ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DESARROLLO DE UNA APLICACIÓN MÓVIL DE SEGUNDA PANTALLA PARA COMPLEMENTAR PRODUCTOS AUDIOVISUALES TRANSMEDIA (BACK-END)"

INFORME DE MATERIA INTEGRADORA

Previa a la obtención del Título de:

INGENIERO EN COMPUTACIÓN

SIXTO JAVIER CASTRO REDROBÁN

JORDY GERMÁN VÁSQUEZ CEPEDA

GUAYAQUIL – ECUADOR

AÑO: 2017

AGRADECIMIENTOS

Mis más sinceros agradecimientos a mis padres quienes siempre han dado todo y estado pendiente de mí en el día a día, por haberme enseñado a nunca desfallecer ni rendirme ante nada, y a ser perseverante en todo lo que propongo.

Agradezco a la Ph.D. Cristina Abad por ser mi maestra y mentora durante estos últimos años, por su valioso aporte y asesoramiento tanto en la investigación como en el proyecto de tesis, por sus consejos y apoyo incondicional.

Gracias a todas las personas que han apoyado de manera directa o indirecta a lo largo de la realización de este proyecto.

Sixto Javier Castro Redrobán

Agradezco a Dios por darme la salud y fortaleza para llegar a concluir esta valiosa meta en mi camino, y por darme sabiduría para poder tomar las mejores decisiones en esta etapa académica.

Mis más sinceros agradecimientos a mis padres quienes siempre han confiado en mí y me han alentado a cumplir mis sueños, quienes me han enseñado los valores más importantes en mi vida y así poder formarme como una persona de bien.

Agradezco a la Ph.D. Cristina Abad por ser mi maestra y tutora durante estos 2 últimos años, por su valioso apoyo y guía que me ayudó a poder culminar esta meta.

Jordy Germán Vásquez Cepeda

DEDICATORIA

El presente proyecto lo dedico primeramente a Dios por permitirme llegar a este momento tan especial de mi vida, y por darme sabiduría y fortaleza en el día a día para culminar esta etapa de estudios con éxito.

A mis padres por brindarme apoyo incondicional tanto en los buenos y malos momentos, y por guiar mi camino a lo largo de esta carrera para poder llegar a la meta.

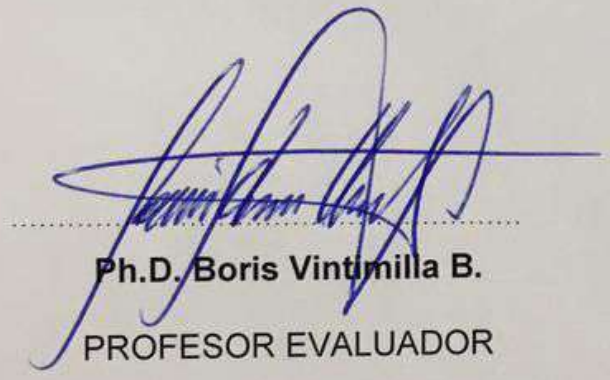
Sixto Javier Castro Redrobán

El presente proyecto lo dedico primeramente a Dios por darme la salud y fortaleza a lo largo de mi vida y permitirme guiarme por el sendero de su fe.

A mis padres por brindarme apoyo incondicional tanto en los buenos y malos momentos y el sacrificio que han hecho para hacer realidad esta meta.

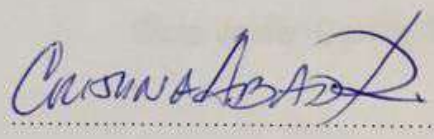
Jordy Germán Vásquez Cepeda

TRIBUNAL DE EVALUACIÓN



Ph.D. Boris Vintimilla B.

PROFESOR EVALUADOR



Ph.D. Cristina Abad R.

PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, me(nos) corresponde exclusivamente; y doy(damos) mi(nuestro) consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Sixto Javier Castro Redrobán

Jordy Germán Vásquez Cepeda

RESUMEN

Hoy en día, el uso de internet y redes sociales en los dispositivos móviles están quitando terreno, interés y fidelidad a los programas de televisión debido a que estos programas presentan un contenido escaso, monótono y poco interactivo hacia el televidente.

Con el fin de captar la atención del televidente mediante una aplicación móvil, se decidió realizar el proyecto "*Desarrollo de una aplicación móvil de segunda pantalla para complementar productos audiovisuales transmedia (back-end)*". Este proyecto se centra en el componente back-end, el cual se basa en la implementación de la arquitectura de la aplicación móvil ofreciendo los debidos servicios al componente front-end; además de garantizar escalabilidad, flexibilidad, y óptimo rendimiento a la aplicación. Los servicios ofrecidos son los siguientes: sincronización entre primera y segunda pantalla, comunicación bidireccional entre ambas pantallas, servicio de mensajería o chat, y petición de recursos adicionales transmedia.

Se resalta que este proyecto es multidisciplinario donde participan integrantes del Club MAEC de la facultad de Mecánica como actores de los videos, estudiantes de EDCOM encargados de la producción de contenidos audiovisuales, y estudiantes de FIEC encargados tanto de los componentes front-end y back-end de la aplicación móvil de segunda pantalla.

El proyecto tiene como objetivo principal crear una experiencia de usuario enriquecedora y envolvente al televidente, captando su atención y potenciando su interacción con el contenido visualizado.

ÍNDICE GENERAL

CAPÍTULO 1

1 ANÁLISIS DEL PROBLEMA

El *transmedia* es una forma de poder narrar una misma historia a través de múltiples plataformas: web, cine, tv, teatro, libros, cómics, videojuegos, apps móviles, redes sociales, etc. En el transmedia storytelling no se intenta repetir la misma historia en una película, en un programa de televisión, en una novela o en un videojuego; sino más bien se utiliza cada medio con el fin de contar una historia completa en diferentes formatos: libro, post, spot, película, y mediante esta combinación lograr una enriquecedora experiencia [1].

La televisión seguirá existiendo debido a su capacidad de producción y realización de programas. Conforme la tecnología va avanzando, los que dirigen las cadenas de televisión con el fin de poder captar la fidelidad y atención de los televidentes, los invitan a participar, y a ser parte del mismo gracias al uso de la *segunda pantalla*. Lo cual en un principio fue mediante simples tuits sobre preferencias acerca de concursantes en un reality show [2].

La segunda pantalla es cualquier dispositivo digital que se utiliza junto a la primera pantalla y permite al usuario interactuar con el contenido de la primera pantalla en tiempo real recibiendo toda aquella información complementaria que se desea, tales como: fechas, lugares, acontecimientos o personajes que aparecen en una película o serie; y ahora incluso es capaz de hacer que el usuario interactué con otros espectadores.

Figura 1.1: Plataformas que aportan al contenido transmedia [3].

Figura 1.2: Aplicación de segunda pantalla sincronizada con el televisor (primera pantalla) [4].

1.1. Descripción del problema.

Hoy en día, cada vez son más las personas que navegan en la web, consultan los correos o visitan las redes sociales a través de sus dispositivos móviles mientras ven la televisión. Dado este motivo, la televisión de cierto modo ha estado perdiendo terreno, interés y fidelidad por parte de los usuarios.

En la actualidad, lo complejo ha sido poder retener, llamar la atención al telespectador sin que éste se desvincule del programa que está viendo. La amenaza del internet y los dispositivos móviles podrían acabar con la televisión si no se los utilizan de manera adecuada con el fin de captar el interés del público social, mas no utilizar la segunda pantalla como una distracción o amenaza para el contenido televisivo.

Este proyecto multidisciplinario consta de cuatro componentes, los mismos que sirven para poder atacar esta problemática. Se tiene el componente de diseño y línea gráfica de las pantallas de las aplicaciones tanto móvil como web, además del logo y nombre del mismo, así como el concepto que se comunica. Otro componente es el de la creación de los contenidos audiovisuales que forman parte de la aplicación, ya sean videos, imágenes, guiones, etc. Por último, se tiene los componentes de desarrollo de la aplicación, los cuales son: front-end y back-end.

El back-end se encarga de la arquitectura de la aplicación, así como de ofrecer los debidos servicios para poder ser consumidos del lado del front-end. Además, deberá garantizar flexibilidad y escalabilidad para la aplicación.

El front-end se encarga de la aplicación móvil del proyecto y la interacción con el usuario final.

Nuestro proyecto se centra en todo lo referente al componente back-end, es decir, a la implementación de la arquitectura de una aplicación de segunda pantalla. Este componente también presenta una página web para la administración de la plataforma, la cual sirve para definir el contenido de las escenas, crear, editar o eliminar las mismas. Por último, el back-end involucra la definición de la estructura para el paso de mensajes de la primera con la segunda pantalla.

1.1.1 Causas

Dentro de las posibles causas se puede destacar las siguientes:

- Contenido ofrecido por la televisión resulta ser a veces escaso en su contexto.
- Debido al contenido monótono, o falta de creatividad mostrado por los programas de televisión.
- La falta de interacción de los programas hacia los usuarios e interacción entre ellos mismos.

1.1.2 Efectos

Los posibles efectos a destacar son los siguientes:

- Una caída en el rating del programa y caída en los ingresos por parte de las televisoras.
- Las televisoras deben buscar nuevas formas de poder enganchar al televidente creando contenido novedoso y entretenido.
- El telespectador decide buscar contenido adicional en la Web.
- El televidente deja de seguir el programa por las causas mencionadas, y se dedica a realizar cualquier otra actividad perdiendo el vínculo con el programa que estaba viendo.

1.2. Objetivos

1.2.1 Objetivos finales

- Crear experiencia de usuario envolvente y entretenida para fidelizar la atención al contenido transmedia que se esté observando.
- Poder trabajar colaborativamente con integrantes de la misma facultad (componente front-end) y también con aquellos de

EDCOM (producción contenido audiovisual) para llevar a cabo la realización de este proyecto en menor tiempo posible.

- Desarrollar una arquitectura para el back-end que sea escalable, flexible y que posea un óptimo rendimiento además de ser segura de usar.

1.2.2 Objetivos específicos

- Implementación de servicios del lado del servidor para peticiones de recursos que el cliente realice al momento de ver un video.
- Poder sincronizar la primera pantalla con la segunda (aplicación) mediante un método eficiente y rápido.
- Realizar sala de chats entre los distintos clientes que estén observando el video.

