ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA DE SOFTWARE

Curso Académico 2022/2023

Trabajo Fin de Grado

OPENVIDU CALL EN VUE

Autor: Vanesa Reina Hernández

Director/Tutor: Micael Gallego Carrillo

Agradecimientos

Este trabajo no habría sido posible sin la ayuda de mi tutor Micael Gallego Carrillo quien siempre estaba disponible y me ha apoyado durante todo el desarrollo guiándome para que no me perdiera ante este trabajo de gran envergadura.

Además, agradezco los esfuerzos continuos del desarrollador de OpenVidu, Carlos Santos, por las innumerables preguntas que le lanzaba en el correo y que siempre acababan teniendo solución.

Finalmente, agradecer la ayuda ofrecida por mi compañero de la carrera, Juan Carlos Moreno García, por las horas gastadas cuando Docker daba bastantes más problemas de los esperados y me ayudó a comprender mejor esta tecnología.

Resumen

Este trabajo de fin de grado pretende crear una versión de OpenVidu Call sustituyendo el frontend de Angular por Vue e implementar las funcionalidades básicas de la aplicación. Estas incluyen: inicio de sesión, selección de nombre de sala de llamada, selección de opciones, entrada en llamada y poder desconectarse, compartir pantalla y manejar la salida de micrófono y cámara en medio de la llamada.

Este proyecto nació por la curiosidad por otras tecnologías frontend diferentes de Angular y con la idea de que esta aplicación sirva para que aquellos interesados en OpenVidu y en Vue tengan fácil acceso a una web que use esas dos tecnologías. Además de servir como base para ampliarla en el futuro añadiendo otras funcionalidades que posee OpenVidu Call más avanzadas como grabar las llamadas y la opción de chatear dentro de la llamada con otros usuarios.

En el siguiente documento se encontrará la información pertinente al trabajo, concretamente en el capítulo 1 se hablará del contexto de la aplicación y de los diferentes servicios que compone la plataforma de OpenVidu. En el capítulo 2 se tratará concretamente de los distintos objetivos dispuestos para la creación de esta aplicación. Mientras que en el capítulo 3 se explicarán cada una de las librerías, tecnologías y herramientas usadas para la implementación de este proyecto entrando más en detalle en aquellas que hayan tenido mayor importancia, al igual que la metodología seguida durante el desarrollo. En el capítulo 4 se explicará el groso del proyecto, comenzando por los requisitos que se quisieron abordar y completar, junto con aquellos que se añadieron durante el transcurso del desarrollo, inclusión de algunos diagramas que ayudarán a entender el funcionamiento de la aplicación, explicación de distintos problemas encontrados a lo largo del desarrollo y cómo se solucionó, así como los aspectos más destacables del trabajo, las pruebas realizadas y el despliegue en Docker. Finalmente se cerrará el documento con una reflexión sobre el proyecto y comentar las distintas posibilidades de ampliar el mismo a futuro.

ÍNDICE

CAPÍTULO 1 Introducción y Motivación

OpenVidu [1] es una plataforma para la creación de aplicaciones de videoconferencia que permite a los usuarios la total customización de la misma. Cuenta con una aplicación llamada OpenVidu Call que se creó a partir de la plataforma de OpenVidu, su backend se encuentra implementado en Java [2] o Node [3] (según la preferencia del usuario) y el frontend en Angular [4], framework conocido por su velocidad y capacidades para crear páginas web dinámicas.

De este modo, existen tres maneras de utilizar OpenVidu: mediante la aplicación de OpenVidu Call, incrustando un componente web especial basado en OpenVidu Call en nuestra aplicación o controlando completamente los elementos, implementando desde el principio OpenVidu en nuestra aplicación. Esta última opción es la que se ha usado en este proyecto.

El objetivo de este trabajo de fin de grado es aprovechar todas las herramientas disponibles de OpenVidu y parte de OpenVidu Call para trasladar las funcionalidades básicas que posee OpenVidu Call a otro frontend de un framework diferente, en este caso será Vue [5]. Se quiere recalcar la parte de "básicas" puesto que la aplicación de OpenVidu Call cuenta con características avanzadas e implementar todas sería una enorme tarea para el alcance de un solo trabajo de fin de grado.

Para comprender mejor el funcionamiento de la aplicación y de sus funcionalidades, a continuación, se explicarán las mismas divididas por las diferentes secciones que posee:

- **Login:** En la Imagen 1 se puede ver la primera página que se encuentra un usuario al iniciar OpenVidu Call. En ella el usuario solo puede realizar dos acciones, pulsar el logo de OpenVidu para dirigirse a la página principal de OpenVidu o introducir las credenciales para iniciar sesión. Dichas credenciales han de ser "admin" y "MY_SECRET" obligatoriamente para acceder al resto de pantallas. Esta página no será visitada por el usuario si ya inició sesión anteriormente y no cerró la misma, ya que automáticamente aprovechará los datos guardados para iniciarla.

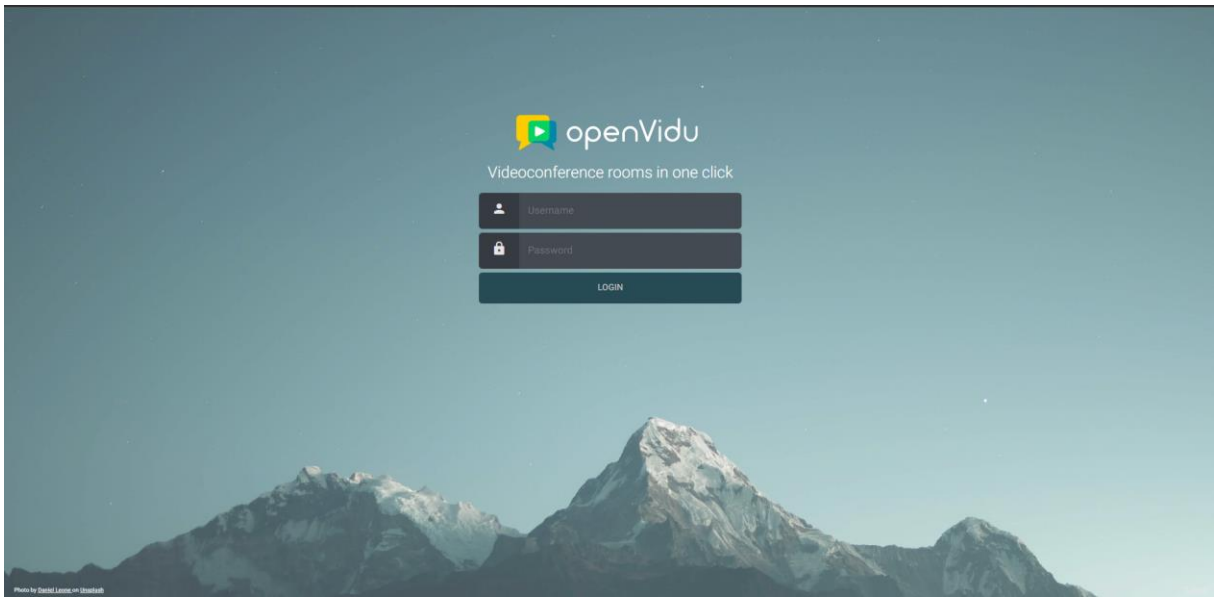

Imagen 1. Página Login OpenVidu Call. OpenVidu Call.

- **Join:** En la Imagen 2 se puede ver el diseño de esta página. Su función es la de permitir introducir el nombre de sala de llamada para que el usuario se pueda unir a la misma o crear una nueva, para ello se puede introducir a mano o generar un nombre aleatoriamente. Además, también se puede cerrar la sesión de usuario, borrando los datos referentes a la misma del navegador.

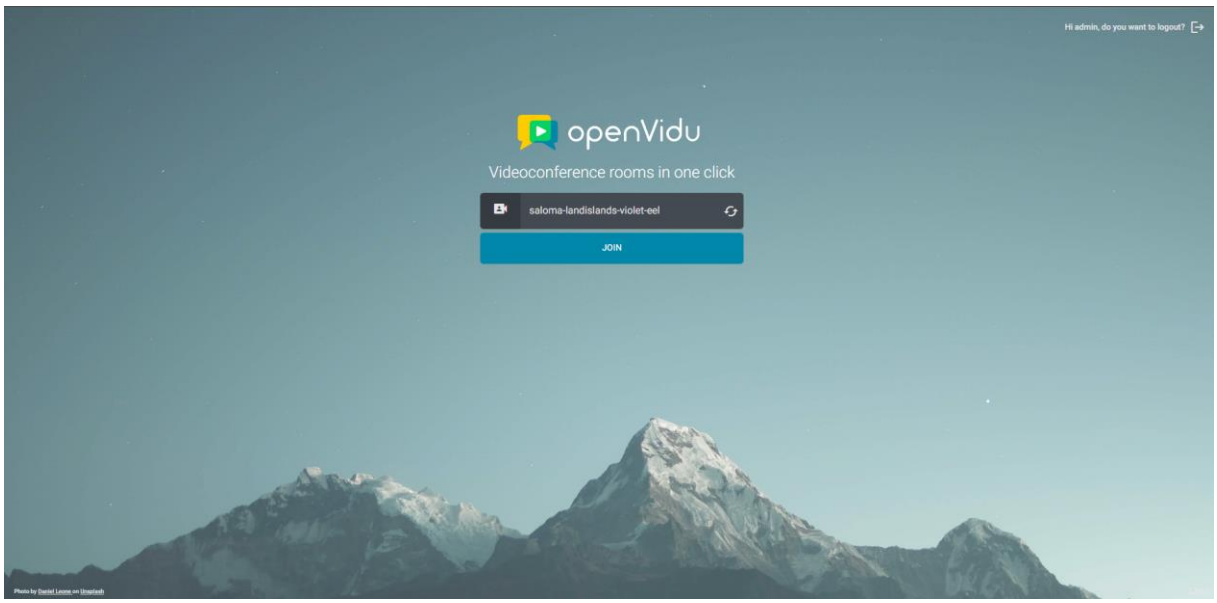

Imagen 2. Página Join OpenVidu Call. OpenVidu Call.

- **JoinSession:** Una vez se decide el nombre de sala se accede a la página mostrada en la Imagen 3 en donde se podrán acceder a diferentes opciones antes de entrar en la llamada en sí. En ella se puede decidir con qué dispositivos se desea entrar a la misma (cámara y micrófono), el si estos mismos se encuentran bloqueados o silenciados en el momento de entrar, escoger nombre de usuario dentro de la llamada, seleccionar filtros e incluso cambiar el idioma.

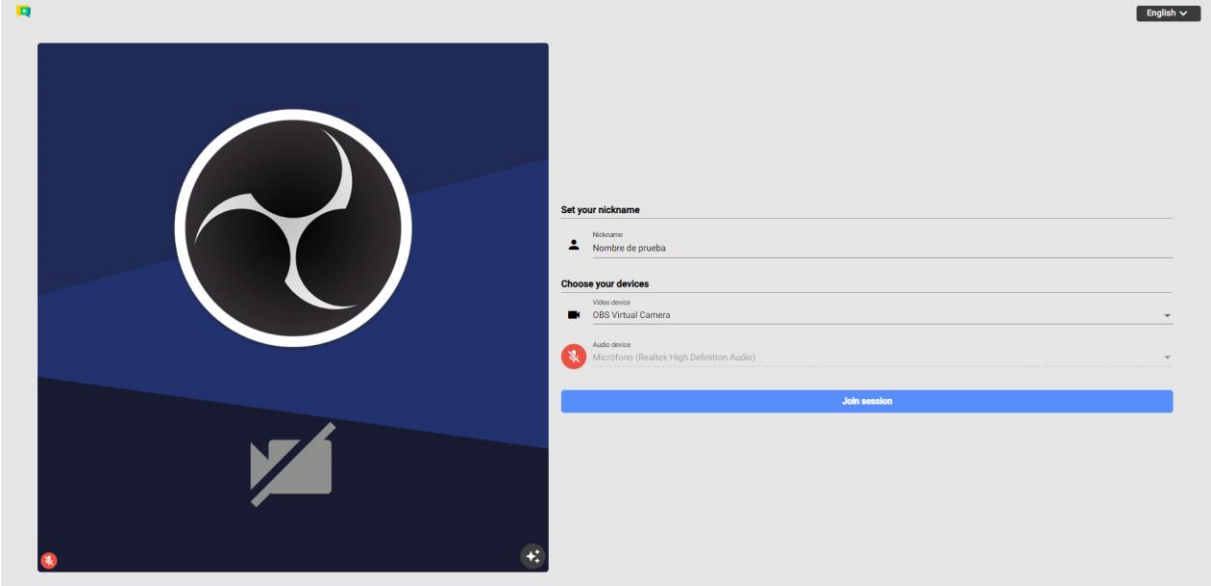

Imagen 3. Página JoinSession OpenVidu Call. OpenVidu Call.

- **Session:** Esta es la página en la que los usuarios podrán comunicarse entre sí. Tal y como se puede ver en la Imagen 4, a simple vista posee un formato parecido a la mayoría de las aplicaciones de videoconferencia, y por tanto posee funcionalidades compartidas con las mismas, entre ellas se encuentra un botón de cerrar la llamada que traslada al usuario a la página de Join, un botón de silenciar el micrófono propio y que ningún usuario escuche entonces la voz, al igual que bloquear la cámara y mostrar una pantalla en negro en su lugar y compartir pantalla. Además de esto OpenVidu cuenta con la posibilidad de silenciar de manera individual a otros usuarios sin que nadie más de la sesión sea perjudicado por ello, es decir, que la persona a la que se silencia de manera individual lleva a que solo el usuario que ha realizado dicha acción sea el que note el cambio y no lo pueda escuchar. Además, independientemente de la cantidad de usuarios, las cámaras se adaptarán para cubrir todo el espacio disponible, pudiendo además seleccionar qué cámaras queremos ver más grandes que el resto.

Por otro lado, también cuenta con ciertas funcionalidades que los separan de otras plataformas de videoconferencia, como puede ser el poder grabar las llamadas, subtítulos que se forman automáticamente a la vez que se habla, efectos de fondo de cámara, lista de usuarios y chat incorporado todo ello en una misma página.

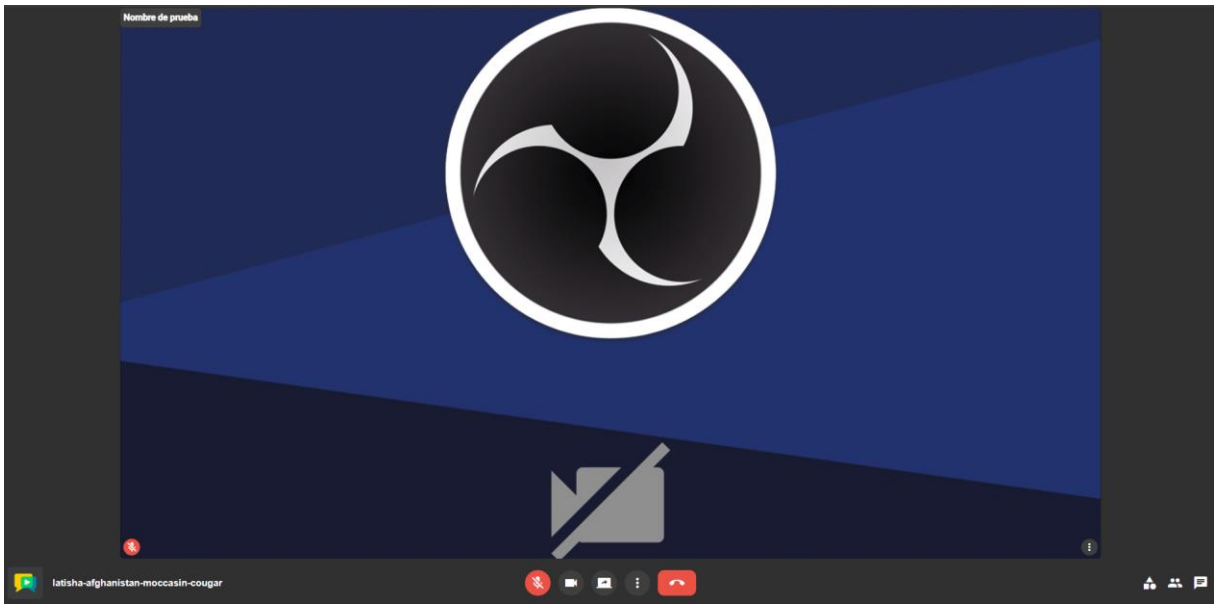

Imagen 4. Página Session OpenVidu Call. OpenVidu Call.

La intención de este proyecto nació de la curiosidad por este nuevo framework que con el tiempo ha conseguido ganarse bastante seguidores gracias a su sencillez. Vue era desconocido hasta el momento de comenzar este trabajo de fin de grado, pero tras investigar un poco se descubrió todas las características que hacen tan llamativo a Vue y por las que se decidió emprender este proyecto. A título personal se podría mencionar que tras haber trabajado con Angular y TypeScript [6] se hacía muy interesante el explorar otros frameworks con sus características singulares. Todo ello impulsado por el gusto por las aplicaciones web y su versatilidad, formato de trabajo, resultados rápidos en cuanto a programación, una retroalimentación inmediata sobre los progresos realizados o los problemas que haya con cualquier componente y sobre todo la utilidad y facilidad de uso para los usuarios comunes habituados a las páginas web más que a aplicaciones nativas.

Vue es conocido por ser un framework que "compite" con Angular. Ambos cuentan con sus ventajas y desventajas, ambos sirven para crear Single Page Applications (SPAs), de ahí que sea una opción comprensible para sustituir a Angular en la aplicación de OpenVidu Call. Sin embargo, la diferencia fundamental viene dada por la manera de organizar los diferentes

apartados de HTML, CSS y JavaScript. Angular suele dividir los apartados de diseño, funcionalidad y estilo en tres archivos con su correspondiente extensión, mientras que Vue está pensado para simplificar toda esa cantidad de archivos y condensar esas tres partes en uno solo, tal y como se explicará más adelante. Esto permitirá una mejor lectura de los diferentes elementos de la aplicación y como interactúan entre sí, ya que no hace falta estar cambiando entre los archivos de TypeScript, HTML y CSS constantemente para verlo.

