Universidad de la República
Facultad de Ingeniería
Institito de Computación

Gestión de contenidos en formato video

Informe del proyecto de grado de Ingeniería en Computación presentado al Tribunal Evaluador

Proyecto de grado presentado por: María del Carmen Invernizzi
Juan Bertoni
Diego Martínez

Supervisor: Antonio López
Cliente: *Hopclip*

Montevideo, 21 de noviembre de 2018

Proyecto de grado:
Gestión de contenidos en formato video

Resumen:

El contenido audiovisual ocupa actualmente una gran porción del tráfico en Internet y se estima que siga creciendo en los próximos años. Este formato es ampliamente utilizado en varios ámbitos: resúmenes deportivos, cursos y material educativo, eventos sociales y noticias, entre muchos otros usos posibles.
En cada uno de estos ámbitos, es fácil encontrar ejemplos en los que una persona que visualiza un video en línea necesite conocer el contenido del video de antemano y pueda ubicar fácilmente el momento en que ocurre un hecho que le interesa particularmente. Paralelamente, durante la generación de contenido en formato video, es deseable contar con la posibilidad de generar marcas para poder clasificar el material y proveer un mecanismo que permita navegar en el contenido del video mientras se lo reproduce. Esto resulta más interesante aún si la generación de metadatos puede realizarse sin intervención humana.

Este proyecto de grado plantea utilizar el concepto de índice para identificar los eventos relevantes dentro del contenido audiovisual, de manera similar a lo que ocurre con los índices en los libros. Los metadatos que interesa incluir en los videos en línea consisten en una descripción asociada a un instante de tiempo de un video. A estos metadatos le llamaremos hitos, los índices que utilizaremos consisten en una lista de hitos que pueden pertenecer a diferentes videos.

En el presente trabajo se aborda el problema de generar una plataforma para gestionar estos índices de videos en línea. Este proyecto se centra en brindar una interfaz para usuarios finales, integrándose con otro proyecto de grado para cubrir la generación automática de los hitos. Como parte de este proyecto de grado, se evaluaron las herramientas existentes relacionadas a la temática, se estudiaron distintos tipos de tecnologías, se diseñó y construyó una plataforma para gestionar índices de videos en línea.

Se implementó una capa de servicios web, un componente indexador para crear índices de videos, un reproductor para visualizarlos y una *progressive web app* para mostrar el contenido e integrar los distintos componentes, esta aplicación funciona tanto como portal web así como aplicación móvil.

Palabras claves:
indexación, video, índice de video, metadatos, búsquedas dinámicas

Tabla de Contenido

Lista de Figuras

Capítulo 1

Introducción

1.1. Introducción

El tráfico en Internet está cambiando cada día. Cada vez son más los usuarios que utilizan los teléfonos móviles, las tablets, televisores *smarts* y demás dispositivos inteligentes para navegar en Internet, lo que implica una constante generación de datos. El mundo de las redes sociales y de los archivos multimedia hacen que gran parte de esa información dispersa por la red esté en formato video. Una investigación realizada por *Cisco* [CIS] estima que para 2021 el 82 % de todo el tráfico de Internet será de tipo video; lo que lleva a decir que cada segundo, casi un millón de minutos de contenido de video cruzará la red.
Este aumento exponencial de información, hace muy compleja la tarea de buscar y encontrar lo que realmente el usuario desea ver, siendo aún más difícil encontrarlo dentro del contenido de un video propiamente dicho.

Hoy en día hay cientos de videos *on line* publicados en múltiples plataformas como *YouTube*, *Vimeo*, *DailyMotion*, etc; que son visualizados, compartidos y comentados por millones de usuarios, pero surgen las siguientes interrogantes: ¿Los videos que retornan los motores de búsqueda de estas plataformas, son aquellos que realmente queríamos ver? ¿Dichos videos contienen la información que buscamos? ¿Qué tan fácil o complejo es encontrar en el contenido del video específicamente lo que nos interesa? ¿Qué tan frustrante es para los usuarios tener que navegar por la línea de tiempo del video para poder ir previsualizando las diferentes secciones e ir buscando lo que necesita, y en muchos casos ni siquiera encontrarlo?

Actualmente los motores de búsqueda de las plataformas de video *on line*, utilizan metadatos tales como el título, descripción, etiquetas, comentarios, puntuación, tiempo promedio de visualización, número de reproducciones, entre otros, para ofrecerle al usuario una lista de videos que cumplen con ciertos criterios pero sin tener la certeza de que lo buscado se encuentra incluido en su interior.

Para ejemplificar, no resulta práctico tener que ver los 90 minutos de un partido de fútbol cuando solo se está interesado en ver los goles que se hicieron. Así como tener que ver completamente una conferencia extensa, una capacitación o un videotutorial para encontrar un solo tema relevante y específico. Llevándolo a un ambiente más cotidiano, podríamos querer tener un resumen de un viaje con grabaciones de videos en diferentes partes del mundo, donde fuera sencillo poder identificar el país que se está mostrando en cada instante de tiempo; o un video de una ceremonia fa-

miliar que se divide en diferentes momentos y solo se quiere visualizar uno de ellos y obviar los otros.

Esta necesidad no solo es interesante desde el punto de vista de la eficiencia en las búsquedas dentro de los videos y el ahorro de tiempo a la hora de navegar a un punto determinado, sino que también es muy beneficioso poder conocer de antemano todo el contenido o temario que se va a tratar en un video, para que sin tener la necesidad de visualizarlo, poder definir si el mismo engloba todo el conocimiento relevante.

A partir de estas premisas, y basado en la idea de indexación utilizada en los libros cuyo objetivo es clasificar, organizar y registrar ordenadamente el contenido incluido en el mismo, surge la idea de utilizar un índice en los videos.

En éste ámbito, un **índice** se puede definir como la organización del contenido audiovisual incluido en un video a partir de la definición de marcas o eventos puntuales en la línea de tiempo, que representan algo o tienen algún significado importante en el contexto del video que se quiera resaltar. A cada una de las marcas identificadas en el video se le denomina **hito**, que no es más que una tupla a la que se le asigna una descripción junto al instante de tiempo en el que fue definido. En la figura 1.1 pueden identificarse los conceptos antes descriptos con un ejemplo generado en la plataforma construida.

Para este proyecto, ésta definición de **índice** se puede generalizar aún más contemplando la indexación de múltiples videos donde para cada uno de ellos puede definirse un conjunto de hitos asociados.
Además, un **índice** es el resultado de la búsqueda de contenido a partir de un cierto criterio.

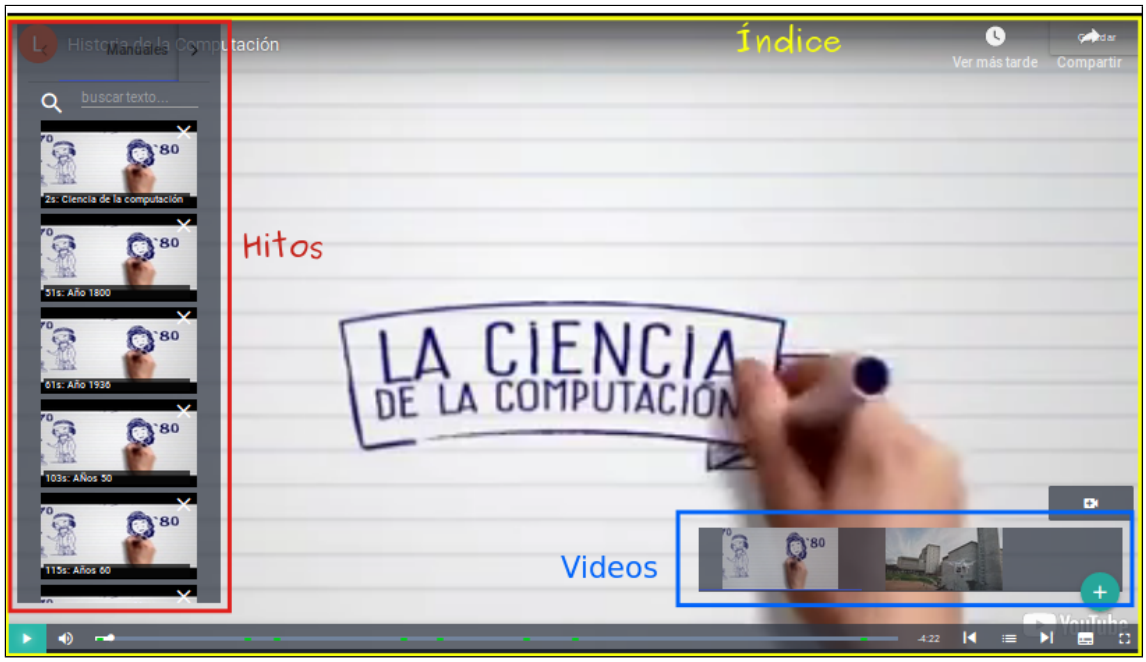


Figura 1.1: Definición de Índice

De este modo, cada video incluido en un índice tendrá un conjunto de hitos definidos manualmente por un usuario, que enriquecerán la información de los videos de forma tal que a la

hora de realizar búsquedas de contenido en cualquier video, puedan ser utilizados como metadatos y puedan ser encontrados más alla del título o de la descripción que pueda tener el propio video.

Además de las búsquedas, al elegir visualizar un video indexado, mejora notoriamente la experiencia de usuario al ser capaz de conocer todo el contenido del mismo sin importar que tan extenso es el video, cuales son sus puntos más importantes y lo mejor de todo, poder navegar fácilmente y sin pérdidas de tiempo al momento específico del video que se desea reproducir.

Identificado el problema que plantea este proyecto, se propone construir una completa plataforma de gestión de información contenida en videos *on line* (*YouTube*, *Vimeo*, etc) que permita a cualquier usuario de Internet poder crear índices sobre cualquier video *on line*, asociando descripciones a instantes de tiempo de videos y navegarlos utilizando dicha información.

Para el alcance de este proyecto se pretenden generar índices de forma manual, permitiendo que los usuarios finales creen sus propios índices marcando los hitos importantes que consideren pertinentes, pero se espera poder extender esta funcionalidad de modo que a través de herramientas que permitan el reconocimiento automático de texto, extracción de subtítulos, reconocimiento facial, de objetos, entre otras; permita la generación de hitos automáticamente. Para esto, se implementará una interfaz que se comunique con la solución planteada por el proyecto de grado *Video++2*, de modo de nutrir aún más la información contenida en los videos sin la necesidad de intervención manual del usuario.

El proyecto consta entonces de la construcción de la plataforma, con sus componentes/servicios principales, tal como se observa en la figura 1.2:

1 Un indexador que permita a un usuario interactuar con un video *on line* y crear índices (un *timestamp* + una descripción)

2 Un reproductor que permita mostrar el video y "navegarlo" usando su índice

3 Un portal web basado en servicios que permita presentar todos los videos indexados existentes (utilizando el indexador y el reproductor) y que dispondrá de un buscador.

4 Una aplicación móvil que consuma servicios del portal y permita indexar, visualizar y navegar sobre los contenidos del portal

5 Una interfaz que exponga servicios a ser consumidos por cualquiera de los componentes del sistema

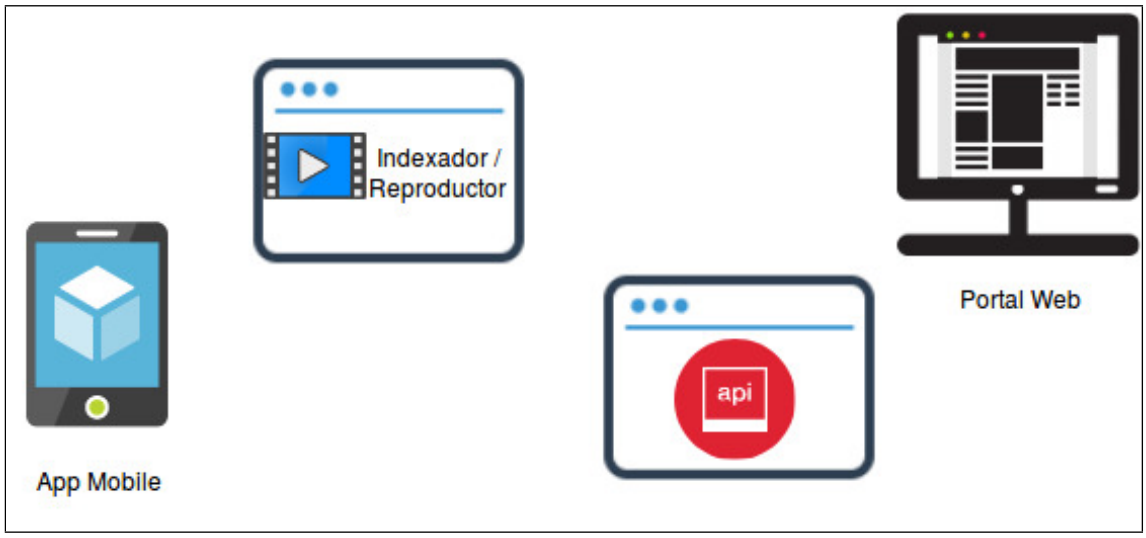


Figura 1.2: Componentes

Para llevar a cabo la propuesta, se realizará una revisión exhaustiva de herramientas existentes similares a lo que se propone construir. Posteriormente se adquirirán conocimientos para construir aplicaciones web y aplicaciones móviles que interactúan con una plataforma de servicios (también construida en el marco de este proyecto) que deberá interactuar con múltiples proveedores de servicios de video en línea (*YouTube*, *Vimeo*, etc). Tanto el desarrollo de este tipo de aplicaciones con tanta lógica del lado del cliente como la construcción de plataformas de servicios requiere del dominio de múltiples tecnologías avanzadas, y su aprendizaje y aplicación a este problema concreto será uno de los objetivos del proyecto.

1.2. Organización del documento

El resto de los capítulos están organizados de la siguiente manera:

En el Capítulo 2 se dará un pantallazo del estado del arte, mencionando algunas de las herramientas existentes para la gestión de videos que contemplan, en parte, ciertos problemas a resolver en el marco de este proyecto.

En el Capítulo 3 se plasma a través de bosquejos la idea de solución y a partir de allí se realiza el análisis de requerimientos y se define el alcance del proyecto con sus restricciones.

En el Capítulo 4 se describe la arquitectura y se mencionan algunas decisiones de diseño que tuvieron que ser tomadas, como la elección de tecnologías para el desarrollo de la aplicación móvil. Se detallan algunos puntos importantes en el diseño de los diferentes componentes e interfaces y se describe cómo se modelaron los datos.

El Capítulo 5 explica cómo fue implementada la solución de cada uno de los componentes y servicios diseñados, donde se explican las tecnologías y *frameworks* utilizadas para llevar a cabo el trabajo. Además se menciona el reproductor de video empleado y cómo se manipulan sus elementos para incluir las funcionalidades que se bosquejaron.

En el Capítulo 6 se detallan algunas de las pruebas funcionales y tests realizados para verificar y validar que la implementación adhiere a la especificación realizada, con sus resultados obtenidos.

El Capítulo 7 contiene una breve reseña del proceso de gestión, cómo el mismo se desvió del plan inicial y sus justifiaciones.

Por último el Capítulo 8 describe algunas de las conclusiones a las cuales se llegó una vez finalizado el proyecto, junto con posibles funcionalidades a ser incluidas como trabajo a futuro en el corto o mediano plazo.

Capítulo 2

Estado del arte

En este capítulo se presenta una investigación que pretende mostrar el estado actual de la gestión de metadatos en videos, encontrar qué herramientas o productos existen en funcionamiento, cuáles son las limitantes por la que muchos de ellos no han perdurado en el tiempo, e intentar entender la razón por la cual no se cuenta todavía con índices en la mayoría de los videos que los usuarios finales utilizan cotidianamente. Como forma de ordenar las ideas en ésta búsqueda de antecedentes, se decidió dividir el capítulo en diferentes secciones.

En la primera sección, se presentan diferentes herramientas que administran contenidos de videos *on line* similares a la propuesta de este proyecto. De cada una se realiza una breve introducción de sus funcionalidades y su estado actual.

En una segunda sección se presentan herramientas utilizadas para la gestión de contenido audiovisual en emisoras de televisión, lo que aporta grandes conocimientos de gestión de contenido gráfico de mucho volumen. Además se incluye una breve reseña de entrevista con una autoridad de uno de los canales de TV más importantes de Uruguay.

Finalmente, se redondea el capítulo con una breve síntesis.

2.1. Herramientas para la gestión de videos en línea

A continuación se presenta una lista de plataformas (aplicaciones, servicios web y complementos), algunos discontinuados, que pueden encontrarse realizando búsquedas en Internet y presentan similitudes con el trabajo que se plantea en este proyecto.

Aplicación DirecTV Sports

La empresa de televisión para abonados *DirecTV* ofrece a sus clientes una serie de aplicaciones móviles (para *Android*, *iOS*, *Kindle*) que permite visualizar transmisiones en vivo de algunas de sus señales. Para este relevamiento se utilizó la aplicación para *Android DIRECTV Sports* [DIR]. Como se observa en las siguientes capturas de pantalla (figuras: 2.1, 2.2, 2.3 y 2.4), cuenta con una línea de tiempo que muestra hitos ocurridos durante el evento, únicamente disponible para transmisiones en vivo. Cada uno de estos hitos tiene asociada una descripción y el reproductor permite al usuario

ir al momento de tiempo donde ocurrió cada evento, al seleccionar un lugar de la línea de tiempo donde hay varios eventos, se despliega un detalle de cada evento cercano. Si se selecciona un hito, en la parte derecha del menú de navegación el botón verde permite volver a la transmisión en vivo.

Figura 2.1: DirecTV Sports, lista de eventos en vivo.

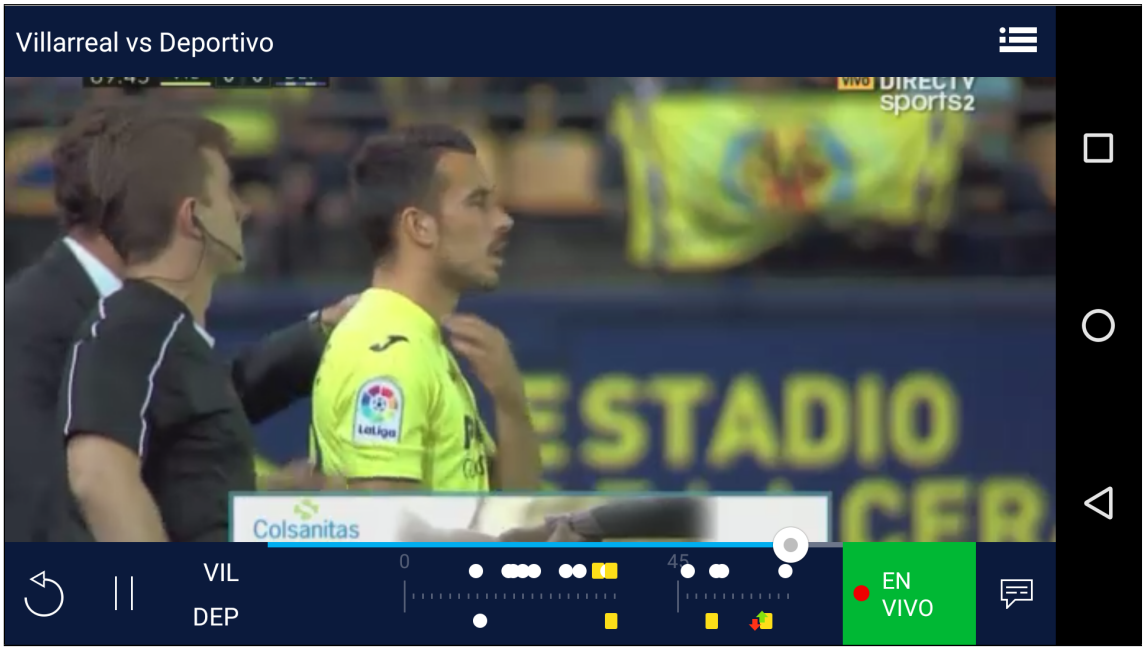


Figura 2.2: DirecTV Sports, transmisión en vivo con hitos en línea de tiempo.

Figura 2.3: DirecTV Sports, detalle de hitos.

Figura 2.4: DirecTV Sports, visualizando hito anterior.

Si bien esta funcionalidad presenta la limitante de estar disponible únicamente para transmisiones en vivo de *DirecTV*, la aplicación en sí es muy similar a uno de los componentes que se plantea construir en el presente proyecto.

Skip To

Existe una extensión para el navegador *Google Chrome/Chromium*, llamada *SkipTo* [SKIb], que le permite al usuario agregar información sobre dónde se encuentran las mejores partes de un video en *YouTube*. La extensión agrega dos botones sobre el reproductor de *YouTube*, uno para definir una nueva marca y otro para avanzar hasta la siguiente marca definida. La extensión permite saltar hasta la próxima marca en el video, ya sea ésta agregada por el usuario que está visualizando el video o por otros usuarios que hayan agregado previamente sus marcas. La visualización de las marcas de otros usuarios puede activarse mediante una opción de la extensión.

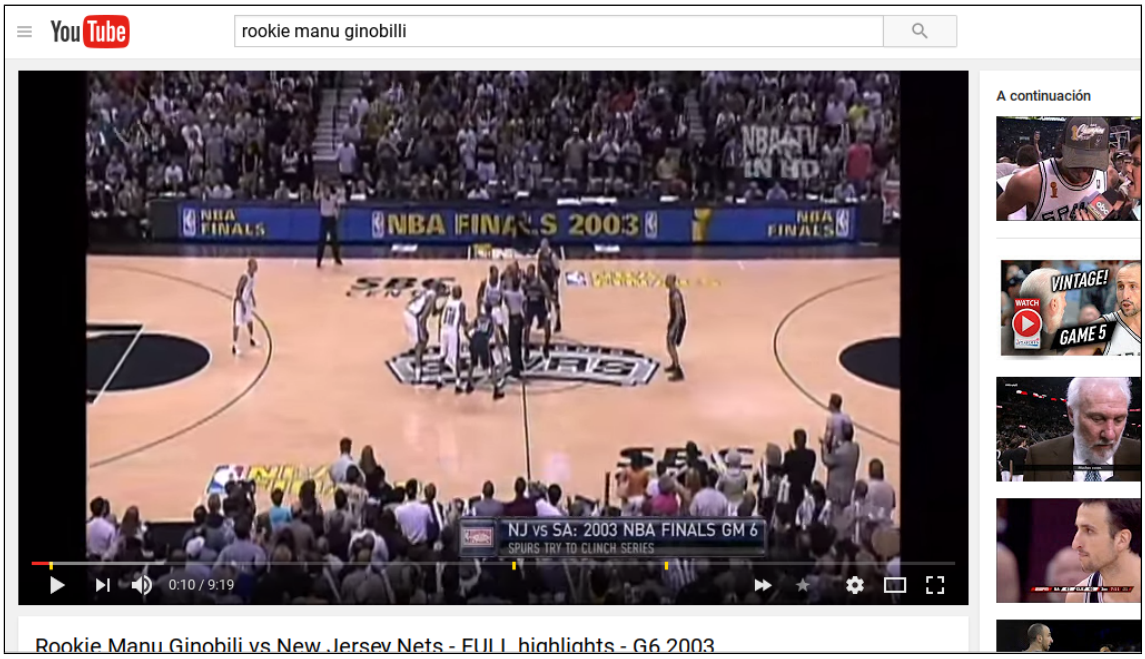


Figura 2.5: Extensión SkipTo con video con tres marcas.

En las pruebas realizadas durante la etapa de relevamiento, se observó que solo se permite agregar una marca por video y por usuario.

Esta implementación presenta las limitantes de ser una extensión de un navegador, solo funciona con videos de un único proveedor, no permite agregar información junto con la marca de tiempo y actualmente solo admite una marca por usuario en cada video. De todas formas, puede tomarse como referencia para la implementación del visualizdor planteado, [SKIa] [SKIc].

The Mad Video

The Mad Video [MADb] intentó ser una plataforma que tenía como objetivo agregar información adicional a los videos en línea (*YouTube*, *Vimeo*, etc; a partir de su *URL*), para convertirlos en videos interactivos. Las principales aplicaciones señaladas por los fundadores del proyecto son el comercio electrónico, educación y divulgación [MADa].

La idea de este emprendimiento era etiquetar información de distinto tipo dentro de un video: lugares, personas, objetos; así como integrar la herramienta con redes sociales y televisiones inteligentes.

Actualmente no es posible acceder a la página web del proyecto (www.themadvideo.com) y parece no tener actividad desde 2014 [MADd], [MADc].

Panopto

Panopto ([PANa], [PANb]) es una plataforma que permite grabar clases, conferencias, etc; grabar emisiones en pantalla, realizar *video streaming* y gestionar contenido en video. Fue fundada por docentes de la Universidad Carnegie Mellon, es utilizada principalmente con fines educativos y muy conocida como plataforma de educación a distancia.

Consiste en dos componentes, que permiten grabar y emitir en vivo presentaciones, demos, charlas y demás. Este material audiovisual es almacenado en el sistema y mediante interfaces web se pueden editar estos videos, buscar en el contenido e integrarlos con otras herramientas.

Al ser una herramienta comercial pensada para crear y gestionar el contenido audiovisual en centros educativos, está orientada a tabajar con contenido local.

La página web del proyecto nombra varias funcionalidades y tecnologías utilizadas, como por ejemplo el uso de ASR (del inglés *Automatic Speech Recognition*) que permite capturar e indexar cada palabra pronunciada por cualquier orador en un video. También dice utilizar reconocimiento OCR (del inglés *Optical Character Recognition*) para capturar e indexar las palabras que aparecen "impresas" en pantalla. Todas estas palabras capturadas desde los videos, además de las notas que los usuarios pueden tomar digitalmente mientras ven la grabación, son utilizadas para realizar las búsquedas.

La web del proyecto *Panopto* hace énfasis en que no es necesario buscar en la línea de tiempo de un video para encontrar un contenido específico, es posible utilizar el motor de búsqueda para ir directamente al momento donde se trata determinado tema.

Además, esta web hace referencia a varias entradas de *blog* donde se habla sobre el sistema de búsqueda inteligente de *Panopto*, sin embargo al estar todas estas entradas publicadas en la propia página de *Panopto* y carecer de autoría, no se considera relevante incluir un análisis más profundo de su contenido en este capítulo.

Microsoft Video Indexer

En setiembre de 2016 se lanzó *Video Breakdown* como un proyecto de *Microsoft Garage* [MS-d], de acuerdo al sitio del proyecto se incorporó la experiencia de los usuarios para re-lanzar el proyecto con el nombre de *Video Indexer* [MS-e], como un servicio de *Microsoft Cognitive Service* [MS-c].

Los objetivos de este proyecto son:

- hacer que el contenido sea más fácil de descubrir: mediante la extracción de metadatos visuales y del discurso se mejora la búsqueda en las aplicaciones existentes (por ejemplo buscando

rostros de personas conocidas, emociones, palabras pronunciadas en un video, etc)

- mejorar las recomendaciones a los usuarios utilizando los metadatos extraídos de los videos
- ofrecer un conjunto de servicios que son consumidos mediante APIs y widgets (componentes web), sin necesidad de conocer la infraestructura subyacente

Video Indexer es una plataforma cognitiva de indexado de videos, los videos son subidos por los usuarios a la plataforma y ésta los procesa para crear un índice cognitivo del contenido. Reconoce audio y texto, detecta pistas de audio e identifica rostros e interlocutores, extrae palabras clave y tópicos, analiza sentimientos, clasifica actividad de voces, etc. Sobre esta plataforma, los usuarios pueden buscar, explorar, editar y reproducir videos y listas de reproducción. Un objetivo de esta plataforma es que estas acciones puedan hacerse de forma simple, potente e inteligente.

El servicio cuenta con una API de alto nivel que permite:

subir contenido: subir videos a la plataforma mediante una URL. Se comienza con el procesamiento cuando el video es subido. Se utilizan varias tecnologías de Inteligencia Artificial para extraer ideas a través de mútiples dimensiones (habla, rostros, texto, objetos, etc).

descargar ideas: una vez que el video fue procesado, se puede descargar el conocimiento extraído en forma de archivos JSON

búsquedas: se puede enviar consultas de búsqueda para encontrar momentos relevantes dentro de un video o a través de todos los videos en una cuenta de *Video Indexer*

reproductor de video: se puede obtener un reproductor para un video, que puede ser embebido en cualquier aplicación web

control de ideas: al igual que el reproductor, se puede obtener un control (*widget*) de las ideas extraídas de un video para embeber en aplicaciones web, es posible elegir qué partes de este control se muestran

Según la documentación consultada en [MS-a], la plataforma incluye algunas tecnologías como las descriptas a continuación, que son aplicadas a cada video que es subido a *Video Indexer*:

transcripción de audio: ofrece una transcripción de las conversaciones de un video, por el momento soporta los idiomas: inglés, español, francés, alemán, italiano, chino (simplificado), portugés, japonés y ruso. Esta funcionalidad de "audio a texto" está basada en el mismo motor usado por *Cortana* y *Skype*

seguimiento e identificación de rostros: tecnología para detectar rostros en videos. Los rostros detectados son comparados con una base de datos de celebridades para evaluar qué personas famosas aparecen en cada video. Los usuarios también pueden etiquetar caras que no pertenezcan a una celebridad. *Video Indexer* construye un modelo basado en estas etiquetas y puede reconocer esos rostros en videos subidos en el futuro.

indexación del orador: *Video Indexer* tiene la habilidad de asociar y entender cuál orador pronuncia cada palabra y cuándo

reconocimiento de texto: esta tecnología permite extraer el texto que es mostrado en un video

También brinda funcionalidades como: detección de actividad vocal, detección de escenas, extracción de cuadros clave, análisis de sentimientos, traducción, moderación del contenido visual, extracción de palabras clave y permite crear anotaciones basándose en una base de objetos predefinidos.

A los efectos de este relevamiento, se realizaron pruebas en la *web* de *Video Indexer Preview* [MS-b]. Esta interfaz permite subir archivos de video, pero no agregarlos mediante una URL desde plataformas como *YouTube*. En los videos con poca calidad (resolución baja, poca luz, ruido ambiente, etc) no se detectaron conversaciones ni rostros. Sí se detectó texto sobre impreso en el video, pero no siempre de forma correcta ni en el orden adecuado.

En las figuras 2.6 y 2.7 se puede ver la pantalla inicial de la web, donde se muestran los videos subidos por el usuario y algunos videos de ejemplo, respectivamente.

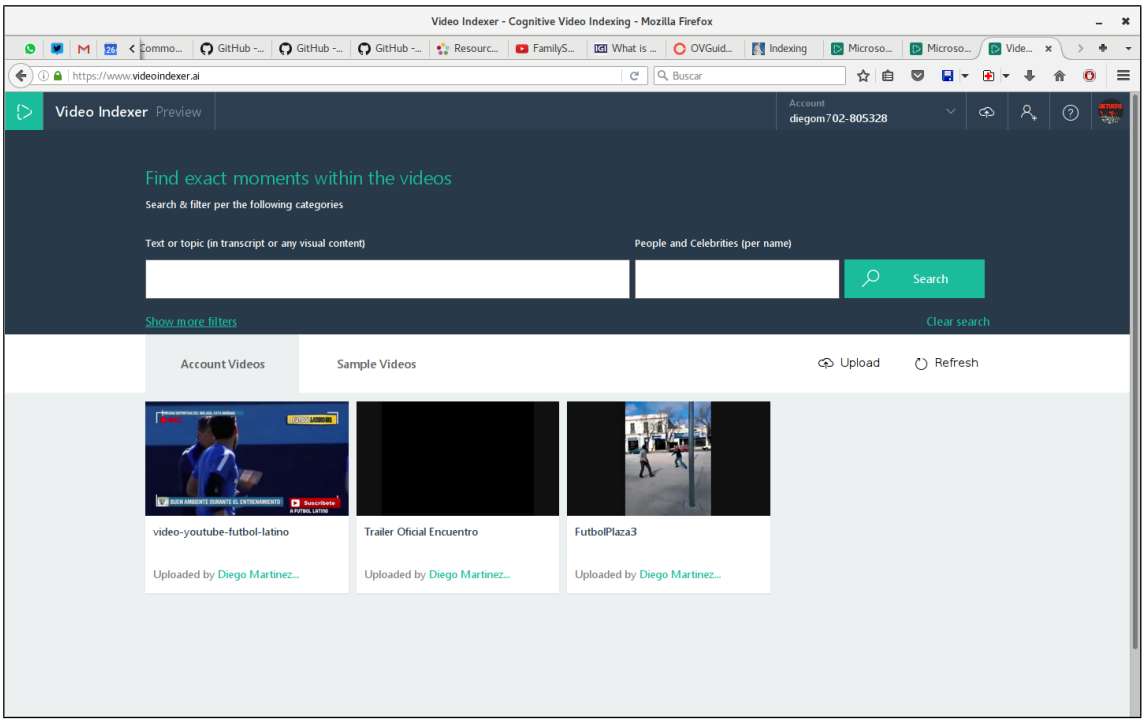


Figura 2.6: Video Indexer: pantalla inicial.

