla una aplicación web basada en el uso de la API para crear una página web capaz de generar y reproducir intervalos y escalas musicales haciendo uso del ratón y del teclado como si de un instrumento musical se tratase. Asimismo, el objetivo de la misma es entrenar el oído mediante unos ejercicios musicales en los que el usuario deberá introducir por medio de la aplicación las notas que escuche.

La aplicación aprovecha tanto las capacidades de la Web Audio API para generar y modificar sonido, como el poder crear audio en base a los inputs del usuario, todo esto para crear una aplicación con una finalidad didáctica que puede ser aprovechada por cualquiera que esté interesado en el mundo de la música para aprender a reconocer las diferentes notas, escalas o intervalos.

2. OBJETIVOS

El objetivo de este proyecto es crear una aplicación didáctica para así ayudar al usuario a entrenar el oído musical a la vez que distinguir entre los diferentes intervalos y escalas. Esto es muy útil para cualquiera que esté dentro del mundo de la música ya que la aplicación propone diversos ejercicios de dictado que se podrían hacer en una clase convencional empleando un piano o algún otro instrumento.

Para ello, la aplicación debe ser capaz de reproducir notas y escalas musicales, cada nota con su frecuencia exacta. Asimismo, también debe poder reproducir las notas que decida el usuario mediante los inputs del ordenador y ser capaz de comparar las notas generadas por el usuario con las predefinidas para así valorar la habilidad auditiva de cada usuario en cada ejercicio. Cabe mencionar que todos los sonidos reproducidos por la aplicación serán generados por la misma, pudiendo así prescindir del uso de bibliotecas de audio. En resumen, el objetivo principal es implementar una serie de ejercicios basados en el dictado musical para así entrenar el oído y aprender las distancias entre los diferentes tonos y las armaduras de las diferentes escalas.

Por tanto, debemos crear una aplicación que combine conocimientos de programación en Javascript con conocimiento básico de música, para ser capaz de trabajar con notas, acordes y escalas musicales, sabiendo diferenciar entre sus diferentes tipos. Esto es debido a que un objetivo de los ejercicios propuestos en la aplicación web es saber las diferentes distancias (en semitonos) entre las notas de un intervalo musical o saber distinguir entre una escala mayor y otra menor, cosas que requieren fundamentos básicos de música.

Por otra parte, otro objetivo de la aplicación es proporcionar al usuario la opción de personalizar el sonido de la misma, modificando una serie de parámetros relacionados con la curva ADSR o el poder cambiar la forma de onda del sonido, para así añadir una capa de ajustes individualizados a la aplicación.

Quitando los ejercicios, otro de los objetivos del proyecto es dar al usuario la posibilidad de hacer sonar las diferentes notas de forma libre, para que así pueda experimentar por su cuenta los diferentes sonidos y los diferentes intervalos que aparecen al tocar varias notas al mismo tiempo.

Es de vital importancia señalar que todo esto se debe conseguir desde una aplicación web y no por una de escritorio. Por tanto, todo el tratamiento de audio que contenga la aplicación debe provenir de Javascript y no de una aplicación externa como pueda ser un motor de audio.

3. ESTADO DEL ARTE

El uso de distintas formas audio en aplicaciones web no es nada nuevo y lleva implementando desde 1996, hace ya casi treinta años. Cabe aclarar que no es lo mismo el audio de las aplicaciones web y las de escritorio ya que son dos tipos de aplicaciones distintas. Por tanto, la principal diferencia entre estas dos radica en que mientras que las de escritorio tienen acceso a motores de sonido con muchas más posibilidades y capacidades, las aplicaciones web están mucho más limitadas teniendo que depender en la mayoría de casos de los recursos del propio navegador, los cuales no son tan potentes y no dan juego a aplicaciones igual de complejas como las de escritorio.

En cuanto al sonido en los navegadores, la primera manera de introducir audio en una página web fue gracias a Microsoft en 1996 mediante la etiqueta html <bgsound>. Esta instrucción se puede seguir usando hoy en día, aunque no es recomendable ya que solo nos permite reproducir un audio en bucle un número de veces deseado al cargar una página. Por tanto, no nos permite editar el sonido de ninguna manera ni aplicarle ningún tipo de efecto. Por si esto fuese poco, solo está disponible en el navegador Internet Explorer, lo cual hace que su uso sea todavía más anecdótico. Fue (y sigue siendo), una manera primitiva de introducir audio en la web, la cual ha quedado completamente desfasada pero que tuvo cierta importancia en su día al sentar un precedente.

Más adelante apareció Flash como opción para poder incluir sonido de forma embebida en las aplicaciones web, pero tenía el inconveniente de que debía de estar instalado en nuestro ordenador para que funcionase, y por tanto no dependía solamente de la web, limitando el alcance que tenían las aplicaciones solo a los equipos con Flash instalado[1][2].

Figura 3: logo de adobe Flash

Aun con todo, la ventaja fue que flash sí que daba pie a aplicaciones más complejas que permitían dar más opciones a la hora de editar audio y por tanto era una mejor opción que la etiqueta <bgsound>.

Después con la llegada de HTML5 en 2008, llegó la etiqueta <audio>[3], la cual venía a facilitar el uso de audio en la web. El problema que surgió fue que en por aquel entonces no había un códec estándar para todos los navegadores, lo que hacía que para que funcionase el audio se tuviese que importar este en muchos codecs distintos, lo cual hacía que el proceso de incluir audio en una aplicación fuese muy engorroso, dificultando así su implementación. Finalmente, en 2015, HTML permite que con solo incluir un fichero en MP3, este funcione en todos los navegadores. Aún con todo, la etiqueta <audio> es bastante limitada, ya que no permite editar el sonido de forma avanzada cambiando parámetros como puedan ser su frecuencia de muestreo. Simplemente tiene algunas opciones para gestionar la reproducción del sonido y el buffer (pausar sonido, rebobinar, controlar el volumen) pero no nos deja editar sonido como tal.

Figura 4:ejemplo de uso de la etiqueta audio

Es aquí donde llegamos a la Web Audio API[4][5], una API de Javascript cuya primera versión apareció en 2011 y que lleva desde entonces recibiendo nuevas versiones constantemente, siendo soportada por la grandísima mayoría de navegadores. Cabe destacar que no fue la única API relacionada con audio que apareció a principios de la década de 2010. Otros ejemplos de APIs serían la WebMidi API, una API para poder usar un teclado MIDI en la web, o la Firefox Audio Data API, una alternativa a la WebAudio API desarrollada por Mozilla, aunque ha sido la Web Audio API la que ha conseguido destacar, ya que esta proporciona lo mismo que un sintetizador pero desde la web, y siendo a su vez compatible con la mayoría de navegadores. Volviendo a la misma, esta tiene un funcionamiento modular que se basa en el uso de diferentes nodos con inputs y outputs por los que viaja el sonido y en los que es modificado hasta que llega al nodo de destino y se reproduce.

Con ella es posible tener aplicaciones que generen, editen y reproduzcan sonidos, dando pie a una infinidad de posibilidades, como puedan ser por ejemplo sintetizadores, aplicaciones que por ejemplo recojan audio de un input y lo editen o lo analicen o aplicaciones que generen sonidos las cuales pueden tener diferentes finalidades, siendo una de ellas la aplicación propuesta.

4. JAVASCRIPT

Javascript es la base del proyecto, ya que es el lenguaje de programación usado para construir la aplicación y el lenguaje al que pertenece la web audio API. Este es, junto a HTML y CSS, uno de los tres lenguajes principales a la hora de desarrollar una página web, siendo el lenguaje destinado a dotar de funcionalidad e interactividad a la aplicación.

Figura 5: los tres principales componentes de una aplicación web.