1.3. Trabajos relacionados

Wii U es una consola de Nintendo que tiene una segunda pantalla integrada en el controlador. Eso hace posible que los juegos muestren contenido adicional con el que el usuario pueda interactuar a través del controlador.

Figura 1.3: Controlador de la consola Wii U como segunda pantalla [5].

Chrome Maze es una aplicación web que utiliza un dispositivo con un navegador como primera pantalla y se acopla con una segunda pantalla que actúa como controlador para la primera pantalla. La idea de la aplicación es construir un laberinto desde cualquier sitio web y dejar que el usuario guía una pelota a través del laberinto utilizando una segunda pantalla como controlador [6].

Figura 1.4: Google Chrome World Wide Maze [7].

Walkers Kill Count es una aplicación móvil de segunda pantalla de la serie *The Walking Dead*, la cual se sincroniza con el televisor mediante la detección de marca de agua digital (línea de código incrustada en el video). La aplicación invita al usuario estimar el número de zombies a morir; y conforme va transcurriendo el episodio, el contador de la *app* va incrementando cada vez que un zombie muere [8].

Figura 1.5: Aplicación de segunda pantalla de la serie The Walking Dead [9].

1.4. Propuesta de la solución

Tal como se mencionó en la sección de Definición del problema, nuestro proyecto se centra en la implementación de la solución del componente back-end.

La solución para el componente back-end consta de dos módulos: módulo de configuración de contenido y sincronización entre primera y segunda pantalla, y el módulo de sala de chat.

La descripción de la metodología, procedimiento, herramientas utilizadas, y la arquitectura para la solución respectiva son detallados en el capítulo 2.

En la Figura 1.6, se muestra la interacción entre los diferentes elementos y componentes que componen la arquitectura para la segunda pantalla. Se muestra para el lado del servidor los dos módulos ya mencionados antes, los mismos que configuran el dispositivo, proporcionan contenido, y permiten crear una sala de chats para los usuarios de la segunda pantalla.

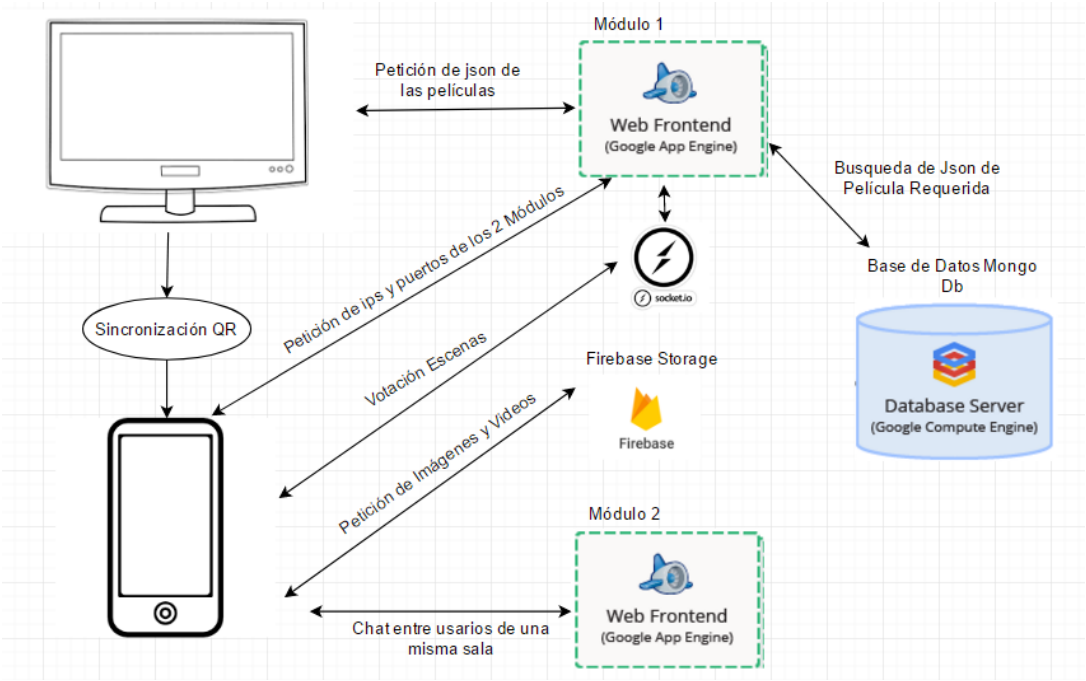


Figura 1.6: Segunda Pantalla - Caso de uso

CAPÍTULO 2

2. ANÁLISIS DE LA PROPUESTA.

En el capítulo 2, se detalla más a fondo la propuesta del proyecto, incluyendo la metodología del mismo, los módulos a desarrollarse con sus respectivas herramientas, arquitectura e implementación de la solución. También, se incluye pruebas para *streaming* de video en el servidor del primer módulo con Node.js, y pruebas con dos tipos de sincronización entre la primera y segunda pantalla.

2.1 Metodología del proyecto

La metodología usada en el proyecto fue Scrum debido a las siguientes razones:

- Cada sprint se llevó a cabo en bloques temporales cortos y fijos de dos a cuatro semanas.
- Nos permitió flexibilidad y adaptabilidad en los cambios, esto resultó importante debido a que nuestro proyecto es multidisciplinario y al momento de integrarlo como uno solo con los demás grupos los aspectos anteriores fueron fundamentales.
- Reuniones cortas y concisas con el equipo de trabajo.
- Nos permitió mostrar resultados al cliente al final de cada sprint para poder tomar medidas de corrección necesarias.

2.2 Descripción de los módulos

Este proyecto está dividido en dos módulos: módulo de configuración de contenido y sincronización entre primera y segunda pantalla, y módulo de sala de chat.

El primer módulo se centra en la creación y edición de los elementos que posee una película, además de la configuración de contenido donde el usuario administrador puede subir, editar o eliminar contenido tales como imágenes,

videos y URLs. También involucra los servicios para la sincronización entre la primera y segunda pantalla, y la atención de llamadas a los diferentes recursos que tienen disponibles las escenas en la primera pantalla.
El segundo módulo se enfoca en la sala de chat donde los usuarios sincronizados a una misma sala de película pueden enviar mensajes de texto desde la aplicación de segunda pantalla.

2.2.1 Módulo de Configuración de contenido y Sincronización entre primera y segunda pantalla

El primer módulo se basa en la fase de configuración del dispositivo de segunda pantalla para poder configurar la base del mismo con todos los eventos o contenidos que se consumen durante la transmisión de una película. Este módulo también incluye el método de sincronización entre primera y segunda pantalla. Finalmente, como integración a este módulo, se tiene a disposición una plataforma para alojar todo el contenido transmedia adicional (imágenes o videos adicionales) y el contenido live (escenas de una película) que se suben desde la página del usuario administrador.

2.2.2 Módulo de Sala de Chat

Con el fin de que el usuario mediante la aplicación móvil se sienta más involucrado con el contenido visualizado, se decidió realizar este módulo como un adicional al módulo anterior para que el usuario pueda interactuar con los demás a través de mensajes referentes a la película que se está transmitiendo.

Este módulo está dirigido específicamente para crear una sala de chat entre los distintos usuarios pertenecientes a una sala de película. La implementación de este módulo está ideada para que éste soporte

multicanal y multiusuario; es decir cada sala de película consta con su respectiva sala de chat entre los distintos usuarios conectados.

2.3 Módulo de Configuración de contenido y Sincronización entre primera y segunda pantalla

2.3.1 Metodología

Este módulo contiene la página web de administración de contenido en donde se configura los elementos que contiene una película, ya sean contenido adicional como imágenes o videos, así como también la secuencia de escenas de acuerdo al flujo de historias que se desee poner en una película. Además, permite subir los videos e imágenes que forman parte de una película, así como administrarlos mediante una tabla de resumen de las mismas.

En lo que respecta a la comunicación en tiempo real, la aplicación móvil de segunda pantalla primero realiza una petición a la siguiente URL fija "http://config-admin.appspot.com/ips"; la cual le proporciona un archivo *json* que contiene las IPs y puertos (Figura 2.1) que sirven para que la aplicación pueda conectarse a los servidores de los dos módulos por medio de socket.io. De esta manera, poder consumir los distintos servicios ofrecidos: comunicación bidireccional con la primera pantalla y sala de chat.

Figura 2.1: Estructura *json* de IPs y puertos de servidores.

El método para la sincronización entre primera y segunda pantalla se escogió después de haber realizado pruebas con varios clientes a la vez, con el fin de saber qué método era más rápido y cuál ofrecía una tasa de error aceptable al momento de la sincronización.

Herramientas y tecnologías utilizadas

Dentro de las herramientas que se han utilizado para el respectivo desarrollo de este módulo, tenemos las siguientes con su respectiva justificación:

- *Node.js:* Concebido como un entorno de ejecución de JavaScript orientado a eventos asíncronos. Está diseñado para construir aplicaciones escalables y flexibles del lado del servidor, y contiene una gran variedad de librerías de terceros que enriquecen su entorno [10]. Este framework fue útil del lado del servidor, para crear nuestra arquitectura con ambiente javascript.
- *Express.js*: Es el framework más conocido de Node.js, brinda robustez, rapidez y flexibilidad [11]. Fue utilizado para gestionar las peticiones de una manera sencilla tanto para la primera pantalla y segunda pantalla.
- *Bootstrap:* Es el framework más popular para desarrollo de aplicaciones web y móviles con páginas responsive [12]. Herramienta empleada para el diseño de la página web del administrador en la cual se sube los archivos *json* de configuración, y también empleada en la página de inicio de sesión del administrador.
- *MongoDB*: Base de datos no relacional ideal para el manejo de estructura *json*, la cual es usada para pasar información entre primera y segunda pantalla. Se utilizó para guardar las

credenciales del usuario administrador y las estructuras *json* de configuración.

- *Firebase Storage:* Con respecto al almacenamiento de imágenes y videos, se utilizó el servicio de Firebase *Storage* vinculado con la plataforma de *Google Cloud* para alojar todo el contenido transmedia adicional y contenido live.
- *HTML5*: Esta última versión de HTML permitió manejar la reproducción de videos sin necesidad de tener el plugin *Adobe Flash Player* instalado del lado del cliente en la primera pantalla.
- *JSON Schema Based Editor:* Esta librería toma una estructura *json* para generar un formulario HTML; una vez llenado el formulario, la librería retorna el archivo *json* con los valores ingresados en el formulario [13]. Librería utilizada para generar una estructura *json* cada vez que se edite el formulario de contenido de una película en la página web del administrador.
- *JSON Editor:* Esta librería tiene como función editar o modificar la estructura (parámetros o atributos) de cualquier estructura *json* de una manera fácil sin correr riesgo de dañar la estructura por error [14]. En este caso, se la utilizó para las estructuras *json* de IPs y de contenido de una película.
- *Crypto-js/aes:* Es un módulo de Node.js en el lado del servidor y una librería de javascript en el lado del cliente, y soporta uno de los algoritmos de encriptación simétrica más seguros y utilizados hoy en día. Opera con bloques de 128 bits y claves de diferentes tamaños: 128, 192, y 256 bits [15]. Se lo aplicó para la encriptación del *salt,* el cual es un string aleatorio que sirvió para potenciar la función hash *SHA-512* al momento de enviar la contraseña del administrador al servidor. Esto será descrito más adelante.

- *SHA-512:* Es una de las funciones hash más seguras de la familia SHA-2, provee un hash con extensión de 512 bits, y su proceso consta de 80 rondas (pasos) de operaciones [16]. Función hash aplicada para enviar la contraseña del usuario del lado del cliente al servidor de manera segura.
- *MD5*: Función hash especialmente utilizada para validación de claves de acceso a sistemas, ésta devuelve como resultado un hash de 128 bits [16]. Esta función hash se aplicó a la contraseña ingresada por el administrador desde el lado del cliente, y esta misma se encuentra guardada en la base de datos en ese formato de hash del lado del servidor.
- *Socket.io:* Librería que permite la comunicación bidireccional en tiempo real basada en eventos, ideal si se trabaja con Node.js [17]. Se la usó para la sincronización y comunicación entre la primera y segunda pantalla.
- *Cookie Parser:* Es un módulo de Node.js que sirve para configurar y manejar sesiones dentro del servidor [18]. Este módulo fue fundamental en el manejo de sesiones del usuario administrador en la plataforma de web de administración de contenidos.
- *qrcode.js:* Librería de javascript que genera un código QR en base a un valor o texto otorgado [19]. Para este módulo, esta librería sirvió para generar el código QR a partir del ID de la sala de la película.

2.3.2 Implementación

En esta sección, se describe más a fondo la implementación de este módulo, con subsecciones detallando lo siguiente: estructura de los archivos *json* para configuración y almacenamiento de contenido, acceso al usuario administrador para control de contenido, listado de películas en

la primera pantalla, *streaming* de video en la primera pantalla, e interacción entre primera y segunda pantalla.

a. Estructura de los archivos *json* para configuración y almacenamiento de contenido.

En este apartado, se describe la estructura de los *json* para configuración de contenido, y también acerca del almacenamiento de contenido transmedia adicional y contenido live.

Estructura de los archivos *json* para configuración de contenido

La estructura del *json* de configuración utilizada para subir contenido a la plataforma, se encuentra dividida en dos categorías principales: contenido transmedia adicional y contenido live. Estos son descritos a continuación:

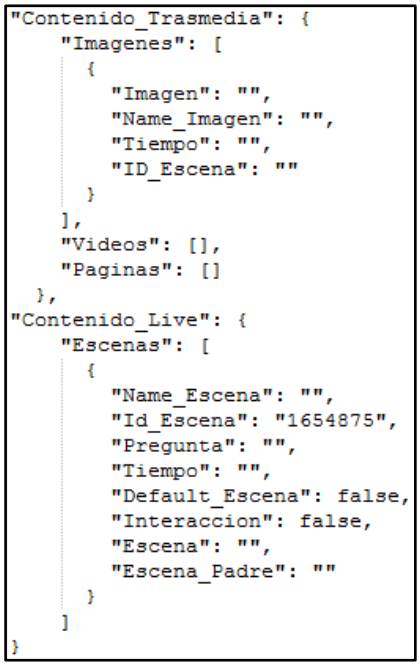
Contenido Transmedia: Por medio de este contenido, el usuario de la aplicación móvil puede visualizar e interactuar con la primera pantalla a través de contenidos adicionales a las escenas de una película tales como: imágenes, videos o páginas externas. Cada uno de estos contenidos contiene su URL de consulta y el minuto en el cual deben aparecer en la aplicación móvil.

Contenido Live: A través de este contenido, el usuario puede interactuar desde la segunda pantalla viendo reflejado sus acciones o decisiones en la primera pantalla. El contenido live de una película está compuesto por un conjunto de escenas, y cada una posee los siguientes campos: ID, título, URL de la escena, pregunta de interacción, tiempo en aparecer la interacción de escenas siguientes, ID de la escena padre, etc. Estos campos se encuentran más detallados en la descripción de la Figura 2.6.

Cabe recalcar que se utilizó compresión en base64 tanto para el *json* de IPs como el *json* de elementos de una película, los mismos que se encuentran guardados en la base de datos MongoDB. La compresión se la realizó por motivos de seguridad ya que al ser capturados los *json* por

un ataque a la base de datos podría revelarse información confidencial, así como URLs privadas de nuestro host.

En la Figura 2.2, se observa un ejemplo de cómo se encuentra definida la estructura del *json* para subir contenido referente a una película con sus respectivos atributos.



```
"Contenido_Trasmedia": {
    "Imagenes": [
      {
        "Imagen": "",
        "Name_Imagen": "",
        "Tiempo": "",
        "ID_Escena": ""
      }
    ],
    "Videos": [],
    "Paginas": []
  },
"Contenido_Live": {
    "Escenas": [
      {
        "Name_Escena": "",
        "Id_Escena": "1654875",
        "Pregunta": "",
        "Tiempo": "",
        "Default_Escena": false,
        "Interaccion": false,
        "Escena": "",
        "Escena_Padre": ""
      }
    ]
}
```

Figura 2.2: Estructura de archivos *json* para subir contenido a la plataforma.

Para facilitar al administrador la subida de contenido y edición de la estructura del *json* de contenido, se desarrollaron las siguientes vistas o páginas tal como se muestran a continuación:

Figura 2.3: **Página modo edición de *json* de Configuración de contenido.**

En la Figura 2.3, se puede observar la página que le permite al administrador poder definir la estructura del *json* de configuración de contenido de una película con sus distintos campos. Esta página le otorga al administrador la opción de poder modificar o agregar campos a la estructura *json* de una película, y de esta manera poder definir qué contenidos pueden ser adjuntados a una película.

Para evitar que el administrador dañe la estructura del *json* de configuración al momento de editarlo, se incluyó la librería *JSON Editor* para mostrar este archivo en modo de árbol. De esta manera, solo tendrá que modificar el nombre o valor de las variables, y agregar más variables al *json* en caso de ser necesario.

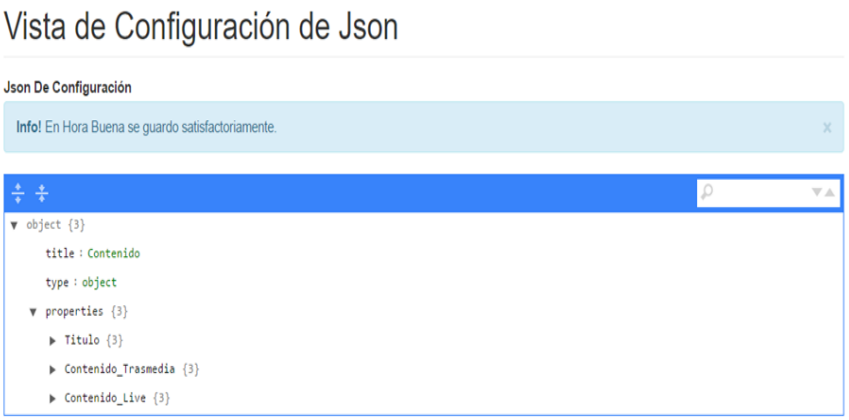


Figura 2.4: Página modo vista de *json* de Configuración de contenido.

Una vez modificada la estructura del *json* de configuración de contenido en modo edición, ésta se actualiza en la base datos MongoDB en formato base64, y se muestra inmediatamente el mismo *json* en modo de árbol con los campos creados o modificados en la página modo vista tal como se observa en la Figura 2.4.

A continuación, en la Figura 2.5, se aprecia la página que le permite al usuario administrador poder crear, editar o modificar una película. Cada película está compuesta por los siguientes campos: título, descripción, foto de portada, contenido transmedia y contenido live.

Figura 2.5: Página para subir contenido en el servidor.

En el extremo derecho de la Figura 2.5, se observa el *json* de la película, el mismo que se actualiza conforme se edita el formulario.

Luego de haber llenado los distintos campos de la película, el administrador da clic en *Subir*, y automáticamente el *json* de salida (*json* generado) con los datos llenados en el formulario se guarda en la base MongoDB en formato base64 mediante la librería *lzstring*.

Con respecto al almacenamiento de contenido transmedia adicional y contenido live, se lo detalla de mejor manera en el apartado *Almacenamiento de contenido audiovisual*.

Dentro del mismo formulario, en la sección *Contenido Live*, se le muestra al administrador la opción de poder subir escenas de videos a la película que se está creando o editando tal como se percibe en la figura de abajo.

Figura 2.6: Estructura de *json* de escenas.

En la Figura 2.6, se observa la estructura de las escenas de video de una película. Cada escena está compuesta de un nombre, ID escena, pregunta, tiempo, default escena, interacción (indica si la escena tiene escenas siguientes), escena (URL del video subido en Firebase), y escena padre.

La variable *tiempo* representa los últimos segundos antes que la escena acabe, y en los cuales se muestran las opciones (siguientes escenas) en la segunda pantalla.

La variable *pregunta* representa una imagen conteniendo la pregunta de interacción que se visualiza en la segunda pantalla por el usuario al momento de escoger la escena siguiente.

Una escena puede depender de una *escena padre* tal como se muestra en la Figura 2.6. Las escenas *indexNO* y *indexSI* tienen como padre a la escena *Inicio*. Lo cual quiere decir que cuando esta última esté por finalizar, en la aplicación móvil se le muestra al usuario las dos escenas para que pueda elegir o votar por una de ellas; y de esta manera el usuario pueda definir el rumbo de la historia de la película.

Se realizó el mecanismo para recibir las votaciones de las escenas siguientes desde la segunda pantalla. La escena con mayor votación es

la que se reproduce en la primera pantalla, en caso de empate se escoge la última votación en orden de llegada.

Además, se implementó un método para poder reproducir escenas por defecto en caso de que no ocurra ninguna votación por parte de la segunda pantalla. Variable *default* indica que dicha escena puede ser reproducida por defecto.

Por último, en la Figura 2.7, se aprecia la página para poder administrar los contenidos subidos de cada película por el administrador de la plataforma web permitiéndole crear, editar, ver o eliminar los contenidos.

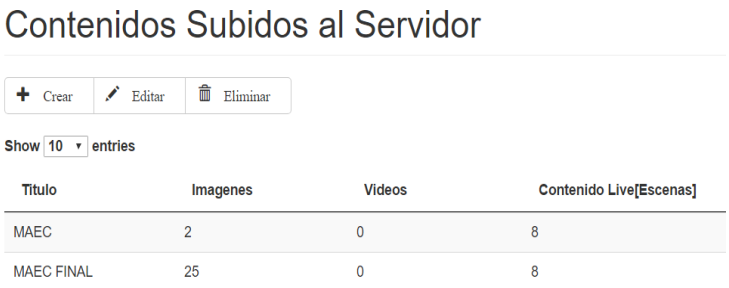


Figura 2.7: Página para administrar los contenidos subidos.

Almacenamiento de contenido audiovisual

Con respecto al almacenamiento de imágenes y videos, se utilizó *Firebase Storage* (Figura 2.8) para almacenar tanto contenido transmedia como contenido live.

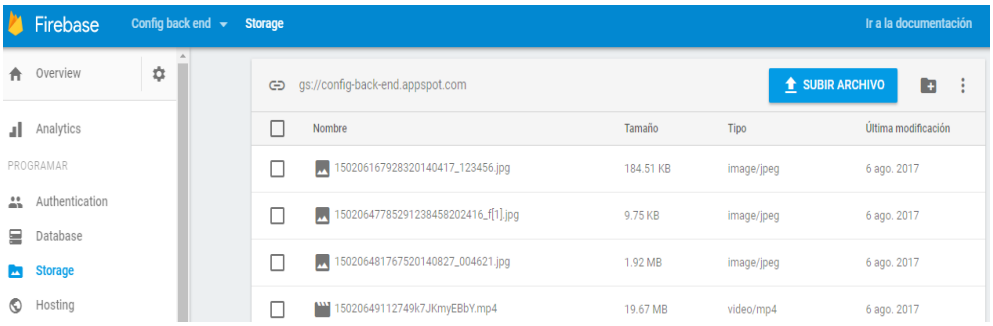


Figura 2.8: Firebase como almacenamiento de contenido transmedia.

Cabe destacar que cuando el administrador hace *submit* al momento de crear una película, primero se almacenan los videos e imágenes de la película en *Firebase Storage*, y esta plataforma le asigna una URL a cada contenido. Finalmente, esa URL es la que se guarda en el *json* de la película con los demás parámetros llenados en el formulario.

b. Acceso al usuario administrador para control de contenido.

Para esta sección, se creó una pantalla de inicio de sesión para que el usuario administrador tenga acceso para poder administrar los contenidos subidos al servidor tal como se observa en la Figura 2.9.

A continuación, la pantalla de inicio de sesión del usuario administrador:

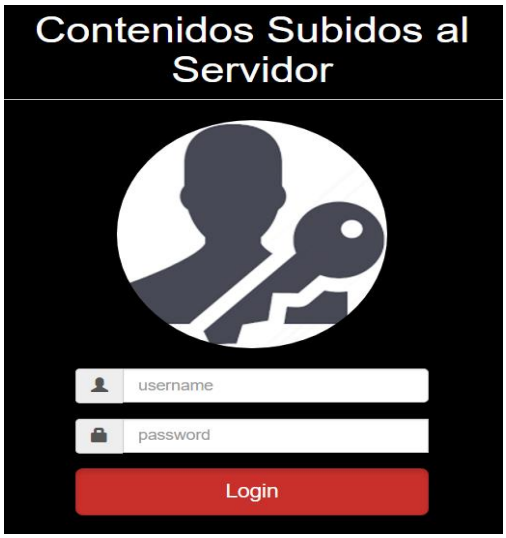


Figura 2.9: Pantalla de inicio de sesión para el administrador.

El administrador ingresa su nombre de usuario y contraseña, donde esta última se encuentra en la base de datos guardada en forma de hash gracias a la función de hash MD5.

El usuario administrador al enviar o hacer *submit* de sus datos, la misma aplicación del lado del cliente aplica una función hash MD5 a la contraseña enviada por el usuario. Luego viene la parte importante que es la

transmisión de dicha clave al servidor en modo seguro, para lo cual se utilizó la función de hash SHA-2 más un *salt* (string aleatorio de longitud de 16 caracteres) con el fin de que la contraseña (hash resultante) sea enviada de forma segura y no sea la misma cada vez que se la envíe al servidor.

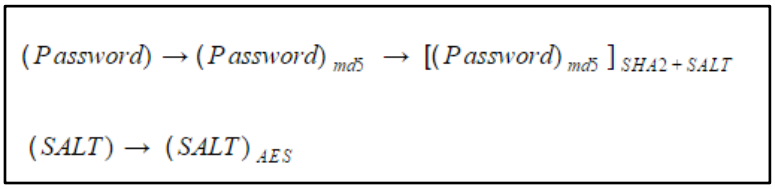


Figura 2.10: Parámetros a enviar del cliente al servidor.

Tal como se observa en la Figura 2.10, desde el lado del cliente a la contraseña otorgada por el administrador se le aplica la función hash MD5, luego SHA-2+salt, y finalmente la contraseña resultante es enviada al servidor. También se le envía al servidor un string aleatorio *salt*, el cual sirve para que el servidor pueda obtener la misma contraseña recibida desde el lado del cliente luego de aplicar la misma función hash SHA-2+salt a la contraseña guardada en la base (contraseña en formato MD5), y así poder compararlas.

Cabe destacar que, si no se envía el *salt*, al aplicar la función de hash SHA-2 a la contraseña guardada en la base; nunca se volverá a obtener el mismo resultado que la contraseña resultante enviada del lado del cliente debido a que el *salt* es único en cada inicio de sesión y no puede ser generado más de 1 vez. Motivo por el cual, el *salt* debe ser enviado al servidor; y con el fin de enviarlo de manera segura, se lo envía encriptado con AES.

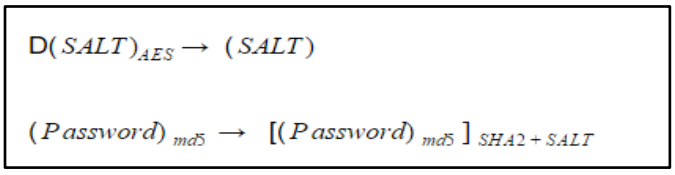


Figura 2.11: Parámetros usados en el lado del Servidor.

En el lado del servidor (Figura 2.11), se consulta en la base la contraseña del usuario que se encuentra en formato MD5, y el *salt* cifrado recibido desde el lado del cliente es desencriptado con AES.

Una vez obtenida la contraseña de la base, se le aplica la función hash SHA-2 más el *salt* descifrado. Luego, esta contraseña resultante del lado del servidor se la compara con la contraseña recibida del lado del cliente y si son iguales; automáticamente el usuario administrador obtiene acceso a la página de administración de contenido tal como se muestra en la Figura 2.12.

Figura 2.12: Comparación de las contraseñas resultantes tanto del servidor como del cliente.

Se decidió utilizar encriptación simétrica en la contraseña del usuario debido a que ésta utiliza menos recursos (menos pesada, utiliza menos ciclos de CPU) que la encriptación asimétrica. Esta encriptación es útil y fácil de usar para el intercambio de mensajes entre emisor y receptor utilizando la misma clave en ambos lados para encriptar y desencriptar el *salt*.

Con respecto a la clave utilizada para cifrar y descifrar el *salt*, se utilizó una frase de acceso en vez de contraseña (palabra de acceso) debido a que las frases contribuyen con los requisitos de complejidad con el fin de volver difícil descifrar la clave a fuerza bruta. Las características que las frases de acceso deben cumplir para ser seguras son las siguientes:

- Conjunto de palabras para crear una frase.
- Utilizar mayúsculas al menos una letra en cada palabra.
- Incluir signos de puntuación o exclamación.
- Dejar espacios entre palabras.
- Reemplazar letras por números o símbolos.

Listado de características obtenido de [20].

Ejemplo:

```
var salt_encrypt = CryptoJS.AES.encrypt(salt, "My Secret Passphrase");
var salt_decrypt = CryptoJS.AES.decrypt(salt_encrypt.toString(), 'My Secret Passphrase');
```

Figura 2.13: Funciones para encriptar y desencriptar el *salt* utilizando AES.

En la Figura 2.13, la función *encrypt* es utilizada para encriptar el *salt* en el lado del cliente, y luego es enviado al servidor donde este mismo mediante la frase de acceso puede desencriptar (función *decrypt*) el *salt* cifrado.

A continuación, se describen los escenarios de cierre de sesión y sesión expirada de la plataforma web para administrar contenidos.

Figura 2.14: Cierre de sesión.

En la Figura 2.14, se observa que una vez que el administrador haya cerrado sesión se lo redirige a la pantalla de inicio de sesión. Internamente la variable de sesión asignada al usuario es eliminada o expirada gracias a la función otorgada por la librería *cookie-parser,* la cual se llama *clearCookie*.

Figura 2.15: Expiración de sesión.

En la Figura 2.15, se muestra que una vez que la variable de sesión (cookie) haya expirado debido a la duración asignada a la misma; el usuario administrador es redirigido a la pantalla de inicio de sesión donde se muestra el mensaje: *"La sesión ha expirado. Por favor, ingrese de nuevo"*.

Cabe recalcar que por cada petición que el usuario realice en la página de administración de contenido, el tiempo de duración asignado a la variable de sesión se restablece. El tiempo sólo llega a 0 cuando el usuario presenta inactividad; en otras palabras, no haya realizado ninguna petición.

c. Listado de películas en la primera pantalla.

En la aplicación de primera pantalla, se tiene la página *Lista de Películas* donde se exhibe un listado de todas las películas existentes. Este listado se genera a partir de un archivo *json*, el cual contiene la lista de las películas creadas con sus respectivos atributos: ID, título y un *json* embebido tal como se aprecia en la Figura 2.16.

Cabe recalcar que el *json* embebido contiene información comprimida en base64, y este *json* es el que se genera cuando se llena el formulario para subir contenido de una película.

Figura 2.16: Estructura *json* de las películas.

A partir de las estructuras *json* embebidas, se obtiene información para mostrar en la página *Lista de Películas* (Figura 2.17) tales como: foto de portada, título y descripción de la película.

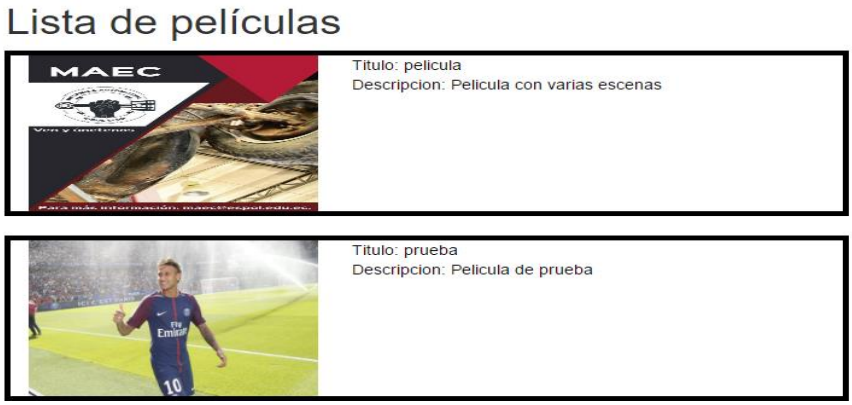


Figura 2.17: Lista de películas en la aplicación de primera pantalla.

Al momento del dar clic en una de las películas mostradas en la Figura 2.17, el usuario es redireccionado a la sala o *room* de dicha película. Cada vez que se entra a una sala, se genera un código QR conteniendo el ID de

la sala que sirve para que la aplicación de segunda pantalla pueda sincronizarse con la primera pantalla.

d. Medios de sincronización entre primera y segunda pantalla.

Esta sección abarca los medios de sincronización que se utilizaron para decidir cuál era el más eficiente y efectivo al momento de unir a varios usuarios a una misma sala de película.

Las pruebas de sincronización se realizaron con los siguientes métodos:

Sincronización por ultrasonido

En la Figura 2.18, se muestra la interacción entre los diferentes elementos y componentes que intervienen durante la sincronización entre los dispositivos de primera y segunda pantalla a través del método de ultrasonido.

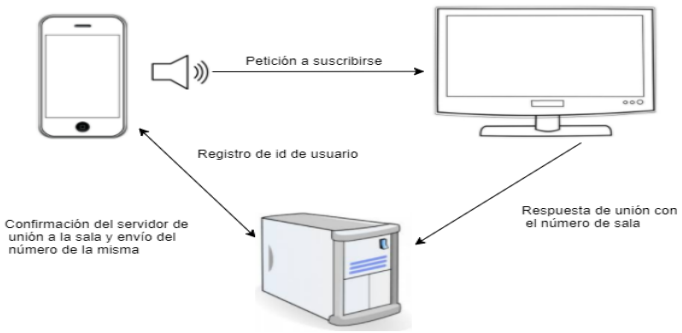


Figura 2.18: Caso de uso - Sincronización vía ultrasonido

El dispositivo de segunda pantalla emite un pulso ultrasónico conteniendo un ID para poder unirse a la sala de video y en paralelo registra este ID en el servidor como un usuario a la espera de conectarse. Del lado de la primera pantalla, ésta recibe la petición y la envía al servidor para que éste compruebe que existe ese ID de usuario y lo pueda unir a la sala. Mediante un mensaje directo al cliente con el ID del *room* antes mencionado, se confirma su suscripción.

Para este mecanismo de sincronización por ultrasonido, se lo realizó de dos formas: la primera mediante la librería sonic.net [21] y la otra por medio

de quiet.js [22]. En ambos casos, los resultados no eran los esperados debido a que la distancia máxima a la cual el dispositivo móvil debía estar para sincronizarse era de 70 a 80 cm; lo cual dificultaba al usuario además de los factores externos que influían al momento de sincronizarse tal como el ruido externo del medio ambiente, y cada usuario tenía que sincronizarse en serie, es decir uno a uno.

Sincronización por código QR

En la Figura 2.19, se muestra el código QR, el cual contiene el ID de la sala del video al cual el usuario de segunda pantalla se sincroniza. A través de un analizador de QR integrado en la aplicación de segunda pantalla, se recibe ese ID; y por medio de la librería socket.io, el usuario puede unirse a dicha sala. A partir de ese momento, el usuario es capaz de interactuar con el video de la primera pantalla.

Figura 2.19: Código QR en la primera pantalla.

También se muestra el número de usuarios conectados en la parte superior del código QR (cuadro rojo), el cual es un contador que aumenta o disminuye según la audiencia conectada en ese instante.

La ventaja de poder utilizar código QR es que este mecanismo es de fácil uso al transmitir el ID de la sala de película para la sincronización entre primera y segunda pantalla; la cual se la realiza en menor tiempo posible

(prácticamente de manera inmediata) comparado al método de ultrasonido debido a que éste último a veces requiere de varios intentos porque depende de factores tal como el ruido del ambiente. Otro factor a tomar en cuenta es la distancia de sincronización entre ambas pantallas por medio de QR, se realizaron pruebas de manera exitosa hasta de 10 metros utilizando la pantalla de la sala de cine del auditorio de la biblioteca central de ESPOL como primera pantalla.

Un último punto a resaltar es que mediante código QR, se puede sincronizar más de un usuario a la vez; en cambio por ultrasonido es necesario esperar que un usuario finalice la sincronización para que los demás usuarios puedan realizar la misma acción.

e. Streaming de video en la primera pantalla.

En esta sección, se muestran dos formas en las cuales se implementó el *streaming* de video en el servidor del primer módulo. Estas pruebas se las realizaron tanto de manera local como en la nube de Google.

Streaming de video desde *Google Cloud*

En la Figura 2.20, se observa el *streaming* de video realizado desde la plataforma de *Google Cloud*. La URL de la película es la siguiente: config-admin.appspot.com/escenas/9055008.

Para esta prueba específica, se tuvo inconvenientes al reproducir videos desde la nube debido a que los videos otorgados por los estudiantes de EDCOM eran de alta calidad sobre todo cuando la conexión de internet era lenta. Para realizar esta prueba de manera exitosa, siempre era necesario tener una conexión de internet rápida; lo cual siempre era difícil de obtener en cualquier facultad de ESPOL.

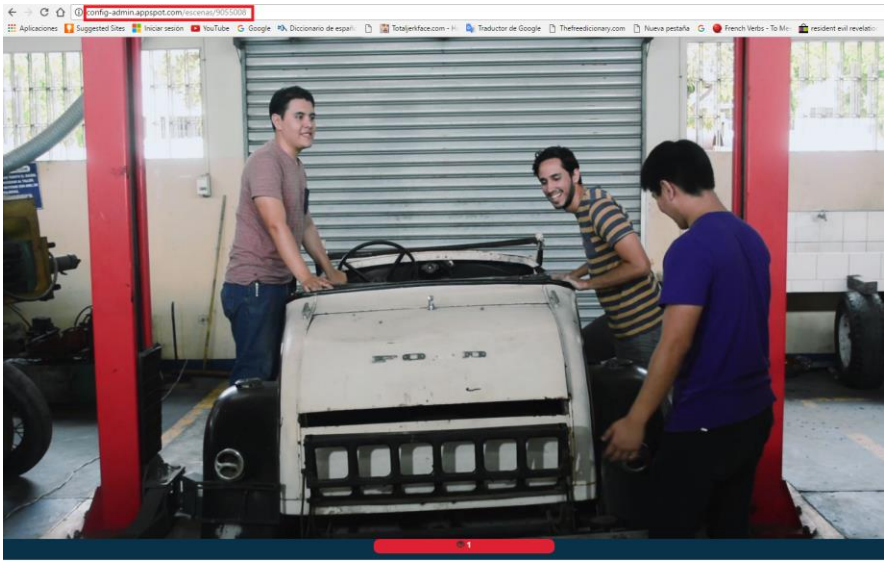


Figura 2.20: Streaming de video desde la plataforma de Google.

Streaming de video de manera local

Dado el inconveniente existente al reproducir videos en alta definición, se decidió crear una versión de película de manera local (Figura 2.21) donde solamente los videos se encuentran almacenados localmente, y se mantiene en la nube los demás servicios brindados al componente front-end. La URL de la película es la siguiente: http://localhost/escenas/9055008.

Gracias al almacenamiento y trasmisión de las escenas de videos de manera local, se solucionó completamente el retraso de los videos. De tal manera que se pudo realizar la transmisión de los mismos de manera fluida.

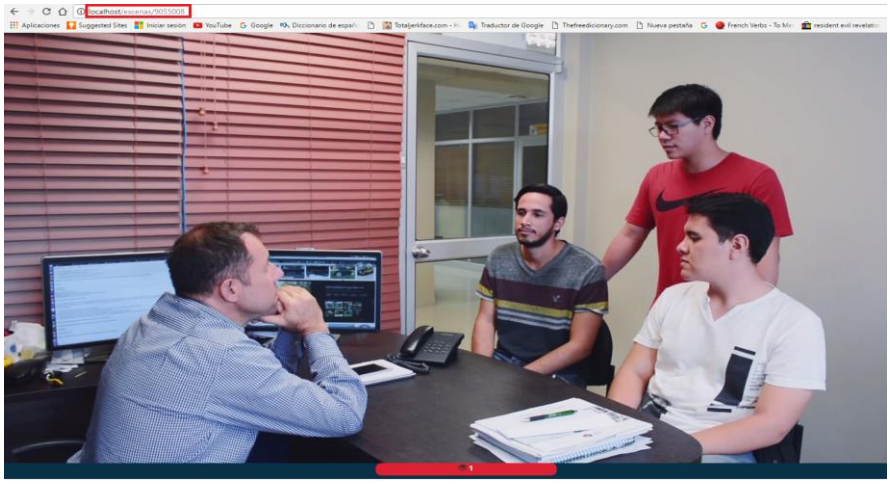


Figura 2.21: Streaming de video de manera local.

Para este modo de *streaming*, también se destaca que las URLs de las escenas se guardan en el *json* de la película con el siguiente formato: http://localhost/peliculas/05_002_PREG_1.mp4. Tal como se observa en esta URL, de manera local se alojaron las escenas, para lo cual se creó la carpeta *películas* dentro del directorio principal del módulo 1.

f. Interacción entre primera y segunda pantalla.

Selección de escenas siguientes

Para poder probar la selección de escenas siguientes mediante la votación de distintos usuarios conectados a una sala de película, se creó una página de prueba llamada *2v* la cual simula la aplicación móvil de segunda pantalla.

En la Figura 2.22, se aprecia las opciones de escenas siguientes de la escena actual en la segunda pantalla en forma de botones (*indexSI* y *indexNO*); los mismos que aparecen segundos antes de finalizar la escena actual si y sólo si dicha escena posee interacción. Finalmente, durante ese tiempo de votación cada usuario conectado a la sala escoge la escena siguiente para definir el rumbo de la historia de la película, y la escena con mayor votación es la que aparece al finalizar la escena actual.

Figura 2.22: Segunda pantalla de prueba con las opciones de escenas siguientes.

En la Figura 2.23, se muestra un ejemplo del archivo *json* de todas las escenas posibles de una película en el lado del cliente en la primera pantalla luego de haber leído el *json* desde la base de datos. Debido a que la escena actual tiene interacción (valor *true*), se parsea el *json* de escenas y se compara el ID de la escena actual con el campo Escena_Padre de las demás escenas.

Si ambos IDs coinciden, entonces se guarda en un arreglo la escena cuyo padre es la escena actual. Luego, se emite desde la primera pantalla al servidor un evento conteniendo un *json* compuesto por las escenas siguientes y la pregunta de interacción de la actual escena. Finalmente, el servidor emite un evento similar a la segunda pantalla con los mismos parámetros en los últimos 20 segundos de la escena actual.


```
"Escenas": [
  {
    "Name_Escena": "inicio",
    "Id_Escena": "12708989",
    "Pregunta": "https://firebasestorage.googleapis.com/v0/b/config-admin.
    "Tiempo": "20",
    "Default_Escena": false,
    "Interaccion": false,
    "Escena": "https://firebasestorage.googleapis.com/v0/b/config-admin.ap
    "Escena_Padre": ""
  },
  {
    "Name_Escena": "indexNO",
    "Id_Escena": "7080108",
    "Pregunta": "",
    "Tiempo": "",
    "Default_Escena": false,
    "Interaccion": false,
    "Escena": "https://firebasestorage.googleapis.com/v0/b/config-admin.ap
    "Escena_Padre": "12708989"
  },
  {
    "Name_Escena": "indexSI",
    "Id_Escena": "12363269",
    "Pregunta": "",
    "Tiempo": "",
    "Default_Escena": false,
    "Interaccion": false,
    "Escena": "https://firebasestorage.googleapis.com/v0/b/config-admin.ap
    "Escena_Padre": "12708989"
  }
]
```


Figura 2.23: Estructura *json* de todas las escenas posibles de una película.

Petición de contenido adicional transmedia

Este punto abarca una de las interacciones entre primera y segunda pantalla al momento de enviar una imagen adicional correspondiente a la escena que se está visualizando en un instante de tiempo dado.

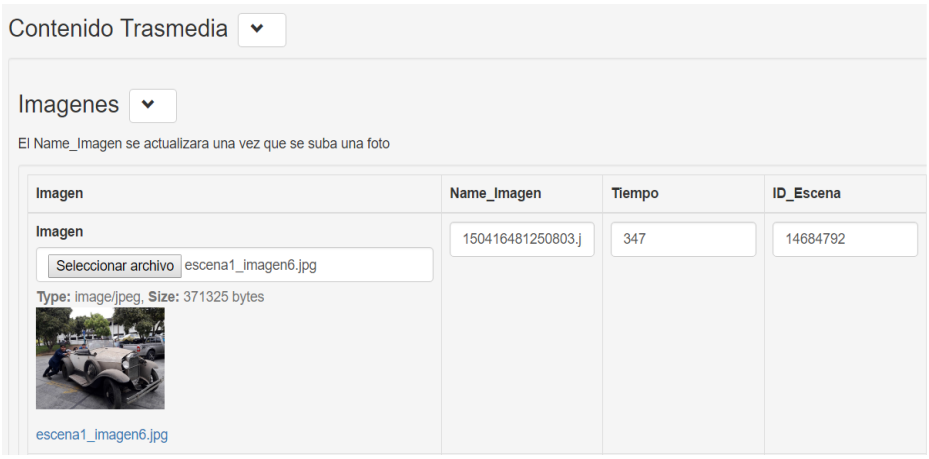


Figura 2.24: Contenido adicional transmedia - imágenes.

Tal como se percibe en la Figura 2.24, en el *json* de configuración de contenido de una película en la sección *Contenido Transmedia*, el administrador puede subir imágenes referentes a una escena dada. Cada imagen está compuesta por los siguientes campos: imagen a subir, nombre de la imagen (Name_imagen) generado en base a la hora actual, tiempo medido en segundos que indica el momento en el cual debe aparecer la imagen, y el ID de la escena a la cual la imagen va estar vinculada.

Por ejemplo, en la misma figura se observa un valor de 347 en la variable *tiempo*; lo cual indica que la imagen debe aparecer en la segunda pantalla cuando a la escena que se está transmitiendo le restan 347 segundos por finalizar.

Cuando llega el tiempo indicado, la primera pantalla emite por medio de socket.io el evento *image* enviando la URL de la imagen al servidor. Luego, el servidor emite un evento con la misma URL de la imagen a la segunda pantalla. Finalmente, una vez recibida la URL, en la aplicación móvil se

muestra la imagen respectiva de la escena en el tiempo indicado. Cabe destacar que una escena puede tener más de una imagen asociada.

2.4 Módulo de Sala de Chat

2.4.1 Metodología

Para este módulo, se utilizó la librería socket.io debido a que ésta permite suscribir una conexión socket a una sala de chat dada, y poder emitir *broadcasts* de todos los mensajes que llegan al canal a todos los usuarios suscritos o sincronizados.

El usuario al iniciar la aplicación móvil, inmediatamente ésta inicia sesión a través de la cuenta de Facebook del usuario obteniendo datos como la foto de perfil y nombre de usuario. Estos mismos datos sirven para poder iniciar la sala de chat desde la aplicación.

Con el mismo ID de la sala de la película obtenido desde el código QR, la aplicación móvil suscribe al usuario a la sala de chat correspondiente a la película que se transmite.

Herramientas y tecnologías utilizadas

Con respecto a las herramientas utilizadas en este módulo al igual que en el primero, se utilizaron las siguientes librerías: node.js, express.js y socket.io.

2.4.2 Implementación

Cada mensaje de los usuarios suscritos a una sala de chat es enviado por medio de socket.io desde la aplicación móvil, esta misma emite un evento el cual le envía al servidor un *json* compuesto por el logo del perfil de Facebook del usuario, y un texto conformado por su nombre de usuario de Facebook más el mensaje de texto enviado desde la aplicación.

Una vez que el servidor recibe el *json* desde la aplicación móvil, éste emite un evento a la primera pantalla con los mismos parámetros del *json*. Finalmente, en la primera pantalla (Figura 2.25), se muestra en tiempo real los 3 parámetros mencionados cada vez que un usuario envía un mensaje.

Figura 2.25: Chat entre los usuarios suscritos a una sala de película.

CAPÍTULO 3

3. ANÁLISIS Y RESULTADOS

En esta sección, se muestran los detalles de los resultados con sus respectivos análisis con el fin de poder presentar una propuesta de mejora al problema ya planteado en la sección Introducción. También se presentan las pruebas de rendimiento con pedidos simultáneos con el fin de evaluar la latencia y *throughput* del servidor del primer módulo.

3.1 Resultados del producto final

A continuación, se muestra los distintos resultados del producto final tales como: código QR, integración con el componente front-end, pruebas de funcionamiento y pruebas de rendimiento.

3.1.1 Primera pantalla: Código QR.

En la Figura 3.1, se observa en la primera pantalla el código QR que representa el ID de la sala de la película.

Figura 3.1: Primera pantalla - Código QR.

La aplicación de segunda pantalla mediante el escaneo de este código recepta el ID de la sala, y a través de socket.io se une a la sala. A partir

de ese momento, la segunda pantalla se encuentra sincronizada y puede interactuar con la primera pantalla tanto en la votación de escenas siguientes como en las consultas de imágenes adicionales referentes a las escenas de la película.

3.1.2 Integración con el componente front-end

En esta sección, se muestran las distintas interacciones entre primera y segunda pantalla tales como la selección de escenas siguientes, sala de chat y contenido transmedia adicional.

a. Selección de escenas siguientes en la segunda pantalla.

En la Figura 3.2, se observa la interacción del usuario entre la primera y segunda pantalla mediante la elección de escenas siguientes. Si la escena actual que se está transmitiendo tiene interacción de selección de escenas siguientes, en los últimos 15 o 20 segundos de la escena actual se le muestra al usuario en la aplicación móvil las dos opciones SI o NO que representan las escenas siguientes y sirven para que el usuario a través de la selección de una de ellas pueda definir el rumbo de la película.

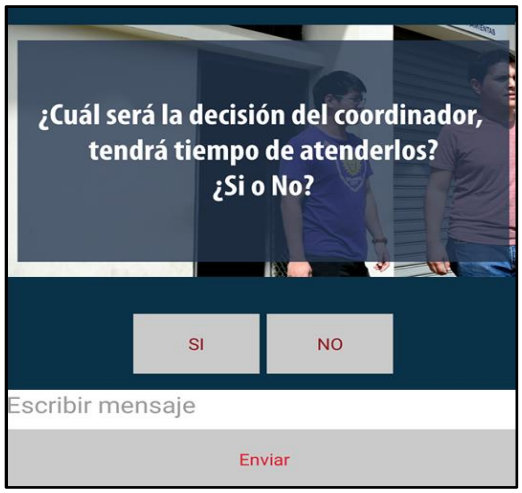


Figura 3.2: Segunda pantalla - selección de escenas siguientes.

Luego, la escena que haya recibido mayor votación por parte de los usuarios suscritos a la sala de la película es escogida para ser mostrada tal como se ve en la Figura 3.3.

Figura 3.3: Primera pantalla - Escena escogida.

b. Sala de chat en la primera pantalla.

En la Figura 3.4, se muestra la interacción mediante una sala de chat entre los usuarios suscritos al canal de una película mientras se realiza la transmisión en vivo de la misma.

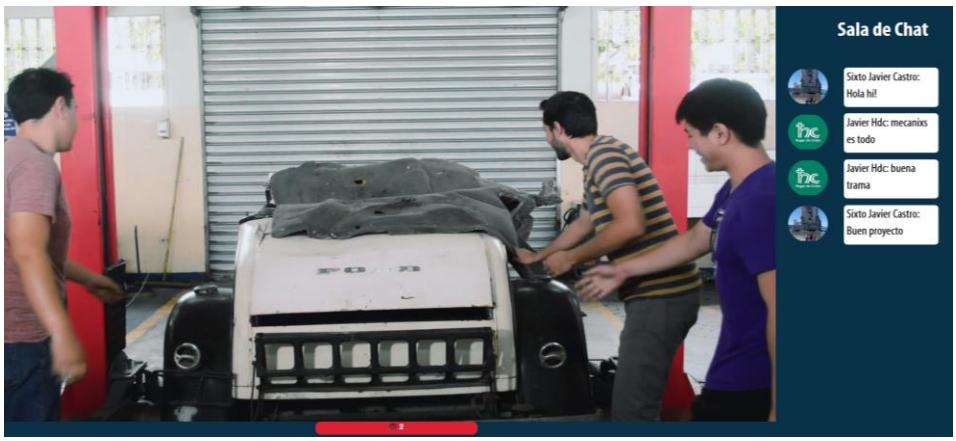


Figura 3.4: Primera pantalla - Sala de chat.

En la Figura 3.5, se muestra la opción para que el usuario desde la segunda pantalla envíe un mensaje de texto; y este mensaje a su vez sea enviado a la primera pantalla usando como intermediario al servidor del segundo módulo a través de socket.io.

Figura 3.5: Mensaje enviado desde la aplicación móvil.

Tal como se aprecia en la Figura 3.6, el mensaje enviado desde la aplicación móvil aparece en la primera pantalla (encerrado con rojo), el mismo que está compuesto por la foto de perfil, nombre de usuario de Facebook y el mensaje enviado desde la aplicación de segunda pantalla.

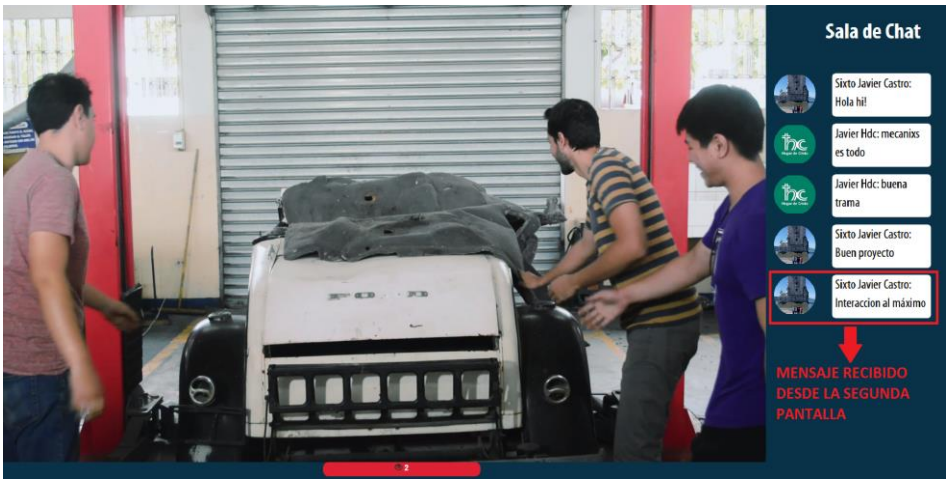


Figura 3.6: Mensaje recibido desde la segunda pantalla.

Debido al hecho de que cada sala de película tiene su respecto ID, cada sala posee su respectivo chat. Por la tanto, el módulo de chat está implementado para ser multiusuario y multicanal.

c. Contenido adicional transmedia.

En la Figura 3.7, se aprecia el contenido transmedia adicional que en este caso son las imágenes complementarias a la escena que se transmite en la primera pantalla. Cada imagen está sincronizada con la primera pantalla, la misma que es enviada desde la primera a la segunda pantalla en un tiempo indicado mientras se está transmitiendo una escena de la película.

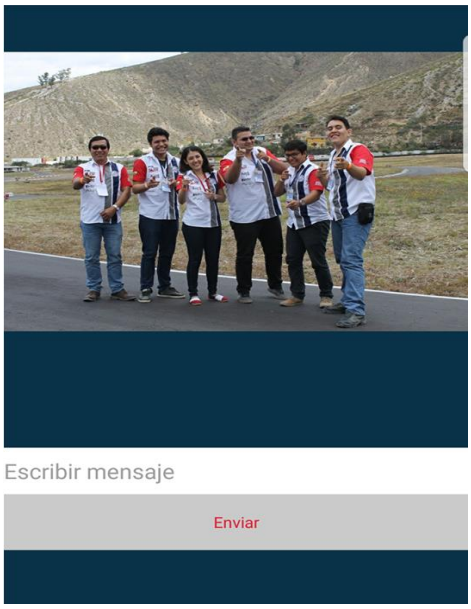


Figura 3.7: Segunda pantalla - Contenido adicional.

3.2 Pruebas de funcionamiento

Con respecto a estas pruebas, se probó las distintas interacciones entre primera y segunda pantalla con el fin de evaluar el estado de las redes disponibles en

ESPOL y ver qué puertos están disponibles. A continuación, se muestra el listado de redes donde se hizo el *testing* de ambas pantallas:

- Red CVR (Centro de Visión y Robótica): Prueba exitosa y experiencia satisfactoria. Puertos desbloqueados tanto para sincronización y para sala de chat, y conexión de internet rápida.
- Red ESPOL: Se probó con la red de Cine ESPOL donde se obtuvo inconvenientes tales como conexión de internet lenta debida a que muchas estudiantes se conectan a dicha red, y no se pudo sincronizar ambas pantallas y chatear debido a los puertos bloqueados. También se presentó el inconveniente de escaneo de código QR debido a la falta de iluminación de la sala, y la distancia a la cual el usuario intenta escanear dicho código.
- Red FIEC: Internet lento e interacción satisfactoria con retraso en la carga y precarga de los videos a transmitirse.

3.3 Pruebas de rendimiento

El objetivo de estas pruebas es poder evaluar el rendimiento del servidor del primer módulo en términos de latencia y *throughput*.

Para poder realizar estos experimentos, se utilizó un servidor de prueba con el fin de simular muchos usuarios haciendo petición del archivo *json* de IPs de manera simultánea. Y cada punto en las gráficas es el promedio de haber ejecutado o repetido el experimento 5 veces.

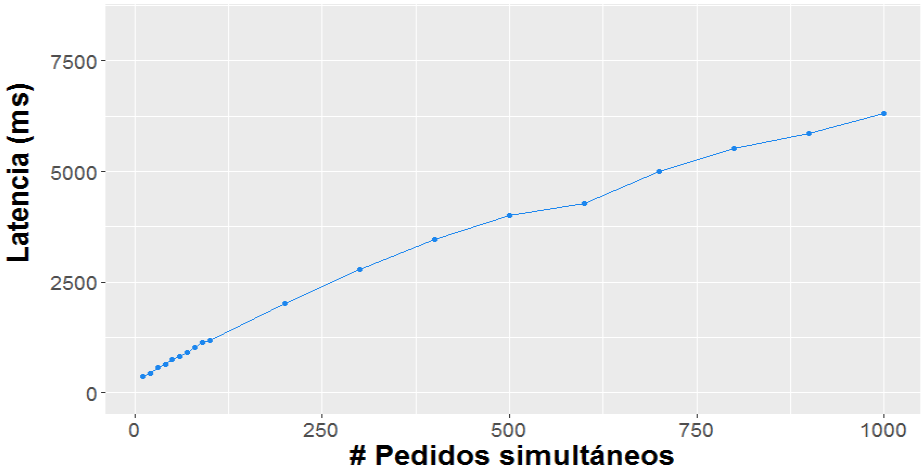


Figura 3.8: Latencia vs # Pedidos simultáneos.

En la Figura 3.8, se evaluó el rendimiento del servidor de configuración en términos de latencia a medida que el número de usuarios incrementaba con el fin de saber qué tiempo le tomaba al servidor en responder al cliente cuando este último realizaba una petición del archivo *json* de IPs.

Se puede destacar que la gráfica presenta un comportamiento creciente y es aproximadamente lineal. A medida que el número de pedidos simultáneos (número de usuarios) aumenta, la latencia incrementa.

Se aprecia que, para 5000 pedidos simultáneos, la latencia es igual a 4 segundos; en cambio para 1000 peticiones, la latencia es de 6.3 segundos. Lo cual indica que a medida que incrementa el número de pedidos, la plataforma de *Google Cloud* trata de minimizar la latencia aumentando más recursos al servidor.

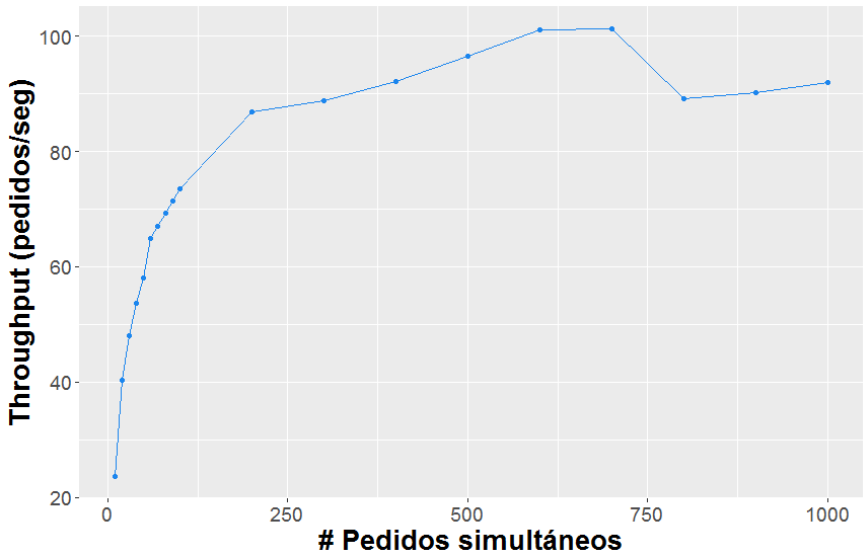


Figura 3.9: Latencia vs # Pedidos simultáneos.

En la Figura 3.9, se muestra la gráfica de Throughput (pedidos/seg) vs # Pedidos simultáneos, donde se aprecia que a medida que el número de pedidos incrementa, el *throughput* también aumenta llegando a un máximo de 101 pedidos/seg en el tramo de 10 a 700 pedidos simultáneos. Para 800 peticiones, se observa que el *throughput* decae; pero inmediatamente obtiene un leve incremento en el último tramo entre 900 y 1000 peticiones.

Se puede decir que, debido a la elasticidad de la plataforma de Google, el número de recursos aumenta para la aplicación al llegar a un cierto tráfico; es decir la plataforma trata de nivelar el rendimiento del servidor aumentando más memoria y más núcleos sin dejar que el *throughput* siga decayendo ante la presencia de más pedidos simultáneos.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Se logró realizar el “Desarrollo de una aplicación móvil de segunda pantalla para complementar productos audiovisuales transmedia (back-end)“ con tecnologías nuevas así como Node.js, HTML5, MongoDB dejando atrás las más comunes como Java, Python, y .Net.

La interacción con el usuario se logró de tal manera que éste pueda decidir o escoger las escenas siguientes a visualizar y poder obtener los recursos adicionales relacionados a las mismas en este caso imágenes.

El método de sincronización escogido fue por Código QR debido a que no se pudo encontrar una herramienta que permita agregar marcas de agua o huellas digitales a los videos de manera dinámica sin tener que antes pre-procesar los videos. Además, la sincronización por mecanismos de sonido es muy sensibles al ruido del ambiente.

La plataforma *Google Cloud* provee escalabilidad y elasticidad a mayores cargas de trabajo aumentando más recursos al servidor tanto como memoria y número de núcleos con el fin de minimizar el tiempo de respuesta (latencia), y maximizar el *throughput* (pedidos/seg).

Recomendaciones y Trabajos futuros

En esta sección, se detallan las recomendaciones y trabajos futuros del proyecto realizado.

Recomendaciones

Como el proyecto fue adaptado a la arquitectura que nos proveía la plataforma de *Google Cloud*, si se desea replicar este proyecto para una producción comercial se recomienda para su funcionamiento óptimo contratar como hosting la misma plataforma. También debido a que *Google Cloud* soporta las herramientas utilizadas tales como: node.js, express.js, HTML5, mongoDB, socket.io, etc.

Al momento de visualizar la película en la plataforma se recomienda asegurarse de disponer de una aceptable conexión a internet y que los puertos para conectarse por socket.io estén habilitados. Con respecto al caso de conexión lenta de internet, se sugiere precargar cierto porcentaje de las escenas antes de reproducirlas con el fin de contrarrestar el *lag* de las mismas, y se pueda hacer una transmisión fluida.

La sincronización por código QR también tiene sus desventajas al momento de leer el código influyen factores como el color de fondo del mismo, la iluminación del lugar donde se lo proyecta, y la distancia a la cual el usuario utiliza el lector de código QR. Motivo por el cual, si el contenido audiovisual es obtenido con anterioridad, se recomienda usar como medio de sincronización marcas de agua por ultrasonido.

El navegador a usar es Google Chrome y Firefox para la página de administrador debido a las restricciones de CSS que impone Microsoft Edge al momento de cargar las páginas de administración de contenido.

Trabajos futuros

Contratar la versión de paga del hosting de *Google Cloud* para poder obtener mayor escalabilidad al momento de una sobrecarga o alta concurrencia de usuarios utilizando la aplicación.

Se podría adaptar este proyecto a cualquier cliente que desee implementar una aplicación de segunda pantalla para sus contenidos siempre y cuando se desarrolle una aplicación móvil personalizada para la misma.

Implementar la reproducción de videos en vivo y no subidos a la plataforma para que se pueda interactuar con diferentes usuarios en diferentes lugares del mundo sin necesidad de estar visualizándolos en la misma pantalla.

Desarrollar una aplicación para un televisor Smart para poder visualizarla nativamente en el mismo.

BIBLIOGRAFÍA

[1] Christian Pastrana (2013, noviembre 12). Narrativa Transmedia: una historia, un mundo, múltiples plataformas [online]. Disponible en:

http://comunidad.iebschool.com/iebs/gamification-narrativa-transmedia/que-es-la-narrativa-transmedia/. [Accedido el 18 de mayo de 2017].

[2] José Ángel Domínguez Calatayud (2013, julio 7). Second Screen: lo que observamos [online]. Disponible en:

http://www.jadominguez.com/2013/07/second-screen-lo-que-observamos/. [Accedido el 18 de mayo de 2017].

[3] Cómo conectar tu móvil o tu tablet con tu televisor, 2014 [online]. Disponible en: http://www.samsung.com/es/article/screen-mirroring-como-conectar-el-movil-con-el-televisor/. [Accedido el 15 de mayo de 2017].

[4] Oliverio Pérez Villegas (2013, marzo 3). Segunda pantalla, la tendencia del marketing sin TV [online]. Disponible en:

http://www.altonivel.com.mx/34384-segunda-pantalla-la-tendencia-del-marketing-sin-tv/. [Accedido el 15 de mayo de 2017].

[5] Rob Crossley (2014, noviembre 18). Wii U: The year two review [online]. Disponible en:

https://www.gamespot.com/articles/wii-u-the-year-two-review/1100-6423304/. [Accedido el 17 de mayo de 2017].

[6] Karppinen, J. (2013). Discovering Social TV and Second Screens-Proposing an architecture for distributing second screen content.

[7] Ittousai (2013, marzo 21). Google Chrome World Wide Maze [online]. Disponible en:

http://japanese.engadget.com/2013/03/21/google-chrome-world-wide-maze/. [Accedido el 18 de mayo de 2017].

[8] Jaime Fernández de la Puente-Campano (2015). ¿Qué es el Audio Watermarking y Audio Fingerprint? [online]. Disponible en: http://www.jaimefernandez.com/que-es-el-audio-watermarking-y-audio-fingerprint/. [Accedido el 17 de mayo de 2017].

[9] YouTube (2012, febrero 22). The Walking Dead - Walkers Kill Count iPhone iPad app [online]. Disponible en:

https://www.youtube.com/watch?v=LmosA8sgvws. [Accedido el 18 de mayo de 2017].

[10] Node.js (2017). Acerca de Node.js, 2017 [online]. Disponible en: https://nodejs.org/es/about/. [Accedido el 30 de mayo de 2017].

[11] Alejandro Morales (2012, enero 22). Express – El framework web para nodejs [online]. Disponible en:

http://www.nodehispano.com/2012/01/express-el-framework-web-para-nodejs/, Enero 21, 2012. [Accedido el 5 de junio de 2017].

[12] W3Schools (2017). Bootstrap 3 Tutorial [online]. Disponible en: https://www.w3schools.com/bootstrap/. [Accedido el 7 de junio de 2017].

[13] Github (2017). JSON Schema Based Editor [online]. Disponible en: https://github.com/jdorn/json-editor. [Accedido el 10 de junio de 2017].

[14] Github (2017). A web-based tool to view, edit, format, and validate JSON [online]. Disponible en: https://github.com/josdejong/jsoneditor. [Accedido el 13 de junio de 2017].

[15] Sergio De Luz (2010, noviembre 4). Criptografía: Algoritmos de cifrado de clave simétrica. Disponible en:

https://www.redeszone.net/2010/11/04/criptografia-algoritmos-de-cifrado-de-clave-simetrica/. [Accedido el 13 de junio de 2017].

[16] Sergio De Luz (2010, noviembre 9). Criptografía: Algoritmos de autenticación (hash) [online]. Disponible en:

https://www.redeszone.net/2010/11/09/criptografia-algoritmos-de-autenticacion-hash/. [Accedido el 15 de junio de 2017].

[17] npm (2017). Socket.io [online]. Disponible en:

https://www.npmjs.com/package/socket.io. [Accedido el 15 de junio de 2017].

[18] Jean Carlos Mariños Urquiaga (2017). ¿Cómo configurar sesiones en ExpressJS?, DevCode Tutoriales [Online]. Disponible en:

https://devcode.la/tutoriales/como-configurar-sesiones-en-expressjs/. [Accedido el 17 de junio de 2017].

[19] Github (2017). Generate artistic QR code [online]. Disponible en: https://github.com/kciter/qart.js. [Accedido el 14 de julio de 2017].

[20] ESET Latinoamérica (2016, mayo 5). Contraseñas fuertes y fáciles de recordar: ¡Se necesita una frase! [online]. Disponible en: http://www.eset-la.com/centro-prensa/articulo/2016/contrase%C3%B1as-fuertes-faciles-recordar/4239. [Accedido el 10 de junio de 2017].

[21] Boris Smus (2013, agosto 8). Ultrasonic networking on the web. Disponible en: http://smus.com/ultrasonic-networking/. [Accedido el 15 de junio de 2017].

[22] Github (2017). quiet/quiet-js [online]. Disponible en: https://github.com/quiet/quiet-js. [Accedido el 22 de junio de 2017].