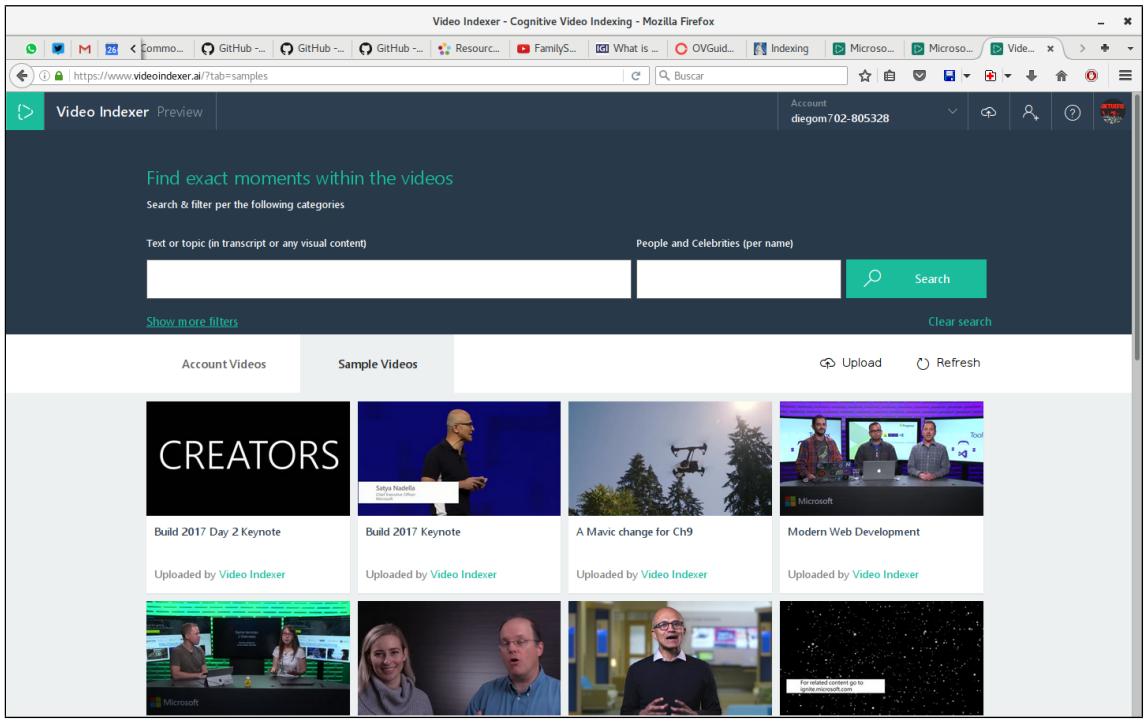


Figura 2.7: Video Indexer: videos de ejemplo.

En las siguientes figuras se utilizó un video de la colección de ejemplos. Se observa como a la derecha de la pantalla aparece la transcripción de audio a texto, siendo posible modificar el idioma 2.8.

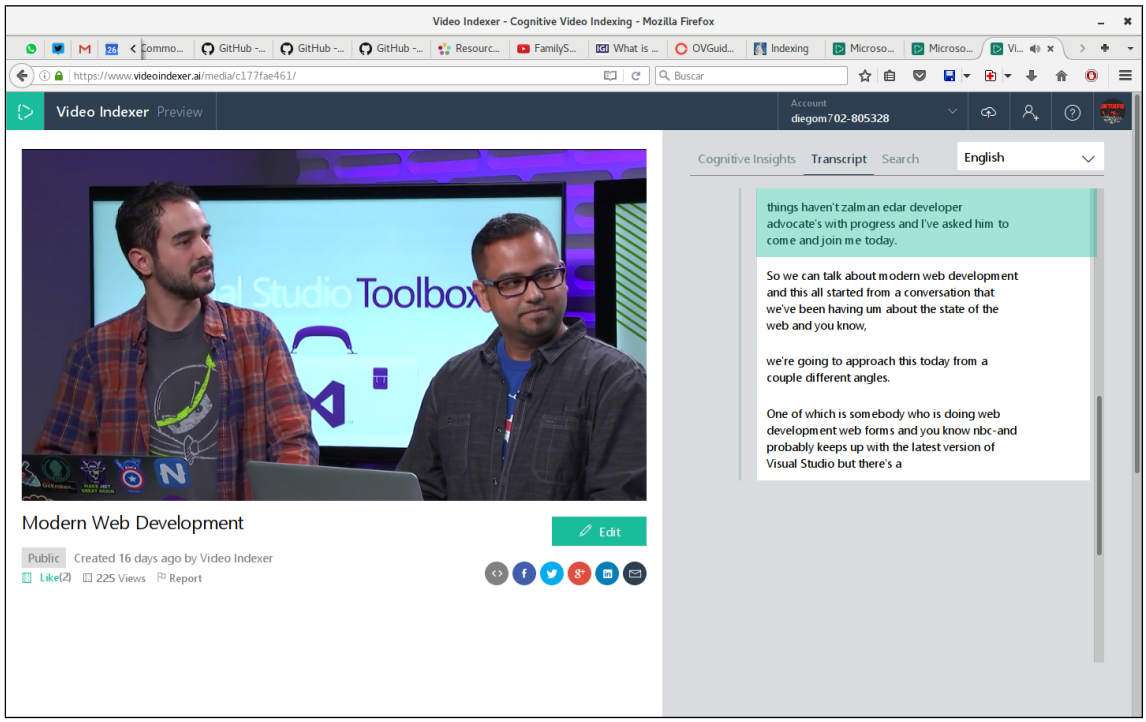


Figura 2.8: Video Indexer: transcripción del audio.

En la figura 2.9 se observa que se reconoció a 3 personas, se muestra un resumen de la línea de tiempo con las intervenciones de cada uno, las palabras claves que pronuncia a lo largo del video y un análisis de las emociones detectadas durante su intervención (neutro, positivo, negativo).

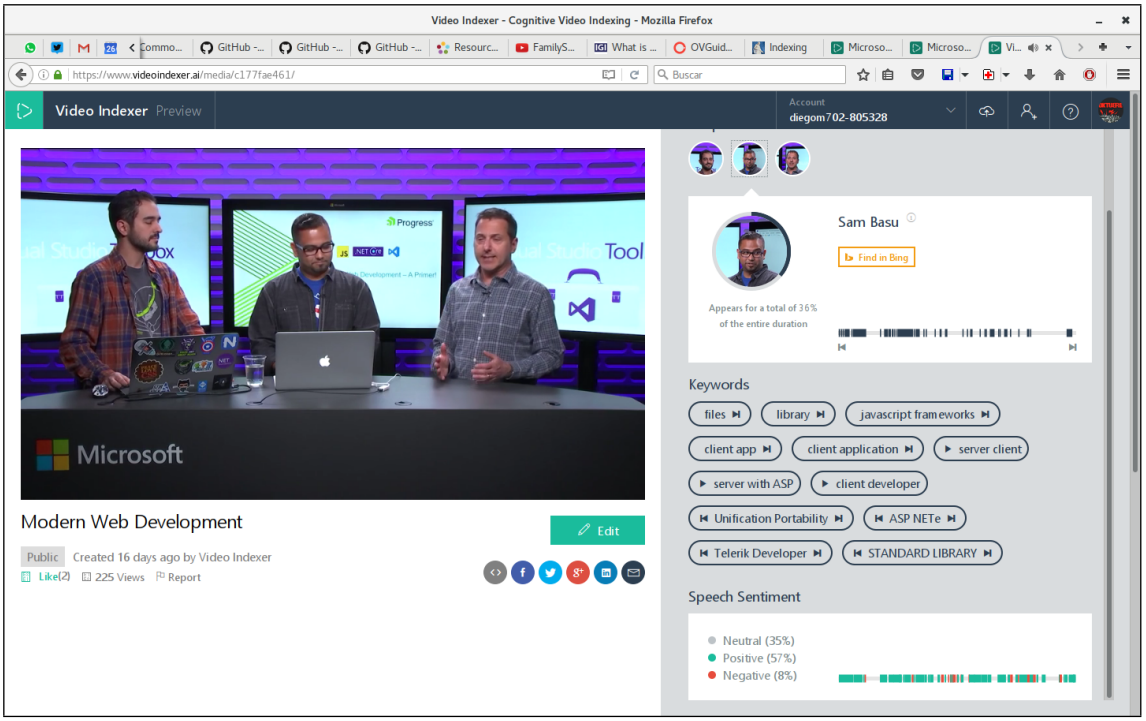


Figura 2.9: Video Indexer: análsis del contenido, palabras claves, emociones, etc.

Por último, en la figura 2.10 se utiliza la transcripción para navegar por el video.

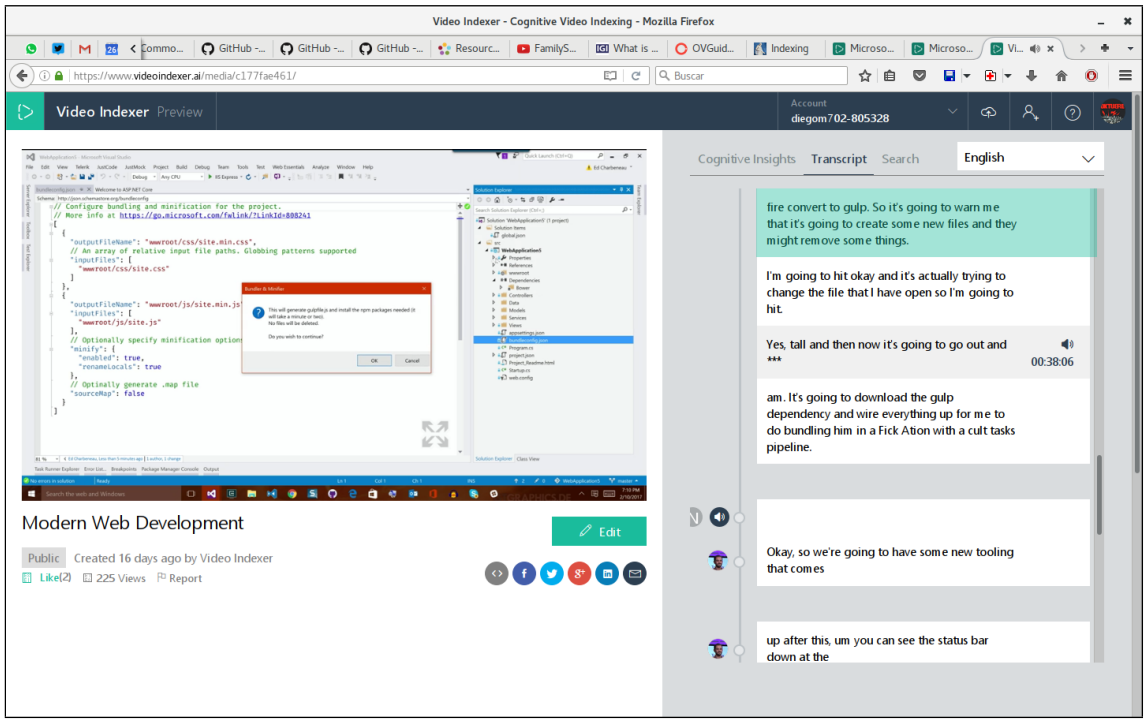


Figura 2.10: Video Indexer: navegando mediante la transcripción.

Si bien la aplicación *Video Indexer* se basa en servicios de la plataforma cognitiva de *Microsoft Azure*, que escapa al alcance de este relevamiento; cabe destacar que el portal *Video Indexer Preview* presenta una interfaz sencilla pero potente, donde se aplica la idea de utilizar un índice para visualizar un video.

Esta aplicación web puede usarse como referencia para el desarrollo del portal web planteado en nuestro proyecto. También puede resultar útil analizar más detenidamente la plataforma de servicios cognitivos utilizada, para poder aprovechar la extracción automática de metadatos de un video con el objetivo de generar un índice a partir de estos metadatos.

Google Cloud Video Intelligence

La empresa *Google* cuenta con un servicio similar al de *Microsoft* para analizar y extraer contenido desde videos, [GL-b]. Este proyecto también se plantea como una solución para hacer que los videos sean "buscables" y más fáciles de descubrir mediante la extracción de metadatos. Estos servicios son ofrecidos a través de una API.

En la página principal del proyecto se puede probar el servicio, suministrando un enlace a un video previamente cargado a la plataforma *Google Cloud Storage* (GCS) o eligiendo un video de ejemplo. En esta "prueba rápida" se puede ver los objetos reconocidos, así como el resultado "crudo" de las operaciones ofrecidas en la API.

La principal utilidad que intenta ofrecer este servicio, es poder buscar en un catálogo de videos como si se estuviese buscando en documentos de texto. Para realizar esta tarea, el servicio

elimina el "ruido", detecta etiquetas, cambios de escena, etc. Al momento de consultar la web y documentación oficial de este proyecto [GL-a], se encuentra en etapa "Beta" y se anuncian nuevas funcionalidades para el futuro.

Este proyecto de *Google* parece tener un objetivo similar al de *Microsoft Video Indexer*, sin embargo por el momento parece estar en una etapa más temprana de desarrollo, la documentación es más escueta y no cuenta con una aplicación como *Microsoft Video Indexer Preview*. De todas formas, se considera relevante conocer este servicio para eventualmente utilizarlo en este proyecto de grado.

Brieftube

Brieftube es una extensión de *Google Chrome/Chromium* [BRI] que consiste en extraer información a partir de los subtítulos del video para así generar un índice y buscar en ellos.

La extensión consiste de 3 pestañas que muestra la siguiente información:

- índice por capítulos generados automáticamente (figura 2.11)
- buscador por palabras encontradas en los subtítulos (figura 2.12)
- nube de palabras con las palabras más usadas (figura 2.13)

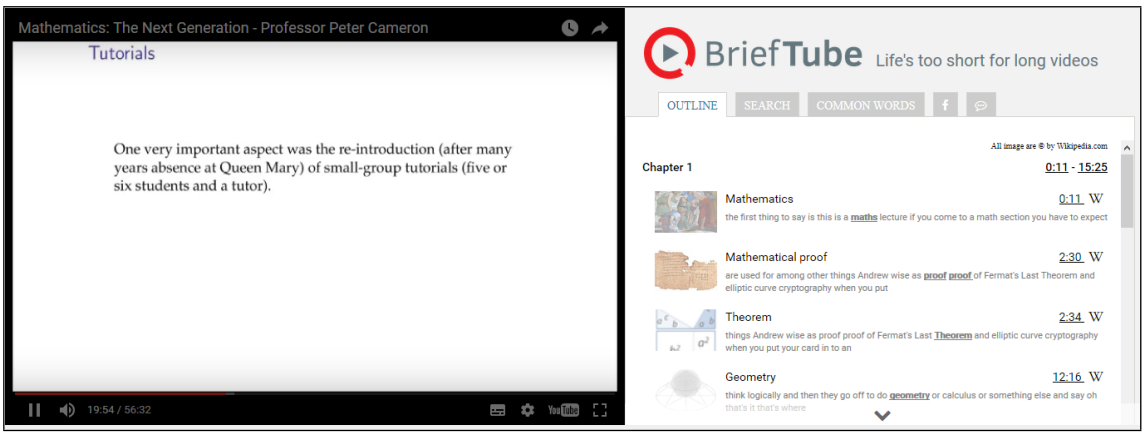


Figura 2.11: Brieftube: índice autogenerado.

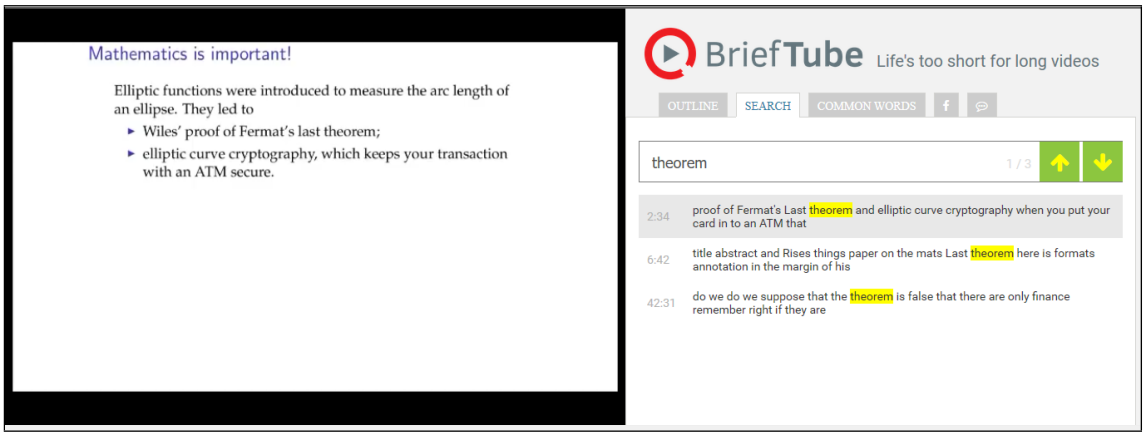


Figura 2.12: Brieftube: busqueda dentro de los subtitulos.

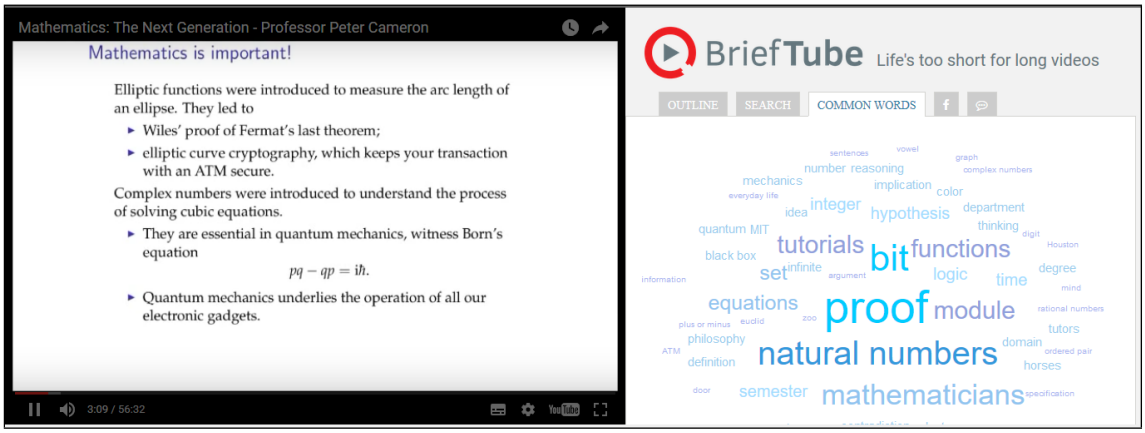


Figura 2.13: Brieftube: nube de palabras.

Este servicio se restringe a videos de *YouTube* con subtítulos en inglés de más de 5 minutos de duración y que si bien es un complemento de pago cuenta con un período de prueba de una semana.

Clipmine

Clipmine [CLIa] es una plataforma que se autodenomina como "la primera plataforma de indexado y anotaciones de video" [CLIb].

El funcionamiento es el de importar videos de las plataformas *YouTube*, *Vimeo* y *Dailymotion*, mediante un editor se le agregan las anotaciones pertinentes (figuras 2.14 y 2.15). El video con las anotaciones se puede embeber en una página web. Se cuenta con un portal en donde se pueden reproducir los videos a los cuales se les agregó las anotaciones (figura 2.16).

Cuenta con una parte de anotación inteligente para la ayuda del indexado traduciendo información visual (objectos, textos, logos, rostros, escenas, etc) en metadatos [CLIc].

A la fecha el sitio [CLId] esta caído y no se encuentra información sobre el estado de la plataforma en Internet. Nos pusimos en contacto con el CEO de la empresa, Zia Syel via Facebook y nos respondió que la empresa fue adquirida y pronto va a ser relanzada.

Figura 2.14: Clipmine, editor de video.

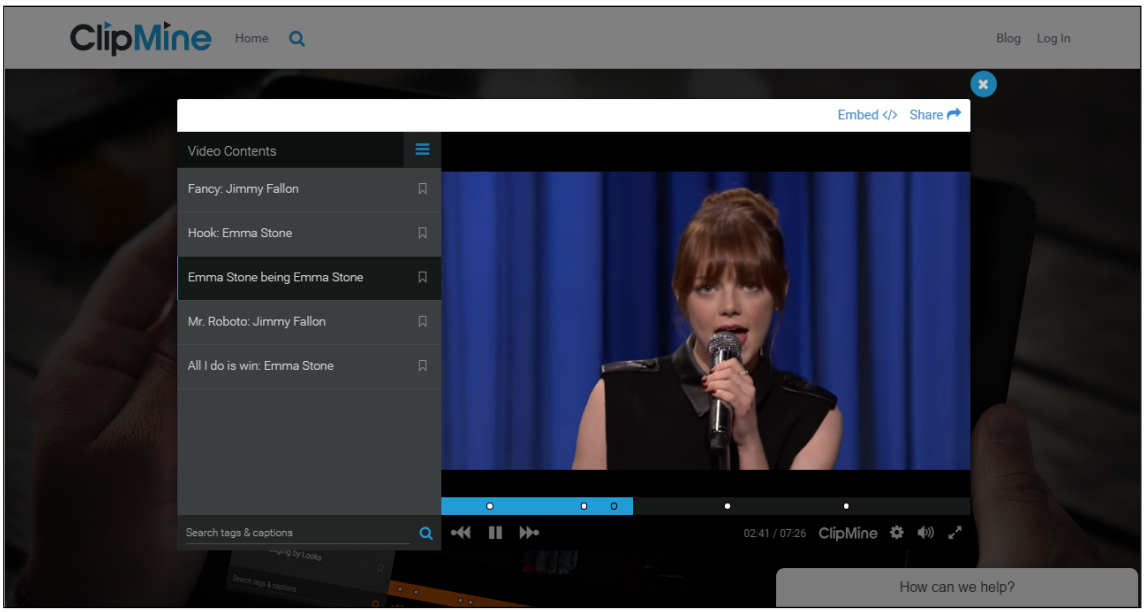


Figura 2.15: Clipmine, editor de video.

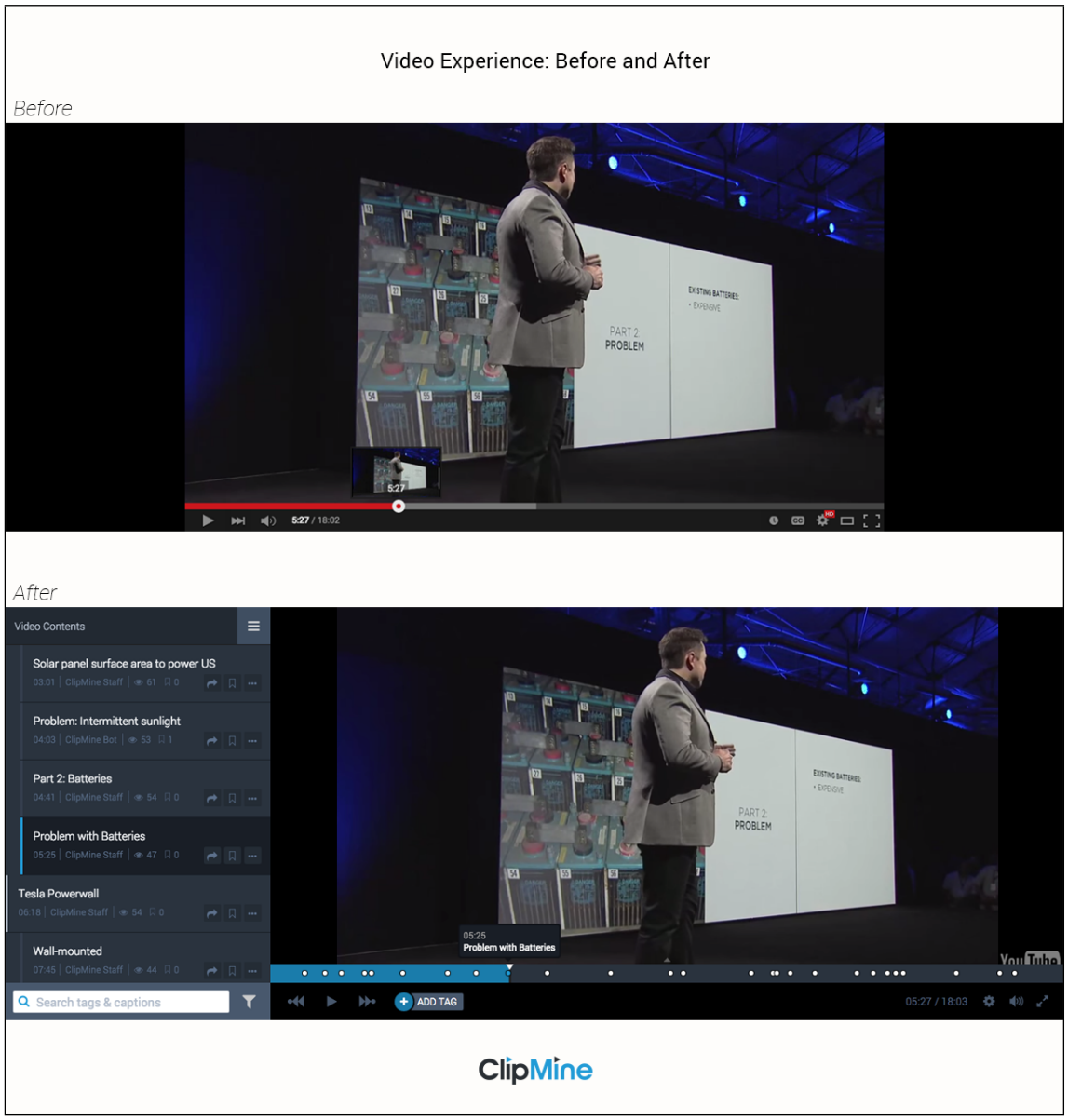


Figura 2.16: Clipmine, reproductor de video.

KZO Innovations

KZO [KZO] es una robusta plataforma colaborativa de gestión de contenido de videos empresariales, fundada en 2007, con el propósito de ser ofrecida como servicios y permitirle a los usuarios conectarse con una *Suite* de aplicaciones propias a través de Internet con un simple navegador. Esta herramienta ofrece soluciones a la hora de realizar búsquedas de contenidos por palabras claves, gracias a la creación y anotación de videos, que le ofrece a los espectadores la posibilidad de acceder y navegar rápidamente al momento exacto de lo que están buscando. La lista de videos puede ser filtrada por título, subtítulos, hilos de discusión formulados entre los usuarios, etc.

Con la ayuda de la herramienta Flash y de HTML5, los videos *on line* pueden ser reproducidos desde cualquier dispositivo móvil o de escritorio, y adiciona la propiedad de codificación automática

del video a todos los formatos y velocidades de reproducción.

Por otra parte, permite a los usuarios realizar grabaciones en tiempo real desde sus cámaras web o dispositivos, grabaciones de voz, crear capturas de pantalla, agregar documentos o diapositivas sincronizadas al video, entre otras funcionalidades. Este contenido es grabado directamente en la *Suite* de video de KZO y puede ser publicado poco después para su visualización.

Dispone de una fácil indexación de contenido añadiendo marcadores de capítulos y además realiza una sincronización automática de las conversaciones que se van produciendo entre los usuarios en tiempo real, indexando el video con estas discusiones.

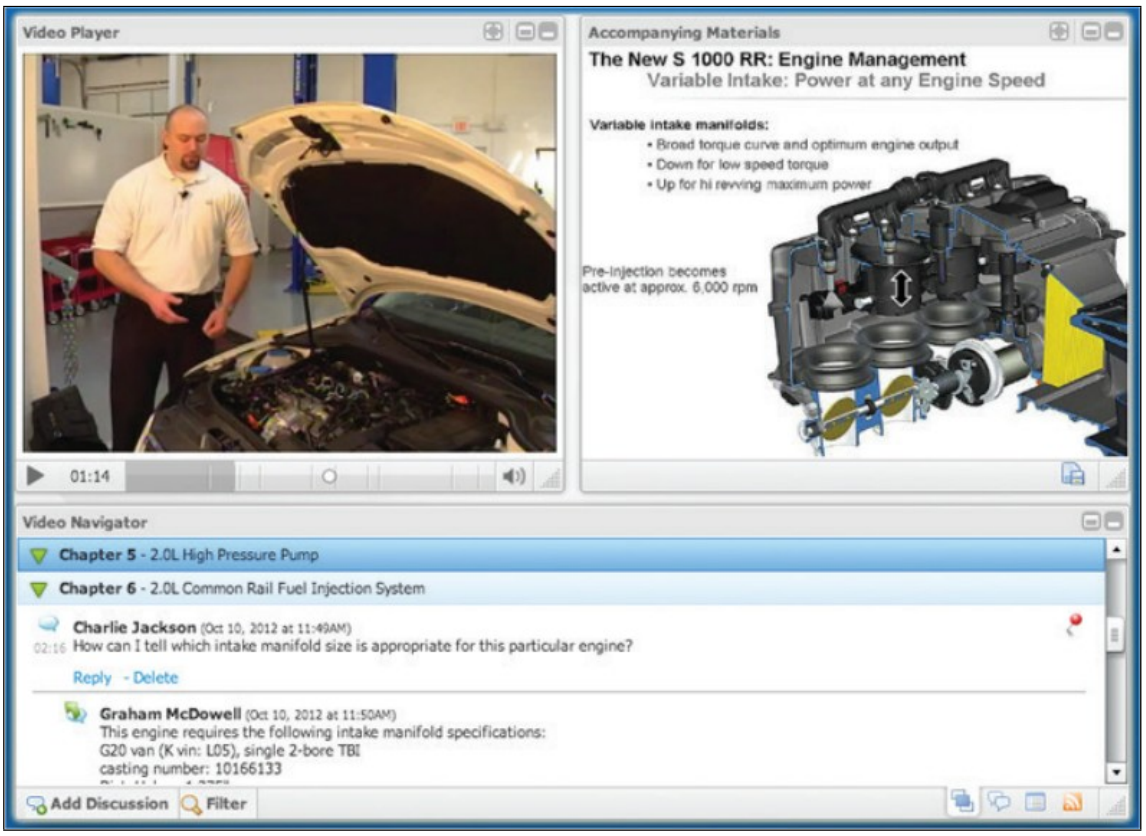


Figura 2.17: KZO Video Suite

Timestamp en descripción de videos de YouTube

El proveedor de servicios *YouTube* no cuenta actualmente con indexación de instantes de tiempo con momentos ocurridos durante un video. Sí dispone de métodos de indexación para posicionamiento en los buscadores del portal, pero no para gestionar el contenido del mismo. Analizando el sitio, y navegando entre diferentes tipos de video, se descubre la posibilidad de definir en la descripción del video, instantes de tiempo con una etiqueta como se ve en la imagen 2.18.

Figura 2.18: Timestamp en descripción de YouTube

Con este tipo de descripción, el usuario que accede a ver el video en cuestión, puede navegar a los diferentes hitos marcados como *timestamp* con el formato *hh:mm:ss* brindándole un estilo de indexación. Suele utilizarse en videos de larga duración como por ejemplo el concierto completo de una banda, donde cada marca corresponde al inicio de una canción.

También es comúnmente utilizado por los jugadores de videojuegos que suelen grabar las partidas completas y marcar acciones específicas ocurridas durante el juego para trasmitirlas a los demás jugadores.

Muchos de los usuarios de *YouTube* que hacen uso de este tipo de anotaciones, las utilizan para que al ser buscados por palabras claves en el portal, puedan ser localizados con mayor facilidad posicionándolos mejor en el ranking de videos disponibles.

De cualquier modo, esta "indexación" creada con la ayuda de la descripción de los videos, no es una funcionalidad propia del portal del proveedor de videos *YouTube*, los *tags* no son interpretables, es decir que si un usuario desea describir el video conteniendo el formato utilizado para anotar el *timestamp* pero sin ese propósito, se definiría un *link* a un instante del video que quizás ni siquiera exista. A su vez, tiene la desventaja de que solo es posible realizar este marcado de *timestamp* por el propietario del video, no así por cualquier usuario que quiera facilitar esta información.

2.2. Herramientas para videos gestionados localmente

Con el objetivo de realizar un relevamiento completo sobre tecnologías relacionadas con lo planteado en el proyecto, se realizó una revisión de herramientas similares para gestionar contenido local (no en línea). Un ámbito dónde se puede encontrar este tipo de herramientas es en canales

de televisión, dónde se manejan grandes volúmenes de datos en formato video. A continuación se presentan los resultados encontrados.

2.2.1. Gestión en televisión

La digitalización de contenidos audiovisuales ha servido de puntapié inicial para la gestión de elementos de video no solo en el ámbito *on line*, o distribuidos en Internet, sino también en aquellos videos que se encuentran almacenados localmente por diferentes organismos con un propósito definido.

En particular, los centros de producción de medios digitales, como lo son los canales de televisión, manejan diariamente un gran volumen de información en formato de video que sin una buena organización de la misma, termina siendo un contenido digital irrelevante o inexistente, al que simplemente no se puede acceder o reutilizar. Las diferentes emisoras de televisión manejan contenido de grabaciones periodísticas para producir un noticiero, recepción de señales nacionales o internacionales de otras emisoras para su posterior análisis y/o edición, grabación completa de las transmisiones, producciones y postproducciones, etc. Asimismo, cada vez son más las emisoras que deciden impulsar sus plataformas *on line* facilitándole al espectador la reproducción del material audiovisual transmitido, donde quiera que esté.

Por estos motivos, decidimos ampliar esta rama de investigación a los medios de comunicación con la idea de conocer cuál es la realidad actual de este sector empresarial con un alto contenido gráfico que podría brindar grandes aportes al proyecto. Concretamente, resultó interesante investigar cómo es el manejo de los contenidos multimedia en la televisión uruguaya, y más específicamente cómo se realiza la catalogación de dichos elementos para su posterior búsqueda y utilización. No parece ser tarea fácil buscar en un archivo de televisión de horas de grabación, alguna sección en particular cuando no se conoce exactamente el momento en el que ocurrió el evento buscado y no se cuenta con etiquetas que faciliten la tarea.

Se agendó una entrevista con el encargado del área técnica de la empresa *S.A.E.T.A. TV Canal 10* quien comentó a grandes rasgos cómo es el manejo de archivos realizado por el canal.
Parte de los programas de televisión, noticieros y archivos de prensa, y programación en general que trabajan en el canal, se puede decir que son catalogados como empresas separadas, por lo que actualmente cada una de ellas cuenta con su propio sistema de gestión. Sí poseen una grabación de transmisión centralizada y digitalizada que luego es consultada por veedores para su utilización. Esto les permite tener un filtrado de *frames* por día y por rangos de horas y minutos de todo lo difundido en ese lapso de tiempo, con 5 días de anterioridad. Si bien esto les es relativamente útil, no tiene ninguna descripción asociada que les permita por ejemplo realizar una búsqueda por nombre o evento; además de que deben apelar a la memoria de lo ocurrido en determinado momento.

Se conversó sobre una investigación realizada años atrás por la empresa donde conocieron la herramienta de producción *Adobe Premiere*, que entre otras cosas, permite sincronizar un guión con un elemento gráfico de video a través del reconocimiento automático de palabras o *speech*. Esta investigación no llegó a concluirse, y si bien les pareció una herramienta muy potente no es utilizada por el canal en la actualidad. El encargado mencionó la importancia y potencia de este tipo de herramientas, pero se considera que es necesario invertir demasiado dinero y tiempo para investigarlas, desarrollarlas e implementarlas.

De todos modos, el canal cuenta con un *player* que permite tener catalogado los archivos a partir de un etiquetado de *tiempo-descripción*, lo cual tiene alguna semejanza con el propósito del proyecto, pero tampoco es utilizado por el canal debido a que no disponen de dicha fuente de entrada o personal que realice esta tarea.