Antes de su aparición las páginas web eran completamente estáticas, siendo el desarrollo de lenguajes de *scripteo* algo completamente necesario, apareciendo Javascript en 1995 (desarrollado por NetScape) y convirtiéndose con el paso de los años en una parte esencial del desarrollo web y en el lenguaje estándar usado por todos los navegadores[6][7].
Hoy en día el 90% de las webs usan Javascript en el cliente y el 64% de los desarrolladores web trabajan en Javascript, siendo usado de base en una enorme cantidad de librerías, APIs y Frameworks como puedan ser React o Angular.

Javascript es un lenguaje interpretado y no compilado, lo que significa que el código no necesita pasar por un compilador antes de ser ejecutado si no que un intérprete recorre todo el código y lo ejecuta al momento. Cabe destacar que Java si es un lenguaje compilado y es muy diferente a Javascript, ya que, aunque tienen cosas en común, la semblanza entre los dos nombres puede hacer que pensemos que Java y Javascript son iguales o muy parecidos, cosa que no es así ya que Javascript cogió el nombre de Java para ganar notoriedad en sus inicios[8].

Algunas de las características que definen a Javascript son:

- Es un lenguaje ligero: esto es muy útil ya que está hecho para tratar datos en la parte del cliente de una aplicación.
- Escritura dinámica: la escritura dinámica significa que al declarar una variable la puedas almacenar en el tipo de dato que quieras, lo que implica que al definir la variable no tienes que indicar si es por ejemplo un número o una cadena de texto.
- Ejecución individual: Javascript no puede ejecutar varias cosas a la vez, aunque para paliar esto da la opción de usar promesas que permiten ejecutar código de forma asíncrona. Cabe destacar que sí que permite el uso de *Web Workers* como método de ejecución paralela.
- Validación de datos en la parte del cliente: al ser un lenguaje orientado al cliente, Javascript permite hacer validaciones de datos desde el navegador.
- Soporte a la programación orientada a objetos: aunque inicialmente Javascript no era un lenguaje orientado a objetos como pueda ser por ejemplo Java, desde hace ya años da soporte a esta forma de programar, pudiéndose encapsular o heredar código[9].

4.1 HTML Y CSS

Javascript se encarga de darle dinamismo a la web y le proporciona interactividad a la misma pero no es el único pilar a tener en cuenta a la hora de diseñar una aplicación web.

Para la creación de la página web en sí, el texto que vamos a ver en la página, usamos HTML, que es un lenguaje de marcas de hipertexto. Por tanto, HTML es el encargado de crear cualquier elemento que vemos por pantalla[10] y es considerado un lenguaje de marcas porque usa diferentes "etiquetas" entre < > para agrupar el texto en diferentes elementos. Ejemplo de estas etiquetas o marcas son , <header>, <body> etc, las cuales nos permiten que el texto este estructurado por medio de su uso.

Otras características que definen a este lenguaje son, por ejemplo, que no es un lenguaje de programación, ya que no tiene funcionalidades dinámicas, ya que estas vienen dadas por Javascript, estando algunos elementos Javascript relacionados con otros HTML para dotar a estos últimos de funcionalidad. También destaca el uso de hipervínculos para desplazarte por las distintas páginas de la red, ya que una aplicación puede estar formada por muchas páginas HTML.

En cuanto a las versiones, la última es HTML5[11], la cual es usada por las principales páginas web como puedan ser Facebook o Google a la vez que tiene soporte para prácticamente todos los navegadores. También implementa nuevas etiquetas como por ejemplo la etiqueta <embed> para incorporar contenido externo o la etiqueta <audio> mencionada anteriormente, que permite añadir audio, aunque de forma un poco básica.

Asimismo, las aplicaciones también necesitan dotar a estos elementos de unos estilos visuales determinados y una estética, para hacer la página o aplicación más atractiva visualmente. Aquí es donde entra CSS[12], un lenguaje de hojas en cascada que sirve para dotar a las páginas HTML de estilos visuales. CSS es muy importante y permite que un mismo documento tenga varios estilos, para que así, por ejemplo, se pueda visualizar en distintas pantallas con diferentes relaciones de aspecto, como puedan ser una pantalla de un ordenador y la de un teléfono móvil. Esto es posible ya que el código CSS se encuentra separado del de HTML y por tanto no interfiere en el mismo. Es, en definitiva, el responsable de toda la parte visual de nuestra página web.

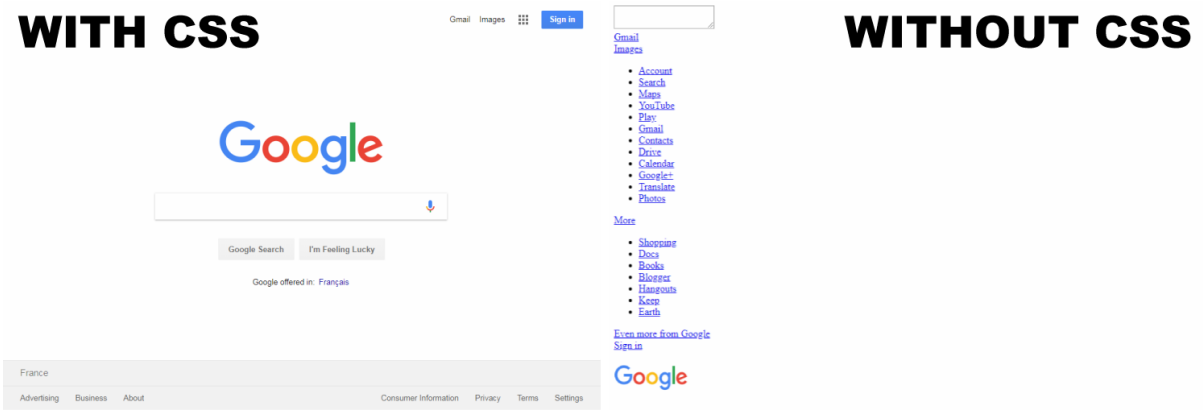

Figura 6: comparativa de como se ve la página principal de Google con y sin CSS.

Entre otras cosas, CSS permite aplicar muchos estilos visuales a nuestros elementos HTML, como por ejemplo cambiar el tamaño, el color o la fuente de los mismos o poner fondos y posicionar correctamente cada elemento.

Su funcionamiento es jerárquico con diferentes niveles, lo que permite definir estilos globales que se apliquen a muchos o a todos los elementos a la vez, y a su vez definir estilos específicos para determinados elementos en concreto. También es destacable que CSS implementa herencia, cosa que facilita la aplicación de diferentes estilos para elementos padre e hijo[13].

En resumen, CSS es una parte fundamental del desarrollo web ya que permite dotar al código HTML de una presentación visual. Asimismo, esto también hace que las páginas sean mucho más intuitivas y fáciles de utilizar, por lo que CSS también es una parte que agrega funcionalidad a la web y no es solo algo estético.

4.2 NODE

Para garantizar un completo funcionamiento de la aplicación, esta también hará uso de un servidor básico de Node.js[14]. Esto permitirá que la aplicación se ejecute en un servidor local en un determinado puerto elegido.

En cuanto a Node en sí, este es un entorno de código abierto que nos permite ejecutar JavaScript en el lado del servidor. Esto es muy diferente al uso principal que se le da a Javascript, el cual es escribir aplicaciones pero del lado del cliente. Por tanto, usando Node podemos obtener una aplicación entera basada en el uso de Javascript, tanto en el cliente como en el servidor.

Figura 7: lista de empresas que emplean Node.js

Asimismo, una característica fundamental es que es un entorno de programación asíncrono[15], permitiéndole esto tratar con múltiples solicitudes sin que esto implique detener la ejecución de otros procesos. La clave para lograr esto es usar funciones async/await, callbacks o promesas, siendo estas instrucciones típicas de Javascript.

