UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Migración a Javascript de funciones desarrolladas en Matlab para su ejecución sobre el cliente web

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: Voronov , Aleksandr

Tutor/a: Bachiller Martin, Maria Carmen

Cotutor/a: Martínez Zaldívar, Francisco José

CURSO ACADÉMICO: 2021/2022

Resumen

Mediante el presente trabajo se pretende desarrollar y adoptar unos algoritmos de audio en el entorno web, que simulen el comportamiento de unos dispositivos reales: teléfono de baquelita, fonógrafo, gramófono, magnetófono, radio y receptor Morse. A la par, se pretende proporcionar una representación sonora de calidad y representativa de estos aparatos, ofreciendo en todos los casos una experiencia del uso del programa accesible en cualquier momento, y con la posibilidad de recuperar el fichero de audio procesado. El marco de desarrollo e integración del programa requiere su integración y ejecución en el lado del cliente, esto es en el navegador web del usuario.

Resum

Mitjançant el present treball es pretén desenvolupar i adoptar uns algorismes d'àudio en l'entorn web, que simulen el comportament d'uns dispositius reals: telèfon de baquelita, fonògraf, gramòfon, magnetòfon, ràdio i receptor Morse. A l'una, es pretén proporcionar una representació sonora de qualitat i representativa d'aquests aparells, oferint en tots els casos una experiència de l'ús del programa accessible en qualsevol moment, i amb la possibilitat de recuperar el fitxer d'àudio processat. El marc de desenvolupament i integració del programa requereix la seua integració i execució en el costat del client, això és en el navegador web de l'usuari.

Abstract

Through this work it is intended to develop and adopt audio algorithms in the web environment, which simulate the behavior of real devices: Bakelite telephone, phonograph, gramophone, tape recorder, radio and Morse receiver. At the same time, it is intended to provide high quality sound of these devices, offering in all cases an experience of using the program accessible at any time, and with the possibility of recovering the processed audio file. The development and integration framework of the program requires its execution on the client side, that is, in the user's web browser.

Índice

Capítulo 1. Introducción

1.1 Objetivos

El objetivo principal de este trabajo es contextualizar el equipamiento de audio y hacerse una idea de cómo funcionaban, mediante una serie de algoritmos que digitalmente producen una sensación sonora similar a la que producían los dispositivos del "Museo de la Telecomunicación Vicente Miralles Segarra", ubicado en el edificio de la Escuela Técnica Superior de Ingeniería de Tecnologías y Servicios de Telecomunicación (ETSIT) en la Universidad Politécnica de Valencia (UPV). Dichos algoritmos tendrán que reunir los requisitos necesarios de seguridad y de poder ejecutarse en el lado del usuario, es decir en su navegador web, con el fin de no sobrecargar el servidor de la UPV.

La aplicación estará diseñada para cualquier dispositivo electrónico que permita el acceso a la web y soporte las últimas versiones de navegadores web más populares (Chrome, Safari, Edge, Mozilla, Opera), con el fin de ofrecer una recreación fidedigna de la realidad del funcionamiento de los equipos de audio ubicados en el museo: fonógrafo, gramófono, magnetófono de hilo, teléfono de baquelita, equipo de recepción de radio y receptor Morse.

Sin la necesidad de dar una descripción exhaustiva de cada uno de los aparatos, así como de su funcionamiento, se describirá en qué consisten, así como los atributos principales sonoros de cada uno de estos, lo que implica estudiar los efectos de audio específicos de cada uno de los aparatos: "wow", "flutter", distorsión, respuesta impulsiva, ruido y codificación (en caso de codificación Morse); fenómenos que se definirán en detalle.

Por último, pero no menos importante, y cómo objetivo secundario, se tratará de ofrecer al usuario un uso sencillo e intuitivo de aplicación de audio en la web para facilitarle el manejo de la aplicación y por tanto conseguir su satisfacción.

Con todo ello, se pretende alcanzar una alta sensación de realismo y ofrecer al usuario una experiencia educativa, innovadora y satisfactoria, complementaria a una visita al museo (presencial o virtual), sin perder de vista los fines recreativos del uso de la aplicación de audio en la web.

1.2 Motivación

Una de las principales motivaciones de este trabajo consiste en desarrollar una aplicación completamente funcional en el lado del cliente (navegador web) haciendo uso de lenguaje de marcado HTML, lenguaje de estilo CSS, y lenguaje de programación web JavaScript; las tres herramientas forman un conjunto de software de programación web imprescindible y que se ejecuta en el denominado "front-end" o parte del cliente web. Así, se ha propiciado desde el principio que todos los algoritmos de procesado se ejecuten en el lado de cliente, lo que permite

una integración directa en un sistema de gestión de contenidos (CMS), que en el marco del proyecto será WordPress.

Otra de las motivaciones consiste en el creciente despliegue de las redes de acceso de banda ancha (redes fijas y redes móviles), que según la Fig. 2, incluyendo a la fibra óptica (FTTH y FTTB), el cable Docsis 3.0 (redes HFC), VDSL; las redes fijas, dieron abasto a un 92.3% de los hogares en España durante el 2020. Por otro lado, según el informe de Ministerio de Asuntos Económicos y Transformación Digital el despliegue de las redes móviles de las tecnologías UMTS/HSPA y LTE suponían de 100% y 99.9% [1]. Dichas infraestructuras proveen de una tasa binaria y latencia adecuadas; para hacer un uso eficiente de las aplicaciones web.

Otro hecho, no menos importante, se podría observar en la Fig. 1, que indica un creciente uso de internet, durante el 2021, conforme la población más joven siendo esta, el público objetivo de la aplicación a desarrollar.

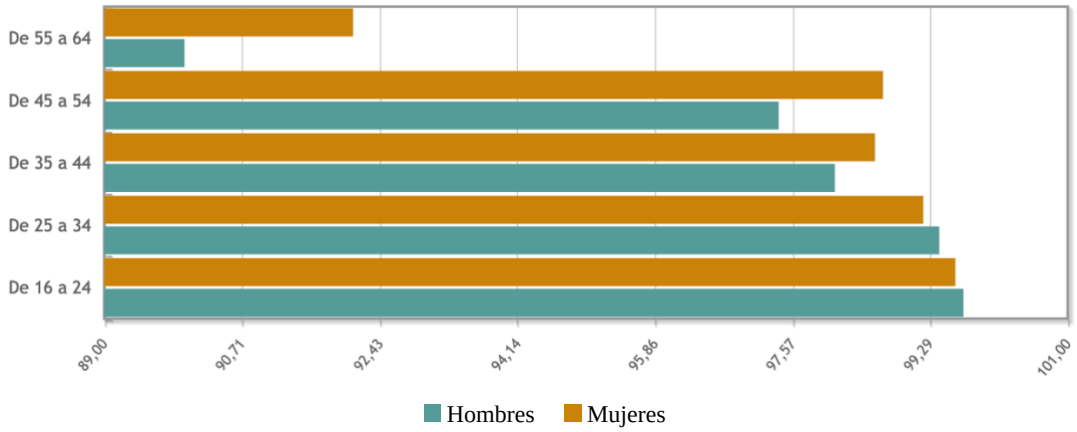

Fig. 1. Porcentaje de población que ha usado Internet en los últimos tres meses por grupos de edad en 2021. Fuente: INE.

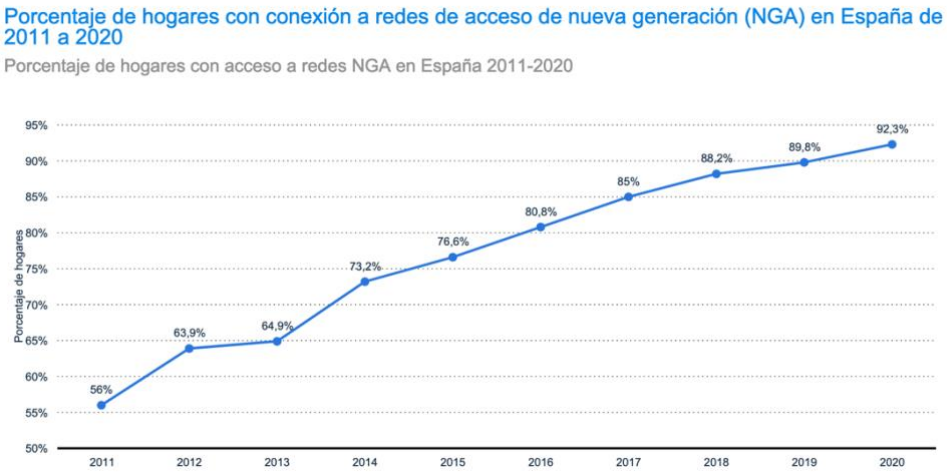

Fig. 2. Porcentaje de hogares con conexión a redes de acceso NGA en España de 2011 a 2020. Fuente: "Statista".

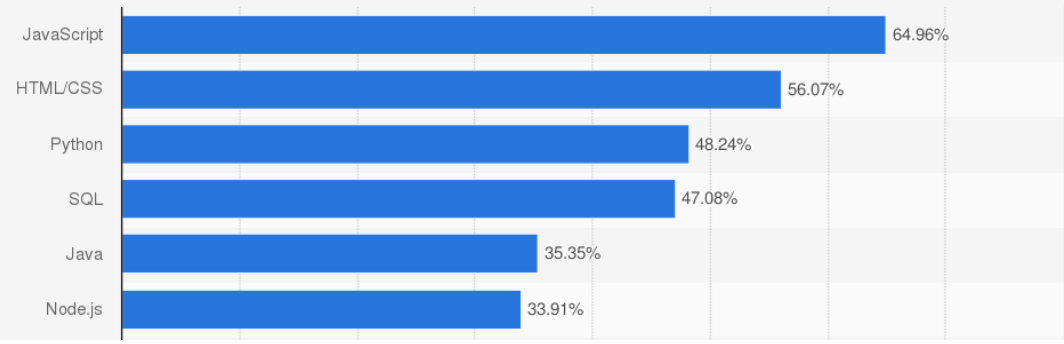

Fig. 3. Lenguajes de programación más utilizados entre los desarrolladores de todo el mundo, a 2021. Fuente: "Stack Overflow".

Por otro lado, como se puede observar en la Fig. 3, un estudio realizado por "Stack Overflow" (un sitio web especializado en preguntas y respuestas para los programadores profesionales), confirma que de los 58.031 programadores que desempeñan su labor en la industria de tecnologías de información alrededor de 64.96% confirman que el lenguaje de programación JavaScript es más utilizado, adelantando a los otros dos lenguajes HTML/CSS, y que se suelen usarse en conjunto para la programación de aplicaciones en el lado de cliente ("Front-end"). De esta forma, se ha propuesto utilizar estos lenguajes de programación por su uso frecuente en el desarrollo de las aplicaciones web.

Asimismo, es importante remarcar las tres misiones que marcan el funcionamiento del museo: conservación del patrimonio, actividad de investigación y la divulgación. Entre los fines divulgativos de este proyecto, se encuentra el de ofrecer a los visitantes la experiencia de disfrutar del sonido de los equipos de audio de los tiempos antaño. Entre otros fines se destacan el fin de entretenimiento, de investigación tecnológica e histórica.

Finalmente, el uso de las tecnologías web en conjunto con los diversos API (interfaz de programación de aplicaciones) supone un reto que podría ser enriquecedor para complementar las visitas al museo, mediante el empleo de una nueva tecnología y a la vez hacer posible la reproducción de los equipos virtualmente, sin la necesidad de tocarlos físicamente, con el fin de preservarlos.

1.3 Plan de Trabajo

El proceso de desarrollo de la aplicación se ha dividido en dos partes. La primera parte ha consistido en el estudio de los equipos de audio y sus características sonoras, así como del código proporcionado por los profesores tutores, que había sido empleado en desarrollar aplicaciones similares por los antiguos alumnos de la escuela en lenguajes Matlab y C [2]. La segunda parte ha consistido en la elección de herramientas necesarias para el desarrollo de la aplicación: la búsqueda de lenguajes de programación adecuados, la preparación del entorno de desarrollo integrado (IDE), así como la búsqueda de las API necesarias; finalmente el desarrollo de la propia aplicación y su integración con WordPress.

1.4 Estructura

En el "Capítulo 2. Estado del Arte", se estudiarán aplicaciones web similares en otros museos.

En el "Capítulo 3. Ámbito", se describirán brevemente los aparatos reales que se implementarán digitalmente, así como los efectos relacionados; también se presentará una breve evolución de las herramientas de "Front-end" (HTML, CSS, JS), "Back-end" (PHP), CMS (WordPress).

En el "Capítulo 4, Desarrollo del Programa", se describirá la estructura del programa, en cuanto a la organización de los módulos y sus funcionalidades principales y se integrarán los módulos en WordPress del sitio web del museo.

En el "Capítulo 5. Análisis y Discusión de los Resultados" se discutirán los resultados obtenidos de la integración en cuanto a tiempos de respuesta (y su comparación con tiempos obtenidos mediante el uso de mismos algoritmos definidos en Matlab y C), carga de módulos mediante un protocolo HTTP con el método GET cuando se soliciten archivos y los diagramas de cascada para detectar qué módulo se carga más lento. Se hará uso de una encuesta elaborada para evaluar la satisfacción de los usuarios.

En el "Capítulo 6. Conclusiones y Líneas Futuras" se discutirán las líneas futuras y conclusiones del proyecto de audio en la nube.

Capítulo 2. Estado del Arte

Este capítulo consta de una primera subsección que describe la evolución en el uso de las tecnologías digitales vanguardistas en el marco de las visitas a los museos hasta el día de hoy. En la subsección posterior se describe la principal innovación que introduce el presente trabajo en dicha materia.

2.1 La evolución del uso de las tecnologías digitales por los museos

El turismo cultural es una de las formas más antiguas de viajar con fines de ocio, que se ha convertido en una importante fuente de recursos para la industria del turismo. El uso de las tecnologías de procesamiento digital de audio e imagen están muy presentes en el turismo de la sociedad de hoy día, llegándose a conocer como "turismo digital".

La tecnología digital ha servido como una herramienta importante para innovar en todas las áreas del museo, quienes consiguen una nueva posición en el mercado de turismo, resultando en nuevas formas de consumo. Los museos pueden no estar completamente preparados para aceptar la visita del turista digital cultural, aunque el reconocer el valor del nuevo consumidor puede liberar a los museos de los límites impuestos por sus muros físicos para poder explorar nuevos horizontes en el mercado digital de información [3].

Cabe remarcar que se había celebrado una jornada de "Transformación Digital de Museos y Patrimonio Cultural" en el "Museo Arqueológico Nacional" organizada por la plataforma eNEM, siendo un organismo I+D+i del sector de contenidos digitales y las industrias culturales y creativas de AMETIC, "Asociación Multisectorial de Empresas de Tecnologías de la Información, Comunicaciones y Electrónica" agrupa a fabricantes, comercializadores, distribuidores y en el caso del sector de Telecomunicaciones a instaladores de la infraestructura. Durante el evento, enmarcado dentro del "Año Europeo del Patrimonio Cultural 2018", se ha destacado el papel de las nuevas tecnologías para mejorar la accesibilidad al patrimonio cultural y darlo conocer, contribuyendo a protegerlo. La aplicación de estas innovaciones ha supuesto una importante revolución para los museos y centros culturales, ya que pueden dar mayor visibilidad a las obras y exposiciones, así como poner en valor las actuaciones para preservar la cultura y el patrimonio. Gracias a ello, se ha logrado universalizar la cultura permitiendo que el público visite virtualmente los museos más famosos del mundo y contemple las obras cómodamente desde casa [4].

Asimismo, entre las posibles motivaciones que justifiquen la digitalización de las visitas al museo podrían encontrarse en las consecuencias de la pandemia producida por la pandemia de enfermedad de coronavirus, "Covid-19", que introdujo la reducción de la movilidad durante el estado de la cuarentena. Por lo tanto, los museos habían evolucionado para ofrecer las visitas virtuales a través de los dispositivos electrónicos domésticos (portátil, ordenador, teléfono, etc.) que posibiliten la conexión a la web. Una muestra de los museos que ofrecen esta experiencia es el famoso "Museo del Prado" ubicado en Madrid, con una guía virtual que ofrece un paseo y la visualización virtual de sus obras de arte, véase Fig. 4.

Fig. 4. La sala 3, con las obras de las Poesías. Fuente: "Museo del Prado".

Es importante remarcar una vez más que el uso de las tecnologías digitales ha tenido un desarrollo importante durante los últimos años hecho que se puede observar en la Fig. 5, dónde España lidera a los países europeos seleccionados (Francia, Alemania, Italia, Reino Unido) en cuanto a la introducción de servicios y productos digitales (cómo podrían ser las aplicaciones entre otros) en la industria de las exposiciones.

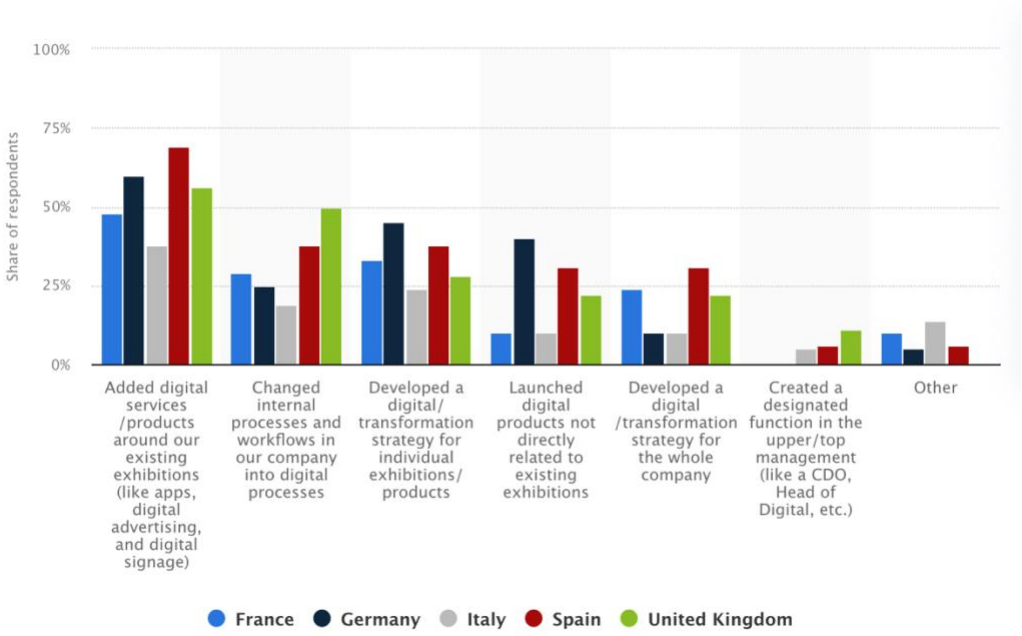

Fig. 5. Mejoras digitales implementados en la industria de exposiciones en países europeos seleccionados a junio de 2021. Fuente: "statista"

Actualmente se están evaluando algunas de las opciones del uso de las tecnologías emergentes cómo podría ser el uso del metaverso, un mundo virtual que ofrece una experiencia inmersiva facilitada por el uso de gafas de realidad virtual (VR), entre otros dispositivos que podrían ofrecer las experiencias sensoriales que imanan de la interacción con los objetos VR. En el contexto de la web, ya existe una interfaz de programación de aplicaciones que permite desarrollar las aplicaciones de realidad virtual, se denomina “WebXR Device API” ofreciendo herramientas de desarrollo web necesarias para acceder a dispositivos de realidad virtual y realidad aumentada, como podrían ser las gafas virtuales “Oculus Rift”, entre otras [5].