A partir de la propuesta de este proyecto, referente a la indexación de contenidos de videos *on line*, y con la introducción de la TV en vivo y la interacción del usuario con el contenido *on line*, la empresa ve un producto muy interesante pero considera que la idea les sería redituable si la misma infiere el mínimo esfuerzo humano posible, llevando la automatización a su mayor potencial.

Si se mueve el foco de investigación al ámbito internacional, existen actualmente algunos sistemas de información denominados MAM (del inglés *Media Asset Management*) que facilitan la tarea de gestión de contenidos y que sí son utilizados por las grandes emisoras de televisión.

A continuación se detalla una de estas herramientas que cuenta con un alto potencial:

Interplay de Avid

El producto *Interplay* [INT], es un administrador de recursos perteneciente a *Avid* que forma parte de una familia de productos. Esta herramienta se compone de un poderoso administrador de *assets* (recursos gráficos) y herramientas productivas para media *on line* y *nearline*. Ofrece aplicaciones nativas como *Interplay Central* que a través de una interfaz web en la nube permite a los miembros del equipo, entre otras cosas, buscar y reproducir videos, colocar marcadores, almacenar y localizar todos los activos en varios sistemas remotos o locales, con la particularidad de realizarlo desde cualquier dispositivo y desde cualquier parte del mundo solo con un navegador web.

Comenzó siendo un producto bastante complejo, de gran tamaño y sustancialmente caro que no cumplía con las expectativas de sus consumidores, pero *Avid* logró realizar grandes procesos con *Interplay* por ejemplo en noticias y deportes que mejoró su productividad, agilidad y eficiencia. De todos modos, este producto está destinado al ambiente empresarial como pueden ser las cadenas de televisión, las empresas de *marketing*, profesionales de cine, audio y video, *broadcast*, etc. Logra una gran flexibilidad de metadatos personalizados que permiten realizar búsquedas de contenidos específicos de audio o video de una forma más fácil y automatizar otros procesos complejos.

Dentro de los sistemas y opciones para organizar, hacer el seguimiento y maximizar el valor de los recursos gráficos de *Interplay* se pueden encontrar *Interplay - MAM*, *Interplay - Production*, *MediaCentral*, *Media - Index*, entre otros. En la figura 2.19 se puede ver la apariencia de uno de estos productos.

Figura 2.19: Interplay - MAM

2.3. Resumen

A modo de síntesis de esta sección dedicada a la investigación de antecedentes en el área, se puede decir que si bien existe en el mercado un gran número de productos similares al que se pretende desarrollar, ninguna herramienta logra cumplir con todas las expectativas con las que partimos. No podemos afirmar cuál es la explicación de este fenómeno, o por qué un proveedor tan importante como *YouTube* no cuenta aún con índices para optimizar el contenido de los videos. Probablemente, entre otros motivos, influyan las costumbres de los usuarios que frecuentan la visualización de videos directamente desde el portal de los proveedores (*YouTube*, *Vimeo*, etc) o través de las redes sociales donde se comparten y esto conspira contra proyectos como *Clipmine*. También se puede especular con motivos relacionados a intereses comerciales, considerando que los proveedores normalmente quieren vender publicidad junto con el contenido o fomentar la visualización de material relacionado con lo que un usuario busca.

Sí se puede afirmar que sin una manera de buscar en los videos el contenido deseado y más aún si los videos son largos, los espectadores pueden sentirse frustrados por la dificultad de no encontrar lo que están buscando o ni siquiera saber si el contenido existe en dicho video. Por lo tanto contar con herramientas que permitan tener ordenado los metadatos de los recursos gráficos y poder realizar posteriores búsquedas de contenido parece ser una puerta bastante clara para la satisfacción en la experiencia de los usuarios.

Capítulo 3

Análisis de requerimientos

Con el fin de comprender mejor el problema y dejar claro lo que se necesita implementar, se comienza con una etapa de análisis donde se define conjuntamente con el tutor cúal será el alcance del proyecto, se realiza un relevamiento de los requerimientos para el sistema, tanto desde un punto de vista formal como con la elaboración de bosquejos. Y por último se definen las prioridades de las funcionalidades a la hora de realizar la implementación, así como las restricciones del mismo.

3.1. Alcance del proyecto

Luego de la exhaustiva investigación de herramientas se puede definir el alcance del proyecto, detallando lo que se va a llevar a cabo en el prototipo de software y cuáles son sus restricciones.

El objetivo principal de este proyecto es proporcionar una herramienta práctica e intuitiva que básicamente permita gestionar el contenido de videos. Para ello se pretende desarrollar una versión de reproductor de videos que le permita a los usuarios cargar uno o varios videos y especificar marcas en el transcurso de tiempo de los mismos de modo que se logre crear una indexación del contenido incluido en los videos. Estas marcas manuales denominadas hitos, pueden ser navegadas por el reproductor, simplificando el avance en la visualización del video para acceder al momento exacto de tiempo deseado. El componente a desarrollar, debe permitir generar hitos automáticos, es decir no generados manualmente de a uno por el usuario sino proporcionados por alguna interfaz que brinde estos datos, que si bien por lo general implica un número importante de hitos autogenerados, los mismos aportan mucho valor mayormente para futuras búsquedas. Entre ellos se encuentra la generación de hitos a partir de subtítulos que ofrecen proveedores como *Youtube* y de reconocimiento automático provisto por la API del proyecto *Video++2*.

El indexador y reproductor se comportarán como componentes similares, con la diferencia que el primero incluye al segundo (con las funcionalidades de visualización), y el indexador adiciona las funcionalidades de gestión de hitos agregando, borrando y reordenando las marcas dispersas en la línea de tiempo de los videos.

Como parte de la implementación de la solución, se incluye un portal web que gestiona y visualiza el listado de índices que los usuarios van generando en el sistema organizados según categorías. Los usuarios autenticados podrá consultar sus índices además de modificarlos o crear nuevos índices, mientras que los usuarios invitados solo tendrán la posibilidad de consultar aquellos índices disponibles. Además, ambos tipos de usuarios podrán buscar hitos relacionados entre sí

pertenecientes a múltiples índices según algún criterio de búqueda seleccionado.
Los índices consultados en el portal contendrán un embebido con el indexador o reproductor de índices con la información del mismo, como ser: hitos y videos contenidos, información básica donde se encuentra la fecha de creación y modificación, cantidad de visualizaciones, categorías del índice, usuario propietario, etc.

Conjuntamente, y para llegar a aquellos usuarios que hoy en día utilizan asiduamente los dispositivos inteligentes, la solución incluirá una versión del portal optimizada para visualizar y utilizar el indexador/reproductor en dispositivos móviles, con la idea de que los usuarios puedan instalar la aplicación en sus dispositivos y puedan interactuar con el sistema de forma transparente para ellos. Inicialmente, la aplicación móvil será desarrollada para su completo funcionamiento en el sistema operativo *Android*, con algunas restricciones en *iOS*, como la imposibilidad de visualizar videos en modo pantalla completa.

Por otra parte, se restringe a seleccionar videos a partir de URL's (tanto con la dirección completa, como solo ingresando el identificador de video del proveedor) y aún no estará disponible la carga de archivos locales a *Youtube*.

Como parte del alcance, se definirán pruebas relevantes que garanticen el correcto funcionamiento del prototipo y verifiquen la completitud de las funcionalidades descriptas.

3.2. Bosquejos

Como parte del análisis, se propone plasmar en bosquejos las diferentes funcionalidades con sus componentes y acciones que ayudan a visualizar mejor el problema a resolver. Para el diseño de estos bosquejos nos basamos fuertemente en los productos similares relevados, principalmente en la aplicación *DIRECTV Sports* [DIR] y en *Clipmine* [CLIa].

La visualización que ofrecen los bosquejos también facilita la posterior toma de decisión con respecto a las tecnologías a ser aplicadas, seleccionando aquellas que cumplen con los requisitos allí diagramados.

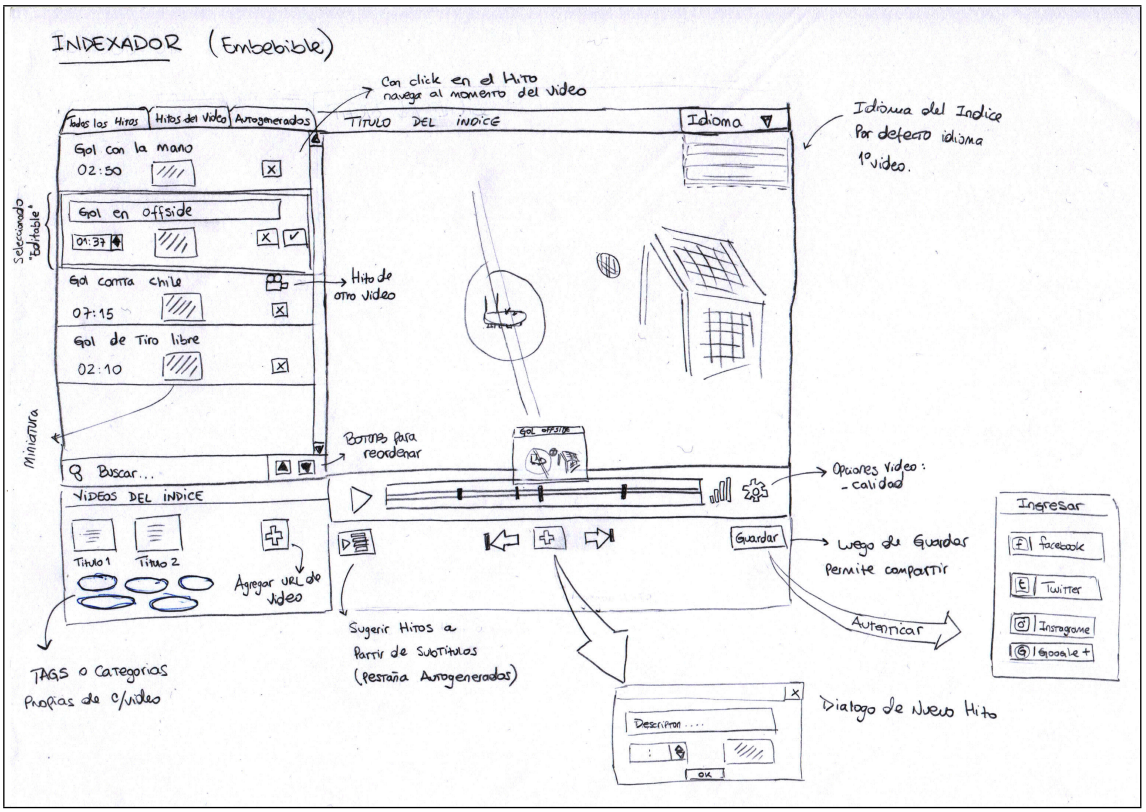


Figura 3.1: Indexador

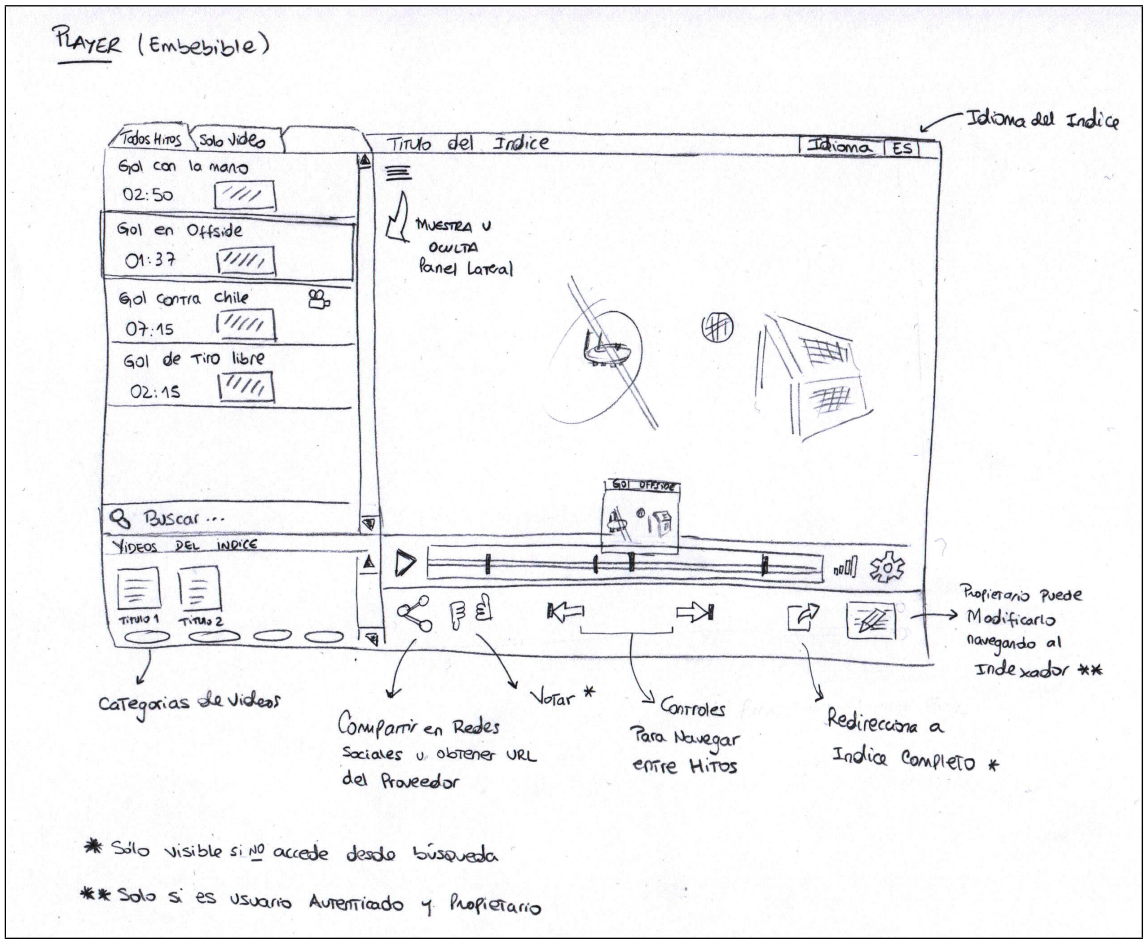


Figura 3.2: Reproductor

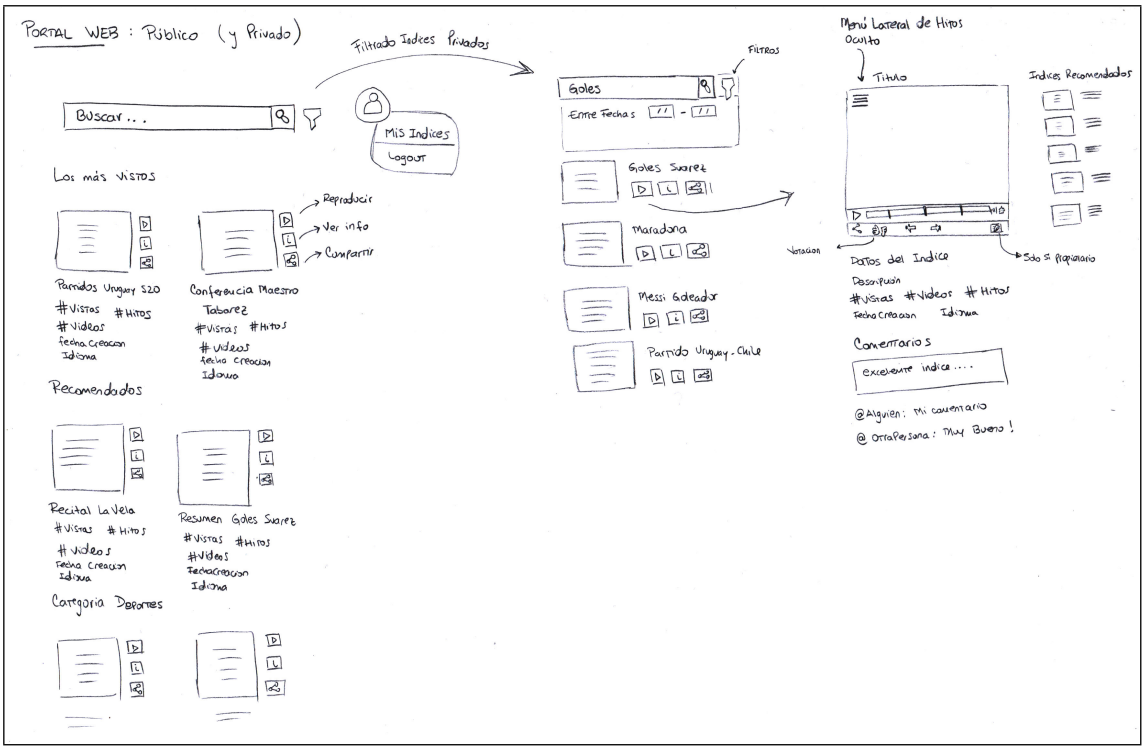


Figura 3.3: Portal Web Público

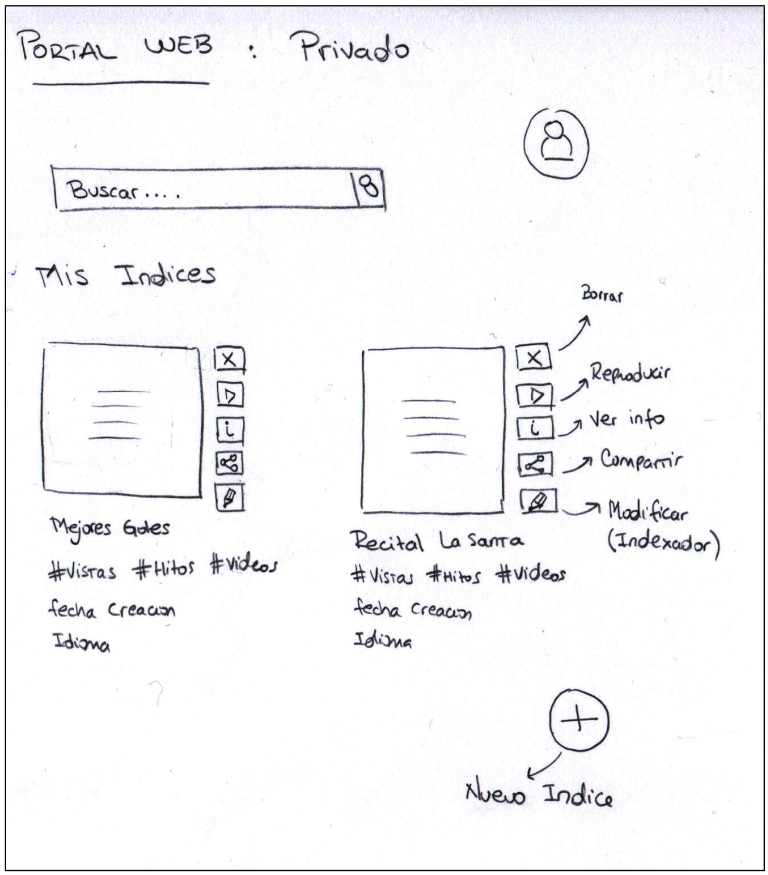


Figura 3.4: Portal Web Privado

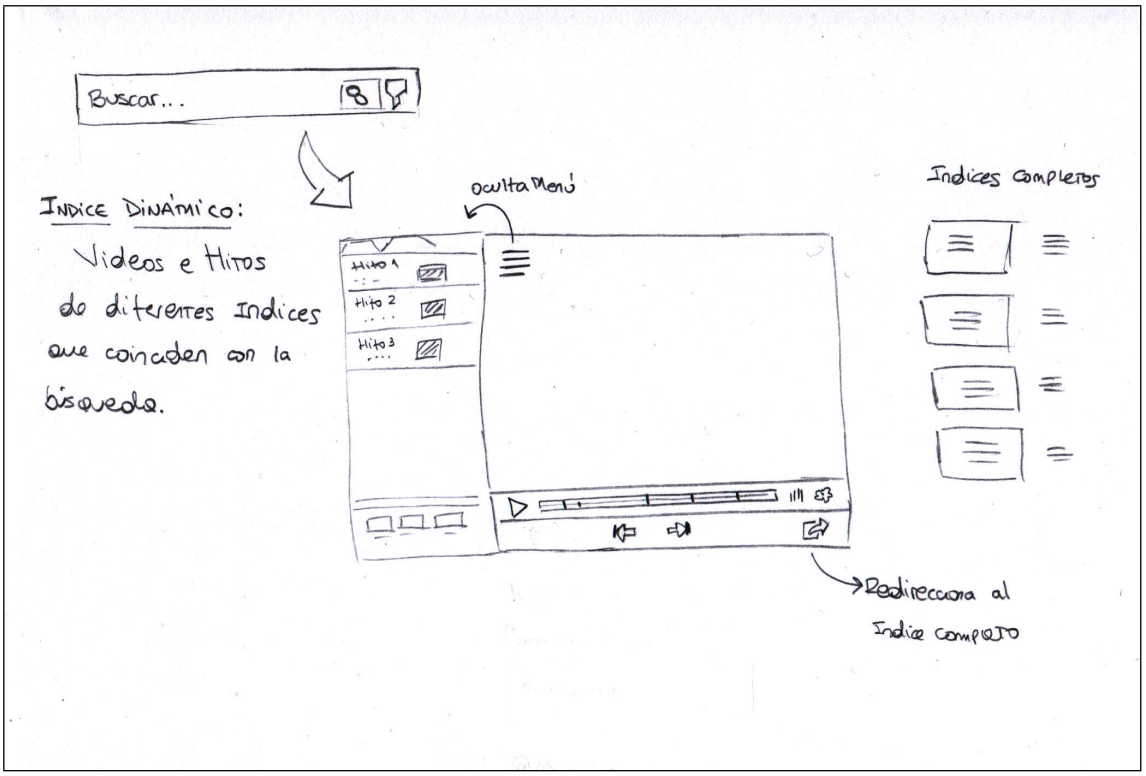


Figura 3.5: Búsquedas dinámicas

3.3. Requerimientos

A continuación se especifican los requerimientos funcionales y no funcionales que deberá cumplir la solución planteada.

Requerimientos funcionales

Para la especificacion de requerimientos funcionales se utiliza el formato indicado en la norma IEEE 830 (*SRS – Software requirements specification*).

Identificación del requerimiento:	RF01
Nombre del requerimiento:	Autenticación de Usuarios.
Actor:	Usuario Invitado
Características:	Los usuarios deberán identificarse para acceder a las funcionalidades privadas y gestionar su actividad en el sistema. Los usuarios invitados podrán permanecer en el sistema sin estar autenticados, pero solo visualizarán aquellas funcionalidades que se encuentren públicas.
Descripción del requerimiento:	El sistema gestionará la autenticación de usuarios únicamente a través de redes sociales (Facebook, Twitter, Google+)
Prioridad del requerimiento: Alta	

Identificación del requerimiento:	RF02
Nombre del requerimiento:	Consultar índices públicos
Actor:	Usuario Invitado / Usuario autenticado.
Características:	El sistema ofrecerá al usuario un listado de índices existentes en el sistema, ordenados según ciertos criterios.
Descripción del requerimiento:	El listado de índices se realizará según criterios específicos como “Los más vistos”, “Índices recomendados”, “Categorías”, etc. De cada uno se muestra: Título, cantidad de vistas, cantidad de hitos definidos, cantidad de videos del índice. Además ofrece las funcionalidades de: ▪ Iniciar sesión ▪ Reproducir el índice ▪ Ver información más detallada ▪ Compartir el índice
Prioridad del requerimiento: Alta	

Identificación del requerimiento:	RF03
Nombre del requerimiento:	Consultar índices privados
Actor:	Usuario autenticado.
Características:	El sistema ofrecerá al usuario logueado, la posibilidad de consultar los índices previamente creados por él.
Descripción del requerimiento:	En la sección "Mis Índices" del perfil del usuario logueado, se listarán los índices creados por ese usuario, mostrando Título, cantidad de vistas, cantidad de hitos, cantidad de videos del índice y le permitirá acceder a las siguientes funcionalidades para cada índice: ▪ Borrar índice ▪ Reproducir índice ▪ Ver información detallada ▪ Compartir índice ▪ Modificar índice Además, el sistema le ofrece al usuario crear un nuevo índice. La lista de índices se podrá filtrar por palabras contenidas en su título o descripción.
Prioridad del requerimiento: Alta	

Identificación del requerimiento:	RF04
Nombre del requerimiento:	Reproducir índices públicos.
Actor:	Usuario invitado. Usuario autenticado.
Características:	El sistema ofrecerá al usuario visualizar el índice que seleccionó permitiendo elegir los videos del índice y navegar en la línea de tiempo, así como ver la lista de hitos y navegar a través de los mismos
Descripción del requerimiento:	El sistema ofrecerá al usuario visualizar el índice que seleccionó. Para ello se despliega el reproductor embebido en el portal web, mostrando la lista de videos que contiene el mismo, la lista total de hitos del índice y la lista de hitos del video seleccionado. Permite navegar en la línea de tiempo del video, así como navegar por hito. De cada hito se muestra la descripción del mismo y el tiempo (timestamp) en formato hh:mm:ss. Si el hito no corresponde al video que se encuentra reproduciendo, se agrega un ícono que identifica que el mismo pertenece a otro video, y si se selecciona se reproduce el video correspondiente. Al seleccionar un hito en la línea de tiempo, se previsualiza una miniatura (*thumbnail*) del instante de tiempo al que corresponde en el video además de la descripción del hito. El sistema ofrece un buscador de hitos dentro del índice buscando palabras contenidas en la descripción de los hitos. Además, el sistema permite: ▪ ver información básica del índice, ▪ compartir el índice u obtener la URL del mismo ▪ ver y realizar comentarios ▪ puntuar el índice según nivel de satisfacción ▪ ver lista de índices similares recomendados para el usuario Cada vez que un usuario reproduce un índice, el sistema registra la visualización para contabilizar la cantidad de vistas del índice.
Prioridad del requerimiento: Alta	

Identificación del requerimiento:	RF05
Nombre del requerimiento:	Reproducir índices privados.
Actor:	Usuario autenticado
Características:	El sistema ofrecerá al usuario autenticado visualizar el índice propio seleccionado, permitiendo elegir los videos del índice y navegar en la línea de tiempo, así como ver la lista de hitos y navegar a través de los mismos. Además permite modificar el índice
Descripción del requerimiento:	El sistema ofrecerá al usuario autenticado visualizar el índice propio que seleccionó. Para ello se despliega el reproductor embebido en el portal web, mostrando la lista de videos que contiene el mismo, la lista total de hitos del índice y la lista de hitos del video seleccionado. Permite navegar en la línea de tiempo del video, así como navegar por hito. De cada hito se muestra la descripción del mismo y el tiempo (*thumbnail*) en formato hh:mm:ss. Si el hito no corresponde al video que se encuentra reproduciendo, se agrega un ícono que identifica que el mismo pertenece a otro video, y si se selecciona se reproduce el video correspondiente. Si se selecciona un hito en la línea de tiempo, se previsualiza una miniatura (thumbnail) del instante de tiempo al que corresponde en el video además de la descripción del hito. El sistema ofrece un buscador de hitos dentro del índice buscando palabras contenidas en la descripción de los hitos. Además, el sistema permite: ▪ ver información básica del índice, ▪ compartir el índice u obtener la URL del mismo ▪ modificar índice ▪ ver y realizar comentarios ▪ puntuar el índice según nivel de satisfacción ▪ ver lista de índices similares recomendados para el usuario
Prioridad del requerimiento: Alta	

Identificación del requerimiento:	RF06
Nombre del requerimiento:	Modificar índice
Actor:	Usuario autenticado
Características:	El sistema permite modificar un índice.
Descripción del requerimiento:	Por un lado, permite modificar los datos básicos. Entre ellos permite modificar: título, descripción, tags. Luego permite modificar el índice, navegando al Indexador. Allí muestra la lista de videos (con la opción de quitar o agregar videos a través de la URL), brindándole al usuario la posibilidad de elegir uno y navegar en la línea de tiempo para agregar hitos manualmente. Además, ofrece la funcionalidad de generar hitos automáticamente a partir del componente seleccionado. Los componentes disponibles de hitos automáticos son: "subtitulos del video" ó "API *Video++2*". Estos hitos se agregan a la pestaña de "autogenerados" y son almacenados por el sistema con una etiqueta o tag específico para su identificación en un posterior uso de búsquedas o filtrados. Para cada nuevo hito manual, el sistema muestra un diálogo con un formulario donde debe ingresar los datos del nuevo hito. Entre ellos incluye descripción, permite ajustar el timestamp del mismo y se le asigna la miniatura del video del instante seleccionado. La lista de los hitos se encuentra ordenada por defecto en el orden en que fueron ingresados, mostrando los datos de cada uno junto con la miniatura que le corresponde. Los mismos pueden ser seleccionados para ser eliminados o reordenados. Si se selecciona un hito, se navega al video en el instante de tiempo al que corresponde. Si el hito seleccionado es un hito de otro video (marcado con ícono especial), se navega al tiempo correspondiente de dicho video. El sistema agrega la posibilidad de realizar búsquedas de hitos por texto incluido en la descripción del mismo, en cada una de las pestañas (tanto en el listado de hitos totales del índice, lista de hitos del video seleccionado, ó lista de hitos autogenerados). Los controles de video son play y pausa, volumen y ajustes propios del video (calidad del video, entre otros). Los controles del índice son: sugerir hitos automáticos, agregar hito, navegar al hito anterior o al siguiente. Por último permite guardar el índice y persistir los cambios.
Prioridad del requerimiento: Alta	

Identificación del requerimiento:	RF07
Nombre del requerimiento:	Nuevo índice.
Actor:	Usuario Invitado. Usuario Autenticado.
Características:	El sistema permite ingresar un nuevo índice.
Descripción del requerimiento:	El usuario selecciona crear un nuevo índice y el sistema lo redirecciona al Indexador donde podrá ingresar los datos básicos del índice: título, visibilidad, idioma y lista de videos. La visibilidad puede ser: público, privado (a través del link), no listado (solo propietario). El idioma asignado por defecto es el idioma del primer video agregado. Los videos son agregados ingresando una URL y son listados junto con las opciones de quitar o agregar nuevos videos. Por cada video el sistema obtiene su información consultando la interfaz que el proveedor proporciona (API) y muestra la lista de categorías o etiquetas del mismo. El indexador permite seleccionar un video y navegar en la línea de tiempo para agregar hitos manualmente. Además, ofrece la funcionalidad de generar hitos automáticamente a partir del componente seleccionado. Los componentes disponibles de hitos automáticos son: "subtítulos del video" o "API *Video++2*". Estos hitos se agregan a la pestaña de "autogenerados" y son almacenados por el sistema con una etiqueta o tag específico para su identificación en un posterior uso de búsquedas o filtrados. Para cada nuevo hito manual, el sistema muestra un diálogo con un formulario donde debe ingresar los datos del hito. Entre ellos incluye descripción, permite ajustar el timestamp del hito y se le asigna la miniatura del video en el instante seleccionado. La lista de los hitos se encuentra ordenada por defecto en el orden en que fueron ingresados, mostrando los datos de cada uno junto con la miniatura que le corresponde. Los mismos pueden ser seleccionados para ser eliminados o reordenados. Si se selecciona un hito, se navega al instante de tiempo al que corresponde. Si el mismo es un hito de un video diferente al que se encuentra reproduciendo (marcado con ícono especial), se navega al instante de tiempo del video al que corresponde. El sistema agrega la posibilidad de realizar búsquedas de hitos por texto incluido en la descripción del mismo, en cada una de las pestañas (tanto en el listado de hitos totales del índice, lista de hitos del video seleccionado, ó lista de hitos autogenerados). Los controles de video son play y pausa, volumen y ajustes propios del video (calidad del video, entre otros). Los controles del índice son: sugerir hitos automáticos, agregar hito, navegar al hito anterior o al siguiente. Por último permite guardar el índice y persistir los cambios, solicitando autenticación si es un usuario invitado. Una vez guardado, el sistema ofrece la posibilidad de compartir el índice u obtener su URL (esta URL se corresponde con la página que contiene al reproductor).
Prioridad del requerimiento: Alta	

Identificación del requerimiento:	RF08
Nombre del requerimiento:	Búsqueda de hitos
Actor:	Usuario invitado. Usuario autenticado
Características:	Es posible buscar hitos, que se muestran como un nuevo índice dinámico.
Descripción del requerimiento:	El portal web ofrece la posibilidad de buscar hitos en todos los índices públicos (y privados si el usuario está autenticado). La búsqueda se realiza por palabras contenidas en la descripción de un hito. El resultado es un conjunto de hitos que pertenecen a varios índices existentes, este conjunto genera un nuevo índice temporal que es mostrado al usuario. El usuario tiene la posibilidad de persistir el índice que resulta de una búsqueda. Para cada hito resultante de una búsqueda, el sistema ofrece la opción de navegar al índice "original" al que pertenece para visualizar el contexto del índice completo.
Prioridad del requerimiento: Alta	

Identificación del requerimiento:	RF09
Nombre del requerimiento:	Subir vídeo
Actor:	Usuario de móvil autenticado.
Características:	Es posible subir un video local desde la aplicación a un proveedor.
Descripción del requerimiento:	La aplicación móvil permite subir videos locales del dispositivo móvil a un proveedor de video. Se completan los datos mínimos que pide el proveedor para subir el video. Luego de subirlo permite agregar dicho video a un índice existente o crear un índice nuevo.
Prioridad del requerimiento: Baja	

Identificación del requerimiento:	RF10
Nombre del requerimiento:	Notificaciones tipo push para aplicación móvil.
Actor:	Usuario de móvil autenticado.
Características:	El sistema envía notificaciones a la aplicación móvil.
Descripción del requerimiento:	La aplicación móvil permite que el usuario se suscriba a los índices de cualquier otro usuario para ser notificado cuando ocurra algún evento relacionado con ese índice. Las notificaciones son de tipo push, por lo tanto es el portal web quien envía las notificaciones a los usuarios a través de la aplicación móvil.
Prioridad del requerimiento: Media	

Requerimientos no funcionales

A continuación se describen los principales requisitos no funcionales para el sistema.

RNF01 - Interfaz del sistema: El sistema debe tener una interfaz de uso intuitiva y sencilla.
Prioridad: alta.

RNF02 - Ayuda en el uso del sistema: La interfaz de usuario deberá presentar un sistema de ayuda para facilitar el uso del sistema al usuario. La interfaz debe estar complementada con un buen sistema de ayuda.
Prioridad: baja.