Otra característica reseñable de Node a tener en cuenta es su arquitectura basada en el bucle de eventos. Este permite que Node.js tenga un comportamiento eficiente a la hora de manejar solicitudes entrantes al ejecutar determinadas operaciones de manera asíncrona y así, delegar el trabajo pesado a segundo plano, para así garantizar una escalabilidad efectiva y poder gestionar todas las solicitudes.

En resumen, Node permite a los desarrolladores crear aplicaciones del lado del servidor utilizando JavaScript, aprovechando su eficiencia y capacidad para manejar múltiples conexiones simultáneas gracias a su modelo de programación asincrónico y su arquitectura basada en el bucle de eventos.

5. WEB AUDIO API

La Web Audio API es una API de Javascript destinada al tratamiento de audio en el navegador, sin necesidad de utilizar ninguna aplicación de escritorio. Su primera versión se lanzó en 2011 y desde 2021 es el estándar recomendado por la W3C en cuanto a tratamiento de audio en la web[16][17].
Esta API ofrece un gran abanico de posibilidades a la hora incluir audio en la web ya que nos permite desde reproducir audio, hasta editarlo y aplicarle efectos, generarlo y visualizar características sobre el mismo, cosa que permite crear una amplia gama de aplicaciones web que generen o incluyan efectos de sonido, procesamiento de audio, juegos basados en el sonido y un largo etcétera.

La Web audio API funciona de forma modular[18], aunque siempre dentro de un objeto 'AudioContext'. Dentro de este 'AudioContext' se definen los diferentes nodos los cuales aplican diferentes efectos. Los nodos están conectados entre sí formando una cadena desde la fuente del audio, ya sea un audio importado o el propio audio generado por la API hasta el final donde podemos reproducirlo o incluso guardar el sonido generado en un archivo si quisiéramos. En resumen, la API acompaña al sonido en todas sus etapas desde su generación hasta su reproducción, pasando por todas las capas de procesamiento oportunas, mediante el enlazamiento de nodos, teniendo el usuario todo el control sobre el audio, pudiendo procesarlo o personalizarlo como este crea oportuno.

Figura 8: funcionamiento de la Web Audio API

Por tanto, el funcionamiento de la Web Audio API es el siguiente:

- Primero definimos un AudioContext.
- Dentro de este contexto definimos el audio con el que vamos a trabajar, ya sea generado estableciendo diferentes parámetros como puedan ser por ejemplo la frecuencia de muestreo o directamente trabajamos con un archivo de audio importado.
- Después formamos una cadena de nodos en la que cada uno modifica el audio de la forma en que queramos. Por ejemplo, podríamos usar un nodo de ganancia para hacer que se escuche más o un filtro selectivo en banda para que suenen solamente las frecuencias que deseemos.
- A continuación, elegimos un destino para el audio. Aquí es donde lo podríamos guardar en nuestro ordenador o conectar el audio al sistema de audio del pc para poder reproducirlo. Cabe destacar que, para que esto funcione, todos los elementos tienen que estar conectados formando una cadena desde la fuente del sonido hasta su destino final.

5.1 NODOS

Como hemos dicho antes, el funcionamiento de la web audio API se basa en el uso de distintos nodos conectados entre sí, que realizan alguna transformación o tarea sobre el audio para así generarlo, tratarlo o reproducirlo[19].

Los nodos cuentan con los métodos *connect* y *disconnect*, los cuales nos permiten conectar y desconectar el nodo a otros nodos, formando así una cadena por la que pasa el audio. Por norma general, un nodo tiene tanto *input* como *output* aunque hay algunos como los *SourceNode* que solo tienen *input* y pueden tener varios *outputs*, los cuales se suelen usar para generar el sonido. Otro ejemplo serían los *DestinationNode* que son lo contrario y sirven para mandar el audio directamente a los altavoces. Quitando estos tipos, nos quedan los

nodos de procesamiento, los cuales son los más comunes. Estos van entre el nodo *Source* y el *Destination* y sirven para tratar el audio recibiendo un *input* al cual aplican procesos para generar un *output* que pasa a otro nodo como *input*. Cabe destacar que, en aplicaciones complejas, el uso de nodos puede provocar latencia y que ocurra, que, por ejemplo, al darle a un botón para reproducir un sonido, este tarde unos instantes en llegar a los altavoces. Por tanto, es recomendable solo usar los nodos necesarios y así no introducir código redundante.

Algunos tipos de nodos muy usados son los siguientes:

- AnalyserNode: estos son muy útiles en aplicaciones dedicadas a obtener datos sobre un audio ya que permite generar información en tiempo real sobre el audio. El nodo funciona de forma que no modifica el audio, siendo el *input* igual que el *output*. El nodo permite acceder a la información generada, la cual es independiente del *output*.
- Filtros: Los nodos nos permiten implementar distintos tipos de filtros. Para ello podemos usar el *Biquadfilter()* especificando los parámetros del filtrado. Podemos representar varios tipos de filtro y especificar a partir de qué valor de frecuencia corta el filtro.
- BufferNode: permite tratar el audio que está en el buffer. Es muy útil, para, por ejemplo, audios que requieran reproducirse en un instante determinado y para los cuales el retardo producido por cargar el audio desde el disco duro pueda ser un problema.
- GainNode: Estos nodos también son muy importantes ya que son los encargados de controlar el volumen del audio que se esté tratando. Se suele poner el último para que los efectos de volumen sean inmediatos.

5.2 CONTEXTO DE AUDIO

En cuanto al *AudioContext*, este es un elemento indispensable de la API, ya que es la instancia de la API que envuelve los nodos y todos los procesos citados anteriormente,

estando todos estos dentro del propio *AudioContext*[20]. Asimismo, el *AudioContext* proporciona herramientas para gestionar la reproducción o la sincronización del audio. Unos ejemplos serían los métodos *AudioContext.output Latency* o *AudioContext.baseLatency*, los cuales nos permiten ver los diferentes tipos de latencia presentes en nuestra aplicación.

En cuanto a su funcionamiento, debido a que es necesario para el uso de cualquier otro nodo, el *AudioContext* debe ser creado antes de hacer ningún tratamiento de audio. También es aconsejable usar solo un *AudioContext* por aplicación, y reusarlo las veces que haga falta aunque sea con distintos audios, para así ahorrar memoria y trabajo al procesador y también para que no aumente la latencia de la aplicación.

5.3 OSCILADORES

El nodo oscilador que nos proporciona la Web Audio API será una parte vital de nuestra aplicación ya que es la herramienta que vamos a usar para crear todos los sonidos necesarios para hacerla funcionar[21].

El oscilador crea un tono con una forma de onda periódica y con una determinada frecuencia. Los parámetros que recibe un oscilador son los siguientes:

- Frecuency: es obviamente la frecuencia del oscilador, la cual viene dada en Hz. Su valor por defecto es 440, correspondiendo al La4 que se suele usar como nota de referencia en la música.
- Detune: es un parámetro que nos permite que la nota suene un poco desafinada, viene dado en centésimas de semitono y funciona de forma que aplica un desplazamiento a la frecuencia base establecida del valor de este parámetro. Es muy útil para provocar algunos efectos musicales como pueda ser el vibrato. El valor por defecto es 0.

- Type: por último, el parámetro *type* nos permite cambiar la forma de onda del oscilador. Podemos hacer que la forma de onda sea como queramos (*custom*) o elegir entre 4 modos predefinidos siendo estos *sine*, *square*, *triangle* y *sawtooth*. El modo que está por defecto es la onda senoidal.