Este proyecto permitirá que otras personas que se encuentren interesados en OpenVidu Call y no posean conocimientos de Angular, puedan adentrarse más fácilmente en su funcionamiento gracias a que Vue cuenta con una curva de aprendizaje más fácil que Angular, ya que es necesario estudiar TypeScript para manejarlo. Se podría establecer como una versión ampliada del tutorial de OpenVidu implementado con Vue que ya posee la página oficial de la aplicación. Esta versión ampliada permitirá a todos los nuevos usuarios entender el concepto con el tutorial básico y luego poder echar un vistazo a la composición más específica con este trabajo de fin de grado, ya que el tutorial y la aplicación de OpenVidu Call poseen backends diferentes.

CAPÍTULO 2 Objetivos

El siguiente trabajo de fin de grado pretende crear una versión de la aplicación conocida como OpenVidu Call implementada en el framework de Vue aprovechando las diferentes herramientas que proporciona este último y reutilizando el backend en Java de OpenVidu Call y sus servidores. Debido a la gran cantidad de funcionalidades que posee la aplicación original en Angular se pretende realizar una versión básica del mismo de modo que en el futuro se pueda ampliar de manera sencilla y añadir estas funcionalidades avanzadas como chat escrito, grabación de llamadas o filtros. De tal modo, las funcionalidades con las que contará este proyecto son las siguientes:

- **Formulario de inicio de sesión:** El sistema de autentificación está basado en una simple comprobación de nombre de usuario y contraseña, por lo que el objetivo es recoger los datos introducidos por el usuario y mandar la comprobación al backend.
- **Inicio de sesión automáticamente mediante cookies:** Para no tener que estar iniciando constantemente la sesión se guardará en las cookies las credenciales encriptadas y la aplicación se encargará de usarlas automáticamente para ahorrarle pasos al usuario.
- **Formulario de selección de nombre de la sala:** Para que los usuarios puedan personalizar o seleccionar un nombre aleatorio de la sala de videoconferencia.
- **Formulario de selección de nombre, cámara y micrófono:** Previamente a la conexión de la llamada el usuario podrá seleccionar mediante qué dispositivos accederá, su nombre y si la cámara o el micrófono se encuentran disponibles en primer lugar.
- **Funcionalidades de los botones básicos dentro de la llamada:**
  - **Mutar audio:** El usuario bloquea su sonido o deja de escuchar a otro concreto.
  - **Mutar cámara:** El usuario bloquea su cámara para todos los demás.
  - **Compartir pantalla:** El usuario podrá compartir pantalla como si fuera otra cámara cualquiera.
  - **Agrandar las cámaras que se deseen**: El usuario agrandará para ver mejor.
  - **Finalizar llamada:** El usuario se desconecta de la llamada y vuelve al menú.
- **Vista de las participantes basadas en una librería dinámica:** La librería se denomina opentok-layout-js y permitirá que las cámaras se ajusten automáticamente independientemente del número de usuarios conectados.

CAPÍTULO 3 Tecnologías, herramientas y Metodologías

En el siguiente capítulo se explicarán las diferentes tecnologías, librerías y herramientas empleadas para la elaboración de este trabajo de fin de grado, así como la metodología seguida durante todo el desarrollo.

3.1 Tecnologías y herramientas

A continuación, se explicarán las diferentes tecnologías utilizadas para la implementación del proyecto.

3.1.1 Java y Maven

Java es una tecnología de desarrollo de aplicaciones multiplataforma. A pesar de que no se ha programado en este lenguaje es necesario mencionar que se ha utilizado para el uso del backend. OpenVidu Call cuenta con dos backends disponibles completamente funcionales que son Node o Java, dada la familiaridad del uso de Java y de Spring para inicializarlo se optó por usar este último en el proyecto.

Por otro lado, Maven [7] es un gestor de proyectos que es usado para inicializar el backend de Java de OpenVidu Call de manera local y posteriormente se usará en las instrucciones referentes al backend para realizar el despliegue de la aplicación tal y como se explicará más adelante.

3.1.2 Vue

Vue es un framework basado en JavaScript para la creación de Single Page Applications de manera sencilla y compacta. A nivel de estructura, cada componente .vue se divide en tres partes que tienen su homólogo en HTML:

1. **Diseño:** Comenzando con un apartado “template” que es donde irá el lenguaje de marcado similar al HTML, pero sin necesidad de división de encabezado y cuerpo,

directamente este template se cuenta como el cuerpo de la página web. Angular consigue marcarse como una página web dinámica gracias al uso de condicionales y bucles dentro del apartado de diseño, Vue es igual en este sentido. Cuenta con v-if y v-for para manejar la página y hacerla cambiante en función de lo que desee el desarrollador.

Un apartado interesante en este ámbito es la capacidad de incrustar otros componentes Vue dentro de otros o la de pasar valores a estos componentes para personalizarlos. Un ejemplo sencillo sería: La página principal de Vue posee un listado de otros elementos, dichos elementos son otras páginas de Vue que contienen un apartado de texto propio. Cada uno de esos componentes de la lista no es más que el mismo componente .vue repetido y al que se le añade el texto que se desee, de esta manera no hace falta crear un archivo por cada componente y se hace uso de la reutilización, para ello se usan los props, que no son más que variables que se sabe que son pasadas desde el principio por el componente padre. Este ejemplo puede ser visualizado en la Imagen 5 y 6.

App.vue BlogPost.vue × +

```
1  <script>
2  export default {
3    props: ['title']
4  }
5  </script>
6
7  <template>
8    <h4>{{ title }}</h4>
9  </template>
```

Imagen 5. Ejemplo de uso de props BlogPost. Vue.

App.vue BlogPost.vue × + Import Map PREVIEW JS CSS SSR

```vue
<script>
import BlogPost from './BlogPost.vue'

export default {
  components: {
    BlogPost
  },
  data() {
    return {
      posts: [
        { id: 1, title: 'My journey with Vue' },
        { id: 2, title: 'Blogging with Vue' },
        { id: 3, title: 'Why Vue is so fun' }
      ]
    }
  }
}
</script>

<template>
  <BlogPost
    v-for="post in posts"
    :key="post.id"
    :title="post.title"
  ></BlogPost>
</template>
```

My journey with Vue

Blogging with Vue

Why Vue is so fun

Imagen 6. Ejemplo de uso de props App. Vue.

Esta propiedad ha sido usada para insertar los componentes de vídeo en la llamada, pasándoles a los mismos los streamManager necesarios para ejecutar el vídeo de las cámaras de los usuarios. Por otro lado, se ha aprovechado el uso de props de manera anónima, es decir, se han insertado etiquetas template dentro del template principal para añadir ciertos componentes, en este caso para las pestañas de ayuda que aparecen cuando se mueve el cursor sobre algunos botones.

Además, es posible mostrar, igual que en Angular, variables presentes en el apartado de programación, para ello se usará el símbolo de doble llave, tal y como se

muestra en el ejemplo anterior con {{ title }}, esto indica a la web que debe leer el valor de dicha variable.

2. **Programación:** Se encuentra escrito en JavaScript y se divide en diferentes apartados para mayor claridad. En primer lugar, se le puede añadir nombre al componente y se indica el listado de otros componentes Vue que serán usados, a continuación, se entra en el apartado de variables, que puede ser el de props y/o data. En este último podremos nombrar todas las variables globales que deseemos como si fuera un objeto Java. Para acceder a los mismos a lo largo del programa se deberán llamar con el prefijo "this.".

   Por supuesto la página web no se podría comportar de manera dinámica sin la propia programación, y para ello se encuentra el apartado methods en el que se haya el listado de todas las funciones. Para llamarlas dentro del programa será necesario usar el prefijo ".this". Estos métodos pueden ser llamados entre sí como se ha mencionado o desde el apartado de diseño mediante algún evento como un click en un botón.

   El apartado de mounted sirve de equivalente directo a la espera de carga de página en JavaScript, mientras que el apartado de created sirve para lo mismo, pero sin esperar a que se carguen los componentes de la página, por lo que todavía no se pueden acceder, pero se pueden inicializar las variables de la página, por ejemplo. Finalmente, el apartado composed sirve para crear funciones que serán ejecutadas constantemente cuando se cambien los valores de las variables internas [8]. Para que se entienda mejor existe el siguiente escenario: Se desea que un botón cambie de color de fondo en función de una variable, entonces esa variable de background obtendrá su valor (rojo o azul) a partir de la función presente en composed, que devolverá uno de esos dos valores en función de un condicional, cuando la condición cambie también cambia el valor devuelto y la página se cambia de manera dinámica.

3. **Estilo:** El apartado de estilo es exactamente igual que un archivo CSS, con la diferencia de que se encuentra incrustado en el propio archivo vue mediante las etiquetas style. A nivel de lenguaje o composición no se diferencia en nada de CSS y el uso para convocar esos estilos en el apartado de diseño es exactamente igual que en un archivo HTML por lo que no se puede añadir mucho más a este apartado.

3.1.3 Vuetify

OpenVidu Call está implementada con Material Design [9], que es una librería de estilo, por ello para conseguir uno parecido se decidió utilizar Vuetify [10]. Una librería de estilo creada especialmente para Vue, cómo su propio nombre puede hacer intuir. Concretamente se usó Vuetify 3 y se creó un proyecto de Vuetify desde cero en vez de uno de Vue al que después se introdujera la librería. El proyecto de Vue no difiere mucho del de Vuetify más que en organización y la propia inclusión de la librería.

Los componentes vienen dados por etiquetas con propiedades diferentes que sirven para multitud de usos y capacidad de customización sin CSS. Por ejemplo, v-btn es un botón que al añadirle la propiedad icon se volverá redondeado automáticamente ya que entiende que cuando se introduce esta propiedad es porque se desea un botón con solo un icono.

La documentación cabe mencionar que se encuentra muy completa, que es fácil de seguir y que ofrece ejemplos constantes con todo el código disponible, tanto apartado de script como de diseño.

3.1.4 OpenVidu

OpenVidu es una plataforma creada para facilitar la adicción de videollamadas a otras páginas web. Cuenta con una aplicación completamente funcional llamada OpenVidu Call y con un componente web basado en esa misma aplicación preparado para ser incrustado en cualquier proyecto. Sin embargo, también permite integrar OpenVidu en una aplicación desde cero, este trabajo de fin de grado se encuentra en un punto medio en esta última opción. La idea de implementar OpenVidu desde el principio es la que se ha seguido, aunque, se han aprovechado dos partes fundamentales de OpenVidu Call. Siendo la primera el backend de OpenVidu Call de Java y la segunda la imagen Docker [11] del servidor de OpenVidu. Este último puede ser sustituido por la versión del servidor de demos para aquellos ordenadores que no tengan instalado Docker o simplemente para probar. A nivel personal se fue probando la aplicación con ambas versiones para comprobar su funcionamiento y debido a las diferencias de velocidad de uno u otro. En el ordenador usado para trabajar la versión de demos era más rápida que la de Docker.

La aplicación entonces se divide en tres partes: el servidor de OpenVidu, el backend en Java de OpenVidu Call y el frontend que en este caso será de Vue. Resulta importante recalcar

el cómo se comunica cada parte para el funcionamiento de la aplicación. Como se puede ver en la Diagrama 1, todos los componentes se comunican de manera direccional salvo el backend que solo manda información al servidor de OpenVidu. El servidor se encargará de todo lo que tenga que ver con el propio servicio que ofrece OpenVidu, como puede ser el conectarse una llamada, el backend se encargará de realizar las peticiones web al servidor y finalmente la aplicación del cliente será el que organice la información y permita el manejo de la aplicación por parte del usuario.

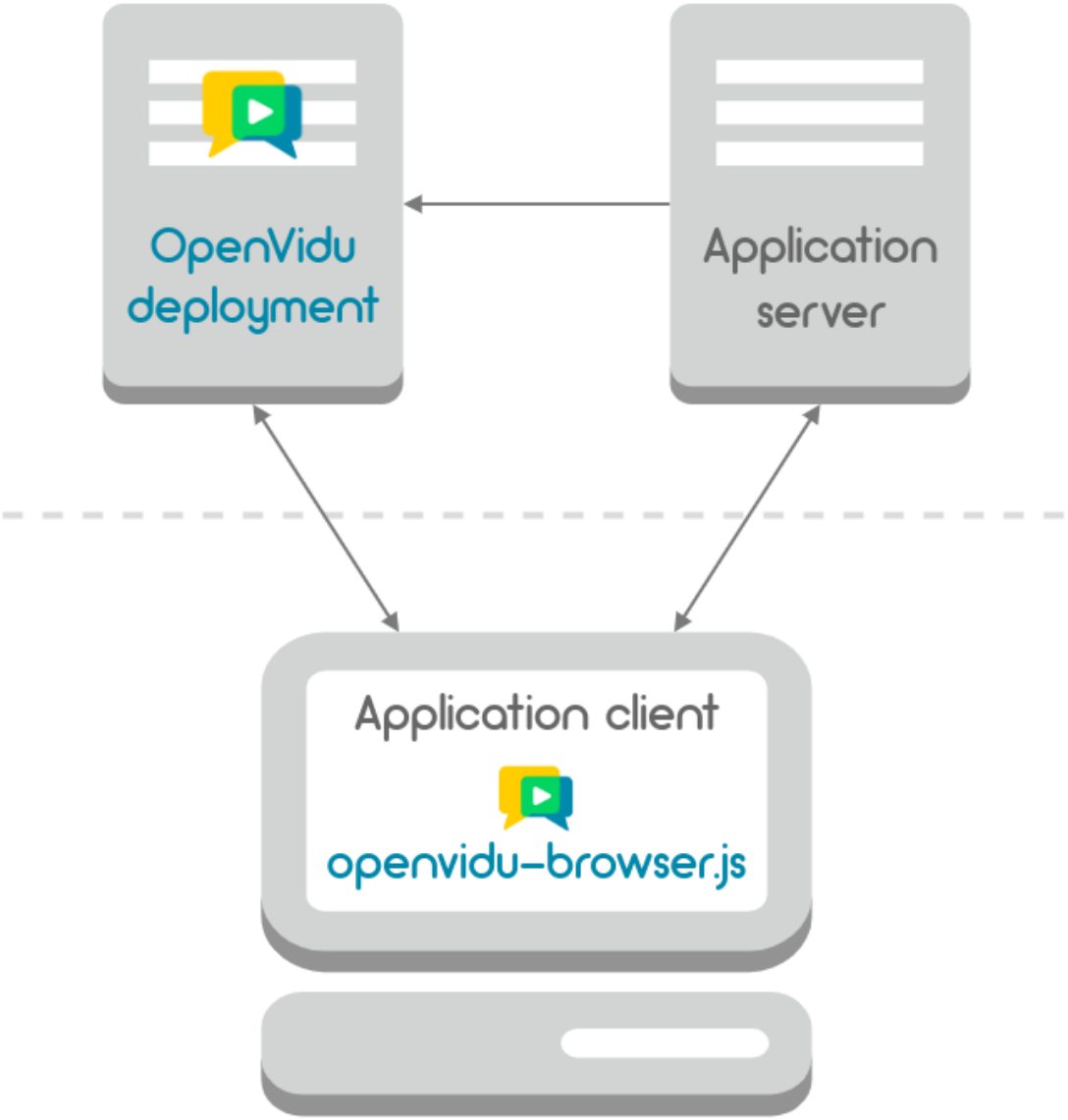

Diagrama 1. Comunicación de partes de una aplicación OpenVidu. OpenVidu.

3.1.5 Visual Studio Code

A nivel de editor de desarrollo se decidió usar Visual Studio Code [12] por distintas razones. La primera siendo que ya es un editor con el que se ha trabajado anteriormente y que a nivel personal es agradable trabajar con él. Su interfaz es sencilla pero completa y bien organizada. Sin embargo, una de las razones por las que se decidió usar este editor es debido al terminal integrado que posee. Esto hace que sea más cómodo trabajar en web desde este editor ya que se puede levantar Docker, backend y frontend todo en un mismo editor y acceder más fácilmente a la ruta de los archivos cuando hay que instalar nuevas dependencias a lo largo del desarrollo.

3.1.6 GitHub

GitHub [13] es la plataforma que se ha usado para compartir el código del proyecto con el tutor y lanzar los releases. GitHub es una plataforma en el que todos los usuarios pueden publicar sus repositorios y gestionarlos ya sea en grupo o en solitario. De esta manera se puede ver cómo se ha ido desarrollando el proyecto paso a paso a lo largo del tiempo y además comprobar qué se subió a cada commit.

Por otro lado, es destacable otra parte de GitHub que ha sido usado, y en este caso es GitHub actions para ejecutar automáticamente todos los test siempre que se realicé una subida de código (push) o se realice una petición de descarga de código (pull_request). El archivo .yml que controla GitHub actions se encargará de levantar el servidor Docker, el backend y el propio OpenVidu Call de Vue para después llamar a ejecución de todas las pruebas que se explicarán más adelante.

3.1.7 Trello

Trello [14] es una plataforma cuya funcionalidad es la administración de proyectos. Siguiendo el sistema Kanban [15] permite la creación de listas, tareas, asignación de tareas, subida de imágenes, comentarios, enlaces además de temporizadores y otras herramientas que sirven como plugins o, tal y como los llama la aplicación, “power-ups”. Al ser un proyecto realizado por una sola persona obviamente no se han usado todas las herramientas que pone a disposición la página web, sobre todo se ha aprovechado la creación de tarjetas y la creación de diferentes listas y división por etiquetas. De hecho, a pesar de que existe total libertad para

el uso de la aplicación se decidió seguir la misma metodología Kanban anteriormente mencionada. Esta metodología se caracteriza por agrupar las tareas en tres listas e ir moviendo las tareas de una a otra en función de su estado actual, siendo estas: Pendiente de realizar (To do), En proceso (Doing) y Completado (Done) tal y como se puede ver en la Imagen 8. Además, de manera personal se incluyó algunas otras listas como problemas encontrados y bibliografía entre algunos otros, para poder ir apuntando información más concreta a lo largo del desarrollo y tener muy en cuenta algunas tarjetas, como las de problemas hasta conseguir solucionar la cuestión que estaba ocasionando percances.