RNF03 - Desempeño: El sistema garantizará a los usuarios un óptimo desempeño en cuanto a los datos almacenados en el sistema ofreciéndole una alta confiabilidad en el mismo. La información almacenada o registros realizados podrán ser consultados y actualizados permanente y simultáneamente, sin que se afecte el tiempo de respuesta.
Prioridad: media.

RNF04 - Nivel de autorización de usuario: El sistema garantizará al usuario el acceso de información y funcionalidades de acuerdo al nivel que posee. Si un usuario no se encuentra logueado (invitado), no puede realizar acciones que necesiten identificación, por ejemplo: agregar índices o modificarlos.
Prioridad: media.

RNF05 - Seguridad de la información: El sistema garantizará a los usuarios seguridad en cuanto a la información que se ingresa en el sistema, por ejemplo las credenciales para autenticación de los usuarios.
Prioridad: baja.

RNF06 - Múltiples navegadores: El sistema garantizará el correcto funcionamiento del portal web en los navegadores modernos. Se debe soportar al menos Mozilla Firefox en su versión 51 (o superior), Google Chrome en su versión 58 (o superior) e Internet Explorer (IE) en su versión 10.0 (o superior).
Prioridad: media.

RNF07 - Comunicación con proveedores: El sistema garantizará la correcta comunicación con los proveedores de videos on line. Se soportará al menos el proveedor *Youtube*.
Prioridad: alta.

RNF08 - Modularidad y escalabilidad: Modularidad en el diseño de interfaces de comunicación con proveedores de video. El sistema debe garantizar un diseño modular para poder incorporar nuevos componentes o interfaces de comunicación con otros proveedores de video, de modo que sea escalable.
Prioridad: alta.

Capítulo 4

Arquitectura y diseño

En este capítulo se describe la solución planteada desde el punto de vista del diseño.

Se comienza definiendo el modelo de datos y la arquitectura que tendrá la solución a partir de los requerimientos principales. Se describen los principales componentes del sistema y la interacción entre ellos, especificando las operaciones que surgen de los requerimientos en un nivel alto de abstracción, dejando para el siguiente capítulo las consideraciones de implementación. Este capítulo pretende dar un panorama general sobre la solución desarrollada.

4.1. Modelo de datos

Luego de especificar de manera formal los requerimientos del sistema y relevar herramientas existentes, planteamos la definición del modelo de datos desde un enfoque tradicional, siguiendo las principales pautas del desarrollo de un sistema de información.

Se analizaron los casos de uso y los bosquejos con el objetivo de determinar las principales entidades y cómo se relacionan entre ellas, a partir de esto se plantea el modelo relacional que se desarrolla con más detalle en el Apéndice A de este informe.

Durante el proceso de diseño y desarrollo de la solución, el modelo de datos sufrió modificaciones desde el punto de vista conceptual, adaptándose a nuevos requerimientos y cambios en funcionalidades. Luego de la elección de tecnologías, decidimos implementar la base de datos utilizando *mongoDB* [MONa] debido a que es una herramienta muy utilizada en sistemas web en los cuales el rendimiento y la capacidad de manejar grandes volúmenes de datos son aspectos fundamentales. Esta forma de representar los datos además de evitar el problema conocido como *Impedance Mismatch*, en nuestro caso permite implementar una solución del tipo *full stack JavaScript* ya que *mongoDB* está muy asociado a tecnologías basadas en *JavaScript* [JS] por representar los datos como documentos *JSON*.

En este momento nos enfrentamos al cambio de paradigma entre el modelo relacional y una herramienta pensada para representar bases de datos no relacionales. Teniendo en cuenta el contexto académico de este trabajo y que era de interés del grupo manejar estas herramientas como parte del proceso de aprendizaje, consultamos el material disponible en el curso de la asignatura Bases de Datos No Relacionales (BDNR) [BDN]: "clase 6" [doc] y *"Next Generation Databases (JSON Databases, pág. 58)"* [Har15].

Si bien, a priori no es lo más recomendable implementar un modelo relacional utilizando una base de datos de tipo *JSON* -aunque es una alternativa posible-, obtuvimos un resultado satisfactorio ya que el producto funciona correctamente, con tiempos de respuesta adecuados y sin la necesidad de utilizar técnicas o herramientas no estandarizadas para el acceso a los datos. Al mismo tiempo, pudimos llevar a cabo la experiencia de utilizar este tipo de tecnologías con las que no teníamos ninguna práctica ni académica ni laboral.

4.2. Componentes

La propuesta planteada en el proyecto define los siguientes componentes: **Indexador/Reproductor**, **Portal** y **Aplicación Móvil**.
Partiendo de esta definición como requerimiento inicial para la arquitectura de la solución, se decidió utilizar una arquitectura en capas. Por un lado, implementando un conjunto de servicios web (**API**) que brinde las funcionalidades de acceso a datos uniformemente para los distintos componentes, y por el otro construyendo un Portal, un Indexador/Reproductor y una Aplicación Móvil que cumplen las funciones de presentación de datos e interacción con el usuario utilizando los servicios ofrecidos por la API.

Se decidió mantener una base de datos centralizada, a la que se accede mediante un conjunto de operaciones que implementan las reglas de negocio básicas. Este tipo de arquitectura en capas tiene como ventaja definir un punto único de acceso a los datos, lo que simplifica la gestión de la integridad y seguridad de los mismos, así como también evita que se duplique código en distintos componentes.

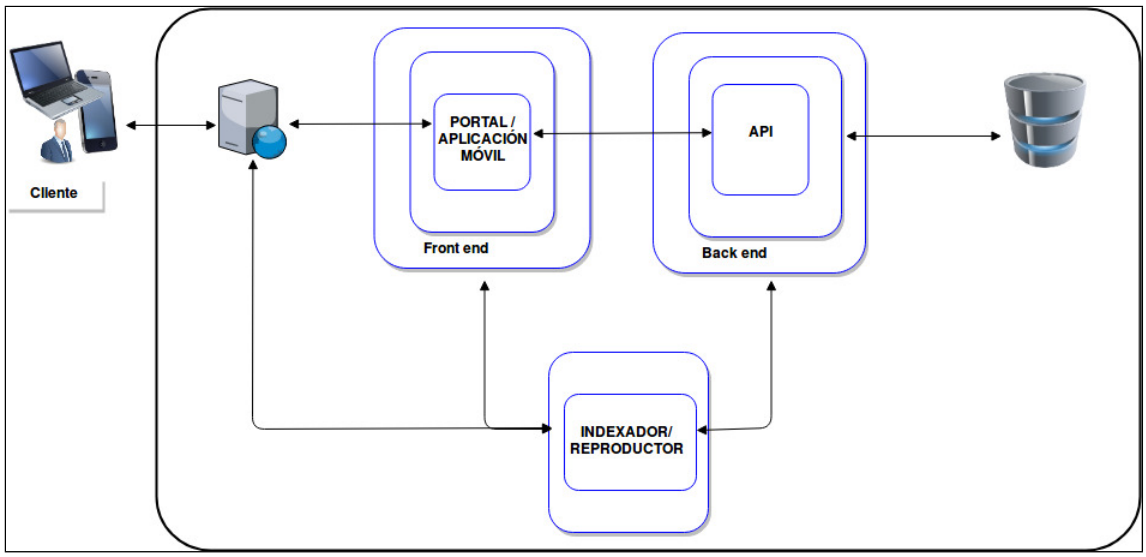


Figura 4.1: Arquitectura.

4.2.1. API

Se definió como una API de tipo REST ya que el objetivo es obtener un servicio simple, eficiente, confiable y relativamente sencillo de escalar.

Otro objetivo importante para el diseño de la API es que debe permitir buena interoperabilidad, es decir que pueda ser utilizado por distintos tipos de clientes, eventualmente implementados con diferentes tecnologías.

Operaciones de la API

Las operaciones definidas en la API reflejan las funcionalidades que surjen de los requerimientos del sistema:

- Listar todos los **índices**.
- Listar **índices** con algún criterio: más vistos, nuevos en la semana, etc.
- Obtener un **índice** a partir de su identificador.
- Obtener el **índice** al que pertenece un determinado hito.
- Obtener un **índice** y todos sus videos o hitos.
- Obtener todos los **índices** creados por un usuario.
- Obtener todos los **índices** a los que está suscripto un usuario.
- Crear un nuevo **índice**.
- Crear un **índice** a partir del resultado de una búsqueda de hitos (índice dinámico).
- Modificar atributos del **índice**, por ejemplo actualizar la cantidad de reproducciones.
- Modificar **índice**: agregar o quitar videos o hitos.
- Obtener un **video** a partir de su identificador.
- Buscar **videos** a partir de su título o descripción.
- Obtener datos de un **usuario** a partir de su identificador.
- Actualizar datos de un **usuario**.
- Buscar **hitos** a partir de su descripción.
- Suscribir y desuscribir a un **usuario** a determinado **índice**.
- Enviar información a **Video++2** sobre los hitos de un video.
- Obtener desde un **proveedor** automático, hitos para un video determinado.

A continuación se detallan algunas de las operaciones más relevantes:

Buscar hitos a partir de su descripción: Índices Dinámicos

Se puede decir que esta funcionalidad es una de las que más aprovecha la información recopilada por toda la aplicación. A medida que los usuarios van indexando y generando hitos, éstos van nutriendo la base de datos, y permiten devolver resultados cada vez más ricos en contenido.

Cada vez que se realiza una búsqueda de esta índole, se invoca a la operación de la API que obtiene una lista de hitos filtrados por descripción y a partir de esta lista se construye un nuevo índice, al cual se le denomina "dinámico", que va a contener los hitos y los videos pertenecientes a diferentes índices ya creados. El término "dinámico" surge debido a la construcción del índice a partir de los datos que se encuentran almacenados en el sistema en el momento de la búsqueda y va enriqueciéndose a medida que se generan más datos. A su vez, este índice permite consultar el contexto al que pertenece cada hito encontrado en la búsqueda, accediendo al índice original con el que está asociado.

Para visualizar los índices que se generan dinámicamente, se optó por reutilizar la forma en que se visualizan los índices en el componente Indexador. Para ello, se persisten las búsquedas como índices, sin persistir los hitos y los videos ya que los mismos son solamente referenciados con el identificador de los ya creados en el sistema. Además, éstos índices dinámicos son diferenciados de los otros índices para no tenerlos en cuenta a la hora de visualizar los listados de índices en el portal.

En principio, este *overhead* de índices dinámicos persistidos, puede conllevar a una limpieza de datos por un proceso diario o programado, para que borre todos los índices referentes a búsquedas. Pero a la vez, esta información puede llegar a ser útil a la hora de obtener estadísiticas o analizar las búsquedas realizadas frecuentemente por los usuarios.

Suscribir y desuscribir a un usuario a determinado índice

Otra de las funcionalidades más interesantes que plantea el sistema consiste en aprovechar las posibilidades de las tecnolgías móviles y de los sitios webs modernos, para notificar a un usuario cuando ocurre un evento que le puede interesar.

Para incluir este tipo de funcionalidad, se decidió agregar la posibilidad de que los usuarios se suscriban a índices que no les pertececen para ser notificados cuando estos índices son modificados.

El diseño de esta funcionalidad contempla los siguientes requisitos:

- un usuario puede suscribirse a varios índices
- un índice puede tener varios suscriptores
- una vez que un usuario se suscribe (o desuscribe) a un índice, esta suscripción es válida para todos los dispositivos (que soporten la tecnología de suscripción) que utilice el usuario

Para poder llevar a cabo esta funcionalidad teniendo en cuenta las restricciones de implementación, se creó el concepto de "perfil de usuario", mediante el cual se puede determinar si un usuario debe estar suscripto o no a algún índice, esta información es consultada cada vez que el usuario inicia sesión en el portal y permite manejar las suscripciones independientemente del dispositivo que está utilizando. A su vez, cada índice tiene una lista de suscriptores a los que se notifica cuando

el índice es modificado.

Enviar información a Video++2 sobre los hitos de un video

En las etapas finales surgió la necesidad de retroalimentar al proyecto *Video++2*. Ese proyecto es un proveedor de hitos generados automáticamente a partir del análisis que realiza en los videos.

Como forma de mejorar el funcionamiento de sus algoritmos, decidimos agregar la posibilidad de que el usuario indique si los hitos generados automáticamente son adecuados al video. Para esto se agrega en los hitos que fueron generados por *Video++2* un botón donde el usuario puede indicar que dicho hito "le gusta". Se usó el ícono *thumbs up* (pulgar hacia arriba).

Otros datos que pueden resultar útiles para el proyecto *Video++2* es la información que los usuarios agregan a los videos en forma de hitos ingresados manualmente. Por este motivo, consideramos en conjunto que al momento de solicitar el procesamiento automático de hitos para un video, se podría enviar la información sobre los hitos que se agregaron de forma manual para un video. Este tipo de información generada por los usuarios puede ser de utilidad para etiquetar datos y agregar información para los algoritmos de reconocimiento.

La incorporación de estas funcionalidades al diseño, además de cambios en la interfaz implica agregar dos operaciones en la API para disparar los eventos correspondientes en el otro proyecto.

4.2.2. Portal

Una vez diseñado el componente visual que facilitará la reproducción e indexación de los distintos videos, es necesario disponer de un portal web donde embeberlo para facilitarle el acceso y búsqueda de índices creados a todos los usuarios del sistema. Por lo tanto sus principales funciones son las de mostrar información de los índices, embeber el Indexador/Reproductor, mostrar listados de índices relevantes, permitir el ingreso de usuarios y generar vistas específicas para éstos.

Originalmente el portal estaba pensado como un sitio web simple, donde se incluiría el Indexador/Reproductor de forma embebida. Por lo tanto comenzamos evaluando alternativas como manejadores de contenidos (*CMS*) como *WordPress*, a los que se podría agregar algún complemento de terceros para autenticar a los usuarios mediante redes sociales (cuentas de *Google*, *Facebook*, *Twitter*, etc).

Durante el desarrollo del proyecto se agregaron dos nuevas funcionalidades que modificaron la arquitectura planteada para el portal:

- los usuarios pueden suscribirse a índices que no le pertenecen y ser notificados cuando estos índices son modificados
- los usuarios pueden valorar positivamente un hito en un video y esta información es enviada al proyecto *Video++2*

Por otro lado, surgieron inconvenientes tecnológicos relacionados con la alternativa de embeber el indexador en el portal mediante la etiqueta HTML *iframe*, así como problemas de integración entre componentes (portal-API, portal-indexador).
Como contrapartida, se planteaba la posibilidad de cubrir los requerimientos de acceso a través de

dispositivos móviles, mediante el uso de *Progressive Web Application.*

Por estos motivos decidimos rediseñar el portal pensando en una implementación web propia que permita evitar los problemas planteados y brinde más libertad para desarrollar los requerimientos definidos.

4.2.3. Indexador/Reproductor

El diseño del Indexador/Reproductor se basa en la especificación de requerimientos y fuertemente en los bosquejos planteados en la etapa de análisis.

El Indexador es el componente donde se visualiza un índice, permite reproducir los videos asociados a los hitos del índice, así como navegar a través de estos hitos. También permite crear y modificar índices, agregando y quitando videos y definiendo hitos sobre esos videos. El Indexador fue diseñado a partir de un reproductor de video al cual se le agregan las funcionalidades para gestionar hitos.

El Reproductor es el mismo componente que el Indexador sin las funcionalidades de edición de un índice.
Las funcionalidades para editar índices se muestran si se va a crear uno nuevo o si el usuario es propietario del índice que está visualizando.

Además de las funcionalidades básicas de un reproductor; como son reproducir y pausar el video, modificar el volumen, mostrar una línea de tiempo de reproducción, pantalla completa, etc; se le agregan controles para manipular los hitos de los videos. Estos controles son los siguientes:

- agregar un hito.
- navegar a hito siguiente/anterior en la línea de tiempo.
- agregar un video.
- listar opciones de hitos automáticos y generar los mismos.
- guardar/modificar índice.

El Indexador deberá mostrar marcas en la línea de tiempo, que representarán los hitos que se hayan definido de forma manual para el video que se está reproduciendo, cada marca indica el momento específico donde ocurre el hito.

Se definieron distintos sub-componentes dentro del Indexador:

- Componente que muestra los datos de los hitos. Está estructurado en pestañas: la primera muestra los hitos definidos manualmente, la segunda muestra los hitos generados a partir de los subtítulos del video (en idioma español, si existen en el proveedor). Luego de la pestaña de los subtítulos se muestra una pestaña para cada proveedor de hitos automáticos definido. Se muestra una etiqueta (*caption*) del instante junto a la descripción.
- Componente que muestra los videos del índice. Este muestra una minuatura para cada video definido en el índice.

Para la generación de hitos automáticos es posible definir nuevos módulos, que serán visualizados como pestañas a continuación de la correspondiente a hitos manuales.

Durante el desarrollo de este proyecto, se definió como proveedores de hitos generados automáticamente el servicio de subtítulos de *YouTube* y el reconocimiento automático provisto por el Proyecto de Grado *Video++2*.

4.2.4. Progressive Web App

Para la construcción del componente móvil, se realizó un análisis de cuáles serían las mejores opciones y herramientas para llevar a cabo la aplicación requerida.
No todas las opciones se adecuaban a las necesidades del proyecto y además se partía de la inexperiencia del equipo de trabajo en el desarrollo en tecnologías móviles, lo que redujo considerablemente la posibilidad de utilizar algunas alternativas.

En primera instancia se investigó la opción de realizar una implementación nativa para el sistema operativo *Android* y otra para *iOS*. En un desarrollo nativo, se aprovecha al máximo el rendimiento y funcionalidades del dispositivo por implementarse en un lenguaje específico y por ejecutarse directamente sobre el sistema, por lo que resulta la opción más adaptable a las características de cada dispositivo y brinda una experiencia de usuario mucho más fluida.
En contrapartida, este desarrollo conlleva un costo importante ya que deben destinarse recursos para escribir la misma aplicación en lenguajes diferentes y específicos para cada plataforma, duplicando el tiempo de desarrollo y no necesariamente aumentando los beneficios.

Si se abstrae al tipo de aplicación al que hace referencia el proyecto, donde no se necesitan grandes funcionalidades del dispositivo (como por ejemplo *GPS*, acceso a la cámara, micrófono, contactos, etc) y dado que el equipo de desarrollo no ha trabajado en dicha programación, esta opción deja de parecer una buena elección por la cual seguir adelante y la misma es descartada.

Como alternativa, se investigó en las tecnologías híbridas, como ser *React Native* [REAa] o *Ionic Framework* [ION]. Estos *frameworks* pretenden reutilizar código de modo que a partir de *JavaScript* y lenguajes de programación web como *HTML* y *CSS* se convierta a código nativo, mediante librerías como *Apache Cordova* [APC], tanto para la plataforma *Android* como para *iOS*. En este último caso además se necesita el programa de *iOS*, *Xcode* [XCO].
Si bien esta opción pretende ser bastante más favorable que la opción anterior, es decir desarrollar nativamente para cada plataforma por separado, tiene la desventaja de igualmente tener una curva de aprendizaje bastante alta. *React Native* utiliza sus propios componentes similares a *XML* y un sistema propio de estilos para diseñar las pantallas que también influye en la curva de aprendizaje y en el tiempo de programación.

Por último, se analizaron las tendencias en el desarrollo de aplicaciones web, que gracias a las últimas tecnologías de los navegadores da lugar a las *Progressive Web Applications* (*PWA*) impulsadas por *Google*. Éstas pretenden combinar las mejores características de la Web y de las aplicaciones nativas, para que a partir de una única implementación, la misma funcione como página web y como aplicación sin depender del sistema operativo, incluyendo notificaciones de tipo *push* [PUS] y acceso a través de un ícono en la pantalla de inicio del dispositivo o en el cajón de aplicaciones, pero sin necesidad de instalación desde una tienda.

Para poder tomar la decisión de optar por las *PWA*'s como alternativa para el componente móvil, se investigaron las ventajas que la misma ofrecía para adaptarse a la planificación del proyecto.

Como ventaja principal, las *PWA*'s implican un menor tiempo de programación. Gracias a los *Services Workers* soportados por muchos navegadores web (ver figura: 4.2), el portal puede ser adaptado para que a partir de una *URL* a través de un navegador, se acceda a una aplicación móvil con un "*look and feel*" nativo (quitando la barra de direcciones) y con funcionalidades como las notificaciones *push* que mantienen al usuario enlazado al producto (el navegador informará al usuario de las notificaciones recibidas y le redireccionará directamente a la aplicación). De este modo, se logra un portal web y un componente móvil casi en simultáneo con los mismos recursos.

Figura 4.2: Compatibilidad de service workers con navegadores

Otra ventaja a destacar es que si lo comparamos con las aplicaciones nativas, éstas últimas requieren de instalación a partir de las tiendas de *Apple Store* y/o *Google Play* (con las restricciones que dichas tiendas imponen), y consumen espacio de almacenamiento del dispositivo. Estos puntos no son relevantes en las *PWA's* ya que las mismas tienen una fácil distribución a través de enlaces, acceden a los contenidos por medio de servidores web sin necesidad de ocupar espacio y se mantienen siempre en su versión más actualizada. Además, tiene la gran ventaja de que los usuarios pueden probar su usabilidad tan rápido como se accede a una URL, rompiendo la barrera de la instalación previa para conocer el producto. Se ha investigado que cada vez es menor la cantidad de aplicaciones descargadas e instaladas por los usuarios, y es sustancialmente mayor la cantidad de visitantes de *Mobile Webs* [MAR].

Se tuvo en cuenta que durante el desarrollo del proyecto se agregaron dos nuevas funcionalidades:

- los usuarios pueden suscribirse a índices que no le pertenecen y ser notificados cuando estos índices son modificados
- los usuarios pueden valorar positivamente un hito en un video y esta información es enviada al proyecto *Video++2*

Considerando estos requerimientos, decidimos que era conveniente incluir estas funciones en el portal. Estas funcionalidades pueden ser resueltas utilizando una *PWA* en lugar de un portal simple como se planteó originalmente.
Por otra parte, como las *PWA*'s se acceden a través de los navegadores, es fácilmente posicionable en los motores de búsquedas como *Google*, llegando a más usuarios que en las tiendas de aplicaciones.

Como desventaja, al momento de la investigación los navegadores de los dispositivos de *Apple* como *Safari* no contaban con el soporte para *Services Workers*, limitando así las *Push notifications*,

almacenamiento cache, uso off line, etc. Pero en el transcurso del proyecto, más precisamente en 2018 la empresa *Apple* introdujo los *Services Workers* alejando la desventaja plasmada inicialemente.

Capítulo 5

Implementación de la solución

En este capítulo se describe a grandes rasgos la implementación realizada, las consideraciones técnicas y aspectos relacionados con la tecnología utilizada, así como las dificultades encontradas.

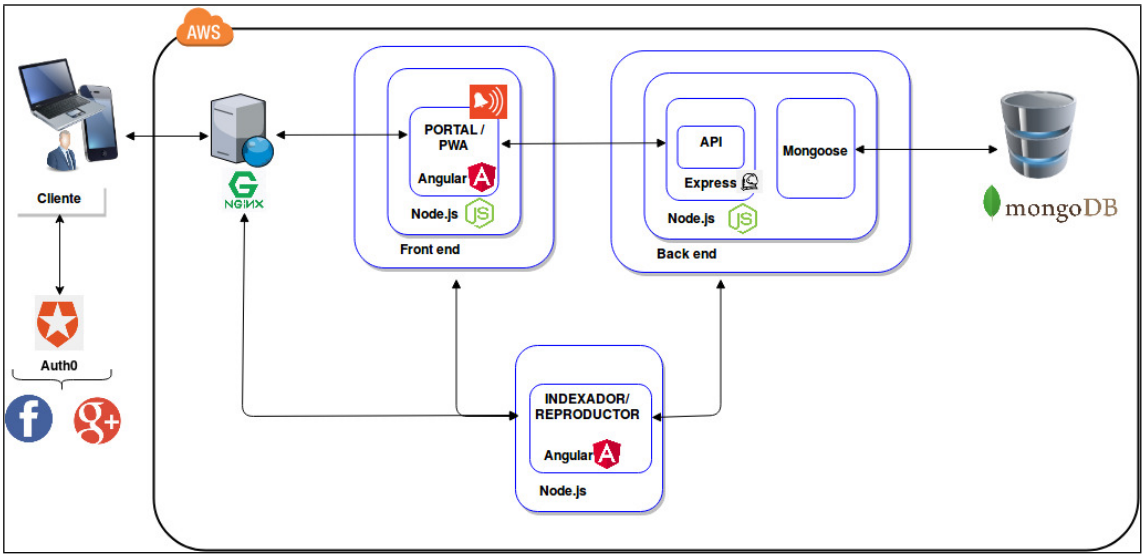


Figura 5.1: Diagrama de despliegue.

5.1. API

Para su implementación se utilizó *Node.js* [NOD] y *Express.js* [EXPa] como *framework* de aplicación web; y *MongoDB* [MONa] como manejador de base de datos de tipo *noSQL*. Este software de base es ampliamente utilizado en la actualidad para sistemas web modernos, ligeros y robustos [MEAa] [MEAb]; es parte del esquema conocido como *MEAN stack* [MEAc], junto con *Angular* como *framework* para el desarrollo *front-end*.[1]
Tanto en la API como en en los distintos clientes, se utilizó el servicio *Auth0* [AUT] como mecanismo de seguridad (autenticación y autorización).

[1]El término *MEAN stack* refiere originalmente a *AngularJS*, en este proyecto se utiliza *Angular2+*.

Implementación de la API

El *framework Express* implementa un servidor HTTP utilizando el lenguaje de programación *JavaScript* (JS). Funciona mediante el enrutamiento de pedidos y el uso de funciones denominadas *middleware* [EXPb], que forman una pila al ser invocadas en etapas. Básicamente estas funciones permiten acceder a los objetos del protocolo HTTP *request* (pedido) y *response* (respuesta) y a la siguiente función de la pila. Estas funciones pueden realizar las siguientes tareas: ejecutar cualquier código, realizar cambios en los pedidos y respuestas HTTP, finalizar el ciclo pedido-respuesta e invocar a la siguiente función definida en la pila.

La librería *Express* permite utilizar distintos tipos de funciones *middleware*: de aplicación, de enrutamiento de pedidos, manejo de errores, funciones embedidas y de terceros. Utilizando estos conceptos, la API se define como un conjunto de rutas que son evaluadas para cada pedido HTTP recibido, y para las cuales se definen ciertas acciones para resolver el pedido; típicamente se obtienen datos de la base a través de la librería *Mongoose* [MONb], se invocan funciones propias o de terceros y se genera la respuesta HTTP con el resultado. Las operaciones para obtener o consultar datos se corresponden con pedidos cuyo método HTTP es GET, mientras que para las modificaciones o generación de nuevas instancias se utilizan los métodos HTTP POST o PUT.

Librerías de terceros

Para facilitar la implementación y agregar ciertas funcionalidades que se fueron generando durante el desarrollo del proyecto, se utilizaron las siguientes librerías para implementar la API:

- **mongoose**: Librería para *Node.js* que permite acceder a un servidor de base de datos *MongoDB* de forma asincrónica. Ofrece una representación de datos a nivel de aplicación basada en el esquema subyacente, generación de consultas aprovechando todas las funcionalidades de *MongoDB*, amplio manejo de tipos de datos, validación y otras funciones útiles.
- **express**: Esta librería provee un conjunto de herramientas pequeño y robusto para implementar servidores HTTP, siendo adecuado para distintos tipos de aplicaciones web; como pueden ser: página única, sitios web complejos o APIs. Permite utilizar distintos *frameworks* de mapeo objeto-relacional así como bases de datos no relacionales.
- **helmet**: Esta librería se utilizó para agregar seguridad a la API implementada con *Express*. Fue necesario implementar el servicio sobre *Secure Socket Layer* (*SSL*, por sus siglas del inglés), dado que ciertas funcionalidades como las notificaciones de tipo *web-push* requieren que todo el portal use *HTTPS*.
- **auth0**: Uno de los requisitos iniciales del proyecto era ofrecer un mecanismo sencillo para la gestión de usuarios, principalmente para poder identificarse en el sistema mediante cuentas de redes sociales como Google, Facebook, Twitter, etc. El servicio que ofrece *Auth0* permite integrar la gestión de usuarios en distintos tipos de aplicaciones web, mediante el uso de cuentas de redes sociales.
- **request**: Se utilizó esta librería para enviar pedidos HTTP hacia el proyecto **Video++2**, quien provee un servicio de generación de hitos para videos.
- **web-push**: Esta librería nos permite enviar mensajes de acuerdo al protocolo *Web Push Notifications* [WEB] cuando se produce un evento con los datos que debe generar una notificación

al usuario. También permite manejar la plataforma de mensajería anterior, *GCM* [GCM] (siglas en inglés para *Google Cloud Messaging*), aún utilizada por algunos navegadores como *Google Chrome*. El servicio GCM fue discontinuado en abril de 2018.

5.2. Portal

La implementación del portal debe ser una aplicación web que utilice la API. Luego de evaluar distintas alternativas junto con el tutor del proyecto, decidimos implementar el portal utilizando el *stack MEAN*, que ya es utilizado para implementar la API y al que se le agrega el lenguaje *Angular 2+* [ANGb] para el desarrollo del *front-end*.

El portal consiste en varios componentes y servicios de *Angular 2+*. Los componentes propiamente dichos fueron utilizados para representar distintas páginas del portal, típicamente asociados a una entidad del modelo de datos o a una pantalla definida en los bosquejos. Cada componente posee una clase que define su comportamiento, una plantilla *HTML* y una hoja de estilos que definen la vista para esa clase. Se definieron los siguientes componentes:

- **HomeComponent:** Componente que define la página principal del portal, aquí se muestran los principales índices. Este listado puede ser adaptado para ofrecer resultados distintos según si el usuario está logueado o no o basarse en datos del usuario para seleccionar determinados índices interesantes o bien utilizar algún criterio para el listado, como los añadidos recientemente.
- **HeaderComponent y FooterComponent:** En estos componentes se implementan las secciones del portal que estarán fijas en todas las páginas. Por un lado en el cabezal se muestra el título y el menú de usuario (o la posibilidad de iniciar sesión), así como el menú de tipo *sidenav*. Mientras que en el pie de página se muestra información sobre el sitio.
- **AdminComponent:** Este componente reúne las funciones disponibles para usuarios que posean el rol de administrador, permite por ejemplo que se agreguen proveedores de hitos definiendo la *URL* a donde se realizan los pedidos.
- **CallbackComponent:** A este componente se redirigen todos los pedidos luego de que un usuario inicia sesión en el portal, luego desde él se selecciona la página correspondiente (desde dónde se inició el proceso de *login*).
- **EmbedComponent:** Componente creado para embeber el componente indexador dentro del portal.
- **IndiceComponent:** Este componente accede a los datos de un índice, para mostrar los objetos relaciondos a él (hitos y videos) y la información detallada sobre el índice. Invoca al componente Embed para mostrar el indexador/reproductor en la página junto con los detalles.
- **IndiceDinamicoComponent:** Componente que maneja el resultado de una búsqueda. Cada vez que se realiza una búsqueda de hitos, se genera un índice de forma dinámica que luego será persistido. Éste componente también invoca al componente Embed para mostrar el reproductor.
- **PerfilComponent:** Componente utilizado para mostrar el perfil de un usuario logueado. El mismo incluye la información básica del usuario, cantidad y lista de índices creados; y cantidad y lista de índices suscriptos.

Aprovechando el enfoque de *Angular 2+*, se definieron los siguientes componentes como servicios:

- **AuthService:** Este servicio se encarga de proveer a los demás componentes del portal de las funcionalidades de autenticación y autorización de usuarios, así como de mantener información sobre el perfil del usuario, el estado de la sesión, las suscripciones que posea el usuario y las notificaciones que deba recibir.
- **ApiService:** Se definieron dos servicios que interactúan con la *API*, uno que implementa los *endpoints* para todas las operaciones relacionadas con índices, hitos y videos y otro para las operaciones sobre usuarios. Cada vez que desde el portal se necesita consumir un servicio de la *API*, se utilizan estos servicios. De esta forma se encapsula el acceso evitando realizar invocaciones de bajo nivel.
- **UtilService:** Servicio que proporciona funciones auxiliares, por ejemplo el formato en que se muestran las fechas.

El desarrollo de la integración del portal con el servicio de autenticación *Auth0* estuvo basado en el artículo descripto en [REAb].

Diseño estético del portal

Como criterios para elegir un estilo de diseño más allá de lo definido por los bosquejos, planteamos los siguientes objetivos: lograr un diseño moderno, minimalista y que resulte familiar para un usuario promedio de Internet. Entendemos que el concepto de diseño *Material Design* [MATa] cumple con estos requisitos y en los últimos años se ha expandido su uso y ha ido evolucionando. Además es impulsado por importantes compañías de *software*.

Este *framework* para diseño intenta unificar conceptos, criterios, comportamientos, etc; entre distintas plataformas como aplicaciones web y móviles. Por este motivo lo consideramos adecuado tanto para el Portal/PWA como para el componente Indexador.

Para implementar un estilo basado en *Material Design* tanto en el Portal/PWA como en el Indexador, se utilizó la librería *Materialize* [MATb]. Esta librería permite aplicar los conceptos y elementos definidos por *Material Design* a elementos HTML estándar, simplificando en gran medida la implementación y logrando un código fuente más entendible.

5.3. Indexador/Reproductor

Se resolvió definir un único componente para la creación, modificación y reproducción de un índice. Este componente tiene dos estados:

- **Indexador**: cuando se permite crear/modificar un índice.
- **Reproductor**: cuando solo permite reproducir un índice.

Dicho estado queda definido dependiendo de los parámetros recibidos cuando se invoca al componente Indexador.

Los posibles llamados son:

- `<url>/?i=new`
- `<url>/?i=new&v=<URLPROVEDOR>`
- `<url>/?i=<IDINDICE>`
- `<url>/?d=<IDINDICE>`

El parámetro "`i`" corresponde al identificador del índice. Si el valor es "`new`" o si es "*IDINDICE*" y el usuario es el propietario del índice, decimos que el componente tiene estado **Indexador**. Para los demás casos el estado es **Reproductor**, incluyendo el caso donde el parámetro es "*d*" correspondiente a los índices dinámicos.

Las funcionalidades que no se muestran en el Reproductor son las siguientes:

- agregar un hito
- agregar un video
- guardar un índice
- pedir hitos automáticos
- reordenar los hitos

Para cumplir con los requerimientos se buscó una solución que cumpla los siguientes puntos:

- se pueda reproducir videos de varios proveedores.
- sea multiplataforma.