5.4 UTILIDAD DE LA WEB AUDIO API

Con toda la gama de opciones que ofrece esta API, podemos afirmar que estamos ante un Software que permite hacer gran cantidad de cosas y que por tanto puede tener una infinidad de aplicaciones. Algunos ejemplos serían los siguientes:

- Creación de experiencias interactivas online basadas en música. La API es una herramienta que puede ser explotada por artistas para la creación de sistemas interactivos manejados por potenciales usuarios.
- Videojuegos. La API también puede usarse o bien para crear videojuegos web basados en audio o bien implementar el sonido de videojuegos web en general, pudiendo generar los sonidos por sí misma.
- No debemos obviar la implementación más evidente, que es la capacidad de añadir sonido y efectos personalizados a las diferentes aplicaciones y sitios web en los que se utilice.
- Otra posible aplicación es el tratamiento y procesamiento de audio, el cual es posible gracias a la gran cantidad de nodos, los cuales permiten editar y procesar el audio de muchas formas al mismo tiempo que nos permite aplicar distintos efectos como por ejemplo eco o reverberación.
- Por último, tenemos el uso que le estamos dando a la API en la aplicación propuesta, que es un uso didáctico para aprender usando de base el sonido, ya sea para aprender música como tal o por ejemplo para aprender a distinguir diferentes sonidos.

6. SÍNTESIS

Una parte vital del proyecto consiste en la síntesis de audio para hacer sonar las diferentes notas y sonidos. Por tanto, es necesario explicar en qué consiste la síntesis de audio y las diferentes maneras mediante las que podemos generar sonido en una aplicación.

Por tanto, definimos síntesis como el proceso de crear ondas de sonido mediante un dispositivo electrónico, usando un oscilador que puede ser bien o analógico o digital (digital en nuestro caso). Para formar un sonido se necesita un generador de señal con una forma de onda particular para cada sonido, y para generarlas, usamos sintetizadores. Por tanto, existen diferentes tipos de síntesis, siendo algunos de ellos los siguientes[22]:

- Síntesis sustractiva[23]: este es el método de síntesis más primitivo, desarrollado en las décadas de los 60 y los 70. Consiste en el uso de generadores de señales periódicas ricas en armónicos para crear ondas mediante presión eléctrica, que son alteradas a posteriori para modificar parámetros como su envolvente temporal o envolvente espectral. En este tipo de sintetizadores podemos diferenciar tres componentes esenciales que son los generadores de señal, los modificadores que alteran el sonido, aquí entrarían elementos como filtros o amplificadores, y finalmente los controles mediante los que el usuario puede hacer funcionar el oscilador. Cabe destacar que este método también es llamado síntesis analógica debido a que puede implementarse usando componentes analógicos como sucedió en su origen donde no se usaban componentes digitales.

Figura 9: Ejemplo de sintetizador sustractivo

- Síntesis aditiva: este tipo de síntesis pone el foco en imitar el cómo se forman los sonidos en la vida real. Por tanto, funciona de manera que tenemos cierto número de osciladores cada uno de los cuales crea una onda senoidal con una frecuencia y una amplitud determinadas[24]. A continuación, se suman todas estas ondas para crear una onda que represente un determinado sonido.
- Modulación de frecuencia: esta forma de síntesis guarda similitud tanto con la síntesis aditiva (por el empleo de diferentes osciladores) como con la sustractiva (por el uso de filtros para ecualizar la envolvente espectral) pero con la diferencia de que es más barata y fácil de implementar. Funciona de forma que contamos con varios osciladores (dos o más) que crean ondas senoidales con amplitudes y frecuencias determinadas. La diferencia con la síntesis aditiva radica en la forma de combinar las diferentes ondas senoidales. En vez de combinarlas directamente, la salida del primer oscilador sirve para modular al siguiente, y así constantemente hasta el final. Esta forma de operar permite sintetizar sonidos con muy pocos osciladores (normalmente seis), mientras que con aditiva igual necesitaríamos el doble o el triple de osciladores, siendo este el por qué es un método más barato y más fácil de implementar. Es destacable saber que este método fue el primero en ser exitoso comercialmente, apareciendo en 1983 con el sintetizador Yamaha DX7[25].

Figura 10: Sintetizador Yamaha DX7

- Sample/síntesis: este método se caracteriza por usar muestras de audio pregrabadas como fuente de sonido. Este tipo de síntesis encuentra su utilidad en combinar sonidos ya existentes con otros generados con otros tipos de síntesis.

7. ONDAS

Como hemos descrito anteriormente, podemos cambiar la forma de onda del sonido para que así suene de distintas maneras. Para explicar esto primero tenemos que definir qué es una onda sonora, siendo ésta una onda mecánica o longitudinal que se propaga por un medio elástico, como podrían ser tanto el agua como el aire y que produce variaciones de presión en el medio, que podemos percibir como sonido. Por tanto, las formas de onda representan en un eje el desplazamiento que provoca el sonido en las moléculas del aire por donde se propagan a través del tiempo. A su vez, esta onda es provocada por una vibración, la cual puede ser causada por un altavoz, la cuerda de una guitarra, nuestras cuerdas vocales, etc.

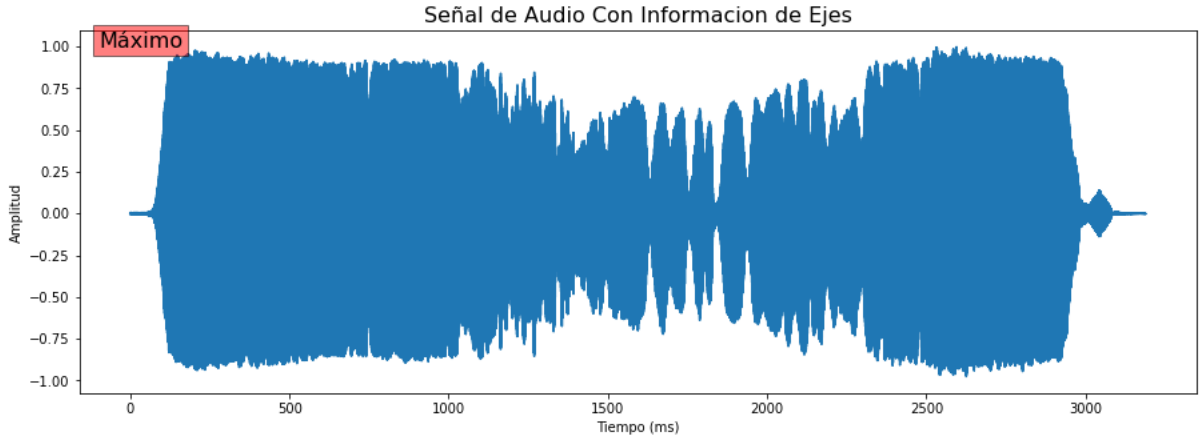

Figura 11: ejemplo de una señal de audio.

Para medir el desplazamiento de las moléculas usamos la amplitud, normalmente normalizada desde 0 hasta 1 que nos indica a su vez el volumen del sonido, aunque cabe señalar que nuestra percepción sobre este también depende de otros factores como puedan ser por ejemplo la frecuencia o la distancia a la cual se encuentre el sonido de otros sonidos superpuestos en el tiempo, cancelándose así unos a otros.

Observando la forma de onda, también podemos apreciar la frecuencia del sonido (dada en Hz), siendo esta el número de veces que se repite la forma de onda en un segundo. Este

parámetro es responsable de lo aguda o grave que se escucha una nota, y cuanto más corto o rápido sea el periodo de repetición (es decir, a mayor frecuencia) más agudo será el sonido. Tenemos que tener en cuenta que esto solo aplica a ondas periódicas o casi periódicas en las que se repite todo el rato la misma forma de onda, provocando siempre el mismo tono. También cabe señalar que el oído humano solo capta un determinado rango de frecuencias y que este va disminuyendo con la edad, estando éste aproximadamente entre los 20 Hz y los 20000 Hz, siendo los sonidos graves los que van de 20 a 300 Hz, los medianos de 300 Hz hasta 2000 Hz y los agudos de 2000 Hz hasta el final del espectro audible. Normalmente, con la edad disminuye la frecuencia máxima audible en los seres humanos.