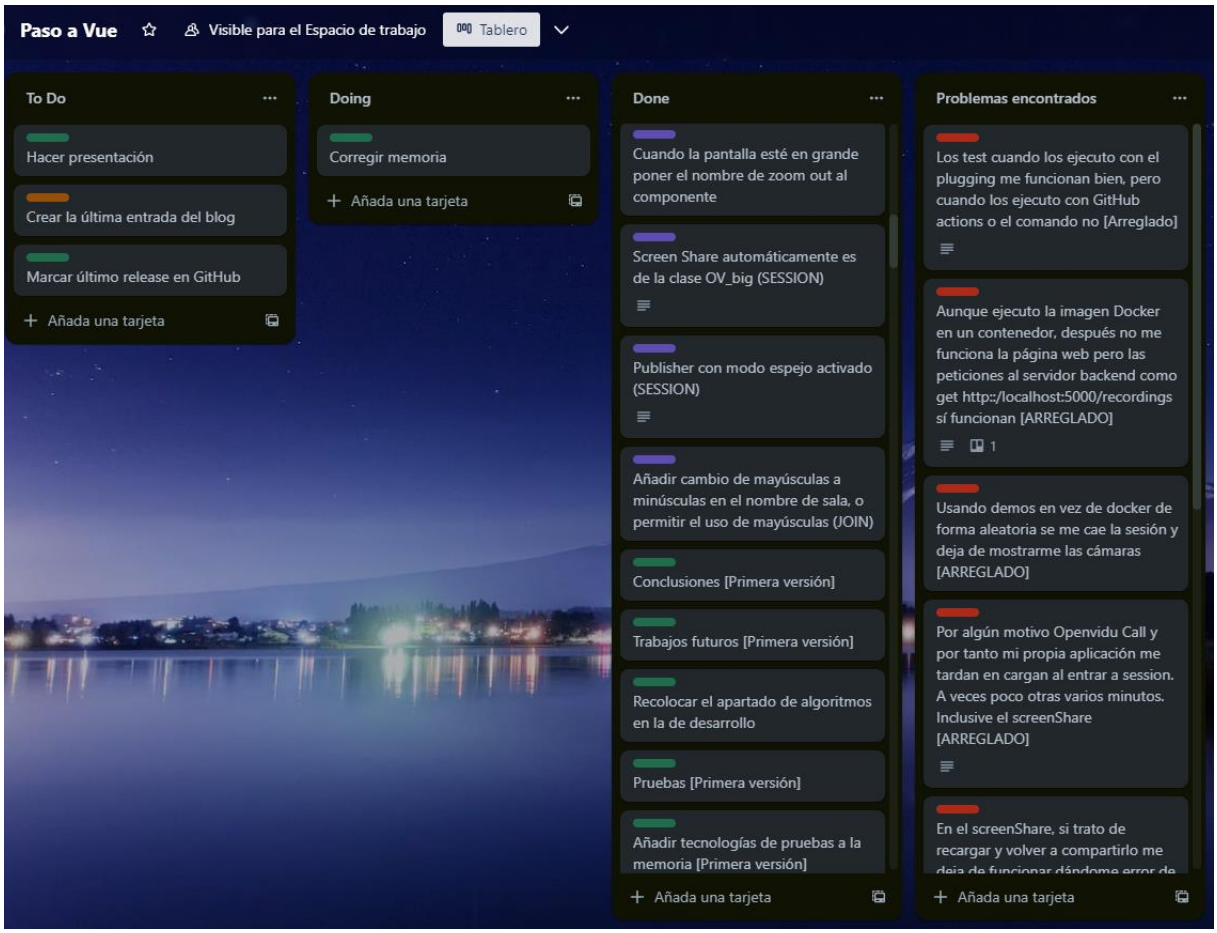

Imagen 7. Tablero de Trello del proyecto. Elaboración propia.

3.1.8 Docker

Docker es un sistema que permite el despliegue de aplicaciones dentro de contenedores software. Dicha tecnología ha sido usada en dos partes del proyecto, la primera siendo para ejecutar la imagen Docker de OpenVidu para el correcto funcionamiento de la aplicación, ya que hacía la función de servidor de OpenVidu con el que se comunicaría el backend.

Por otro lado, el propio trabajo de fin de grado en la fase final del proyecto se ha dockerizado para que todos los usuarios puedan usarlo libremente sin necesidad de tener que instalar los componentes necesarios para compatibilidad, tan solo descargar el código y crear la propia imagen Docker junto con la tecnología necesaria para ejecutarla en función del sistema operativo en el que se trabaje, siendo una opción sencilla la aplicación de Docker Desktop.

3.1.9 Medium

Medium [16] es una página web gratuita destinada a la creación de blogs. Durante el desarrollo del proyecto se hizo uso de esta herramienta para documentar los avances importantes del trabajo de fin de grado a lo largo del tiempo. De modo que cuando la aplicación contaba con una nueva funcionalidad destacable se creaba una entrada en el blog explicando la misma, el cómo se realizó y aportando enlaces e imágenes interesantes para su comprensión. Además, las entradas a dicho blog se encuentran en formato público y en inglés para que todo el mundo pueda ver el proceso y entenderlo.

3.1.10 Windows 10

Windows 10 [17] es conocido como uno de los sistemas operativos más utilizados a nivel internacional, por ello el ordenador usado para trabajar contaba con él. Como formatear para trabajar en Linux [28] tal y como se desarrolló OpenVidu no era viable y el montar una máquina virtual habría acabado ocasionando bajadas de rendimiento o problemas con los periféricos se decidió trabajar directamente en Windows 10. Esta decisión supondría la necesidad de solventar algunos problemas de configuración con Docker y Hyper V [19] para su correcto funcionamiento. En general Windows 10 tiene dificultades para comenzar a usar Docker, si se hubiera realizado en Linux este proceso habría sido automático ya que está mejor preparado para esa tecnología.

3.1.11 Google Chrome

Como navegador web para realizar las pruebas manuales se podría haber usado cualquiera, como por ejemplo Firefox [20], pero se escogió Google Chrome [21] debido a su rendimiento aceptable y costumbre de uso.

3.1.12 Playwright

Playwright [22] es un servicio de automatización de pruebas compatible con GitHub actions. Como su propia definición indica ha sido la herramienta seleccionada para la creación de pruebas de la aplicación web. En su correspondiente sección se explicará más en profundidad el cómo se usó esta tecnología, pero cabe destacar dos cosas. Siendo la primera su sintaxis sencilla y de fácil acceso que ha permitido la creación de pruebas de manera rápida y, en segundo lugar: la útil función de grabar las acciones que va realizando el usuario para crear test. Esta función solo se ha usado para encontrar los comandos concretos para las acciones referentes a seleccionar qué cámara o micrófono se desea usar.

El poco uso que se ha dado a esta potente función como es crear pruebas mediante las interacciones del usuario ha sido debido a que en la mayoría de los casos complicaba bastante la sintaxis de algo sencillo como encontrar un botón mediante una etiqueta y la grabación de pruebas lo encontraba mediante el tipo de componente del botón, por ejemplo. Además de que cuando se grababan pruebas llegaba a entrar en problemas cuando se quería dotar de los permisos de cámara y micrófono a la página web, es decir, que dejaba de funcionar correctamente la grabación cuando no se trataba directamente con la propia página web y se juagaba con la configuración de la misma.

3.1.13 StarUML

Para la realización de los diagramas que se especificarán más adelante en la sección 4.1 se ha utilizado la aplicación gratuita de StarUML [23]. A pesar de existir otras como MagicDraw [24] se ha optado por utilizar esta debido a la familiarización previa con este programa. Además, debido a que los diagramas no iban a ser excesivamente grandes no era necesario que las imágenes fueran vectoriales para poder ampliarse y verse correctamente, por lo que las imágenes basadas en píxeles de StarUML no eran un impedimento para la realización de esta memoria.

3.1.14 Opentok-layout-js:

Opentok-layout-js [25] es una librería de terceros que tiene la funcionalidad de ajustar el tamaño de las cámaras para que ocupen todo el espacio disponible. De esta manera cada vez

que se añade o elimina alguna cámara la aplicación deberá actualizarse para acoplar a todos en el espacio disponible.

Todas las propiedades para manejar la distribución, el tamaño y la forma se encuentran descritas en el GitHub de la librería además de poseer especificaciones claras de lo que hace cada una de las propiedades con las que cuenta.

3.1.15 MergeProps:

En el apartado 3.2.2 Vue se ha explicado el cómo funcionan los props, o variables que se le pasan a apartados de diseño, esta librería sirve para que un componente pueda poseer dos de esos props a la vez. En el caso del proyecto ha sido usado en los menús de opciones disponibles para cada usuario conectado en la sesión. Concretamente para que muestren el tooltip de “Settings” cuando se coloca el cursor encima y activar la lista de opciones del menú desplegable al pulsarlo. Como cada uno se enlaza al botón mediante los props era necesario el combinar los dos en uno solo gracias a esta librería.

3.1.16 Unique-names-generator

Como su propio nombre indica es un generador de nombres aleatorios [26] que cuenta con una gran cantidad de grupos de nombres diferentes, desde animales hasta ciudades y colores. Todos ellos serán usados para la creación de nombres de sala. El propio OpenVidu Call hace uso de esta misma librería para ese mismo propósito. Así que es una librería comprensible de aprovechar al no depender del framework utilizado para ahorrar tiempo.

3.1.17 RxJS

La aplicación de OpenVidu Call cuenta con un servicio para el manejo de las peticiones web, este servicio se encuentra separado y además crea un objeto observable, siendo la variable “logged”. Esta variable será consultada a la hora de cambiar de rutas y por lo tanto será necesario tratarlo como un BehaviorSubject conseguido gracias a la librería externa de RxJS [27]. Se podría haber buscado quizás algún otro método para no tener que hacer uso de esta librería, pero se quiso priorizar el intentar hacer el proyecto lo más similar a la aplicación original en la medida de lo posible.

3.1.18 Axios

Ni Vue, ni comprensiblemente Vuetify, cuentan con un sistema para peticiones web, una funcionalidad necesaria para la correcta comunicación cliente-servidor. Para solucionarlo se acudió a la librería de axios [28], cuya funcionalidad es precisamente el emitir peticiones y recuperar las respuestas. Concretamente en el proyecto se ha usado en los componentes JavaScript de la carpeta api. Es perfectamente compatible con los archivos vue, pero se decidió separarlo en un archivo a parte para mejorar la legibilidad y la modularización del proyecto.

3.1.19 Material Design Icons

Los distintos iconos proceden de la librería de Material Design Icons [29]. Se utilizó concretamente una página web que permite buscar a base de categorías o nombres los distintos iconos que dispone y para implementarlos simplemente hacía falta añadir el sufijo "mdi" al nombre encontrado en esa página web en el componente de diseño deseado.

3.2 Metodología

Para la realización de este trabajo de fin de grado se ha seguido una metodología iterativa basada en incrementos del proyecto. En primera instancia se realizó una reunión con el tutor para aclarar el cómo se realizaría el proyecto y los objetivos que se debían de cumplir además de una breve introducción sobre la propia tecnología de Vue y OpenVidu Call. Una vez realizado esto se organizó el proyecto siguiendo los siguientes pasos:

Fase 1. Se estudió a programar con Vue sirviéndose de la página principal de Vue con su extensa documentación. Dichas páginas serían usadas de referencia en otras ocasiones. Por supuesto, para poder trabajar con Vue y probar sus capacidades se instalaron las herramientas necesarias en Visual Studio Code y probar un poco su funcionamiento en el proyecto de prueba automático que se inicia al crear un nuevo proyecto en Vue.

Fase 2. Una vez hecho el paso anterior se procedió a decidir qué clase componentes de estilo se utilizarían. Las opciones propuestas por el tutor fueron Vuetify o Vue Material [30]. Se compararon ambas librerías y se descubrió que en formato, componentes disponibles y uso eran muy parecidas así que el resultado acabaría siendo muy similar. Al final se decidió por Vuetify por el cómo estaba distribuida la información en la

documentación oficial, ya que se veía más completo, con muchos ejemplos y organizado de una manera más clara. Por supuesto la documentación de Vuetify será consultada constantemente durante el tiempo que dure el proyecto para ir creando y manejando los diferentes componentes de los que dispone.

Fase 3. Es imposible realizar una implementación en otro frontend de OpenVidu sin conocer la propia aplicación. Así que se leyó gran parte de la documentación disponible y que por supuesto fuera relevante para el proyecto en cuestión. La documentación oficial dispone de un tutorial de la plataforma de OpenVidu en la que se implementa un frontend en Vue así que se estudió el funcionamiento del mismo para entenderlo a rasgos generales y trabajar a partir de este.

Fase 4. Una vez entendido el cómo funciona OpenVidu Call se procedió a organizarse el trabajo gracias a un tablero de Trello [31] que seguiría la metodología Kanban. Se fue observando las diferentes funcionalidades y se fueron apuntando de manera corta en una lista de tarjetas que se organizarían en la lista de "To do", además de dividirlas por etiquetas que ayuden en el futuro a identificarlas adecuadamente como por ejemplo "Frontend" o "Problemas".

Fase 5. A partir de este paso es cuando se comienza el proceso iterativo. Cada día de trabajo se seleccionaba las tareas que había que realizar de la lista previamente creada o añadiendo alguna tarea que fuera necesaria y que no se hubiera contemplado con anterioridad a la columna de "To do", las tareas seleccionadas se movían a la columna de "Doing".

Fase 6. Se trabaja en las tareas asignadas y durante el proceso se van apuntando también las páginas consultadas para las dudas, los problemas que surgieron y si se lograron solucionar el cómo y crear subdivisiones de las tareas en listas de ser necesario. Al terminar alguna funcionalidad se sube la misma al repositorio GitHub [32]. Cuando la tarea era completada se trasladaba la tarjeta correspondiente a la columna de "Done".

Fase 7. Si surge algún problema que no se solucionó y ha perdurado durante algunos días se procede a escribir al tutor por correo para intentar trabajar en una solución. En caso de no llegarse a una solución mediante correo se establece una reunión mediante Teams para tratar el tema o algún otro aspecto del trabajo de fin de grado.

Fase 8. Cuando se ha alcanzado un tamaño considerable de trabajo realizado o se ha terminado de implementar alguna funcionalidad importante entonces se procede a escribir una entrada al blog de Medium [33] informando de este y sobre su implementación y detalles.

Por otro lado, es notable destacar el orden en el que se afrontaron esas tareas que acabarían conformando la aplicación. En primer lugar, se estableció los diferentes componentes que conforman la aplicación de OpenVidu Call, es decir, los cuadros de texto, botones y navegación entre páginas. En este punto la aplicación se encuentra bastante "vacía" puesto que solo funciona la navegación. En total se podría determinar que hay cuatro páginas que el usuario puede visualizar y moverse entre ellas, sin embargo, estas están divididas en dos componentes, cada una: Home (con los apartados de Login y Join cuyo diseño final se puede ver en la Imagen 8 y9) y Call (con los apartados de JoinSession y Session cuyo diseño final se puede ver en la Imagen 10 y 11). Luego de este paso se comenzó a trabajar comenzando por el Login hasta llegar a Session con todas sus funcionalidades. A medida que se hacían tareas se pulía el diseño de la aplicación. Finalmente se trató con las pruebas y por último dockerizar la aplicación.

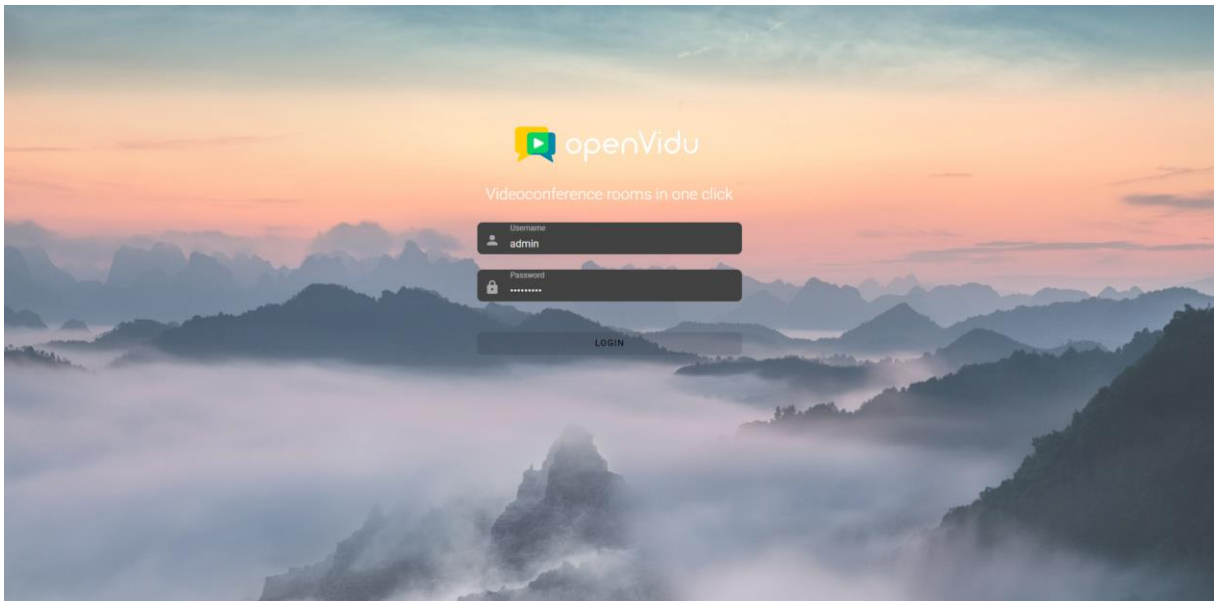

Imagen 8. Página Login. Elaboración propia.

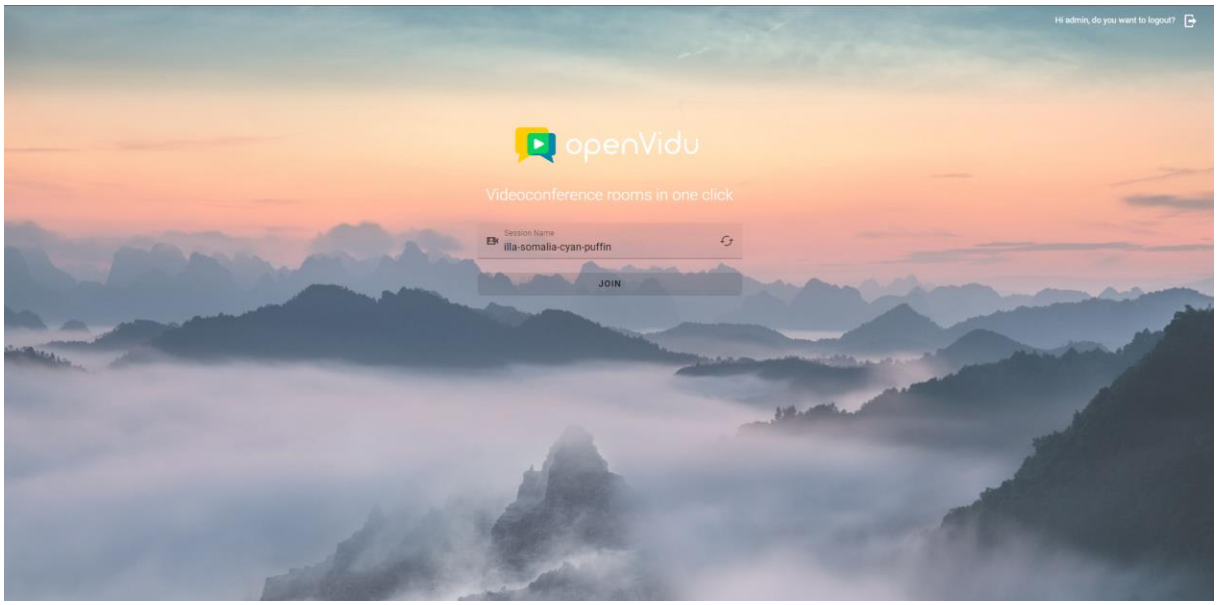

Imagen 9. Página Join. Elaboración propia.

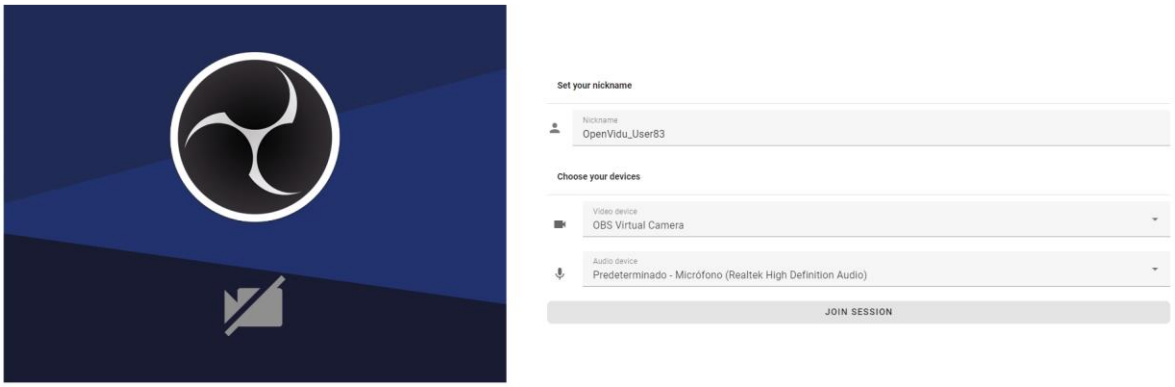

Imagen 10. Página JoinSession. Elaboración propia.

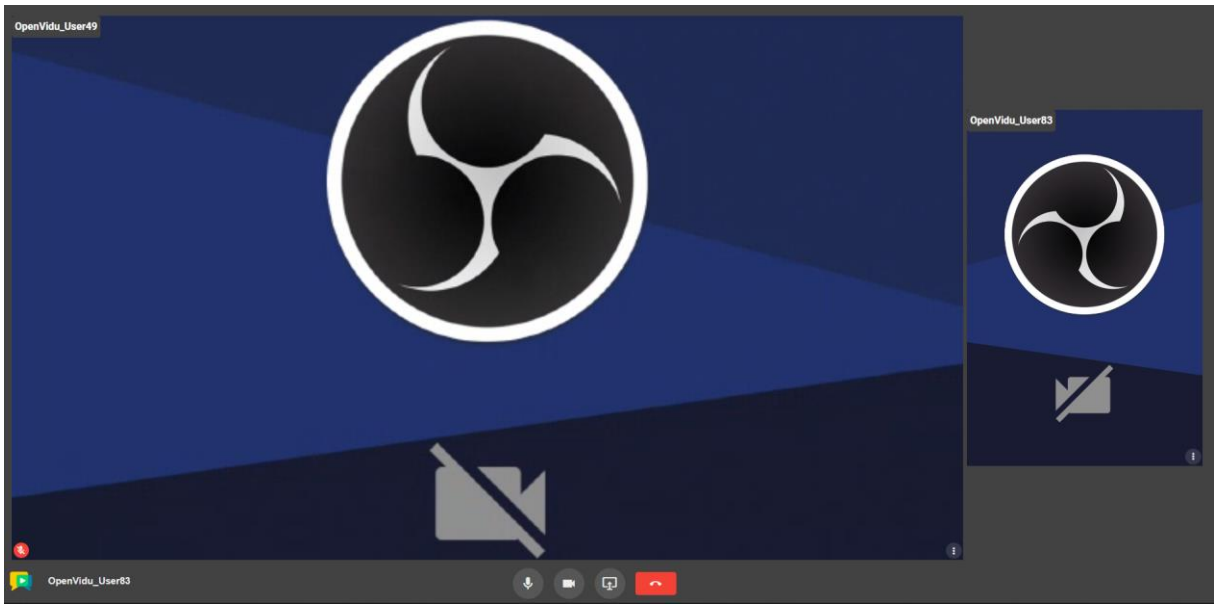

Imagen 11. Página Session. Elaboración propia.

CAPÍTULO 4: Descripción informática

En este capítulo se tratarán en profundidad distintos aspectos del desarrollo de OpenVidu Call en Vue.

4.1 Requisitos

A continuación, se describirán las funcionalidades principales predispuestas al inicio del proyecto en formato de historias de usuario. Prácticamente engloban todos los objetivos descritos anteriormente. Por supuesto, a lo largo del desarrollo se encontraron pequeños requisitos no funcionales en su mayoría que serán descritos en la sección 4.1.2 además de realizar un par de cambios con respecto a la aplicación de OpenVidu Call que serán descritos en esa misma sección.

4.1.1 Historias de usuario

A continuación se mostrará una lista de las diferentes historias de usuario contempladas antes de comenzar el desarrollo:

- Historia de usuario 1. Iniciar sesión de usuario Tabla 1.
- Historia de usuario 2. Salir de la sesión de usuario Tabla 2.
- Historia de usuario 3. Iniciar sesión automáticamente Tabla 3.
- Historia de usuario 4. Seleccionar nombre de sala Tabla 4.
- Historia de usuario 5. Seleccionar las opciones de sesión Tabla 5.
- Historia de usuario 6. Mutar audio Tabla 6.
- Historia de usuario 7. Mutar video Tabla 7.
- Historia de usuario 8. Compartir pantalla Tabla 8.
- Historia de usuario 9. Ampliar cámaras Tabla 9.
- Historia de usuario 10. Ver cámaras dinámicamente Tabla 10.
- Historia de usuario 11. Conectar a videoconferencia Tabla 11.
- Historia de usuario 12. Finalizar llamada Tabla 12.
- Historia de usuario 13. Mutar audio de otro usuario Tabla 13.

HU-1	Iniciar sesión de usuario
Descripción	Como usuario de la aplicación quiero poder iniciar sesión nada más acceder a la página para poder acceder al resto de funcionalidades.
Prioridad	Alta.
Estimación	Media.
Conversación	❖ Seleccionar casilla a rellenar → Permite escribir en dicha casilla. ❖ Seleccionar el botón de "LOGIN" con los campos rellenos → Comprueba que los datos introducidos son correctos (usuario y contraseña). ❖ Seleccionar el botón de "LOGIN" sin los campos rellenos → No realiza la acción de inicio de sesión. ❖ Si la acción 'Seleccionar el botón de "LOGIN"' produce un resultado positivo → Se permite el acceso a la aplicación. ❖ Si la acción 'Seleccionar el botón de "LOGIN"' produce un resultado negativo → Se avisa de que son incorrectos y que se intente de nuevo.
Criterios de aceptación	**Dada** la página inicial de Home con los campos de texto completados. **Cuando** se pulsa el botón de "LOGIN". **Entonces** te permite comprobar si los datos introducidos si son correctos.

Tabla 1. Historia de usuario 1. Elaboración propia.

HU-2	Salir de la sesión de usuario
Descripción	Como usuario de la aplicación quiero poder salir de la sesión de mi usuario para que no se quede abierta en el navegador.
Prioridad	Alta.
Estimación	Muy corta.
Conversación	❖ Seleccionar el botón de "Hi X, do you want to logout?" → Elimina los datos guardados de sesión del usuario y devuelve al mismo a la sección de Login.
Criterios de aceptación	**Dada** la sección de seleccionar nombre de sala tras iniciar sesión. **Cuando** se pulsa el botón de "Hi X, do you want to logout?". **Entonces** se sale de la sesión de usuario.

Tabla 2. Historia de usuario 2. Elaboración propia.

HU-3	Iniciar sesión automáticamente
Descripción	Como usuario de la aplicación quiero que cuando ya he iniciado sesión anteriormente en mi navegador se guarden mis datos para que se inicie la sesión de nuevo automáticamente y pueda ahorrar pasos.
Prioridad	Media.
Estimación	Corta.
Conversación	❖ Navegar a cualquier ruta → Convoca el inicio de sesión automático si los datos están guardados en una cookie del navegador.
Criterios de aceptación	**Dada** la aplicación iniciada. **Cuando** se navega a cualquiera de las rutas. **Entonces** te permite comprobar si los datos introducidos si son correctos.

Tabla 3. Historia de usuario 3. Elaboración propia.

HU-4	Seleccionar nombre de sala
Descripción	Como usuario de la aplicación quiero poder escoger un nombre de sala propio o aleatorio para poder personalizar la sala de videoconferencia.
Prioridad	Baja.
Estimación	Muy corta.
Conversación	❖ Seleccionar casilla a rellenar → Permite escribir en dicha casilla. ❖ Seleccionar el botón de "Generate new session name" → Se genera automáticamente un nombre de sesión aleatorio. ❖ Seleccionar el botón de "JOIN" → Modifica el nombre introducido para que concuerde con los estándares apropiados (transformar los espacios en "-" y restringir a solo letras en minúscula).
Criterios de aceptación	**Dada** la sección de seleccionar nombre de sala tras iniciar sesión. **Cuando** se rellena manualmente o con el botón de aleatoriedad el campo de texto. **Entonces** se utilizará este dato para la creación de la sala de videoconferencia.

Tabla 4. Historia de usuario 4. Elaboración propia.

HU-5	Seleccionar las opciones de sesión
Descripción	Como usuario de la aplicación quiero poder escoger el nombre con el que apareceré, así como periféricos
Prioridad	Baja.
Estimación	Corta.
Conversación	❖ Seleccionar casilla a rellenar de "Nickname" → Permite escribir en dicha casilla. ❖ Seleccionar en el desplegable de "Video device" → Despliega todos los periféricos de vídeo conectados en ese momento. ❖ Seleccionar en una de las opciones del desplegable de "Video device" → Establece esa opción como la actualmente seleccionada en su correspondiente sección. ❖ Seleccionar en el desplegable de "Audio device" → Despliega todos los periféricos de audio conectados en ese momento. ❖ Seleccionar en una de las opciones del desplegable de "Audio device" → Establece esa opción como la actualmente seleccionada en su correspondiente sección. ❖ Seleccionar en el botón de "JOIN SESSION" → Guarda esas opciones y las usará como preferencias para la conexión a la sala.
Criterios de aceptación	**Dada** la página de selección de opciones. **Cuando** se seleccionan las deseadas (o predeterminadas) y se pulsa el botón "JOIN SESSION". **Entonces** estas preferencias serán usadas en la sesión.

Tabla 5. Historia de usuario 5. Elaboración propia.

HU-6	Mutar audio
Descripción	Como usuario de la aplicación quiero poder silenciar mi micrófono dentro de la videollamada para no retransmitir sonido cuando lo desee.
Prioridad	Muy baja.
Estimación	Muy corta.

Conversación	❖ Seleccionar el botón de "Mute your audio" → Silencia el micrófono del usuario. ❖ Seleccionar el botón de "Unmute your audio" → Volverá a permitir que el micrófono del usuario retransmita sonido.
Criterios de aceptación	**Dadas** las opciones dentro de la videoconferencia. **Cuando** se pulsa el botón de "Mute/Unmute your audio". **Entonces** la retransmisión de sonido a través del micrófono se desactivará o activará respectivamente.

Tabla 6. Historia de usuario 6. Elaboración propia.

HU-7	**Mutar video**
Descripción	Como usuario de la aplicación quiero poder dejar de retransmitir mi cámara dentro de la videollamada para no dejar verla para los demás usuarios.
Prioridad	Muy baja.
Estimación	Muy corta.
Conversación	❖ Seleccionar el botón de "Mute your video" → Bloquea la cámara del usuario. ❖ Seleccionar el botón de "Unmute your video" → Volverá a permitir que la cámara del usuario retransmita.
Criterios de aceptación	**Dadas** las opciones dentro de la videoconferencia. **Cuando** se pulsa el botón de "Mute/Unmute your video". **Entonces** la retransmisión de video a través de la cámara se desactivará o activará respectivamente.

Tabla 7. Historia de usuario 7. Elaboración propia.

HU-8	**Compartir pantalla**
Descripción	Como usuario de la aplicación quiero poder compartir pantalla en las videoconferencias para poder mostrársela al resto de usuarios.
Prioridad	Media.
Estimación	Media.

Conversación	❖ Seleccionar el botón de “Enable screen share” → Inicia el proceso de compartir pantalla. ❖ Seleccionar el botón de “Disable screen share” → Deja de compartir pantalla. ❖ Seleccionar la pantalla a compartir deseada y pulsar “Aceptar” → Inicia la compartición de pantalla. ❖ Seleccionar el botón “Cancelar” tras el inicio de proceso de compartir pantalla → Cancela el proceso y vuelve todo a su estado original.
Criterios de aceptación	**Dada** la sesión iniciada de videoconferencia. **Cuando** se pulsa el botón “Enable/Disable screen share”. **Entonces** se permite compartir pantalla o dejar de compartirla respectivamente.

Tabla 8. Historia de usuario 8. Elaboración propia.

HU-9	**Ampliar cámaras**
Descripción	Como usuario de la aplicación quiero poder hacer más grandes las cámaras que desee de la videoconferencia para verlas mejor.
Prioridad	Media.
Estimación	Corta.
Conversación	❖ Seleccionar en la opción “Zoom in” de “Settings” de una cámara pequeña → Amplia esta cámara para que pertenezca al grupo de las agrandadas. ❖ Seleccionar en la opción “Zoom in” de “Settings” de una cámara agrandada → Retira la cámara de la clase de ampliadas y la vuelve del grupo de las pequeñas.
Criterios de aceptación	**Dada** la sesión iniciada de videoconferencia. **Cuando** se pulsa el botón “Zoom in” de “Settings”. **Entonces** se amplía o reduce la cámara a la que pertenezca dichas opciones en función del estado actual de la cámara.

Tabla 9. Historia de usuario 9. Elaboración propia.

HU-10	Ver cámaras dinámicamente
Descripción	Como usuario de la aplicación quiero que las cámaras de la sesión de videoconferencia se adapten al espacio disponible de la pantalla para que sea más agradable de ver.
Prioridad	Baja.
Estimación	Corta.
Conversación	❖ Cambiar el tamaño de la pantalla → Actualiza la disposición de las cámaras conectadas para que dejen el menor espacio en vacío posible. ❖ Conectarse a la sesión → Actualiza la disposición de las cámaras conectadas para que dejen el menor espacio en vacío posible. ❖ Desconectarse de la sesión → Actualiza la disposición de las cámaras conectadas para que dejen el menor espacio en vacío posible. ❖ Compartir pantalla → Actualiza la disposición de las cámaras conectadas para que dejen el menor espacio en vacío posible. ❖ Ampliar o reducir una cámara → Actualiza la disposición de las cámaras conectadas para que dejen el menor espacio en vacío posible.
Criterios de aceptación	**Dada** la sesión iniciada de videoconferencia. **Cuando** se produce algún cambio en el número de cámaras o del tamaño del espacio disponible. **Entonces** se ajustan las cámaras en pantalla automáticamente.

Tabla 10. Historia de usuario 10. Elaboración propia.

HU-11	Conectar a videoconferencia
Descripción	Como usuario de la aplicación quiero poder conectarme a una sala de videoconferencia para hablar y/o escuchar a otras personas.
Prioridad	Muy alta.
Estimación	Grande.
Conversación	❖ Seleccionar el botón de “JOIN SESSION” → Conectará al usuario a la sesión y trasladará al mismo a la pantalla de sesión.
Criterios de aceptación	**Dada** las opciones seleccionadas de videoconferencia. **Cuando** se pulsa el botón “JOIN SESSION”. **Entonces** se conecta el usuario a la sesión de videoconferencia.

Tabla 11. Historia de usuario 11. Elaboración propia.

HU-12	**Finalizar llamada**
Descripción	Como usuario de la aplicación quiero poder salir de la videoconferencia cuando desee para poder dar por finalizada la llamada.
Prioridad	Alta.
Estimación	Corta.
Conversación	❖ Seleccionar el botón de "Leave the session" → Desconecta al usuario de la sesión y lo devuelve al menú Home.
Criterios de aceptación	**Dada** la sesión iniciada de videoconferencia. **Cuando** se pulsa el botón "Leave the session". **Entonces** se desconecta al usuario de la sesión.

Tabla 12. Historia de usuario 12. Elaboración propia.

HU-13	**Mutar audio de otro usuario**
Descripción	Como usuario de la aplicación quiero poder silenciar a otro usuario para no escuchar su micrófono sin afectar a otros usuarios.
Prioridad	Baja.
Estimación	Corta.
Conversación	❖ Seleccionar en la opción "Mute sound" de "Settings" de otro usuario → Silencia al usuario exclusivamente para el usuario que ha pulsado esta opción. ❖ Seleccionar en la opción "Unmute sound" de "Settings" de otro usuario → Vuelve a conectar el sonido del usuario previamente silenciado exclusivamente para el usuario que ha pulsado esta opción.
Criterios de aceptación	**Dada** la sesión iniciada de videoconferencia. **Cuando** se pulsa el botón "Mute/Unmute sound" de "Settings". **Entonces** silencia o se vuelve a conectar el sonido de esa cámara para el usuario que seleccionado la opción respectivamente.

Tabla 13. Historia de usuario 13. Elaboración propia.

4.1.2 Requisitos no contemplados al principio

A continuación, se dividirán en tres listas los requisitos que posee el proyecto pero que, debido a su carácter anecdótico, estético o que se encontraba oculto en el propio código del proyecto y no era sencillo observarlo a simple vista, no se llegaron a contemplar en un principio como las historias de usuario del apartado anterior:

- **Requisitos funcionales:**
  - Validación de nombres de sala: No permitiendo nombres vacíos o demasiado cortos.
  - Formateo de nombres de sala: separándolos con "-" en vez de espacios y solo permitiendo letras y no puntos o símbolos.
  - Crear un interceptor de peticiones web al que se le añada las credenciales de usuario una vez se haya iniciado sesión en la página web.
  - Pulsar la tecla "enter" con el cursor seleccionado en los cuadros de texto de Login y Join hace que se realice la función de los botones "LOGIN" y "JOIN" respectivamente.
  - Bloquear el acceso a JoinSession y/o Session sino se ha iniciado la sesión de usuario de manera automática o manual.
  - Bloquear cámara a sí mismo en la sección de JoinSession.
  - Silenciar el micrófono a sí mismo en la sección de JoinSession.
- **Requisitos no funcionales:**
  - Diseño similar a la de OpenVidu Call de la página de Login.
  - Diseño similar a la de OpenVidu Call de la página de Join.
  - Diseño similar a la de OpenVidu Call de la página de JoinSession.
  - Diseño similar a la de OpenVidu Call de la página de Session.
  - Ocultar contraseña.
  - Cambiar el símbolo y fondo del botón de silenciar micrófono cuando se pulse.
  - Cambiar el símbolo y fondo del botón de bloquear cámara cuando se pulse.
  - Cambiar el fondo del botón de compartir pantalla cuando se encuentre activado.
  - Añadir el símbolo de "silenciado" cuando un usuario se encuentre silenciado por su propia voluntad y que sea visible para todos los demás usuarios de la sesión.
- **Requisitos inesperados:**
  - Cambiar el símbolo y etiqueta de "Zoom in" en función de si el componente posee la etiqueta "OV_big" o no, es decir, que cuando la cámara esté en grande

aparezca “Zoom out” y en caso contrario “Zoom in” en un funcionamiento parecido al de “Mute/Unmute sound”.

- No bloquear los componentes de selección de micrófono y cámara en JoinSession cuando se silencie o se bloquee la cámara en esa pantalla para que, aunque el usuario no quiera ser escuchado o visto nada más entrar a la llamada, pueda escoger cómo será escuchado o visto si decide cambiar de opinión en mitad de la llamada.

4.2 Arquitectura y Análisis

A continuación se explicarán dos diagramas que ayudarán a la comprensión de la estructura del proyecto.

4.2.1 Diagrama de flujo

Cuando se habla de una aplicación web es muy importante las interacciones del usuario con la misma y el cómo eso afecta a la navegación del usuario. Por ello se ha optado por mostrar los pasos que seguiría un usuario que accediera a la aplicación de OpenVidu Call de Vue. Es más, este flujo es el que se utiliza como referencia para la prueba end-to-end que se explicará en la sección 4.4.2 más adelante.

Comenzando por observar el Diagrama 2, se puede observar las cuatro partes de la aplicación bien diferenciadas y separadas por los carteles que anuncian la entrada a una nueva sección y cuyo nombre indicativo durante todo el documento se encuentra entre paréntesis. Nada más iniciar la aplicación se inicia la página de Login, en caso de que con anterioridad el usuario haya iniciado su sesión y no haya salido de la misma entonces se recuperará la información guardada en la cookie y se usarán esos datos para iniciar la sesión automáticamente y se pasa a la siguiente fase. En caso de que no haya sucedido esto entonces se le solicita los datos pertinentes al usuario hasta que estén correctos y ya se puede llegar a la sección de Join. En esta se da la opción de cerrar la sesión, volviendo a la sección anterior y borrando los datos guardados. Sin embargo, si se decide continuar se puede introducir el nombre de sala deseado (o generado aleatoriamente) y pulsar el botón para validar y en caso de que cumpla los requisitos entonces se continua a la sección de JoinSession. En esta sección se puede escoger entre los diferentes dispositivos de cámaras y micrófonos conectados al ordenador, editar el nombre de

usuario aleatorio y decidir si se desea entrar en la llamada con el micrófono y/o la cámara bloqueados. Una vez hecho este paso ya se puede entrar en la llamada, dentro de la misma se pueden usar alguna de las funciones disponibles, siendo estás: silenciar/conectar el micrófono, bloquear/desbloquear la cámara, compartir o no la pantalla, aumentar o disminuir cualquiera de las cámaras conectadas a la llamada y silenciar el sonido de cualquiera de los usuarios que no sea a sí mismo. Independientemente de si se usa cualquiera de estas opciones siempre se puede abandonar la llamada mediante el botón de colgar y el usuario será trasladado a la sección de Join automáticamente.

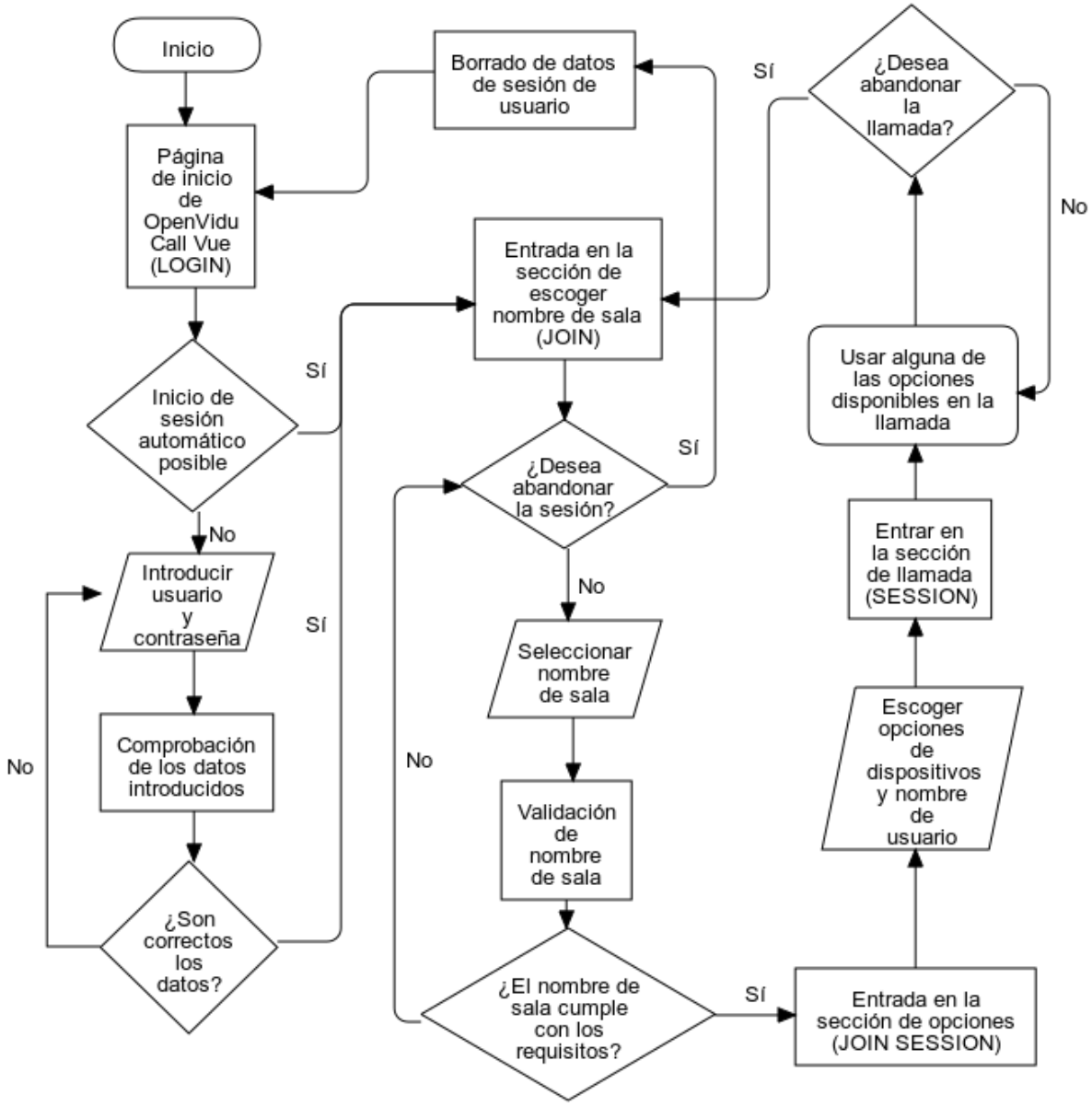

Diagrama 2: Diagrama de flujo. Elaboración propia.

4.2.2 Diagrama de secuencia

No existen muchos diagramas que aporten información importante respecto las llamadas entre clases, ya que Vue está ideado para agrupar los archivos del frontend en uno solo. Aun así, hay una parte que resulta interesante mirar en profundidad para comprender cómo se comunican el frontend y el backend, y esta es cuando se desea obtener los tokens que usará OpenVidu para conectar al usuario a la llamada, como se puede ver en el Diagrama 3.

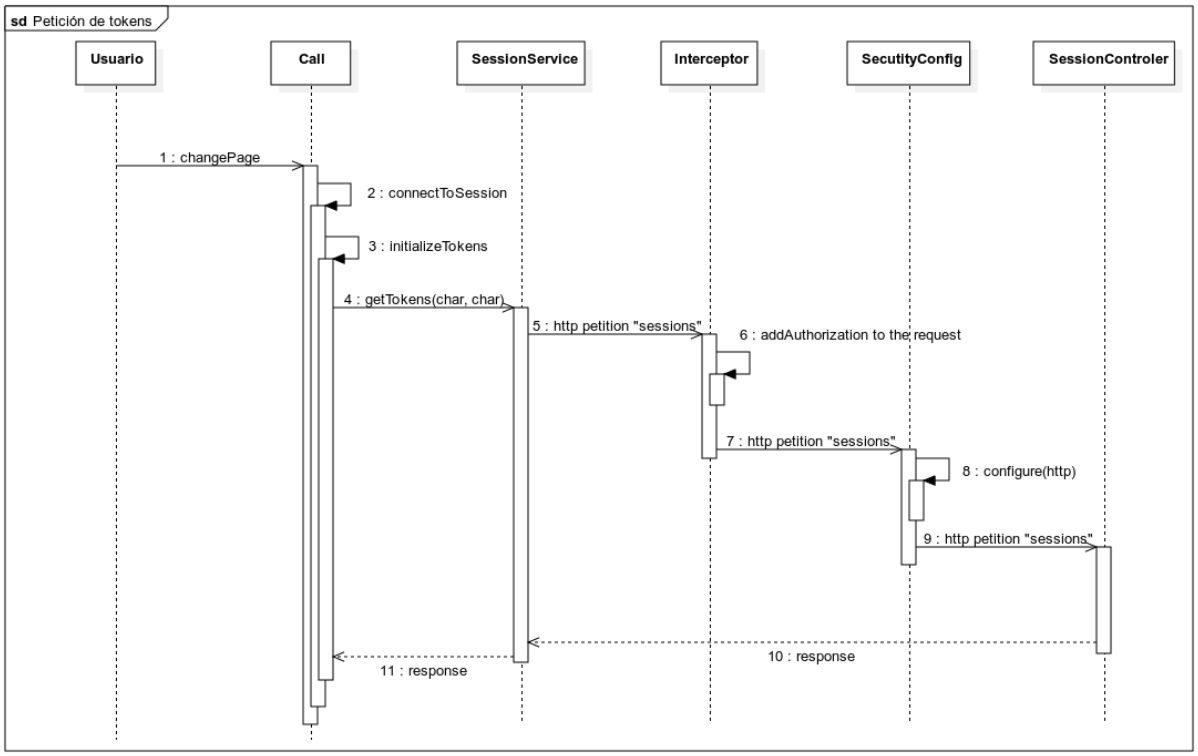

Diagrama 3: Diagrama de secuencia. Elaboración propia.

Todo comienza cuando el usuario pulsa el botón de JOIN SESSION iniciando la función changePage(), esta función llamará a la de connectToSession() que a su vez llamará a la de initializeTokens(). Todos estos métodos se encuentran dentro de Call.vue. A continuación, se llamará a la función getTokens(sessionId, nickname) de SessionService.js, que es uno de los dos servicios junto a AuthService.js que se encarga de conectar el frontend con el backend. Este servicio lanzará una petición POST a la ruta del backend en http::/localhost:5000/sessions, sin embargo, existe un interceptor que se encargará de añadir a dicha petición la autorización requerida incluyendo las credenciales de usuario encriptadas en la cabecera de la petición antes de que llegue al backend. A continuación, esta petición es captada por la configuración de seguridad especificada en SecurityConfig.java, entrando así en la primera clase perteneciente

al backend, la seguridad se asegura de que esa cabecera de autentificación exista y deja pasar la petición al SessionControler.java, que finalmente especificará los tokens y los devolverá al SessionService.js que a su vez los pasará al Call.vue donde podrán ser usados.

4.3 Diseño e Implementación

A continuación se describirán algunos de los problemas que se enfrentaron a lo largo del desarrollo, así como aspectos más concretos del mismo.

4.3.1 Integración entre librerías problemáticas

De las estimaciones previstas en las historias de usuario la mayoría se ajustó en mayor o menor medida a lo previsto, siendo quizás más tiempo gastado en la comprensión de todas las partes que componen cada tarea. Sin embargo el uso de una librería en concreto llevó a semanas de problemas, implementaciones a medias, gran cantidad de correos de consulta enviados y hasta una reunión a través de Teams con el tutor y un desarrollador de OpenVidu. Esta librería no es nada más y nada menos que la de opentok-layout-js.

En un principio la librería presentaba un aspecto sencillo de implementación, ya que es la idea principal detrás de toda esa librería, sin embargo, no parecía funcionar correctamente así que se decidió pasar a un nivel más sencillo e implementarlo en el tutorial de OpenVidu de Vue disponible en la página web. Esto llevó a una versión rústica de la misma, funcionaba correctamente pero no aplicaba ciertos valores. Por ejemplo, no animaba las cámaras en el cambio de tamaño de pantalla ni se notaba ninguna diferencia en los cambios de valores de tamaño, por ejemplo. Tras mucho tiempo intentando arreglar ese problema e investigar maneras alternativas de tratar con esa librería (como el uso de boxes) se decidió consultar mediante correo al tutor y a uno de los desarrolladores de OpenVidu Call.

Esa consulta llevó a bajar a un nivel todavía más bajo e intentar un proyecto Vue de cero que solo contenga elementos para organizar. Concretamente se hizo una página sencilla en el que había cuatro elementos textuales, estos elementos nuevamente no mostraban ninguna clase de animación como se esperaba ni se terminaba de extender verticalmente por la página, solo horizontalmente. Por supuesto se intentó implementar directamente el código descrito en la propia librería de opentok-layout-js, sin embargo, para ello se necesitaba una api_key que solo se podía obtener registrándose en la página oficial de Opentok [34], debido a ser una página de

pago aunque con un periodo de prueba gratuito y que no iba a ser algo fundamental para el proyecto ya que la librería de opentok-layout-js no necesitaba de nada más para funcionar se decidió abandonar esa línea de trabaja.

Finalmente debido a todos estos problemas se estableció una reunión en la que se acordó que Carlos Santos (desarrollador de OpenVidu y una de las personas que ha ayudado a resolver dudas a lo largo del proyecto), crearía esa página web en JavaScript básico para poder entender mejor el funcionamiento de la propia librería. Dicho ejemplo sería la clave para por fin comprender de verdad el funcionamiento de la librería y poder implementarlo finalmente en este trabajo de fin de grado. Pronto se entendió una de las partes claves para la correcta visualización de los elementos que no aparecía especificado en la documentación de GitHub de la librería, quizás por sobreentenderse, y es que la librería necesitaba de componentes CSS que establecieran ciertos parámetros para su correcta visualización. Por ejemplo, las animaciones vienen dadas por una propiedad de un componente que establece tanto el tipo de animación como el tiempo que tarda en efectuarla, o que los elementos debían tener valores de altura y ancho para que la librería entendiera hasta dónde abarcaba el espacio disponible para trabajar.

Aun así, no se confiaría directamente en los nuevos conocimientos y se procedería a realizar cada paso desde lo más simple a lo más complejo sin saltarse nada. Primero se comprendió el código del ejemplo proporcionado y se probarían sus capacidades cambiando valores, después se realizó el mismo ejemplo, pero en un proyecto de Vue simple. A continuación, se pasó ese ejemplo de cuadros de colores en el proyecto de tutorial de OpenVidu Call para Vue para confirmar que la librería funcionaba correctamente antes de añadir elementos más complejos como las cámaras. Hecho este paso sin muchos problemas se pasó a implementar la librería, pero con una sola cámara. Una vez hecho ese paso solo se necesitaba implementar para varias. Este era el último paso antes de avanzar a implementar la librería en la aplicación final. En este punto se tuvo que cambiar algunos elementos, como por ejemplo la manera de llamar a initLayoutContainer, antes se podía usar la función require, pero al llamar ahora a opentok-layout-js desde un método en vez de en la sección mounted se tuvo que importar directamente la función al principio del script. También se tuvo que añadir otras clases que previamente no había hecho falta como OT_video-element a ov-video de UserVideo. En definitiva, la mayor cuestión para que se ajustaran correctamente las cámaras eran cuestiones de estilo de altura y anchura a otros elementos, como ya se ha mencionado anteriormente. Por supuesto, no hay que olvidar que no solo cuando se cambia el tamaño de la pantalla se debe actualizar el layout de opentok-layout-js, sino también cuando nuevos usuarios se conectan o

desconectan al igual que cuando se agranda o empequeñece alguna cámara y se comparte pantalla. En todos esos casos ha sido clave la gestión de las llamadas de actualización del layout de manera automática.

El resto de las librerías de terceros no presentaron ningún problema para su implementación.

4.3.2 Resolución de bugs

A continuación, se enumerarán los diferentes problemas encontrados y su respectiva solución:

- **Petición al backend emitida correctamente, pero con resultado negativo:** Una vez se había completado las funcionalidades asociadas a la página de Home con el apartado de Login completado se encontró que al intentar hacer la petición /sessions había algún problema que no se conseguía determinar con claridad. En un principio se barajeó que el problema fuera relacionado con la manera de desplegar el frontend con el comando:
  "npm run dev"
  y se intentó cambiar por:
  "npm run serve"
  Pero esto no llegó a ningún resultado. Examinando detenidamente la petición se descubrió que se enviaba de manera encriptada las credenciales de usuario.

  En este punto es cuando se descubrió la presencia del interceptor de peticiones web en la aplicación original, por lo que se añadió como funcionalidad a implementar más adelante y temporalmente se continuó trabajando cambiando una de las propiedades del backend del archivo application.properties: "CALL_PRIVATE_ACCESS: DISABLED" sustituyendo a "CALL_PRIVATE_ACCESS: ENABLED".
- **Modificación del favicon:** Se deseaba añadir un favicon del icono de OpenVidu a la página web y saltaba un error con ello. Resultaba que se estaba intentando hacer cargar una imagen .png en vez de formato .ico para este trabajo. Se desconocía que debía tener ese formato. Una vez descubierto fue tan simple como encontrar el archivo apropiado.
- **Problemas con el centrado en estilo:** En la primera fase de diseño básico no hubo problemas en el apartado más estético pues tan solo se deseaba desentrañar qué clase de componentes se iban a necesitar. Sin embargo, cuando ya se pasó a perfilar el apartado artístico se encontró un problema para centrar los componentes adecuadamente. Las

instrucciones CSS no parecían hacer su función correctamente. Tras investigar se descubrió que era debido a la falta de la etiqueta v-app que engloba todo el proyecto Vuetify. Una vez incluido todos los componentes del proyecto se distribuían adecuadamente. Esta etiqueta era necesaria para que Vuetify leyera correctamente los componentes.

- **Inicio de la web por primera vez error 404:** Cuando se iniciaba la página web por primera vez se mostraba el error "Failed to load resource: the server responded with a status of 404 (Not Found)" del archivo VApp.sass. Consultando diferentes fuentes algunos usuarios sugerían que la url base debía ser "" en vez de "/", sin embargo, esto no produjo ningún cambio notable.

  La clave de la solución residió en añadir una importación a la clase App.vue principal, siendo esta: import 'vuetify/dist/vuetify.min.css'. Esto es debido a los componentes de estilo que carga vuetify al iniciarse, así cuando se inicie esperará a que estos se carguen adecuadamente antes de continuar. Además, es importante que el proyecto cuente con la última versión de sass para que no se vuelva a producir este problema.

- **Error de uso en plataformas móviles:** Cuando se estaba implementando el apartado referente a compartir pantalla se detectó un error en el navegador que impedía la conexión de la sesión exclusiva de compartir pantalla debido a la incompatibilidad con navegadores en dispositivos móviles, este error no tenía sentido ya que el desarrollo se hizo completamente en ordenador, y además solo se producía en ocasiones. Para restaurarlo y que volviera a funcionar era necesario reiniciar la aplicación desde la consola de comandos.

  Como no era una solución viable se consultó y en una reunión fue señalado que se tenía activado la opción "Device toolbar" de la consola del navegador, esto hacía que el servidor de OpenVidu lo detectara como un dispositivo móvil y por tanto daba problemas. Al desactivarlo se solucionó.

- **Error ICE_CONNECTION_DISCONNECTED:** Cuando un usuario se conecta a una llamada el navegador de cada uno ha de actualizarse y crear la conexión y el espacio adecuado para ese nuevo subscriber. Sin embargo, desde prácticamente el principio del desarrollo se notó que a veces tardaba demasiado en crearse esa conexión, un tiempo bastante anómalo y en el peor de los casos no parecía conectarse nunca.

En una reunión se acabó comentando este problema y se determinó que era cuestión del despliegue local de Docker, probablemente debido a que el desarrollo fue hecho en el sistema operativo de Windows 10 en vez de Linux. Aun así, para probar correctamente todas las funcionalidades se sustituyó temporalmente Docker por una conexión al servidor de demos [35].

4.3.3 Aspectos importantes del desarrollo

A continuación se comentarán diferentes aspectos de la implementación de OpenVidu Call en vue:

- **Control de opciones de usuario:** Una de las partes más interesantes de realizar fue el control de opciones por usuario dentro de la llamada. Cada usuario tiene su propia lista de opciones independientes del resto y por supuesto cada sesión puede tener su propia configuración, por ejemplo, un usuario puede tener silenciado a otro y no por ello un tercero tiene que verse afectado. Eso quiere decir que cada usuario que se conecte a la llamada tendrá que controlar una lista de opciones para cada usuario conectado, o lo que viene a ser una lista de listas.

  Cuando un usuario se introduzca a la sesión, es decir, que se active el evento de streamCreated, cuando esto sucede no solo el nuevo stream se subscribe a la sesión del usuario principal, sino que se añade una nueva lista de opciones cuyo índice corresponderá al mismo que el susbcriptor a su lista de subscriptores, tal y como se puede ver en la Figura 1.

```
// Specify session´s publisher behavior
this.sessionPublisher.on("streamCreated", async ({ stream }) => {
  const subscriber = this.sessionPublisher.subscribe(stream);
  this.subscribers.push(subscriber);
  await this.optionsSubscriber.push(
    [
      {name: "mute",
      activated: subscriber.stream.audioActive,
      activatedLocal: true},
      {name: "zoom",
      activatedLocal: false,
      }
```

```
    ]
  );
  this.updateLayout();
});
```

Figura 1: Fragmento de código de streamCreated. Elaboración propia.

Como se puede ver, cada opción tendrá sus propiedades necesarias, pero todas ellas necesitarán de un nombre y correspondería al menos una propiedad que indicase si está activo o no. Esto servirá más adelante para controlar el aspecto de las opciones además de tener una variable de control en caso de necesitarlo.

Por supuesto las opciones realizarán acciones y la implementación de estas se encuentra en el método, que se puede ver en la Figura 2, al que se le pasa como variable la opción seleccionada y al usuario al que se le va a aplicar esa opción. En cualquiera de las dos opciones disponibles se cambia la variable de activada y después se realiza la acción, en el caso de "mute" se suscribe (o cancelar la suscripción en función de la variable) al audio de ese usuario. Mientras que en el caso de "zoom" se añade o quita la clase OV_big del elemento de video correspondiente, con una comprobación extra por si el elemento al que hay que añadir dicha clase es la del propio usuario que realiza la acción, ya que en ese caso el elemento de vídeo se manera ligeramente diferente.

```
async actionListOptions(option, user) {
  switch (option.name){
    case 'mute':
      option.activatedLocal = !option.activatedLocal;
      user.subscribeToAudio(option.activatedLocal);
      break;
    case 'zoom':
      option.activatedLocal = !option.activatedLocal;
      if (user.stream.connection.connectionId ==
this.publisher.stream.connection.connectionId) {
        document.getElementById("Publisher").classList.toggle('OV_big');
      } else {
        document.getElementById(user.stream.connection.connectionId).class
List.toggle('OV_big');
```

```
      }
      this.updateLayout();
      break;
  }
},
```

Figura 2: Fragmento de código de actionListOptions. Elaboración propia.

Como elemento final, se muestra en la Figura 3 el cómo se conforma el apartado de diseño de estas opciones. Sin entrar en detalles de los diferentes props, es notable destacar que para realizar un menú siempre es necesario incluir una lista con sus elementos, que en este caso se construyen recorriendo la lista de opciones disponibles, el Publisher solo posee la opción de "zoom", así que se muestra la composición de diseño de las opciones de los subscribers. En función de la opción que corresponda entonces se le incluye un diseño u otro además de tener en cuenta si está activo o no para los diferentes iconos y nombres.

```
<div class="circles_options right">
  <v-menu>
    <template v-slot:activator="{ props: menu }">
      <v-tooltip location="top">
        <template v-slot:activator="{ props: tooltip }">
          <v-btn class="background_tonal" variant="tonal" size="x-small"
icon v-bind="mergeProps(menu, tooltip)">
            <v-icon size="x-small" icon="mdi-dots-vertical"/>
          </v-btn>
        </template>
        <span>Settings</span>
      </v-tooltip>
    </template>
    <v-list size="x-small">
      <v-list-item v-for="(option, i) in optionsSubscriber[indexSub]"
:key="i" :value="option" @click.stop="actionListOptions(option, sub)">
        <template v-slot:prepend v-if="option.name == 'mute'">
          <v-icon :icon="option.activatedLocal ? 'mdi-volume-high' : 'mdi-
volume-off'"/>
```

```
          <v-list-item-title v-text="option.activatedLocal ? 'Mute sound'
: 'Unmute sound'"/>
        </template>
        <template v-slot:prepend v-if="option.name == 'zoom'">
          <v-icon icon='mdi-magnify-plus-outline'/>
          <v-list-item-title v-text="'Zoom in'"/>
        </template>
      </v-list-item>
    </v-list>
  </v-menu>
</div>
```

Figura 3: Fragmento de código de las opciones por usuario en llamada.
Elaboración propia.

- **Control de la previsualización del vídeo antes de entrar a la llamada:** A primera instancia el control del Publisher (o vídeo) y de los diferentes componentes no es algo que se pueda considerar complejo, sin embargo, el proceso para llegar hasta ese punto sí que lo fue. La estimación de la HU-5 era de corta, pero se tardó tiempo en terminar de comprender cómo organizar los elementos de conectar a la sesión explicados en los tutoriales de OpenVidu Vue para poder crear un vídeo que cambiara en función de los valores que se escogieran. Por un lado, se encuentra la sección "created" que se activa recién se crea la página y es en la que se inicializan los diferentes elementos de OpenVidu y se inicia la sesión que controlará la llamada y el Publisher junto con los distintos eventos que deberán tener en cuenta como la creación de streams, destrucción de los mismos, excepciones y cuando haya un cambio de propiedades.

  Finalmente se entra en la parte en la que se obtiene el listado de los distintos dispositivos de vídeo y audio conectados. Una vez obtenidos se selecciona el primero para el seleccionador correspondiente y estos son usados para iniciar el Publisher que será mostrado en pantalla mediante el método updateInputSource(), tal y como se puede ver en la Figura 4.

```
// Get input sources
this.OVPublisher.getDevices().then(devices => {
```

```
  var videoDevices = devices.filter(device => device.kind ===
'videoinput');
  for (var numDevice in videoDevices) {
    this.cameras.push({name: videoDevices[numDevice].label , id:
videoDevices[numDevice].deviceId});
  }

  var microphoneDevices = devices.filter(device => device.kind ===
'audioinput');
  for (var numDevice in microphoneDevices) {
    this.microphones.push({name: microphoneDevices[numDevice].label , id:
microphoneDevices[numDevice].deviceId});
  }

  this.camera = this.cameras[0].id;
  this.microphone = this.microphones[0].id;
  this.updateInputSource();

});
```

Figura 4: Fragmento de código de getDevices. Elaboración propia.

Cada vez que se seleccione una cámara o micrófono diferente se volverá a iniciar un nuevo Publisher con los valores deseados que sustituirá al anterior. Para ello se cambiarán las variables cámara y micrófono según corresponda y se llamará de nuevo a updateInputSource() mostrado en la Figura 5.

```
updateInputSource() {
  let publisher = this.OVPublisher.initPublisher(undefined, {
    audioSource: this.microphone,
    videoSource: this.camera,
    publishAudio: this.audio_activate,
    publishVideo: this.video_activate,
    frameRate: 30,
    insertMode: "APPEND",
    mirror: false,
  });
```

```
  this.mainStreamManager = publisher;

  this.publisher = publisher;
},
```

Figura 5: Fragmento de código de updateInputSource. Elaboración propia.

Cuando se pase a iniciar la llamada este Publisher será el que se publique para la sesión tras la conexión a los servidores de OpenVidu.

Como se ha podido ver, el proceso no es excesivamente complicado en sí, sin embargo, por las interacciones entre diferentes componentes y la compartimentalización de las partes se tardó bastante tiempo en llegar a esta solución en vez de intentar realizar la conexión a la sesión cuando todavía no se le había dado la oportunidad al usuario de seleccionar sus preferencias.

- **Compartir pantalla:** Una vez comprendido todos pasos necesarios para crear sesiones y el tratamiento del Publisher se encuentra que para compartir pantalla hay que tener en cuenta un par de detalles. Uno de ellos es un limitante, y es que por cada sesión solo se puede publicar un solo Publisher, por lo que para compartir pantalla es necesario crear otra sesión a parte como si fuera un segundo usuario conectándose a la sesión. Por ello se sigue los mismos pasos que el Publisher normal del usuario y se añade sus propias propiedades de en caso de destrucción de stream y excepción, no es necesario que realice ninguna acción cuando se crean otros streams, porque es el Publisher normal el que realizará estos. Por supuesto se tuvo que modificar el código para cerrar las conexiones de la pantalla compartida a la vez que la de la cámara cuando fuera necesario. En la Figura 6 se puede ver el cómo se inicializa el compartir pantalla.

```
shareScreen() {
  if (!this.screen_activate) {
    this.publisherScreen = this.OVScreenShare.initPublisher(undefined, {
      videoSource: "screen",
      publishAudio: false,
      frameRate: 30,
      insertMode: "APPEND",
    });
```

```
    this.publisherScreen.once('accessAllowed', (event)=> {
      this.publisherScreen.stream.getMediaStream().getVideoTracks()[0].a
ddEventListener('ended', () => {
        console.log('User pressed the "Stop sharing button"');
        this.sessionScreenShare.unpublish(this.publisherScreen);
        this.publisherScreen = undefined;

        this.screen_activate = false;
      });

      this.sessionScreenShare.publish(this.publisherScreen);
    });


    this.publisherScreen.once('accessDenied', (event) => {
      console.warn('ScreenShare: Access Denied');
      this.screen_activate = false;
    });
  } else {
    this.sessionScreenShare.unpublish(this.publisherScreen);
    this.publisherScreen = undefined;
  }

  this.screen_activate = !this.screen_activate;
},
```

Figura 6: Fragmento de código de shareScreen. Elaboración propia.

4.3.4 Métricas y evolución

Una vez realizado todo el proyecto es interesante entrar en la evaluación de las diferentes métricas que nos pueden ayudar a comprender el tamaño del proyecto. Se podría comprobar el porcentaje de cada lenguaje mediante el repositorio GitHub, sin embargo, este es impreciso ya que dentro del apartado HTML incluye documentos que no debería (como los de Vue). Así que para enseñar los resultados se ha usado VS Code Counter, una extensión de Visual Studio Code. Para usarlo simplemente hay que darle click derecho al directorio que se quiere comprobar y

seleccionar “Count lines in directory”. Además, los resultados generados te los divide entre código, en blanco y comentarios, así que da más información que GitHub. En la Tabla 14 se muestran los resultados del repositorio completo.

Languages

language	files	code	comment	blank	total
JSON	4	5,189	0	4	5,193
Java	12	992	53	257	1,302
Vue	5	969	5	75	1,049
JavaScript	12	347	124	101	572
Markdown	3	114	0	58	172
HTML	2	72	0	7	79
YAML	2	61	0	4	65
XML	1	45	0	11	56
Docker	1	19	0	13	32
Java Properties	1	12	0	3	15
JSON with Comments	1	8	12	0	20
Shell Script	1	1	1	0	2
SCSS	1	0	9	2	11

Tabla 14. Lenguajes del proyecto. VS Code Counter.

Como se puede observar, el lenguaje con mayor cantidad de líneas de código es JSON, esto es debido principalmente por la gran extensión del archivo packaje-lock.json del frontend, eso es código generado automáticamente así que no tiene gran valor a la hora de valorar el trabajo. Sin embargo, los apartados de Java y Vue sí que son más significativos ya que se puede observar que a grandes rasgos, backend y frontend tienen un tamaño parecido, con la diferencia de que Vue lo hace en menos de la mitad de los archivos, siguiendo así la idea de la que nació Vue de juntar todos sus componentes en un mismo archivo.

Finalmente, es notable mostrar la división de lenguaje del frontend mediante la Tabla 15. Como se puede observarse sigue manteniendo la misma línea que el anterior, con la adición de que así se puede diferenciar los archivos JavaScript a cuáles de las dos partes pertenecen. Demostrando así que el frontend posee más JavaScript que el backend. Estos datos provienen de los tres archivos usados para comunicarse con el backend (AuthService, axiosInterceptor y SessionService) y el resto son archivos propios de Vuetify y Vue.

Languages

language	files	code	comment	blank	total
JSON	2	4,625	0	2	4,627
Vue	5	969	5	75	1,049
JavaScript	10	216	32	47	295
Markdown	1	39	0	19	58
HTML	1	13	0	4	17
JSON with Comments	1	8	12	0	20
SCSS	1	0	9	2	11

Tabla 15. Lenguajes del frontend. VS Code Counter.

Como punto final de este apartado, en el Diagrama 4 se muestra la evolución del repositorio GitHub en un diagrama de barras que muestra la cantidad de commit por semana. Como se puede observar, se comenzó a trabajar en el repositorio el 1 de febrero de 2023 y se ha mantenido el desarrollo hasta el 27 de junio de 2023.

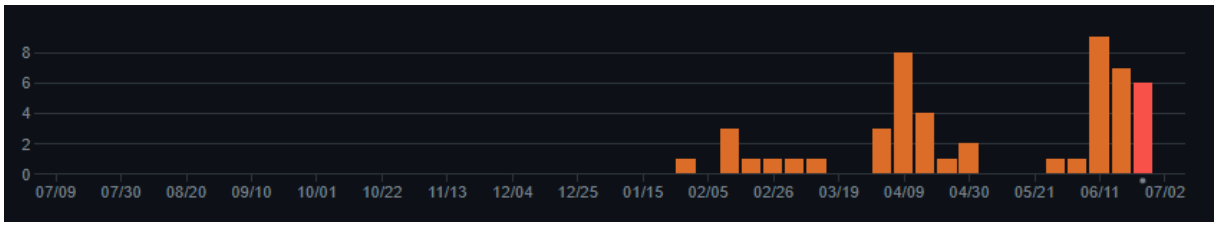

Diagrama 4. Diagrama de barras de cantidad de commits. GitHub.

4.4 Pruebas

Las pruebas son una parte fundamental en todo desarrollo y en el desarrollo web nos permite comprobar que todos los componentes con los que interaccionará el usuario funcionan adecuadamente. Es por ello por lo que se ha decidido hacer dos tipos de pruebas: pruebas de componentes de caja negra y una prueba end-to-end para comprobar que las funcionalidades básicas de la página web están operativas. Playwright se puede configurar para realizar todas las pruebas que se deseen en distintos navegadores, de manera predeterminada siempre son Chromium [36], FireFox y Webkit [37], pero también se puede configurar para incluir otros navegadores en la batería de pruebas, debido a que el desarrollo se realizó usando Google Chrome y aparecieron algunos problemas para garantizar los permisos de dispositivos en otros navegadores e incompatibilidad de realizar las pruebas varios navegadores a la vez, se decidió usar solo Google Chrome/Chromium únicamente. En la Tabla 16 se muestra los resultados satisfactorios de las pruebas en dicho navegador.

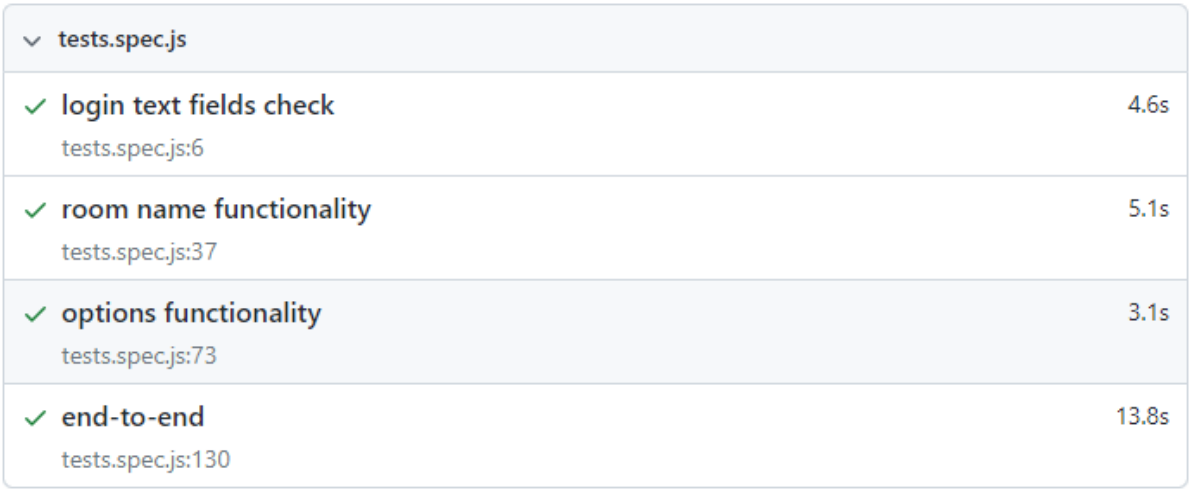

Tabla 16.Resultados de los test del proyecto. Playwright.

4.4.1 Pruebas de componentes

Como ya se ha dicho anteriormente, las pruebas de componentes serán clasificadas como pruebas de caja negra. Las pruebas de caja negra se definen como aquellas en las que no es necesario conocer la estructura interna del código y se busca verificar que cumple las funcionalidades pertinentes. Dicho lo cual se ha asegurado la correcta funcionalidad de tres apartados en los que el usuario necesita seleccionar o introducir datos.

- **Comprobación de los campos de Login:** Los apartados de campos de texto “Username” y “Password” junto con el botón de “LOGIN” componen un sistema de verificación de las credenciales y de la validación de los datos. El test comprueba los diferentes estados en los que se puede encontrar este apartado.

  En primera instancia se asegura que el botón de “LOGIN” está desconectado hasta que ambos campos se encuentren rellenos. Luego se comprueba que el mensaje de error no ha aparecido para que cuando se rellenen erróneamente las credenciales se compruebe que efectivamente ha aparecido en ese momento. Finalmente se comprueba el caso correcto en el que se rellenan correctamente las credenciales y se espera a que se transporte al usuario a la siguiente sección (Join) así asegurando que se ha iniciado sesión.

- **Comprobación de los campos de Join**: Este formulario está compuesto por un único campo de texto, un botón que genera nombres aleatorios y el botón de submit (también llamado JOIN).

  En primer lugar, pasa por la sección de Login poniendo las credenciales correctas, ya que esta parte no se puede evitar, y después pasa a comprobar la funcionalidad de generar un nombre de sesión aleatorio recuperando el nombre con el que empieza la web, pulsando el botón, recuperando el segundo nombre y comparando ambos. Si son diferentes entonces es que el botón funciona adecuadamente e introduce un nombre aleatorio en el apartado de nombre de sala.

  Por supuesto también necesita otras validaciones, siendo estas que siempre debe haber un nombre de sesión para que el botón de JOIN traslade al usuario a la siguiente pantalla y que el nombre introducido tenga una longitud mínima, siendo esta de seis caracteres. Como se puede intuir para comprobar ambos casos se rellena el cuadro de texto de ningún carácter o de cinco (por ejemplo), se realiza la acción de pulsar el botón de JOIN y después se comprueba que se muestra el mensaje de advertencia correspondiente, tal y como se hizo en el test anterior. Finalmente se comprueba el caso correcto en el que se rellena el campo con un nombre lo suficientemente largo y que después se traslada al usuario a la pantalla correcta.

- **Comprobación de los campos de JoinSession:** Las opciones previas a entrar en una llamada están compuestas por dos tipos de elementos que serán comprobados, el nombre de usuario para entrar en la sesión y las listas desplegables de micrófono y cámara.

Se comienza igual que el anterior pasando rápidamente por las secciones de Login y Join rellenando todo adecuadamente para poder llegar a la sección de llamada. Para comprobar la primera opción es tan sencillo como rellenar el campo de texto de "Nickname" con un nombre de prueba. Sin embargo, las otras dos listas son más complicadas de comprobar. Para comenzar será necesario comprobar que las listas no están vacías, para ello previamente se ha dado permisos de cámara y micrófono al navegador. Esta característica también implica que es necesario poseer cámara y micrófono para que este test funcione correctamente.

Entrando en la implementación de las pruebas de las listas, se obtiene en primera instancia la lista de los elementos que compone la lista de "Video device", una vez hecho esto se filtra dicha lista para que no haya elementos vacíos ni repetidos y se comprueba que esa lista filtrada tenga al menos un elemento. Para finalizarlo se escoge el último elemento de la lista para seleccionarlo. Se repite este proceso para los micrófonos con la adición de que en el filtrado de "Audio device" se comprueba que no haya ningún elemento de la lista de cámaras. Esta última condición ha sido necesaria de añadir debido a que por la gran velocidad del test capta la lista de las cámaras como si pertenecieran a la de micrófonos.

Cabe destacar que los últimos dos test no funcionarán correctamente si la sección de Login y Join no trasladan adecuadamente al usuario. No se ha podido encapsular mejor cada test debido a que siempre se necesita pasar por la fase de inicio de sesión cuando se usa la aplicación por primera vez ya que las pruebas no tienen acceso a cookies para iniciar automáticamente la sesión entre test y test.

4.4.2 Prueba end-to-end

Una prueba end-to-end quiere decir que es un tipo de test que imita los pasos que realizaría un usuario para manejar la aplicación. De esta manera esta prueba se encargará de comprobar que el usuario puede navegar a las cuatro secciones de la aplicación y que los valores predeterminados funcionan correctamente tanto en Join como en JoinSession. Estos valores predeterminados son el nombre de sesión y de usuario en las secciones Join y JoinSession correspondientemente, y la selección de los dispositivos de vídeo y audio por defecto si el navegador tiene los permisos de cámara y micrófono en la sección de JoinSession.

Una vez se llega a la sección de Session se realizan varias comprobaciones. Para ello es necesario la simulación de dos usuarios a la vez, por lo que solamente es necesario abrir otra pestaña y navegar hasta la misma ruta de la aplicación con el mismo nombre de sala al que ha accedido el primer usuario. Se espera un tiempo prudencial para dar tiempo a la página y a los servidores de OpenVidu a conectar a ambos usuarios entre sí y actualizar la vista. Una vez se ha esperado entonces se comprueba el número de subscriptores conectados en la primera pantalla, en este caso ha de ser uno. Luego el segundo usuario sale de la sesión y se vuelve a comprobar el número de usuarios conectados sin contar consigo mismo, número que obviamente ha de ser cero.

Para finalizar el segundo usuario comprueba que se pueda salir de la sesión de usuario, que vuelve a estar en la sección de Login y que efectivamente ha podido cerrar la sesión. El primer usuario también se desconecta de la llamada.

Hubiera estado interesante el comprobar que funcionase la función de compartir pantalla y que efectivamente el número de subscriptores aumentaba con ello ya que cada pantalla compartida cuenta como si fuera un nuevo usuario conectado, sin embargo, debido a que el control de compartir pantalla se realiza mediante un pop-up, Playwright no puede controlar ese tipo de interfaz. Aun con ese detalle la prueba cubre todas las funcionalidades más básicas y primordiales de la aplicación, así que constituye un buen método para comprobar el correcto funcionamiento del mismo.

4.5 Distribución y despliegue

Como ya se ha mencionado anteriormente, se ha utilizado Docker para permitir la fácil distribución de la aplicación. Para poder desplegar esta aplicación mediante Docker es necesario tres archivos presenten en la carpeta docker del repositorio. A continuación, se explicará cada archivo de manera más detallada:

- **Create_image.sh:** Es un script que ejecutará el comando de creación de imagen Docker. Cabe decir que también se podría obviar este script y simplemente ejecutar la instrucción de construcción de la imagen que posee desde la consola de comandos directamente situándose en la dirección de la carpeta docker.
- **Dockerfile:** Este archivo contiene las instrucciones necesarias para crear la imagen Docker. Para ello se encarga de generar los archivos necesarios del frontend para su ejecución y después incrustarlos en el backend, de manera que cuando se ejecute el

backend también se esté ejecutando el frontend. Esto es lo que se conoce como multistage. De esta manera, se podría dividir en tres etapas:

- Frontend: Se selecciona Node para tratar con el frontend, se establece un directorio que será /usr/share/node/app (aunque esta ruta podría ser modificada), después se copian los archivos de nuestra carpeta de frontend a esa misma ruta especificada, los archivos que necesitaremos serán el propio código de la carpeta src, el index.html y los archivos de configuración: package.json y vite.config.js. A continuación se instalan las dependencias especificadas en el package.json mediante el comando:

  "npm install"

  y finalmente se generan los archivos de desarrollo mediante el comando:

  "npm run build"

  gracias a la configuración de vite.config.js.
- Backend: Igual que todas las secciones se comienza especificando la herramienta que se usará, en este caso será Maven, por supuesto también se asigna un directorio de trabajo, siendo este /usr/share/app. Se copian los archivos importantes para la ejecución del backend desde el propio backend al directorio especificado, siendo estos el pom.xml y la carpeta src. Luego se copia los archivos generados en la anterior fase a la ruta /usr/share/app/src/main/resources/static/ para que entonces el frontend quede incrustado en el backend y la aplicación pueda acceder a todos los archivos necesarios para su ejecución. Finalmente se instalan las dependencias especificadas en el pom.xml mediante el comando:

  "mvn clean install package".
- Combinación y ejecución: Usando openjdk y estableciendo el mismo directorio que en la etapa anterior se copia el archivo openvidu-call-bak-java.jar de la carpeta target al directorio especificado. Este archivo compondrá la aplicación completa. Finalmente se especifica el puerto en el que se ejecutará, siendo el 5000 para mantener similitud con el backend ejecutado en local y finalmente se establece el comando de inicio del archivo anterior, siendo este:

  "java -jar openvidu-call-back-java.jar".

- **Docker-compose.yml:** Por supuesto, la imagen generada mediante el Dockerfile anterior puede ser ejecutada perfectamente mediante el comando:

  "docker run -it -p 5000:5000 --name openvidu-vue -d openvidu-vue"

(por ejemplo), sin embargo, no podrá realizar todas las funcionalidades de OpenVidu Call en Vue debido a que falta la conexión con el servidor de OpenVidu. Para solucionar precisamente eso entra en juego docker-compose. Con este archivo se podrá ejecutar la imagen del servidor de OpenVidu y de la web a la vez mediante el comando:
"docker-compose up".
Para ambas imágenes se establece un nombre, el nombre de la imagen, el puerto que será utilizado y la red en la que se trabajará. Este último apartado es muy importante, ya que, sin él, docker no entenderá que ambas imágenes deben trabajar de manera conjunta, cabe añadir que se tardó bastante tiempo en llegar a esta conclusión. Se añade la propiedad a la web que se reinicie si es necesario.

Finalmente se establecen las variables de entorno, para el servidor de OpenVidu será la contraseña necesaria, mientras que para la web será especificar que la url del servidor ha de leerse de la imagen generada del servidor y no directamente del puerto del contenedor. Esto último se logró añadiendo una variable de la que leerá el backend en el archivo application.properties, siendo así el único cambio que se ha realizado al backend, se sustituyó "OPENVIDU_URL: http:///localhost4443" por "OPENVIDU_URL: ${OPENVIDU_URL_PARAM:http://localhost:4443}", lo que hace esto es que si no se pasa como parámetro OPENVIDU_URL_PARAM se obtenga el valor de OPENVIDU_URL de http:///localhost4443.

CAPÍTULO 5 Conclusiones

En este capítulo final se realizará una reflexión del trabajo realizado y se comentará las posibilidades de ampliar este proyecto a futuro.

5.1 Conclusiones

Poniendo la vista en perspectiva el trabajo realizado respecto al que se planteó en primera instancia se ha cumplido todos los objetivos establecidos. Aun así, en su momento se planteó algunas funcionalidades adicionales para implementar en caso de que el proyecto avanzase a muy buen ritmo para hacerla más completa, estas funcionalidades eran la de lista de participantes y la de implementación de un chat dentro de la llamada. Desafortunadamente estas funcionalidades no pudieron ser desarrolladas debido a que se tardó más tiempo del estimado en completar las funcionalidades básicas. Concretamente se gastó más tiempo del esperado en la introducción de opentok-layout-js y en el propio servicio de llamada con el uso de Publisher, subscribers y la conexión.

Si se pudiera repetir el desarrollo sabiendo lo que se sabe actualmente, se comenzaría antes el desarrollo de la aplicación, no refiriéndose a empezar antes a nivel de fecha, sino el no gastar tanto tiempo en el estudio de Vue y todos sus componentes. Sobre todo, porque después no se utilizarían en gran medida al estar usando los componentes de la librería de Vuetify. Además se habría comenzado antes la escritura de la memoria para poderla ir intercalando con el desarrollo cuando este ya estuviera más avanzado.

Este trabajo de fin de grado ha sido una experiencia que ha afianzado gran parte de los conocimientos aprendidos a lo largo de la carrera, sobre todo los referentes a la asignatura de Desarrollo de Aplicaciones Web, cuyos apuntes fueron especialmente útiles en el apartado de Docker [38]. Además, ha supuesto un aprendizaje constante de la necesidad de seguir ciertas pautas a la hora de desarrollar cualquier clase de proyecto. Comenzando por organizar al principio las tareas que realizar y todos los pasos necesarios para llegar al producto final, ya que a simple vista da la sensación de tratar con un proyecto de tamaño colosal que sería imposible de realizar a tiempo. Pero dividiendo cada tarea en sus componentes más pequeños, y avanzando poco a poco centrándose solamente en lo que había que hacer a continuación en vez de todo lo que faltaba ha permitido llegar a concluir un desarrollo que comenzó hace aproximadamente seis meses.

A nivel de conocimientos adquiridos se añade el uso de herramientas desconocidas como el propio Vue (ya que solamente se había llegado a trabajar con Angular en la asignatura mencionada anteriormente), el uso de Playwright, GitHub actions y el afianzamiento de la creación de imágenes Docker. Sin tener en cuenta el aprendizaje de la creación de una memoria para el calibre de un trabajo de fin de grado.

5.2 Trabajos futuros

La idea detrás de este trabajo de fin de grado es dejar la puerta abierta para la expansión en un futuro de esta página web en Vue y realizar una implementación completa de todas las funcionalidades de OpenVidu Call en este proyecto. Siguiendo esa misma idea se procuró durante todo el desarrollo facilitar la legibilidad del código mediante comentarios, nombres de variables adecuados y explicativos y organización de cada apartado en diferentes secciones y métodos. Para hacerlo más accesible al público, todo el código, pruebas y el blog de Medium se encuentran escritos en inglés.

Ya que se entra en el apartado de accesibilidad, a lo largo del proyecto ha sido interesante darse cuenta de que esta aplicación es el escalón perfecto para aquellos que deseen introducirse a OpenVidu y además no sepan sobre Angular y deseen trabajar en un framework más amigable como Vue. Como la idea del propio OpenVidu es la de permitir la personalización completa para que cada usuario maneje la aplicación como desee, así cada usuario puede escoger en qué frontend trabajar, en el futuro incluso OpenVidu podría plantearse con contar con una versión de su aplicación en otros frameworks como React, así los usuarios afines a esos frameworks se verán atraídos por las facilidades para trabajar con una aplicación nueva en un entorno ya conocido por ellos. Es importante tener en cuenta que existe una gran ventaja si se diera el caso en el que cualquier usuario pudiera trabajar con el frontend que más le gustase, y es que no habría problemas en las conexiones entre aplicaciones diferentes ya que la conexión de la llamada la realiza el propio servidor de OpenVidu.

Se podría extender el proyecto para abarcar todas las funcionalidades de OpenVidu sean simples o avanzadas, como el chat, la función de grabar las videollamadas, inclusión de otros idiomas o funcionalidades más estéticas como el símbolo de detección de voz de cada usuario durante la llamada. Además de poder añadir otras nuevas, como ocultar completamente la cámara de algún usuario y que esta se “guardase” en una barra de herramientas a un lado de la pantalla. Para toda aquella funcionalidad pensada para aplicar solo a alguno de los usuarios

conectados se ha montado las opciones de usuario de manera que la escalabilidad sea sencilla. La forma en la que se ha priorizado esto es mediante una lista de opciones. Cuando se desea añadir una nueva solo debe añadirse a la lista del evento streamCreated, adaptar el diseño de la clase "circles_options right" y añadir la funcionalidad correspondiente en el método actionListOptions(option, user).

OpenVidu cuenta con un componente web para introducir directamente OpenVidu en cualquier aplicación como si fuera un botón o un título, pero siendo la propia aplicación basada en OpenVidu Call de Angular. Uno de los trabajos que se podrían realizar en el futuro podría ser un componente web basado en OpenVidu Call de Vue.

Fuera del ámbito de la ampliación o incorporación de otras formas de uso, se podría realizar un experimento para comprobar cuál de las dos versiones es más rápida a nivel de carga de los componentes en la página web, en el envío y recepción de las peticiones web y sobre todo de la velocidad de conexión a los servidores de OpenVidu.

BIBLIOGRAFÍA

[1] *OpenVidu*. (s. f.). https://openvidu.io/

[2] *¿Qué es Java? - Explicación del lenguaje de programación Java - AWS*. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/java/

[3] *Node.js*. (s. f.). Node.js. https://nodejs.org/es

[4] *Angular*. (s. f.). https://angular.io/

[5] *Vue.js - The Progressive JavaScript Framework | Vue.js*. (s. f.). https://vuejs.org/

[6] *JavaScript With Syntax For Types.* (s. f.). https://www.typescriptlang.org/

[7] *Maven – Welcome to Apache Maven*. (s. f.). https://maven.apache.org/index.html

[8] *Difference between the created and mounted events in Vue.js*. (s. f.). Stack Overflow. https://stackoverflow.com/questions/45813347/difference-between-the-created-and-mounted-events-in-vue-js

[9] *Material Design*. (s. f.). Material Design. https://m3.material.io/

[10] *Vuetify — A Vue Component Framework*. (s. f.). https://vuetifyjs.com/en/

[11] Docker (2023). *Docker*. https://www.docker.com/

[12] *Visual Studio Code - Code Editing. Redefined*. (2021, 3 noviembre). https://code.visualstudio.com/

[13] *GitHub: Let's build from here*. (s. f.). GitHub. https://github.com/

[14] *Gestiona los proyectos de tu equipo desde cualquier lugar | Trello*. (s. f.). https://trello.com/es

[15] *¿Qué es Kanban? Principales características y funciones*. (s. f.). Kanban Software for Agile Project Management. https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban

[16] *Medium – Where good ideas find you.* (s. f.). Medium. https://medium.com/

[17] *Descargar Windows 10*. (s. f.). https://www.microsoft.com/es-es/software-download/windows10

[18] Aguilar, L. (2023, 7 junio). Qué es Linux: el sistema operativo de código abierto. *ADSLZone*. https://www.adslzone.net/reportajes/software/que-es-linux/

[19] Scooley. (2022, 22 septiembre). *Introducción a Hyper-V en Windows 10*. Microsoft Learn. https://learn.microsoft.com/es-es/virtualization/hyper-v-on-windows/about/

[20] *Descarga el Firefox más rápido que nunca*. (s. f.). Mozilla. https://www.mozilla.org/es-ES/firefox/new/

[21] Google Chrome. (s. f.). *Navegador web Google Chrome*. https://www.google.com/intl/es_es/chrome/

[22] Fast and reliable end-to-end testing for modern web apps | Playwright. (s. f.). https://playwright.dev/

[23] *StarUML*. (s. f.). https://staruml.io/

[24] *UML 2 diagramming, OO software modeling, Source code engineering Tool MagicDraw UML from No Magic*. (s. f.-b). https://www.magicdraw.com/shop

[25] Aullman. (s. f.). GitHub - aullman/opentok-layout-js: Layout Manager for OpenTok - automatically lays out your Publishers and Subscribers nicely. GitHub. https://github.com/aullman/opentok-layout-js

[26] *npm: unique-names-generator*. (s. f.). npm. https://www.npmjs.com/package/unique-names-generator

[27] *npm: rxjs*. (s. f.). npm. https://www.npmjs.com/package/rxjs

[28] *npm: axios*. (s. f.). npm. https://www.npmjs.com/package/axios

[29] *Material Design Icons - Icon Library - Pictogrammers*. (s. f.). Pictogrammers. https://pictogrammers.com/library/mdi/

[30] *Vue Material*. (s. f.). Vue Material. https://www.creative-tim.com/vuematerial/

[31] *Trello*. (s. f.). https://trello.com/b/5Omwqulb/paso-a-vue

[32] Reina, V. (s. f.). *GitHub - codeurjc-students/openvidu-call-vue*. GitHub. https://github.com/codeurjc-students/openvidu-call-vue

[33] *Vanesa Reina Hernández – Medium*. (s. f.). Medium. https://medium.com/@v.reina.2019

[34] Vonage. (s. f.). *Communications APIs for Video | Vonage*. https://www.vonage.com/communications-apis/video/

[35] *OpenVidu Demos*. (s. f.-b). https://demos.openvidu.io/

[36] *Home*. (s. f.). https://www.chromium.org/chromium-projects/

[37] *WebKit*. (s. f.). WebKit. https://webkit.org/

[38] Gallego, M. (2021, febrero). *Desarrollo de Aplicaciones Web*. Universidad Rey Juan Carlos.

ANEXOS

Instrucciones de construcción del software desde el repositorio GitHub

Todo el proyecto se encuentra subido en el repositorio dedicado a los estudiantes de la URJC (https://github.com/codeurjc-students/openvidu-call-vue). Dentro del mismo repositorio se encuentran las instrucciones necesarias para desplegar la aplicación, sin embargo, a continuación, se explicará una versión de la misma en español. Cabe destacar que hay dos maneras de acceder a la aplicación:

- **Local:**
    1. En primer lugar, se ha de ejecutar la imagen docker de OpenVidu oficial mediante el comando:
       "docker run -p 4443:4443 --rm -e OPENVIDU_SECRET=MY_SECRET openvidu/openvidu-dev:2.27.0".
       Este paso se podría saltar si se modificara una variable del archivo application.properties y se sustituyera:
       "OPENVIDU_URL: ${OPENVIDU_URL_PARAM:http://localhost:4443}" por:
       "OPENVIDU_URL: https://demos.openvidu.io".
       Lo que hace esto es informarle al backend que para acceder al servidor de OpenVidu ha de hacerlo en vez de a través del puerto 4443, que lo haga accediendo directamente a la web de OpenVidu demos.
    2. A continuación, se ha de iniciar el backend. Para ello nos situaremos mediante una terminal en la carpeta correspondiente (openvidu-call-vue/openvidu-call-back-java) y a continuación instalaremos las dependencias del proyecto mediante Maven con el siguiente comando:
       "mvn install", una vez haya terminado el proceso iniciaremos la aplicación mediante el comando:
       "mvn spring-boot:run".
    3. Por supuesto, se ha de iniciar el frontend. Nos situaremos en la carpeta del correspondiente (openvidu-call-vue/openvidu-call-vuetify-frontend),

instalaremos las dependencias mediante Node y el comando “npm install” y ejecutaremos la aplicación con el comando:
“npm run dev”.

4. Oficialmente se puede comenzar a usar la aplicación en la siguiente página: http://127.0.0.1:3000/
5. El usuario y contraseña son “admin” y “MY_SECRET” respectivamente.

❖ **Docker:**

1. Primero se ha de construir la imagen docker de la aplicación, por lo que nos situamos en la carpeta correspondiente (openvidu-call-vue/docker) y ejecutamos el comando:
“docker build -f Dockerfile -t openvidu-vue ../”.
Esta imagen contendrá el frontend y el backend a la vez.
2. Para ejecutar la aplicación también es necesario el servidor, y conectar la primera imagen docker al mismo, así que para ello solo hace falta ejecutar el comando:
“docker-compose up”,
para que el archivo de docker compose levante la imagen del servidor de OpenVidu y la creada previamente.
3. La aplicación se puede usar ahora accediendo a la siguiente ruta: http://localhost:5000/. En caso de que no cargara la página se recomienda acceder al puerto 5000 mediante el link del contenedor de Docker Desktop si se ejecuta en Windows.
4. El usuario y contraseña son “admin” y “MY_SECRET” respectivamente.

Instalación del software para utilizar la aplicación

Para la correcta instalación de del software para utilizar la aplicación se necesitarán Node, JDK, Maven y Docker. Mientras que estos tres elementos funcionen será posible utilizar la aplicación tanto en local como mediante imagen docker. En caso de que se quisiera trabajar más cómodamente se podría plantear descargar Visual Studio Code para obtener un terminal integrado en el programa de edición de código, o git para descargar el propio repositorio mediante comando. A continuación, se especificará solo los programas absolutamente necesarios:

- **Node:** Los pasos de instalación más detallados se pueden encontrar en (https://www.solvetic.com/tutoriales/article/12465-instalar-node-js-en-visual-studio-code/). Sin embargo, de manera resumida se puede inferir que se debe acceder a la página oficial de Node para descargar la versión adecuada para el ordenador en el que se va a realizar este proceso (https://nodejs.org/es). Se ejecuta el archivo descargado y se siguen los pasos para la instalación, se aceptan los términos de licencia, se selecciona la ruta en donde se desea instalar Node y se seleccionan las preferencias, se recomienda instalar todas las características de Node que vienen predeterminadas para que no haya ningún problema. Una vez se termina la instalación ya se puede comenzar a usar. Para ello se abrirá una consola de comandos (CMD) y se podrá comprobar la instalación mediante el comando:

  "node -v"

  para comprobar que efectivamente se tiene la versión de Node descargada.

  Automáticamente el proceso de instalación añade la ruta de Node a la variable de entorno PATH del sistema, pero en caso de que no lo capte correctamente se recomienda acceder al enlace ya solicitado o seguir los mismos pasos que la sección "Añadir variables de entorno" del apartado de Maven que se muestra a continuación.
- **JDK**: El JDK o el Java Developer Kit es necesario para la ejecución del backend, por lo que habrá que instalarlo también, para ello se navega hasta la página oficial de Oracle para descargar la versión concreta para el ordenador en el que se va a ejecutar (https://www.oracle.com/es/java/technologies/downloads/). De abre el archivo y se inicia el proceso de instalación aceptando los términos de uso. Se puede modificar la ruta predeterminada de instalación si se desea, es importante que se marque la casilla de "Agregar el directorio al PATH" para que el sistema lo reconozca. Una vez terminado el proceso de instalación se puede comprobar que funciona correctamente abriendo una CMD e introduciendo el comando:

  "java -version".
- **Maven:** Maven hará uso de Java, por lo que es necesario que se instale Java previamente a Maven. Por supuesto se ha de acceder en primer lugar a la página oficial de descarga y seleccionar el adecuado para el ordenador en el que se realice el procedo y preferencia del usuario (https://maven.apache.org/download.cgi), ya sea compresión mediante formato .tar o .zip, ya que los archivos binarios se ignorarán. Una vez extraído los archivos y colocados en la carpeta que se deseen entonces se ha de añadir el directorio

bin a la variable de entorno PATH. A continuación, se describirá el proceso en sistema operativo Windows.

Se puede encontrar una guía más detallada en el siguiente enlace (https://dev.to/vanessa_corredor/instalar-manualmente-maven-en-windows-10-50pb#:~:text=Instalar%20Maven,-Primero%20descomprime%20el&text=Ahora%20hay%20que%20a%C3%B1adir%20a,boton%20llamado%20variables%20de%20entorno%20.), de manera corta: se ha de buscar "Editar las variables de entorno del sistema" en el buscador de aplicaciones, a continuación se seleccionará Variable de entorno y Nueva en la sección de Variables del sistema. Se añadirá una variable llamada "MAVEN_HOME" y cuya ruta será la de la carpeta que contenga los archivos descomprimidos de Maven. A continuación, se le dará a la variable PATH de esa misma sección, Editar, Nuevo y se añade %MAVEN_HOME%\bin. Se guardan los cambios y ya se puede comprobar que funciona adecuadamente abriendo una CMD e introduciendo el comando:

"mvn -v".

a. **Docker:** La instalación será diferente dependiendo de en qué sistema operativo se encuentre, sin embargo, para Windows 10 se recomienda instalar Docker Desktop (https://www.docker.com/). Una vez descargado el instalador, se ejecuta y se siguen los pasos que aparecen en pantalla. Windows puede generar problemas con Hyper-V y Docker, por ello es necesario en algunos casos el habilitarlo (https://www.profesionalreview.com/2019/01/06/habilitar-hyper-v-windows-10/). Para ello solamente hay que acceder a "Activar o desactivar las características de Windows" mediante el buscador y asegurarse de tener activado la casilla de Hyper-V. Después de ello se deberá reiniciar el equipo para que se apliquen las características. Una vez hecho esto se puede comprobar que funciona ejecutando el comando:
"docker run hello-world" en la CMD.

Una vez instalado todo se pueden seguir los pasos descritos en el anexo anterior para iniciar la página web. Cabe destacar que cada sección deberá ser ejecutada en su propia CMD (frontend, backend e imagen del servidor de OpenVidu).

Manual de uso básico

La idea detrás de este proyecto es que sea bastante intuitivo para el usuario el manejarse dentro de la aplicación. A continuación se explicará las diferentes acciones que puede realizar el usuario en cada una de las secciones.

- **Login:** En esta pantalla el usuario solo tiene la oportunidad de introducir las credenciales, de manera predeterminada el usuario será "admin" y la contraseña "MY_SECRET". Una vez se rellenan las casillas el botón de LOGIN quedará activado para que el usuario lo pulse y se compruebe si se han introducido las credenciales correctas, en caso de que no lo sean se muestra un mensaje informando del error, en caso de que sean correctas se inicia sesión y se pasa a la siguiente sección. Para visualizar la página, acudir a la Imagen 8 de este documento.
- **Join:** El usuario tiene dos opciones en esta pantalla, bien puede cerrar la sesión de usuario, borrando así todos los datos del mismo en el navegador, mediante el botón de la esquina superior derecha o introducir un nombre de sala para entrar a una llamada. Este nombre se genera aleatoriamente nada más entrar en la pantalla, sin embargo, se puede introducir el nombre que el usuario desee (siempre y cuando contenga más de seis caracteres) o darle al botón a la derecha del recuadro para generar otro nombre aleatorio. Una vez conforme con el nombre de sala se pulsa el botón JOIN para validar el nombre y pasar a la siguiente pantalla si está correcto, si no está correcto entonces se avisa mediante un mensaje para que se introduzca uno adecuado. Para visualizar la página, acudir a la Imagen 9 de este documento.
- **JoinSession:** En esta pantalla se muestra las diferentes opciones que tiene el usuario para controlar los dispositivos con los que entrará a la llamada. Por supuesto, se han de conceder permisos al navegador para usar tanto cámara o micrófono. Una vez realizado ese paso, mediante las listas de selección, se puede escoger con cuáles de ellos se desean interaccionar. También existe la opción de bloquear la cámara o silenciar el micrófono mediante los botones de sus símbolos correspondientes al lado del nombre del dispositivo usado actualmente. Ambos dispositivos son usados en la imagen de la cámara de la izquierda para que el usuario pueda ver cómo será visualizado dentro de la llamada. También se puede introducir un nombre de usuario propio, que será usado dentro de la llamada o mantener el generado automáticamente. Una vez conforme con las opciones se pulsa el botón de JOIN SESSION y el usuario será conectado a la llamada automáticamente. Para visualizar la página, acudir a la Imagen 10 de este documento.

- **Session:** Esta es la pantalla final y en la que el usuario podrá disfrutar de la llamada a la que se podrán conectar a otros usuarios siempre y cuando todos accedan a la misma sala. Dentro de la misma se puede controlar el bloqueo de la cámara propia, el silenciamiento de micrófono y compartir pantalla. Además, al pulsar el icono de los tres puntos presentes bajo la cámara de cualquier usuario entonces se desplegarán unas opciones exclusivas para ese usuario, cuyos efectos solo verá el usuario que haya activado dicha opción. Para la cámara del usuario solo está la opción de agrandar (o encoger) la pantalla mientras que la del resto de usuarios tienen esa opción y además puedes silenciar el sonido que emiten o volverlo a conectar. Además, se puede abandonar la llamada y el usuario será trasladado a la página de Join. Para visualizar la página, acudir a la Imagen 11 de este documento.

Como última anotación, es importante tener en cuenta que una vez que se introduzcan las credenciales no será necesario volverlas a introducir ya que se iniciará sesión automáticamente siempre y cuando el usuario no haya cerrado la sesión de usuario.