Esto llevó a la utilización de la herramienta *Video.js* [VIDa]. La misma se define como "*una librería de código abierto para trabajar con videos en la web, también conocido como reproductor de video de HTML*" [VIDa]. Esta herramienta ofrece una API y un mecanismo de plugins mediante los cuales se puede extender la funcionalidad del reproductor.
Los plugins de *Video.js* utilizados fueron el de reproducir videos de Youtube [VIDc] y el de agregar marcas a la barra de tiempo [VIDb].

Con la API y los plugins de *Video.js* no se pueden implementar todos los requerimientos que son necesarios para la interacción con un índice. Por este motivo se decide utilizar el *framework* para JavaScript de *Angular2+* para resolver estos casos.
El uso de este *framework* nos ofrece una manera sencilla para el manejo de datos correspondiente al índice y de la creación de componentes visuales para interactuar con un índice.

Esto implicó algunas dificultades a la hora de interactuar con *Video.js*, ya que desde la instancia de *Video.js* no se tiene acceso a los datos del contexto de *Angular*. Estas dificultades son asumidas dado el complejo trabajo que implica una solución implementada exclusivamente en *JavaScript*.

Por esas razones la lógica del indexador queda estructurada en dos partes:

Por un lado tenemos la inicialización de una instancia de *Video.js*. Al invocar al Indexador se llama a la lógica del archivo `inicializacion-video-js.js` que es el encargado de crear una

instancia de *Video.js* e inicializar los plugins que utiliza.

La otra parte refiere a la lógica del indexador/reproductor definida en un componente *Angular* denominada `indexador.component.ts`. En este archivo se declaran e inicializan las variables que definen a un índice y es donde reside la lógica para la creación y modificación de los mismos.

Dentro de este componente se importa la librería *Video.js*, se obtiene la instancia inicializada previamente y se interactúa con la misma para utilizarla en los distintos métodos que manejen las funcionalidades del indexador. Entre ellos se encuentra la creación de hitos manuales, generación y listado de hitos automáticos, manejo de los controles de *Video.js*, persistencia de índices, entre otros.
Para comunicarse con la base de datos, el componente Indexador utiliza las operaciones de la misma API que consume el portal.

Por último, para el diseño y definición de estilos del Indexador se utilizaron las librerías *Materialize CSS* [MATb] y *Angular Material* [ANGa]. Estas librerías implementan componentes visuales, basados en *Material Design*, que definen guías de diseño enfocadas en la visualización de aplicaciones móviles y páginas web. Para el caso particular de dispositivos móviles, se definen gestos táctiles utilizando la librería *hammerjs* [HAM].

5.3.1. Hitos automáticos

Para la creación de hitos automáticos el Indexador ofrece una API con la cual se pueden agregar múltiples generadores de hitos automáticos. Esta API está compuesta por dos operaciones:

- **generarHitos(idVideo, idioma, tipo) : ticketID**. Operación asincrónica que genera hitos para un video en particular y retorna un ticketID que identifica a la solicitud.
- **cargarHitos(ticketID, resultado: JSON)**. Operación implementada por quien solicitó el procesamiento de un video e invocada por quien genera contenido. La misma devuelve el resultado en formato JSON de la solicitud de procesamiento identificada con ticketID.

Para que estos proveedores sean listados en la opción de autogenerados del Indexador se debe agregar un registro en la tabla “Autogenerados” de la base de datos. En dicha tabla debe guardarse el nombre que identifica al proveedor y la URL que indica el *endpoint* a ser invocado para la creación de los hitos automáticos.

Para probar el uso de la API realizamos una prueba de concepto con los subtítulos de Youtube. Posteriormente se integró con el proyecto *Video++2* quienes implementaron las operaciones de nuestra API.

5.4. Progressive Web App

Luego de realizar un relevamiento de tecnologías, decidimos cubrir el acceso desde dispositivos móviles utilizando una *Progressive Web Application* (PWA). Para esto, el objetivo fue desarrollar un portal con tecnologías modernas, que permitan convertir un sitio web en una aplicación móvil dependiendo del dispositivo cliente.

En el momento de comenzar con este desarrollo encontramos información heterogénea sobre el tema, desde nuestro punto de vista aún se estaba definiendo a nivel de los grandes proveedores de servicios de Internet qué se entendía por *PWA*. De todas formas, siempre hubo cierto consenso en definir las características principales que debe cumplir una *PWA*. Inicialmente nos guiamos por la publicación *Your First PWApp* en el sitio de desarrolladores de *Google* [PWAe]. Con el paso del tiempo, *Google* ha impulsado este tipo de tecnologías y en ese marco se han publicado nuevas referencias, que también consultamos: [PWAb], [PWAc] y [PWAd].

5.4.1. Criterios

Partimos de los siguientes requisitos para convertir nuestro portal en una *PWA*, el objetivo definido fue abarcar las características principales sin tener que cumplir cada punto de forma estricta:

- **progresiva:** funciona para todos los usuarios, independientemente de su navegador
- **responsiva:** se adapta a todos los formatos: escritorio, movil, tablet o nuevas plataformas
- **independiente de la conexión:** mediante el uso de *Service Workers* puede funcionar sin conexión o con conexiones de red de baja calidad
- **estilo APP:** la experiencia de uso es similar a una aplicación móvil, dado que la estructura separa las funcionalidades del contenido
- **frescura:** siempre actualizada gracias a los procesos de actualización que realiza el *Service Worker*
- **segura:** accesible mediante HTTPS para evitar modificación del contenido y demás problemas de seguridad
- **descubrible:** identificable como aplicación gracias al uso de *W3C Manifest* [APP] [MDN] y el registro de *Service Worker*, lo que permite ser encontrada por buscadores
- **vinculación con el usuario:** a través de funcionalidades como las *push-notifications*, es más fácil mantener al usuario conectado a la aplicación
- **instalable:** permite que el usuario la instale sin necesidad de una tienda de aplicaciones
- **compartible:** es fácilmente compartible mediante una URL, no necesita una instalación compleja

Para nuestro caso, definimos que las características más importantes son: i) la posibilidad de instalar el sitio como aplicación móvil, ii) que tenga un estilo de aplicación y iii) que ofrezca la posibilidad de enviar notificaciones de tipo *push* (generadas desde el servidor).

Dado que los requisitos planteados previamente son en general conceptuales, para realizar la implementación y medir si se cumplían los criterios utilizamos la herramienta *LightHouse* de Google [LIG].

5.4.2. Implementación

Los criterios antes definidos se llevan a cabo mediante alguno de estos mecanismos:

metadatos: A través de un archivo manifiesto, se define el comportamiento y apariencia que los navegadores web deben darle al sitio.

***Service Workers*:** Se instala código en el dispositivo cliente para correr procesos en segundo plano, los objetivos pricipales son mejorar el rendimiento del sitio mediante la gestión de cache y recibir y mostrar las notificaciones que se generan del lado del servidor.

adaptaciones en el sitio: Los distintos componentes del sistema (Indexador, Reproductor, Portal y API), son modificados para cumplir requisitos o brindar ciertas funcionalidaes. Por ejemplo: todo el contenido debe ser servido mediante HTTPS, se debe almacenar la información necesaria para enviar notificaciones y disparar los eventos correspondientes, etc.

En el apéndice D se incluyen íntegramente el manifiesto de la aplicación y el código de los *Service Workers* que gestionan el cache y las notificaciones.

En la figura 5.2 se visualizan los elementos que definen a un sitio como *PWA*, mientras que en la figura 5.3 se presenta un esquema sobre el ciclo de vida de los *Service Workers*.

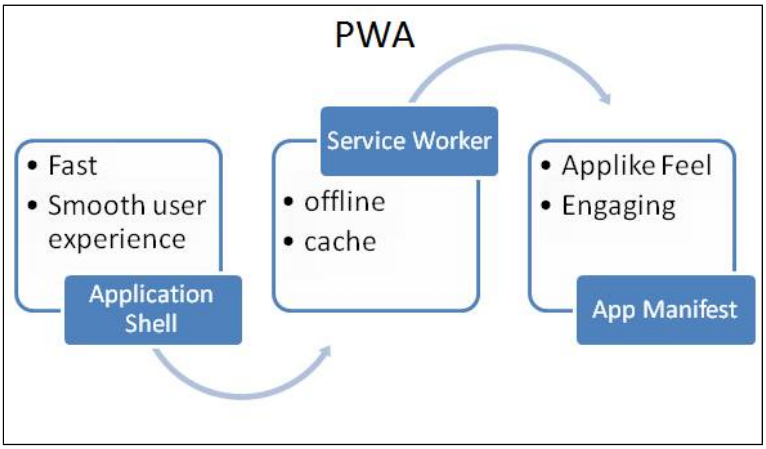


Figura 5.2: Implementación PWA

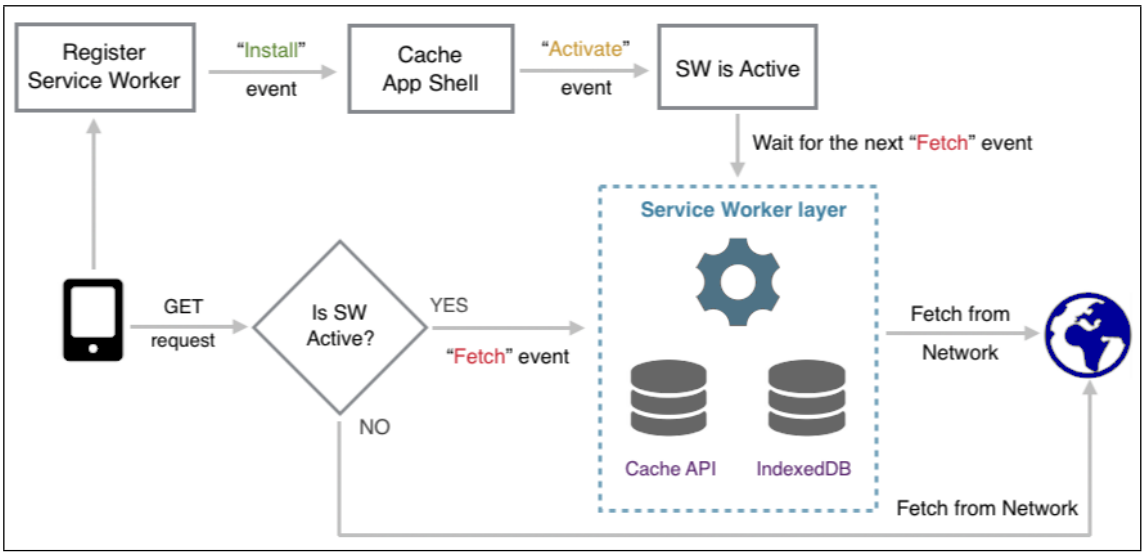


Figura 5.3: Service Worker

Una de las principales características que ofrecen las *Progressive Web Applications*, además de recibir notificaciones de tipo *Web Push*, es que le brindan al usuario la posibilidad de ser instaladas como aplicaciones nativas. Los navegadores web actuales, identifican que el sitio al que accede el usuario es una *PWA* y muestran un *banner* con la opción de instalar el sitio como aplicación nativa. En la figura 5.4 puede verse un ejemplo de este comportamiento en la *PWA* desarrollada.

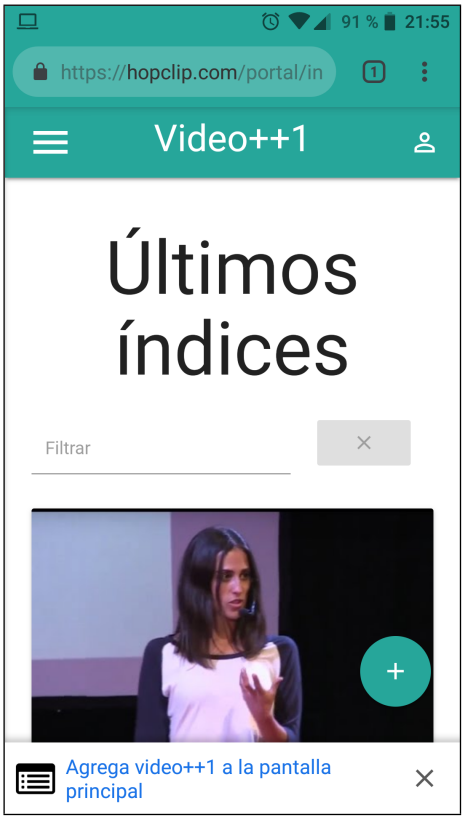


Figura 5.4: Banner de instalación de PWA en Chrome

Al momento de desarrollar nuestra *PWA*, los criterios que aplica *Google Chrome* para decidir si debe mostrar la opción de instalación, son los siguientes [PWAa]:

- Contener un archivo de manifiesto de aplicación web con:
  - un `short_name` (usado en la pantalla de inicio)
  - un `name` (usado en el banner)
  - un ícono png de 192 x 192 (en las declaraciones del ícono se debe incluir un tipo de mime `image/png`)
  - una `start_url` que se carga
- Contener un *service worker* registrado en tu sitio.
- Transmitirse a través de HTTPS (un requisito para usar el *service worker*).
- Recibir visitas al menos dos veces, con al menos cinco minutos de diferencia entre las visitas.

Manifiesto

El manifiesto de la aplicación, en nuestro caso, define ciertos elementos generales como: nombre del sitio, lenguaje, forma de mostrar la aplicación, orientación, color, etc. También se incluye un identificador para utilizar el servicio de mensajería de *Google* (GCM), necesario para enviar notificaciones a algunas versiones del navegador *Google Chrome*.

Es necesario definir el ícono de la aplicación en dos tamaños y es posible vincular aplicaciones relacionada con el sitio. El contenido del archivo `manifest.webmanifest` se incluye en el apéndice D.

Notificaciones Web-Push

Se implementó un *Service Worker* (SW) para mostrar las notificaciones en el dispositivo cliente. Además fue necesario implementar métodos en el portal y operaciones en la API para ofrecer esta funcionalidad.

Para implementar notificaciones, se eligió la funcionalidad que permite a un usuario suscribirse a uno o varios índices y recibir notificaciones cuando estos son modificados. En el portal se incluyó un botón para suscribirse a un índice, este control aparece en el detalle del índice (componente `IndiceComponent`) siempre que éste no le pertenezca al usuario que inició sesión. Cuando el usuario se suscribe, se debe registrar el SW y solicitar que el usuario habilite el envío de notificaciones para el sitio. Estas funciones están implementadas en el componente `AuthService`.

En ese momento, el navegador genera un *endpoint* que lo identifica, mediante el cual el sitio puede enviar notificaciones a ese dispositivo. Dado que nuestra funcionalidad debe ser brindada a nivel de usuario, debemos generar suscripciones para este usuario y el índice indicado, en cada dispositivo que use.

Esto se logra manteniendo una lista de índices a los que el usuario está suscripto, cada vez que el usuario inicia sesión en un dispositivo, se obtienen los índices a los que está suscripto en su perfil de usuario y se generan las suscripciones para este dispositivo si no existen. Además, en cada índice

se mantiene la lista de suscripciones (a nivel de dispositivos) a los que se debe notificar.

Para realizar la desuscripción, se procede de forma inversa, a partir del identificador del usuario se eliminan todas las suscripciones que posee para el índice indicado. También se borra la referenia al índice en el perfil del usuario.

Cuando un índice es modificado, la operación de la API `updIndice` recorre la lista de suscripciones y envía una notificación a cada dispositivo.

Evaluación con la herramienta Lighthouse

La implementación del sitio como PWA se realizó siguiendo las recomendaciones que ofrece la herramienta *Lighthouse* [LIG]. Esta herramienta de código abierto permite realizar una auditoría de un sitio, con el objetivo de determinar en qué medida se cumplen los criterios que definen a un sitio/aplicación como PWA.

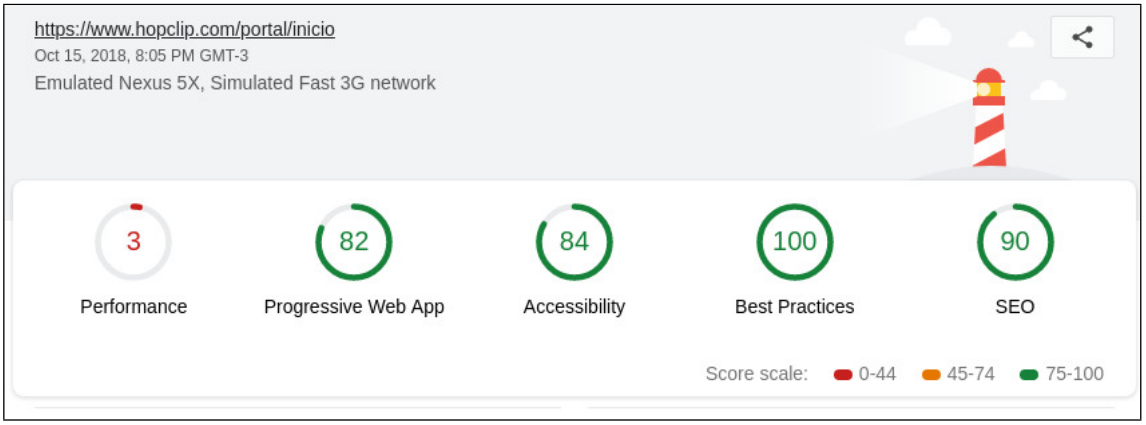


Figura 5.5: Análisis con Lighthouse

Luego de presentar un resumen numérico (con escala de 0 a 100), para las categorías: *Performance, Progressive Web App, Accesibility, Best Practices y SEO*, la herramienta detalla dentro de cada categoría los criterios de auditoría aprobados y ciertas oportunidades de mejora.

En el análisis que corresponde a la figura 5.5, se observa que se cumple correctamente con varios ítems. Destacamos los más importantes:

Performance :

- imágenes de tamaños adecuados
- código JS y CSS *minificado* (reducido)
- se evitan múltiples redirecciones
- codificación de imágenes eficiente

Progressive Web App :

- se responde con estado HTTP 200 cuando no hay conexión
- se le pregunta al usuario si quiere instalar la *Web App*

- se registran SW
- se configura una pantalla de inicio

Accessibility :

- los elementos tienen nombres discernibles
- los elementos describen correctamente el contenido
- los elementos están bien estructurados
- se usa correctamente los *Meta Tags*

Best Practices :

- se evita usar el cache de aplicación, WebSQL DB y `document.write()`
- se usa HTTPS
- no se muestran errores del navegador en la consola
- se evita el uso de librerías de JS con vulnerabilidades conocidas, APIs desactualizadas y solicitar permiso para geolocalización al cargar la página

SEO: estos controles evalúan si la página está optimizada para los rankings de los motores de búsqueda

- presenta un archivo robots.txt válido, no se bloquea la indexación
- tiene etiquetas para el título y metadatos, los enlaces tienen texto descriptivo
- presenta características *Mobile Friendly*

En la figura 5.6 se muestan algunas de las pruebas aprobadas para la sección *Best Practices*.

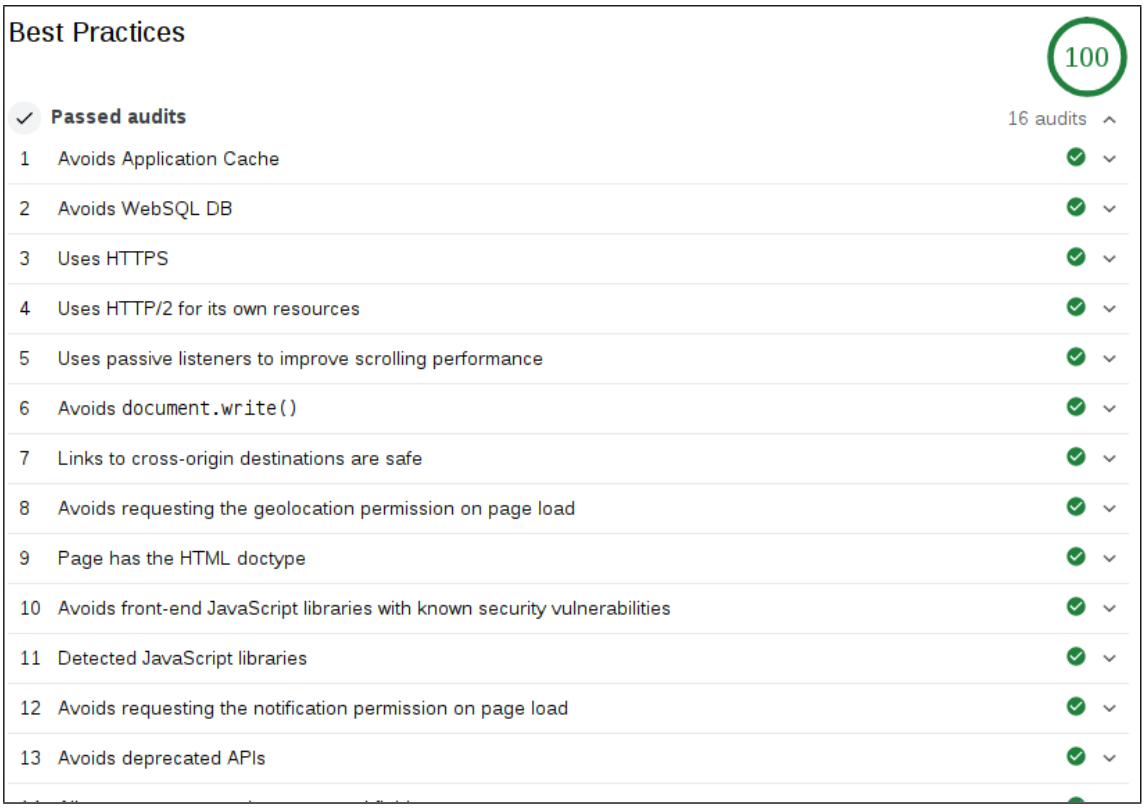


Figura 5.6: Mejores prácticas según Lighthouse

Las principales mejoras sugeridas asociadas al análisis correspondiente a la imagen 5.5 son:

Performance :

- habilitar compresión de texto
- precargar pedidos clave
- eliminar recursos que bloquean el *renderizado* de la página
- quitar reglas no usadas en hojas de estilos

Progressive Web App :

- el tiempo de carga de la página no es suficientemente rápida en conexiones 3G

Accessibility :

- algunos elementos con imágenes no tienen definido el atributo “alt”

En la figura 5.7 se muestan algunas de las oportunidades de mejora que identifica la herramienta para la categoría *Performance*.

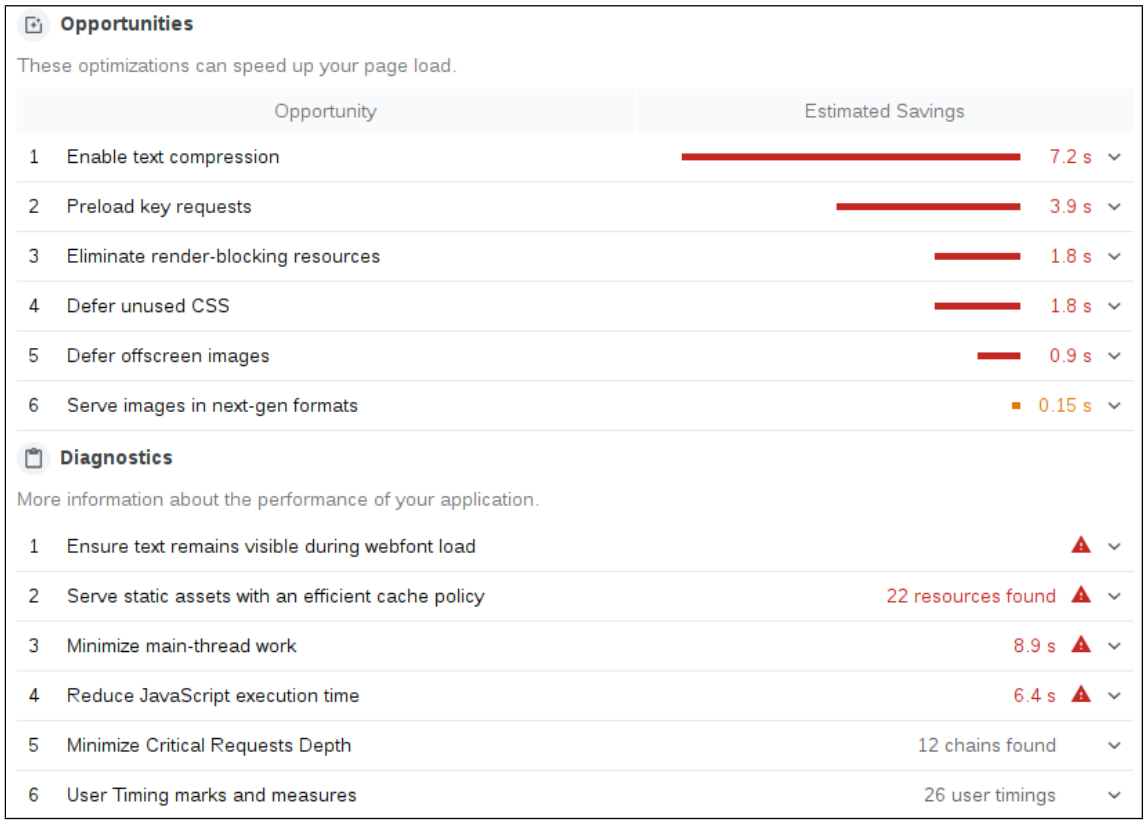


Figura 5.7: Aspectos a mejorar en rendimiento según Lighthouse

Capítulo 6

Pruebas

6.1. Enfoque y plan de pruebas

Verificar que el software diseñado cumple con las expectativas iniciales y que funciona acorde a los requerimientos, es tan importante como el propio desarrollo del mismo. La idea principal es encontrar defectos lo antes posible, poder corregirlos y asegurarle al usuario final que el producto cumple con el funcionamiento esperado.

Para ello, se analizaron los distintos tipos de pruebas y se decidió acorde al tipo de sistema construido. En este caso, como el producto construido implica en su mayoría un desarrollo visual, donde la interfaz gráfica juega un rol muy importante, es que se optó por realizar pruebas funcionales denominadas de "Caja Negra" (ver esquema en la figura 6.1), cuyo fin es probar las funcionalidades del sistema comprobando "lo que el sistema hace" a través de la utilización de la interfaz. Además, a pesar de no entrar en detalle en "cómo se realizan las tareas", se puede comprobar el correcto funcionamiento de la integración de los distintos componentes entre sí.

Figura 6.1: Pruebas de caja negra.

Para llevar a cabo estas pruebas, se planificó un plan de pruebas donde se pretende cubrir el máximo posible de funcionalidades e interacciones realizadas.

Como primera parte, se realizan pruebas unitarias a diferentes secciones del sistema para comprobar su funcionamiento independientemente. Luego, se realizan pruebas de integración para verificar que el sistema se comunica correctamente con los distintos artefactos con los que interactúa, como por ejemplo la API.

Posteriormente se realizan pruebas de interfaz con la intención de comprobar que se cumplen

las funcionalidades especificadas en los requerimientos. Además de pruebas de aceptación, donde se obtiene una comparativa de bosquejos de la etapa de diseño con las distintas pantallas contruidas en el producto final.

Por último se realizan algunas pruebas no funcionales para verificar si cumplen con los resultados esperados.

6.2. Pruebas funcionales

Para llevar a cabo las pruebas funcionales del sistema, se realizó un plan de pruebas partiendo de la especificación de requerimientos, más precisamente desde los bosquejos diseñados, para que a través de distintos escenarios de entrada se compararen los resultados obtenidos y así puedan ser validados. A continuación se plantean las diferentes pruebas funcionales realizadas según el plan de pruebas:

6.2.1. Pruebas unitarias

La estrategia en estas pruebas fue aislar las diferentes partes del sistema de modo de probarlas independientemente para comprobar su correcto funcionamiento y no depender de la integración del mismo. Además ayudan a la detección temprana de errores mientras se está desarrollando cada funcionalidad.

Una de las pruebas realizadas que fue de mucha ayuda a la hora de comprobar que se estaba implementando correctamente, fue realizada a través de la herramienta de línea de comandos *cURL* para probar el funcionamiento de la *API*. A medida que se implementaban las distintas operaciones de la API, encargada de interactuar con la base de datos, se realizaban pruebas para comprobar que las acciones realizadas por cada una de ellas eran las esperadas y el resultado retornado era el deseado. Por lo tanto, se envían peticiones *HTTP* a las distintas *URL's* de la *API* y se verifican las salidas.

A continuación se detallan algunas de las pruebas realizadas:

Prueba 1:
Se prueba el método "*api/indicesHomeMasVistos*" que debe retornar una lista de todos los índices del sistema ordenados por el atributo "cantVistos":

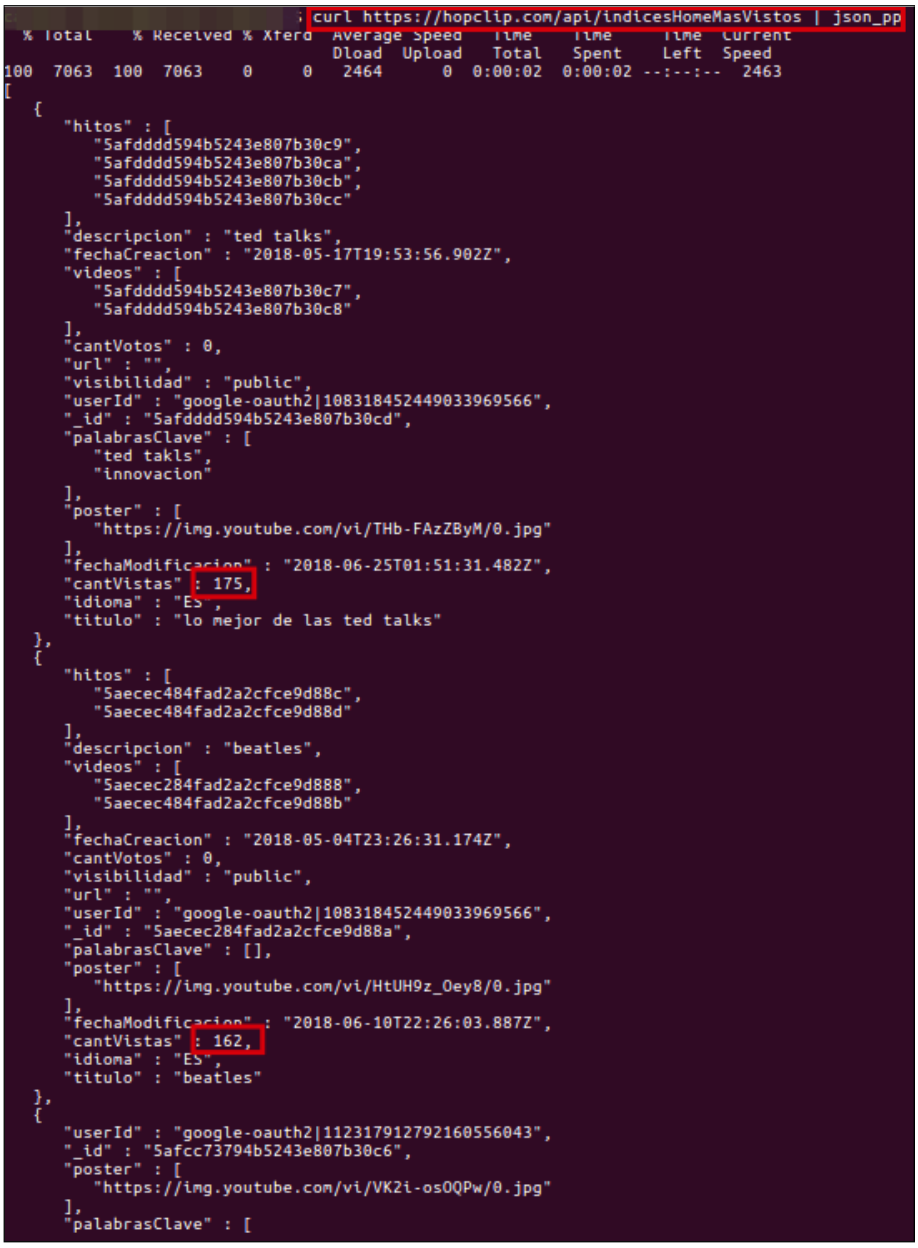


Figura 6.2: cURL: Índices más vistos

Como se puede ver en la imagen 6.2, la invocación al método se ejecuta correctamente y la misma devuelve un conjunto de datos en formato JSON. Contiene la información de los distintos índices donde el primero tiene 175 cantidad de vistas, el siguiente 162 y así sucesivamente de modo que también se verifica que el listado es ordenado descendentemente por número de vistas.

Prueba 2:

Se prueba el método "*api/indiceByHito/:id*" que dado un id de Hito debe retornar los datos del índice que lo contiene:

Figura 6.3: cURL: Índice que contiene a un hito

En esta imagen 6.3, se puede apreciar que la invocación al método se realiza de forma exitosa, retornando los datos del índice que contiene al hito con dicho identificador. Éste método es utilizado en las búsquedas dinámicas para poder recrear el contexto de los hitos que coinciden con el criterio de búsqueda, permitiendo conocer el índice al que pertenece originalmente.

Prueba 3:

Se prueba el método "*api/suscribirse/:idIndice*" que suscribe el *endpoint* incluido en el *body* del *request* al índice con determinado id. Para corroborar el correcto funcionamiento del método se verifica antes y después de la invocación el estado de la base de datos para ese índice.