Volviendo al sonido en sí, como hemos aclarado antes, este es provocado por una vibración que se propaga en el aire creando una reacción en cadena y finalmente un sonido. Asimismo, diferentes formas de onda (aún con la misma frecuencia fundamental) provocan distintos sonidos. Esto tiene un por qué, y es debido a los armónicos. Para entender el fenómeno de los armónicos primeros debemos saber que una onda senoidal es 'pura' en el sentido de que cuando suena, suena solo un tono sin ninguna otra frecuencia adicional. Sin embargo, al tener cualquier otra forma de onda como pueda ser, por ejemplo, triangular o cuadrada aparecen armónicos. Los armónicos son frecuencias adicionales generalmente de menor amplitud que pueden aparecer según la forma de onda de esta y que por tanto alteran el timbre del sonido. Los armónicos siempre son múltiplos de la frecuencia fundamental, siendo el segundo armónico la frecuencia fundamental multiplicada por dos y así sucesivamente. Como ejemplo en una onda triangular aparecen los armónicos impares siendo el 3º,5º y 7º los más audibles. Generalmente, a más alto armónico, menor amplitud del mismo, por tanto los primeros armónicos se van a escuchar más.

Todas estas formas de onda con armónicos se consiguen sumando ondas senoidales de diferente amplitud y frecuencia, siendo las ondas senoidales las ondas 'base' por las cuales se construyen todos los demás tipos, ya que cualquier otra onda se puede descomponer en ondas senoidales[26][27].

8. CURVA ADSR

Obviamente el volumen de un sonido es un factor fundamental a la hora de ser percibido. Es necesario señalar que un sonido natural tiene variaciones en el volumen a lo largo del tiempo. Por ejemplo, si pulsamos con fuerza la tecla de un piano durante un instante y seguidamente soltamos la tecla, esta producirá un sonido con un volumen muy elevado el cual decaerá muy rápidamente hasta que deje de sonar. Si por el contrario pulsamos una tecla suavemente el sonido será más débil desde un primer instante, pudiendo por tanto tocar una misma nota de diferentes maneras y teniendo ésta variaciones de volumen a lo largo de su duración en el tiempo.

Los tonos producidos por un sintetizador suelen tener por defecto siempre el mismo volumen, desde que empiezan a sonar hasta que acaban de manera abrupta produciendo un sonido completamente estático, de forma que cuando los escuchamos nos producen una sensación de artificialidad que no se asemeja nada a como sería tocar una nota con un instrumento real. Por fortuna, la Web Audio nos permite aplicar una curva ADSR a nuestro sonido para que suene más natural y así evitar que este sea únicamente un tono robótico que no varía con el paso del tiempo.

En cuanto a la curva ADSR en sí, esta es una manera de definir el volumen de un sonido a lo largo de su duración. Viene dado por cuatro parámetros, que nos permiten dividir el sonido en cuatro fases que son las siguientes[28]:

- **Attack**: es el tiempo que tarda en aparecer el sonido y llegar a su pico inicial, Es, básicamente un *fade in* en el que el sonido va desde 0 hasta su valor máximo inicial.
- **Decay**: es el tiempo en el que decae la nota después de alcanzar su pico máximo hasta alcanzar el periodo denominado *sustain*, que representa la amplitud en estacionario.
- **Sustain**: este parámetro es el volumen que tiene la nota mientras se quede sonando después del ataque inicial y el *decay*, y antes de soltar la nota. Esto sería, por ejemplo, el volumen de un piano mientras mantengamos la nota pulsada o el volumen de una nota de una trompeta mientras sigamos soplando.

- **Release**: para acabar tenemos el *release*, siendo este parámetro el *fade out* que se produce desde que soltamos la nota hasta que esta desaparece completamente (el volumen llega a cero).

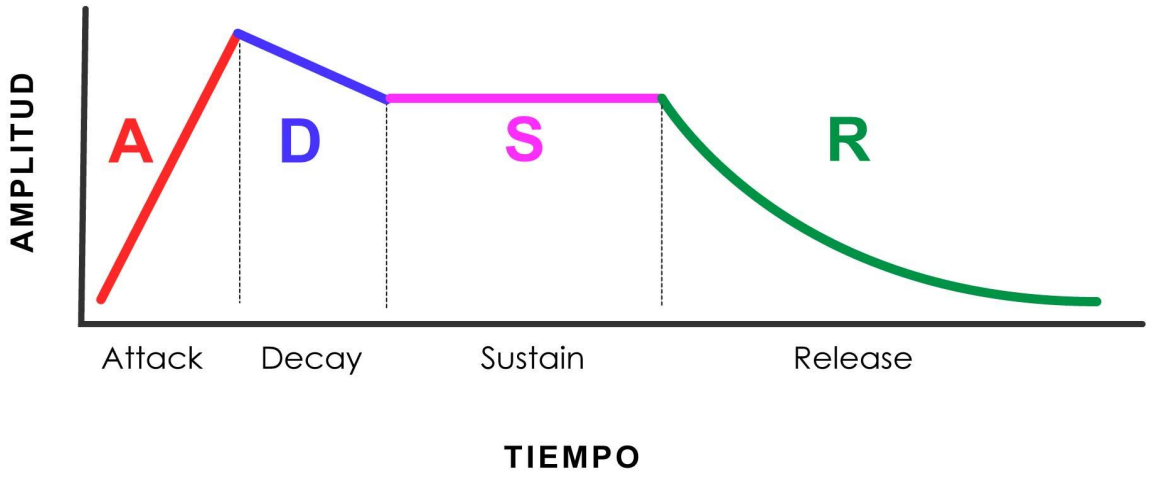

Figura 12: ejemplo curva ADSR

Jugando con estos cuatro parámetros podemos conseguir aplicar distintos efectos a nuestro sonido a la vez que conseguimos que suene más natural. Algunas aplicaciones serían, por ejemplo, hacer sonar una nota con un ataque y un *release* corto, lo que provocaría un efecto de picado. También podemos alargar el *release* para intentar simular algunos instrumentos percusivos como la caja. Otra opción sería alargar el tiempo de ataque para conseguir un efecto de sonido exuberante que de una sensación de grandeza.

9. Fundamentos musicales

Para entender el correcto funcionamiento de la aplicación también es necesario entender unos conceptos básicos sobre música. Concretamente es recomendable tener unas bases de cómo funcionan las diferentes escalas, intervalos y los acordes, para así saber en qué se basa la aplicación propuesta.

Empezando por los intervalos, un intervalo musical se define por la distancia en tonos o semitonos entre dos notas. Sirven para poder medir la distancia entre varias notas y por ello, sirven de base para los acordes y las escalas, ya que estos se distinguen por los intervalos entre las notas que los componen. Los intervalos se pueden clasificar entre disminuidos, aumentados, mayores o menores dependiendo de la distancia en tonos/semitonos, También se clasifican por la distancia entre las notas que los forman. Por ejemplo, un intervalo Sol-La sería uno de segunda ya que las dos notas son consecutivas. Es decir, hay dos notas de diferencia, mientras uno de Sol-Si de tercera, ya que hay tres notas de diferencia: Sol-La-Si, y así consecutivamente. Asimismo, si tenemos un intervalo de tercera con una distancia dos tonos, será una tercera mayor, si es de distancia de un tono y medio será una tercera menor, si fuera de un tono de distancia sería disminuida, y si es de dos tonos y un semitono de distancia sería una tercera aumentada

Después tenemos las escalas, las cuales son secuencias de siete notas compuestas por unos intervalos entre notas específicos. Estas son muy importantes porque son la base para crear una melodía ya que proporcionan una especie de plantilla para definir las notas y los acordes que sonarán en ella (a diferente tonalidad, diferente armadura y acordes). Según los tipos de acordes que las forman, las escalas se dividen en mayores y menores, produciendo estas primeras sonidos que evocan emociones felices como la alegría mientras que las segundas producen tonos más tristes o melancólicos. Básicamente tenemos la escala mayor y la menor, por cada nota. Puesto que esto incluye a las notas con alteraciones, podemos encontrar

escalas que sean exactamente iguales como por ejemplo las escalas mayores o menores de La bemol y las de Sol sostenido.