De este modo el propio usuario podría identificarse en el metaverso por medio de un “avatar”, una representación gráfica que se asocia a un usuario real en particular. En el contexto de las visitas virtuales, algunas de las aplicaciones de metaverso podrían encontrarse en el uso de los expositores de los NFT (“not fungible tokens”) que son a la vez activos digitales y productos digitales únicos, de tipo histórico o del patrimonio cultural, como por ejemplo los cuadros digitales, véase Fig. 6.

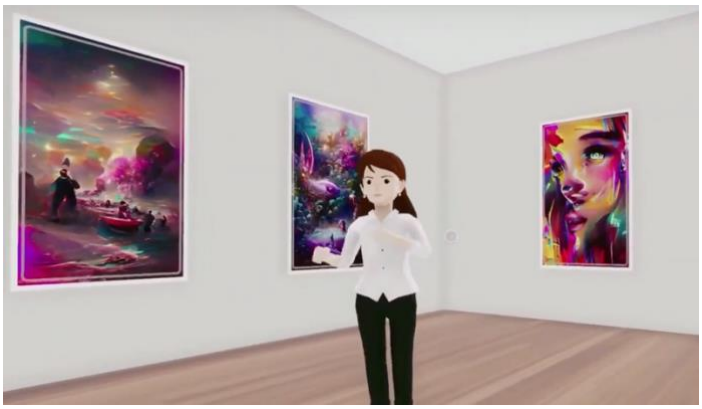

Fig. 6. Cuadros digitales NFT y un avatar. Fuente: "CollecOnline"

"Musee Dezentral" es uno de los múltiples museos en metaverso con los NFT que permiten hacer visitas virtuales e interactuar con otros usuarios mediante un avatar a través del siguiente enlace "https://musee-dezentral.com/museum/".

Fig. 7. Ejemplo de un museo en el metaverso. Fuente: "Musee Dezentral".

2.2 Relación con el trabajo.

Aunque se había descrito el uso de las tecnologías inmersivas a lo largo de este capítulo, por su suma importancia en la evolución de las visitas virtuales en los museos, el presente trabajo se enfoca en el desarrollo de una aplicación de audio que permita un procesado del audio en línea.

La innovación principal que ofrece la aplicación desarrollada, consiste en que permite procesar el audio en el sitio web del museo, una innovación que sin embargo es única a nivel de los museos de las universidades de España, puesto que ofrece una experiencia al usuario en todo el momento y en cualquier lugar, siendo el único requerimiento el de acceso al Internet mediante un terminal fijo o móvil.

Capítulo 3. Ámbito

En este capítulo se hará una descripción de las características sonoras de los dispositivos de audio: teléfono de baquelita, fonógrafo, gramófono, magnetófono, radio, receptor Morse. De este modo, se describirá el marco histórico de estos aparatos y se acompañaran los distintos efectos característicos con el código escrito en JS, en los casos de distorsión, "wow", flutter, ruido y respuesta ante el impulso.

Acto seguido se describirá la evolución de las herramientas de desarrollo web, del lado de cliente que son HTML, CSS, JS; y por el lado del servidor: PHP. Se comentará la importancia del uso de un sistema de gestión de contenidos WordPress para manejar el diseño y contenidos multimedia del sitio web del museo.

3.1 Efectos sonoros

En esta parte se describirán los efectos sonoros que se han simulado.

- Conversión de estéreo a mono

La mayor parte de las grabaciones audio hoy en día constan de dos canales independientes (canal izquierdo y canal derecho) y que forman un sonido estereofónico, lo que recrea una perspectiva audible tridimensional y multidireccional a diferencia de sonido en mono definido solo por un canal.

No fue hasta diciembre de 1931, cuando se patentó por la primera vez la grabación estéreo por Alan Dower [6]. Antes de la invención de tecnología de reproducción y grabación estereofónica se habían empleado otras tecnologías en grabación monofónica cómo cilindros cubiertos de cera en caso de fonógrafo, disco plano en caso de gramófono o hilo metálico en caso de magnetófono de hilo.

Con el fin de simular los dispositivos analógicos de forma fidedigna se hará la conversión de audio estereofónico de dos canales a un único canal, mediante el promediado de los dos canales digitales:

$$x_{mono} = \frac{x_{ch\_1} + x_{ch\_2}}{2}$$

Ecuación 1. Conversión de estéreo a mono. Fuente: elaboración propia.

- Filtrado digital

Las herramientas de grabación actuales han evolucionado hasta el punto de llegar a tener un ancho de banda de grabación y de reproducción elevados (llegando a tener tasas de muestreo de 44.100 Hz en CD, 48.000 Hz en caso de televisión digital, 96.000 Hz en HD-DVD y Blu-

ray), que permite una reproducción verídica de la fuente. No obstante, en caso de los dispositivos que se van a simular, estos presentan una limitación y es que degradan la respuesta frecuencial de la fuente, actúan a modo de filtros digitales; y esto viene impuesto tanto por el método de grabación como por los soportes físicos de grabación (cilindros de cera, hilos metálicos, discos planos).

Por esta razón ha sido necesario extraer la respuesta impulsiva de los filtros digitales que simulan el comportamiento de los aparatos de audio y para este fin se ha acudido al código proporcionado en Matlab, "archivo.mat" de cada uno de los filtros asociados al equipo de audio y por medio de las siguientes instrucciones, se ha extraído la respuesta ante un impulso.

```
fs=44100;
imp_resp=impz(h,1,[],fs);
filename = 'radio.wav';
audiowrite(filename, imp_resp, fs);
```

Para este fin se ha utilizado la tasa de muestreo de 44.100 Hz, que es la mínima que permite una recuperación de audio analógico en el espectro audible de 20 Hz – 20 kHz. Esta respuesta ante un impulso se guardará en formato .wav y se va a emplear en el nodo de convolución de la interfaz de audio digital que se va a desarrollar en la propuesta de trabajo.

Finalmente, se crea un nodo de convolución para almacenar la respuesta en frecuencia del filtro generado en JS.

```
const convolver = offlineAudioCtx.createConvolver();
```

- Distorsión en amplitud

La distorsión es un efecto común en los dispositivos de reproducción de audio analógicos, por lo que se ha aplicado un algoritmo que lo simule de cada uno de los aparatos de audio. Para este fin se ha utilizado la distorsión de amplitud similar a la que se había aplicado en el trabajo fin de grado de un antiguo alumno de escuela que lo había implementado en C, mediante desarrollo de un método basado en una función sigmoide [7].

En el caso del presente proyecto se había hecho uso de un módulo desarrollado en JavaScript [8], que proporciona un vector de coeficientes de distorsión especificando el parámetro k que permite variar el grado de distorsión.

- Wow y Flutter

Son las variaciones cíclicas de la velocidad de reproducción de los soportes físicos que originan los efectos de "wow" y "flutter" en los dispositivos analógicos. Este fenómeno provoca una modulación en frecuencia, que hace que haya una ligera variación de la frecuencia de muestreo con la que se reproduce el audio.

En cuanto al efecto "wow", este se caracteriza por desviaciones de frecuencia de reproducción del orden de 5 a 10 Hz y es consecuencia de cambios lentos de la frecuencia angular (por ejemplo, en caso de un disco plano que no es perfectamente circular, presenta deformaciones radiales).

En cuanto al efecto "flutter", este se origina por las desviaciones de frecuencia superiores a las de "wow", llegando a ser de orden de 100 Hz – 500 Hz, como consecuencias de variaciones de velocidad de reproducción más rápidas (por ejemplo, debido a la fluctuación de la velocidad angular del mecanismo de arrastre en un mecanismo de transporte de cinta durante la reproducción).

- Ruido

Con el fin de caracterizar completamente los dispositivos de audio se añade el ruido característico de cada aparato, que se ha obtenido a través de los ficheros de audio del proyecto desarrollado en código Matlab que contienen el ruido grabado.

Asimismo, se ha realizado un submuestreo con una tasa de 44.100 Hz para el procesado de las pistas de audio.

3.2 Equipos de audio

En la siguiente sección se describen detalladamente los efectos que conciernen a cada uno de los aparatos, así como las principales características sonoras.

3.2.1 Fonógrafo

"El primer prototipo de fonógrafo fue enunciado por Thomas Alva Edison en 1877. Desde el primer momento constituyó una auténtica revolución tecnológica, dado que se trataba del primer aparato capaz de grabar y reproducir la voz humana.

Visto con la perspectiva del tiempo, el instrumento parece ahora muy elemental. Consta de un receptor, un grabador y un reproductor. El receptor no es más que una bocina de forma cónica en cuyo vértice se sitúa una membrana de metal. Las ondas sonoras entraban por la bocina, que las concentraba sobre la membrana, la cual cumple una función de diafragma, y está unida al centro a través de una aguja. Los movimientos percibidos por la membrana eran transmitidos a la aguja, que registraba la presión sonora detectada sobre la superficie de un cilindro que inicialmente era de papel, pero con el tiempo fue evolucionando hacia la cera y, posteriormente, el celuloide.

Para la reproducción del sonido el sistema funcionaba al contrario. Se introducía en el sistema un cilindro previamente grabado y se hacía que la aguja siguiera estos surcos. De este modo, las oscilaciones eran transmitidas a la membrana del diafragma en forma de vibraciones, y luego convertidas en ondas sonoras y amplificadas por la bocina.

Los primeros fonógrafos presentaban una manivela manual para producir el movimiento del cilindro. Estas manivelas fueron posteriormente sustituidas por motores mecánicos [9].”

- **Flutter**

Puesto que la velocidad del giro del cilindro sobre el estilete del fonógrafo no es idealmente constante, se produce una variación en frecuencia de reproducción de alrededor de 100 Hz.

Este efecto se ha desarrollado mediante las siguientes líneas de código JavaScript (JS):

```
hfo = offlineAudioCtx.createOscillator();
hfo.type = ‘sine’;
hfo.frequency.value = 100;
```

Estás líneas crean un oscilador en la interfaz de audio virtual, el tono modulador en este caso es un seno (el que modulará la variación frecuencial de 100 Hz sobre la portadora del canal audio mono). Para este fin se creará una copia del audio original y se convertirá la pista original en dos pistas estéreo.

```
L_ch = offlineAudioCtx.createGain();
R_ch = offlineAudioCtx.createGain();

panNodeR = offlineAudioCtx.createStereoPanner();
panNodeL = offlineAudioCtx.createStereoPanner();

panNodeR.pan.value = 1; //canal derecho
panNodeL.pan.value = -1; //canal izquierdo
```

En las líneas anteriores se crean dos nodos de amplificación con la ganancia unitaria, también se crean los nodos de paneo en estéreo para el canal derecho e izquierdo, se establecen los valores tales que permiten enrutarlos a los canales respectivos.

```
sound_source.connect(L_ch);
sound_source.connect(R_ch);

L_ch.connect(panNodeL);
R_ch.connect(panNodeR);

panNodeL.connect(vibrato_merger, 0, 0);
panNodeR.connect(vibrato_merger, 0, 1);
```

La pista fuente, es suministrada a los nodos de amplificación, los cuales se conectan a las pistas de paneo, y finalmente estas se fusionan en una pista de audio mono la que finalmente es suministrada al usuario.

- **Distorsión**

Se aplica la distorsión en amplitud con un factor de compresión de 100.

```
distNode = offlineAudioCtx.createWaveShaper();
distNode.curve = makeDistortionCurve(100);
```

- **Filtrado paso banda**

La respuesta en frecuencia del fonógrafo es fruto de la limitación del soporte material usado en grabación y de la bocina que reproduce el sonido. La respuesta en frecuencia suele ser aceptable en el rango de 400 a 2.000 Hz, véase Fig. 8.

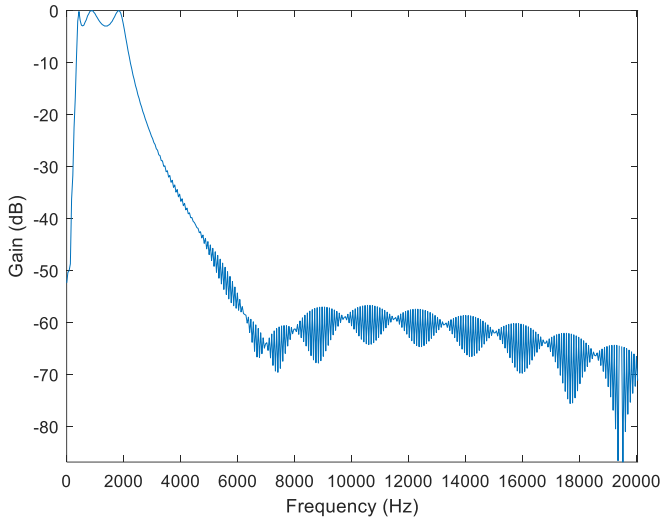

Fig. 8. Respuesta en frecuencia del fonógrafo. Fuente: elaboración propia.

3.2.2 Gramófono

"El gramófono fue patentado en 1888 por Emile Berliner. La característica diferencial de este nuevo instrumento era su capacidad de grabar y reproducir sonido sobre un disco plano, lo que lo distinguía de otros sistemas anteriores como, por ejemplo, el fonógrafo, que utilizaba un cilindro como soporte físico. El gramófono reproducía el sonido con mayor calidad, aunque presentaba la seria desventaja de que los usuarios no podían realizar sus propias grabaciones con música o voces, como sí sucedía con los cilindros del fonógrafo.

El gramófono consta principalmente de un plato giratorio, un brazo movible que sostiene la aguja lectora en el extremo y un amplificador. El disco sobre el que se graba el sonido está fabricado en baquelita lo que, desafortunadamente, hacía que se rompiera con facilidad. El plato giratorio es un soporte plano y circular sobre el que se apoya el disco y que gira a velocidad constante.

Respecto al sistema de grabación de los discos, se trata de un procedimiento analógico según el cual las ondas sonoras son transformadas en vibraciones mecánicas que se transmiten hasta una púa, provocando el trazado de surcos en forma de espiral sobre la superficie de un disco metálico o de cera. Estos discos tienen que ser posteriormente tratados por medios químicos. Con un único molde original podían realizarse miles de copias. Durante la reproducción, el

brazo de la aguja se deja caer sobre el disco y va recorriendo los surcos, produciendo vibraciones mecánicas que se transmiten a un diafragma ubicado en el cabezal reproductor del brazo. En el cabezal reproductor las vibraciones se transforman finalmente en sonido, emitido y amplificado a través de una bocina [10].”

- **Wow**

Se produce este efecto debido a que la aguja suele no encontrarse completamente centrada en la ranura del disco plano, provocando que la velocidad de giro no sea constante. Puesto que los gramófonos habitualmente funcionaban con una velocidad de giro de 33 rpm del disco, para hacer más realista el efecto se ha ajustado el desplazamiento en frecuencia con un seno que modula a la portadora de audio, siguiendo la siguiente relación [2]:

$$f_{mod} = \frac{33\ r.p.m}{60\ s/min} = 0.55\ Hz$$

Ecuación 2. Frecuencia de modulación del gramófono. Fuente: elaboración propia.

- **Distorsión**

Se aplica la distorsión en amplitud con un factor de compresión de 25.

```
distNode = offlineAudioCtx.createWaveShaper();
distNode.curve = makeDistortionCurve(25);
```

- **Filtrado paso banda**

El ancho de banda del gramófono suele ser de 350 a 2.900 Hz, véase Fig. 9.

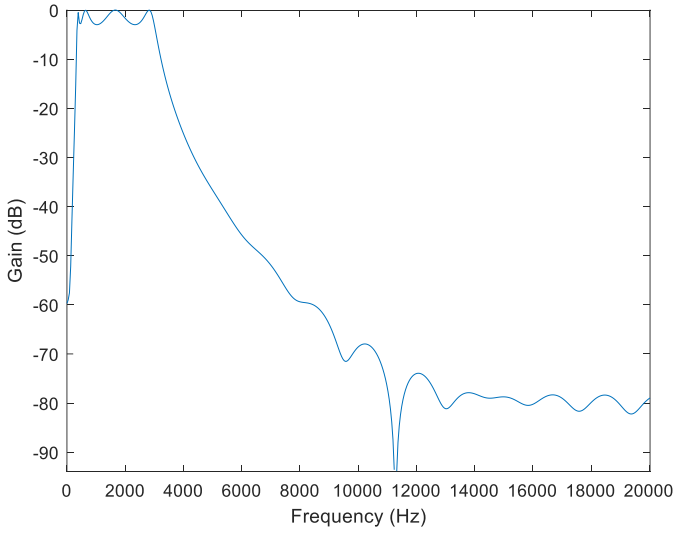

Fig. 9. Respuesta en frecuencia del gramófono. Fuente: elaboración propia.

3.2.3 Magnetófono de hilo

Puede decirse que el magnetófono de alambre tiene el honor de ser el primer aparato de grabación de sonido sobre soporte magnético, a fines de la década de 1890. Fue Valdemar Poulsen, un ingeniero danés que desarrolló una grabadora de hilo magnético. Se basó en las teorías de grabación magnética de Oberlin Smith de 1878. El proceso de grabación y reproducción se puede visualizar esquemáticamente de la siguiente forma en la Fig. 10.

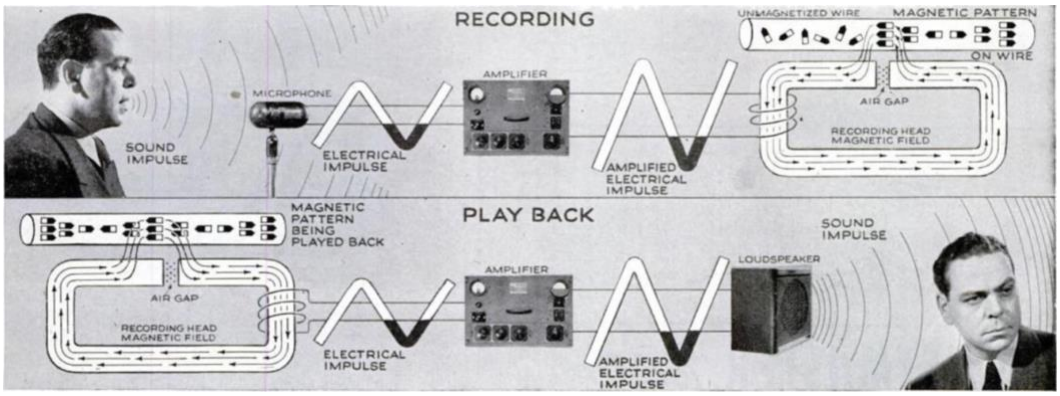

Fig. 10. Funcionamiento del magnetófono de hilo. Fuente: revista científica "Science"

Para grabar una fuente emisora de voz, la onda acústica es transformada mediante un micrófono en una señal eléctrica, esta señal es amplificada y su variación induce un campo magnético que atravesando un gap en el vacío en el material magnetizado crea patrones que se graban mediante la orientación de los dipolos magnéticos.

- **Flutter**

Puesto que la velocidad angular del motor del magnetófono no es constante, se producen pequeñas variaciones de velocidad, estas variaciones son el origen de un desplazamiento frecuencial de 80 Hz. Para este efecto, a diferencia de los otros, se ha desarrollado un algoritmo en JS, tomando como fuente a un método desarrollando en código Matlab propuesto en el libro DAFX – Digital Audio Effects, vibrato.m [11].

- **Distorsión**

```
dist1 = offlineAudioCtx.createWaveShaper();
dist1.curve = makeDistortionCurve(25);
```

- **Filtrado paso banda**

En este caso la respuesta frecuencial tiene un ancho de banda entre 200 a 5.000 Hz, véase Fig. 11.

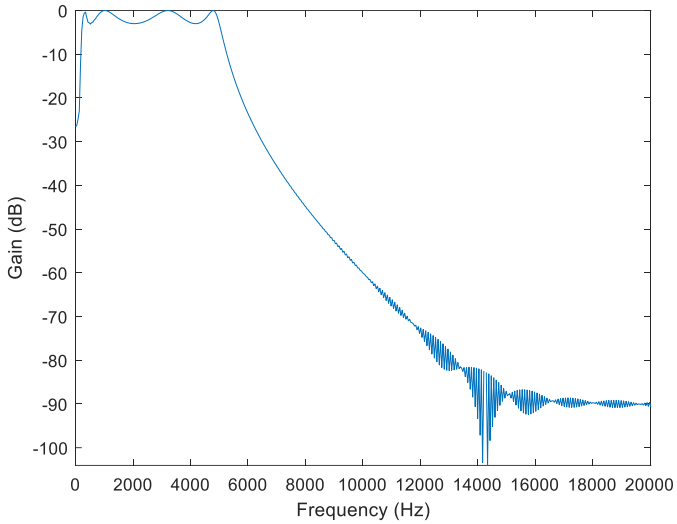

Fig. 11. Respuesta en frecuencia del magnetófono de hilo. Fuente: elaboración propia.

3.2.4 Teléfono de baquelita

Cuando el teléfono de baquelita de Ericsson se distribuyó por primera vez en todo el mundo en la década de 1930, se lo llamó el teléfono de tipo sueco y estableció el estándar de cómo debería verse un teléfono de plástico moderno. La baquelita era el material perfecto para los teléfonos de la época. Básicamente, se pueden moldear de cualquier forma posible, incluso se prefieren las formas suaves y ergonómicas. El material es homogéneo, uniformemente coloreado y duro, con un hermoso brillo. El pase de metal a baquelita ha permitido reducir los tiempos de fabricación, de una semana a siete minutos, tiempo que tardaba el compuesto de moldeado caliente en curarse y entonces los componentes estaban listos para el ensamblaje [12]. La respuesta de estos dispositivos (tanto de su altavoz como su micrófono) está pensada sólo para voz y no para música, así, la banda eliminada de su respuesta empieza en 3.000 Hz [7].

- **Filtrado paso banda**

A continuación, se representa la respuesta en frecuencia del teléfono de baquelita, véase Fig. 12.

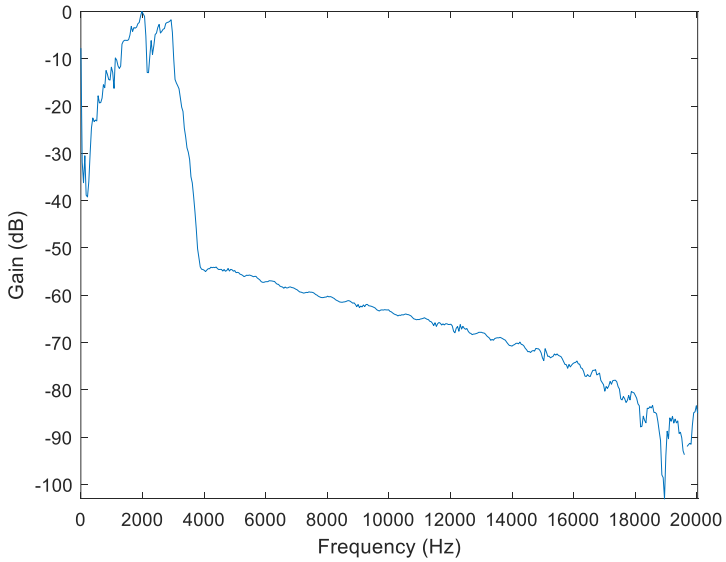

Fig. 12. Respuesta en frecuencia del teléfono de baquelita. Fuente: elaboración propia.

3.2.5 Equipo de radio

"La irrupción de la radio como medio de comunicación tuvo lugar a finales de la década de 1910. En 1916 se instaló en los Estados Unidos de América la primera emisora de radiodifusión, que funcionaba con una potencia de 100 W y una longitud de onda de 360 metros (lo que equivale a una frecuencia de 833 kHz). Estas primeras emisoras utilizaban modulación en amplitud (AM) en la banda de Onda Media (de 530 a 1.700 kHz) u Onda Corta (diferentes bandas entre 2 y 26 MHz). En lo que respecta a España, la radio empieza a emerger en 1924, durante la Dictadura del General Primo de Rivera, concebida como un medio de propaganda política. Se emitía una media de tres horas diarias. Los receptores de radio funcionaban con tubos o válvulas de vacío conocidas vulgarmente como lámparas, una tecnología muy rudimentaria y capaz de generar poco volumen de señal. Las radios solían presentar un disco gradual centesimal para la sintonía, lo que hacía que el proceso de sintonización fuera realmente particular respecto a la selectividad y sensibilidad.

En la década siguiente los aparatos de radio mejoraron. Ahora se conectaban a la red eléctrica y no necesitaban baterías. Eran percibidos como un mueble más de la casa. Entre los diversos modelos que se popularizaron destacan, sin duda alguna, las llamadas radios de capilla, que gozaron de una enorme popularidad.

Los años 40 suponen el final de la Guerra Civil y principio de la 2.ª Guerra Mundial. En esta época el nivel económico del país era muy bajo, sin embargo, hubo una gran demanda de receptores de radio, lo que condujo a la fundación de nuevas empresas fabricantes tales como "Iberia", "Optimus" o "Vica", por citar sólo unos pocos ejemplos. Los receptores usaban un circuito superheterodino de 5 o 6 válvulas, con una recepción mejorada, tanto en selectividad como en sensibilidad.

A finales de los años 50 algunos de los receptores incorporan ya la modulación en frecuencia (FM). La señal FM usa la banda de VHF, de 30 MHz a 300 MHz, aunque la calidad de estas primeras transmisiones todavía debería experimentar sustanciales mejoras. Durante los años 60 se empieza a utilizar el transistor en la fabricación de los receptores de radio, lo que conlleva importantes mejoras. Desaparecen las lámparas y el subsistema de alimentación de alta tensión asociado: un pesado transformador, válvulas rectificadoras, un voluminoso condensador de filtrado. La introducción del transistor aporta multitud de ventajas: tamaños reducidos de los aparatos, bajo consumo de energía, un coste menor y, además la posibilidad de trabajar en baja tensión. Lo que posibilita que los aparatos puedan alimentarse con pilas y ser portátiles [13].”

- **Filtrado paso banda**

El altavoz del receptor de radio presenta un ancho de banda de 200 a 2.000 Hz, véase Fig. 13.

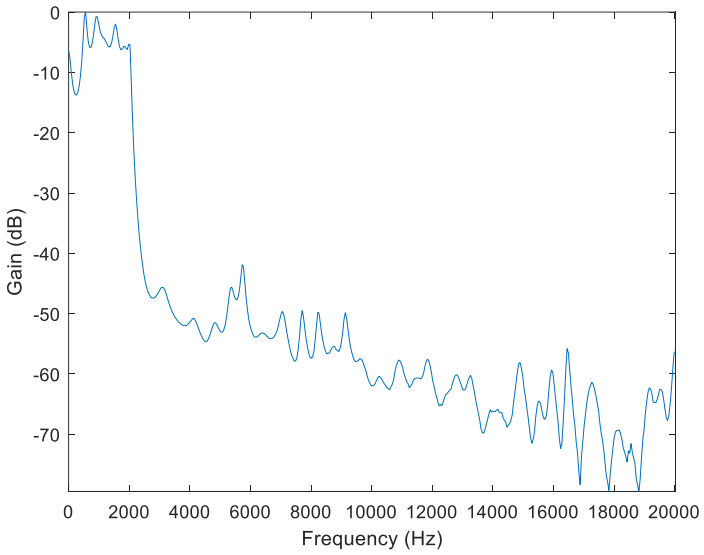

Fig. 13. Respuesta en frecuencia del radio. Fuente: elaboración propia.

- **Código Morse**

Para codificar los mensajes de texto se ha utilizado la tabla de codificación en Morse, que se puede observar en la Fig. 14, un estándar respaldado por ITU-R y que estuvo en uso desde el 1844, [14].

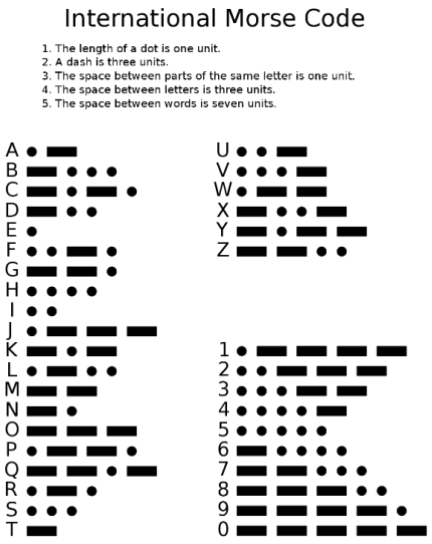

Fig. 14. Código Morse. Fuente: ITU.

Se ha elegido para duración de cada “punto” (unidad) unos 80 ms.

3.3 Herramientas de desarrollo

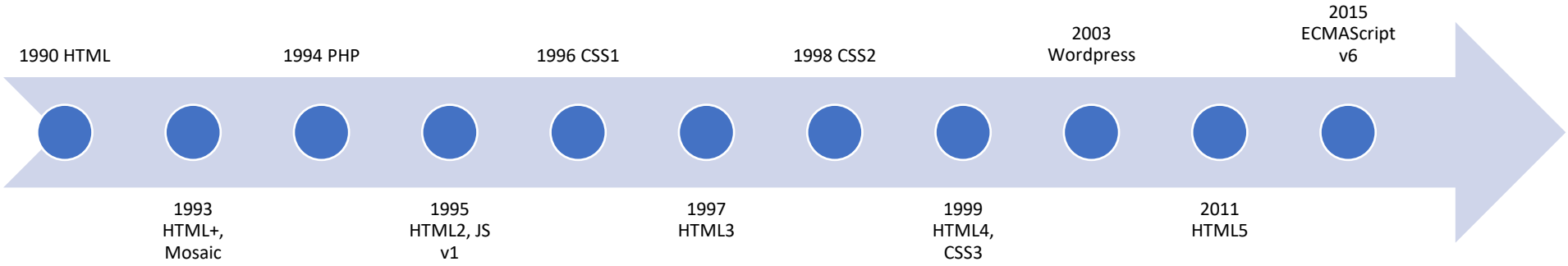

Fig. 15. Desarrollo de las Herramientas de programación en la web. Fuente: elaboración propia.

Antes de entrar a describir el uso de las herramientas utilizadas para crear el programa de audio en la nube, se precisa hacer una breve mención a la evolución de la web, así como a las herramientas principales que se usan para programar el lado del servidor y el lado del cliente de una aplicación web, tal cómo se muestra en la Fig. 15.

En 1989, Tim Berners-Lee (entonces miembro del Laboratorio CERN en Europa) describió su concepto de una plataforma informática que podría facilitar la colaboración entre investigadores que se encontrasen en diferentes partes del mundo, la "World Wide Web" (WWW). Esto condujo a la invención del lenguaje de marcado de hipertexto (HTML) en 1990. HTML se ha desarrollado como un lenguaje fundamental para el desarrollo web estructural de las páginas y forma el núcleo de cualquier sitio web.

Sin embargo, no fue hasta el año 1993, con la introducción del navegador web Mosaic cuando la World Wide Web ganó interés público generalizado. Antes del navegador Mosaic, gran parte del diseño de los sitios web consistía en texto y tablas. Aunque no fue el primer navegador, Mosaic innovó el formato al mostrar texto junto con imágenes de una manera que definió el camino para el futuro del diseño web y sirvió como la aplicación estrella ("killer application") de la WWW. Desde jerarquías de menú hasta fuentes y esquemas de color, el diseño web se convirtió en una forma de arte que combinaba el conocimiento tecnológico con la sensibilidad estética.

Otro hecho importante es que el estándar HTML sufrió su primera especificación formal en 1993. Con respecto a HTML, HTML+ aporta características nuevas como imágenes, tablas y formularios (FIGURE, FORM, INPUT, TEXTAREA, SELECT, OPTION); identificadores de elemento (atributo id), elementos cabecera (h1…h6). Asimismo, se desarrolló una evolución posterior al HTML+ que fue publicada en 1995 [15], HTML 2.0 fue capaz de modelar el cambio hacia el fondo de página, el color del texto, la cara del texto, el uso de tablas y cuadros de texto, etc. En esta época se creó el organismo W3C ("The World Wide Web Consortium"), una organización que desarrolla estándares web actualmente, HTML 2.0 también comenzó a soportar más navegadores web. Las siguientes evoluciones del HTML fueran desarrolladas en consonancia con las recomendaciones de W3C: HTML3 en 1997, HTML4 en 1999 y finalmente HTML5 (2014-1017). Cabe destacar que HTML4 potenció el uso de CSS, lo que permitió modificar fácilmente aspectos como el texto, el color, las fuentes y los fondos. En lugar de que estos aspectos se incluyeran directamente dentro de la página web, ahora estaban separados, lo que permitía su fácil introducción en una página web. En cuanto a HTML5, se puede usar para escribir aplicaciones web que permitan manejar video (tag video), incorporar el audio (tag audio) en la web y ofrecer gráficos extraordinarios (tag canvas) [16]. HTML 5 continúa evolucionando y es compatible con todos los navegadores más importantes, como Firefox, Chrome, Safari, Opera y Edge.

Entre finales de 1994 y 1995 Lie y Bos se unieron para definir un nuevo lenguaje para aplicar de forma consistente diferentes estilos a los documentos electrónicos lo que dio luz a CSS (*Cascading Style Sheets*), un lenguaje de estilo en cascada. En 1995, el W3C decidió apostar por el desarrollo y estandarización de CSS y lo añadió a su grupo de trabajo de HTML. A

finales de 1996, el W3C publicó la primera recomendación oficial, conocida como "CSS nivel 1" que permitía, entre otras propiedades, modificar propiedades de las fuentes, como tipo, tamaño, énfasis; color de texto, fondos, bordes; atributos del texto, como espaciado entre palabras, letras, líneas, etcétera; alineación de textos, imágenes, tablas u otros; propiedades de identificación y presentación de listas, etc. A principios de 1997, el W3C decide separar los trabajos del grupo de HTML en tres secciones: el grupo de trabajo de HTML, el grupo de trabajo de DOM y el grupo de trabajo de CSS. La especificación CSS2 fue desarrollada por la W3C y publicada como recomendación en mayo de 1998. Como ampliación de CSS1, se ofrecieron, entre otras [17]: posicionamiento relativo, tipos de medios de destino (para adaptar el sitio web para los usuarios de móviles entre otros), sombras, etc. A diferencia de CSS 2, que es una gran especificación única que define varias características, CSS 3 se divide en varios documentos separados llamados "módulos". Cada módulo agrega nuevas capacidades o amplía las características definidas en CSS 2, preservando la compatibilidad con versiones anteriores. El trabajo en el nivel 3 de CSS comenzó alrededor del momento de la publicación de la recomendación original de CSS 2. Los primeros borradores de CSS 3 se publicaron en junio de 1999.