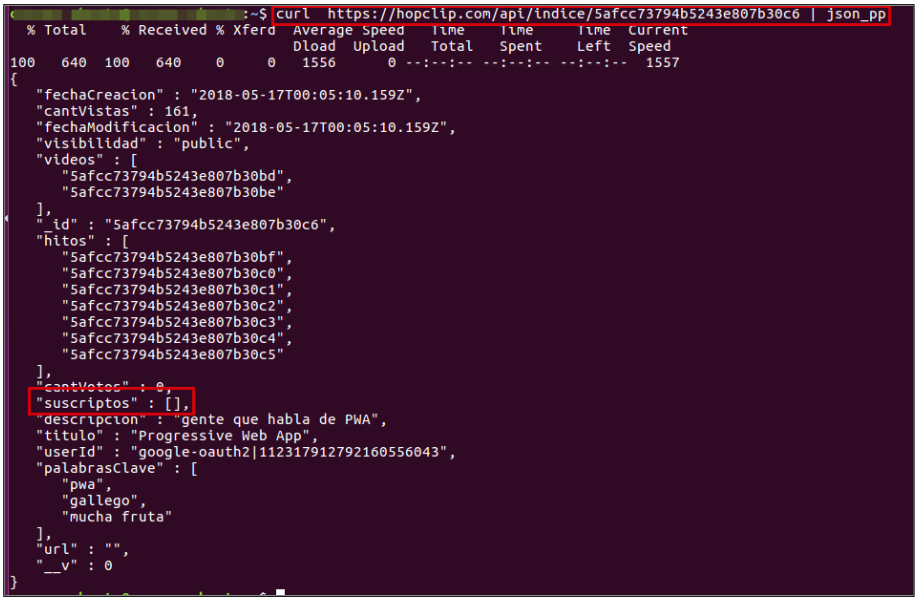


Figura 6.4: Índice antes: sin suscripciones

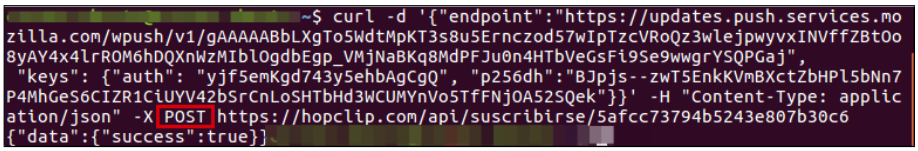


Figura 6.5: Invocación al método de suscripción al índice

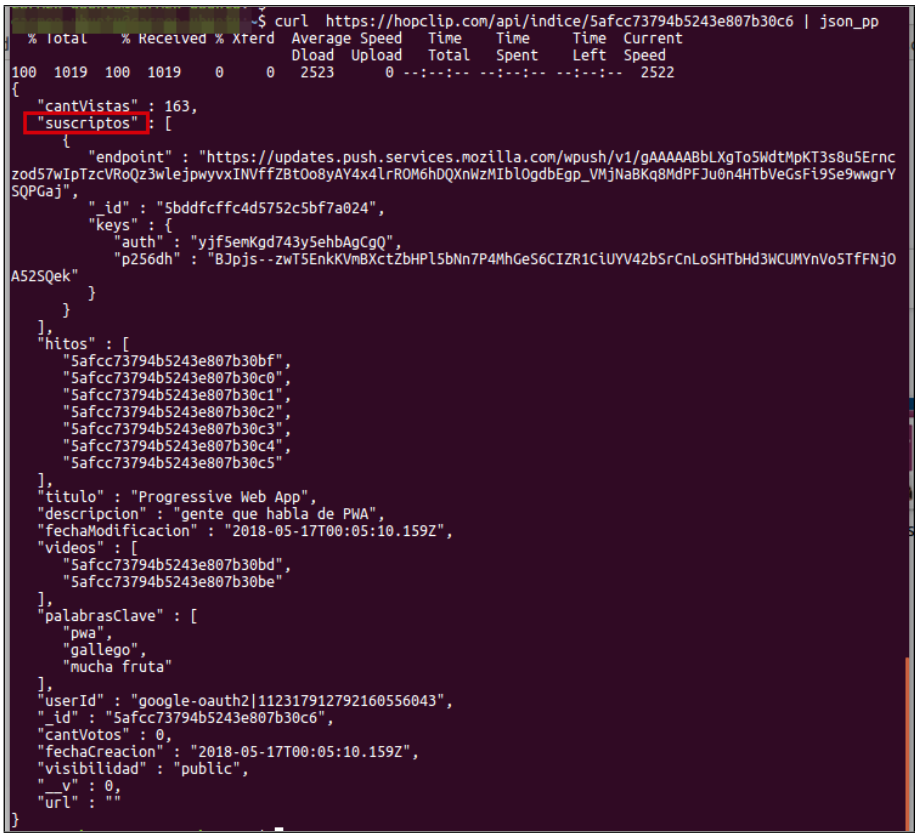


Figura 6.6: Índice después: con suscripciones

6.2.2. Pruebas de integración

En estas pruebas se pretende testear las diferentes interfaces e interacciones de los distintos componentes que deben trabajar en conjunto para que el sistema funcione correctamente. Para ello se plantearon varias pruebas que se detallan a continuación:

Prueba 1:

Descripción: Se pretende probar que el componente **Indexador** consume correctamente de la interfaz expuesta por la **API**, quien a la vez es la encargada de consumir los datos de la base de datos **MongoDB**.

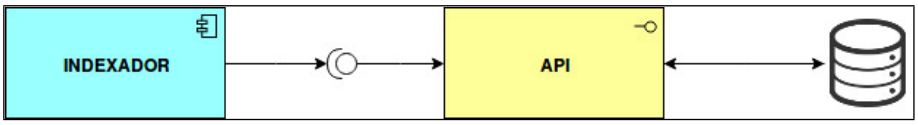


Figura 6.7: Interacción con API

Se intentó contemplar todos los casos incluyendo los casos borde, para cubrir la mayor cantidad de situaciones posibles:

- 1.1:

  Resultados esperados: Visualizar un Índice con 1 Video pero sin Hitos.

  Resultados obtenidos:

Figura 6.8: 1 Video - 0 Hitos

- 1.2:

  Resultados esperados: Visualizar un Índice con 1 Video y 1 Hito.

  Resultados obtenidos:

Figura 6.9: 1 Video - 1 Hitos

- 1.3:

  Resultados esperados: Visualizar un Índice con 1 Video y más de 1 Hito.

  Resultados obtenidos:

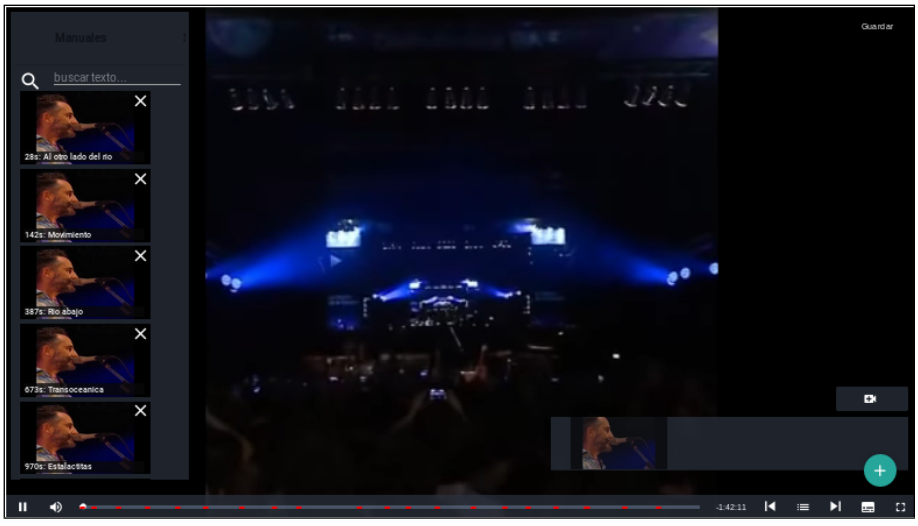


Figura 6.10: 1 Video - 21 Hitos

- 1.4:

  Resultados esperados: Visualizar un Índice con más de 1 Video y más de 1 Hito.

Resultados obtenidos:

Figura 6.11: 3 Videos - 4 Hitos

Prueba 2:

Descripción: Se pretende probar que el componente **Indexador** obtiene los subtítulos de un video de forma correcta a través del consumo de la interfaz expuesta por la **API de YouTube**.

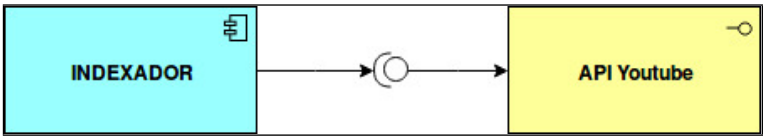


Figura 6.12: Interacción con API YouTube

Para esta prueba también se contemplaron los casos borde:

- 2.1:

Resultados esperados: Intentar obtener los subtítulos de un video de *YouTube* pero el mismo no tiene subtítulos y por lo tanto no debe mostrar ningún conjunto de subtítulos.

Resultados obtenidos:

Figura 6.13: Video sin subtítulos

- 2.2:

  Resultados esperados: Obtener los subtítulos de un video de *YouTube*. El mismo debe retornar un conjunto con más de un subtítulos.

  Resultados obtenidos:

Figura 6.14: Video con subtítulos

Prueba 3:

Descripción: Se pretende probar que el componente **Indexador** obtiene hitos automáticos de reconocimiento a partir de la interfaz de comunicación con el proyecto "*Video++2*".

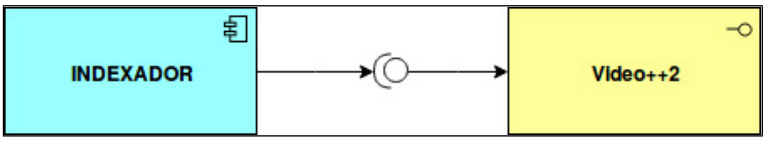


Figura 6.15: Interacción con *Video++2*

- 3.1:

  Resultados esperados: Obtener lista de múltiples hitos automáticos de un video de *YouTube* a partir del resultado en formato JSON retornado por el proyecto "*Video++2*".

  Resultados obtenidos:

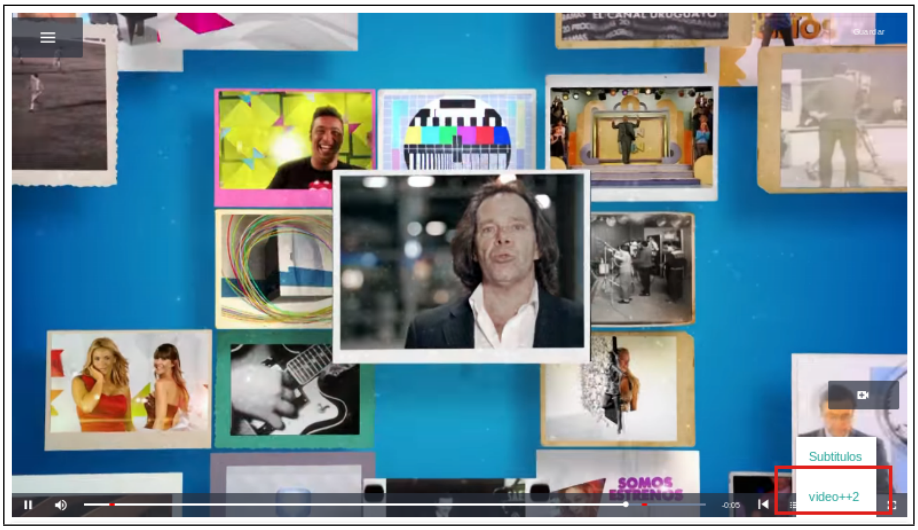


Figura 6.16: Selección de opción *Video++2* como hitos autogereados.

Figura 6.17: Hitos autogereados de *Video++2* en proceso.

Figura 6.18: Hitos autogereados de *Video++2* exitosamente.

Prueba 4:

Descripción: Se pretende probar que el componente **Indexador** obtiene las imágenes en miniatura (*thumbnail*) de diferentes instantes de tiempo de un video de *YouTube*.

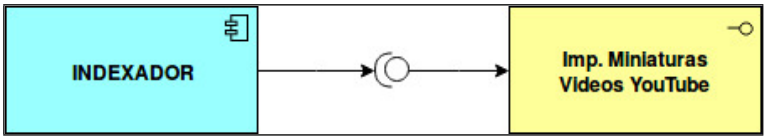


Figura 6.19: Interacción con Impl. miniaturas YouTube

Para esta prueba se testean tanto casos donde el video tiene miniaturas asociadas, como casos donde no las tenga.

- 4.1:

  Resultados esperados: Se visualiza las miniaturas de un video de *YouTube* para distintos instantes de tiempo.

  Resultados obtenidos:

Figura 6.20: Video con miniaturas (thumbnail)

- 4.2:

  Resultados esperados: Se visualiza una miniatura por defecto para el caso en que el video de *YouTube* no tiene miniaturas asociadas para los instantes de tiempo probado.

  Resultados obtenidos:

Figura 6.21: Video con miniatura por Defecto

Prueba 5:

Descripción: Se pretende probar que el componente **Portal** se comunica correctamente con la interfaz expuesta por la *API* de modo que obtiene listados de Índices con sus respectivos datos, además pretende probar que embebe exitosamente el componente *Reproductor/Indexador*.

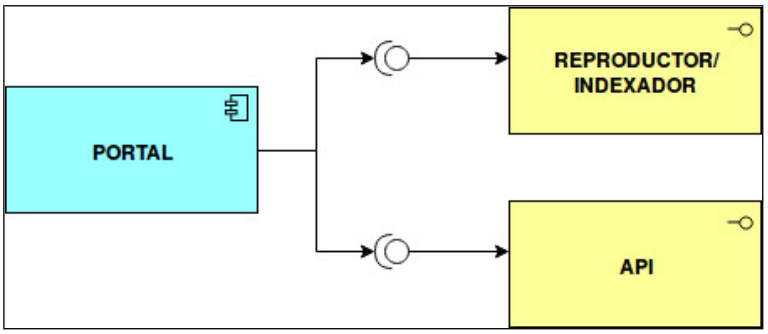


Figura 6.22: Interacción Portal con API y Reproductor/Indexador

- 5.1:

  Resultados esperados: Se visualiza un listado con los últimos Índices persistidos en el sistema.

  Resultados obtenidos:

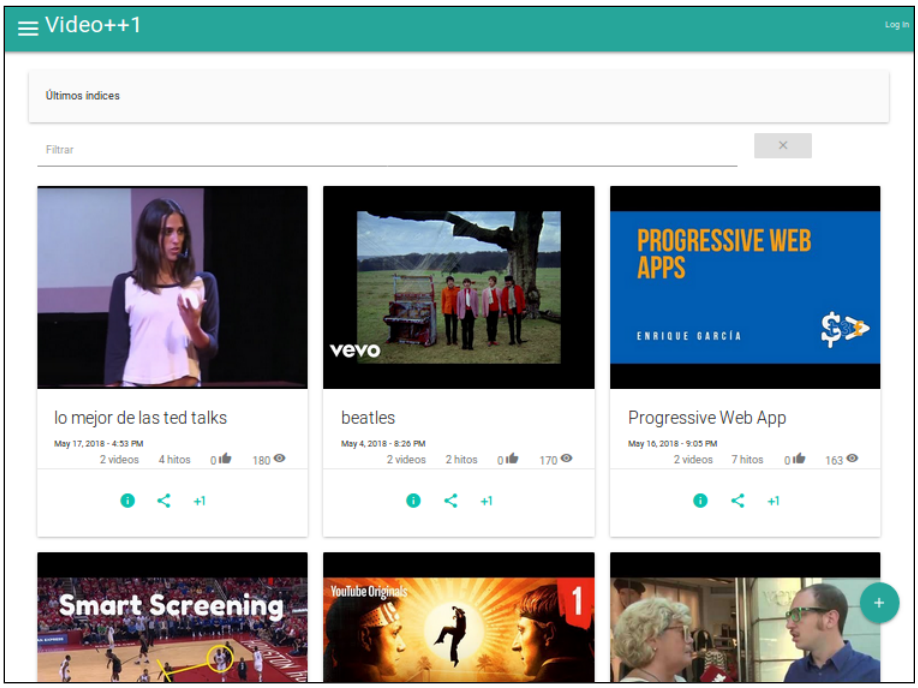


Figura 6.23: Portal con últimos índices

- 5.2:

  Resultados esperados: Se reproduce un Índice público embebido en el portal con sus Hitos y Videos asociados.

Resultados obtenidos:

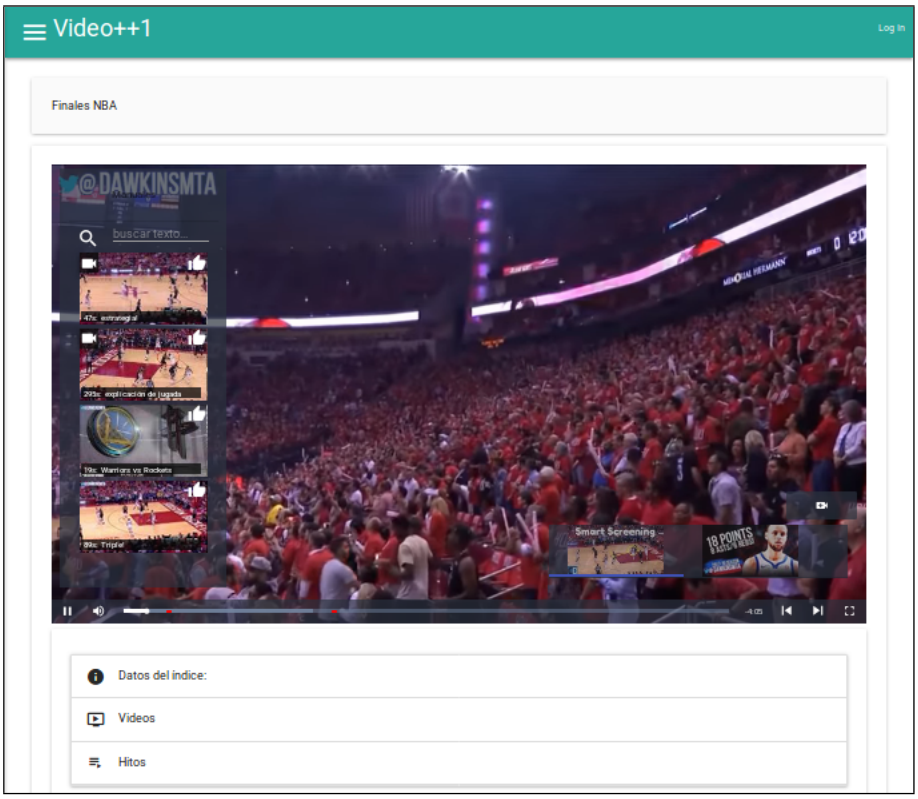


Figura 6.24: Reproductor embebido en portal

- 5.3:

Resultados esperados: Se visualiza un Índice privado embebido en el portal con sus Hitos y Videos asociados con los controles de indexación para ser modificado por su propietario.

Resultados obtenidos:

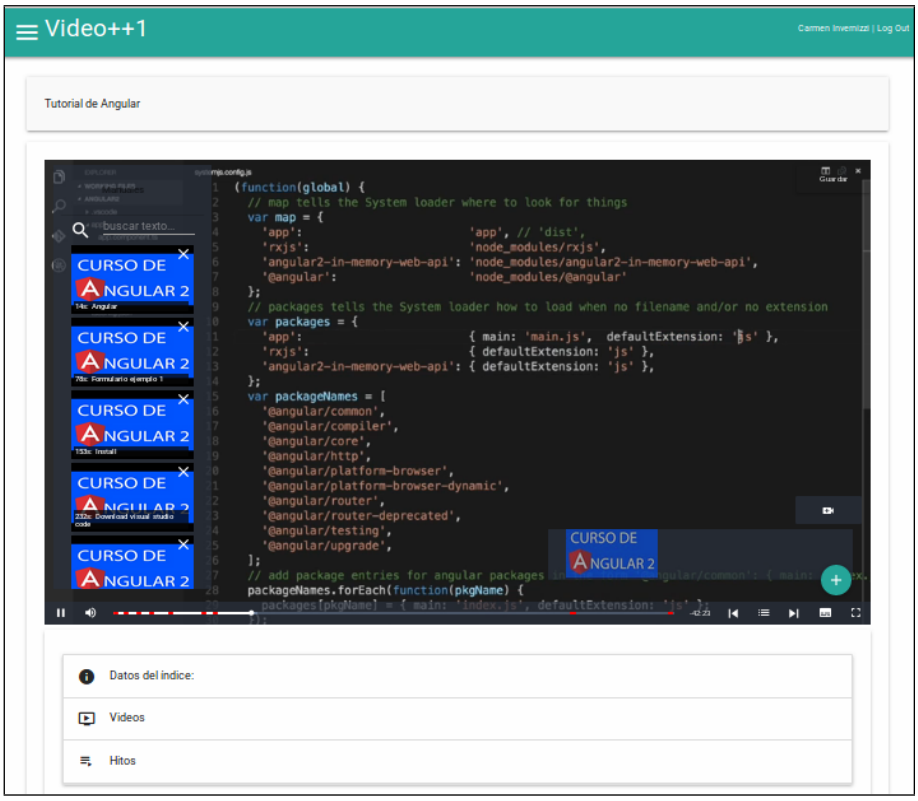


Figura 6.25: Indexador embebido en portal

- 5.4:

Resultados esperados: Se visualiza un Índice embebido en cualquier otro portal web.

Resultados obtenidos:

Figura 6.26: Indexador embebido en otro portal

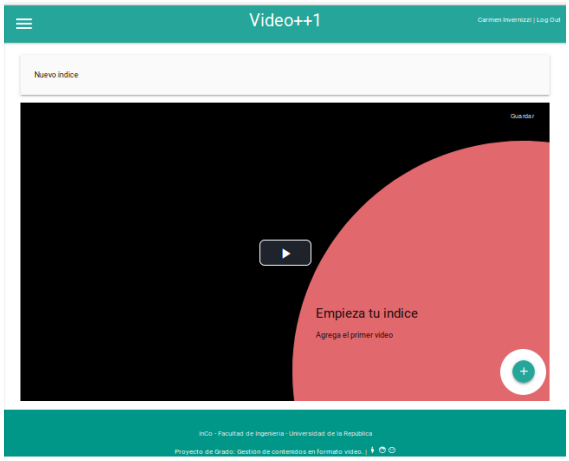
6.2.3. Pruebas de interfaz

Con estas pruebas se intenta comprobar que el funcionamiento de los distintos requerimientos es el correcto. Se intentó abarcar aquellos requerimientos más importantes haciendo énfasis en probar lo que brindara mayor valor.

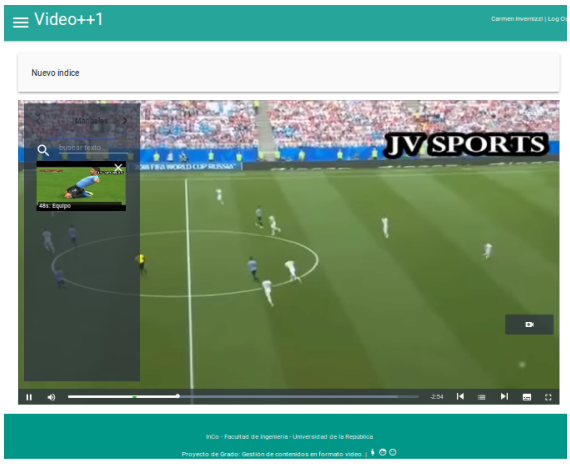
Prueba 1: *Crear Índice.*

La interfaz para crear un nuevo índice es muy intuitiva y siguiendo las pautas de Material Design con un simple botón (+) se accede a la funcionalidad de "Nuevo Índice".

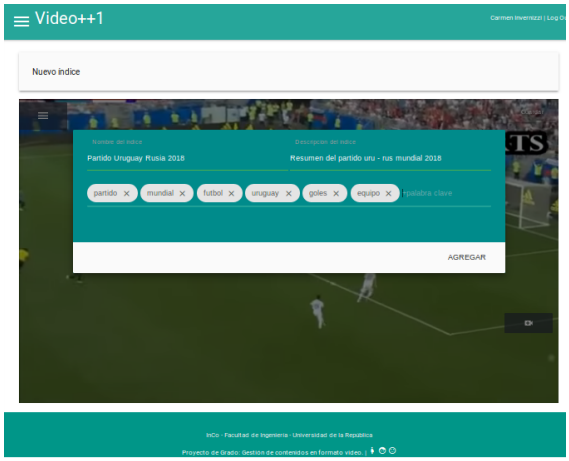
Para realizar esta prueba se accede a la funcionalidad, primero se solicita agregar un nuevo video a partir de la url y el proveedor y se agrega una marca en la línea de tiempo asignándole una descripción. En este instante, la marca creada no se encuentra aún persistida por lo tanto se muestra con un color verde como se aprecia en la imagen 6.27b.
Posteriormente, se guarda el índice compltando un simple formulario con los datos básicos "nombre", "descripción" y "palabras claves". Se chequea que las marcas son guardadas correctamente comprobando que el color del hito cambia de verde a rojo como se puede ver en la imagen 6.27d.
Por último, accediendo a la sección "Mis Índices" se observa en el listado de índices creados por el usuario, el índice que se acaba de guardar.



(a) Nuevo índice

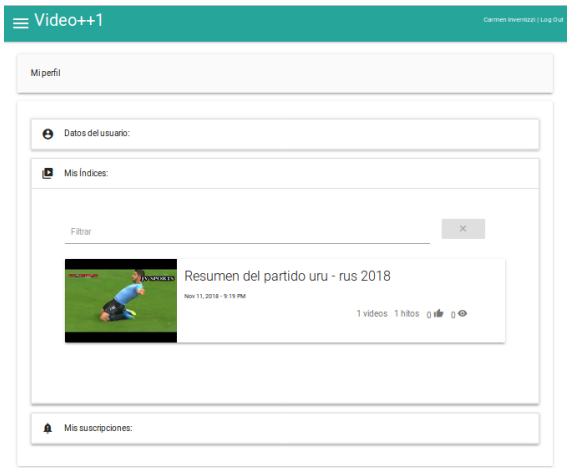


(b) Se añade nueva marca



(c) Formulario para guardar

(d) Marca e índice guardado



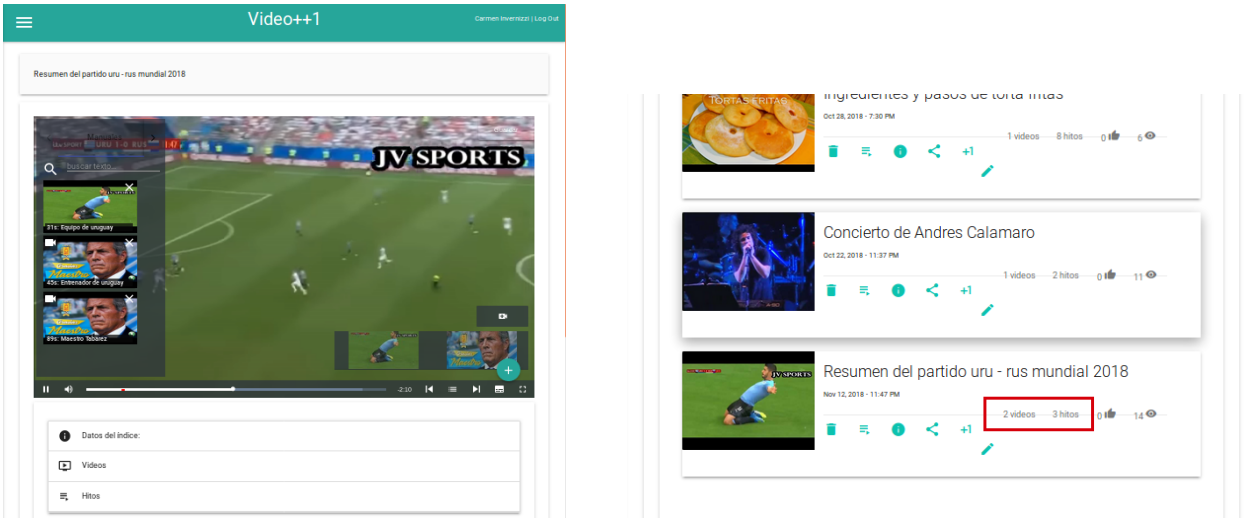
(e) Índice en Mis Índices

Figura 6.27: Secuencia que verifica correcta creación de un nuevo índice.

Prueba 2: *Modificar Índice.*

Para verificar que es posible modificar un índice y que dicha funcionalidad se realiza correctamente, se pretende realizar la prueba modificando el índice creado anteriormente.
Se comienza borrando el hito “Equipo” y se agrega un nuevo hito “Equipo de uruguay” en el mismo intervalo de tiempo (31 segundos). Además, se agrega otro video y a éste se le cargan dos nuevos hitos, “Entrenador de uruguay” a los 45 segundos y “Maestro Tabarez” a los 89 segundos.

Se guarda el índice y se comprueba que tiene dichos hitos y videos.



(a) Índice modificado (b) Índice modificado con 3 hitos y 2 videos

Figura 6.28: Secuencia que verifica correcta modificación de un índice.

Prueba 3: *Búsquedas dinámicas.*

Las búsquedas dinámicas, pretenden explorar en todos los índices creados en el sistema, y encontrar aquellos hitos que coinciden con una determinada descripción. Para probarlo, se pretende acceder a la funcionalidad y buscar por la descripción: “uru”. En la prueba de modificación de índice, se cambió el hito “Equipo” por “Equipo de uruguay” (31 seg.) y se agregó un segundo video con el hito “Entrenador de uruguay” (45 seg.). De este modo ambos hitos coinciden con el criterio buscado y deberían ser retornados.

Por otro lado se crea un nuevo índice, “Prueba dinámico” que contiene el mismo video que el índice anterior pero con el hito “otro hito entrenador uruguay” a los 44 seg. Este hito se agrega con diferencia de 1 segundo con el hito “Entrenador de uruguay” del otro índice para poder verificar que si dos hitos coinciden en un rango de tiempo configurable, solo se debe mostrar uno de ellos. Por lo tanto, como el hito “otro hito entrenador uruguay” se encuentra antes que “Entrenador de uruguay”, solo se debe mostrar el primero.

Figura 6.29: Resultado de buscar por descripción: uru

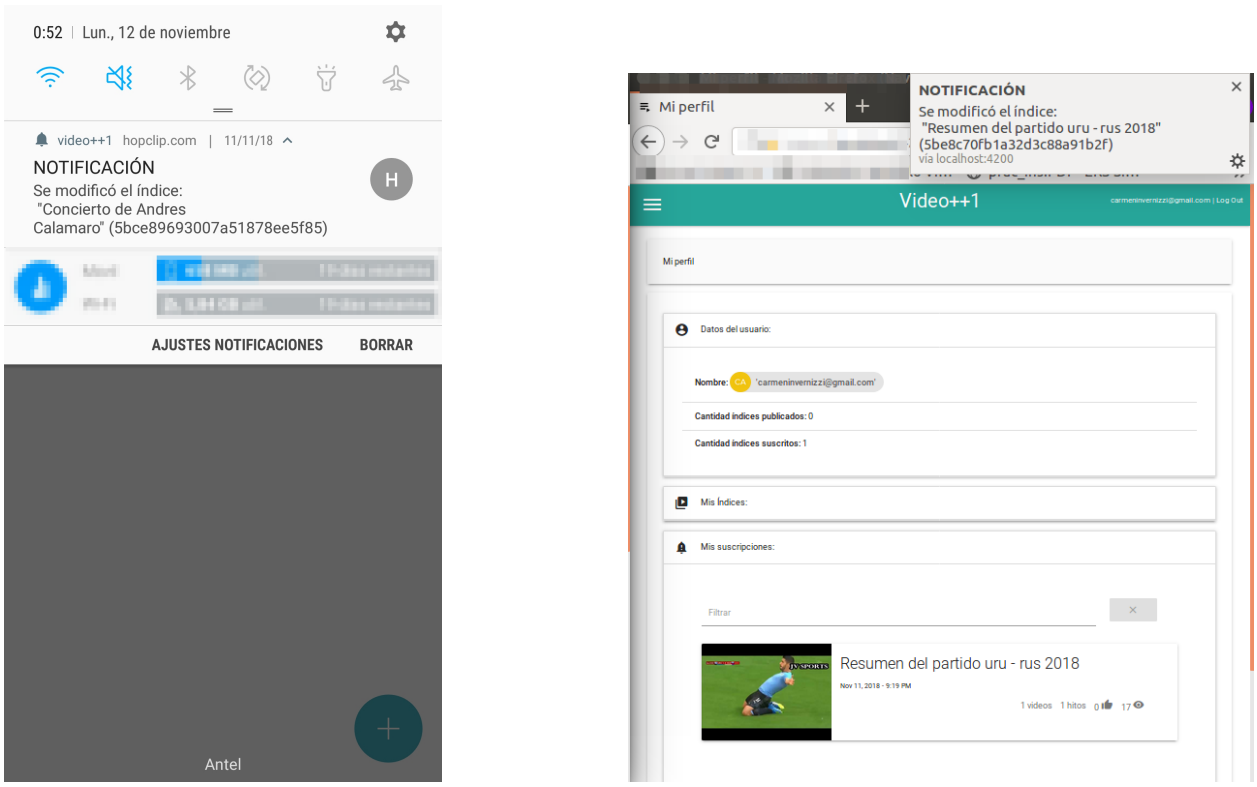
Como se puede ver en la imagen 6.29, al realizar la búsqueda solo se encontraron los hitos "Equipo de uruguay" y "otro hito entrenador uruguay" como se preveía.

Prueba 4: *Notificaciones push.*

El objetivo de esta prueba es comprobar que un usuario suscripto a un índice de otro usuario es notificado cuando dicho índice sufre algún cambio.

Para ello, se realiza la suscripción al índice "Concierto de Andrés Calamaro" desde un dispositivo móvil y por ejemplo el propietario del índice le agrega un nuevo hito. Por otra parte, el mismo usuario se suscribe desde un navegador web al índice "Resumen del partido uru - rus 2018" y ante un cambio en ese índice, el usuario también debe ser notificado.

Ambas notificaciones, tanto en la versión web como en la versión móvil, pueden comprobarse con las siguientes imágenes 6.30:



(a) Notificación móvil

(b) Notificación web

Figura 6.30: Notificaciones push

Para concluir este tipo de pruebas de interfaz, podemos decir que lo ideal es automatizar las mismas con herramientas como por ejemplo *Selenium IDE* [SEL]. Con estas herramientas, a través de la grabación de la pantalla durante su uso, se van generando scripts que ayudarán a comprobar el correcto funcionamiento de los diferentes escenarios analizando los elementos que componen la pantalla sin intervención manual.

6.2.4. Pruebas de aceptación

A continuación se comparan las distintas pantallas realizadas con los bosquejos diseñados, de modo que pueda comprobarse visualmente que el producto final se asemeja a lo planteado desde las primeras etapas.