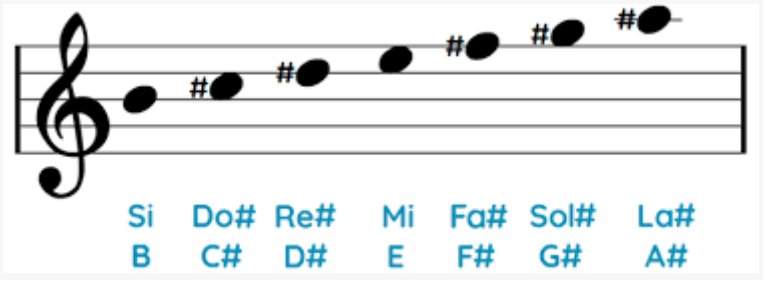

Figura 13: escala de Si Mayor

También es importante aclarar qué es un acorde, siendo éstos combinaciones de las notas de una escala. Los más básicos que son los que aparecen en la aplicación están formados por la primera, la tercera y la quinta nota de la escala y se llaman tríadas o acordes fundamentales. Los acordes se usan como base en la música y dependiendo de la escala en la que estemos, en una determinada pieza aparecerán acordes propios de esa escala.

10. APLICACIÓN

Una vez terminados de explicar todos los conceptos teóricos, pasamos a la aplicación en sí. Esta se compone de una sola página HTML, los elementos de la cual van variando según pulsemos un botón u otro. Como hemos dicho anteriormente, la aplicación consiste en una serie de ejercicios con los que podemos entrenar nuestro oído, a la vez que disponemos de un sintetizador que nos permite tocar las notas o acordes que queramos ya sea dentro de los propios ejercicios o de manera libre.

Asimismo, cabe destacar que al estar formada la aplicación por una sola página HTML, los elementos de la misma van variando según donde clickemos de forma dinámica, estando los diferentes ejercicios y funciones encapsulados dentro de elementos *div* los cuales solo se muestran cuando son necesarios. Para ello contamos con funciones denominadas *set*, las cuales se encargan de bloquear ciertos elementos HTML y mostrar otros según naveguemos por la página.

Cuando iniciamos por primera vez, observamos una pantalla con una serie de botones arriba, los cuales sirven para activar los distintos ejercicios. También contamos con una barra lateral desplegable, la cual nos permite modificar el timbre del sonido de la aplicación.

Figura 14: botones con los que podemos navegar por la aplicación

Asimismo, lo que destaca en la misma es el teclado del piano que aparece en el centro de la pantalla, el cual es interactivo.

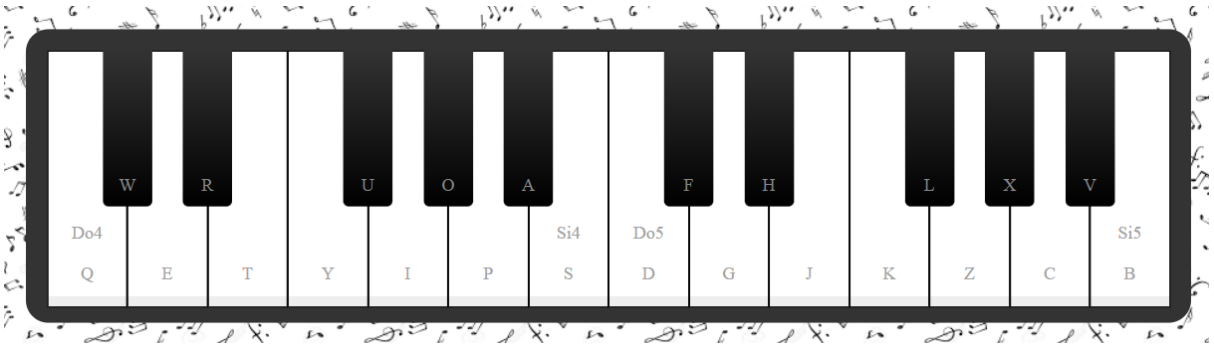

Figura 15: el piano de la aplicación

Tenemos dos maneras de hacerlo sonar, siendo estas o bien clicar con el ratón en las diferentes teclas o usar el teclado del ordenador como si estas fueran las teclas de un piano. Sea como sea el sonido es exactamente el mismo.

El piano funciona de manera que al pulsar una tecla de la manera que sea se activa un *event listener* determinado que hace funcionar un oscilador con la frecuencia que toque. Asimismo, mediante otro e*vent listener* el oscilador deja de sonar en el momento en el que dejamos de pulsar ya sea la tecla o el botón del ratón. Para ello nos valemos de la función *StartOsc*, la cual tiene como parámetros las características del oscilador que tiene que sonar y un booleano que indica si el oscilador debe empezar a sonar o si ya está sonando y debe parar.

Si indicamos que la nota no está sonando, primero la función asigna los valores a los parámetros de la curva ADSR según el *slider* que encontramos en la aplicación.
Después, con otro contador comprobamos si la nota está sonando o no ya que al mantener pulsada una tecla se llama a esta función constantemente, pero nosotros solo queremos que el sonido se active la primera vez. Dicho esto, al detectar que no hay nada sonando asignamos el valor de frecuencia correspondiente al oscilador, la forma de onda correspondiente y los valores de la curva ADSR con un nodo de ganancia. Hecho esto, iniciamos el oscilador. En caso de que este contador no sea nulo, la función no hace nada.

Por otra parte, si el booleano es falso la función simplemente detiene el oscilador para que así pare de sonar, evento que se produce al dejar de pulsar la tecla o al soltar el botón del ratón.

Asimismo, arriba a la izquierda tenemos una barra que consta con cuatro botones, uno para cada ejercicio y otro para un menú lateral desplegable.

10.1 BARRA LATERAL

La barra lateral consta de un seleccionador con el que podemos elegir entre las cuatro formas de onda predefinidas, siendo éstas cuadrada, triangular, senoidal y de diente de sierra. Esta parte del código funciona mediante un *event listener* que detecta si hemos pulsado el botón y entonces o bien despliega o esconde la barra lateral.

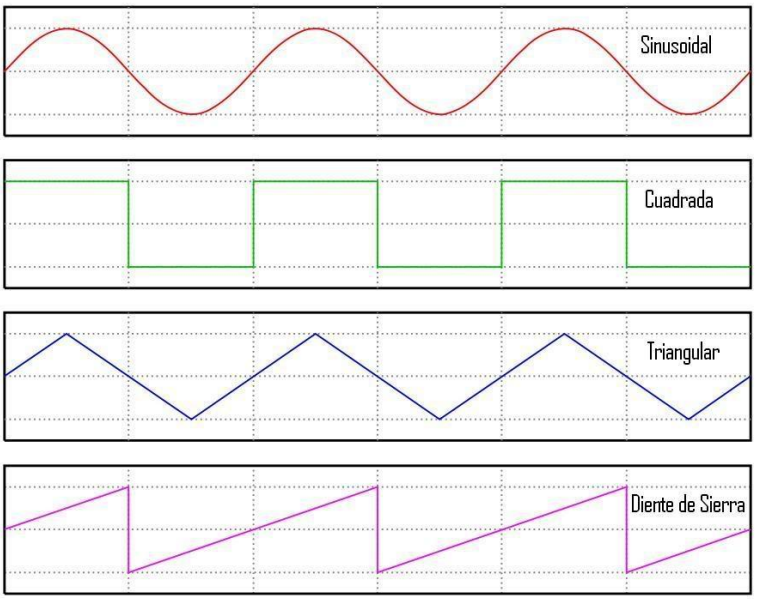

Figura 16: las cuatro formas de onda implementadas en la aplicación.