PHP Hypertext Processor, comúnmente conocido cómo PHP, fue concebido en algún momento del otoño de 1994 por Rasmus Lerdorf. El código PHP suele ser procesado en un servidor web por un intérprete PHP implementado como un módulo, un “daemon” (es un tipo especial de programa que se ejecuta en segundo plano, en vez de ser controlado directamente por el usuario) o como un ejecutable de interfaz de entrada común (CGI, que permite a un cliente solicitar datos de un programa ejecutado en un servidor web). Las primeras versiones no publicadas se utilizaron en su página de inicio para realizar un seguimiento de quién estaba mirando su página web en línea. La primera versión utilizada por otros usuarios no estuvo disponible hasta principios de 1995 y consistía en un motor analizador muy simple que solo entendía algunas macros especiales y una serie de utilidades que eran de uso común en las páginas de inicio en ese entonces. PHP siguió creciendo en popularidad y se utilizó para crear el popular CMS: Wordpress en el año 2003, que consiste en un sistema de gestión de contenidos multimedia, sobre el que está desplegado el sitio web del museo.

JavaScript se introdujo en 1995 como una forma de agregar programas a páginas web en el navegador Netscape por Brendan Eich, un ingeniero-programador de la empresa. Microsoft, al ver el movimiento de uno de sus principales competidores, también decidió incorporar su propia implementación de este lenguaje, llamada JScript, en la versión 3 de su navegador Internet Explorer. JavaScript no ha dejado de evolucionar desde entonces. Esto contribuyó todavía más a la popularización del lenguaje, pero comenzaron a presentarse pequeños problemas por las diferencias entre implementaciones. Por lo que se decidió estandarizar ambas implementaciones del lenguaje mediante la versión JavaScript 1.1, como propuesta a la ECMA (“European Computer Manufacturers Association”, una organización internacional basada en membresías de estándares para la comunicación y la información), que culminó con el estándar ECMA-262. Este estándar dicta la base del lenguaje ECMAScript a través de su sintaxis, tipos, sentencias, operadores y objetos, y sobre la cual se pueden construir distintas implementaciones. La versión JavaScript 1.3 fue la primera implementación completa del

estándar ECMAScript. Los estándares ECMA han seguido evolucionando, y una de las versiones, en particular ECMAscript versión 6 permite el uso de las promesas. Una promesa permite manejar un valor no necesariamente conocido en el momento en el que la promesa es creada, por lo tanto, permite asociar manejadores que actuarán asíncronamente sobre un eventual valor de la promesa en caso de éxito, o alternativamente en caso de un fallo cuando se devuelva un estado de rechazo.

La web avanzó mucho en los años posteriores al colapso tecnológico de 2000-2001 (conocido como "burbuja puntocom"). Durante este periodo, el gobierno comenzó a desempeñar un papel cada vez más influyente en la web, mientras que, al mismo tiempo, empresas tecnológicas sólidas emergieron de las cenizas del gran colapso para establecer el nuevo rumbo para el comercio y la cultura digital (Amazon, Google, etc.). Y a medida que se establecieron estos cimientos más nuevos y sólidos, Internet se convirtió cada vez más en el principal canal de telecomunicaciones en la era moderna.

A medida que creció la infraestructura de redes y el ancho de banda, la web respondió permitiendo a los diseñadores una variedad de contenido multimedia para incorporar (video, audio), de este modo, diversificando los posibles elementos para añadir en un sitio web, se consigue que la programación web sea realmente flexible.

Las hojas de estilo en cascada permitieron el diseño web con nuevas formas de organizar y mostrar contenido y la transmisión de video cambió la forma en que las personas consumen imágenes en movimiento para siempre. Aun así, con todas estas revoluciones y progresiones en el desarrollo web, la interfaz básica y la estructura de la página web ha mantenido su integridad y equilibrio de forma y función, que está en el lenguaje de marcado HTML.

Paralelamente el desarrollo de las comunicaciones móviles ha abierto una vía amplia de mejora, desarrollo y adaptación de las páginas web para satisfacer las necesidades de los usuarios de móvil. Y en los años transcurridos recientemente, la página web ha cambiado gradualmente de forma para adaptarse a los diversos contextos del uso de las comunicaciones móviles (lo que se denomina un diseño responsivo). Actualmente la aplicación móvil eventualmente podría incluso desplazar a la página web convencional tal como la conocemos actualmente. Pero, aun así, deriva la mayor parte de sus características básicas de los conceptos de diseño de la disciplina del desarrollo web [18].

Por lo tanto, la aplicación de audio en la web se desarrolló en el contexto de integración de la interfaz de usuario con la posibilidad de usar la app en los terminales móviles, incluyendo los últimos avances del lenguaje de programación basado en eventos JavaScript, cómo es la programación asíncrona mediante promesas y el uso de librerías integradas en el navegador para el manejo del audio digital (Web Audio API). Su principal objetivo es proporcionar flexibilidad en cuanto a que permite el uso de la aplicación en cualquier entorno con acceso a internet y accesibilidad mediante el uso de dispositivos móviles o terminales fijos (ordenador, portátil).

Cabe remarcar que, con el fin de hacer pruebas de la aplicación desarrollada, excluyendo la posibilidad de provocar un mal funcionamiento para los usuarios al desplegar una versión del programa errónea en el servidor del museo, se ha hecho el uso de un servidor en local. Este servidor local se ha desplegado a base de un paquete de software libre llamado "XAMPP" (un software multiplataforma, que funciona en cualquiera de los sistemas operativos más conocidos cómo Linux, Windows, Mac, etc.) que incluye un sistema de gestión de bases de datos MySQL, el servidor web "Apache", y los intérpretes de lenguajes "PHP" y "Perl".

Por otro lado, se ha utilizado un editor de código "Visual Studio Code" para editar los algoritmos del programa, ya que permite su fácil visualización con colores y fuentes agradables a la vista, una estructuración del código y trazabilidad ágil a la hora de modificar el código, incorporando la herramienta de control de versiones de los módulos, comúnmente conocida como "Git".

En la siguiente sección de trabajo se describe el desarrollo del programa, que incluye los algoritmos, métodos y funciones implementados que definen el programa de audio en la nube en cuestión.

Capítulo 4. Desarrollo del Programa

En este capítulo se describirá el desarrollo de los algoritmos de procesamiento del audio en la web. En cuanto al desarrollo del programa, se ha decidido programar los algoritmos en JS basándose en los programas ya elaborados por los anteriores alumnos de la facultad. Los algoritmos antes mencionados están recogidos en tres módulos, "script.js", que se usa para hacer llamadas a los métodos de lectura y escritura de ficheros binarios y procesamiento de audio, "audioBuf.js" que hace la lectura y escritura de los ficheros binarios y "audioProcesssing.js" cuyo fin es hacer el procesamiento de audio según el efecto seleccionado por el usuario. Los tres módulos en conjunto integran el programa de procesamiento de audio en el sitio web del museo.

De este modo se describirá el funcionamiento del programa y se comentarán algunas líneas relevantes del código escrito, se hará una descripción por medio de los diagramas de flujo del funcionamiento del programa desde el punto de vista del usuario, así como del módulo "audioProcessing.js". Se comentará el proceso de integración del programa al sitio web del museo mediante un bloque de código en WordPress, así como dos alternativas para integrar el código. Se mencionarán algunos problemas que se han encontrado tras hacer varias pruebas, así como las mejoras propuestas. Se finalizará este capítulo comentando algunas otras alternativas del desarrollo de la aplicación.

4.1 Módulo index.html

Este primer módulo describe tanto la estructura de la página web de audio en la nube cómo el módulo CSS que contiene la información sobre los estilos del contenido.

Las dos primeras líneas indican que se trata de un documento electrónico HTML5 escrito en inglés.

```
<!DOCTYPE html>
<html lang="en">
```

La etiqueta head es un contenedor de metadatos (que describe el contenido de datos). En este caso se trata de un título “index.html”, y una etiqueta meta que contiene varios atributos y en este caso: “meta charset” indica que se trata de un formato de codificación unicode “UTF-8” para codificar el contenido de la página web, “meta http-equiv” especifica un encabezado Hypertext Transfer Protocol (HTTP), un protocolo de capa de aplicación para la transmisión de documentos hipermedia: HTML, CSS, JS, imágenes, audio, etc; en este caso los parámetros "X-UA-Compatible" content="IE=edge" se utilizan para mejorar la compatibilidad del código con Internet Explorer (IE), lo que obliga a IE a utilizar la última versión local actual del renderizado en modo estándar. La línea “meta name = viewport content= width=device-width, initial-scale=1.0” indica como controlar la escala y las dimensiones de la página web, donde “viewport” se refiere a la ventana del navegador que observa el usuario, content ="width=device-width, initial-scale=1.0” establece el ancho de la página para seguir el ancho de pantalla del dispositivo (que variará según el dispositivo) y establece el nivel de zoom inicial cuando el navegador carga la página por primera vez, 1:1.

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>index.html</title>
</head>
```

El bloque caracterizado por la etiqueta body, representa el contenido estructural de la página web de audio en la nube, contiene al elemento “form”, un formulario HTML que una vez rellenado por usuario (en este caso proporcionado el fichero de audio de entrada), debería ser enviado al servidor y/o registrar una entrada en la base de datos; no obstante, al establecerse el objetivo previo de que la aplicación se ejecute en el lado del usuario, se hace un envió asíncrono al navegador del cliente, que mediante los ficheros JS procesa el contenido del fichero de audio. Por otro lado, la etiqueta “fieldset” permite organizar en grupos los campos de un formulario, y mediante el atributo “class” permite aplicar los estilos CSS a todos los elementos que contiene.

```
<body>
    <fieldset class="fieldset">
        <form id="submit_form">
<…>
        </form>
    </fieldset>
</body>
```

A continuación, se estudia el elemento “form” más a fondo, éste contiene las diferentes opciones de procesamiento de audio entre las que el usuario puede escoger, cómo lo podrá observar en la Fig. 16.

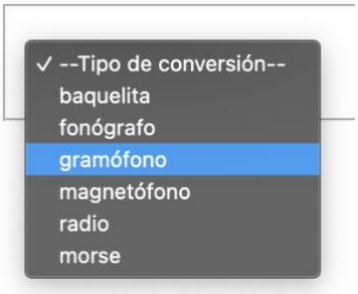

Fig. 16. Recuadro de formulario. Fuente: elaboración propia.

```
<form id="submit_form">

            <p>
                <!-- Elige la tecnología a utilizar -->

                <select id="conversion">

                    <option value="default">--Tipo de conversión--</option>
                    <option value="baquelita">baquelita</option>
                    <option value="fonografo">fonógrafo</option>
                    <option value="gramofono">gramófono</option>
                    <option value="magnetofono">magnetófono</option>
                    <option value="radio">radio</option>
                    <option value="morse">morse</option>

                </select>

            </p>

          <...>

     </form>
```

El párrafo, especificado por la etiqueta “p” y que lleva un identificador “source_audio”, se trata de un bloque que recoge la entrada de tipo fichero de audio y mediante “submit”, proporciona la posibilidad de suministrar el fichero de audio al navegador web por parte del usuario. Aunque la opción “accept = audio/*” aparentemente restringe la entrada de ficheros para permitir el envio solo a los ficheros de audio, el usuario podría insistir en enviar un fichero que no sea audio mediante las opciones, cómo se puede observar en la Fig. 17, por lo que en el módulo script.js se implementará un algoritmo para detectar un archivo suministrado con extensión incorrecta para el procesamiento alertando al usuario de que debe suministrar un archivo de audio con extensión correcta.

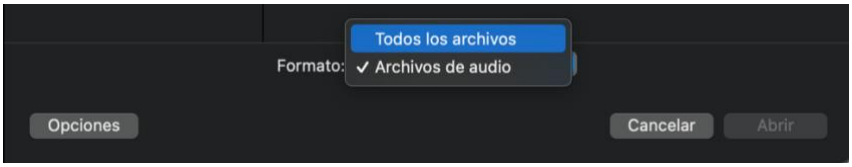

Fig. 17. Opciones de envio de un fichero. Fuente: elaboración propia.

```
<form id="submit_form">

            <p>
                <select id="conversion"> … </select>
            </p>

            <p id="source_audio">
                <input type="file" id="audio_in" accept="audio/*">
                <input type="submit" id="audio_submit" value="Cargar...">
            </p>

          <…>

     </form>
```

El siguiente bloque de código prepara una parte de formulario para el envio de un mensaje de texto con longitud máxima de la frase de 400 caracteres permitidos según la tabla de Morse, la restricción de introducir la secuencia de caracteres permitida se realiza por medio de una expresión regular “^[A-Za-z0-9]+$”.

```
      <form id="submit_form">

                <…>

                  <p id="morse">

                      <!-- No borrar espacio en pattern en [ ]! -->
                      <input type="text" id="morse_text" maxlength="400" required
                        pattern="^[A-Za-z0-9 ]+$" title="solo admite el rango de
                        caracteres a-Z y 0-9">
                      <input type="submit" id="morse_submit" value="Cargar...">

                  </p>
      </form>
```

Finalmente, y antes de cerrar la etiqueta “fieldset”, se especifica otro párrafo que contiene la etiqueta que permite descargar el fichero de audio procesado de salida, cabe remarcar que este elemento se encuentro fuera del formulario, pues se trata de un elemento de salida.

```
                  <fieldset class="fieldset">

                  <form id="submit_form"> … </form>

                          <p id="uploaded">
                              <button id="download"></button>
                          </p>

                  </fieldset>
```

Fuera de la etiqueta “fieldset” se encuentra un elemento “div”, un contenedor cuyo contenido es una imagen y descripción de la opción del procesado utilizado por el usuario.

```
                  <fieldset class="fieldset"> … </fieldset>

                  <!-- Description -->
```

```
<div class="container">

   <div class="imgBx" id="floated"></div>
   <div class="content" id="description"></div>

</div>
```

A continuación, se describen algunas de las líneas de código del módulo de CSS integrado directamente en index.html mediante la etiqueta “style”. El bloque de código a continuación colorea en rojo si el mensaje de Morse a codificar esta incorrecto o está pendiente de introducir.

```
<style>
    <…>
        #morse_text:invalid {
            box-shadow: 0 0 5px 1px red;
        }

        #morse_text:invalid:required {
            border: 1px solid red;
            box-shadow: 0 0 5px 1px red;
        }
    <…>
</style>
```

Finalmente, el último bloque de código del fichero “index.html” indica que se añaden los ficheros JS para poder hacer el uso de los scripts que procesan el audio. El atributo “defer” indica que se cargan los ficheros HTML y JS simultáneamente, pero la ejecución del script se retrasa hasta que todo el código HTML haya sido analizado por el navegador a la hora de cargar la página web. El atributo “src” indica que se trata de un fichero de entrada ubicado en la carpeta contenedora del fichero “index.html”.

```
<html lang="en">
    <body>
        <script defer src="./audioBuf.js"></script>
        <script defer src="./audioProcessing.js"></script>
        <script defer src="./script.js"></script>
    </body>
</html>
```

4.2 Módulo script.js

Éste es el principal módulo que utiliza los algoritmos de procesado implementados en los módulos audioBuf.js y audioProcessing.js. A continuación, se muestra un diagrama de flujo del funcionamiento del programa desde el punto de vista del usuario, véase Fig. 18.

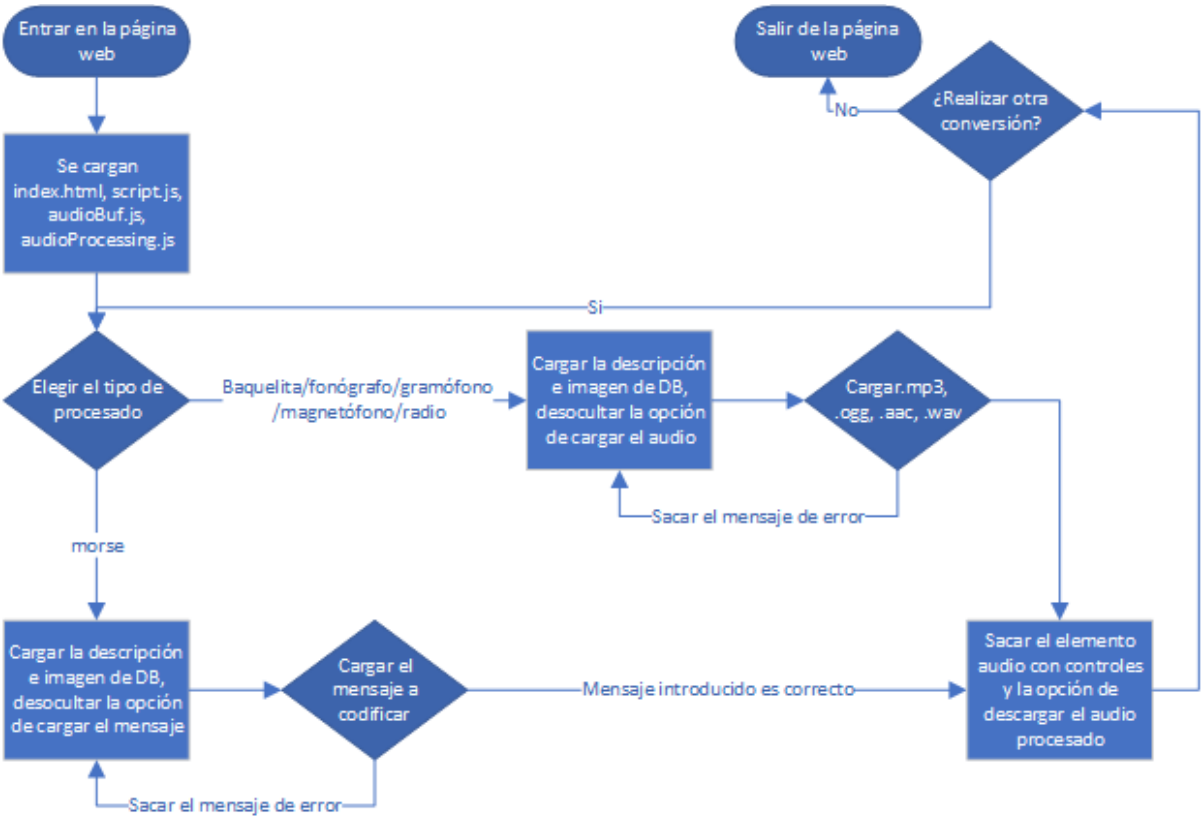

Fig. 18. Diagrama de flujo del funcionamiento del programa. Fuente: elaboración propia.

El diagrama anterior se resume en una serie de pasos que definen el algoritmo del funcionamiento del script.js:

1. El usuario entra en la página web del museo y, a través de un desplegable del formulario elige el tipo de procesado que necesita, véase Fig. 19.

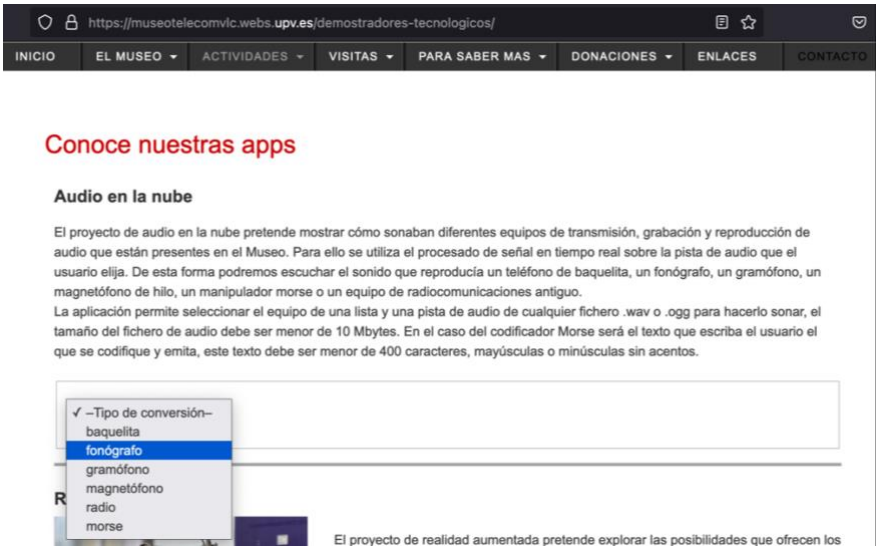

Fig. 19. Vista de la app desde el sitio web. Fuente: elaboración propia.

2. Si el procesado elegido es de alguno de los aparatos (baquelita, fonógrafo, gramófono, magnetófono, radio; entonces se procede a cargar la descripción del efecto seleccionado

y la imagen representativa del dispositivo real, seguidamente, aparecerá la opción de cargar el audio al navegador para que sea procesado a través de los módulos audioBuf.js y audioProcessing.js y sus respectivos métodos. Por el lado contrario, si la opción elegida corresponde a Morse, se cargará tanto la descripción cómo la imagen de una llave de código Morse y se proporcionará la opción de introducir un mensaje en código internacional Morse con una longitud máxima de 400 caracteres, véase Fig. 20.

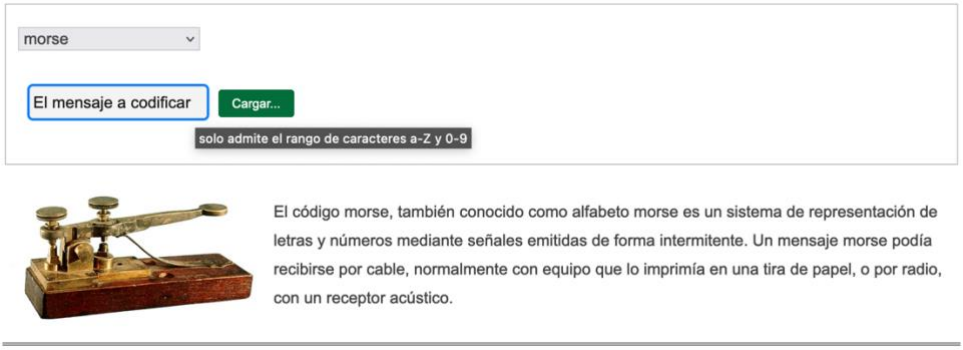

Fig. 20. El formulario, la imagen y descripción del receptor Morse. Fuente: elaboración propia.

3. En cualquier caso, si el usuario decidiese enviar el audio con un formato erróneo (un formato que no sea .mp3, .ogg, .aac o .wav) adjuntando un archivo demasiado grande (que supere 10 MB de tamaño) o enviarse un mensaje que contuviese caracteres incorrectos (guiones, puntos, letras no especificadas en el alfabeto, etc.) se le notificará que cambie de formato de archivo o introdujese los caracteres correctos, véase Fig. 21 y Fig. 23.

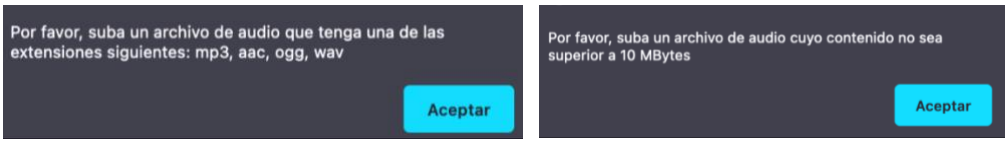

Fig. 21. Formato de archivo enviado erróneo y alerta de error. Fuente: elaboración propia.

4. A continuación, se saca el elemento de audio con los controles de volumen y de reproducción, aparece la opción de descargar el archivo en formato .wav y salvarlo en el dispositivo del usuario en la Fig. 22.

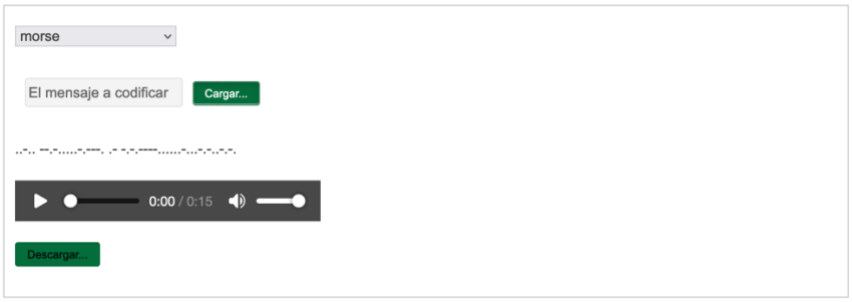

Fig. 22. Los controles de audio y botón de descarga del receptor Morse. Fuente: elaboración propia

5. En el caso de que el cliente decidiese realizar otra conversión, podría proseguir volviendo al paso 1, y repitiendo el proceso seguido; en caso contrario podría abandonar la página web.

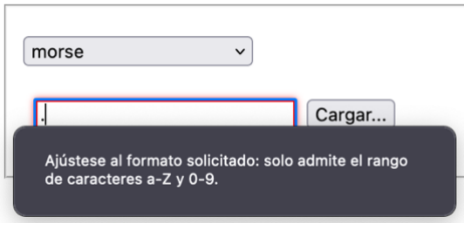

Fig. 23. Mensaje de texto enviado erróneo y alerta de error. Fuente: elaboración propia.

Cabe mencionar que se ha hecho uso del método "preventDefault()" de la librería de métodos y funciones estándar de JS, puesto que permite enviar un formulario sin enviarlo al servidor una vez el usuario pulse "Cargar…", y ejecuta los algoritmos de procesado desarrollados en el lado del cliente (en el navegador).

4.3 Módulo audioBuf.js

Este módulo contiene algunos de los métodos útiles para la transformación de los datos en un array de buffers de datos en binario bruto (cómo es el caso del método "getSourceData"), el método de decodificación de los datos en binario bruto en un array de buffers de datos de audio codificados en PCM de punto flotante de 32 bits, una función que codifica el audio procesado de salida en .wav ("bufferToWave") y una función que genera el nombre del fichero de audio que se guarda en formato "...processed.wav" que sigue al nombre original del archivo: por ejemplo "nombre.processed.wav", o "morse.processed.wav" en caso de la codificación Morse.

Para implementar la codificación del fichero de audio procesado en formato .wav, se ha utilizado un formato definido de la cabecera de un fichero .wav canónico, rellenando los datos necesarios directamente a través de una interfaz de bajo nivel "DataView" para cambiar su contenido, véase Fig. 24.

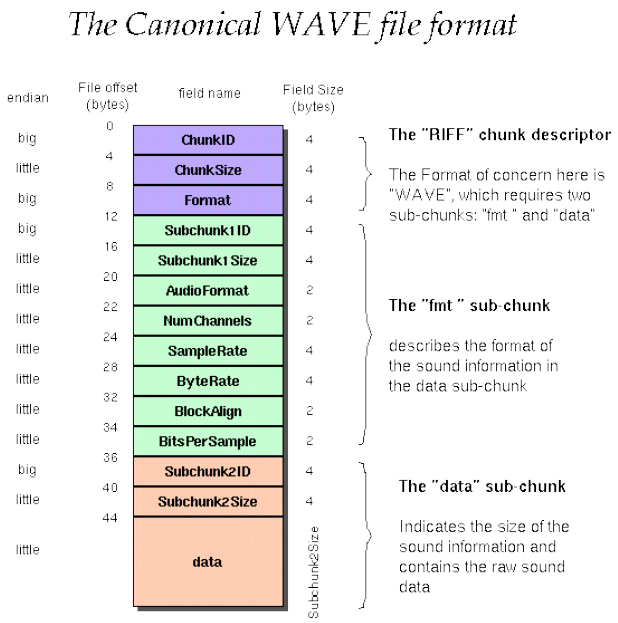

Fig. 24. El formato canónico de fichero .wav. Fuente: "Microsoft", "IBM".

A continuación, se presentan los argumentos que se habían rellenado de la cabecera de la Fig. 23. Se trata de un formato de archivo "RIFF" (archivo genérico para intercambio de recursos almacenados en bloques etiquetados), más concretamente "WAVE", con una longitud del bloque de información de audio de 2 bytes (16 bits), codificado en PCM (no comprimido), en este caso se trata de 1 canal mono, con una tasa de muestreo de 44.1 kHz.

```
setUint32(0x46464952); // "RIFF"
setUint32(length - 8); // file length - 8 bytes (representan los
campos ChunkID y ChunkSize)
setUint32(0x45564157); // "WAVE"
setUint32(0x20746d66); // "fmt " chunk
setUint32(16); // length = 16, represents the size of subchunk
setUint16(1); // PCM (uncompressed)
setUint16(numOfChan); // 1 - mono, 2 - stereo, etc.
setUint32(abuffer.sampleRate); // 44100 Hz
setUint32(abuffer.sampleRate * 2 * numOfChan); // avg. bytes/sec
setUint16(numOfChan * 2); //  2 bytes/sample
setUint16(16); // 16-bit or 2 bytes
setUint32(0x61746164); // "data" - chunk
setUint32(length - pos - 4); // chunk length
```

4.4 Módulo audioProcessing.js

Este módulo contiene las principales funciones de procesado de audio. Se presenta el algoritmo que define este módulo en la Fig. 25.

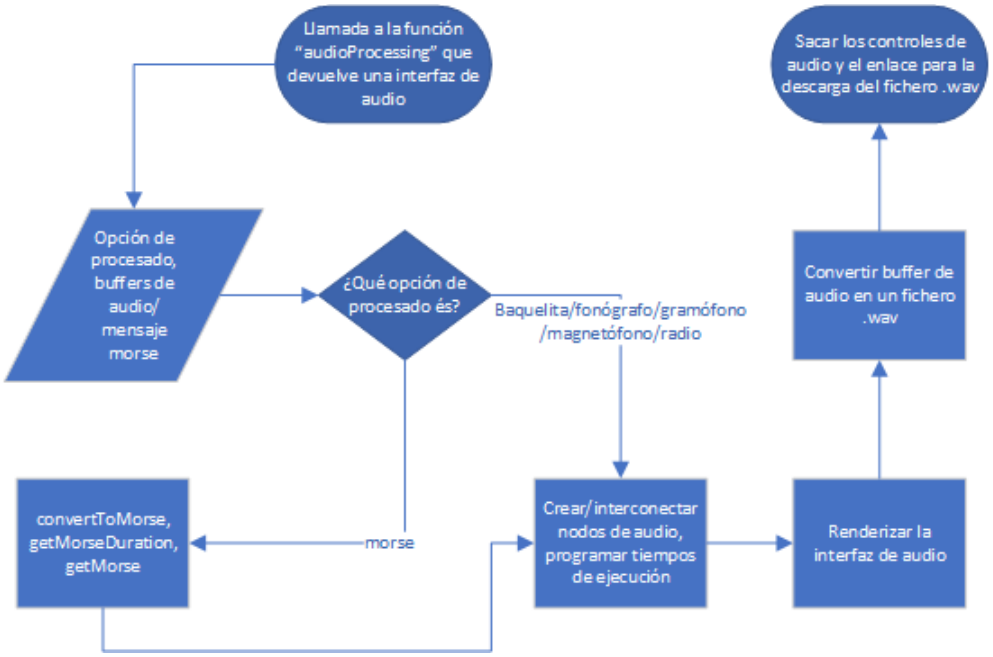

Fig. 25. Flujograma de procesado de un mensaje Morse y/o un fichero de audio. Fuente: elaboración propia.

La llamada a la función "audioProcessing" se realiza a través del módulo principal script.js. Esto se hace una vez el usuario pase la opción de realizar el procesado según el valor enviado en el formulario de la lista desplegable de los efectos; y tras haber decodificado los buffers de trama de entrada, de ruido y de respuesta al impulso en buffers de audio que se pasan como argumentos a la función "audioProcessing".

Una vez recibidos tanto los buffers de audio, mensaje Morse y la opción de procesado; se siguen dos caminos diferentes según el flujograma que convergen a un mismo paso: crear e interconectar los nodos de audio y programar los tiempos de ejecución de una interfaz de audio que define la cadena de efectos y filtros, así como la temporización a la hora de "renderizarlo" y devolver el fichero de audio de salida procesado.

Por un lado, si la opción que había pasado el usuario había sido la de codificación Morse, se hace el uso de tres funciones que lo caracterizan:

- "convertToMorse": con el parámetro de entrada del mensaje a codificar según el alfabeto internacional adaptado; y que devuelve una secuencia de puntos (".") y guiones ("-"). que serán interpretados por la función "getMorse" a la hora de establecer instantes de reproducción para la interfaz de audio.
- "getMorseDuration": con los parámetros de entrada de duración temporal de un punto (la duración del guion es de tres veces el punto y por lo tanto se deduce) la secuencia codificada en guiones y puntos. Esta función sirve para devolver esa duración y en consecuencia para estimar la longitud del buffer de la interfaz de audio (que será necesaria para guardar y reproducir la totalidad del mensaje Morse al "renderizar" dicha interfaz).

- "getMorse": con los parámetros de entrada de duración de trama de entrada (el pitido característico al encender el equipo de radio), la duración temporal de un punto, la cadena codificada en puntos y guiones y el nodo de ganancia se programa la secuencia de reproducción de la interfaz de audio.

Cabe comentar que cada opción de procesamiento de audio que hubiese escogido el usuario de procesado podría llevar a la utilización de distintos nodos de audio, entre los que se destacan:

- Nodo oscilador: es un nodo que permite generar un tono de tipo seno (entre otros), que permite generar un tono de Morse (a 600 Hz) y podría servir para simular el efecto "flutter" (100 Hz) o "wow" (0.55 Hz).
- Nodo de ganancia: permite controlar la ganancia de reproducción o la atenuación de las diferentes fuentes de sonido (audio de usuario, ruido, etc.).
- Nodo de fuente: representa a la fuente de entrada, y por lo tanto se copia a su buffer de audio el contenido de la fuente, que podría ser el audio del usuario, el ruido, etc.
- Nodo de convolución: su contenido es un filtro digital, representado por una respuesta en frecuencia que se había cargado mediante un buffer de audio.
- Nodo de distorsión: representa la distorsión no lineal, especificada por un parámetro que representa la cantidad de distorsión que se quiera alcanzar.

Estos nodos se interconectan entre sí para conseguir elaborar una ruta de procesamiento de la señal de una interfaz de audio. A continuación, se muestra un ejemplo para el caso de código Morse, siendo idéntico al de radio (pues se supone que usan el mismo dispositivo para transmitir tanto el audio, cómo los mensajes codificados).

```
intro_source.connect(intro_atten).connect(output)
oscillator.connect(gainNode).connect(convolver).connect(master).connect(out
put);
noise_source.connect(convolver).connect(master).connect(output);
```

El nodo de la fuente que contiene la trama de entrada se conecta a un nodo de ganancia que acto seguido se conecta a la salida de la interfaz. Paralelamente, el oscilador que simula al tono Morse se conecta a un nodo de ganancia, que se conecta a un nodo de convolución que representa a un filtro o a la respuesta en frecuencia característica del equipo de radio, finalmente se corrige la ganancia de la señal por medio de un nodo de ganancia (se indica como "master" en la cadena de nodos) y se enruta a la salida de la interfaz; ocurre algo similar con la fuente de ruido, que también se conecta en paralelo y se enruta al filtro y acto seguido al mismo nodo corrector de ganancia para finalmente añadirse a la señal de salida.

Siguiendo con el mismo efecto del radio Morse, se comenta la programación de los tiempos de ejecución en la interfaz de audio.

```
intro_source.start(time);
oscillator.start(time + intro_source.buffer.duration);
noise_source.start(time + intro_source.buffer.duration);
```

Si se considera que la variable "time" está inicializada a cero, la primera línea programa la reproducción de la trama inicial desde el instante cero hasta que acabe dicha trama. Acto seguido, se programa que se active el nodo de oscilador para poder emitir la señal de Morse en los momentos oportunos mediante el incremento de la variable "time" en la duración de la trama de entrada anterior, igualmente se activa la fuente de ruido.

Una vez interconectados los nodos, y programados los tiempos de ejecución de diferentes buffers de entrada, se devuelve la interfaz de audio, que queda lista para "renderizarla" y obtener el fichero de audio procesado.

4.5. Las mejoras propuestas

Con el fin de estudiar el rendimiento de la página web se ha hecho el uso de algunas herramientas del análisis del rendimiento que ofrece el navegador "Google" en el panel de inspección del código.

A continuación, se analizarán las posibles mejoras con vistas a ofrecer un mejor servicio web al usuario en cuanto a reducción de tiempo de carga y adaptación a la relación de aspecto de las pantallas de móviles. Dicho proceso de mejoras se podría visualizar a través del diagrama a continuación en la Fig. 26.

Fig. 26. Actualización de las versiones de código. Fuente: elaboración propia.

La primera versión de código consistió en la codificación directa de los recursos multimedia, véase Fig. 27, que recoge: las tramas de entrada, ruido, respuesta frecuencial ante el impulso y las imágenes de los dispositivos. Gracias al uso del codificador de los recursos multimedia

accesible desde la página web “https://base64.guru/” se ha podido codificar a los recursos en Base64 con su posterior decodificación con los algoritmos incluidos en la primera versión.

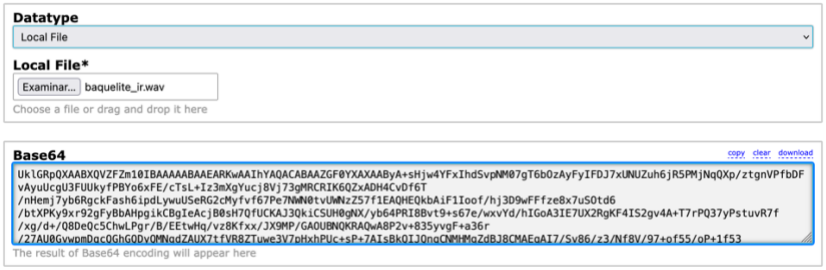

Fig. 27. Conversión del fichero de audio en Base64. Fuente: elaboración propia.

Dicha conversión se ha realizado cómo un mecanismo para no almacenar los recursos en el servidor en absoluto, permitiendo de esta forma no solo ejecutar el algoritmo en el lado del cliente, sino almacenar todos los datos necesarios para el procesamiento. El algoritmo que permite convertir una cadena de texto que contenga la información en Base64 a un buffer de datos binarios se muestra a continuación, siendo el buffer de datos binarios el que luego se usará para la obtención de buffers de datos de audio codificados en PCM de punto flotante de 32 bits, y en consecuencia para el procesamiento de audio.

```
const conv_data = _base64ToArrayBuffer(conv_base64str);

function _base64ToArrayBuffer(base64) {
    var binary_string = window.atob(base64);
    var len = binary_string.length;
    var bytes = new Uint8Array(len);
    for (var i = 0; i < len; i++) {
        bytes[i] = binary_string.charCodeAt(i);
    }
    return bytes.buffer;
}
```

Cabe mencionar, que esta primera versión era ineficiente en cuanto al uso de los recursos de audio (trama de entrada, ruido, respuesta en frecuencia) embebidos directamente en código JS. Esta conclusión fácilmente se obtiene comparando el tamaño del módulo audioBuf.js, de 3 MB. frente a la opción de colocar los recursos en una base de datos del WordPress, en cuyo caso el módulo (sin comprimir) tan solo ocuparía 35 KB; aunque no influya significativamente en tiempo efectivo de procesamiento de la señal de audio, la diferencia principal consiste que en caso de datos codificados en Base64 se gasta un tiempo de 28.5 ms en leer los datos (tiempo de ejecución de la función “getSourceData”) y convertirlo en una matriz con los buffers de datos binarios, Fig. 28, frente a los 2.8 ms en el caso de que los datos se coloquen en base de datos de WordPress, Fig. 29, que sin embargo deberán de ser leídos de la base de datos, como está mostrado en la segunda captura a continuación a través de una petición XMLHttpRequest (petición GET del protocolo HTTP a servidor web del WordPress), en la pestaña “Network”.

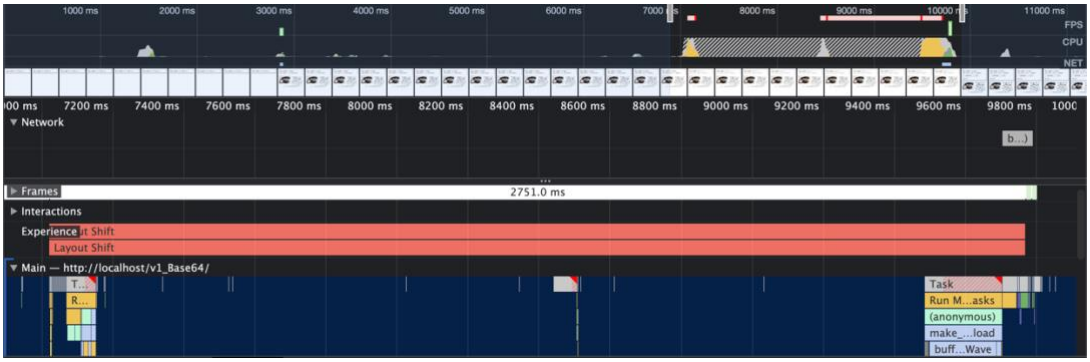

Fig. 28. Captura de rendimiento de la aplicación del audio en la nube con los recursos en Base64. Fuente: elaboración propia.

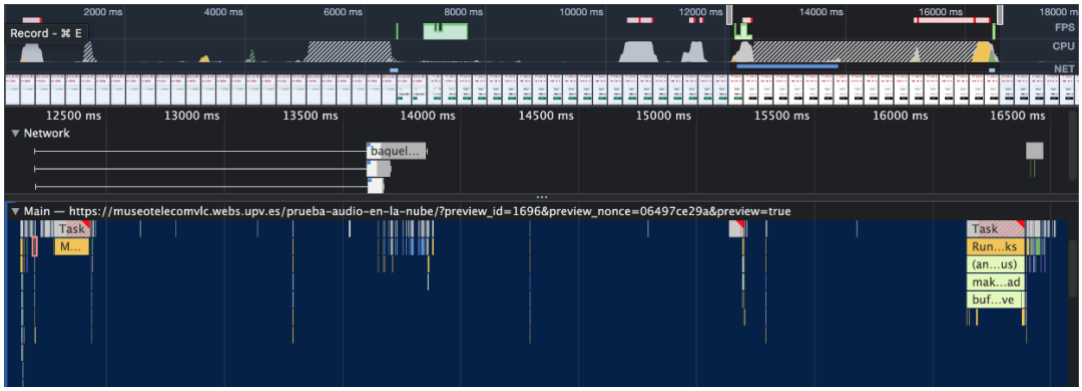

Fig. 29. Captura del rendimiento de la aplicación del audio en la nube con los recursos colocados en una base de datos. Fuente: elaboración propia.

De este modo, cabe mencionar que los recursos de audio, previa colocación en base de datos, se han convertido en un formato de audio .aac con el fin de compatibilizar el uso del programa con el navegador Safari, ya que en el caso contrario se obtenía un error en la decodificación de los recursos de audio, véanse Fig. 30.

Fig. 30. Error de decodificación de los recursos en Safari. Fuente: elaboración propia.

A continuación, en la versión 3 del programa, se ha hecho énfasis en optimizar la apariencia del programa en los dispositivos móviles. Esto es, adaptar mediante código y en concreto instrucciones "media queries" la apariencia en los navegadores desde los dispositivos móviles. El siguiente bloque de código (CSS) especifica que, si no se llega a superar una anchura máxima de 600 px, cómo podría ser un teléfono móvil, se visualiza la descripción del efecto por debajo de la imagen del efecto seleccionado. En caso contrario, la descripción sale junto a la imagen al lado derecho.

```
<style>
```

```
        <…>
            @media (max-width: 600px) {
                .container {
                    margin-top: 1%;
                    width: 100%;
                    display: flex;
                    flex-wrap: wrap;
                    justify-content: center;
                    align-items: center;
                }
            }
        <…>
    </style>
```

Para comprobar que la apariencia en los dispositivos móviles funciona, se ha accedido a la herramienta "Vista del diseño adaptable", y se han seleccionado varios dispositivos cuya anchura sea inferior a 600 px, por ejemplo, un "iPhone 11 PRO"; y se comprueba que efectivamente el diseño se adapta a la resolución de la pantalla, véase Fig. 31.

Fig. 31. Vista desde un dispositivo móvil iPhone 11 PRO. Fuente: elaboración propia.

Cómo otra mejora a mencionar, se ha introducido un bloque de código que permite sacar por la pantalla el mensaje en código Morse codificado, Fig. 32. Se ha decidido que el mensaje cómo máximo será de 400 caracteres minúsculas o mayúsculas sin acentos, pero con espacios.

Fig. 32. Mensaje codificado en código Morse. Fuente: elaboración propia.

Por otro lado, se conoce que los cilindros de cera del fonógrafo albergaban cómo mucho unos cuantos minutos de audio, y por lo tanto se ha decidido dejar un margen de 10 MB. para la carga de los ficheros de audio para cualquier efecto, lo que para un archivo en formato .mp3 con una tasa de bits constante (en caso de aplicar la cuantificación uniforme, sin considerar que haya más niveles para aquellas amplitudes de señal más probables), considerando una tasa de 320 kbps y un canal mono, se obtiene un tiempo de 31.25 s, véase Ecuación 3.

$$t = \frac{10\ MBytes}{320\ kbps} = 31.25\ s$$

Ecuación 3. Duración de un fichero de audio con cuantificación uniforme. Fuente: elaboración propia.

Así también se implementa esta restricción añadiendo las siguientes líneas en el módulo script.js.

```
<script>
<…>
if (input_size > 10) {
    throw "sizeException"
}
<…>
</script>
```

Además, para limitar que el usuario envíe otro tipo de ficheros que no sea audio (.ogg, .aac, .wav, .mp3) a través del formulario, se implementa algoritmos para detectar este comportamiento y en caso de que el usuario fuerce la opción de enviar ficheros erróneos o nocivos a través del formulario, se le alerta de que cambien de fichero (en este caso se comprueba por la extensión del archivo enviado). Se añaden las instrucciones al módulo script.js

```
<script>
<…>
switch (ext) {
    case 'mp3':
    case 'aac':
    case 'ogg':
```

```
            case 'wav':
                break;
            default:
                throw "extensionException"
    }
    <…>
    </script>
```

Cómo otra mejora a mencionar es el uso del módulo vibrato en el procesado de magnetófono descrito en la subsección 3.2.3 Magnetófono, código propuesto en el libro "DAFX – Digital Audio Effects". Dicho procesamiento utilizando en el algoritmo implementado consumía un elevado tiempo, véase Fig. 33, aún para ficheros con una duración de hasta una decena de segundos, alrededor de 1,1 s para procesar un audio de 6 s. Dicho algoritmo, se ha sustituido finalmente por un nodo senoidal de oscilación de 80 Hz que se alimenta a la entrada y produce un efecto auditivo de oscilación (variando la ganancia de la señal, que se percibe a la salida) siendo el tiempo de procesado para el mismo fichero de audio de unos 6.1 ms; una reducción de tiempo de espera del usuario en unos 180 veces aproximadamente, véase Fig. 34.

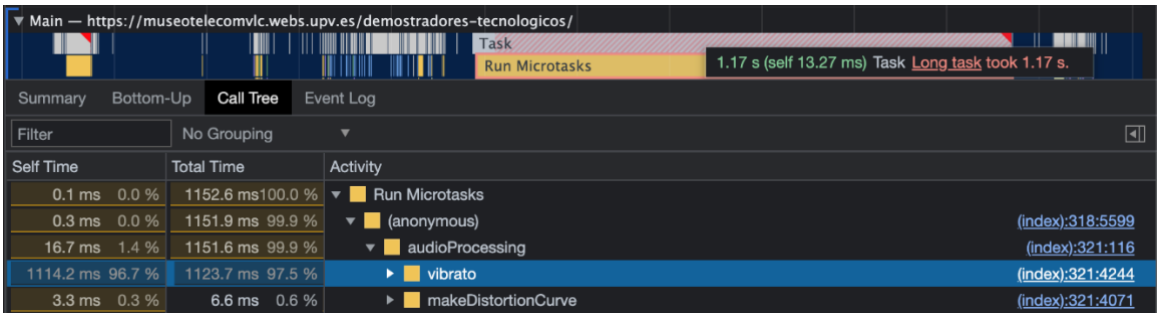

Fig. 33. Rendimiento de procesado de magnetófono con algoritmo de vibrato. Fuente: elaboración propia.

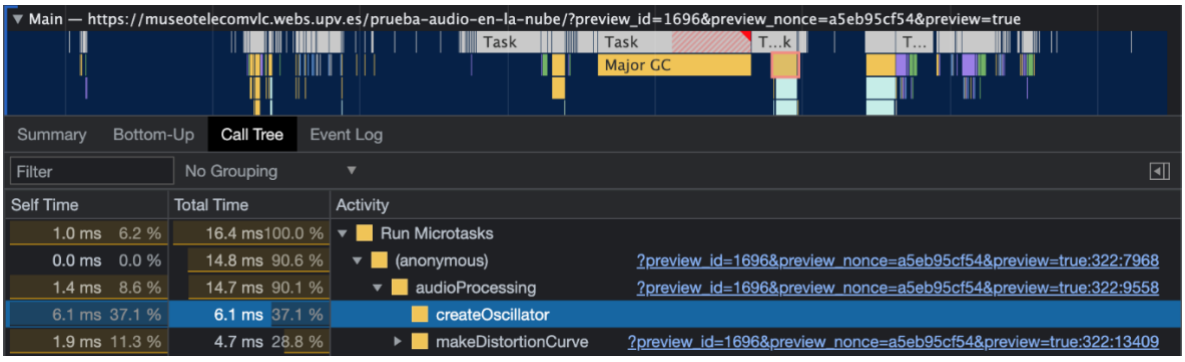

Fig. 34. Rendimiento de procesado de magnetófono con algoritmo de vibrato optimizado. Fuente: elaboración propia.

Finalmente, se ha introducido la compresión de los módulos JS en un único módulo que contenga el mínimo de código para aun así acelerar la carga los ficheros JS. Dicha compresión se ha realizado mediante el sitio web "https://jscompress.com/" que permite comprimir los ficheros JS, en la Fig. 35 se tiene una muestra de ello, ya que permite comprimir casi un 90% del código (incluyendo los comentarios) y por tanto reducir unos 129 KB el tamaño del fichero,

que finalmente ocupa tan solo 14 kB en el servidor; lo que reduciría el tiempo de carga de los algoritmos que hacen el procesado.

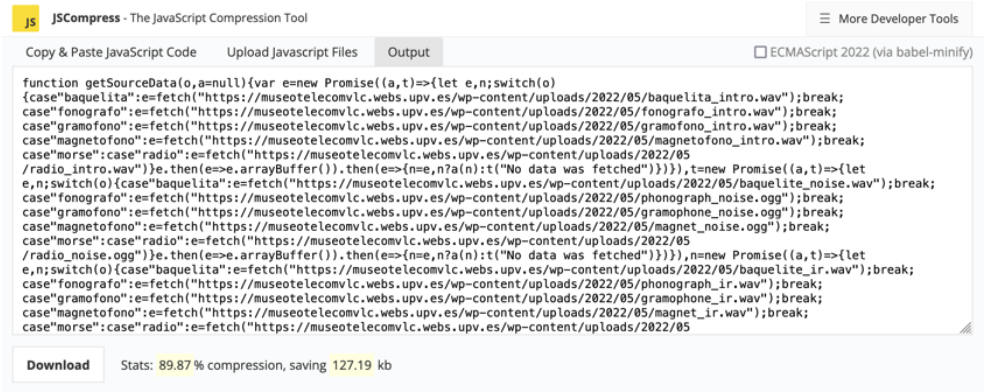

Fig. 35. Compresión de los módulos JS en un único sin comentarios. Fuente: elaboración propia.

4.6 Integración en el sitio web del museo

Antes de entrar a describir el proceso de integración en el sitio web del museo de los módulos desarrollados: index.html, script.js, audioBuf.js y audioProcessing.js cabe destacar que todos los algoritmos definidos en ellos habían sido exhaustivamente comentados, siguiendo unas reglas/etiquetas definidas a continuación, según el estándar [19]:

- Para cada bloque de código relevante (o función/método) podría agrupar varias etiquetas mediante comentarios multilínea. Las líneas de código sueltas se comentan mediante comentarios en una sola línea precedida por doble barra.
- En la cabecera de cada uno de los módulos JS se incluye una breve descripción de la utilidad del módulo y su autoría. Se usa "@file" para especificar el nombre del módulo, "@author" para especificar al autor y "Description" para especificar la descripción.
- Etiqueta "@function" define el nombre de la función que se describe.
- Etiqueta "@param" define que a continuación se describirá un parámetro puede estar seguida de una especificación de que tipo de parámetro se trata encerrada en llaves, por ejemplo: "@param {string}", en este caso el tipo de parámetro es una cadena de texto.
- Etiqueta "@returns" especifica el argumento de salida de una función/método, al igual que "@param" podría especificar el tipo de argumento de salida, cómo por ejemplo "@returns {object}", en este caso el argumento de salida es un objeto.
- Etiqueta "@type" especifica el tipo de objeto que se almacena en una variable mediante asignación, al igual que "@param" podría especificar el tipo de datos de que se trata.
- Etiqueta "@property" especifica la propiedad modificable de un objeto.
- Para una llamada a un método significativo definido en otro módulo, se emplea la etiqueta "Summary", para describirlo, y acto seguido en la etiqueta "Descripion" se pondrá una breve nota: "véase el detalle de la función en módulo X"; se presenta un ejemplo a continuación.

```
/*
* Summary. Recoge de la base de datos de Wordpress: la trama de entrada,
ruido e IR de radio
* Description. véase el detalle de la función en módulo audioBuf.js
<…>
*/
```

- Etiqueta “Link” identifica el enlace hacía la fuente primaria del recurso, en el que se basó el algoritmo implementado

Finalmente se integran los módulos a través de WordPress, en el caso de HTML se accede directamente a la página web utilizando las opciones de edición y añadiendo un bloque de código cómo se muestra en la Fig. 36.

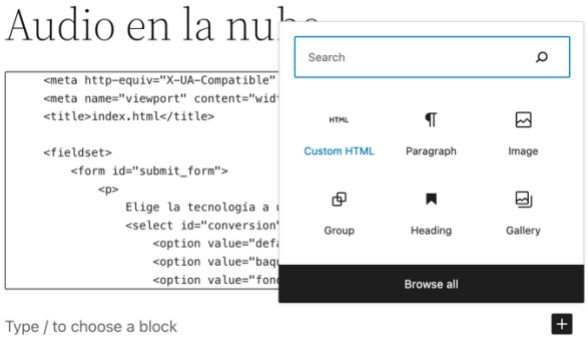

Fig. 36. Incorporación del bloque de código en Wordpress. Fuente: elaboración propia.

Por otro lado, hay otra opción de integrar el código JS en el sitio web de WordPress. Esta opción consiste en integrar los módulos de JS a través de un fichero functions.php ubicado en la ruta dónde se ubica el tema de WordPress usado en el sitio web del museo y por medio de la función “my_script_enqueu” y un identificador de la página web en cuestión, se realiza una llamada a dicha función para que cargue los ficheros de JS en la página web.

```
function my_script_enqueue() {
    if (is_page('29')) {
     ?>
     <script defer src='http://localhost/blog/wordpress/wp-
content/themes/twentytwentytwo/assets/js/script.js'
type="text/javascript"></script>
     <script defer src='http://localhost/blog/wordpress/wp-
content/themes/twentytwentytwo/assets/js/audioBuf.js'
type="text/javascript"></script>
     <script defer src='http://localhost/blog/wordpress/wp-
content/themes/twentytwentytwo/assets/js/audioProcessing.js'
type="text/javascript"></script>
     <?php
  }
}

add_action('wp_footer', 'my_script_enqueue');
```

No obstante, no es la solución idónea para el caso del sitio web de museo, ya que se modificaría un fichero raíz del tema (del servidor), y que supuestamente no se podría modificar explícitamente debido a las políticas del uso de los ficheros ubicados en servidor de la UPV.

4.7 Soluciones alternativas

Finalmente queda por comentar algunas de las soluciones alternativas que se podrían haber adaptado tanto para crear cómo para adaptar el código existente escrito en C y Matlab en vez de diseñar algoritmos directamente en JS. Y esto es el uso del programa "Emscripten", que se caracteriza por ser un compilador de código C, C++ (entre otros) al WebAssembly ("wasm"). "Wasm" es un formato de código binario portable (bytecode), para la ejecución íntegra en navegador de scripts de lado del cliente, se trata de un lenguaje de bajo nivel, diseñado inicialmente como formato destino en la compilación desde C y C++.

De este modo, está solución se valoró desde los comienzos del trabajo, pero debido a múltiples errores de compilación, así como el requerimiento de conocimientos previos del uso de "Emscripten" y del uso de "LLVM", para el manejo de compiladores, se ha considerado adaptar el procedimiento descrito a lo largo del presente trabajo, esto es utilizar la librería integrada de "Web Audio API" y programar algoritmos propios en JavaScript.

Capítulo 5. Análisis y Discusión de los Resultados

En este capítulo se describe la comparativa de los tiempos de ejecución, características sonoras y frecuenciales del audio procesado mediante algoritmos de procesado escritos en C y en JS. Se finaliza este capítulo mediante el estudio de la satisfacción de los usuarios de la aplicación evaluada a través de una encuesta en formato "Google Forms".

5.1 Comparativa con las versiones del programa implementado en C y JS

Por último, se procede a efectuar un breve análisis de tiempos entre las versiones de programa de procesado de audio elaborado en código JS y la versión implementada en lenguaje C, se comparan los espectrogramas de audio procesadas de ambas versiones. Para efectuar dicha comparativa se ha hecho el uso de un clásico de los años 60, "Beatles – Twist and Shout" de una duración de 2:35 minutos, versión en estéreo con una tasa de muestreo de 48 kHz, 16 bits por muestra.

Cómo el efecto de procesamiento se había seleccionado el fonógrafo. Tras un tiempo de procesamiento, que en caso de la versión realizada en lenguaje C, llamando a la función a través del terminal, ocupó unos 2.31 s (en decimal) devolviendo el fichero en la carpeta contenedora del programa; y en caso de la versión en JS, tardó en cargar la página web, procesar y hacer disponible el enlace para la descarga del fichero en unos 2.2 s aproximadamente (no se incluye el tiempo de descarga del fichero de audio procesado), observando el resumen de los tiempos de ejecución de script y de renderizado de la página web en las herramientas de análisis de rendimiento en el navegador de Google. Se obtiene un audio en caso de código en C con tamaño de 15.6 MB, en mono y con una tasa de muestreo de 48 kHz, 16 bits de información por

muestra. En caso del audio procesado por el programa escrito en JS se obtiene un tamaño de 14 MB, 44.1 kHz y 16 bit de profundidad, obsérvese Fig. 37.

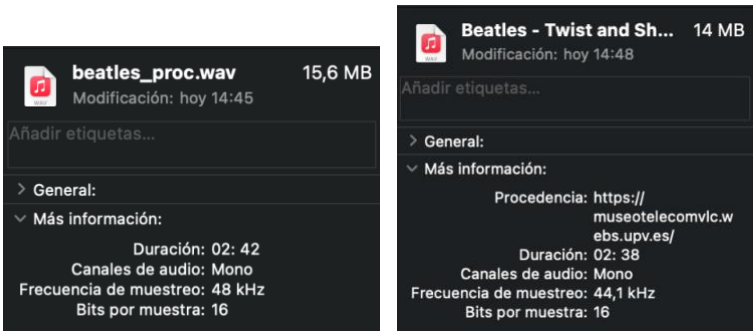

Fig. 37. "Twist and Shout" procesado. Fuente: elaboración propia.

Usando las instrucciones “audioread” y “spectrogram” de Matlab, con un tamaño de ventana de 20 ms, 512 coeficientes de transformada de Fourier discreta (DFT) y un solape entre ventanas de 50% se obtienen los espectrogramas de la señal procesados con el programa en C y en JS respectivamente en las Fig. 38 y Fig. 39.

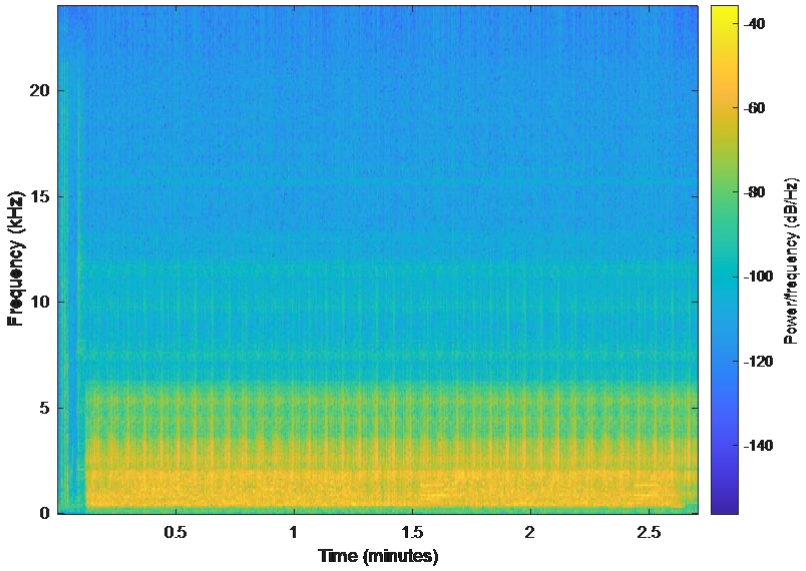

Fig. 38. Espectrograma de "Twist and Shout" procesado por los algoritmos en C. Fuente: elaboración propia.

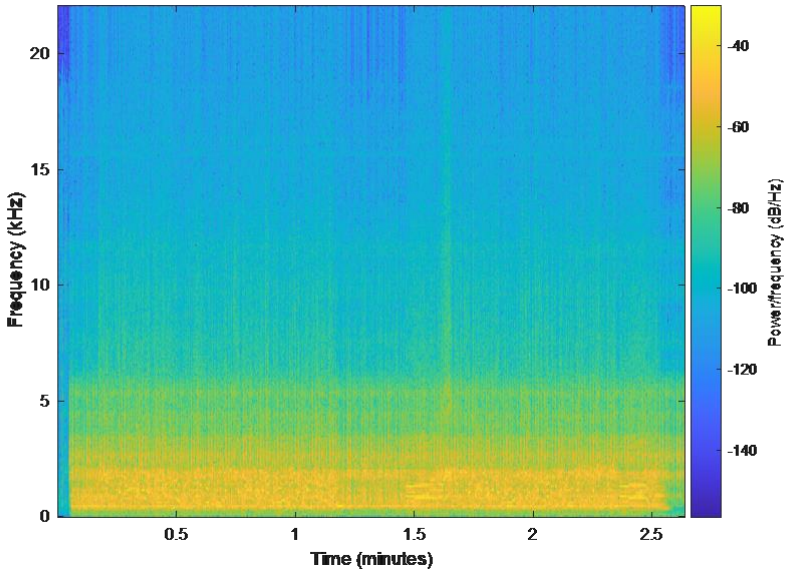

Fig. 39. Espectrograma de "Twist and Shout" procesado por los algoritmos en JS. Fuente: elaboración propia.

Se observa claramente en las figuras anteriores, que en caso del audio de salida del programa en C, se ofrece un audio de salida más nítido, con menor concentración de ruido en las frecuencias medio-altas, en comparación con el audio procesado con algoritmos en JS, se invita al propio usuario de la aplicación a decidir si tanto la versión elaborada por el alumno antiguo de la escuela cómo por el autor de presente procesado en web mediante JS ofrece una representación sonora del dispositivo más o menos fidedigna, así como a subrayar algunas de las ventajas y deficiencias tanto de una aproximación cómo de la otra.

5.2 El diseño de una encuesta de satisfacción de los usuarios

A continuación, se presentan las preguntas y la analítica de las respuestas de los usuarios de la aplicación. En primer lugar, la encuesta se ha dividido en tres secciones: sección de introducción, sección del terminal móvil, sección general.

En la sección de introducción se hace una breve introducción al ámbito de aplicación del programa (procesamiento de audio en la web) y se le pide al usuario unos ocho minutos de su valioso tiempo para completar el cuestionario. La única pregunta que contiene esta sección es la de averiguar si el usuario accede con un terminal fijo (ordenador o MAC) o un terminal móvil (“smartphone”), véase Fig. 40.

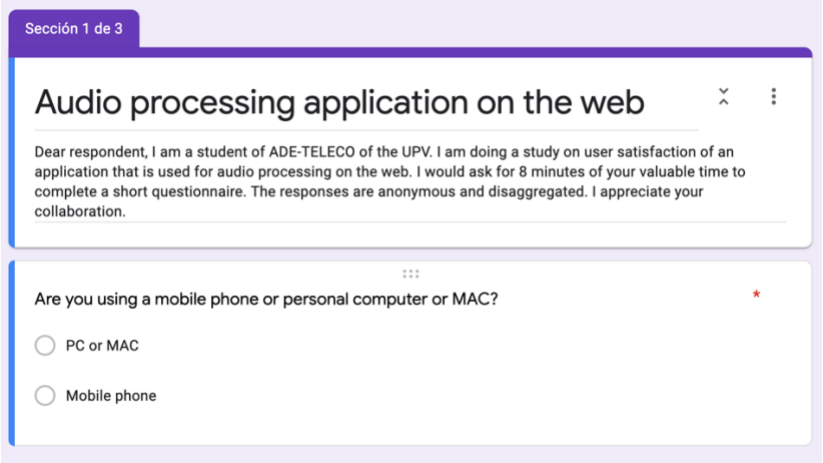

Fig. 40. Sección de Introducción. Fuente: elaboración propia.

En el caso de que el usuario este usando el teléfono móvil se salta a la sección del terminal móvil, que contiene una única pregunta y cuyo fin es averiguar qué modelo del teléfono móvil utiliza el usuario, véase Fig. 41.

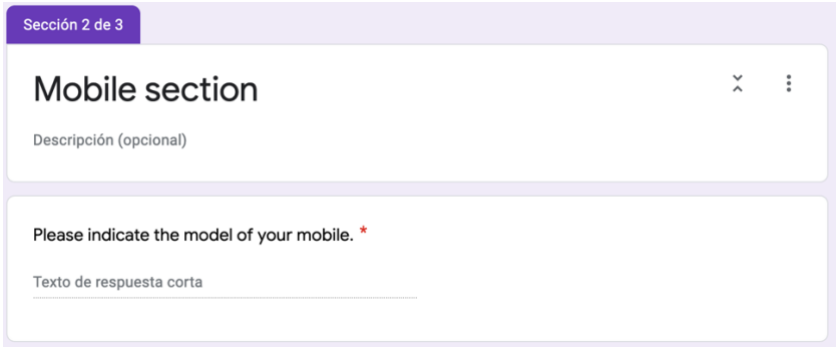

Fig. 41. Sección del Terminal móvil. Fuente: elaboración propia.

La sección general consta de varias partes, por un lado, consta de la descripción del uso del programa para que el usuario pueda entender que acciones deba realizar a la hora de examinar el programa, véase Fig. 42.

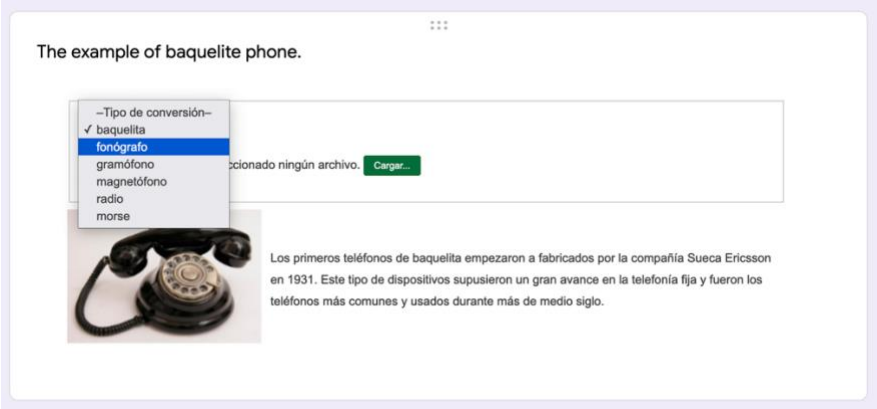

Fig. 42. Muestra del uso de procesamiento de audio, teléfono de baquelita. Fuente: elaboración propia.

Por el otro lado, consta de una serie de preguntas:

- Se averigua si alguna de las aplicaciones había dado algún error cómo lo muestra la Fig. 43.

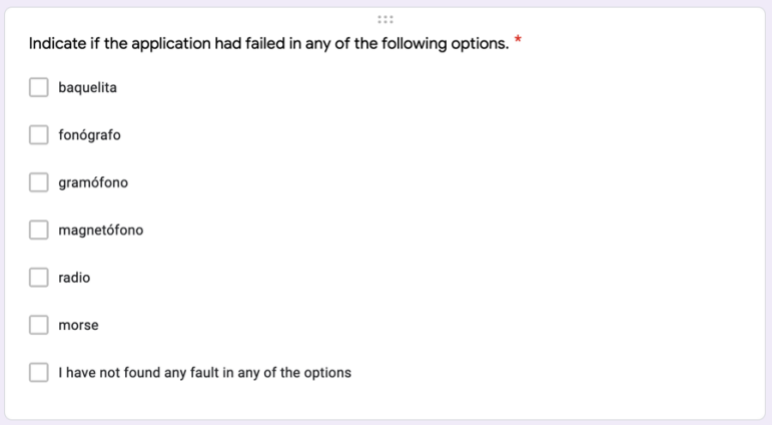

Fig. 43. Consulta sobre un posible error en la aplicación. Fuente: elaboración propia.

- Se pregunta al usuario en caso de que haya algún error y que por lo tanto lo describa, cómo se muestra en la Fig. 44.

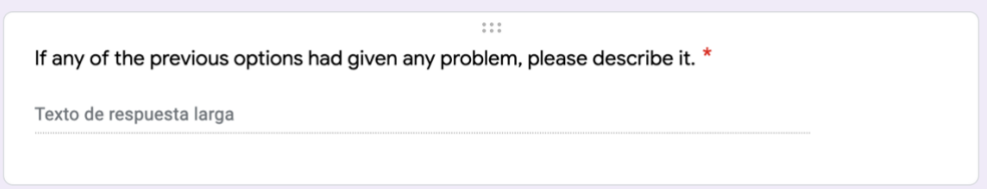

Fig. 44. Un formulario para el envio de la descripción del error. Fuente: elaboración propia.

- Se averigua si el usuario había disfrutado del diseño de la aplicación, si el audio procesado es fidedigno del dispositivo de audio en cuestión, si el procesado había sido rápido, si la experiencia había sido entretenida y si al usuario le gustaría compartir esta experiencia con sus amigos, véase Fig. 45.

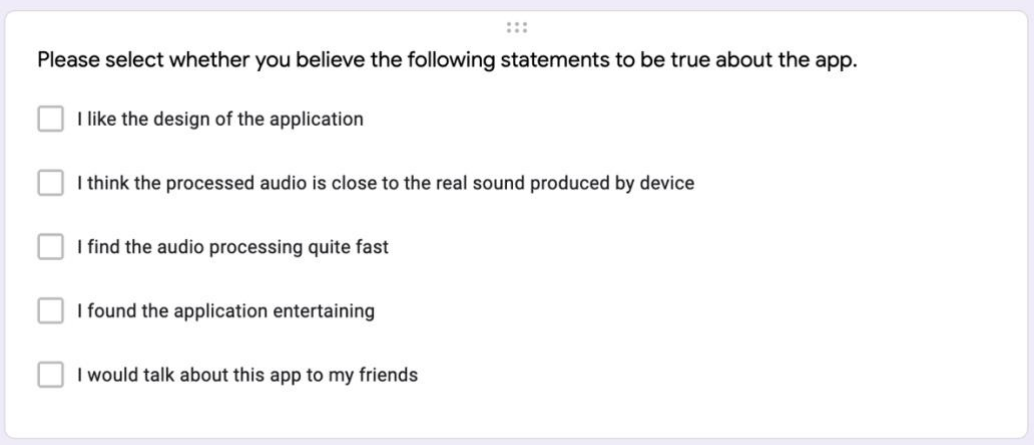

Fig. 45. Consulta de la satisfacción del usuario por parte de alguna de las características de la app. Fuente: elaboración propia.

- El objetivo de la siguiente pregunta, Fig. 46, consiste en averiguar la cuota de los usuarios de los distintos navegadores.

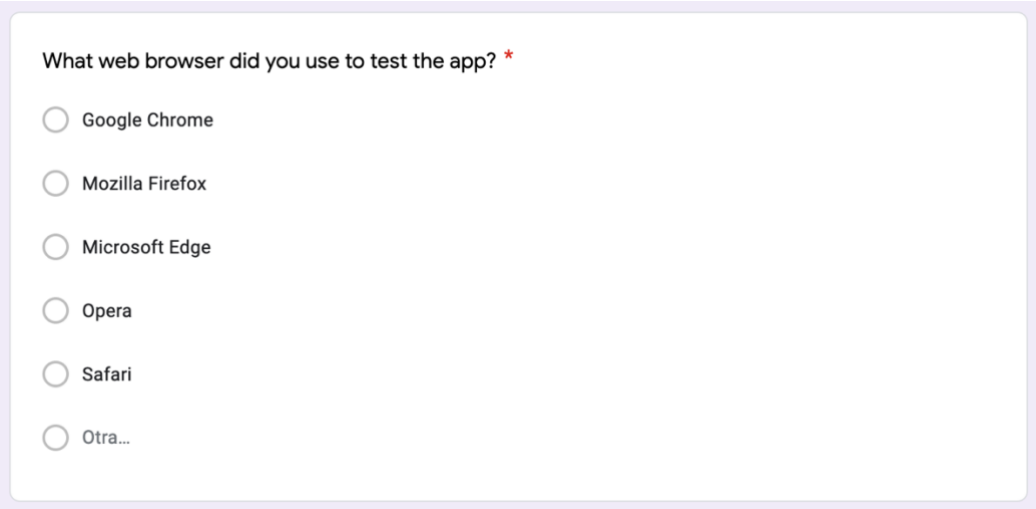

Fig. 46. Cuota de usuarios que usen un navegador particular. Fuente: elaboración propia.

- Finalmente se averigua cuantos de los usuarios habían sido jóvenes de edad, y se despide con los usuarios agradeciéndolos por su colaboración, tal cómo se muestra en la Fig. 47.

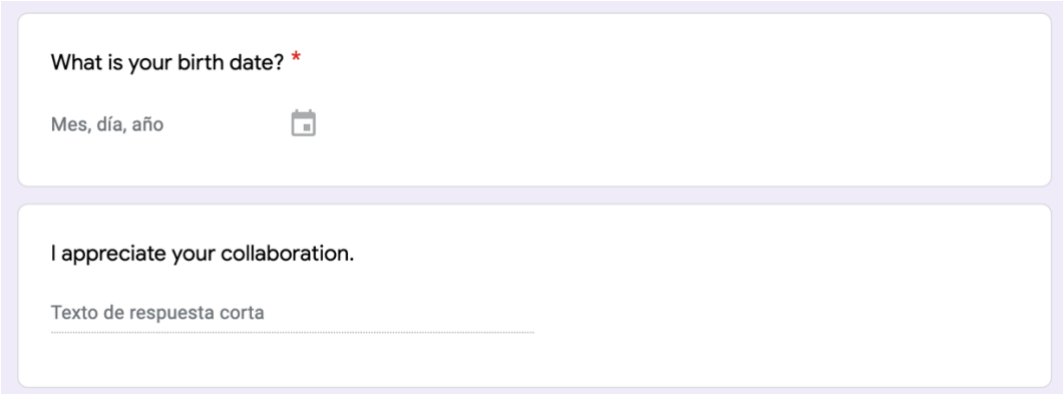

Fig. 47. Consulta de la brecha generacional y mensaje de despedida. Fuente: elaboración propia.

5.3 Análisis de las respuestas

A continuación, se presentan las estadísticas obtenidas de las diez encuestas realizadas. En la Fig. 48 se observa claramente un predominio de terminales móviles en el uso de la aplicación, tal como se había predicho en el capítulo de introducción, el uso de los dispositivos móviles es mayoritario en la sociedad de hoy día.

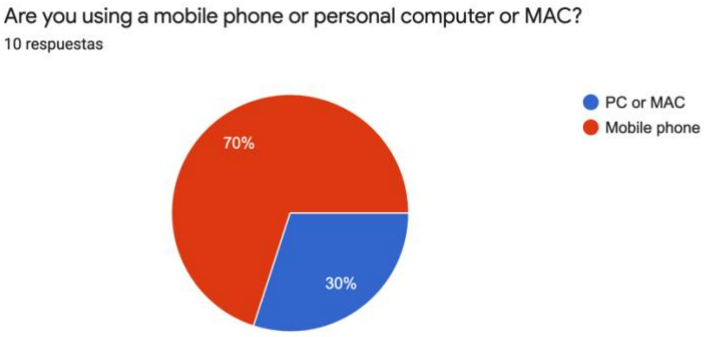

Fig. 48. Terminal fijo/móvil con el que el usuario había accedido a la app. Fuente: elaboración propia.

Son varios los modelos de los terminales móviles que se había usado, véase Fig. 49, entre ellos destacan los teléfonos de la marca "Xiaomi" con un sistema operativo "MIUI".

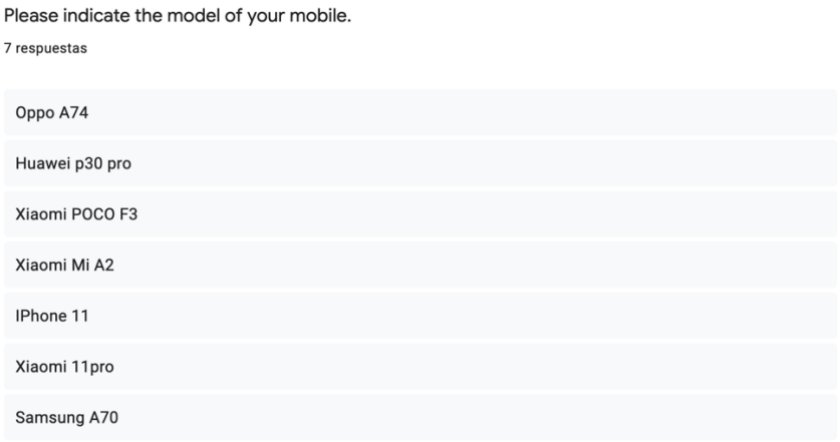

Fig. 49. Marcas de dispositivos móviles. Fuente: elaboración propia.

Se observa en la Fig. 50, que el navegador más popular entre los que se accedió a la web fue "Google Chrome", siguiendo de "Microsoft EdgeDe esta manera, se consigue una distribución del uso de los navegadores web similar al de estudio de mercado realizado por una consultoría tecnológica "Statcounter" [20] del uso de los navegadores web en los años 2021 y 2022, obsérvese Fig. 51.

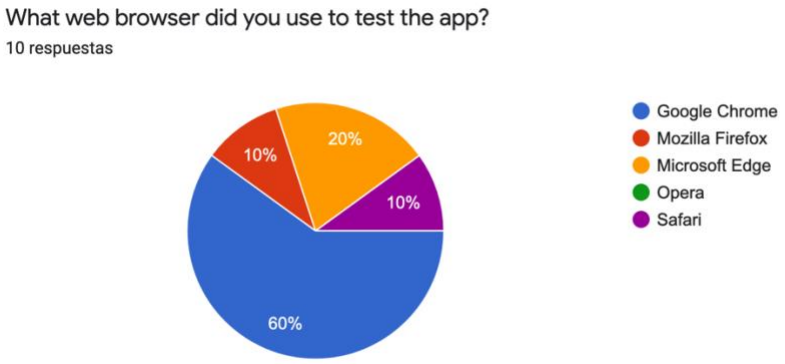

Fig. 50. La proporción de los navegadores web usados por los usuarios. Fuente: elaboración propia.

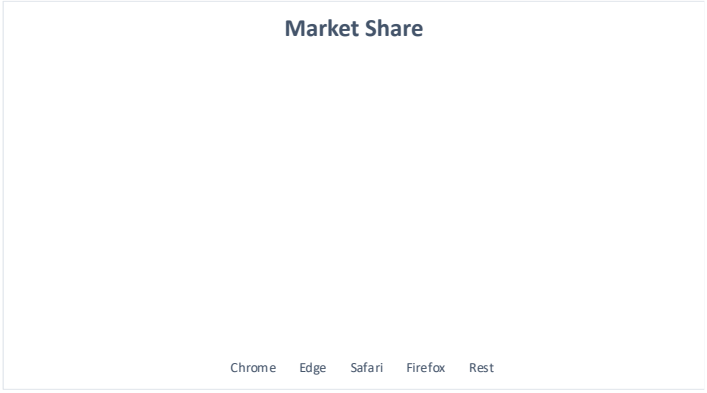

Fig. 51. Los navegadores más usados en 2021-2022. Fuente: "Statcounter"

Se observa en la Fig. 52, que de los diez usuarios que habían probado la aplicación ninguno había detectado algún problema con los efectos.

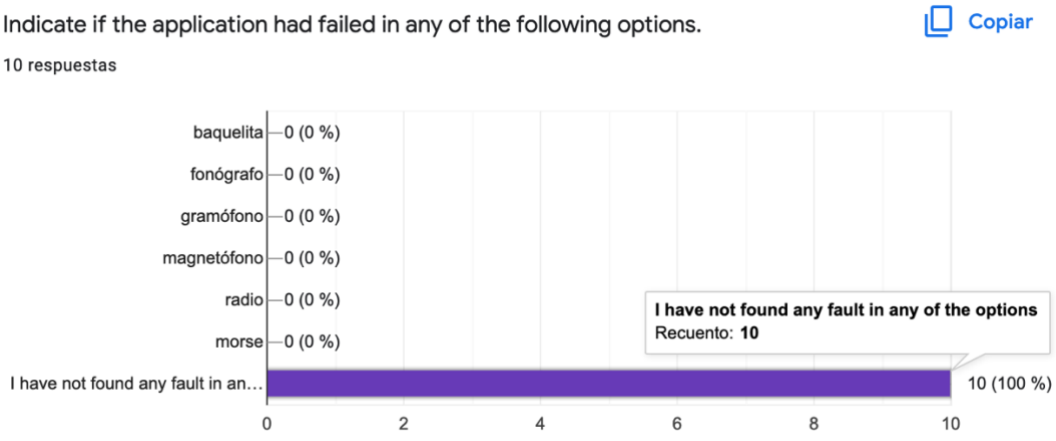

Fig. 52. Errores en alguna de las opciones del procesado. Fuente: elaboración propia.

En la Fig. 53, se observa que a los seis usuarios de la aplicación les gustó el diseño de la aplicación y la habían encontrado entretenida. Los cinco usuarios piensan que el procesamiento había sido rápido.

Por otro lado, los cuatro usuarios que habían probado la aplicación opinan que el audio obtenido se asemeja a las cualidades sonoras de los aparatos y solo dos de los usuarios compartirían sus experiencias con sus amigos.

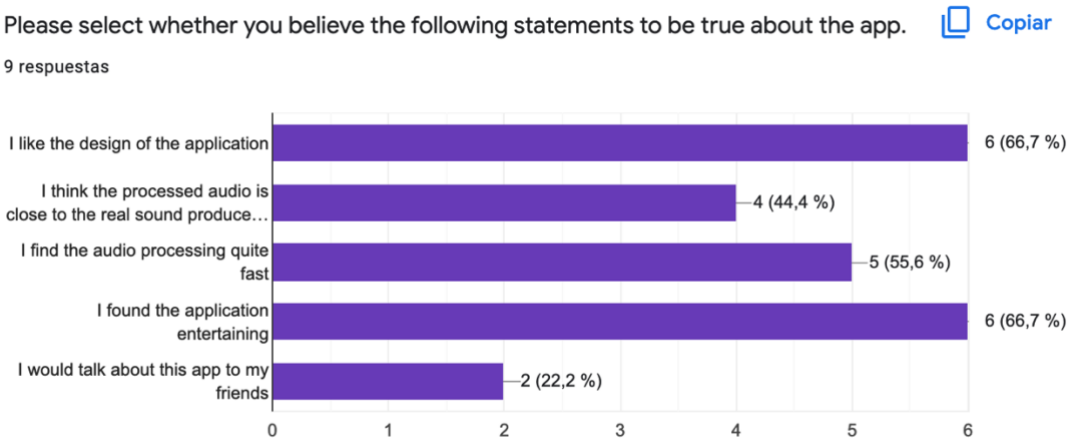

Fig. 53. Las experiencias de los usuarios. Fuente: elaboración propia.

Finalmente, en la Fig. 54, se muestra que la cantidad de los jóvenes (hasta los 25 años) es de cinco personas, hay tres personas adultas entre los 25 años hasta los 40 años, y una persona mayor de 62 años que habían realizado la encuesta.

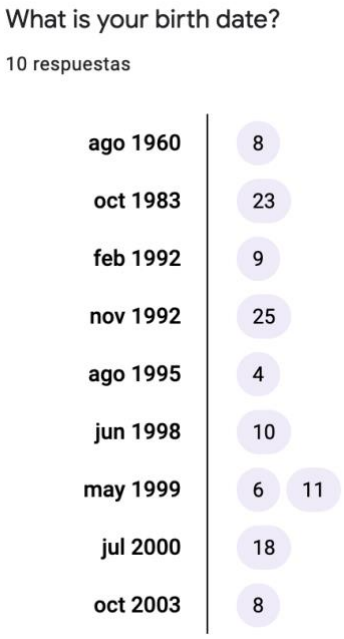

Fig. 54. Brecha generacional. Fuente: elaboración propia.

Capítulo 6. Conclusiones y Líneas Futuras

Cómo ya se ha podido observar a lo largo del capítulo anterior se han alcanzado los objetivos de crear una aplicación segura de procesamiento de audio y simulación de los dispositivos accesible en cualquier momento y que se ejecute en el lado del cliente en cualquiera de los navegadores web principales en las versiones recientes: Google Chrome, Microsoft Edge, Mozilla Firefox, Opera y Safari.

En cuanto a las líneas futuras, el siguiente paso en simulación de las tecnologías antiguas de grabación y reproducción podría consistir en simular virtualmente el procesamiento del video de forma que se viera un fichero de video cómo que se hubiese filmado con un formato cinematográfico “Super - 8”, “película de 35 mm” o se hubiese reproducido en una televisión en blanco y negro o televisión analógica.

Por el otro lado, cómo ya se comentó en la subsección 2.1, el avance de las tecnologías inmersivas VR podría proporcionar una base fundamental para el desarrollo y replanteamiento de las visitas virtuales al museo de ETSIT, adoptando un nuevo paradigma que incluyese una visita virtual interactiva en línea mediante los “avatar” de los usuarios y que además podría aprovechar aquellas aplicaciones que ya se habían desarrollado con anterioridad sobre la web (cómo es el caso de la presente aplicación de audio en la nube), puesto que integran un mismo marco del desarrollo web que son las API de la web moderna.

Bibliografía

[1] Secretaría de Estado de Telecomunicaciones e Infraestructuras Digitales, «Cobertura de Banda Ancha en España en el Año 2020,» 2020.

[2] V. Cases Molés, *Aplicaciones de audio en la nube para el Museo de la Telecomunicación "Vicente Miralles" (Audio on the cloud),* Universidad Politécnica de Valencia.

[3] T. Navarrete, 2019. [En línea]. Disponible: https://repub.eur.nl/pub/118046/REPUB_118046-OA.pdf.

[4] AMETIC, «La revolución digital llega a los museos españoles,» 8 Marzo 2018. [En línea]. Disponible: https://ametic.es/es/prensa/la-revolucion-digital-llega-los-museos-espanoles.

[5] World Wide Web Consortium, «WebXR Device API,» 31 Marzo 2022. [En línea]. Available: https://immersive-web.github.io/webxr/.

[6] Engineering and Technology History Wiki, «Invention of Stereo Sound Reproduction,» [En línea]. [Último acceso: 5 Mayo 2022].

[7] V. Cases Molés, *Aplicaciones de audio en la nube para el Museo de la Telecomunicación "Vicente Miralles" (Audio on the cloud),* Universidad Politécnica de Valencia, p. 7.

[8] K. Ennis, «Stack Overflow,» 10 Mayo 2019. [En línea]. Disponible: https://stackoverflow.com/questions/22312841/waveshaper-node-in-webaudio-how-to-emulate-distortion.

[9] Museo de la Telecomunicación Vicente Miralles Segarra, Universidad Politécnica de Valencia, 2022. [En línea]. Available: https://museotelecomvlc.webs.upv.es/fonografo-2/.

[10] Museo de la Telecomunicación Vicente Miralles Segarra, Universidad Politécnica de Valencia, 2022. [En línea]. Disponible: https://museotelecomvlc.webs.upv.es/gramofono-2/.

[11] U. Zölzer, DAFX: Digital Audio Effects, John Willey & Sons Ltd, 2011.

[12] Ericsson, 2022. [En línea]. Disponible: https://www.ericsson.com/en/about-us/history/products/the-telephones/the-bakelite-telephone-1931.

[13] Museo de la Telecomunicación Vicente Miralles Segarra, Universidad Politécnica de Valencia, 2022. [En línea]. Disponible: https://museotelecomvlc.webs.upv.es/radio-2/.

[14] International Telecommunication Union, «M.1677 : International Morse code,» 26 Noviembre 2009. [En línea]. Disponible: https://www.itu.int/rec/R-REC-M.1677-1-200910-I/. [Último acceso: Febrero 2022].

[15] E. Nebel y L. Masinter, «Internet Engineering Task Force,» 26 Noviembre 1995. [En línea]. Disponible: https://datatracker.ietf.org/doc/html/rfc1867.

[16] University of Washington, «Web Design & Development I,» [En línea]. Disponible: https://www.washington.edu/accesscomputing/webd2/student/unit1/module3/html_history.html.

[17] W3C, «Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification,» 7 Junio 2011. [En línea]. Disponible: https://www.w3.org/TR/CSS21/.

[18] C. Emery, «techopedia,» 5 Mayo 2022. [En línea]. Disponible: https://www.techopedia.com/2/31579/networks/a-brief-history-of-web-development.

[19] JSDoc, «Wordpress Developer Resources,» 2017. [En línea]. Disponible: https://developer.wordpress.org/coding-standards/inline-documentation-standards/javascript/.

[20] «statcounter.com,» 6 Abril 2022. [En línea]. Disponible: https://gs.statcounter.com/browser-market-share/desktop/worldwide.

Índice de Figuras

Índice de Ecuaciones