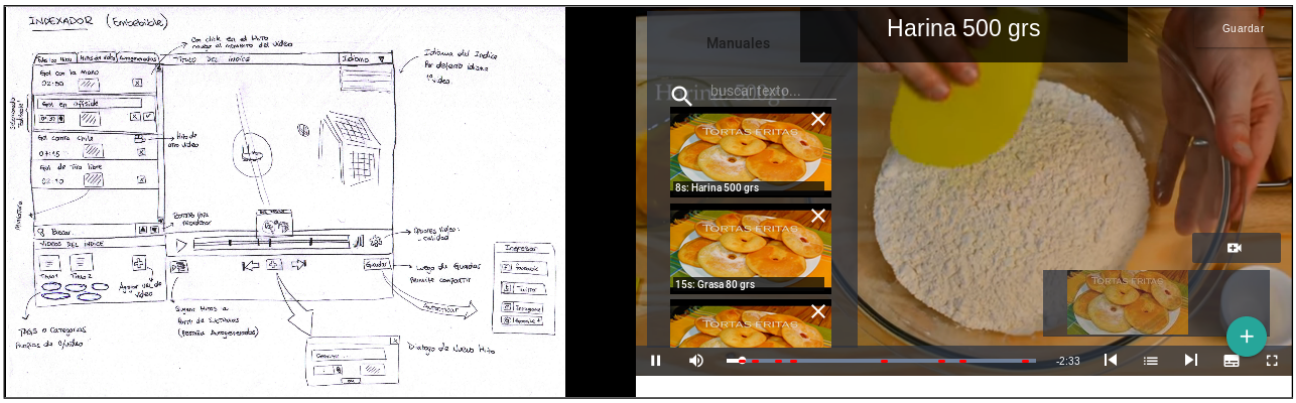


Figura 6.31: Comparación Indexador

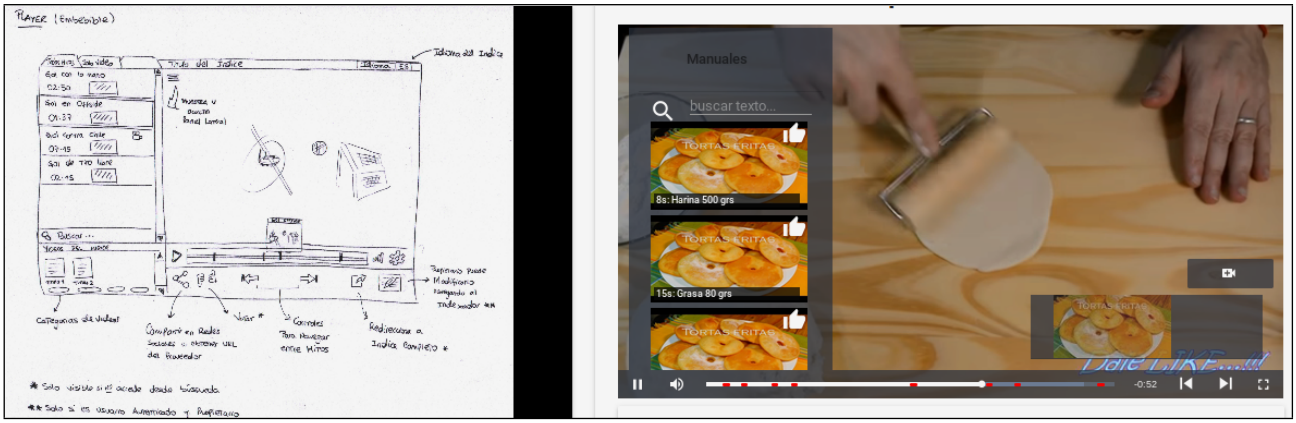


Figura 6.32: Comparación Reproductor

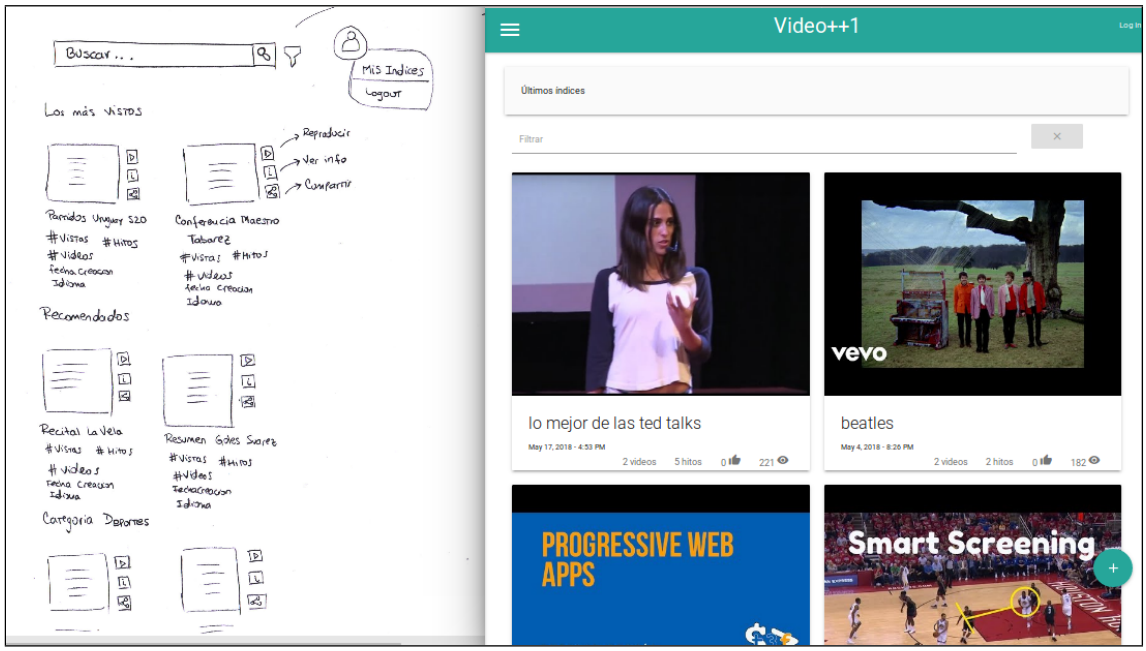


Figura 6.33: Comparación Portal: página inicio

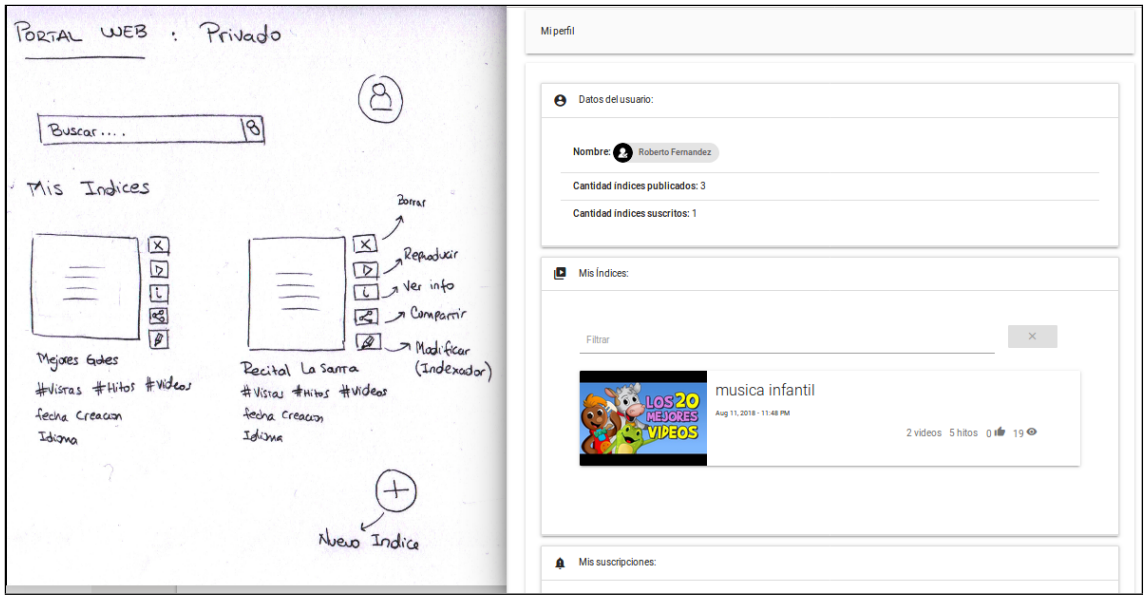


Figura 6.34: Comparación Mis Índices en Portal

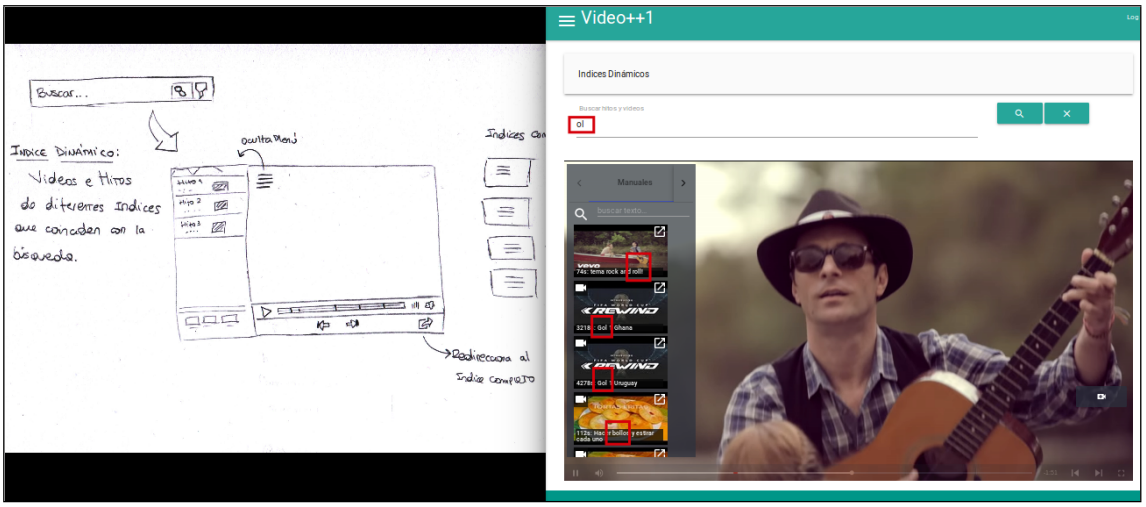


Figura 6.35: Comparacion Búsqueda de hitos

6.3. Pruebas no funcionales

Pruebas de Rendimiento:

Estas pruebas se relacionan con las pruebas de estrés y de capacidad de respuesta en condiciones de carga. Existen múltiples herramientas *open source* para realizar estas pruebas, como ser Locust [LOC], que despliega un servidor local que permite ejecutar scripts de pruebas especificando un número de usuarios determinado. A raíz de esa ejecución es posible consultar gráficos e informes de errores que ayudan a obtener un pantallazo rápido de la salud del sistema. Sin embargo, dado el tiempo requerido para generar los scripts de pruebas y que este tipo de pruebas no tenían alta prioridad, se excluyen del plan de pruebas para este proyecto.

Pruebas de Compatibilidad:

Con estas pruebas se pretende comprobar que el sistema funciona en todos los navegadores y sistemas operativos, de modo de cubrir la mayor cantidad de usuarios posible.

Para ello, se probó la aplicación tanto en Firefox, Chrome y Safari en sus versiones: 63.0, 69.0 y 12.0 respectivamente y la aplicación tuvo un comportamiento esperado.

Para el caso de los sistemas operativos, se realizaron pruebas en Android e iOS. Particularmente para iOS, no es posible reproducir el video en pantalla completa como se mencionó en el alcance del proyecto.

Pruebas de Seguridad:

Consiste en probar las características de seguridad o vulnerabilidades del sistema. Para comprobar con qué nivel de seguridad cuenta nuestro sistema utilizamos herramientas *on line* como “Qualys” [QUA] y “Observatory by Mozilla“ [OBM]. El resultado obtenido coincide en ambos casos y refleja una calificación general de grado: “B”. Un resumen de los mismos se muestra a continuación:

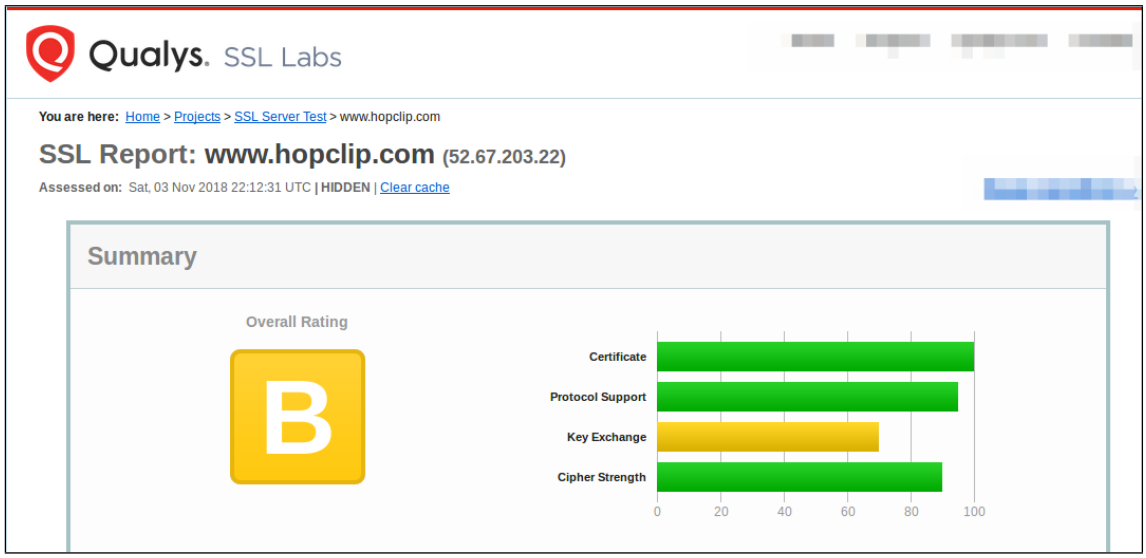


Figura 6.36: Grado de vulnerabilidad: B según Qualys

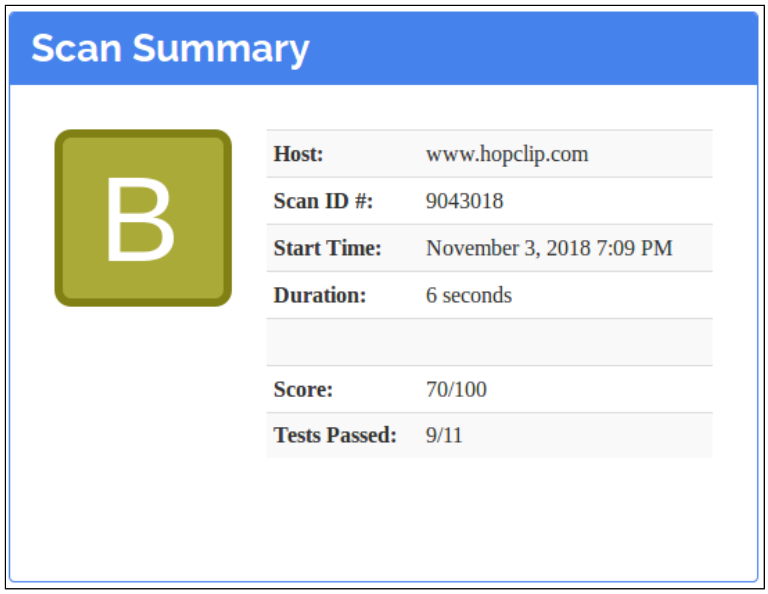


Figura 6.37: Grado de vulnerabilidad: B según Observatory

A modo de resumen, y por ejemplo a partir del reporte completo reflejado por la herramienta "Qualys" ([REP]), se puede decir que nuestra aplicación web tiene un grado aceptable de seguridad, teniendo algunas vulnerabilidades como ser el *Ataque de Logjam* contra el protocolo TLS.

Capítulo 7

Gestión del proyecto

7.1. Planificación inicial

La propuesta para desarrollar el proyecto de grado incluía el siguiente plan de trabajo inicial:

- Mes 1: Relevamiento exhaustivo de herramientas existentes para gestión de información en videos *on line*.
- Mes 2: Especificación formal del sistema a desarrollar.
- Mes 3: Estudio de tecnologías necesarias para su implementación.
- Meses 4, 5 y 6: Definición de arquitectura y diseño. Contrucción de los módulos del sistema. Integración.
- Mes 7: Validación y verificación.
- Meses 7 y 8: Ajustes finales.
- Meses 4 a 8: Elaboración de informe final.

Basándonos en nuestra propia experiencia así como en la del tutor del proyecto, estimamos que el plan de trabajo inicial era factible y se podría llevar a cabo el proyecto en el tiempo previsto, considerando una eventual prórroga.

7.2. Desarrollo

7.2.1. Dificultades encontradas

Durante el desarrollo del proyecto, surgieron ciertas dificultades y situaciones no previstas que nos obligaron a extender los plazos con el objetivo de lograr un resultado aceptable y acorde a un proyecto de grado.

Una de las principales dificultades que se presentaron desde el punto de vista técnico tuvo que ver con el tipo de las tecnologías utilizadas. Por un lado las tecnologías necesarias para lograr las funcionalidades requeridas, que implican poner mucha lógica en el dispositivo cliente, se basan casi

totalmente en *JavaScript*. Sin embargo por cuestiones prácticas, se debió investigar y utilizar *frameworks* más complejos basados en este lenguaje. Estas herramientas ofrecen mayores posibilidades pero a su vez presentan una curva de aprendizaje más pronunciada. Por otro lado, la falta de experiencia que teníamos en este tipo de desarrollo y el desconocimiento de las tecnologías implicaron una etapa de aprendizaje más extensa de lo previsto.

Otro aspecto a tener en cuenta con respecto a estas tecnologías, es su alto grado de cambio comparado con otro tipo de herramientas, lo que puede llevar a que en ciertos casos se presenten inestabilidades en los *frameworks* o librerías sobre los que nos apoyamos. Durante el desarrollo del proyecto varias especificaciones e implementaciones fueron actualizadas, ante esta característica tecnológica nos vimos obligados a realizar una constante evaluación de las herramientas utilizadas.

Los temas relacionados con el relevamiento, el aprendizaje y la elección de teconologías no había sido correctamente dimensionado al inicio del proyecto teniendo en cuenta nuestra experiencia previa.

7.2.2. Evaluación de alternativas

La evaluación de distintas herramientas a utilizar para implementar los distintos componentes del sistema, tuvo como consecuencia una extensión en los tiempos previstos para el estudio de tecnologías (principalmente en la etapa 3). También impactó en la definición del diseño y la construcción del sistema.

Se evaluaron distintas tecnologías, como manejadores de contenidos para la implementación del portal, *React* y *React Native* o *Ionic* para generar una aplicación móvil híbrida. Para implementar los componentes Indexador y Reproductor se comenzó con el desarrollo de 3 prototipos, utilizando *JavaScript* directamente, *React* y *Angular2+*.

7.2.3. Etapas realizadas

Luego de finalizado el proyecto, se puede observar que las tareas realizadas se distribuyeron a grandes rasgos de la siguiente forma, comenzando el trabajo propiamente dicho en el mes de mayo de 2017:

Etapa	**Período**	**Descripción**
Relevamiento de herramientas	mayo - junio	Se buscó información sobre herramientas existentes en Internet, realizamos una entrevista a una autoridad de un canal de televisión y se comenzó a documentar las herramientas relevadas
Especificación	julio - agosto	Mediante reuniones con el tutor y el cliente del proyecto, se especificaron los requerimientos y se ralizaron bosquejos.
Estudio de tecnologías	agosto - octubre	Se evaluaron varias herramientas como alternativas para implementar los distintos componentes del sistema.
Elaboración de prototipos	octubre - diciembre	Se finalizó con la construcción de un prototipo del componente Indexador.
Diseño y construcción	enero - junio	Se implementaron la mayoría de las funcionalidades.
Validación, verificación y correcciones	agosto - setiembre	Se definió el plan de pruebas en conjunto con el tutor y el cliente, se realizaron las pruebas más relevantes y se implementaron algunas correcciones.
Elaboración del informe	agosto - noviembre	Documentación del sistema y las tareas realizadas.

Se observa que todas las etapas insumieron un tiempo mayor al previsto originalmente, la carga horaria destinada no fue constante a lo largo del tiempo por lo que algunas etapas sufrieron retrasos más allá de las dificultades técnicas encontradas.

Decidimos dedicar un tiempo considerable a la elaboración de prototipos como forma de validar tecnologías. Por este motivo se agregó una nueva etapa que funcionó como nexo entre el estudio de tecnologías y el comienzo de la construcción. Consideramos que la realización de un prototipo funcional (*MVP*, siglas del inglés para *Minimum Viable Product*), nos permitió aplicar conceptos y generar una primer versión del principal componente del proyecto (Módulo Indexador), lo cual también aportó a la motivación del equipo.

En la planificación inicial se había previsto realizar el informe a medida que se construía el producto, pero debido a la extensión de los plazos y los inconvenientes encontrados, cambiamos esta idea inicial para posponer la elaboración del informe hasta la fase final del proyecto.

Capítulo 8

Conclusiones y trabajo a futuro

8.1. Conclusiones

Como conclusiones del trabajo realizado, podemos destacar que durante el mismo incursionamos en el uso de diferentes tecnologías que hasta el momento no conocíamos. En este sentido, podemos decir que logramos integrar varias herramientas de distinto tipo y como resultado se dejó operativa una plataforma que incluye la interacción entre distintos sistemas. Se elaboró la infraestructura necesaria, aplicando herramientas libres ampliamente utilizadas tanto en la industria como a nivel académico.

Logramos un dominio razonable de lenguajes de programación y diversas herramientas concebidas para el desarrollo de software de tipo *front-end*, tuvimos que evaluar diferentes alternativas y adaptarnos a la naturaleza cambiante de estas herramientas. Realizamos una investigación y evaluación de las últimas tendencias en el diseño de aplicaciones web, como son las *Progressive Web Applications*, el uso de notificaciones de tipo *web-push* y de *Services Workers*; así como en reproductores de video como fue el caso de *VideoJS*.

Como resultado, se cubrieron las principales funcionalidades que se requerían para el sistema. Durante la elaboración del mismo aplicamos diferentes técnicas y estrategias adquiridas en nuestra formación curricular, tanto para el diseño y construcción del software como para la gestión del proyecto en sí. También debimos consultar material complementario sobre los temas en los que se poseía menos conocimiento.

Tanto el proceso de diseño y construcción del sistema, como el de relevamiento de herramientas y antecedentes, nos llevó a tener que evaluar diferentes alternativas y valorar las mejores soluciones de acuerdo a la propuesta planteada. Encontramos diversas dificultades, principalmente relacionadas con las tecnologías utilizadas al momento de la implementación. También se puede destacar que la gestión del tiempo no se adecuó a lo planificado, por diversas razones tuvimos una dedicación horaria poco constante durante el tiempo, esto influyó en algunas decisiones sobre qué funcionalidades debían plantearse como trabajo a futuro.

Para finalizar, creemos haber logrado razonablemente los objetivos propuestos inicialmente, logrando desarrollar un producto de forma aceptable y adquiriendo competencias generales y específicas en el proceso.

8.2. Trabajo a futuro

A continuación se describen algunas funcionalidades y características que sería deseable desarrollar en el futuro para completar el trabajo realizado en este proyecto de grado:

- agregar el soporte para nuevos proveedores de videos en línea
- quitar la restricción de autenticación previa en el portal para crear nuevos índices
- finalizar ajustes estéticos y funcionalidades como la integración con redes sociales
- implementar la funcionalidad de selección del idioma de los subtítulos por parte del usuario final
- implementar notificaciones para el caso de uso donde se solicita el procesamiento de hitos automáticos (se ejecuta de forma asincrónica)
- implementar la suscripción a usuarios de modo de poder ser notificado ante cualquier evento relacionado a los índices de ese usuario.
- implementar una alternativa para que el indexador funcione en modo pantalla completa desde la plataforma iOS
- mejorar el critero de umbral en las búsqueda de hitos dinámicamente, para que ante un cluster de hitos en un rango de tiempo establecido, seleccione aquel mejor posicionado en un cierto *ranking* de puntuación y no así el primero del cluster como se realiza en el sistema actual
- automatizar las pruebas de interfaz para detectar a tiempo errores o comportamientos inesperados y no depender de las pruebas manuales

Apéndices

Apéndice A

Modelo de datos

A.1. Modelado del dominio del problema

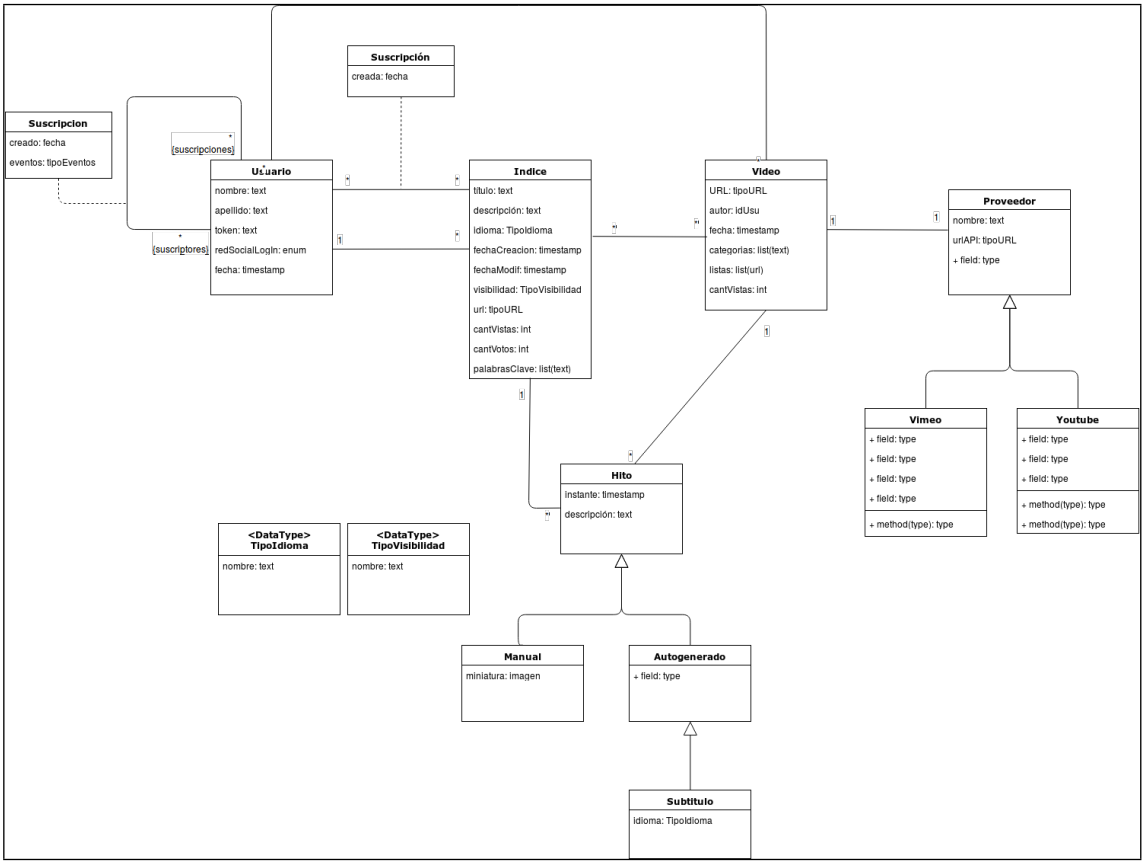


Figura A.1: Modelo de datos ER

De los requerimientos relevados y las funcionalidades diseñadas, se puede inferir que existe un concepto llamado **índice** que está formado por una lista de **hitos**.

Cada hito es un instante de tiempo en un **video**, asociado con una descripción. Un hito pertenece a un único índice, mientras que un video puede tener varios hitos asociados a distintos índices. Los

hitos que pertenecen a un mismo índice pueden estar asociados con diferentes videos.

A.1.1. Entidades

Las principales entidades del modelo de datos poseen los siguientes atributos:

Video: Representa una instancia de un video en línea. Está identificado por su URL.

- **URL**
- autor
- fechaCreación
- categorías
- listas
- cantVistas

Hito: Un instante determinado en un video. Entidad débil de Video, identificado por su instante de tiempo.

- **instante**
- descripción

Índice: Secuencia de hitos que pueden pertenecer a diferentes videos. Posee un identificador interno.

- título
- descripción
- idioma
- fechaCreación
- fechaModificación
- visibilidad: público, privado
- url
- cantVistas
- cantVotos
- palabrasClave

Usuario: Representa a los usuarios del sistema que son quienes crean índices, hitos y videos.

- nombre
- apellido
- token
- redSocialLogin
- fechaCreación

A.1.2. Representación mediante documentos de tipo JSON

Luego de sufrir modificaciones debido a cambios en requerimientos y a la elección de tecnologías, se implementó la base de datos en *mongoDB*, a continuación se muestran ejemplos de documentos de las distintas colecciones definidas:

Video:


```
{
    "_id" : ObjectId("59ee7bcb941b424673dc328c"),
    "autor" : "google-oauth2|112317912...0556043",
    "descripcion" : "descripción de un video de ejemplo",
    "titulo" : "jean claude y volvo",
    "cantVistas" : 0,
    "url" : "https://www.youtube.com/watch?v=M7FIvfx5J10",
    "categorias" : [
        "spot",
        "autos"
    ],
    "listas" : [],
    "idioma" : "en",
    "proveedor" : "youtube",
    "poster" : "https://img.youtube.com/vi/M7FIvfx5J10/0.jpg"
}
```


Hito:


```
{
    "_id" : ObjectId("59ee7cff941b424673dc328e"),
    "instante" : 4.143,
    "captura" : "https://hopclip.com/estaticas/yt/image/M7FIvfx5J10/00-00-05.jpg",
    "descripcion" : "una descripción ahí",
    "idVideo" : "ObjectId('59ee7bcb941b424673dc328c')"
}
```


Indice:


```
{
  "_id" : ObjectId("59ee7ed5941b424673dc3290"),
  "userId" : "auth0|59d13a7e6...64737",
  "descripcion" : "la descripción del primer índice",
  "titulo" : "tremendo índice",
  "cantVistas" : 21,
  "cantVotos" : 0,
  "visibilidad" : "publico",
  "palabrasClave" : [
      "deporte",
      "goles"
  ],
  "hitos" : [
```

```
      ObjectId("59ee7d1a941b424673dc328f"),
      ObjectId("59ee7cff941b424673dc328e")
  ],
  "videos" : [
      ObjectId("59ee7bcb941b424673dc328c")
  ],
  "url" : "https://hopclip.com/api/indice/1",
  "idioma" : "es",
  "fechaModificacion" : ISODate("2018-05-29T02:39:22.973Z"),
  "fechaCrecion" : ISODate("2017-10-23T00:00:00.000Z"),
  "suscriptos" : [
      {
          "keys" : {
              "p256dh" : "BOgCb4bsGg7yB_N3dfP_Xm...",
              "auth" : "ZODizVpOHAk...PWCySQ"
          },
          "_id" : ObjectId("5a83267e1412aa05ecac59f7"),
          "endpoint" : "https://updates.push.services.mozilla.com/wpush/v1/gA...jW-v"
      }
  ],
  "fechaCreacion" : ISODate("2018-05-29T02:39:22.961Z")
}
```

Users:

```
{
    "_id" : ObjectId("5a0e2dd8187861b586a2033e"),
    "userId" : "google-oauth2|112317...56043",
    "name" : "Diego Martinez",
    "nickname" : "diegom702",
    "picture" : "https://lh4.googleusercontent.com/-eLq4SYP...FRHryl4/photo.jpg",
    "email" : null,
    "indicesSus" : [
        ObjectId("59ee7ed5941b424673dc3290")
    ]
}
```

Autogenerados:

```
{
    "_id" : ObjectId("5af0d7f0625484681a2201e4"),
    "nombre" : "video++2",
    "url" : "https://proyecto-de-grado-183619.appspot.com/get_video_analysis"
}
```

SolicitudesHitos:

```
{
    "_id" : ObjectId("5aefdb35811dd361e3d5b3a4"),
    "ticketID" : "918105293",
```

```
    "videoURL" : "https://www.youtube.com/watch?v=M7FIvfx5J10",
    "idVideo" : ObjectId("59ee7bcb941b424673dc328c"),
    "idIndice" : ObjectId("59ee7ed5941b424673dc3290"),
    "fechaSol" : ISODate("2018-05-07T04:51:01.887Z"),
    "estado" : "DONE",
    "idiomaVideo" : "es-ES"
}
```

Apéndice B

Implementación para obtener miniaturas de videos de YouTube

B.1. Introducción

Desde el comienzo del proyecto se planteó la necesidad de poder incluir miniaturas de instantes de un video de *YouTube* para distintos usos dentro del Indexador o el Portal. Esta funcionalidad no es ofrecida por *YouTube* desde su API, sino que está implementado directamente en su reproductor.

Se consultó diversos proyectos de software libre donde se utilizan estas miniaturas, como *youtube-dl* (`https://rg3.github.com/youtube-dl/`), en esta y otras implementaciones se utiliza un recurso denominado *Youtube Storyboard*.

Este recurso consiste en una secuencia de imágenes asociadas a distintos instantes de tiempo del video, no están disponibles para todos los videos de *YouTube*, pero cuando sí lo están son generadas a priori. La cantidad de imágenes que posee cada video depende de la duración del mismo, para un video relativamente corto existirá una imagen por segundo, mientras que para videos más extensos es común encontrar una imagen cada 5 o 10 segundos. Esta funcionalidad no está documentada, por lo tanto todas las implementaciones disponibles para obtenerlas se basan en métodos conocidos como "ingeniería inversa".

Decidimos basarnos en una implementación escrita en lenguaje *PHP* que nos fue proporcionada por el tutor del proyecto. Esta implementación funcionó desde el comienzo de la etapa de implementación, pero en noviembre de 2018 dejó de funcionar debido a que las URL donde se accede a los *storyboard* fueron restringidas.

Por este motivo y debido a la alta prioridad para contar con esta funcionalidad que manifestaron tanto el tutor como el cliente del proyecto, tuvimos que realizar una implementación alternativa. Esta última implementación se describe a continuación.

B.2. Implementación en la API

Decidimos agregar dos operaciones en la API, por lo tanto, la funcionalidad fue implementada utilizando *NodeJS* y *ExpressJS*. Se agregaron dos operaciones que serán invocadas desde el componente Indexador:

- generarCapturas (idVideo)
- obtenerCaptura (idVideo, instante)

Considerando el corto plazo con el que se contaba para implementar nuevamente la funcionalidad, se optó por utilizar el método más directo y efectivo, que consiste en descargar el video en la peor calidad disponible y sin sonido; para luego poder extraer imágenes arbitrarias del mismo. Para estos dos pasos se utilizaron las herramientas *youtube-dl* y el programa *Ffmpeg* (`https://ffmpeg.org/`).