En cuanto al cambio de forma de onda, esto también funciona mediante el uso de un contador por el cual se asigna la forma de onda correspondiente a la hora de crear el sonido, cambiando este contador según el parámetro que seleccionemos en el selector, habiendo cuatro posibles valores.

Por otra parte, también podemos modificar 3 apartados de la curva ADSR mediante *sliders*. Hemos decidido omitir el parámetro *release* porque, aunque técnicamente es perfectamente

posible modificarlo al igual que los otros, creemos conveniente no hacerlo debido a que provoca que al sonar dos notas seguidas el sonido se solape y pueda molestar a la hora de hacer los ejercicios propuestos.

En cuanto a los otros tres parámetros, contamos con un *slider* para poder configurarlos a nuestro gusto, pudiendo modificar el volumen en el estado *sustain* y el tiempo de *attack* y *decay*. Después, el código javascript recoge los valores de los tres *slider* y los aplica al sonido en el método *StartOsc*, al igual que con las formas de onda.

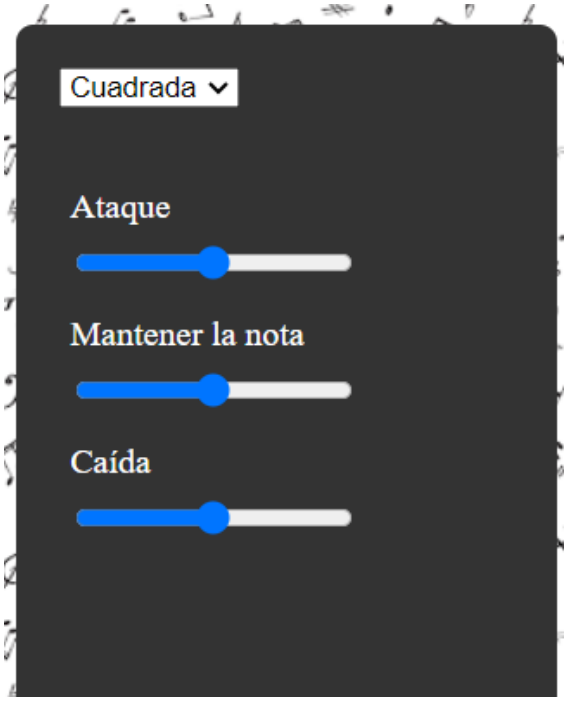

Figura 17: Barra lateral.

11. EJERCICIOS

Si hacemos click en alguno de los tres botones de la parte de arriba a la izquierda al lado del botón de la barra, accederemos a los tres ejercicios de los que consta la aplicación.
Estos funcionan de manera similar, cada uno centrándose en algo diferente, siendo el primero sobre intervalos, el segundo sobre escalas y el tercero sobre arpegios.

Estos ejercicios funcionan de manera que escuchamos una serie de notas, ya sea en un intervalo, un arpegio o una escala, para de seguido tener que reproducirlas el usuario haciendo uso del sintetizador de la aplicación. A continuación, se leen los parámetros introducidos por el usuario para ver si la respuesta introducida es correcta, pasando ésta a una función marcador. Aquí, en caso de que la respuesta del usuario sea correcta, se suma un punto al marcador de ese ejercicio y en caso de que sea incorrecta, el marcador se resetea.

Por otra parte, para poder funcionar, necesitamos tener definidos en el código todos los diferentes arpegios, escalas e intervalos como variables. Para los intervalos tenemos una variable diccionario que almacena las diferentes distancias (2ª, 3ª, 4ª, etc) junto a los valores que representan el número de semitonos en cada intervalo, habiendo uno disminuido, uno menor, otro mayor y finalmente uno aumentado. Por lo que respecta a las escalas y a los acordes, tenemos una lista por cada escala o acorde, la cual está formada por ceros y unos. Estos números representan todas las notas que puede reproducir la aplicación y si el número correspondiente a una nota es uno, significa que esta pertenece a la escala o acorde.

Asimismo, también tenemos definidas funciones que desactivan todos los *event listeners* mientras suenan las notas de los ejercicios, para que así no sea posible tocar nada mientras estas se reproducen, habiendo otras funciones que los vuelven a activar cuando es necesario. Además, el código también cuenta con una función que implementa un sonido de error generado por la propia aplicación en el momento que introducimos una respuesta errónea.

Además, cada ejercicio cuenta con un botón con el que tenemos una nota de referencia, para que así no sea requerido tener oído absoluto para hacer los ejercicios, siendo la nota La en el ejercicio uno y la escala y el arpegio de Do Mayor en los ejercicios dos y tres.

11.1 EJERCICIO 1

Este primer ejercicio se centra en intervalos y funciona de manera que suenan dos notas aleatorias en un intervalo que puede ser desde una segunda a una octava. Por tanto, debemos introducir mediante el teclado o el ratón las dos notas correspondientes. Asimismo, también debemos indicar mediante dos selectores la longitud del mismo y su tipo (mayor, menor, aumentado etc.) Si introducimos todos los datos de manera correcta sumaremos un punto y si no el marcador se reseteará. Es importante aclarar que, al haber incluído intervalos disminuidos y aumentados, tenemos intervalos que se solapan. Por ejemplo, una tercera disminuida y una segunda mayor es el mismo intervalo. En estos casos la aplicación permite introducir cualquiera de las dos opciones siendo las dos consideradas correctas.

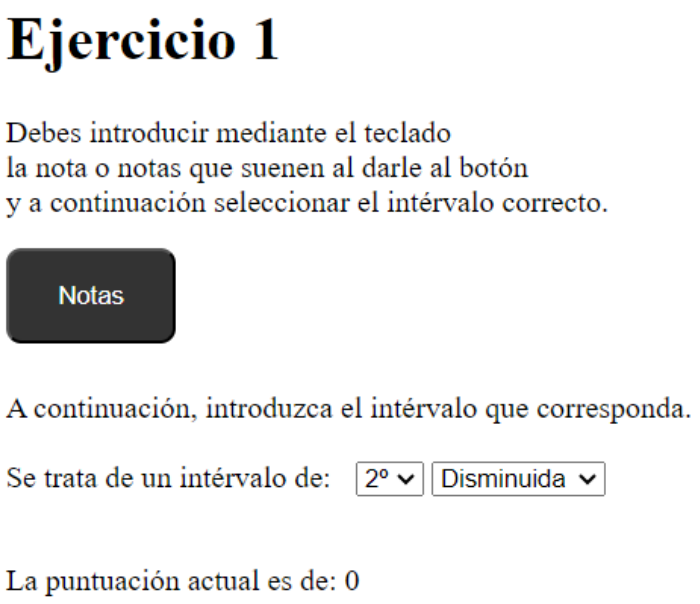

Figura 18: instrucciones ejercicio 1.

11.2 EJERCICIO 2

Este ejercicio se basa en escalas. Cuando hacemos click en el botón, suena una escala aleatoria mayor o menor (para simplificar la aplicación y que no sea tan simple hemos optado por no incluir las escalas basadas en notas con alteraciones). En cuanto al funcionamiento del ejercicio, este es muy similar al ejercicio uno, sonando una escala y teniendo que volverla a tocar mediante las entradas que nos deja usar la aplicación y sumando un punto en caso de que esta sea correcta.