En ambas operaciones se utilizó la función *spawn* del módulo *child_process*, para lanzar en procesos independientes la ejecución de los comandos *youtube-dl* y *ffmpeg*. Se ignora el uso de la entrada y la salida estándar, ya que sólo se necesita ejecutar el proceso y obtener el código de salida. Con el método *on* se incluye el código que debe ser ejecutado al finalizar el comando.

B.2.1. generarCapturas

```
1  app.get('/api/yt/generarCapturas/:id', jwtCheck, (req, res) => {
2    const dir = '/var/www/yt/image';
3
4    if (!req.params.id) {
5      console.log(' generarCapturas: El pedido vino sin ID de video.');
6      return res.status(500).send({message: err.message});
7    }
8
9    var idVideo = req.params.id;
10   var dirVid = dir + '/' + idVideo;
11   var destino = dirVid+'/'+idVideo+'.mp4'
12   if (!fs.existsSync(dirVid)){
13       fs.mkdirSync(dirVid);
14   }
15
16   const opciones = { stdio: 'ignore', cwd: dirVid };
17   const url = 'https://www.youtube.com/watch\?v\=' + idVideo;
18   const ytdl = spawn('youtube-dl',
19       ['-f', 'worstvideo', '-o', destino, '-q', url], opciones);
20
21   // bajó el video
22   ytdl.on('close', (code) => {
23     if (code === 0)
24       fs.open(dirVid+'/hecho', 'w', (err, fd) => {
25         if (err) throw err;
26         fs.close(fd, err => { if (err) throw err; });
27         return res.status(200).send({message: 'capturas disponibles'});
28       });
29     else return res.status(500).send({message: `error en ytdl: ${code}`});
30   });
31 });
```

Listing B.1: generarCapturas

Se utilizaron las siguientes opciones del comando *youtube-dl*:

-f worstvideo: Para minimizar el tiempo de descarga, se elige la peor calidad en modo "solo video". Se realizaron algunas pruebas para determinar si la peor calidad implicaba el menor

tiempo de descarga, ya que *YouTube* utiliza un ancho de banda adaptativo; es decir que a mayor volumen de datos la tasa de transferencia es mayor.

-o: Indica dónde se ubicará el video descargado.

-q: Elimina la impresión de información durante la ejecución del comando.

B.2.2. obtenerCaptura

```
1  app.get('/api/yt/obtenerCaptura/:id/:tiempo', jwtCheck, (req, res) => {
2    const dir = '/var/www/yt/image';
3
4    if (!req.params.id) {
5      console.log(' obtenerCaptura: El pedido vino sin ID de video.');
6      return res.status(500).send({message: err.message});
7    }
8    if (!req.params.tiempo) {
9      console.log(' obtenerCaptura: El pedido vino sin tiempo.');
10     return res.status(500).send({message: err.message});
11   }
12   var regExp = /^[0-9]+$/;
13   if (!regExp.test(req.params.tiempo)) {
14     console.log(' obtenerCaptura: El parámetro tiempo debe ser una cantidad en
         segundos.');
15     return res.status(500).send({message: err.message});
16   }
17
18   var idVideo = req.params.id;
19   var tiempo = timetohs(req.params.tiempo);
20   var dirVid = dir + '/' + idVideo;
21
22   const opciones = { stdio: 'ignore', cwd: dirVid };
23   const arch = dirVid +'/'+ idVideo +'.mp4';
24   const temp = arch +'.ytdl';
25   var captura = tiempo.replace(/:/g,'-') + '.jpg';
26
27   if (!fs.existsSync(dirVid)) {
28     let host = process.env.NODE_ENV === 'dev' ? 'localhost' : 'hopclip.com';
29     request({
30       headers: { 'Content-Type': 'application/json' },
31       uri: 'https://'+host+'/api/yt/generarCapturas/'+idVideo,
32       method: 'GET'
33     }, (err, resp, body) => {
34         if (!err && resp.statusCode == 200) {
35           checkExistsWithTimeout(dirVid, 4000);
36           console.log(' generarCapturas bien desde obtener', body);
37         }
38         else console.log(' error al llamar a generarCapturas: ', err, body);
39     });
40   }
41
42   if (fs.existsSync(dirVid+'/'+captura)) {
43     res.sendFile(dirVid+'/'+captura);
44     return;
45   }
46   else if (fs.existsSync(temp)) {
47     var watcher = fs.watch(temp, (eventType, filename) => {
48       if (eventType === 'rename'){
```

```
49          watcher.close();
50          if (fs.existsSync(arch)){
51            const ffmpeg = spawn('ffmpeg', ['-loglevel', '-8', '-r', '1', '-ss',
52                                tiempo, '-i', arch, '-frames:v', '1', captura],
53                                opciones);
54            // genero una captura
55            ffmpeg.on('close', (code) => {
56              if (code === 0){
57                if (fs.existsSync(dirVid+'/'+captura)) {
58                  res.sendFile(dirVid+'/'+captura);
59                }
60              }
61              else return res.status(500).send({message: `error en ffmpeg: ${code
                    }`});
62            });
63          }
64        }
65      });
66    }
67    // si ya estaba bajado el video:
68    if (fs.existsSync(arch)){
69      const ffmpeg = spawn('ffmpeg', ['-loglevel', '-8', '-r', '1', '-ss',
70                          tiempo, '-i', arch, '-frames:v', '1', captura],
71                          opciones);
72      // genero una captura
73      ffmpeg.on('close', (code) => {
74        if (code === 0){
75          if (fs.existsSync(dirVid+'/'+captura)) {
76            res.sendFile(dirVid+'/'+captura);
77          }
78        }
79        else return res.status(500).send({message: `error en ffmpeg: ${code}`});
80      });
81    }
82    else return res.status(500).send({message: `no existen capturas ni se pueden
        generar: ${idVideo}`});
83  });
```

Listing B.2: obtenerCaptura

Esta operación revisa la existencia del directorio asociado al video, para determinar si éste ya fue descargado. En caso de no existir, se realiza una invocación a la operación *generarCapturas* y se espera durante un intervalo predefinido que se genere el archivo.

Si la imagen asociada al instante de tiempo solicitado ya existe, se devuelve inmediatamente. Sino, en caso de detectar que el video aún se está descargando, se utiliza la función *fs.watch* para esperar que finalice la descarga.

Luego se invoca el comando *ffmpeg*; cabe destacar que el uso del parámetro *-ss* implica una mejora de rendimiento ya que evita procesar todo el video hasta el instante solicitado. Las imagenes generadas se mantienen en el servidor para mejorar el rendimiento de la funcionalidad en las siguientes invocaciones.

Apéndice C

Instalación del sistema en AWS

C.1. Tecnologías y software de base

El sistema consta de los siguientes componentes:

- API
- Portal Web/Móvil
- Indexador/Reproductor

y se utiliza el siguiente software de base:

- servidor HTTP Nginx; versión: 1.10.3 (Ubuntu)
- manejador de BD mongoDB; versión: 3.4.10 (linux x86_64 vanilla)
- NodeJS; versión: 8.9.1 (vanilla)
- manejador de procesos para NodeJS PM2; versión: 2.10.1
- NPM; versión: 5.7.1

Se decidió utilizar la plataforma EC2 de *Amazon Web Services* ya que permite instalar un servidor virtual con todo el software necesario, admitiendo un alto nivel de configuración por parte del usuario.
Decidimos utilizar las mismas versiones para todo el software de base, en los ambientes de desarrollo y en el servidor de *AWS* para evitar problemas de compatibilidad. Para hacer esto, identificamos las versiones en nuestros ambientes locales e instalamos esas versiones en el servidor descargando el paquete adecuado (a los paquetes instalados de esta forma se los conoce como "vanilla").

C.1.1. MongoDB

Se descomprime el paquete obtenido desde el sitio oficial de *MongoDB* en el directorio *$HOME* de nuestro usuario de conexión. Dentro se encuentra el subdirectorio `bin` que contiene todos los ejecutables necesarios. Para que la instalación quede disponible por defecto para todo el sistema, se crea el enlace simbólico: `/usr/bin/mongod ->/home/ubuntu/mongodb-linux-x86_64-3.4.10/bin/mongod`.

Para controlar el servicio se creó una unidad para `systemd`:

```
ubuntu@ip-172-31-28-71:~$ systemctl status mongodb.service
● mongodb.service - An object/document-oriented database
   Loaded: loaded (/lib/systemd/system/mongodb.service; disabled; vendor preset: enabled)
   Active: active (running) since Tue 2018-05-15 01:25:50 UTC; 5 months 0 days ago
     Docs: man:mongod(1)
 Main PID: 2316 (mongod)
    Tasks: 23
   Memory: 87.4M
      CPU: 12h 48min 4.422s
   CGroup: /system.slice/mongodb.service
           └─2316 /usr/bin/mongod --unixSocketPrefix=/run/mongodb --config /etc/mongodb.conf

Warning: Journal has been rotated since unit was started. Log output is incomplete or unavailable.
ubuntu@ip-172-31-28-71:~$
ubuntu@ip-172-31-28-71:~$ systemctl cat mongodb.service
# /lib/systemd/system/mongodb.service
[Unit]
Description=An object/document-oriented database
Documentation=man:mongod(1)

[Service]
User=mongodb
Group=mongodb
RuntimeDirectory=mongodb
RuntimeDirectoryMode=0755
EnvironmentFile=-/etc/default/mongodb
Environment=CONF=/etc/mongodb.conf
Environment=SOCKETPATH=/run/mongodb
ExecStart=/usr/bin/mongod --unixSocketPrefix=${SOCKETPATH} --config ${CONF} $DAEMON_OPTS

[Install]
WantedBy=multi-user.target
```

Figura C.1: Servicio de systemd para mongodb.

Utilizando el programa `mongorestore` se cargaron los datos iniciales en la nueva base de datos, mediante *mongodump* se pueden generar respaldos del contenido actual.

C.1.2. NodeJS

La instalación de *NodeJS* es muy similar a la de *MongoDB*, se extrajo el contenido del archivo comprimido obtenido desde el sitio oficial del proveedor y se creó un enlace simbólico para el ejecutable `node` desde `/usr/bin`. En este caso no se creó una unidad para `systemd`, sino que el proceso es lanzado mediante el manejador de procesos de *NodeJS*, *PM2* [PM2] instalado mediante `npm`. La ubicación del ejecutable de *PM2* se agregó a la variable de entorno `$PATH`, en el archivo `~/.profile`.

C.2. Instalación de los componentes

Se deben copiar los distintos componentes en alguna ubicación adecuada dentro del sistema de archivos, se intentó respetar los directorios utilizados típicamente como `/var/www` para los sitios web públicos.

C.2.1. API

En un directorio en el que se tenga control total, se crea el subdirectorio `logica/`, dentro se incluye el archivo principal `server.js` y el directorio `server/`. En el directorio `logica/server/` se encuentra la implementación de la API (archivos: `api.js` y `config.js` y la definición del modelo de datos en el directorio `modelos/`).

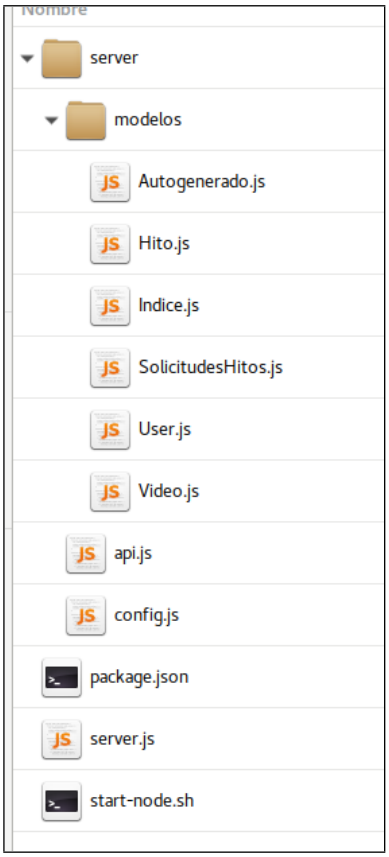


Figura C.2: Directorios API.

Como se explica más adelante en la configuración del servidor HTTP, los pedidos que llegan al servidor son atendidos por el proceso de *NodeJS* que implementa la API. Por este motivo, el archivo `server.js` contiene el código necesario para redirigir los pedidos que no correspondan a solicitudes de la API, hacia el portal:

```
[...]
/*
 |--------------------------------------
 | Routes
 |--------------------------------------
 */

require('./server/api')(app, config);

// Se envía el pedido al sitio (index), solo en ambiente de producción
if (process.env.NODE_ENV !== 'dev') {
  app.get('*', function(req, res) {
    res.sendFile(path.join(__dirname, '/index.html'));
  });
}
```

```
[...]
```

En el directorio `logica` donde se encuentra el archivo principal `server.js`, se crea el enlace simbólico: `index.html ->/var/www/portal/index.html`. Para controlar la ejecución de la *API*, se utiliza el manejador de *NodeJS PM2* de la siguiente forma:

```
# iniciar el proceso principal de la API
$ pm2 start server.js --name "API"
...
```

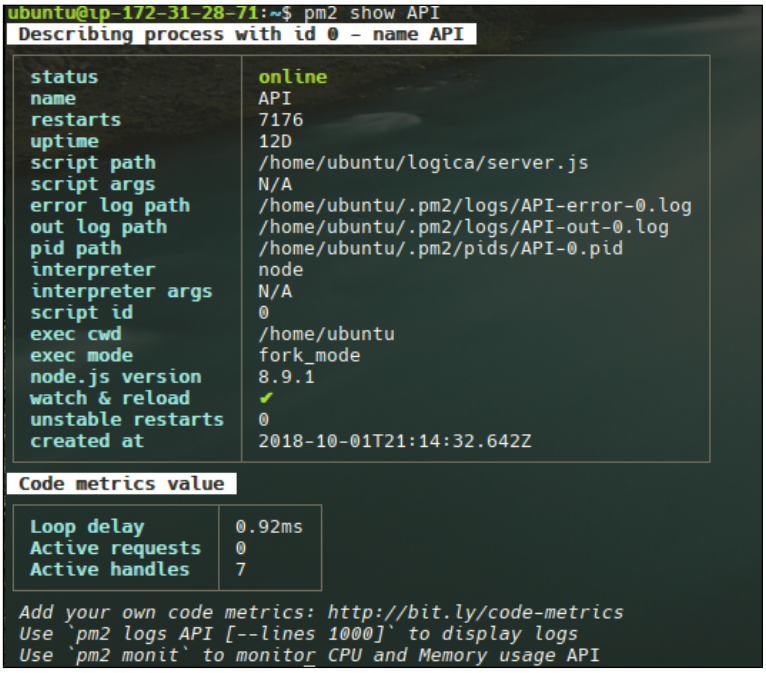


Figura C.3: Información de PM2 sobre el proceso server.js (API).

C.2.2. Sitios web

Tanto el Portal, como el componente Indexador/Reproductor, son aplicaciones hechas en *Angular2+*. Por lo tanto para generar versiones instalables, se ejecuta el siguiente comando:

```
# en el directorio donde se encuentra cada proyecto
$ ng build --prod --env=prod -bh /indexador/

$ ng build --prod --env=prod -bh /portal/
```

El resultado se genera en el directorio `dist`, el contenido de este directorio es copiado a `/var/www/portal` o `/var/www/indexador` según corresponda en el servidor de *AWS*.

C.2.3. Configuración del servidor HTTP Nginx

Primero se configura el nombre de dominio y se redirige todo el tráfico HTTP hacia HTTPS.

```
# /etc/nginx/conf.d/portal.conf
server {
  if ($host = www.hopclip.com) {
    return 301 https://$host$request_uri;
  } # managed by Certbot

  if ($host = hopclip.com) {
    return 301 https://$host$request_uri;
  } # managed by Certbot

  listen 80;
  listen [::]:80;
  server_name hopclip.com www.hopclip.com;
  return 301 https://$server_name$request_uri;
}
```

Se configura el sitio para que esté disponible mediante HTTPS, especificando los nombres de dominio y el certificado digital del sitio, que fue obtenido desde *Let's Encrypt* (`https://letsencrypt.org/`) y es gestionado por *certbot* (`https://certbot.eff.org/`). Luego, se configura a la ubicación raíz del sitio como un proxy reverso (*Reverse Proxy*) para redirigir los pedidos hacia la API, este proceso se ejecuta en el propio servidor y escucha solicitudes en el puerto 8082.

```
server {
  listen 443 ssl http2;
  listen [::]:443 ssl http2;
  server_name hopclip.com www.hopclip.com;

  # SSL certificate
  ssl_certificate /etc/letsencrypt/live/hopclip.com/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/hopclip.com/privkey.pem;


  location / {
    proxy_pass https://localhost:8082/;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-NginX-Proxy true;
```

```
    proxy_set_header Host $http_host;
    proxy_ssl_session_reuse off;
    proxy_cache_bypass $http_upgrade;
    proxy_redirect off;
  }

  location /robots.txt {
    return 301 https://hopclip.com/portal/robots.txt;
  }


  location /portal {
    alias /var/www/portal/;
    try_files $uri $uri/ /index.html;
    add_header X-Proxy-Cache $upstream_cache_status;
  }

  location /indexador {
    root /var/www;
    index index.html;
    add_header X-Proxy-Cache $upstream_cache_status;
  }

  location /reproductor {
    root /var/www;
    add_header X-Proxy-Cache $upstream_cache_status;
  }
}
```

Finalmente, se definen las ubicaciones para el portal, el indexador y el reproductor. En el caso del reproductor, se crea un enlace simbólico desde `/var/www/reproductor` hacia `/var/www/indexador`. El componente Indexador tiene la lógica necesaria para identificar cuándo debe comportarse como indexador y cuándo como reproductor.

C.2.4. Redirección desde hopyoutube.com

Como estrategia para facilitar el uso del servicio de indexación, se agrega una redirección desde el dominio `hopyoutube.com` hacia nuestro sitio, `hopclip.com`. La idea es que si un usuario está viendo un video en *YouTube*, pueda llegar fácilmente al indexador agregando la palabra "hop" como prefijo de la URL en la que se encuentra.

```
server {
  if ($host = www.hopyoutube.com) {
```

```
    return 301 https://$host$request_uri;
  } # managed by Certbot

  if ($host = hopyoutube.com) {
    return 301 https://$host$request_uri;
  } # managed by Certbot

  listen 80;
  server_name hopyoutube.com www.hopyoutube.com;
  return 404; # managed by Certbot
}

## hopyoutube.com
server {
  listen 443 ssl http2;
  server_name hopyoutube.com www.hopyoutube.com;
  ssl_certificate /etc/letsencrypt/live/hopclip.com/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/hopclip.com/privkey.pem;

# procesa las URLs del tipo: hopyoutube.com/embed/M7FIvfx5J10?...
  rewrite ^/embed/(.*)$ https://hopclip.com/portal/idx/?i=new&v=$1 last;

# procesa las URLs del tipo: hopyoutube.com/watch/?v=<ID_VIDEO>&...
  location /watch {
    root /var/www/hopclip;
    if ($arg_v) {
      return 301 https://hopclip.com/portal/idx/?i=new&v=$arg_v;
    }
    return 403;
  }
}
```

Apéndice D

Implementación de PWA

D.1. Metadatos (manifiesto)

Archivo `manifest.webmanifest` ubicado en la raíz del sitio:

```
{
  "gcm_sender_id": "236060841049",
  "name": "Portal Video++1",
  "short_name": "video++1",
  "start_url": ".",
  "scope": ".",
  "manifest_version": 3,
  "version": "3",
  "lang": "es",
  "display": "standalone",
  "orientation": "any",
  "background_color": "#26a69b",
  "theme_color": "#26a69a",
  "description": "Gestión de contenidos en formato video.",
  "icons": [{
    "src": "./assets/images/material/ic_video_library_black_36px.svg",
    "sizes": "36x36"
  },
  {
    "src": "./assets/images/icono-indice-1.png",
    "sizes": "512x512"
  }
],
  "related_applications": [{
    "platform": "play",
    "url": "https://play.google.com/store/apps/details?id=org.mozilla.firefox"
  }]
}
```

D.2. Service Workers

D.2.1. Gestión de las notificaciones

Archivo `assets/js/service-worker.js` encargado de manejar las notificaciones en el dispositivo del cliente:

```
// recibir eventos push desde el servidor y mostrar la
//  notificación correspondiente
self.addEventListener('push', function(event) {
  var payload = event.data ? event.data.text() : 'SIN payload';
  console.log(payload);

  event.waitUntil(
    self.registration.showNotification('NOTIFICACIÓN', {
      body: payload,
      requireInteraction: true,
    })
  );
});
```

D.2.2. Gestión del cache

Archivo `sw.js` ubicado en la raíz del sitio, donde se define el uso del contenido *off line*:

```
dominio = self.location.hostname;

var CACHE_NAME = 'portal-cache-v1';
var urlsToCache = [
  '/',
  '/assets/css/materialize.min.css',
  '/assets/js/materialize.min.js',
  '/favicon.ico',
  '/index.html'
];

self.addEventListener('install', function(event) {
  if (dominio !== 'localhost') urlsToCache.push(['/portal','/indexador']);
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(function(cache) {
        console.log('cache abierto');
        return cache.addAll(urlsToCache);
      })
```

```
  );
});


self.addEventListener('fetch', function(event) {
  event.respondWith(
    caches.match(event.request)
      .then(function(response) {
        if (response) {
          return response;
        }

        var fetchRequest = event.request.clone();

        return fetch(fetchRequest).then(
          function(response) {
            if(!response || response.status !== 200 ||
                response.type !== 'basic') {
              return response;
            }

            var responseToCache = response.clone();

            caches.open(CACHE_NAME)
              .then(function(cache) {
                cache.put(event.request, responseToCache);
              });

            return response;
          }
        );
      })
    );
});

self.addEventListener('activate', function(event) {

  var cacheWhitelist = ['portal-cache-v1'];

  event.waitUntil(
    caches.keys().then(function(cacheNames) {
      return Promise.all(
        cacheNames.map(function(cacheName) {
          if (cacheWhitelist.indexOf(cacheName) === -1) {
            return caches.delete(cacheName);
          }
```

```
        })
      );
    })
  );
});
```

Glosario

API *Application Programming Interface*: Conjunto de definiciones de operaciones, subrutinas, funciones, estructuras de datos, etc; cuyo objetivo es brindar los bloques básicos de construcción para desarrollar algún tipo de software a partir de ellos.

Ataque de Logjam Vulnerabilidad de las conexiones TLS que a través del ataque Man in The Middle y de claves débiles de longitud 512 bits podrían espiar y modificar datos cifrados intercambiados en dichas conexiones.

CMS *Content Management System*

Impedance Mismatch *Object-Relational impedance mismatch*: Término utilizado en ciencias de la computación para referirse a los problemas técnicos y conceptuales que surgen cuando se representan datos modelados mediante la orientación a objetos, en bases de datos relacionales.

JSON *JavaScript Object Notation*: Formato de documento de estándar abierto que utiliza texto para transmitir objetos formados por pares de clave-valor y arreglos de distintos tipos, los documentos de una colección pueden tener campos diferentes.

Middleware *Software* que provee servicios a aplicaciones pero que corre sobre otra plataforma. En el contexto de este proyecto se utilizaron funciones *middleware* en *nodeJS* para implementar el *backend* o *API* del sistema.

PWA *Progressive Web Application*: Aplicación WEB que utiliza ciertas tecnologías para comportarse como una aplicación móvil en determinados casos. Se ejecuta dentro de un navegador web pero ofrece notificaciones *push*, funciona sin conexión, es instalable, *responsive*, tiene un estilo tipo App nativa, etc.

Responsive Una aplicación o web *responsive* se adapta al dispositivo del usuario para ofrecer la mejor experiencia de uso posible, utilizando las ventajas de cada dispositivo.

REST *REpresentational State Transfer*: Estilo de arquitectura que define un conjunto de restricciones para la creación de servicios web. Los servicios web de tipo REST, permiten el acceso y manipulación de representaciones textuales de recursos web mediante el uso uniforme de operaciones sin estado predefinidas (como los métodos HTTP GET, PUT, POST, DELETE, etc).

Service Worker Consiste en un fichero Javascript ejecutado en segundo plano por los navegadores. Se encuentra ubicado entre una aplicación y un navegador y tiene la particularidad de continuar ejecutándose a pesar de encontrarse cerrado el navegador.

Sidenav *Side navigation menu*: Menú lateral plegable para navegación.

TLS *Transport Layer Security*: Protocolo criptográfico que garantiza las comunicaciones en Internet brindando integridad y privacidad en capas de transporte.

Referencias

[ANGa] *Angular-Material*, https://material.angular.io/, último acceso: 29 de octubre de 2018.

[ANGb] *Sitio de Angular.io*, https://angular.io/, último acceso: 10 de agosto de 2018.

[APC] *Apache Cordova*, https://cordova.apache.org/, último acceso: 14 de octubre de 2018.

[APP] *El manifiesto de las apps web*, https://developers.google.com/web/fundamentals/web-app-manifest/, último acceso: 21 de octubre de 2018.

[AUT] *Sitio de Auth0*, https://auth0.com/, último acceso: 10 de agosto de 2018.

[BDN] *Curso Bases de Datos No Relacionales, edición 2017, Facultad de Ingeniería - UdelaR*, https://eva.fing.edu.uy/course/view.php?id=947, último acceso: 30 de setiembre de 2018.

[BRI] *Página web principal del proyecto BriefTube*, http://brieftube.com/, último acceso: 16 de mayo de 2017.

[CIS] *Cisco Visual Networking Index: Forecast and Methodology*, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html, último acceso: 30 de setiembre de 2018.

[CLIa] *Clipmine en AngelList*, https://angel.co/clipmine, último acceso: 16 de mayo de 2017.

[CLIb] *Página de CrunchBase del proyecto Clipmine*, https://www.crunchbase.com/organization/clipmine, último acceso: 16 de mayo de 2017.

[CLIc] *Página de Linkedin del proyecto Clipmine*, https://www.linkedin.com/company/clipmine-inc-, último acceso: 16 de mayo de 2017.

[CLId] *Página web principal del proyecto Clipmine*, https://clip.mn/ (último acceso: –).

[DIR] *Aplicación DirecTVSports para Android*, https://play.google.com/store/apps/details?id=com.fwc2014.directvpan.and, último acceso: 27 de mayo de 2017.

[doc] *BDNR - FIng UdelaR: Clase 6 - Bases de datos de documentos*, https://eva.fing.edu.uy/mod/resource/view.php?id=59861, último acceso: 30 de setiembre de 2018.

[EXPa] *Sitio de Expressjs*, https://expressjs.com/, último acceso: 10 de agosto de 2018.

[EXPb] *Uso de Middleware en Expressjs*, https://expressjs.com/en/guide/using-middleware.html, último acceso: 10 de agosto de 2018.

[GCM] *Google Cloud Messaging*, https://developers.google.com/cloud-messaging/, último acceso: 10 de agosto de 2018.

[GL-a] *Documentación de Google Cloud Video Intelligence*, https://cloud.google.com/video-intelligence/docs, último acceso: 27 de mayo de 2017.

[GL-b] *Google Cloud Video Intelligence*, https://cloud.google.com/video-intelligence, último acceso: 27 de mayo de 2017.

[HAM] *Hammerjs*, https://hammerjs.github.io/, último acceso: 29 de octubre de 2018.

[Har15] Guy Harrison, *Next Generation Databases: NoSQL, NewSQL and Big Data.*, Apress, 2015, ISBN: 978-1-4842-1330-8; (disponible en el Portal Timbó: http://link.springer.com.proxy.timbo.org.uy:443/book/10.1007/978-1-4842-1329-2).

[INT] *Página web principal de Interplay*, http://www.avid.com/products/interplay-production, último acceso: 16 de mayo de 2017.

[ION] *Ionic Freamwork*, https://ionicframework.com/, último acceso: 14 de octubre de 2018.

[JS] *Javascript Mozilla Reference*, https://developer.mozilla.org/en-US/docs/Web/JavaScript, último acceso: 30 de setiembre de 2018.

[KZO] *Página web principal de suite de videos KZO*, https://kzoinnovations.com, último acceso: 25 de mayo de 2017.

[LIG] *Lighthouse*, https://chrome.google.com/webstore/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk, último acceso: 21 de octubre de 2018.

[LOC] *Locust*, https://locust.io/, último acceso: 03 de noviembre de 2018.

[MADa] *El consumo de vídeo evoluciona con The Mad Video*, http://www.abc.es/tecnologia/redes/20130217/abci-madvideo-video-youtube-201301301121.html, último acceso: 15 de mayo de 2017.

[MADb] *The Mad Video en AngelList*, https://angel.co/the-mad-video, último acceso: 15 de mayo de 2017.

[MADc] *The Mad Video en FaceBook*, https://www.facebook.com/TheMadVideo (último acceso: 15 de mayo de 2017), último acceso: 10 de agosto de 2018.

[MADd] *The Mad Video en Twitter*, https://twitter.com/themadvideo, último acceso: 15 de mayo de 2017.

[MAR] *Mobile App Report*, https://static.poder360.com.br/2017/08/2017USMobileAppReport-1.pdf, último acceso: 11 de octubre de 2018.

[MATa] *Material Design*, https://material.io/, último acceso: 10 de agosto de 2018.

[MATb] *Materialize CSS*, https://materializecss.com/, último acceso: 10 de agosto de 2018.

[MDN] *Mozilla web app manifest*, https://developer.mozilla.org/en-US/docs/Web/Manifest, último acceso: 21 de octubre de 2018.

[MEAa] *Mean stack - IBM Developers*, https://www.ibm.com/developerworks/library/wa-mean1/index.html, último acceso: 10 de agosto de 2018.

[MEAb] *Mean stack - MongoDB Blog*, https://www.mongodb.com/blog/post/the-mean-stack-mongodb-expressjs-angularjs-and, último acceso: 10 de agosto de 2018.

[MEAc] *Sitio de MEAN.io*, http://mean.io/, último acceso: 10 de agosto de 2018.

[MONa] *Sitio de MongoDB*, https://www.mongodb.com/, último acceso: 10 de agosto de 2018.

[MONb] *Sitio de Mongoosejs*, http://mongoosejs.com/, último acceso: 10 de agosto de 2018.

[MS-a] *Documentación de Microsoft Video Indexer*, https://docs.microsoft.com/en-us/azure/cognitive-services/video-indexer/video-indexer-overview, último acceso: 26 de mayo de 2017.

[MS-b] *Interfaz de Microsoft Video Indexer Preview*, https://www.videoindexer.ai, último acceso: 26 de mayo de 2017.

[MS-c] *Microsoft Azure - Cognitive Services*, http://www.microsoft.com/cognitive-services, último acceso: 26 de mayo de 2017.

[MS-d] *Microsoft Garage*, https://www.microsoft.com/en-us/garage, último acceso: 26 de mayo de 2017.

[MS-e] *Microsoft Video Indexer*, http://vi.microsoft.com, último acceso: 26 de mayo de 2017.

[NOD] *Sitio de Node.js*, https://nodejs.org, último acceso: 10 de agosto de 2018.

[OBM] *Oobservatory by Mozilla*, https://observatory.mozilla.org/, último acceso: 03 de noviembre de 2018.

[PANa] *Panopto en Wikipedia*, https://en.wikipedia.org/wiki/Panopto, último acceso: 15 de mayo de 2017.

[PANb] *Página web principal del proyecto Panopto*, https://www.panopto.com/, último acceso: 15 de mayo de 2017.

[PM2] *Advanced, production process manager for Node.js*, http://pm2.keymetrics.io/, último acceso: 14 de octubre de 2018.

[PUS] *Push API*, https://w3c.github.io/push-api/, último acceso: 30 de setiembre de 2018.

[PWAa] *App Install Banners*, https://developers.google.com/web/fundamentals/app-install-banners/, último acceso: 25 de octubre de 2018.

[PWAb] *Progressive Web Apps - Google*, https://developers.google.com/web/progressive-web-apps/, último acceso: 21 de octubre de 2018.

[PWAc] *Progressive Web Apps Checklist*, https://developers.google.com/web/progressive-web-apps/checklist, último acceso: 21 de octubre de 2018.

[PWAd] *What, Exactly, Makes Something A Progressive Web App?*, https://infrequently.org/2016/09/what-exactly-makes-something-a-progressive-web-app/, último acceso: 25 de octubre de 2018.

[PWAe] *Your First Progressive Web App*, https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/?hl=en, último acceso: 21 de octubre de 2018.

[QUA] *Qualys*, https://www.qualys.com/, último acceso: 03 de noviembre de 2018.

[REAa] *React Native*, https://facebook.github.io/react-native/, último acceso: 14 de octubre de 2018.

[REAb] *Tutorial sobre integración de Auth0 en una aplicación Web*, https://auth0.com/blog/real-world-angular-series-part-1/, último acceso: 10 de agosto de 2018.

[REP] *Report Qualys*, https://www.ssllabs.com/ssltest/analyze.html?d=www.hopclip.com&hideResults=on, último acceso: 03 de noviembre de 2018.

[SEL] *Selenium ide*, https://www.seleniumhq.org/projects/ide/, último acceso: 10 de noviembre de 2018.

[SKIa] *SkipTo extensión para Google Chrome*, https://chrome.google.com/webstore/detail/skipto/ejodhjghjlaiccblbppbaaabdcllfhgm, último acceso: 15 de mayo de 2017.

[SKIb] *SkipTo Página principal*, https://skipto-fffb6.firebaseapp.com/index, último acceso: 15 de mayo de 2017.

[SKIc] *SkipTo repositorio GitHub*, https://github.com/WyattHRichter/SkipTo, último acceso: 15 de mayo de 2017.

[VIDa] *VideoJS - The player framework*, https://videojs.com/, último acceso: 13 de octubre de 2018.

[VIDb] *Videojs-Markers - displays customizable markers upon progress bars of videojs players*, https://github.com/spchuang/videojs-markers/, último acceso: 29 de octubre de 2018.

[VIDc] *Videojs-Youtube - youtube playback technology for video.js*, https://github.com/videojs/videojs-youtube, último acceso: 29 de octubre de 2018.

[WEB] *Web Push Notifications*, https://developers.google.com/web/fundamentals/push-notifications, último acceso: 6 de noviembre de 2018.

[XCO] *Xcode*, https://developer.apple.com/xcode/, último acceso: 14 de octubre de 2018.