11.3 EJERCICIO 3

Este ejercicio es muy similar al de las escalas pero con la diferencia de que trata sobre arpegios en vez de escalas. A diferencia de los dos primeros ejercicios en este las tres notas que forman el arpegio se superponen de forma que al clickar en el botón solo suena la primera y se van añadiendo la otras de forma que acaban sonando las tres a la vez. Hemos decidido hacerlo de esta forma ya que normalmente los arpegios se suelen escuchar de esta forma.

12. DESPLIEGUE

Hacer funcionar la aplicación es una tarea bastante sencilla aunque para ello necesitamos tener Node.js instalado en nuestro equipo. Por tanto, para desplegar la aplicación solo tenemos que entrar a la consola de Windows en el directorio en el que tengamos el archivo servidor y ejecutar el comando *npm start,* con el cual iniciaremos el servidor en el puerto 3000.

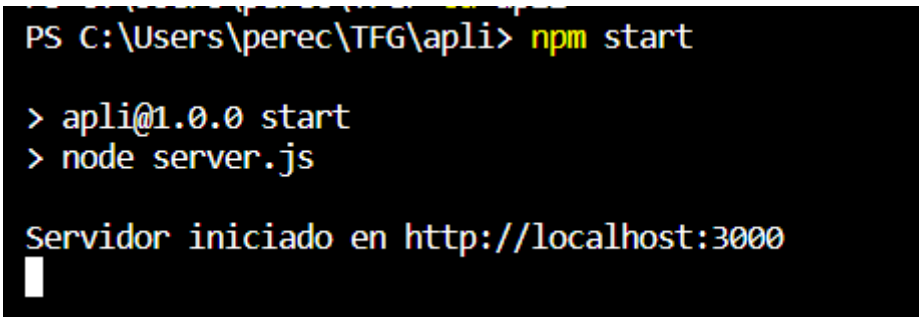

Figura 19: Arranque del servidor de Node

Después de esto, lo único que tenemos que hacer es entrar a la dirección http://localhost:3000, y así ya podremos hacer uso de la aplicación.

Enlace a Google Drive con la aplicación:
https://drive.google.com/file/d/1XXavIkUkO011ahiTVHEJnbxNgzXD0NaR/view?usp=sharing

13. CONCLUSIONES Y PROPUESTA DE TRABAJO FUTURO

Después de haber desarrollado la aplicación propuesta y de haber explorado todos los componentes teóricos que conforman la aplicación, podemos concluir que el uso de la Web Audio API es una herramienta más que válida para el desarrollo de aplicaciones web didácticas y de manipulación y procesado de audio.

Asimismo, también cabe destacar que el trabajo desarrollado se corresponde con el Grado en Tecnología Digital y Multimedia. Por ello se refleja en el mismo un carácter multidisciplinar, al igual que el propio grado. Esto se puede ver en que abarca varios campos como son la programación de aplicaciones web, el tratamiento de audio o el conocimiento sobre música los cuales se emplean para desarrollar la aplicación, resultando algo abordable para un estudiante de este grado el cual ofrece bases tanto en programación como en tratamiento básico de audio.

Por lo que respecta a propuestas de trabajo futuras relacionadas con el proyecto, este se podría ampliar de distintas maneras. Un ejemplo sería implementar el uso de síntesis aditiva para así poder simular el sonido de un piano real o cualquier otro instrumento. Asimismo, también sería posible ampliar la aplicación para ofrecer otro tipo de ejercicios, como pudieran ser algunos relacionados con ritmos o dictados musicales en los que directamente suenen melodías en vez intervalos o escalas.

14. BIBLIOGRAFÍA

Estado del arte:

[1]Historia general del audio en web(5/5/2023):
https://inria.hal.science/hal-03871527/document

[2]Audio en la web e insuficiencia de la etiqueta audio audio(15/5/2023):
http://multimedia.uoc.edu/blogs/fem/es/audio-en-la-web/

[3]Documentación de la etiqueta audio(15/5/2023):
https://www.w3schools.com/html/html5_audio.asp

[4]Documentación de mozilla la Web Audio API(26/5/2023):
https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API

[5]Documentación de W3C de la Web Audio API(27/5/2023):
https://www.w3.org/TR/webaudio/

Javascript:

[6]Fundamentos Javascript(9/6/2023):
https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/JavaScript_basics

[7]Historia de Javascript(9/6/2023):
https://dev.to/dboatengx/history-of-javascript-how-it-all-began-92a

[8]Características de Javascript y el porqué tiene Java en el nombre(9/6/2023):
https://www.hackaboss.com/blog/diferencias-javascript-java#:~:text=Java%20es%20un%20lenguaje%20de,organizaci%C3%B3n%20sin%20%C3%A1nimo%20de%20lucro

[9]Más características(12/6/2023):
https://www.studytonight.com/javascript/javascript-features

[10]Documentación mozilla HTML(27/6/2023):
https://developer.mozilla.org/es/docs/Web/HTML

[11]HTML 5(27/6/2023):
https://www.miformaciongratis.com/blog-post/html5-que-es-y-para-que-sirve/#:~:text=HTML5%20

[12]Documentación mozilla CSS(2/7/2023): https://developer.mozilla.org/es/docs/Web/CSS

[13]Global and local styling(2/7/2023):
https://every-layout.dev/rudiments/global-and-local-styling/

[14]Qué es Node.js(20/8/2023): https://kinsta.com/es/base-de-conocimiento/que-es-node-js/

[15]Otras características de Node.js(20/8/2023):
https://www.geeksforgeeks.org/what-are-the-key-features-of-node-js/

Web audio API:

[16]Documentación W3C(11/7/2023): https://www.w3.org/TR/webaudio/

[17]Estandar W3C(11/7/2023): https://www.w3.org/press-releases/2021/webaudio/

[18]Funcionamiento de la API y pasos a seguir(16/7/2023):
https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API/Using_Web_Audio_API

[19]Nodos(16/7/2023): https://developer.mozilla.org/en-US/docs/Web/API/AudioNode

[20]Contexto de Audio(22/7/2023):
https://developer.mozilla.org/en-US/docs/Web/API/AudioContext

[21]Osciladores(22/7/2023):
https://developer.mozilla.org/en-US/docs/Web/API/OscillatorNode

Síntesis:

[22]Basics of Sound Synthesis(5/8/2023):
https://theproaudiofiles.com/sound-synthesis-basics/

[23]Síntesis sustractiva: Qué es y como funciona(5/8/2023):
https://emastered.com/es/blog/subtractive-synthesis

[24]Síntesis aditiva y FM(5/8/2023):
https://musictech.com/guides/essential-guide/additive-and-phase-distortion-synthesis/#:~:text=Additive%20synthesis%20bears%20some%20resemblance,but%20it%20really%20isn't.

[25]Sintetizador Yamaha(6/8/2023):
https://meganlavengood.com/the-yamaha-dx7-in-synthesizer-history/

Ondas:

[26]Let's learn about Waveforms(9/8/2023): https://pudding.cool/2018/02/waveforms/

[27]What is a sound wave(9/8/2023):
https://www.techtarget.com/whatis/definition/sound-wave

Curva ADSR:

[28]Envelopes explained(10/8/2023):
https://www.masterclass.com/articles/adsr-envelope-explained#4bPVgIZ4W9GAOAyN3yNHDN

Código:

How to build a synthesizer(10/4/2023):
https://www.youtube.com/watch?v=uasGsHf7UYA&ab_channel=NIDevConf

Web Audio API oscillator tutorial(20/4/2023):
https://www.youtube.com/watch?v=Sdz5NI-meM0&ab_channel=TheCodeCreative

Build a piano(20/4/2023):
https://www.codingnepalweb.com/playable-piano-html-css-javascript/

Documentación Event listeners Javascript(25/4/2023):
https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener