Universidad ORT Uruguay

Facultad de Ingeniería

Proyecto Mixit!

Plataforma para la creación de videos musicales colaborativos

Entregado como requisito para la obtención del título de Ingeniero en Sistemas

Rodrigo de la Barrera– N° 160922

Gustavo Gard– N° 164490

Bruno Montaner – N° 151770

Tutor: Martín Solari

2014

Declaración de autoría

Nosotros, Rodrigo de la Barrera, Gustavo Gard y Bruno Montaner, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos el proyecto de fin de carrera;
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Rodrigo de la Barrera

Gustavo Gard

Bruno Montaner

26/08/2014

Agradecimientos

En primer lugar queremos agradecer a nuestras familias y amigos, por su apoyo incondicional durante todo el proyecto, permitiendo que este se desarrollara de la mejor manera posible.

A nuestro tutor Martín Solari, de quien aprendimos mucho, y quien siempre estuvo disponible para ayudarnos y aconsejarnos a lo largo de todo el proyecto.

A los músicos que voluntariamente probaron la aplicación y dieron su opinión permitiendo identificar oportunidades de mejora y obtener conclusiones muy importantes sobre el producto.

Por último, a Héctor Bajac y Homero Pérez Noble, por comprometerse con el proyecto y por su continua disposición para con el mismo.

Abstract

Hoy en día existen aplicaciones web que permiten publicar y compartir videos musicales, sin embargo, estas no proporcionan ningún mecanismo para la creación de los videos en forma distribuida. Los objetivos del proyecto Mixit! son la investigación y el desarrollo de prototipos, para la colaboración entre músicos amateurs. Como resultado se espera un producto que les permita, elaborar de forma simple y rápida, un video musical colaborativo tipo mosaico. Este video será el resultado de la unión de videos grabados por distintas personas alrededor del mundo cantando o tocando algún instrumento. Tiene como principales funciones: crear un nuevo proyecto grabando o subiendo un video inicial, escuchar el audio de los proyectos abiertos, colaborar en los proyectos cantando o tocando algún instrumento y finalmente generar un video colaborativo.

Dada la naturaleza experimental del producto y el contexto del proyecto, se eligió un ciclo de vida evolutivo y un enfoque de gestión ágil. Durante el proyecto se realizó un relevamiento con los clientes y potenciales usuarios, con el fin de identificar los aspectos claves que debería contener la solución. Se realizaron investigaciones de las tecnologías necesarias para el desarrollo. De esta forma se desarrollaron prototipos que sirvieron para probar los diferentes conceptos y mitigar los riesgos tecnológicos. La interfaz de usuario del producto fue desarrollada utilizando el *framework* ASP .NET MVC. En el servidor, se utilizaron tecnologías Java EE en conjunto con las librerías FFmpeg y Sox para el manejo de audio y video y un servidor de *streaming* y grabación de videos implementado con Red5 Media Server. La comunicación entre ambos componentes se realiza mediante una API REST.

Palabras clave

Música, Canción, Grabación, Video Colaborativo, Red5, FFmpeg, Flash, *Streaming,* Prototipos evolutivos, MVP

Índice

1 Introducción

1.1 Objetivos del proyecto

Mixit! es un proyecto de fin de carrera de Ingeniería en Sistemas. El mismo ha sido desarrollado dentro del marco conceptual del Laboratorio ORT Software Factory por tres estudiantes de la carrera de Ingeniería en Sistemas. El proyecto fue propuesto por dos emprendedores y músicos amateurs: Homero Pérez Noble y Héctor Bajac.

Hoy en día, existen aplicaciones web que permiten publicar y compartir videos musicales, sin embargo, estas no proporcionan ningún mecanismo para la creación de los videos en forma distribuida. Los objetivos del proyecto Mixit! son la investigación y el desarrollo de prototipos, para la colaboración entre músicos amateurs. Se busca finalizar el proyecto con, un prototipo que les permita elaborar de forma simple y rápida, un video musical colaborativo tipo mosaico. En la Figura 1 se muestra un ejemplo de video en el cual participan cuatro personas. Este video será el resultado de la unión de videos grabados por cada uno de ellos cantando o tocando algún instrumento. Como factor diferenciador del producto se espera que el prototipo se base enteramente en tecnologías web.

Figura 1: Captura de un video musical colaborativo de cuatro participantes

1.2 Marco conceptual ORTsf

El laboratorio ORT Software Factory (ORTsf), es una organización académica dedicada a la enseñanza de prácticas de Ingeniería de Software, a la mejora de procesos de software, a la transferencia de tecnología a la industria y a la producción de software [1].

El tutor del proyecto es Martín Solari, miembro del equipo de ORTsf. Durante su transcurso, se le han realizado tres revisiones, cada una al finalizar el primer, segundo y tercer trimestre del proyecto. Estas revisiones tuvieron como objetivo evaluar el estado del proyecto en ese momento y la aplicación de prácticas de ingeniería de software por parte del equipo. En cada una de ellas estuvieron presentes el tutor y un revisor asignado por el laboratorio. Todas estas instancias fueron importantes para que el equipo reflexione sobre puntos fuertes y oportunidades de mejora para el proyecto. Se generaron informes indicando las acciones a tomar para realizar las correcciones pertinentes. Estos informes fueron entregados al tutor luego de cada revisión.

1.3 Equipo

El equipo está conformado por Rodrigo de la Barrera, Gustavo Gard y Bruno Montaner, estudiantes del último año de la carrera Ingeniería en Sistemas de la Universidad ORT Uruguay.

Ninguno de los integrantes del equipo a excepción de Bruno poseía algún conocimiento sobre música. Bruno ha participado ocasionalmente como vocalista y guitarrista en una banda amateur. Al momento de comenzar el proyecto todos trabajaban como desarrolladores web en sus respectivas empresas, sin embargo ninguno poseía conocimiento o experiencia en temas clave para el desarrollo del producto como ser el *streaming* y manejo de audio y video en la web.

Para afrontar mejor el proyecto, se designaron roles para cada uno de los miembros del equipo. Esta asignación se realizó teniendo en cuenta la afinidad de cada uno de los integrantes con el área de la ingeniería de software en cuestión. Los conflictos en la elección de los roles se resolvieron mediante un consenso común. La asignación de roles final fue la siguiente:

- **Gerente de proyecto:** Rodrigo de la Barrera
- **Arquitecto:** Bruno Montaner
- **Ingeniero de Requerimientos:** Gustavo Gard
- **SCM:** Bruno Montaner
- **QA:** Gustavo Gard
- **Desarrolladores:** Rodrigo de la Barrera, Gustavo Gard, Bruno Montaner

1.4 Clientes

Los clientes son, Héctor Bajac, docente y catedrático de Marketing en Universidad ORT Uruguay y Homero Pérez Noble, director ejecutivo en Wanako (productora audiovisual). El interés por parte de los clientes en el proyecto surgió a partir de su afición a la música y de la identificación de la problemática.

Al comienzo del proyecto el equipo se reunió semanalmente con los clientes. El objetivo era involucrarse en el problema y comenzar a definir los principales conceptos y funcionalidades que debería contemplar el producto.

Los clientes tienen como expectativas que el resultado del proyecto sea un prototipo funcional adecuado para realizar una muestra ante inversores de modo de obtener financiación y dar inicio a una *startup*. Además, pretenden evaluar si es factible crear videos musicales colaborativos de forma distribuida, con músicos que no tienen porque encontrarse en un mismo lugar físico. El lema de Mixit! es: “Unir al mundo a través de la música”.

1.5 Alcance y desafíos del proyecto

Teniendo en cuenta los objetivos del proyecto, se estableció el siguiente alcance para el mismo:

- Realizar y documentar la investigación de las tecnologías necesarias para construir el prototipo teniendo en cuenta que debe ser una aplicación completamente basada en tecnologías web.
- Desarrollar el prototipo definido.
- A partir del producto desarrollado, experimentar con potenciales usuarios finales, un concepto innovador como lo es la creación musical colaborativa de forma distribuida.

Con este proyecto se presentaron varios desafíos, de los cuales podemos destacar dos:

- **Conocimiento sobre música:** El equipo no tenia experiencia como músicos, por lo que se requirió mucho esfuerzo para entender los conceptos necesarios para encontrar una solución adecuada a la problemática. El equipo tuvo que reunirse varias veces con los clientes para lograr entender algunos conceptos básicos. En algunas ocasiones también fue necesario recurrir a expertos del dominio y a material bibliográfico.

- **Tecnologías para el manejo de audio y video en la web:** Para desarrollar el producto fue necesario utilizar varias tecnologías y librerías referentes al manejo de audio y video. La falta de experiencia del equipo con estas tecnologías se presentó como un gran desafío. Este contexto obligó al equipo a emplear un proceso de investigación y a desarrollar algunos prototipos tecnológicos como pruebas de concepto.

1.6 Descripción del proceso

El proceso de desarrollo se basó en la implementación de una serie de prototipos evolutivos, donde cada prototipo buscaba incursionar en algún aspecto clave para el producto. De lo relevado en las primeras reuniones, se creó una lista inicial de funcionalidades a desarrollar y luego se negoció con el cliente por cuales se comenzaría. A partir de las funcionalidades seleccionadas por el cliente, se identificaron varios desafíos tecnológicos que el equipo debía enfrentar en el desarrollo de la solución. Teniendo en cuenta esto, se realizó una investigación tecnológica cuyo objetivo era encontrar posibles soluciones a cada desafío. Esta investigación tuvo como resultado los dos primeros prototipos funcionales del proyecto. Ambos prototipos pasaron por un proceso de validación con los clientes. De estas validaciones, se decidía si las tecnologías empleadas lograban resolver el problema puntual atacado por cada prototipo o se debía buscar una alternativa.

Luego del proceso de investigación tecnológica, el equipo comenzó a enfocarse en otros aspectos del producto. Se buscó integrar los prototipos tecnológicos desarrollados anteriormente, incorporando mejoras propuestas y priorizadas por los clientes. A partir de ese momento se desarrollaron tres prototipos más. Cada uno evolucionando en cuanto a estética, usabilidad y cumplimiento de otros requerimientos. El objetivo en cada evolución era tratar de afinar el concepto de Mixit! y conformar al cliente en sus prioridades. Todo esto, teniendo como base, una lista abierta con nuevas funcionalidades, sugerencias y mejoras surgidas a partir de las validaciones. Para manejar dinámicamente el alcance del producto con los clientes, el equipo optó por definir lo que se conoce como MVP, *Minimal Viable Product* (Producto Mínimo Viable en inglés). Un MVP consiste en un producto que tiene las mínimas funcionalidades necesarias para poder realizar pruebas con potenciales usuarios, los cuales luego proporcionarán *feedback* acerca del mismo [2].

Dada la naturaleza experimental del producto y el contexto del proyecto se eligió un enfoque de gestión ágil. Se definieron iteraciones de dos semanas. Al principio de cada iteración, el equipo se reunía a estimar las funcionalidades a desarrollar teniendo en cuenta un colchón de tiempo. Este colchón le dio flexibilidad necesaria a la planificación para quitar o incorporar nuevos requerimientos asignados a la iteración. Trabajar de esta forma permitió agregar el máximo de valor posible al producto en cada iteración.

2 Ingeniería de requerimientos

2.1 Descripción del problema

Vivimos en una era donde una gran cantidad de personas tiene acceso a la tecnología necesaria para poder comunicarse con cualquier otra, en cualquier parte del mundo. Esto ha iniciado grandes emprendimientos como Facebook, Twitter, Youtube, Skype, etcétera. Estas redes sociales reúnen a millones de personas. Su principal función es permitirle a sus usuarios publicar y compartir contenido o comunicarse entre ellos. Se ha observado que, aún contando con todas estas facilidades, no han surgido en la web, aplicaciones que permitan la creación de videos musicales colaborativos como los que propone Mixit!.

Hoy en día, en la mayoría de los casos, las grabaciones musicales que se realizan en estudio no son en simultáneo. Esto es, no se graba el sonido resultante de la interacción de los músicos tocando juntos. Cada instrumento se graba por separado. Luego, se realiza un proceso de edición para sincronizar y mezclar las pistas de audio obtenidas de cada instrumento. De esta mezcla surge la canción final. Esto es lo que se conoce como grabación multicanal [3]. Esta forma de crear música, junto con tecnología de grabación digital, abre la posibilidad de que el proceso anterior pueda ser ejecutado de forma distribuida. Por músicos que no tienen porque estar presentes en un mismo lugar físico.

Actualmente existe software exclusivamente dedicado a la creación musical multicanal. Un ejemplo de esto son: GarageBand[1] o Logic Pro X[2], de Apple Inc. Este tipo de software puede llegar a ser complejo de utilizar por un músico amateur. Usualmente, también requieren de interfaces costosas para interactuar con los instrumentos musicales. Para lograr crear una canción con estas aplicaciones, es necesario que una única persona tenga en su computadora todas las pistas que componen la canción. Esta persona es la responsable de realizar el trabajo de edición de forma local en su computadora. Todos los músicos participantes de la creación deben hacerle llegar sus archivos de audio a esta persona, lo cual representa otra dificultad. Todo lo mencionado anteriormente hace que crear música de forma distribuida sea complejo. Además, se ha observado que estas aplicaciones no permiten agregar video a las creaciones.

[1] https://www.apple.com/mac/garageband/
[2] https://www.apple.com/logic-pro/

En la Figura 2 se muestra una captura de la aplicación GarageBand en la cual se encuentran las pistas de audio de una creación musical y el instrumento asociado a cada pista.

Figura 2: Pistas de audio e instrumentos asociados en la aplicación GarageBand, obtenida de [22]

2.2 Usuarios

En conjunto con los clientes se definieron los potenciales usuarios del producto. Estos corresponden a personas aficionadas a la música o músicos amateurs que deseen crear un video musical colaborativo. Se espera que el servicio sea utilizado desde una casa o espacio informal, no desde un estudio de grabación.

El hecho de que el producto deba ser enteramente desarrollado con tecnologías web le permite alcanzar un público mucho más amplio. Las aplicaciones como las mencionadas en la sección anterior son de escritorio y enfocadas a músicos con determinados conocimientos. Con estas aplicaciones se pueden producir resultados mucho más profesionales y trabajados que los que pretende producir Mixit!.

2.3 Principales funcionalidades

El principal objetivo del producto es proveerle a sus usuarios un espacio en la web en el cual puedan crear proyectos musicales colaborativos. Los integrantes de un proyecto podrían ser, por ejemplo, un guitarrista uruguayo, un bajista brasilero, un baterista americano, un vocalista inglés, etcétera. El resultado de cada proyecto es un video musical colaborativo. La idea es que una vez finalizado un proyecto estos videos sean visibles para el resto del mundo. El lema de Mixit! refleja su principal objetivo: "Unir al mundo a través de la música".

Partiendo de este objetivo se definieron como principales funciones del producto, las siguientes:

- **Creación de un nuevo proyecto colaborativo:** La aplicación debería permitir crear un proyecto colaborativo a partir de un video inicial. Para esto, hay dos alternativas: que el usuario suba un video grabado previamente o que lo grabe desde la propia aplicación. Por ejemplo, utilizando el micrófono y la *webcam* de su computadora. Generalmente, el músico que graba la primera pista de audio (video en este caso) de una creación musical requiere de una referencia rítmica como base. Teniendo en cuenta esto, se identificó también la necesidad de incluir en el mecanismo de grabación un instrumento como el metrónomo. El metrónomo es un instrumento utilizado para marcar el tiempo o compás de una composición musical. Este instrumento produce un sonido regular que se mide en BPM: *Beats per Minute* (pulsaciones por minuto en inglés) y que guía al músico durante su ejecución.

- **Colaboración en un proyecto existente:** Un usuario colabora en un proyecto cuando aporta algún rol musical que el proyecto necesita. Lo puede hacer subiendo o grabando un video. En el caso que sea grabando un video, al comenzar la grabación escucharía como base la pista de audio del proyecto. Las pistas de audio son extraídas de los videos subidos o grabados y deben ser aprobadas por el creador del proyecto antes de entrar dentro de la mezcla (audio del proyecto).

- **Sincronización de pistas:** El creador de un proyecto debería poder escuchar la pista de cada uno de los aportes recibidos. Estas pistas deben ser representadas gráficamente mediante su onda. El creador debe poder sincronizarlas mediante *drag and drop* y/o ajustar el volumen de cada una. El audio resultante de esta mezcla corresponde a la pista de audio del proyecto que se mencionó anteriormente.

- **Finalizar y publicar un proyecto:** Finalizar un proyecto implica generar el video colaborativo resultante de la unión de todos los aportes aceptados por el creador del proyecto. La idea es que este video pueda ser publicado y visto por otras personas.

2.3.1 Secciones de la aplicación

A partir de las principales funcionalidades del producto se definieron algunas secciones dentro de la aplicación. A cada una de estas secciones se les asignó un nombre representativo. Las secciones definidas fueron las siguientes:

Rec Room: Sección donde los usuarios graban o suben sus videos para crear un nuevo proyecto o colaborar en uno ya existente. En la Figura 3 se muestra dicha sección en la cual se incluye un metrónomo virtual (izquierda) , un grabador (centro) y una lista con las miniaturas de los últimos cuatro intentos de grabación (derecha).

Figura 3: Sección Rec Room del producto final

Backstage: Sección donde los usuarios pueden visualizar los proyectos abiertos y escuchar su audio. Aquí es donde pueden ingresar a colaborar en algún proyecto aportando un determinado rol requerido por el mismo. En la Figura 4 se muestran varios proyectos en forma de tarjetas. En cada tarjeta se muestra: el nombre del proyecto, la foto de las personas que ya han colaborado, la de los instrumentos faltantes y la descripción del proyecto.

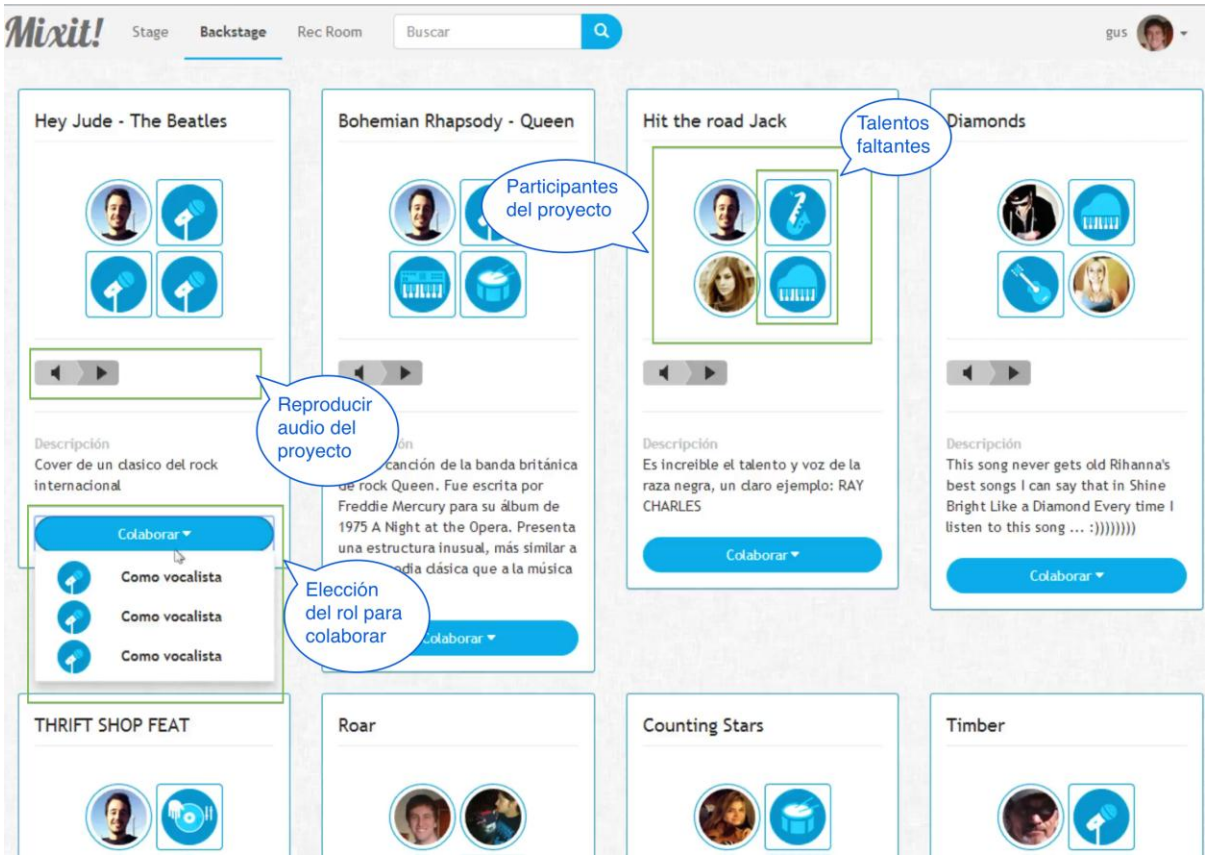


Figura 4: Sección Backstage del producto final

Stage: Sección en la cual se listan los videos colaborativos resultantes de los proyectos finalizados. En la Figura 5 se muestran los proyectos finalizados en forma de tarjetas. En cada tarjeta se muestra: foto del creador del proyecto, una miniatura del video resultante, fotos de los participantes del proyecto, descripción del proyecto y una sección de comentarios.

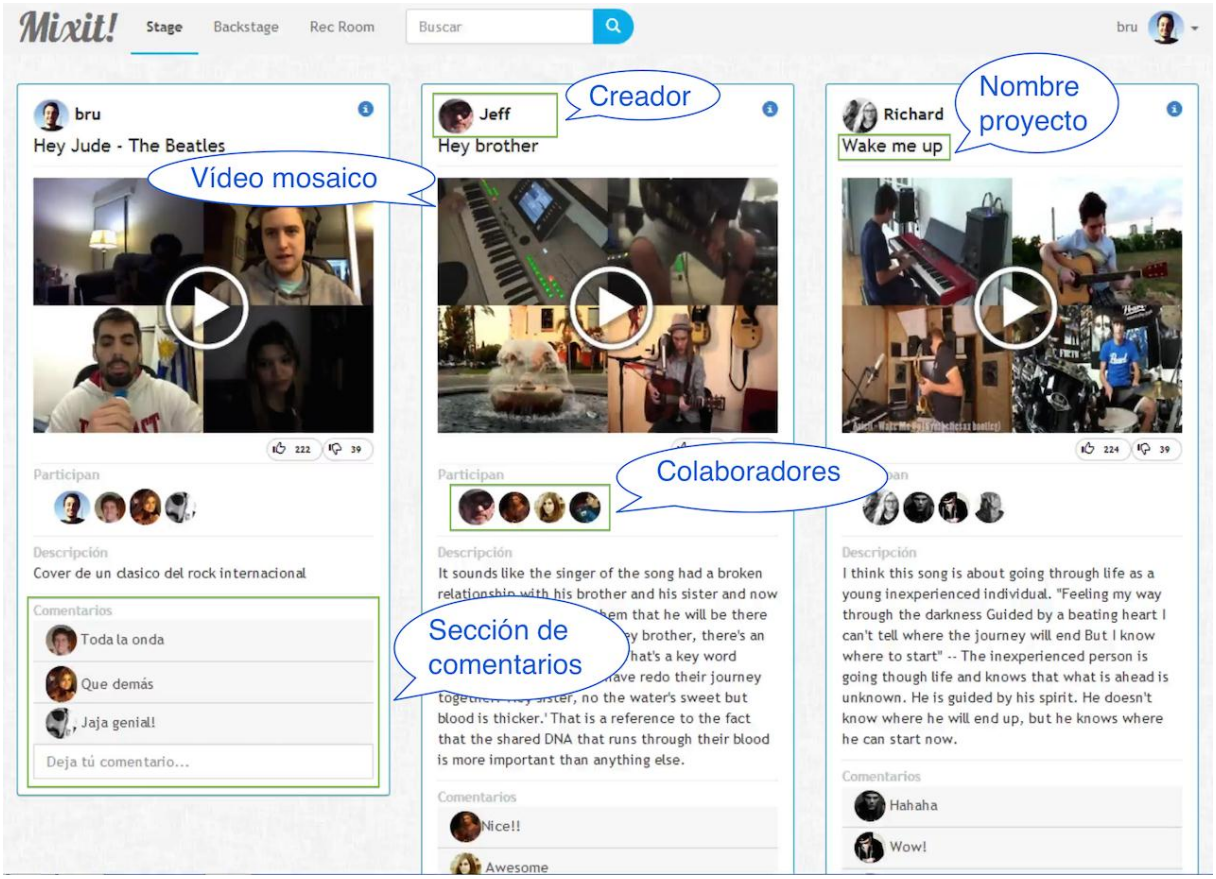


Figura 5: Sección Stage del producto final

2.4 Requerimientos no funcionales

Junto con las principales funcionalidades del producto se definieron algunas restricciones para el mismo. Estas restricciones se desprendieron tanto de los objetivos del producto como de las expectativas del cliente. A continuación se describen cada una de estas restricciones ordenándolas por prioridad, de mayor a menor.

- **"Mostrabilidad":** Es un atributo de calidad definido por el equipo. Este atributo hace referencia al grado en que el producto puede ser mostrado a posibles inversores y que tan bien representa la idea que se quiere transmitir, el concepto de Mixit!. La calidad estética de la aplicación y la claridad con la que se expresan sus funciones son muy importantes para cumplir con este punto. Los encargados de verificar el cumplimiento de este atributo fueron los clientes.

- **Usabilidad:** Se pretende que el producto sea utilizado por músicos amateurs. Estos no tienen porque tener conocimientos o experiencia en el uso en aplicaciones de edición de audio y video. Por este motivo, se esperaba que el producto fuera lo más auto explicativo posible. El producto debía ser eficiente y eficaz para grabar pistas y colaborar en proyectos.

- **Escalabilidad:** En un futuro se espera que la aplicación tenga un uso masivo. Algunas de sus funcionalidades como el procesamiento de videos pueden consumir muchos recursos. Por este motivo, era deseable que se tuviera en cuenta en el diseño de la arquitectura el factor escalabilidad y que se evaluaran distintas opciones y costos para el despliegue de la misma.

- **Modificabilidad y Extensibilidad:** Al inicio del proyecto se definió que el producto final solo abarcaría la generación de videos colaborativos para cuatro integrantes, con el *template* de la Figura 1 (pantalla dividida en cuatro). Sin embargo, se esperaba que en un futuro se pudieran agregar más *templates.* Por otro lado, se definió la necesidad de poder extender la plataforma al mundo de los *smartphones*. Estos requerimientos deberían implementarse con el menor impacto posible sobre la arquitectura.

- **Tecnologías:** Se definió al inicio del proyecto que dentro de lo posible se deberían utilizar tecnologías *open source*. El motivo de esto era ahorrar costos y problemas de licenciamiento en una potencial puesta a producción del producto.

2.5 Estrategias de relevamiento

Con el objetivo de comenzar a comprender la problemática y definir las funcionalidades y restricciones descritas en las secciones anteriores, el equipo utilizó las siguientes estrategias de relevamiento:

Reuniones frecuentes con los clientes

Los clientes fueron los que propusieron la idea del proyecto y ambos son músicos amateurs. Por este motivo, desde un principio se optó por reunirse con los mismos semanalmente. El objetivo era ir comprendiendo la problemática a partir de sus conocimientos como músicos y de sus experiencias al tratar de crear videos musicales colaborativos de forma distribuida.
En las primeras reuniones, los clientes le trasmitieron al equipo su experiencia al tratar de crear un video con algunas herramientas existentes. Esta experiencia consistió en realizar una grabación de un *cover* de la canción "La Bamba".

Previamente a cada reunión se les enviaba un email con todas las consultas y temas a tratar. De este modo, se evitaba perder tiempo durante el desarrollo de las mismas abarcando

temas que no estaban previstos. A su vez, luego de cada reunión se elaboraba otro email con todo lo discutido. Esto ayudó a validar que el equipo haya entendido correctamente lo discutido y a dejar un registro de lo que se iba pautando con el cliente.

Bocetos de interfaz de usuario de baja definición

Durante los primeros meses de proyecto fue muy frecuente el uso de bocetos de UI como estrategia de análisis y relevamiento. El equipo utilizó está técnica de prototipación para realizar un análisis interno luego de relevar con el cliente y para discutir sobre algo más tangible durante las reuniones. Los primeros bocetos se diseñaron cuando se intentaban definir los principales flujos de la aplicación. Consistió en mapear en un pizarrón una posible solución de lo relevado. Una vez que el cliente los validó, se utilizó la herramienta Balsamiq Mockups para pasarlos a formato digital. Esto hizo que fuera sencillo de modificarlos por parte del equipo o del cliente. En Anexo 1 se muestran algunos de los bocetos realizados, en formato digital. En la Figura 6 se muestra un boceto realizado en pizarrón con los principales flujos de la aplicación.

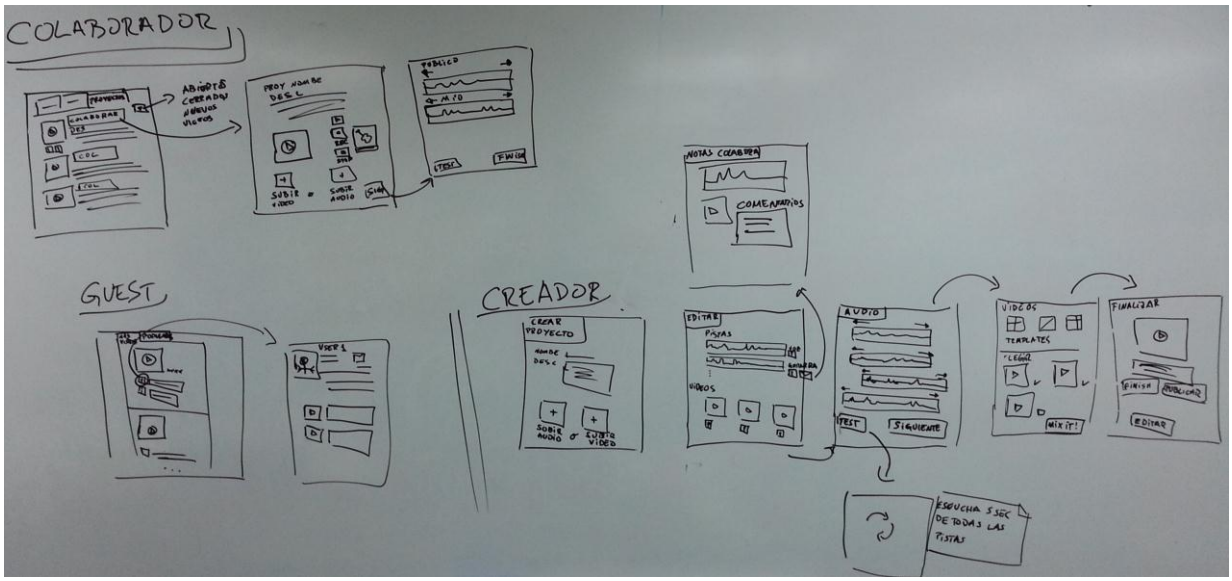


Figura 6: Boceto en pizarrón con los principales flujos de la aplicación

Investigación de producto similares

La investigación de productos similares jugó un papel importante durante el relevamiento. Los clientes frecuentemente trasmitían sus experiencias utilizando determinadas aplicaciones que inicialmente eran desconocidas para el equipo. La investigación de las mismas ayudaron al equipo a comprender algunos conceptos del dominio. Además, a partir de estas surgieron algunas ideas sobre potenciales funcionalidades. La aplicación que más se utilizó como referencia fue GarageBand, por ser la más orientada a músicos amateurs.

3 Definición de la solución

3.1 Selección de tecnologías

A partir de los principales requerimientos, se identificaron varios desafíos tecnológicos a enfrentar. Estos desafíos llevaron a que el equipo tuviese que realizar un proceso de análisis y selección de tecnologías. El proceso consistió en considerar varias tecnologías disponibles, tomar la que inicialmente se adecuó mejor al problema a resolver y construir un prototipo para probarla. Como resultado de este proceso se obtuvieron los dos primeros prototipos del proyecto.

El desarrollo de estos prototipos tuvo dos objetivos bien definidos. El primero era probar la factibilidad de las tecnologías seleccionadas para resolver el problema atacado por el prototipo en cuestión. El segundo consistía en mostrarle un avance del proyecto al cliente para mantenerlo involucrado y además ir definiendo las características del producto.

Como consigna del primer prototipo se definió: “Subir cuatro videos a un sitio web, extraer sus pistas de audio, presentar gráficamente sus ondas, permitir moverlas horizontalmente mediante *drag and drop* para sincronizarlas, reproducirlas a todas juntas para verificar la sincronización y finalmente generar el video mosaico, tomando en cuenta los desfasajes introducidos en la sincronización.”.

El segundo prototipo fue una evolución del primero y su propósito era: “Crear un sitio web que permita realizar la misma tarea del primer prototipo pero capturando los videos desde una cámara y micrófono conectados a la computadora. Además, introducir mejoras relevadas del primer prototipo y comenzar a definir los principales flujos de la solución.”

Con el objetivo de organizar la búsqueda de las tecnologías, se definieron a partir de las funcionalidades propuestas para cada prototipo, los siguientes temas de investigación:

	Prototipo 1	**Prototipo 2**
Tema de investigación tecnológica	Manejo de audio y representación de ondas de sonido en la web	Grabación y reproducción de videos por *streaming* desde la web
	Manejo de audio y video en el servidor	

Tabla 1: Temas de investigación tecnológicas por prototipos

Desde un principio, existieron dos factores que se sabia podrían condicionar la búsqueda de las tecnologías. Por un lado, la restricción impuesta por el cliente de que era deseable que las tecnologías fueran *open source.* Por otro, la elección del lenguaje de programación a utilizar. Por este motivo, se tomo como punto de partida, la elección de un *framework* web para el posterior desarrollo de los prototipos ya que con este estaría implícito el lenguaje.

3.1.1 *Framework* web

La primera decisión a tomar antes de comenzar a desarrollar los prototipos era elegir el *framework* web a utilizar. Para esto, se evaluaron las siguientes dos alternativas:

- **ASP .NET MVC:** Nunca se aprendió en la facultad. Solo un miembro del equipo poseía experiencia laboral en el mismo.
- **Java Server Faces (JSF):** Ningún miembro del equipo poseía experiencia laboral con este *framework.* Sin embargo, todos habían trabajado con él en una materia electiva (Arquitectura de Software sobre plataformas Java).

Lo más importante en el inicio era comenzar a desarrollar rápidamente los prototipos. Se decidió que para ahorrar costos de aprendizaje y enfocarse en el problema a resolver, el *framework* elegido debía ser uno conocido por todos los integrantes del equipo. En ese entonces se terminó optando por JSF. Este fue el que se utilizó para el desarrollo de los dos primeros prototipos e implicó utilizar Java como lenguaje de programación.

JSF es un *framework* web orientado a componentes. Este tipo de *frameworks* proveen un conjunto de componentes pre-desarrollados, donde cada componente se renderiza luego a determinado código HTML. Además, abstraen parte de la lógica referente a la gestión del ciclo de vida HTTP. El manejo de *requests* y *responses* es encapsulado en componentes reutilizables y el estado entre los mismos es manejado automáticamente por el *framework*. Estas abstracciones permiten acelerar el desarrollo web bajo algunos contextos. Sin embargo, se pierde parte del control sobre el HTML generado y la utilización de cualquier componente externo que no sea parte del *framework* se vuelve engorrosa.

A partir del tercer prototipo se sabia que la interfaz gráfica de la aplicación iba a cambiar frecuentemente (ver sección 3.2.3). La integración de HTML y la incorporación de nuevo código JavaScript iba ser constante. Surgió la necesidad de utilizar otro *framework* web que permitiera tener más control sobre el HTML generado y que facilitara la utilización de *plugins* JavaScript como, por ejemplo, jQuery. El *framework* elegido en ese entonces fue ASP .NET MVC. Este, al contrario de JSF, es orientado a *requests* o acciones. La integración de código JavaScript y la elaboración de interfaces de usuario complejas se vuelve mucho más simple [4]. Al momento de cambiar el *framework* web ya se tenia parte del *back-end* desarrollado en Java. Se había escrito mucha lógica en cuanto al manejo de audio y video. Por este motivo, se optó por exponer las operaciones del *back-end* mediante servicios web para que el *front-end* pudiera consumirlas. De este modo, se logró integrar el *front-end* .NET con el *back-end* Java, al mismo tiempo que no se desechaba la funcionalidad ya probada.

3.1.2 Manejo de audio y video en el servidor

Cualquier proceso que involucre procesamiento de audio y video suele ser muy costoso en cuanto a consumo de CPU y memoria. Por este motivo, desde un principio se descartó que las extracciones de audio y el mezclado de los videos se fueran a realizar del lado del cliente. Esto debido a las limitaciones que poseen los navegadores web.

La investigación se centro en tecnologías que proveyeran una API(*Application Programming Interface*) que se pudieran integrar al código para realizar todo el procesamiento del lado del servidor. En esta oportunidad se consideraron dos tecnologías: Java Media Framework (JMF)[3] y FFmpeg[4]. A continuación se describe brevemente cada una de ellas.

JMF es una librería Java que permite realizar diversas operaciones sobre archivos de audio y video en distintos formatos y desde código Java [5]. Es un proyecto que posee mucha documentación de respaldo pero que actualmente se encuentra descontinuado.

FFmpeg es un proyecto *open source* y multiplataforma que provee una serie de utilidades y librerías que permiten decodificar, codificar, multiplexar, desmultiplexar, filtrar y reproducir audio y video [6]. Un gran número de proyectos en el mundo del software libre del rubro multimedia utilizan a FFmpeg dentro de sus soluciones [7]. Además, existe una gran cantidad de desarrolladores respaldando este proyecto, generando actualizaciones y agregando nuevas funcionalidades al producto.

FFmpeg funciona mediante invocaciones desde la línea de comandos. Permite realizar todas las operaciones descritas anteriormente mediante el pasaje de determinados comandos con sus respectivos parámetros. Todos estos comandos están presentes en la documentación oficial del proyecto [8].

Profundizando en cada una de las tecnologías mencionadas anteriormente, mediante búsquedas en foros y comunidades de desarrollo, se concluyó que se utilizaría FFmpeg. Se encontró que esta librería es un referente en cuanto al manejo de audio y video. Además, a diferencia de JMF se encuentra en pleno desarrollo y posee mucha documentación actualizada de cada una de sus funcionalidades. Durante la investigación se encontró un artículo en el cual se explicaba como generar uno de los resultados a los cuales se quería llegar: la generación de un video con formato mosaico a partir de varios videos de entrada, este artículo está disponible en [9]. También se encontró que permitía extraer el audio de videos con distintos formatos. La posibilidad de poder realizar estas dos operaciones llevó a

[3] http://www.oracle.com/technetwork/java/javase/tech/index-jsp-140239.html
[4] https://www.ffmpeg.org/

utilizar FFmpeg como tecnología candidata para realizar todo lo referente al manejo de audio y video del lado del servidor.

3.1.3 Manejo de audio y representación de ondas de sonido

Otra de las necesidades tecnológicas que se identificaron luego de definir las principales funcionalidades fue la de reproducir audio en la aplicación web. Con reproducir hacemos referencia a las todas operaciones básicas de cualquier reproductor: *play*, *pause*, *stop*, *seek,* subir/bajar el volumen*,* etcétera.

Adicionalmente a la funcionalidad de reproducir, era deseable que la tecnología seleccionada fuera capaz de representar gráficamente la onda del audio. Se quería contar con una única tecnología que permitiera hacer las dos cosas, de modo de no tener que combinar más de una.

Para cumplir con estas necesidades se realizó una búsqueda en diversos foros y comunidades de desarrollo. A partir de esta búsqueda surgieron tres librerías JavaScript candidatas a utilizar: SoundManager2, WaveForm.js y WaveSurfer.js.

SoundManager2[5] provee una API JavaScript que permite reproducir audio en un navegador web. Es una solución *cross-browser* ya que posee la capacidad de realizar *fallback* a Flash en el caso de que el navegador no soporte HTML5. Si la versión es HTML5, se permiten reproducir todos los formatos de audio soportados por el estándar implementado por el navegador. En el caso de utilizar Flash los formatos soportados son MPEG-4 (AAC, HE-AAC, H.264).

WaveForm.js[6] es una librería que permite dibujar ondas de audio a partir de un *array* de números en punto flotante, los cuales representan los picos de la onda. El proyecto WaveForm.js provee un servicio web que se encarga de transformar *waveforms* de audio hospedado en el servicio SoundCloud[7] en los *arrays* mencionados anteriormente. Además permite personalizar varios atributos del diseño de la onda, por ejemplo, el color. Por otro lado, no permite reproducir audio.

WaveSurfer.js[8] permite reproducir archivos de audio a partir la URL web en donde se encuentra el recurso. Trabaja únicamente con HTML5 y soporta los mismos formatos de audio que establece el estándar del navegador. Además**,** tiene la capacidad de renderizar la onda del audio mediante *canvas* o SVG (*Scalable Vector Graphics*) y permitir visualizar el

[5] http://www.schillmania.com/projects/soundmanager2/
[6] http://waveformjs.org/
[7] https://soundcloud.com/
[8] http://www.wavesurfer.fm/

progreso de la reproducción en la propia imagen, pudiendo inclusive ir hacia delante o hacia atrás en la reproducción haciendo clic en la misma.

Otra funcionalidad interesante que presenta es la de poder insertar marcas en el dibujo de la onda. Desde un principio se pensó que esto eventualmente podría ser útil en el caso de que se deseara marcar posiciones para cortar un trozo de la pista de audio. También provee la capacidad de cambiar el color de la onda. En la Figura 7 se muestra una captura de una onda de audio renderizada por WaveSurfer.js en la cual se pueden ver las marcas mencionadas anteriormente (una verde y otra roja), un cursor (azul), la parte reproducida en violeta y la parte por reproducir en rosado.

Figura 7: Onda de audio renderizada por WaveSurfer.js

Luego de la comparación entre las librerías se concluyó que la elegida seria WaveSurfer.js, por ser la más completa. Esta librería, además de ser muy fácil de utilizar, contaba con las características necesarias para implementar funcionalidades como la sincronización de pistas. Otra característica que se tuvo en cuenta es la capacidad de personalizar el color de la onda. Esto resultaba interesante teniendo en cuenta que podía adaptarse a la tonalidad de la aplicación web.

3.1.4 Grabación y reproducción de video por *streaming*

La solución debía permitir reproducir y grabar audio y video desde la propia aplicación web en tiempo real. Por este motivo fue necesario investigar tres cosas: una tecnología que permitiese recibir los datos en el servidor, una que permitiese enviar los datos desde la aplicación web y un protocolo de comunicación que fuera compatible con ambas tecnologías.

Las tecnologías que permiten cubrir con el primer punto se conocen como *media servers*. A partir de la investigación surgieron varias tecnologías candidatas a utilizarse. A continuación se describen cada una de ellas.

Flash Media Server (FMS)[9] Es un *media server* multiplataforma desarrollado por Adobe que permite servir contenido multimedia a una serie de dispositivos que soporten Flash. Las

[9] http://www.adobe.com/products/adobe-media-server-family.html

aplicaciones Flash clientes se conectan al FMS mediante el protocolo RTMP (*Real-Time Media Protocol*) para enviar y recibir datos. Esta tecnología no solo permite reproducir contenido multimedia sino que también admite la grabación mediante el envío de datos. Tanto la reproducción como la grabación se realiza mediante *streaming*. Además, la aplicación cliente es capaz de realizar llamadas RPC (*Remote Procedure Call*) al servidor y viceversa con el fin de transmitir otro tipo de datos. Como es una tecnología que requiere pagar una licencia para su utilización se descartó desde un principio.

Wowza Media Server[10] fue otro de los evaluados. Es multiplataforma y cumple con la misma funcionalidad que FMS pero soporta una variedad de protocolos y puede realizar *streaming* a diversos dispositivos que no necesariamente tienen porque contar con Flash, lo cual no ocurría con FMS.

Otra interesante ventaja de Wowza es que permite una sencilla instalación en casi cualquier servicio de *hosting* en la nube, en particular en servicios como EC2 (*Elastic Cloud Computing*) de Amazon Web Services. Proporciona imágenes pre-configuradas del motor de *streaming* que están listas para ser desplegadas en el servicio. Las facilidades de despliegue en este tipo de servicios son un punto a favor teniendo en cuenta la posibilidad de escalar el producto a futuro.

A pesar de las ventajas que proporciona Wowza y el prestigio que tiene dentro de los *media servers* disponibles, se terminó descartando como *media server* por el mismo motivo que FMS.

WebRTC (**Web *Real-Time Communicatio*n**)[11]**,** a diferencia de las tecnologías mencionadas anteriormente, WebRTC no es un *media server*. WebRTC es una API JavaScript que está siendo desarrollada por la World Wide Web Consortium (W3C) para permitir a las aplicaciones web realizar llamadas de voz, chat de video y uso compartido de archivos P2P (*Peer to Peer*) dentro del navegador sin la necesidad de *plugins* de terceros, como por ejemplo, Flash.

En un principio pareció una opción muy interesante dado que es una tecnología nativa a cada navegador. Además, se encontraron algunos ejemplos funcionales en los cuales el equipo podría basarse. Sin embargo, luego de probar cada uno de estos se llegó a la conclusión de que la tecnología todavía estaba muy inestable y no permitía realizar la grabaciones con una calidad aceptable.

[10] http://www.wowza.com/products/streaming-engine/
[11] http://www.webrtc.org/

Red5[12] es un *media server open source* desarrollado en Java que provee características similares a las ofrecidas por FMS y Wowza. Permite reproducir videos por *streaming* en los formatos: FLV, F4V, MP4, 3GP y audio en los formatos: MP3, F4A, M4A, AAC. Por otro lado, la grabación solamente permite generar videos en formato FLV con audio AAC. Este *media server* también hace uso del protocolo RTMP.

Se terminó optando por Red5 para desarrollar el segundo prototipo por ser el único *media server* de los evaluados que es *open source* y que permitía una calidad de grabación de audio y video aceptable.

Para el desarrollo de un cliente para el *media server* se evaluaron las tecnologías Flash y Silverlight. De la investigación surgió que Silverlight no soporta el protocolo RTMP utilizado por Red5. Flash, en cambio, no solo lo soporta sino que es el cliente estándar para este tipo de protocolo. Además, durante la investigación se encontraron algunas aplicaciones de ejemplo en Flash que permitieron probar la grabación y reproducción de video sobre una instalación de Red5 realizada por el equipo [10]. Finalmente, se terminó optando por Flash para el desarrollo de la aplicación de grabación de videos.

3.2 Prototipación

En esta sección se pretende describir el proceso de prototipación realizado para la construcción del producto. En total se realizaron cinco prototipos. Los dos primeros correspondieron fundamentalmente a probar las tecnologías seleccionadas. Los siguientes tres prototipos fueron evoluciones de los anteriores en cuanto a estética, usabilidad y cumplimiento de otros requerimientos. Para cada prototipo se menciona su alcance, la validación realizada y las principales conclusiones obtenidas.

3.2.1 Prototipo 1 – Mezclado de videos

Al inicio del desarrollo de este prototipo se investigaron diferentes tecnologías para el manejo de audio y video (ver sección 3.1.2 y 3.1.3). Luego de esta investigación se comenzó con el desarrollo de un prototipo que tuvo como alcance las siguientes funcionalidades:

- Subir cuatro videos a un sitio web.
- Extraer las pistas de audio de los videos.
- Presentar gráficamente las ondas de sonido de las cuatro pistas.
- Permitir moverlas horizontalmente mediante *drag and drop* para sincronizarlas.
- Reproducir todas las pistas de audio para verificar la sincronización.
- Generar el video mosaico teniendo en cuenta los desfasajes introducidos en la sincronización.

[12] http://www.red5.org/

En la Figura 8 se muestra como se representaban las ondas de audio extraídas de los videos subidos.

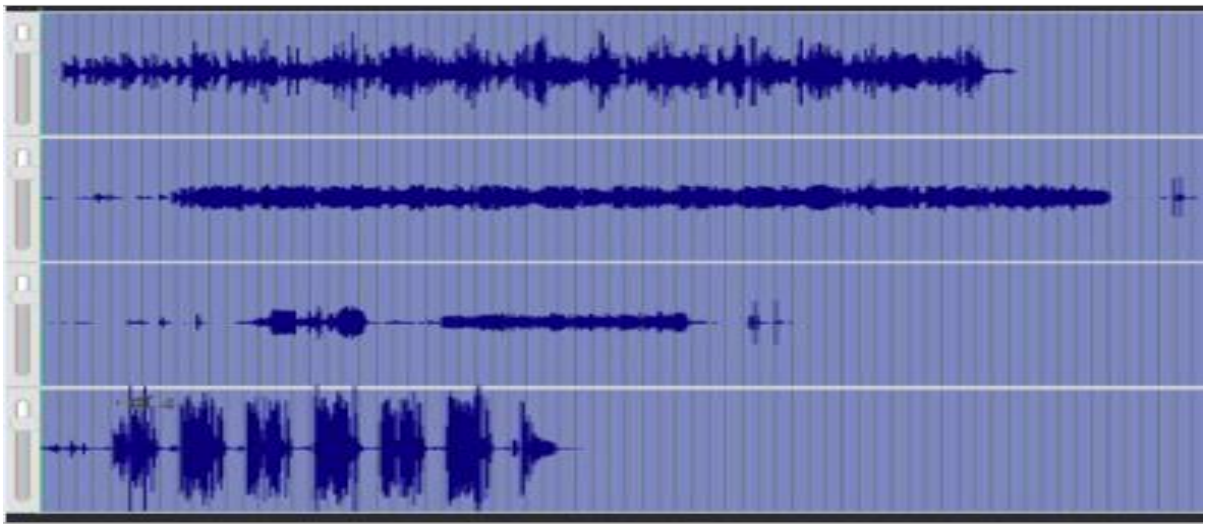


Figura 8: Representación de ondas en el prototipo 1

Debido a un defecto encontrado en la librería FFmpeg, el cual imposibilitaba agregar múltiples desfasajes entre los videos, se buscó una solución alternativa que permitiera solucionar este problema. Esta solución implicó utilizar una nueva librería para el manejo de audio. La librería utilizada en este caso fue Sox [11]. En el Anexo 2 se describe el problema anterior con más detalle.

Al finalizar el desarrollo del prototipo se realizó una sesión de validación con los clientes. Utilizando la *webcam* de una computadora, se grabaron cuatro videos en donde cada uno de los clientes cantaba o tocaba la guitarra sobre una base de la canción "La Bamba". Una vez subidos los cuatro videos se sincronizaron todas las pistas mediante *drag and drop*. Cuando finalizó la sincronización el resultado fue un video mosaico con todas las pistas de audio sincronizadas.

Lo más destacado de esta validación fue que al cliente le resultó difícil realizar la sincronización. Esto se debió a que le hizo falta algún tipo de funcionalidades que facilitaran la tarea, por ejemplo, poder realizar un zoom sobre las ondas de las pistas. En el Anexo 3 se encuentran todas las mejoras y nuevas funcionalidades surgidas a partir esta validación.

3.2.2 Prototipo 2 – Grabación desde la web

Al finalizar la validación del primer prototipo se decidió que el próximo paso sería investigar y probar la grabación de video desde la web (ver tecnologías en sección 3.1.4). Se acordó con el cliente que las funcionalidades a incluir para este prototipo serian las siguientes:

- Grabar videos desde la propia web.
- Generar el flujo de crear un proyecto.

- Generar el flujo de colaborar en un proyecto.
- Poder utilizar un metrónomo como referencia durante la grabación.
- Mejorar aspectos en el proceso de sincronización relevados en el prototipo 1.

En este prototipo se permitía elegir dos roles, el creador que creaba un proyecto y el colaborador que colaboraba en el mismo. Independientemente del rol con el cual el usuario ingresaba, se le presentaban dos opciones, subir o grabar un video. Ambos utilizaban la misma página para realizar la grabación. La diferencia era que el colaborador podía escuchar la pista de audio grabada por el creador. Esta pista se reproducía mientras este grababa su video. Por otro lado, el creador que no poseía una pista de referencia, tenía la opción de escuchar un metrónomo.

En la Figura 9 se muestra la aplicación Flash desarrollada para grabar los videos desde la web.

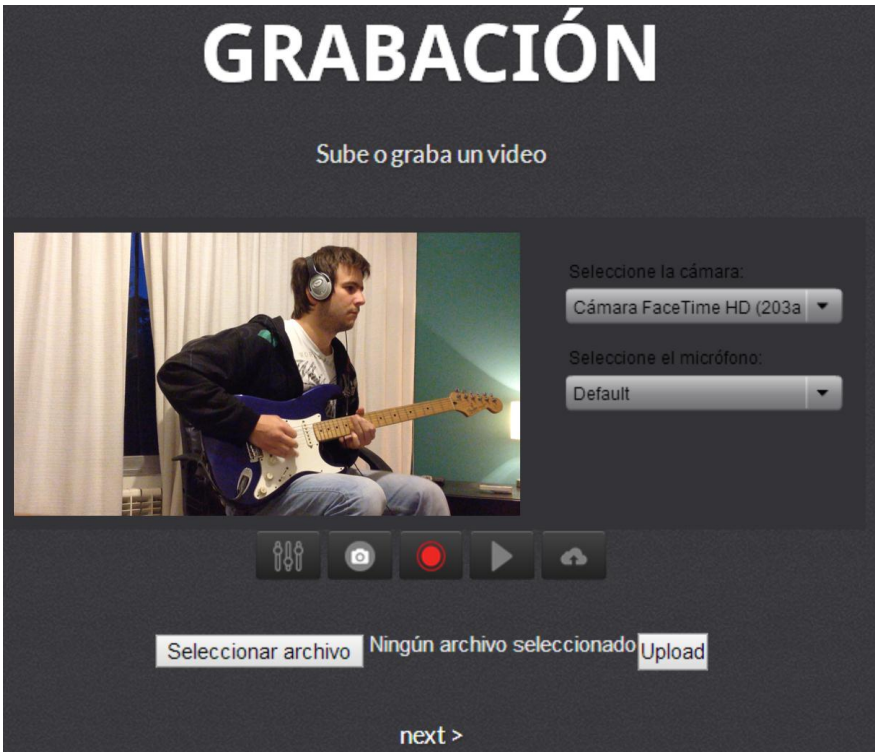


Figura 9: Página de grabación del prototipo 2

Finalizado el desarrollo del prototipo, se realizó una validación del mismo con los clientes. La validación consistió en realizar una sesión de grabación similar a la realizada en el primer prototipo. La diferencia era que en este se brindaba la opción de grabar los videos desde la propia aplicación. Después de grabar, se procedió a sincronizar las pistas de audio utilizando las mejoras implementadas.

Se concluyó que las nuevas mejoras habían ayudado a reducir los tiempos para sincronizar las pistas. A su vez, la grabación de los videos desde el navegador funcionó como se

esperaba. No obstante, se identificaron algunas oportunidades de mejora. En el Anexo 4 se encuentran las conclusiones surgidas a partir de la validación.

3.2.3 Prototipo 3 – Desarrollo de flujos principales

Luego de desarrollar y validar los prototipos anteriores, se concluyó que las tecnologías seleccionadas en un principio eran las adecuadas para continuar con el desarrollo del producto. Se definió que a partir del prototipo 3 inclusive, los prototipos se enfocarían en otros aspectos del producto que no habían sido tenidos en cuenta hasta el momento.
Para este prototipo se plantearon las siguientes metas:

- Implementar una estructura que soporte la creación de varios proyectos (incorporación de persistencia).
- Poder escuchar el audio de un proyecto antes de colaborar en el mismo.
- Manejo de usuarios, poder autenticarse en la web.
- Mostrar los proyectos en curso (Backstage).
- Mostrar los proyectos finalizados y sus respectivos videos (Stage).
- Mostrar los proyectos del usuario autenticado (Mis Proyectos).
- Evaluar algún servicio que permita realizar el despliegue del prototipo.

Los conceptos de Stage, Backstage y Rec Room ya estaban definidos antes de comenzar a desarrollar la solución. Sin embargo, aparecieron recién en este prototipo bajo la necesidad de comenzar a darle personalidad y contexto a la aplicación. Para esto fue necesario introducir un mecanismo de persistencia y de este modo dar soporte a múltiples usuarios y proyectos. En el prototipo anterior, el que interpretaba el rol de creador o colaborador era siempre un mismo usuario. Incorporar un manejo de usuarios permitió distinguir mejor estos roles.

Otra de las incapacidades del prototipo anterior era que cuando se colaboraba en un proyecto, al grabar siempre se escuchaba como referencia únicamente la pista de audio del creador. Por este motivo, se definió que el creador tuviera la posibilidad de generar lo que llamamos, audio del proyecto. El audio del proyecto corresponde a la mezcla sincronizada de la pista de audio del creador más las pistas aportadas por los colaboradores. El audio del proyecto es el que escuchan los potenciales colaboradores en el Backstage y el que tendrán disponible para escuchar como referencia al grabar, en el caso de que decidan colaborar.

En la Figura 10 se muestra la página del Backstage con el audio de los proyectos abiertos.

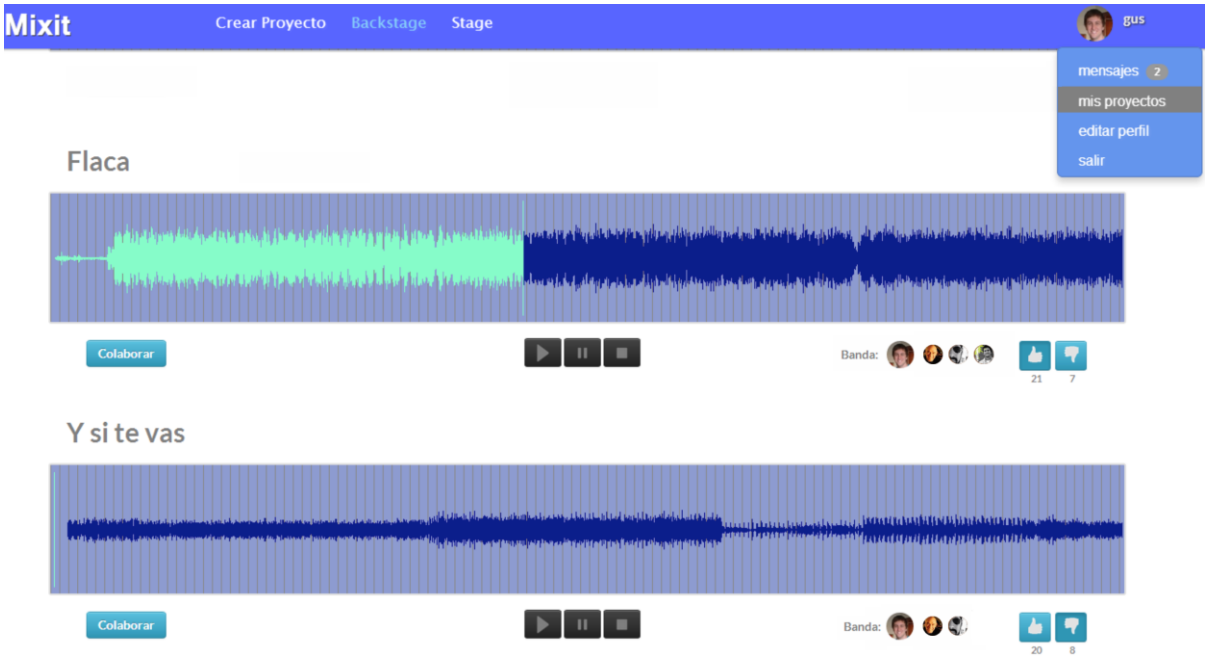


Figura 10: Sección Backstage del prototipo 3

Para que un usuario pudiera acceder a sus proyectos se definió una nueva sección llamada "Mis proyectos". En esta sección un usuario podía visualizar el progreso de todos sus proyectos y sus datos más relevantes. Lo más importante es que esta sección era el punto de acceso para sincronizar las pistas de cada uno de sus proyectos en curso.

Otra de las metas que se planteó fue elegir algún servicio en la nube que permitiera realizar el despliegue de la solución. El servicio elegido fue uno provisto por Amazon Web Services, EC2 (*Elastic Cloud Computing*[13]). Este servicio provee instancias de maquinas virtuales en la nube. La justificación de porque se optó por este servicio se encuentra en el Anexo 5.

Una vez desplegada la solución de este prototipo en una instancia EC2 creada por el equipo, se realizaron algunas pruebas de cómo esta se comportaba en el ambiente. Dichas pruebas no resultaron tan satisfactorias como se esperaba. El servidor de *streaming* que era el encargado de grabar y reproducir los videos no funcionaba. Se propuso para el próximo prototipo investigar cual era el problema.

Finalmente, para validar lo realizado, se organizó una sesión de grabación con un músico amateur que no conocía de que se trataba el proyecto. El resultado fue que se le tuvo que

[13] http://aws.amazon.com/es/ec2/

explicar todas las funcionalidades de la aplicación para que la pudiera utilizar. Las principales conclusiones obtenidas fueron que el producto tenía que ser más auto explicativo y que en la aplicación de grabación se debía incorporar una barra que indicará el volumen del micrófono. Esto último era para no saturar el sonido en la grabación. Además de la validación anterior, se realizo una sesión de grabación más intensa con los clientes en la cual se grabaron varias canciones. La idea era simular el flujo completo, donde el creador creaba un proyecto y varios colaboradores colaboraban. Luego de esta validación, los clientes le notificaron al equipo que próximamente se integraría un diseñador gráfico al equipo. El objetivo de esto era mejorar la calidad estética del producto. En el Anexo 6 se muestran todas las pantallas en este prototipo.

En la Figura 11 se muestra a uno de los clientes realizando una grabación en el prototipo 3.

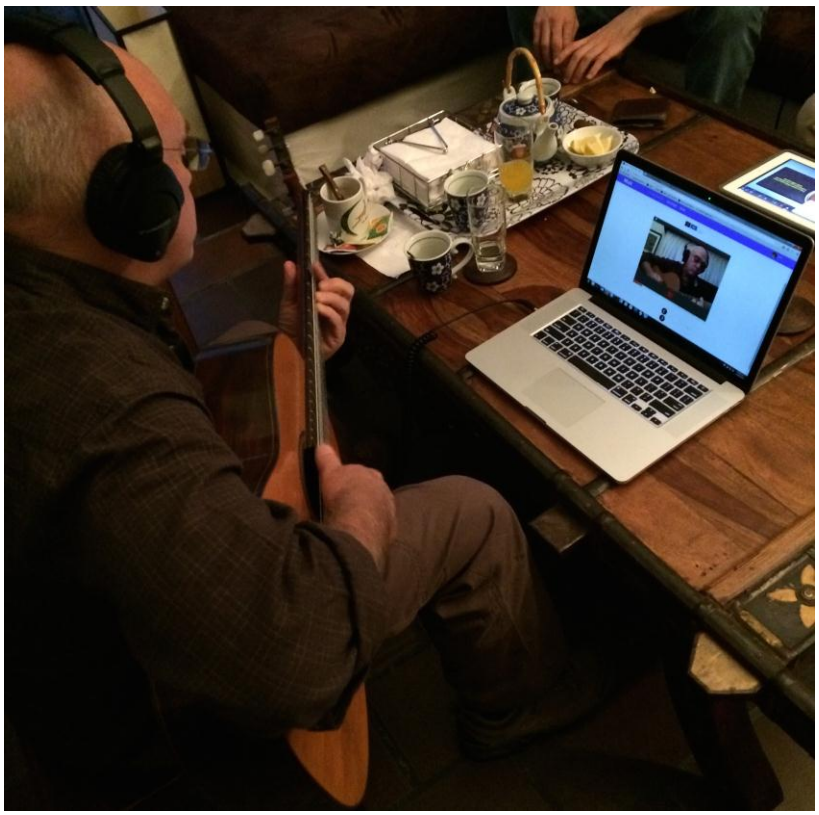


Figura 11: Uno de los clientes grabando en el prototipo 3

Como conclusión general con respecto al desarrollo y validación de este prototipo, se puede decir que salvo por los problemas experimentados con el despliegue la solución en AWS, se cumplieron todas las metas planteadas.

3.2.4 Prototipo 4 – Remodelado de la interfaz de usuario

Las principales metas de este prototipo fueron:

- Mejorar la calidad gráfica de aplicación.
- Mejorar aspectos de usabilidad, hacer que el producto sea más auto explicativo.
- Continuar buscando soluciones a los problemas experimentados en AWS.

Al inicio del desarrollo de este prototipo el equipo se reunió en algunas oportunidades con un diseñador gráfico provisto por los clientes. Durante estas reuniones se discutieron diferentes formas de representar las principales secciones de Mixit!. En el Anexo 7, se muestran algunas de las primeras páginas creadas por el diseñador.

Luego de dos semanas de discutir ideas con el diseñador, este comunicó que no podría seguir con el trabajo. Una de las metas de este prototipo era mejorar el atributo de calidad "Mostrabilidad" del producto. Por lo tanto, el equipo debió realizar un esfuerzo extra para remodelar la interfaz de usuario sin el apoyo del diseñador, ya que este solo proveyó imágenes y no HTML. De modo de agilizar esta remodelación se optó por utilizar un *framework* para desarrollo de interfaces gráficas. El *framework* elegido en esa instancia fue Twitter Bootstrap. Utilizar Bootstrap ayudó a agilizar el desarrollo ya que provee un conjunto de elementos de UI pre desarrollados. Además, permitió mantener una consistencia en cuanto a formatos y colores de todos los componentes dentro de la aplicación. Este esfuerzo fue realizado tratando de mejorar la estética de la aplicación.

Por otro lado, con respecto a la usabilidad, se trató de hacer que la aplicación fuera lo más auto explicativa posible. Se definió una Home que explicara de forma gráfica las principales funcionalidades del producto. En la Figura 12 se puede observar la página de inicio o Home de la aplicación web.

Figura 12: Home Page del prototipo 4

En la sección Backstage se decidió mostrar dentro de un mosaico y para cada proyecto, las fotos de los instrumentos (roles) faltantes y las fotos de los músicos que ya colaboraron sobre el proyecto. En el Anexo 8 se muestran las capturas de los cambios más relevantes sobre la UI realizados en este prototipo.

Lo último realizado para este prototipo fue tratar de mejorar la configuración del servidor de *streaming* y la aplicación Flash de grabación de videos para que funcione en la instancia EC2 de AWS. Si bien se pudo lograr que grabe, no se pudo obtener un comportamiento predecible ya que muchas veces se la grabación se cortaba.

Luego, se realizó una muestra de la aplicación a los clientes. Se obtuvo que se debía re-estructurar el Rec Room y agregar algunas funcionalidades nuevas en la sección de sincronizar pistas.

Se concluyó que la falta del diseñador tuvo como consecuencia una dedicación extra por parte del equipo para mejorar la calidad grafica de la aplicación. Este fue el motivo principal por el cual no se pudo dedicar demasiado tiempo a solucionar los problemas en AWS. Todo este esfuerzo extra impacto en la dedicación horaria del equipo. Sin embargo, valió la pena porque se obtuvo un producto que superó las expectativas del cliente.

3.2.5 Prototipo 5 – Integración diseño y mejoras usabilidad

Al finalizar el prototipo 4, se integró un nuevo diseñador gráfico al equipo. Este diseñador se encargaría de darle una imagen a Mixit!, que era lo que el cliente buscaba. Las metas planteadas para este prototipo fueron:

- Mejorar la estructura del Rec Room.
- Agregar nuevas funcionalidades para la sincronización de las pistas.
- Integrar imágenes realizadas especialmente para Mixit!.
- Probar la aplicación con otros músicos para encontrar problemas de usabilidad.

De todo el nuevo diseño realizado en el prototipo 4, el diseñador trabajó sobre la estética de las páginas más importantes. Estas fueron: la Home, el Stage, el Backstage y el Rec Room. El principal cambio consistió en agregar nuevos colores e imágenes. En Anexo 9 se adjuntan algunas imágenes del nuevo diseño.

La re-estructura del Rec Room consistió en dividirlo en tres columnas. En la primera se mostraba la configuración del metrónomo, en la segunda la aplicación Flash encargada de grabar los videos y en la tercera miniaturas de los últimos cuatro intentos de grabación.

Las nuevas funcionalidades que se incorporaron en la sección de sincronizar pistas fueron:

- Filtrar aportes por instrumento.
- Aceptar colaboración.
- Rechazar colaboración.
- Elegir las pistas a publicar (las que entran dentro de la mezcla del audio del proyecto).

- Mostrar el icono del instrumento de la pista.

En la Figura 13 se muestra la sección de sincronizar pistas con las nuevas funcionalidades (iconos sobre cada pista) y con algunas imágenes provistas por el diseñador.

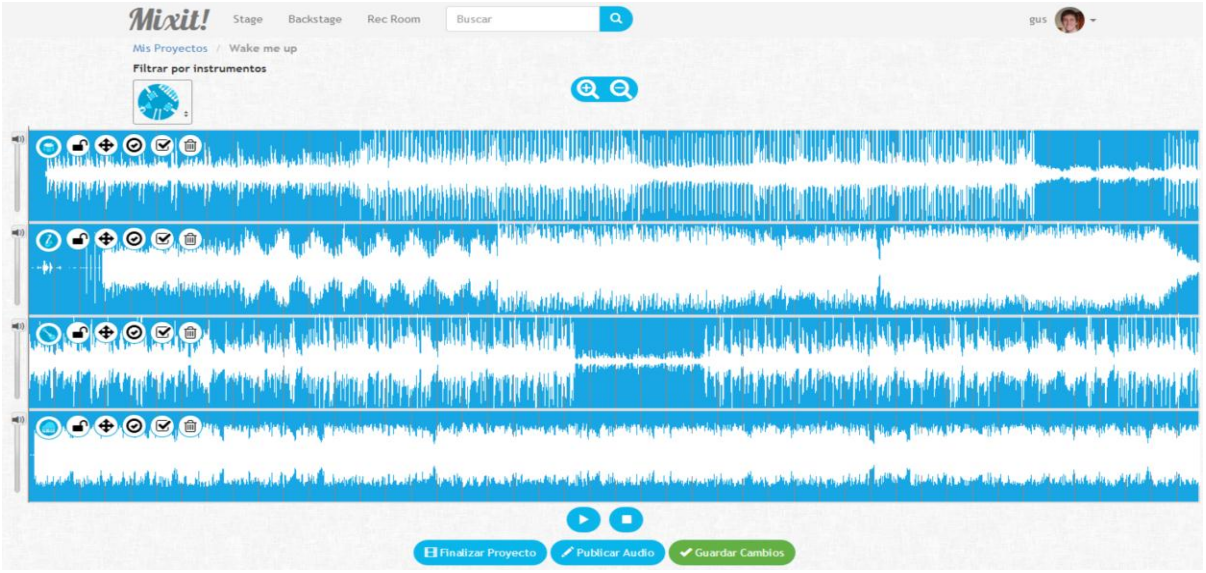


Figura 13: Sección Sincronizar Pistas del prototipo 5

En el Stage, además de los cambios visuales, se agregó la funcionalidad de realizar comentarios sobre los videos.

En el Anexo 10 se encuentras problemas relevados para este prototipo por un músico.

Lo que más aportó el diseñador fueron algunas imágenes de nuevos instrumentos. Una vez finalizada la integración de los recursos del diseñador, se mostraron los cambios al cliente de las nuevas pantallas. El resultado fue que quedó sorprendido cuando vio el producto en funcionamiento. La estructura auto explicativa, las imágenes y las animaciones agregadas le dieron una identidad al producto. Esto era lo que se buscaba desde el principio. Lo destacado en este prototipo fue, el trabajo en conjunto con el diseñador y la buena relación que se tuvo. Además se cerró una etapa importante, la del desarrollo. En el Anexo 11 se encuentran el resto de las pantallas de este prototipo.

3.3 Arquitectura candidata

En esta sección se describen las principales características, componentes y decisiones de diseño tomadas para la construcción de la arquitectura presentada como, arquitectura candidata. Se considera a la arquitectura como candidata por el contexto del proyecto, en donde el ciclo de vida evolutivo puede generar cambios sobre la misma.

3.3.1 Principales características

Las principales características de la arquitectura se derivan de los siguientes requerimientos no funcionales definidos en la sección 2.4: Modificabilidad, Extensibilidad y Escalabilidad. A continuación se describen cada una de estas características vinculándolas con los requerimientos no funcionales que pretenden satisfacer.

- **Exposición del *back-end* a través de servicios REST:** En comparación con otro tipo de servicios web, como por ejemplo SOAP. Los servicios REST son más livianos, proveen mejores tiempos de respuesta y los clientes son más sencillos de implementar. Este tipo de servicios son un estándar dentro de la industria para proveer lógica de negocio o servicios a dispositivos con recursos limitados [12]. Por este motivo, se decidió exponer las operaciones del *back-end* a través de REST. Esto ayudó a cumplir con el requerimiento no funcional de **Extensibilidad**, permitiendo llevar el servicio de Mixit! a *smartphones* sin mayor impacto sobre la arquitectura.

- **Soporte para agregar nuevos *templates* de videos:** El alcance del producto contemplaba crear únicamente proyectos colaborativos de cuatro participantes. Sin embargo, está dentro de lo deseado poder extenderlo a más o menos participantes y a su vez a permitir generar videos con distintos *templates*. Por este motivo, se parametrizaron todos los comandos de FFmpeg que generan los videos mosaicos. Los comandos se cargan desde archivos de configuración. Esto permite generar distintos *templates* agregando nuevos archivos de configuración, sin necesidad de modificar, compilar o desplegar el código nuevamente. Esta solución permitió cumplir con el requerimiento no funcional de **Modificabilidad**.

- **Procesamiento de los mosaicos de forma asincrónica:** Se detectó a partir de algunas pruebas realizadas tanto localmente como en AWS que al generar los mosaicos, los procesos del servidor acaparaban cerca del 100% de la capacidad de la CPU. Teniendo en cuenta esto se realizó una breve investigación de como se manejaban los procesamientos de videos en sistemas ya establecidos, como por ejemplo, YouTube y Vimeo. Se concluyó que, con el fin de prevenir cuellos de botella y no penalizar el resto de la solución, el procesamiento de los videos se debería realizar de forma asincrónica, en un nodo dedicado. Se definió que una vez finalizado este proceso asincrónico se enviaría un email al usuario notificándolo de que su video ya había sido procesado y

publicado en el Stage. La mejor alternativa encontrada para resolver este problema utilizando tecnologías del *stack* JEE fue a través de un MDB[14] (*Message Driven Bean*). Todo lo mencionado anteriormente favorece la **escalabilidad** del sistema.

3.3.2 Descripción de la arquitectura

La solución arquitectónica se divide en dos grandes módulos: el *front-end* y el *back-end*. En el *front-end* se encuentra la interfaz gráfica de la aplicación y en el *back-end* se encuentra lo referente a la lógica de negocio, el acceso a datos, el procesamiento y *streaming* de videos.

El *front-end* interactúa con el *back-end* a través de interfaces REST y por medio del protocolo RTMP para el caso del *streaming* de videos. En la Figura 14 se muestra un diagrama en el cual se indican los módulos *back-end* y *front-end*, sus principales componentes y las dependencias entre los mismos.

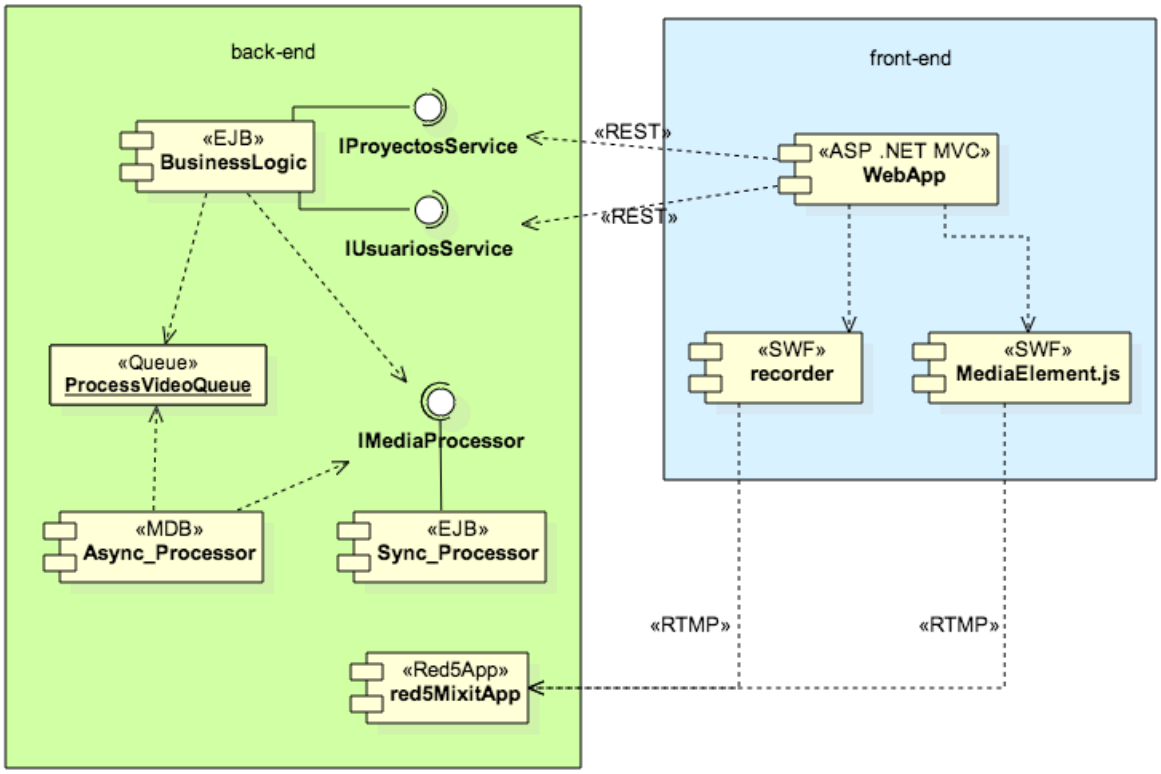


Figura 14: Diagrama de componentes de la solución

A continuación se describen brevemente los componentes que integran el *back-end*:

- **BusinessLogic** expone dos servicios REST. Uno para todas las operaciones sobre los proyectos (crear, colaborar, mezclar, finalizar, extraer audio, etcétera) y otro sobre las operaciones de los usuarios (registrarse, autenticarse, etcétera). BusinessLogic es la

[14] Un MDB es un *Enterprise Bean* de Java que permite el procesamiento de mensajes de forma asincrónica. Interactúa con una cola de mensajes y va desencolando mensajes a medida que los procesa.

fachada con la cual interactúa el *front-end* para realizar todas las operaciones del sistema. Además, se encarga del acceso a datos y la interacción con los procesadores sincrónico y asincrónico. Este componente se implementó como un EJB (*Enterprise Java Bean*).

- **Sync_Processor** encapsula toda la lógica de procesamiento de audio y video. Esto se logra ejecutando comandos de las librerías de FFmpeg y Sox. Todos estos comandos se encuentran parametrizados en archivos de configuración para una fácil modificación. Este componente se implementó como un EJB y sus principales responsabilidades son:
    - Crear los videos mosaicos.
    - Extraer *snapshots* de los videos.
    - Separar los audios de los videos.

- **Async_Processor** tiene como propósito atender los mensajes con solicitudes de procesamiento de mosaicos. Estos mensajes son encolados por BusinessLogic y enviados a procesar por Sync_Processor. Terminado el procesamiento se encarga de marcar el video como finalizado y de enviarle un email al usuario creador del proyecto. Se implementó como un MDB.

- **red5MixitApp** es una aplicación web que corre en el servidor Apache[15] provisto por el *media server* Red5. Es la encargada de aceptar conexiones de los clientes Flash para reproducir y grabar videos. Además posee funciones utilitarias que permiten verificar por ejemplo, si todos los datos de un video grabado han llegado al servidor.

Dentro del *front-end* se encuentran los siguientes módulos:

- **WebApp** es la capa de presentación de la aplicación y se comunica con los servicios REST expuestos por el *back-end*. Se encarga de presentar todas las vistas al usuario. Se implemento como un proyecto en ASP .NET MVC.

- **Recorder** es una aplicación Flash que se encarga de la grabación de los videos. Se comunica con red5MixtApp por medio del protocolo RTMP para enviar los *streams* de video.

- **MediaElement.js**[16] es una aplicación Flash de terceros que se encarga de la reproducción de los videos. Se comunica con red5MixtApp por medio del protocolo RTMP para recibir los *streams* de video.

[15] http://www.apache.org
[16] http://mediaelementjs.com/

3.3.2.1 Vista lógica de los principales componentes

En la Figura 15 se muestra la vista lógica del paquete en el que se encuentran las interfaces que expone **BusinesLogic** y las clases que las implementan.

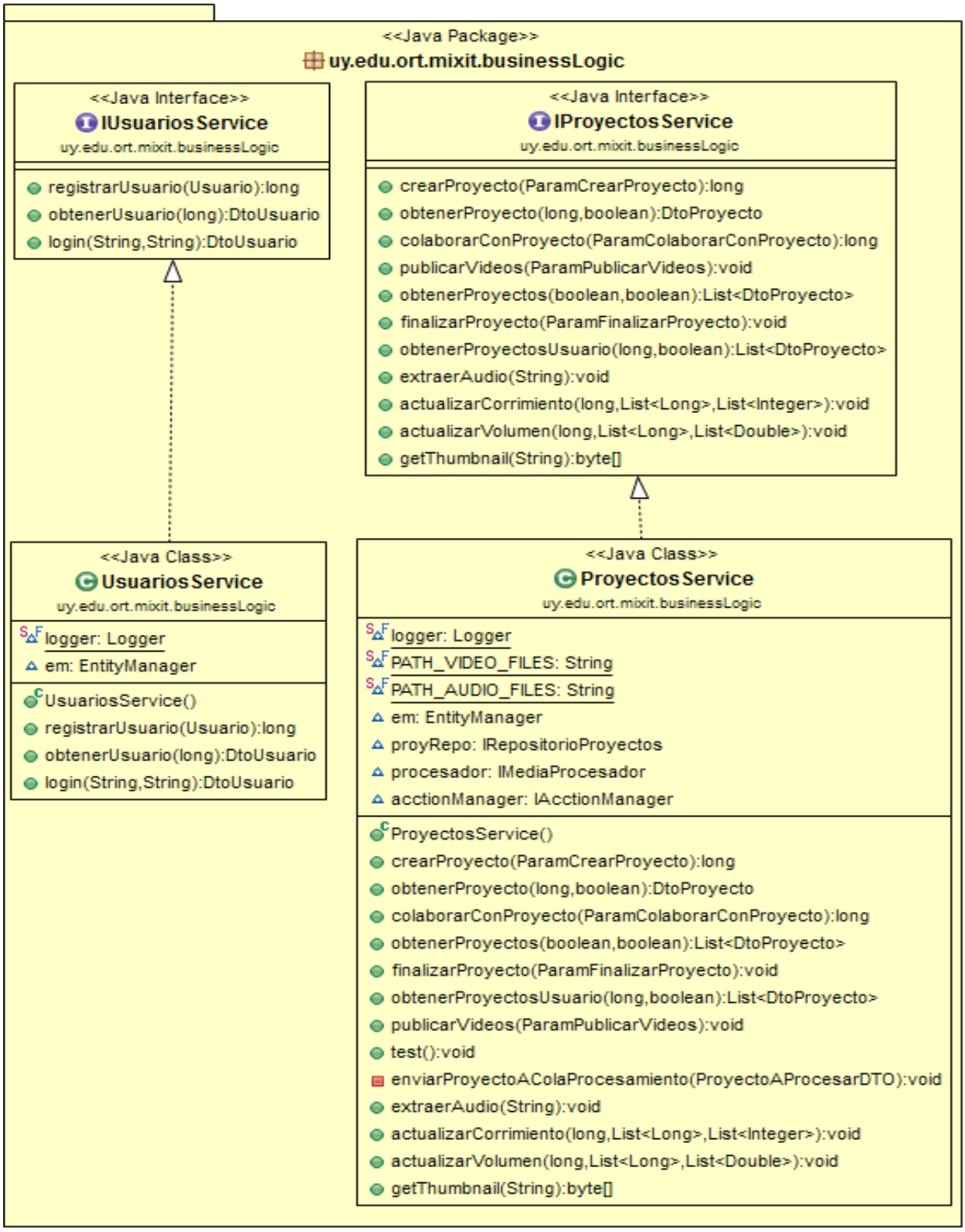


Figura 15: Vista lógica del paquete BusinessLogic

El la Figura 16 se muestra la vista lógica de clases y paquetes del componente **Sync_Processor**.

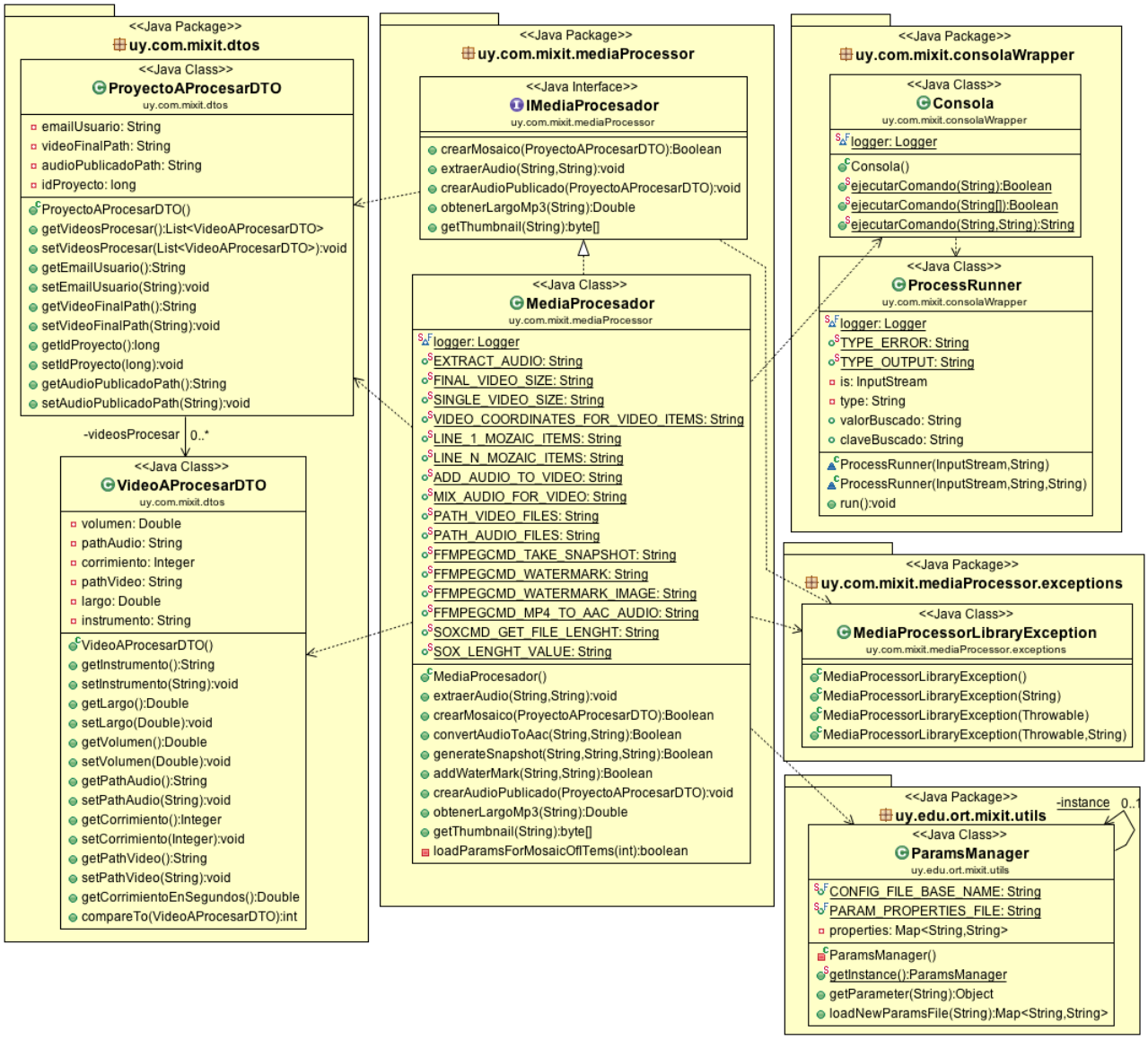


Figura 16: Vista lógica de clases y paquetes del componente Sync_Processor

En la Figura 17 se muestra la vista lógica de clases y paquetes del componente **Async_Processor.**

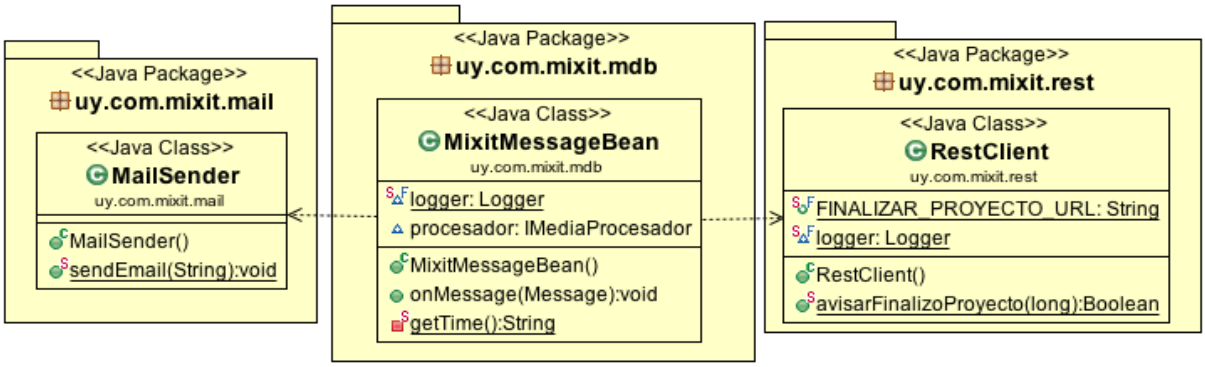


Figura 17: Vista lógica de clases y paquetes del componente Async_Processor

3.3.2.2 Proceso de creación de un video mosaico

En la Figura 18 se muestra un diagrama de interacción del proceso de creación de un video mosaico. Desde que el usuario finaliza un proyecto, hasta que recibe el email avisándole que su video se ha publicado en el Stage.

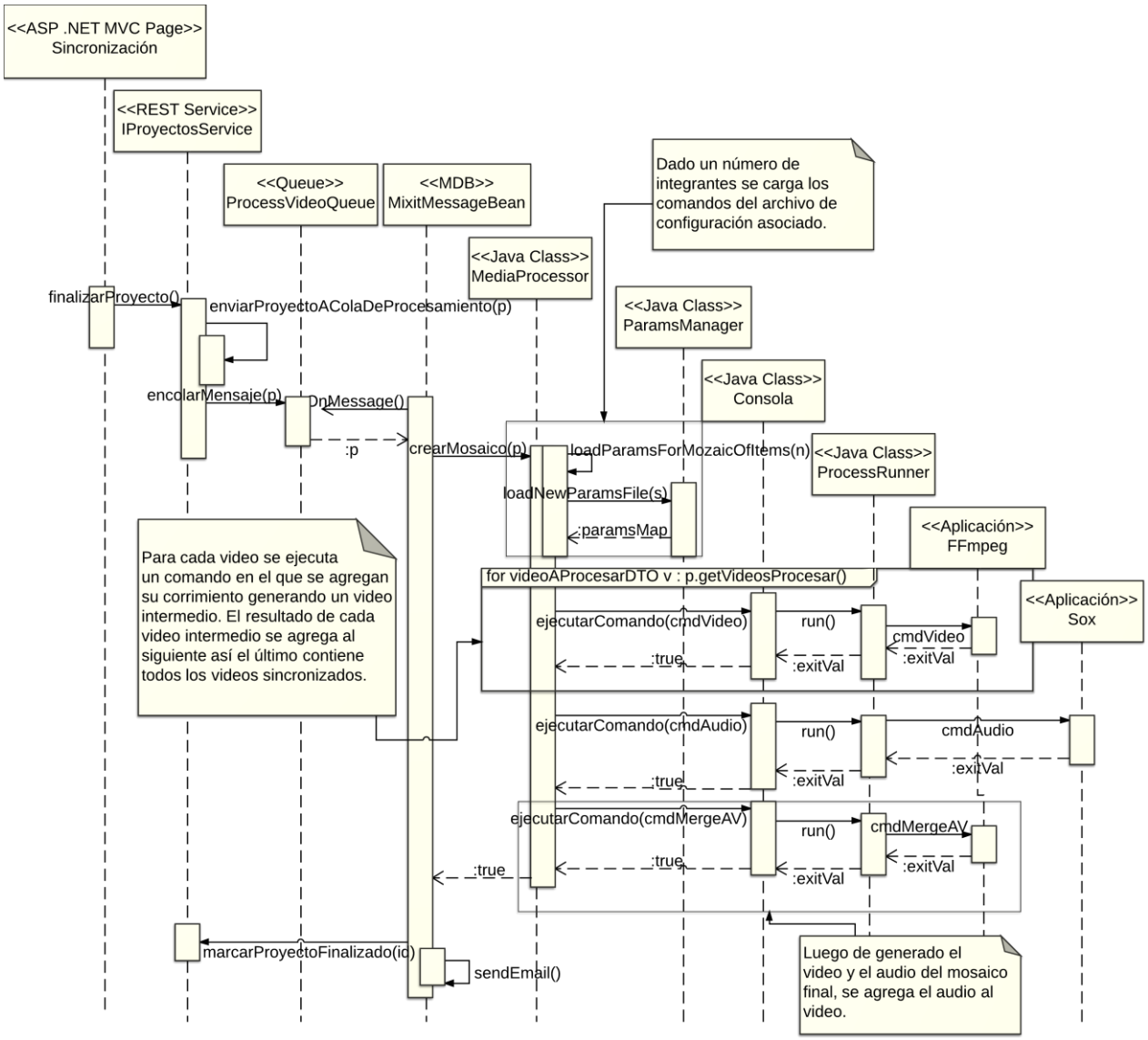


Figura 18: Diagrama de interacción de procesamiento de un video mosaico

Cuando un proyecto se marca como finalizado en la página de sincronizar pistas, la aplicación web se lo comunica al *back-end* mediante el método finalizarProyecto de la interfaz IProyectosService. Luego, BussinessLogic se encarga de crear un mensaje con todos los datos necesarios para el procesamiento y lo encola en ProcessVideoQueue. Async_Processor desencola el mensaje y lo envía a procesar a Sync_Processor. En Sync_Processor, MediaProcessor recibe el mensaje y carga del archivo de configuración los comandos necesarios. Para cada video del proyecto se envía a Consola los comandos con los parámetros necesarios. Consola recibe los comandos y por medio de ProcessRunner los envía a FFmpeg para generar videos temporales intermedios. Cada video temporal intermedio contiene al video anterior. El último generado es el video mosaico. Una vez generado el video mosaico se prepara el comando para realizar la mezcla de los audios teniendo en cuenta los desfasajes y volúmenes introducidos en la sincronización. Este comando es enviado a Sox para procesarlo. Una vez que finaliza ese proceso se ejecuta otro comando de FFmpeg que integra el audio en el video mosaico.

Luego de este proceso se marca el proyecto como finalizado y se envía un email al usuario notificándole que su proyecto se encuentra publicado en el Stage.

3.3.2.3 Vista de despliegue

El la Figura 19 se muestra una vista de despliegue con todos los componentes de la solución.

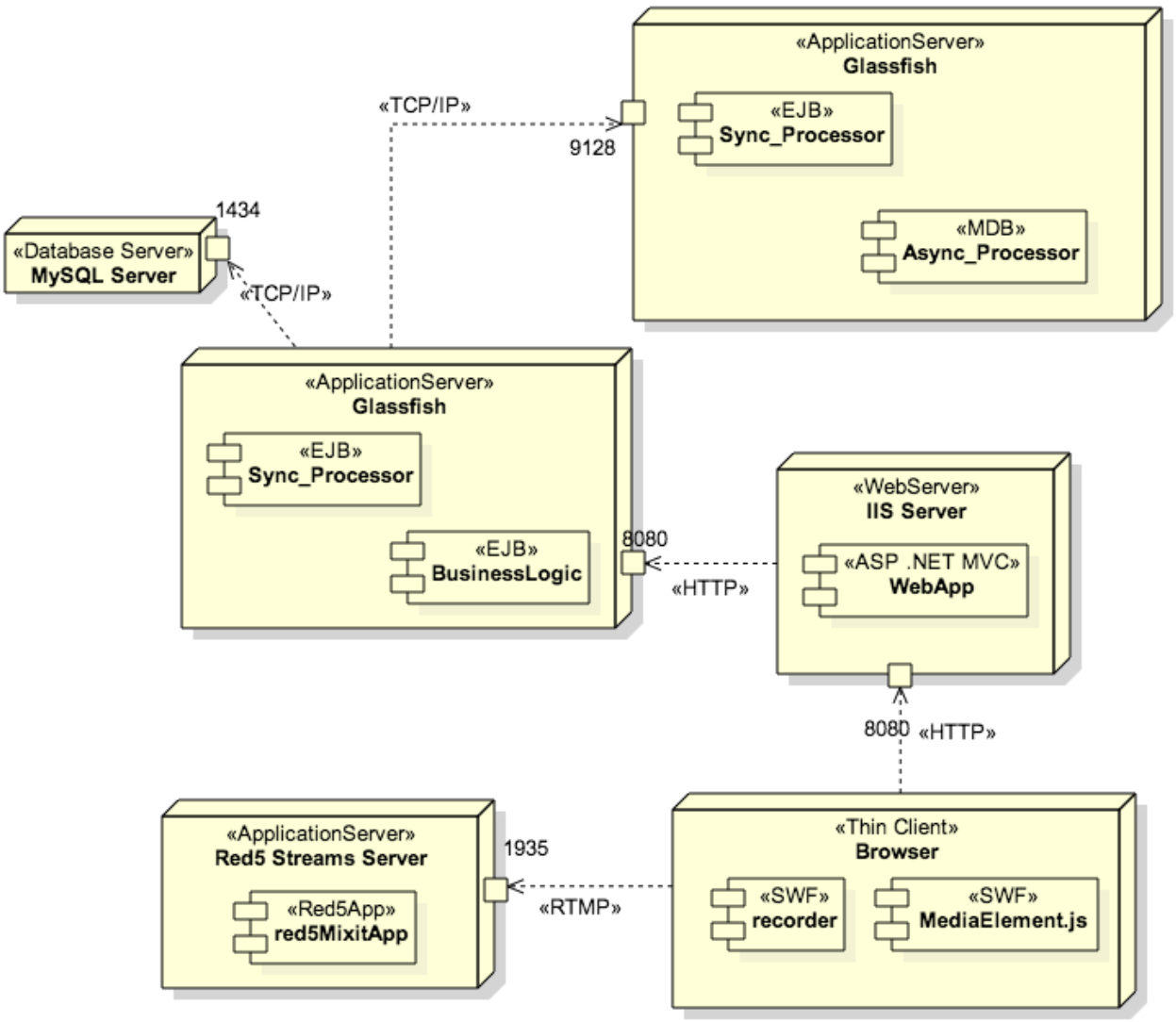


Figura 19: Diagrama de despliegue de la solución

Los componentes se separaron en las siguientes capas físicas (patrón N-Tiers) en función de sus responsabilidades:

- Interfaz Gráfica.
- Lógica de negocio.
- Persistencia de datos.
- Procesamiento asincrónico de videos finales.
- Servidor de *streaming*.

4 Gestión de la configuración

En esta sección se describe el proceso elegido para gestionar la configuración. Se especifican los elementos de configuración de software (ECS) identificados y las herramientas utilizadas, así como algunas decisiones tomadas a lo largo del proceso. Este proceso tuvo como objetivo proveerle al equipo una estructura organizada sobre la cual se pudieran guardar y recuperar en cualquier momento todos los ECS generados durante el proyecto, minimizando los defectos y mejorando la productividad del equipo.

4.1 ECS identificados

4.1.1 Documentos

A esta categoría pertenecen los documentos propios del desarrollo del proyecto y los generados para el marco académico.

Documentos asociados al proyecto:

- Apuntes de relevamiento y minutas de reunión con los clientes
- Bocetos de UI de baja definición
- Apuntes, fotografías y resultados de validaciones de prototipos
- Material sobre investigación tecnológica
- Planillas referentes a la gestión del proyecto
    - Evolución de riesgos
    - Planificación de iteraciones
    - Registro de horas
    - Retrospectivas de las iteraciones

Documentos académicos:

- Grabaciones de voz de revisiones y reuniones con el tutor
- Diapositivas de las revisiones
- Informes finales de las revisiones
- Documentación final

4.1.2 Recursos multimedia

Durante el proyecto fue muy frecuente el uso de videos pre-grabados o archivos de audio como datos de prueba para generar videos colaborativos. Estos archivos debían ser compartidos por todos los miembros del equipo.

4.1.3 Librerías

Las librerías *open source* utilizadas para el manejo de audio y video en el desarrollo de la solución suelen cambiar rápidamente de versión. Frecuentemente, además de introducir correcciones*,* cambian su comportamiento o generan incompatibilidades. Al inicio del desarrollo cada uno ingresó a la página oficial de la librería, descargó la última versión y la instaló en su computadora. Se generaron diferencias de versiones entre las instalaciones, produciendo cambios en las interfaces de las librerías dependiendo de la versión. Esto llevó a cambios constantes en el código de algunos componentes para poder interactuar con una versión específica, generando conflictos con los demás. Por este motivo se optó por guardar una única versión de cada librería en un repositorio común.

4.1.4 Código fuente

Dentro del repositorio de código fuente se almacenaron: los proyectos de los cinco prototipos desarrollados (*front-end* y *back-end*), el proyecto de la aplicación de grabación Flash, scripts SQL con datos de prueba y algunos archivos *batch* necesarios para compilar y ejecutar algunas de las aplicaciones.

4.2 Herramientas

Microsoft Onedrive

Este servicio fue el elegido como repositorio de todos los ECS contenidos en la categoría Documentos. OneDrive permite almacenar todo tipo de archivos. Sin embargo, el equipo optó por utilizarlo principalmente por su compatibilidad con los archivos de Microsoft Office. El servicio permite visualizar y editar documentos (.docx), presentaciones (.pptx) y planillas Excel (.xlsx) de forma online y distribuida. La ventaja que este posee sobre otros productos similares como, por ejemplo, Google Docs, es que al descargar los archivos estos siguen manteniendo su formato original. Esta característica fue de especial utilidad para evitar problemas de formato en la elaboración de la documentación académica final. Además OneDrive mantiene copias de versiones anteriores de cada archivo, lo cual puede ser útil en el caso de realizar un cambio involuntario. Otra función de utilidad es la que permite reproducir de forma online los archivos de audio. Esto evitó, por ejemplo, tener que descargar las grabaciones de voz para escucharlas.

Dropbox

Dropbox fue utilizado como repositorio para los recursos multimedia y los archivos de las librerías. El servicio permite crear espacios de almacenamiento privados para compartir archivos entre distintos usuarios. Además provee un cliente *desktop* que se integra con el explorador de windows. Esto hizo que la herramienta fuera la más adecuada para compartir

archivos con mucha rotación y gran tamaño. Un ejemplo de estos fueron los videos de prueba.

4.3 Versionado del código

4.3.1 Sistema de control de versiones

En el inicio de la definición de las actividades de SCM el equipo manejo dos alternativas a utilizar como sistema de control de versiones. Estas fueron GIT y SVN. Desde un inicio se preveía que durante el desarrollo sería necesario crear nuevos *branches* para implementar y probar funcionalidades o mejoras que fueran surgiendo sin interferir con el resto de los desarrollos. GIT posee un mejor manejo de *branches* que SVN. Esto y su naturaleza distribuida (repositorio local y remoto) hacen con que sea mucho más fácil evitar conflictos de código.

Por otro lado, todos los integrantes poseían experiencia con SVN en el ámbito académico y únicamente uno de ellos con GIT en el ámbito laboral. Por lo cual, además de lo anterior, se optó por la segunda opción para aprender a utilizar un sistema de control de versiones que viene tomando cada vez más adeptos en el mundo del desarrollo de software.

4.3.2 *Hosting* del repositorio

El servicio elegido para hospedar el repositorio de código fuente fue GitHub. Se optó por este porque es uno de los más utilizados y probados en la actualidad. Además, nos permitía asociar cada *commit* a un determinado *ticket* en Assembla, que fue la primera herramienta que el equipo utilizo para la planificación de las tareas de desarrollo. En la Figura 20 se muestra la estructura del repositorio de código fuente en GitHub.

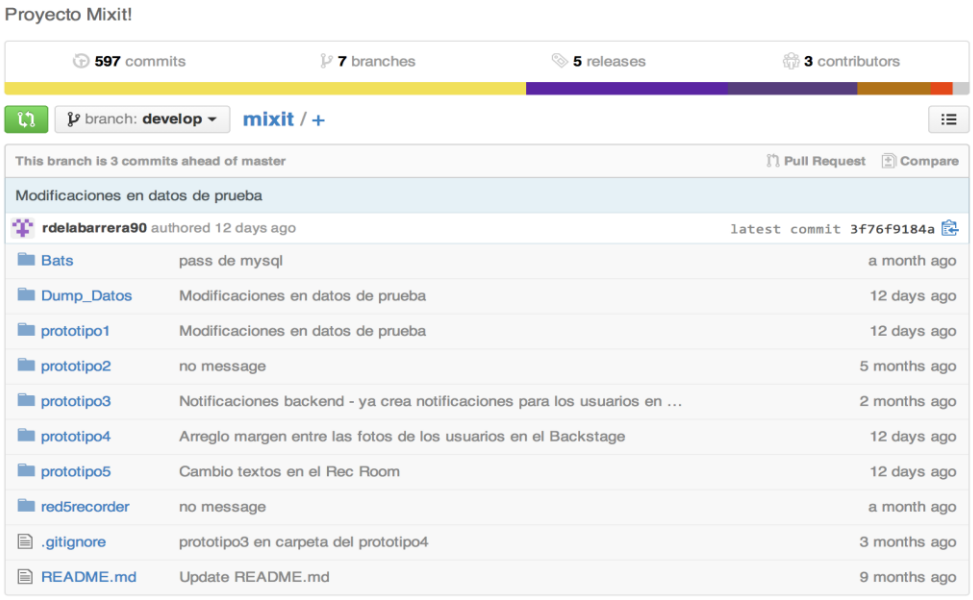


Figura 20: Estructura del repositorio de código fuente

4.4 Respaldos

Teniendo en cuenta que todos los ECS se guardaban en la nube, se optó por realizar un respaldo semanal completo de cada uno de los repositorios: Dropbox, Onedrive y GitHub. Esto tuvo como objetivo contar con una copia local extra en disco duro ante una posible pérdida de datos. El encargado de realizar estos respaldos fue el responsable de SCM.

5 Gestión del proyecto

En esta sección se describe de que forma ha sido gestionado el proyecto. Se justifica la elección del ciclo de vida y se describe la metodología de trabajo utilizada por el equipo. Asimismo, se presenta como se gestionó el alcance y se analiza el esfuerzo dedicado a lo largo de todo el proyecto.

5.1 Ciclo de vida

Al ser un concepto nuevo y experimental el que propone Mixit!, el equipo sabía que los requerimientos del producto iban a evolucionar durante el proyecto. A medida que se avanzará con el proyecto, nuevos requerimientos y cambios iban a surgir. Por esta razón, fue necesaria la utilización de un ciclo de vida evolutivo basado en el desarrollo de prototipos. Esto tuvo como objetivo ayudar a ir comprendiendo la problemática y definiendo los principales conceptos del producto de forma evolutiva.

5.2 Metodología de trabajo

5.2.1 Valores del equipo

Dada la naturaleza del proyecto y su contexto, se eligió un enfoque de gestión ágil. Los valores del equipo se basaron en el manifiesto ágil [13]. En la Tabla 2 se muestra un mapeo entre los valores ágiles y los del equipo.

Valores del manifiesto ágil	Valores del equipo
Individuos e interacciones sobre procesos y herramientas.	Se priorizó la comunicación verbal e informal durante todo el proyecto.
Software funcionando sobre documentación excesiva.	La prioridad siempre fue mostrar avances mediante los prototipos y validar los conceptos. No se generó más documentación que la necesaria.
Colaboración con el cliente sobre negociación contractual.	Se negoció con el cliente que funciones aportaban más valor al producto. El alcance era considerado variable.
Respuesta ante el cambio sobre seguimiento de un plan.	El equipo se adaptó a los cambios. Lo importante era agregar valor al producto y no seguir una planificación estricta a largo plazo.

Tabla 2: Valores ágiles y valores del equipo

5.2.2 Adaptación de *Scrum*

Se buscó un marco de trabajo que se adapte al equipo y al tipo de proyecto. Todos los integrantes del equipo ya poseían experiencia con *Scrum* en el ámbito laboral. Además, *Scrum* es un *framework* que se adapta muy bien a proyectos en contextos cambiantes y con

requerimientos poco definidos. Teniendo en cuenta esto, se optó por utilizarlo como base para definir el marco metodológico.

Por las características del proyecto y por ser un equipo de tres personas, en donde cada uno tenia diferentes disponibilidades horarias (por trabajo y estudios) fue necesario realizar algunos ajustes a esta metodología.

Los ajustes realizados a *Scrum* fueron:

- Las *daily meetings* se realizaban cada dos o tres días y en la mayoría de las casos no eran presenciales.
- La capacidad del equipo era medida en horas en vez de *story points* por iteración.

Las similitudes con *Scrum* fueron:

- Iteraciones de duración fija de dos semanas.
- Se realizaron todas las ceremonias: *planning*, *review* y *retrospective*.
- Se utilizaron los artefactos: *product backlog* y *sprint backlog*.
- El gerente de proyecto fue el responsable del rol de *scrum master*.
- Los clientes ocuparon el rol de *product owner*.
- Se utilizo una unidad arbitraria para la estimaciones (Sucesión de Fibonacci).
- Se evaluaba el progreso de las iteraciones mediante un *burndown*.

5.2.3 *Product backlog*

Todos los requerimientos que surgían a partir de las actividades de relevamiento o validación se registraban en un *backlog* y se priorizaban por el cliente. Este *backlog* se consideraba abierto ya que siempre surgían nuevos requerimientos o mejoras a medida que se mostraban avances y se validaban los prototipos. En el Anexo 12, se encuentra una lista con algunas de las funcionalidades registradas por el equipo.

5.2.4 Proceso realizado en cada iteración

Al comenzar cada iteración se realizaba un *sprint planning*. En esta ceremonia se generaba el *sprint backlog* mediante el siguiente proceso:

1. Se definían las metas que se querían alcanzar.
2. Se definía el esfuerzo que el equipo le iba a dedicar (en horas).
3. Se seleccionaba una porción de las funcionalidades del *backlog* (según prioridad) y se las dividía en tareas para luego estimarlas.
4. Se asignaban al *sprint backlog* las funcionalidades estimadas hasta igualar el esfuerzo pactado.

Durante el desarrollo de la iteración, cada integrante del equipo se auto asignaba tareas según su afinidad con las mismas. Las tareas se mostraban en un *task board* virtual en el cual

se marcaba el progreso de cada una de ellas en la iteración. En la Tabla 3 se muestran los estados que podía tener una tarea durante la iteración.

Estado	Descripción
To do	Tarea lista para desarrollar
In progress	Tarea en progreso
In ice	Tarea que estaba en progreso y por algún motivo se tuvo que detener
To test	Tarea lista para ser probada
Done	Tarea terminada, cuando se podía verificar que cumpliera con lo especificado

Tabla 3: Estado de las tareas

En la Figura 21 se muestra una captura del *task board* virtual utilizado para gestionar las tareas.

Figura 21: *Task board* de herramienta Scrumwise

Al finalizar cada iteración se realizaba una *retrospective*. La *retrospective* consistía en analizar lo que había salido bien y mal durante la iteración. A partir de este análisis, se obtenían los aspectos a mejorar para la siguiente iteración.

5.3 Cronograma de *releases*

A medida que se desarrollaban los prototipos, se definía en conjunto con los clientes, los principales objetivos que se debían cumplir y una fecha tentativa de *release*. En la Figura 22 se muestran los objetivos y las fechas de release fijadas para cada prototipo. Se destaca que

la generación de este cronograma se realizó de forma dinámica, ya que hasta el último trimestre de proyecto no se sabia a cuantos prototipos se iba a llegar.

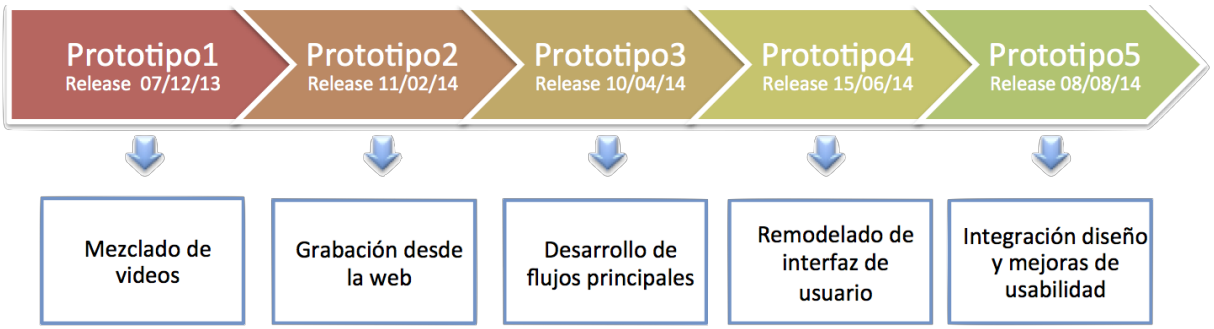


Figura 22: Cronograma de *releases*

5.4 Gestión del alcance

Para definir el alcance del proyecto se debió definir primero el alcance del producto. El alcance del producto se definió en base a lo que se conoce como MVP, *Minimal Viable Product* (Producto Mínimo Viable en inglés). Un MVP consiste en un producto que tiene las mínimas funcionalidades necesarias para poder realizar pruebas con potenciales usuarios, los cuales luego proporcionarán *feedback* acerca del mismo. Se considera que a partir del tercer prototipo ya se había alcanzado esta cota de funcionalidades mínima. Estas funcionalidades corresponden a las tratadas en la sección 2.3. Una vez logrado el MVP, el alcance pasó a ser variable. El *product backlog* paso a ser abierto y se comenzaron a agregar al mismo un conjunto de nuevas funcionalidades o mejoras surgidas a partir de las validaciones. A partir de las nuevas funcionalidades y por lo discutido con el cliente se definió que el producto debía tender a un formato tipo red social.

5.5 Esfuerzo

Esta sección describe y analiza el esfuerzo dedicado a lo largo del proyecto. En la misma, se analiza el esfuerzo dedicado en la primera etapa del proyecto, en donde se realizó el relevamiento inicial. Luego, se analiza el esfuerzo dedicado para todas las actividades que involucró el desarrollo de cada prototipo. Finalmente se muestra una división del esfuerzo total por las distintas áreas del proyecto.

5.5.1 Relevamiento inicial

Durante el relevamiento inicial no se registraron todas las horas dedicadas. Esto se debió a que el equipo recién se estaba comenzando a gestionar y no se tenía el hábito de registrar el esfuerzo. Se estima que para esta etapa se dedicaron 147 horas. Este valor se calculó teniendo en cuenta las reuniones con los clientes, con el tutor, el tiempo dedicado en investigar sobre conceptos del dominio y en diseñar los bocetos de UI.

5.5.2 Prototipos

Para el esfuerzo registrado en las actividades de desarrollo de cada prototipo, se tuvo en cuenta además de la investigación y codificación, las reuniones con los clientes, músicos amateurs/profesionales y el tutor. En la Tabla 4 se muestra el esfuerzo dedicado en horas para cada prototipo y un breve análisis del error en las estimaciones.

Prototipo	Estimado	Real	Error	Análisis del error
1	122	252	106%	Este error es consecuencia de haber subestimado el alcance de las investigaciones y la complejidad de las librerías a utilizar.
2	190	297	56%	Este error se debe a que las tareas de investigación no eran especificas y a la complejidad enfrentada al momento de utilizar el servidor de *streaming*.
3	351	384	9%	Esta mejora en la estimación se debió a que las tareas a realizar se volvieron más concretas.
4	410	513	25%	Esto se debe a que se había planificado incorporar al equipo un diseñador provisto por el cliente pero esto no sucedió. Por lo que, se tuvo que invertir más esfuerzo que el planificado en actividades de remodelación de la interfaz gráfica.
5	160	155	3%	Ocurrió por primera vez que no se superó el esfuerzo estimado.

Tabla 4: Esfuerzo dedicado en horas para cada prototipo

En la Tabla 5 se muestran las iteraciones que abarcó cada prototipo con su fecha de inicio y fin.

	Inicio	Fin	Iteraciones			
Prototipo 1	24/10/2013	04/12/2013	1	2	3	
Prototipo 2	05/12/2013	08/02/2014	4	5	6	
Prototipo 3	09/02/2014	06/04/2014	7	8	9	10
Prototipo 4	07/04/2014	14/06/2014	11	12	13	14
Prototipo 5	15/06/2014	26/07/2014	15	16	17	

Tabla 5: Iteraciones y fechas de prototipos

A continuación, en la Figura 23 y Figura 24 se muestra la evolución del esfuerzo real y el estimado por prototipos y por iteraciones respectivamente.

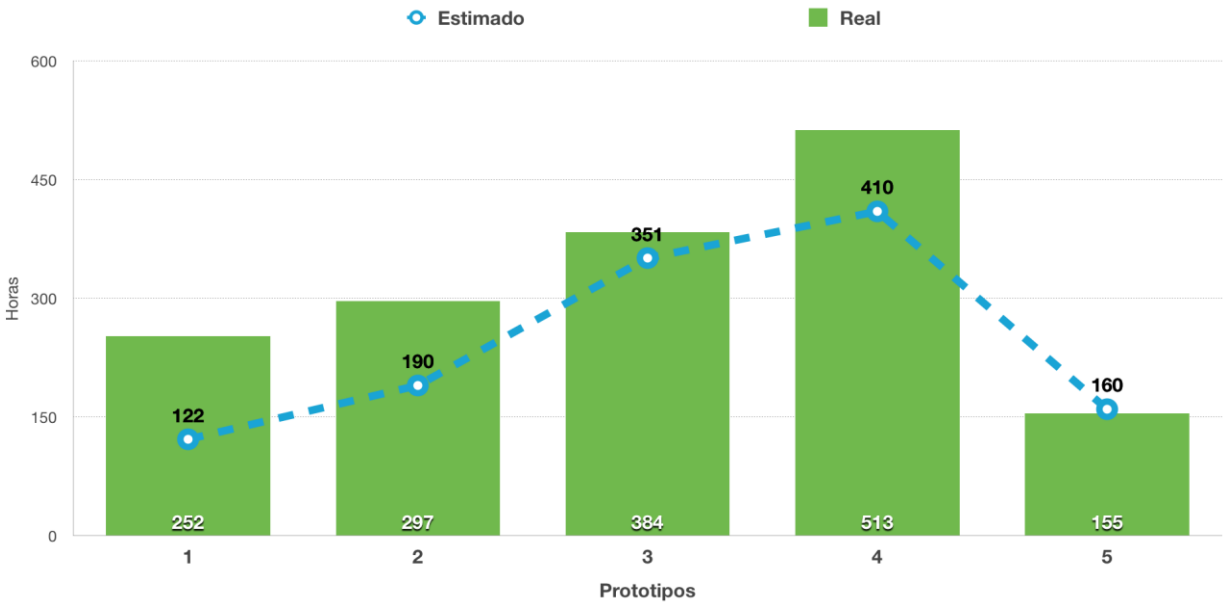


Figura 23: Evolución del esfuerzo real vs estimado para cada prototipo

Se puede concluir que para los prototipos los prototipos 1 y 2, el error en las estimaciones se debe a subestimar las tecnologías a utilizar y a que las tareas de investigación no eran lo suficientemente concretas. En el prototipo 4 el error se debió a la falta de un recurso pactado con el cliente que fue el diseñador.

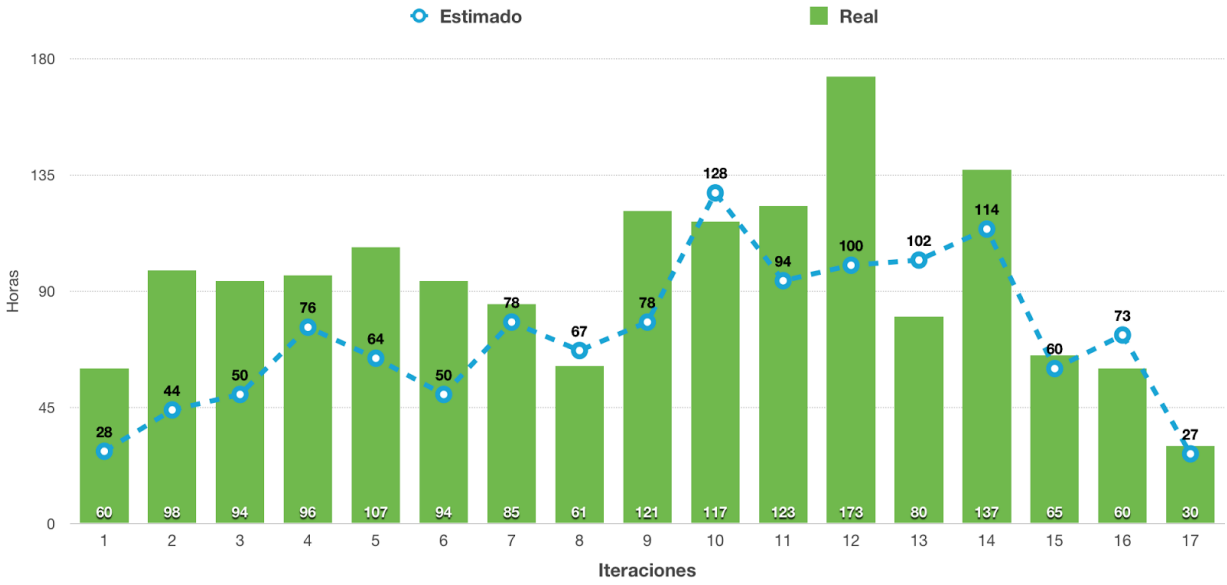


Figura 24: Evolución del esfuerzo real vs estimado en cada iteración

5.5.3 Esfuerzo total y por áreas

En la **Figura 25** y

Tabla 6 se puede apreciar el esfuerzo dedicado en las distintas áreas del proyecto.

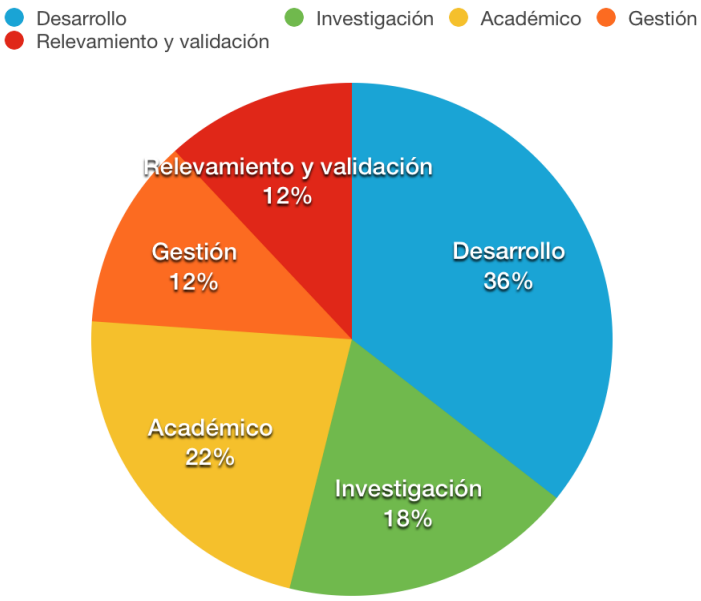


Figura 25: Gráfica de esfuerzo dedicado por área

Actividad	Horas	Descripción
Desarrollo	801	Horas dedicadas al desarrollo y *testing* de los prototipos.
Académico	501	Horas invertidas en la documentación y en las tres revisiones académicas.
Investigación	410	Horas dedicadas en investigar las tecnologías a utilizar y a realizar pruebas de concepto.
Gestión	270	Horas dedicadas en gestionar las iteraciones y reuniones con tutor.
Relevamiento y Validación	267	Horas dedicadas en los bocetos de UI, reuniones con los clientes, diseñador y músicos amateurs/profesionales y a sesiones de observación y grabación.
TOTAL	2249	Total de horas dedicadas en el proyecto.

Tabla 6: Tabla de esfuerzo por área

Se concluye que se dedicó al proyecto un total de 2249 horas. Donde cada integrante aportó aproximadamente 750 horas en un año. Teniendo esto en cuenta, el promedio de dedicación semanal del equipo fue de 15,625 horas.

5.5.4 Herramientas de registro utilizadas

En el transcurso del proyecto las herramientas utilizadas para el registro de esfuerzo fueron variando. Esto se debió a la necesidad del equipo de realizar esta tarea de forma más eficaz y eficiente.

Durante el desarrollo del prototipo 1 y mitad del 2, se utilizó Assembla [14] para el registro de esfuerzo. Se seleccionó esta herramienta web por todas las funcionalidades que poseía. Por ejemplo, gráficas de avance, estadísticas por personas, poder asociar un *ticket* o tarea con un *commit* en GitHub, etcétera. Sin embargo, la misma era compleja de utilizar. Esto provocó que no se registrara todo el esfuerzo durante las iteraciones. Por este motivo, se descartó Assembla y se busco una herramienta más flexible y fácil de utilizar.

De la mitad del prototipo 2 y hasta finales del prototipo 3 se utilizó Excel. La herramienta era flexible y conocida por el equipo. Sin embargo, a medida que crecía el número de tareas realizadas en las iteraciones más difícil se volvía de utilizar. Por este motivo, se descartó Excel y se busco una herramienta con un enfoque en *Scrum*.

A finales del prototipo 3 y hasta el final del proyecto se utilizó Scrumwise [15]. Las ventajas que aportó esta herramienta fueron:

- Sencilla de utilizar.
- Permitía ver el progreso en tiempo real.
- Imposible de que exista una doble asignación de tareas.
- Mostraba el progreso de la iteración, a través de un *burndown*.
- Sencillo de visualizar el *product backlog*.
- Permitía agregarle *tags* a las tareas.
- Más amigable para los clientes cuando priorizaban las tareas.
- Aportaba un registro detallado de las horas dedicadas.
- Permitía agregarle prioridad a las tareas (baja, media, alta ,urgente).
- Contaba con un *task board* en el cual la tarea podría estar en diferentes estados.

6 Gestión de Riesgos

Desde el comienzo del proyecto se identificaron una serie de riesgos que podían influir en el éxito del mismo. A continuación se describe los principales riesgos identificados, un plan de mitigación para los mismos y como estos evolucionaron durante el desarrollo de los prototipos.

6.1 Principales riesgos identificados

- **Requerimientos poco claros y/o difusos:** Los clientes entendían la problemática y contaban con experiencia creando videos colaborativos. Sin embargo, no tenían claro que se debía hacer para resolver la problemática. Al ser un producto experimental y con alcance variable, el equipo sabía que a medida que avanzará iban a surgir nuevos requerimientos. No se sabía a que se iba a llegar al finalizar el proyecto.

- **Desconocimiento del dominio del problema:** Solo un integrante del equipo poseía experiencia como músico amateur. Existía un desconocimiento general sobre la teoría musical y como funcionaba la grabación y mezcla de sonido para crear composiciones.

- **Selección de las librerías para grabación y mezcla de audio y video:** Dado el alcance del proyecto se consideró que era inviable la implementación de librerías propias para la grabación y mezcla de audio y video. Para cada librería de terceros se tuvo en cuenta que se podían encontrar defectos. Estos defectos podrían generar grandes retrasos en el proyecto.

- **Baja disponibilidad de alguno de los integrantes del equipo:** Como era un equipo de tres personas, la baja disponibilidad de uno o más de los integrantes, comprometería seriamente el alcance y el calendario del proyecto.

- **Producto poco usable:** Al ser una aplicación web con funcionalidades complejas, podría suceder que no resultase intuitiva como para que un usuario sin conocimientos de edición de audio pudiera utilizarla. El tiempo de aprendizaje de la aplicación web debía ser corto. Como era necesario una gran inversión de tiempo en realizar investigaciones sobre tecnologías a utilizar, se podía descuidar este aspecto.

- **Pérdida de interés por parte del cliente:** Al tener poco conocimiento del dominio, era crítico mantener el contacto con los clientes. La pérdida de interés por parte de los mismos implicaba un producto con menor nivel de calidad. Era necesario mostrar avances rápidamente para mantener al cliente participativo. Sin una participación

continua por parte de los mismos, la utilización de un ciclo de vida evolutivo para el desarrollo de los prototipos no sería el apropiado.

6.2 Plan de mitigación

- **Requerimientos poco claros y/o difusos:** Se realizaban reuniones cada dos semanas con los clientes para entender la problemática a resolver. Además de las reuniones se aplicaron otras técnicas de relevamiento y análisis, como por ejemplo, el diseño de bocetos de UI. Esto definió cuáles eran las principales funcionalidades y flujos a tener en cuenta en los prototipos a desarrollar.

- **Desconocimiento del dominio del problema:** Para entender el dominio se investigaron los principales conceptos sobre la teoría musical. Se realizaron reuniones con músicos profesionales para entender el proceso de grabación y mezcla de pistas de audio. Además, el equipo se reunía periódicamente con los clientes, los cuales le trasmitían su conocimiento.

- **Selección de las librerías para grabación y mezcla de audio y video:** De todas las librerías de terceros que se investigaron, se le dio prioridad a las que eran *open source* y estaban bien documentadas. En el caso que se encontrara un *bug*, se podría pedir ayuda a la comunidad o de otra forma inspeccionar el código fuente y arreglarlo. De las librerías seleccionadas, se realizó una prueba de concepto para aprender a utilizarlas y no llevarse ninguna sorpresa cuando se aplicaran a los prototipos.

- **Baja disponibilidad de alguno de los integrantes del equipo:** En cada iteración se definía la disponibilidad de cada integrante. A través de las *daily meetings* se logró la transferencia de conocimiento necesaria para mitigar este riesgo. Esto aseguro que si alguno de los integrantes no podía dedicarle las horas pactadas, otro podría continuar con la tarea que este tenía asignada.

- **Producto poco usable:** A partir del desarrollo del prototipo 3, se comenzó a tener en cuenta este riesgo. Las evaluaciones heurísticas realizadas con un experto, las sesiones de observación y las grabaciones realizadas con músicos externos al grupo ayudaron a encontrar problemas de usabilidad en los prototipos.

- **Pérdida de interés por parte del cliente:** Realizando reuniones frecuentes con los clientes, en donde se mostraban avances de las funcionalidades implementadas, se logró mantener el interés. Los prototipos desarrollados se utilizaban en las sesiones de grabación, esto generaba que los clientes siempre estuvieran motivados con el avance del proyecto.

6.3 Evolución de los riesgos

En la Figura 26 se puede apreciar la evolución de la magnitud de los riesgos a lo largo del proyecto para cada prototipo. En el Anexo 13 se encuentra información sobre la probabilidad de ocurrencia e impacto para cada prototipo.

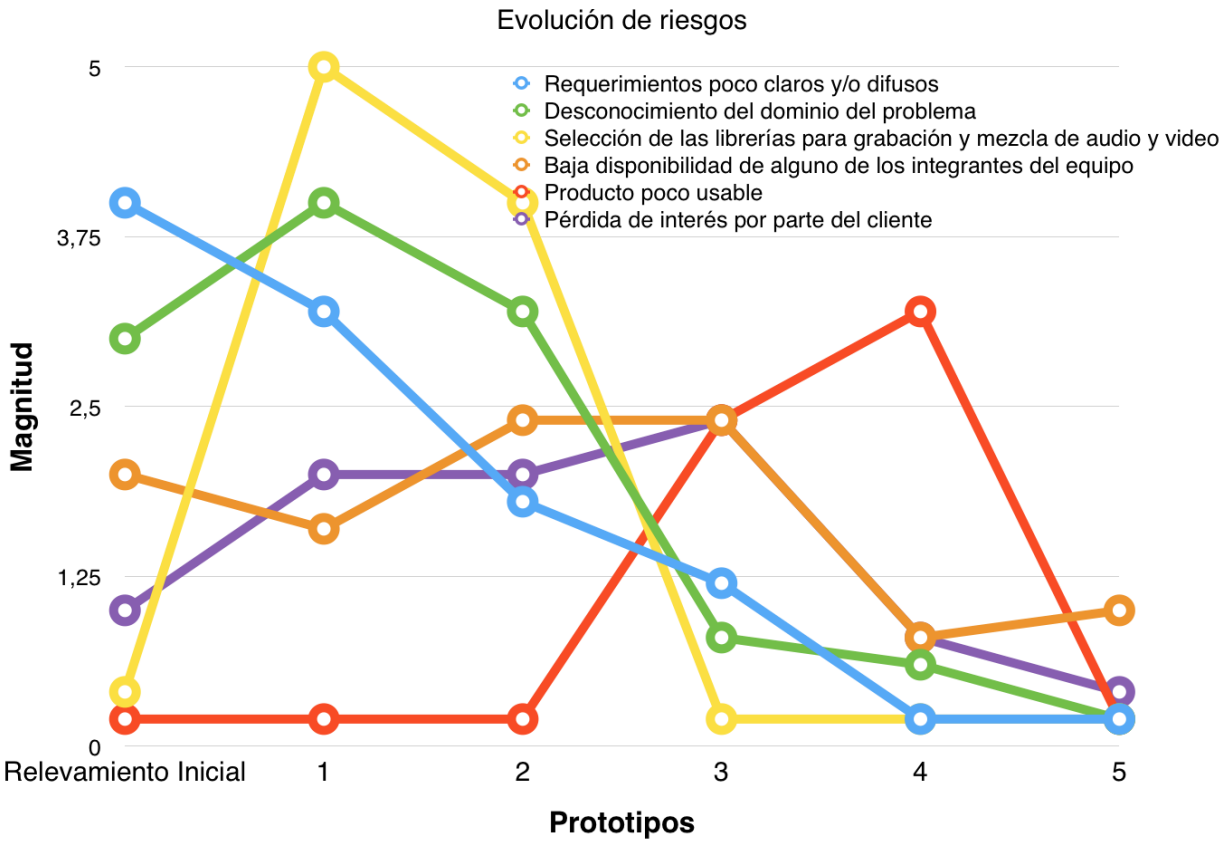


Figura 26: Evolución de los riesgos para los prototipos desarrollados

Observando la gráfica se concluye que a medida que se desarrollaban los prototipos, la mayoría de los riesgos iban bajando su magnitud. Los prototipos ayudaron a mitigar varios de los riesgos identificados en el proyecto. El relevamiento inicial ayudó a bajar el riesgo "Requerimientos poco claros y/o difusos". Los prototipos 1 y 2 ayudaron a mitigar "Desconocimiento del dominio del problema" y "Selección de las librerías para grabación y mezcla de audio y video". Los prototipos 3, 4 y 5 a "Producto poco usable". Cada muestra de un prototipo realizada al cliente ayudó a bajar "Pérdida de interés por parte del cliente". La buena gestión que se logró en el proyecto, hizo que cuando se materializara el riesgo "Baja disponibilidad de alguno de los integrantes del equipo", el equipo supiera cómo organizarse para cumplir con los plazos establecidos.

7 Aseguramiento de la calidad

En esta sección se describen las actividades realizadas para asegurar la calidad del proceso y del producto de software. En el aseguramiento de la calidad del proceso, el énfasis estuvo en la correcta aplicación del ciclo de vida evolutivo y en el enfoque de gestión ágil elegidos. Por otro lado, el aseguramiento de la calidad del producto se baso en garantizar que los prototipos estuvieran en un estado adecuado para realizar las validaciones con los usuarios. A su vez, estas validaciones también fueron un punto clave para asegurar la calidad del producto.

7.1 Del proceso

Para el equipo, asegurar la calidad del proceso significó, por un lado que el ciclo de vida evolutivo se cumpliera. Por otro, que el enfoque de gestión ágil se mantuviera y se tratará de mejorar a lo largo de todo el proyecto.

Para cumplir con el ciclo de vida evolutivo a lo largo de todo el proyecto, se empleo la utilización de un modelo de desarrollo basado en prototipos, la gestión de un alcance variable y la alta participación del cliente. Estas actividades permitieron adaptarse a los cambios en los requerimientos.

Por otro lado, el enfoque de gestión ágil bajo el marco *Scrum* se mantuvo gracias a la realización de las ceremonias y al cumplimiento de los valores ágiles por parte del equipo. En particular, las *retrsopective* permitieron detectar oportunidades de mejora sobre el proceso al finalizar cada iteración. Se puede destacar como mejora en el proceso de gestión, la evolución de la herramientas utilizadas para la planificación y registro de esfuerzo. Esta evolución permitió obtener una mejor precisión en la dedicación por parte del equipo y consecuentemente en la comparación con las estimaciones.

7.2 Del Producto

7.2.1 Código fuente

La calidad de un producto de software depende mucho de la calidad con que este se construye. Por este motivo, el equipo trato de seguir en todo momento las buenas practicas de programación [16].

Por otro lado, se trataron de cumplir los estándares de codificación propuestos por Microsoft [17] para C# y los propuestos por Sun Microsystems [18] para Java. Además, a partir del prototipo 3 se comenzó a realizar *refactoring* de código, ya que desde ese momento se sabia que no se desecharía tanto código, como si ocurrió con el prototipo 1 y 2. Esto permitió contar con un código de mejor calidad, favoreciendo un mejor entendimiento

del mismo por parte del equipo y de los desarrolladores que puedan continuar con la implementación del producto en un futuro.

7.2.2 Prueba de software

Durante el proceso de desarrollo y en particular antes de cada validación, se realizaron pruebas exploratorias sobre los prototipos.

El testing exploratorio se define como el aprendizaje, el diseño de casos de prueba y la ejecución de las pruebas en forma simultánea [19]. Las pruebas exploratorias se basaban en:

- No definir casos de prueba específicos.
- Buscar inestabilidades explorando la aplicación.
- Tratar de probar los escenarios mas frecuentes de uso.
- Si se encuentran defectos, documentar bajo que escenario se producen para luego poder corregirlos.

Como sugiere Bach [19], este tipo de pruebas es útil en situaciones de testeo complejas, como cuando el producto es altamente cambiante y realizar pruebas tradicionales implica un *overhead* importante. Las pruebas exploratorias permitieron asegurar, con un mínimo de planificación, que los prototipos estuvieran razonablemente aptos para realizar una validación con usuarios. Un eventual error en un prototipo podría ocasionar que la experiencia del usuario se viera afectada y por ende, que la validación no se pudiera ejecutar de forma satisfactoria.

Las pruebas exploratorias también se realizaban luego de integrar los desarrollos llevados a cabo por cada miembro del equipo. Todos los fines de semana, el equipo se reunía a desarrollar y al terminar, se realizaban estas pruebas para asegurar que la ultima versión del prototipo en el repositorio estuviera estable.

7.2.3 Validaciones

Durante el transcurso del proyecto, se realizaron tres tipos de validaciones con músicos amateurs y expertos en usabilidad. Las validaciones realizadas fueron las siguientes:

- **Evaluaciones Heurísticas** [20]**:** Con un experto en usabilidad, en este caso Martín Solari, se realizaron dos evaluaciones heurísticas. Una utilizando el prototipo 3 y otra utilizando el prototipo 4. Estas validaciones sirvieron para obtener *feedback* acerca de los siguientes aspectos:
    - Como se mueve el usuario en la interfaz (Navegación).
    - Donde está el usuario y que tiene que hacer (Orientación).
    - Como interactúa el sistema y el usuario (Interacción).
    - Que hay en el sitio y como es comunicado al usuario (Contenido).
    - Como se distribuyen los espacios en la interfaz (Layout).

A partir de estas validaciones, el equipo obtuvo un *feedback* muy valioso. Lo aprendido y posteriormente aplicado en los prototipos significó un importante aumento en la calidad del producto. Donde se puede apreciar mejor esta evolución es en el paso de prototipo 3 al prototipo 4. Esta evolución implicó una serie de cambios estéticos y de usabilidad sobre la interfaz de usuario. En la sección 3.2.4 se detallan algunos de los cambios realizados.

- **Sesiones de observación:** Se realizaron algunas sesiones de observación con músicos que no conocían de que se trataba la producto. Estas sesiones consistían en observar el comportamiento de los usuarios mientras interactuaban con la aplicación y en sacar apuntes sobre las dificultades que se les presentaban. En la Figura 27 se muestra a un músico realizando una grabación sobre el prototipo 3 en una sesión de observación.

Figura 27: Federico Sicardi (músico amateur) improvisando una canción en el prototipo 3

- **Escenarios:** Esta validación consistió en definir escenarios en los cuales los usuarios realizaran tareas concretas, de forma de identificar problemas de usabilidad. Para cada escenario se definía un contexto en el cual el usuario debía realizar una tarea concreta para lograr una determinada meta. En el Anexo 14 se encuentran algunos escenarios que sirvieron para identificar problemas de usabilidad en la aplicación.

  En conjunto con la validación por escenarios se utilizó la técnica *thinking aloud* [21]. Esta técnica consiste en que los usuarios vayan verbalizando lo que piensan, mientras

exploran la aplicación. La utilización de esta técnica permitió descubrir que pensaban los usuarios acerca del diseño e identificar oportunidades de mejora y problemas de usabilidad de forma interactiva. En la Figura 28 se puede observar a un músico realizando una tarea de sincronización de pistas al mismo tiempo que realiza algunos comentarios acerca del proceso.

Figura 28: Joaquín Sosa (músico amateur) sincronizando pistas en el prototipo 5

8 Conclusiones y lecciones aprendidas

8.1.1 Conclusiones generales

Como conclusión con respecto al proyecto se destaca que se han logrado cumplir la mayoría de sus objetivos. Se obtuvo como resultado, un prototipo web, que le permitió a los músicos amateurs crear videos musicales colaborativos sin tener que realizar todos los pasos descritos en la problemática.

Se realizaron muchas instancias de validación en este proyecto. En estas instancias participaron tanto los clientes como músicos amateurs cercanos al equipo. Todas estas validaciones permitieron mejorar aspectos tecnológicos y de usabilidad en el producto.

Por otro lado, se logró cumplir con todas las expectativas de los clientes. El prototipo presenta una estética adecuada como para realizar una muestra ante posibles inversores. Además, los principales conceptos del problema se lograron representar de forma sencilla en el producto. Esto ayudó a que la idea propuesta por Mixit! se entendiese fácilmente. Finalmente, se logró probar que crear videos musicales colaborativos de forma distribuida es posible si se hace uso de las tecnologías adecuadas.

Con respecto a la metodología aplicada durante el proyecto, se concluye que el enfoque de gestión ágil utilizado y el ciclo de vida evolutivo fueron fundamentales para el éxito del mismo. El enfoque de gestión ágil le permitió al equipo planificar de forma flexible, asegurando la capacidad de agregar o quitar requerimientos sobre la marcha sin impactar de forma significativa en los tiempos. Lo anterior, en conjunto con el alto grado de colaboración de los clientes aseguro la calidad del producto a lo largo de todos los prototipos. Por otro lado, la elección de un ciclo de vida evolutivo permitió ir bajando la incertidumbre sobre los requerimientos. El equipo no se ató a ninguna solución en particular sino que fue evolucionando el producto. En algunos casos esto implico desechar funcionalidades.

La estrategia de definir una cota mínima aceptable de funcionalidad para el producto resultó adecuada para gestionar el alcance. El hecho de manejar el alcance como variable a partir de esta cota permitió finalizar el proyecto con un producto que satisface las expectativas del cliente (por cumplir con lo aceptable) y a su vez posee el máximo de valor agregado posible dentro del tiempo disponible.

En general podemos concluir que el proyecto ha sido una buena instancia de aprendizaje. El equipo ha tenido la oportunidad de experimentar una situación de trabajo real, con una problemática compleja y en un contexto bastante particular. El buen clima dentro del equipo y la constante interacción con el cliente han sido factores claves para enfrentar los desafíos que se han presentado a lo largo del proyecto. Como próximos pasos creemos que es ideal realizar una validación del producto final más abierta, con una mayor cantidad de usuarios.

8.1.2 Lecciones aprendidas

Mantener una buena relación con el cliente

Desde el comienzo del proyecto el equipo se reunió frecuentemente con los clientes, tanto para relevar requerimientos como para validar prototipos (muchas veces ambas actividades). Gran parte del *feedback* obtenido para mejorar el producto partió de los mismos. Además fueron ellos los referentes en cuanto a los conocimientos sobre música. Si la relación con el cliente no hubiera sido buena es muy probable que los objetivos del proyecto no se hubieran cumplido.

Realizar pruebas tempranas en el ambiente en donde se piensa instalar el producto

Se destaca la importancia de realizar pruebas tempranas de las tecnologías a utilizar en los ambientes y contextos en los cuales se piensa desplegar la aplicación. El equipo experimentó varios inconvenientes al probar la funcionalidad de grabación en la instancia de Amazon Web Services. Se invirtió mucho esfuerzo para tratar de solucionar estos problemas que no habían sido identificados previamente. Si bien se pudieron realizar algunas pruebas con la aplicación desplegada en la instancia, estas no llegaron a ser completas ya que la funcionalidad de grabación se comportaba de forma impredecible. Este fue un riesgo que no se tuvo en cuenta al principio y que el equipo lo detecto demasiado tarde.

No asumir fechas de entrega, calidad y formato de entregables por parte de terceros que no están directamente involucrados al proyecto

Al comienzo del proyecto el cliente prometió la integración de un diseñador al equipo. Esta persona comenzaría a interactuar con el grupo en el mes de febrero. El equipo había planificado utilizar los dos primeros prototipos para probar tecnologías, por lo cual en un principio la estética no era lo prioritario. Sin embargo, el diseñador se hizo presente recién en el mes de junio. En ese momento ya se estaba finalizando el desarrollo del cuarto prototipo, en el cual ya se había invertido mucho esfuerzo para mejorar la estética de la aplicación. Hubo que realizar un rediseño completo para cumplir con el atributo de calidad definido como "Mostrabilidad". Este esfuerzo no previsto termino penalizando otras actividades del proyecto como por ejemplo la búsqueda de soluciones a los problemas de grabación en AWS. Se concluye que asumir fechas en cuanto a la incorporación del diseñador fue un error. Además, tampoco se debieron asumir el formato y la calidad de los entregables proporcionados por el mismo. El diseñador finalmente le entregó al equipo únicamente un conjunto de imágenes, que si bien sirvieron para integrarlas con en el diseño distaban mucho de lo que se esperaba.

9 Bibliografía

[1] Laboratorio ORT Software Factory. (2014, Agosto) [Online]. http://fi.ort.edu.uy/ortsf

[2] Ries E. (2009, Marzo) Startup Lessons Learned. [Online]. http://www.startuplessonslearned.com/2009/03/minimum-viable-product.html

[3] (2013, Noviembre) HistoryOfRecording.com. [Online]. http://www.historyofrecording.com/Les_Paul.html

[4] BalausC. (2010, July) StackOverflow. [Online]. http://stackoverflow.com/questions/3623911/what-are-the-main-disadvantages-of-java-server-faces-2-0

[5] Oracle. (2014) JMF 2.1.1 - Supported Formats. [Online]. http://www.oracle.com/technetwork/java/javase/formats-138492.html

[6] FFmpeg. (2011, May) About. [Online]. https://www.ffmpeg.org/about.html

[7] FFmpeg. (2013, June) Projects. [Online]. https://trac.ffmpeg.org/wiki/Projects

[8] FFmpeg. (2013, June) Documentation. [Online]. https://www.ffmpeg.org/ffmpeg.html

[9] FFmpeg. (2013, June) Create a mosaic out of several input videos. [Online]. https://trac.ffmpeg.org/wiki/Create%20a%20mosaic%20out%20of%20several%20input%20videos

[10] Technogumbo. (2013, September) Technogumbo. [Online]. http://www.technogumbo.com/tutorials/Recording-Combined-Audio-and-Video-Streams-With-Flash-and-Red5-1-0-2-SNAPSHOT-or-FMS/

[11] SourceForge. (2013, February) SourceForge. [Online]. http://sox.sourceforge.net/

[12] (2013, October) academia.edu. [Online]. http://www.academia.edu/5321199/Performance_of_Web_Services_on_Smart_Phone_Platforms

[13] Beck K. et al. (2001) Manifiesto por el Desarrollo Ágil de Software. [Online]. http://agilemanifesto.org/iso/es/

[14] Assembla Inc. (2014) Assembla. [Online]. https://www.assembla.com/home

[15] Esben Krag Hansen. (2014) Scrumwise, Better Scrum. [Online]. https://www.scrumwise.com/scrum/

[16] Christopher Diggins. (2011, July) The Principles of Good Programming. [Online]. http://www.artima.com/weblogs/viewpost.jsp?thread=331531

[17] Microsoft. C# Coding Conventions. [Online]. http://msdn.microsoft.com/en-us/library/ff926074.aspx

[18] Sum Microsystems. (1997, September) Java Code Convensions. [Online]. http://www.oracle.com/technetwork/java/codeconventions-150003.pdf

[19] Bach J. (2003, April) Exploratory Testing Explained. [Online]. http://www.satisfice.com/articles/et-article.pdf

[20] Nielsen J. (1995, January) 10 Usability Heuristics for User Interface Design. [Online]. http://www.nngroup.com/articles/ten-usability-heuristics/

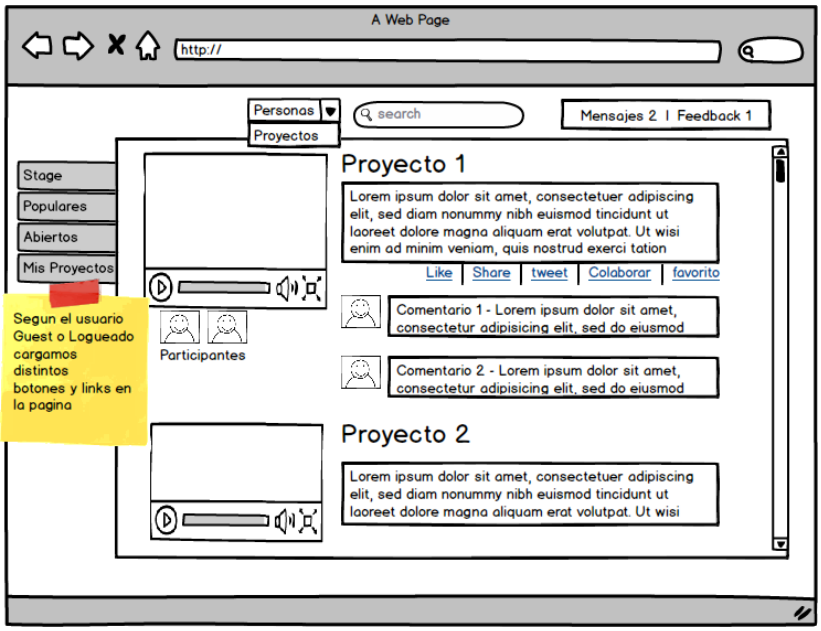
[21] Nielsen J. (2012, Janaury) Thinking Aloud: The #1 Usability Tool. [Online]. http://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/

[22] Apple Inc. (2013) GarageBand. [Online]. https://www.apple.com/mac/garageband/

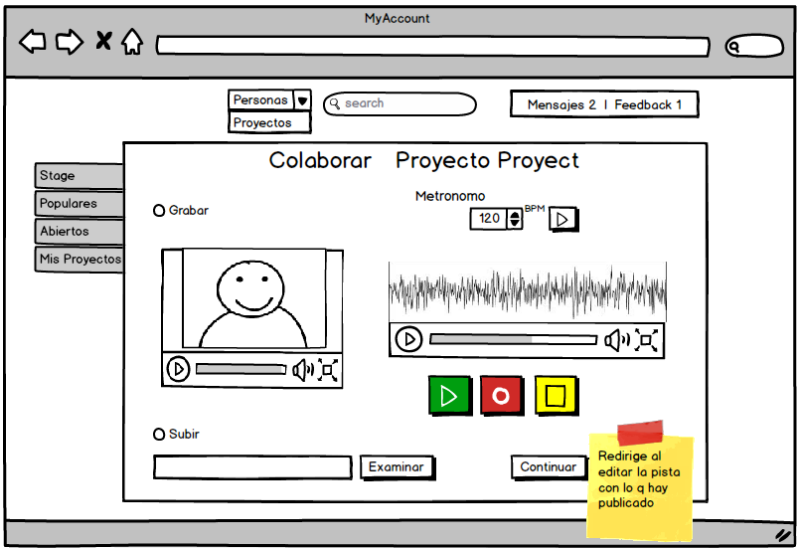
10 Anexos

10.1 Anexo 1 – Bocetos de UI en formato digital

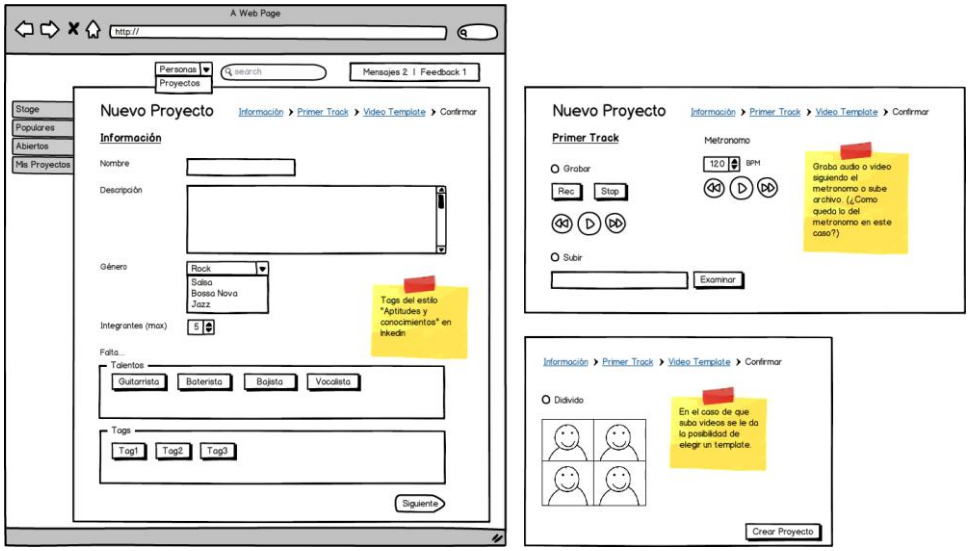
Stage



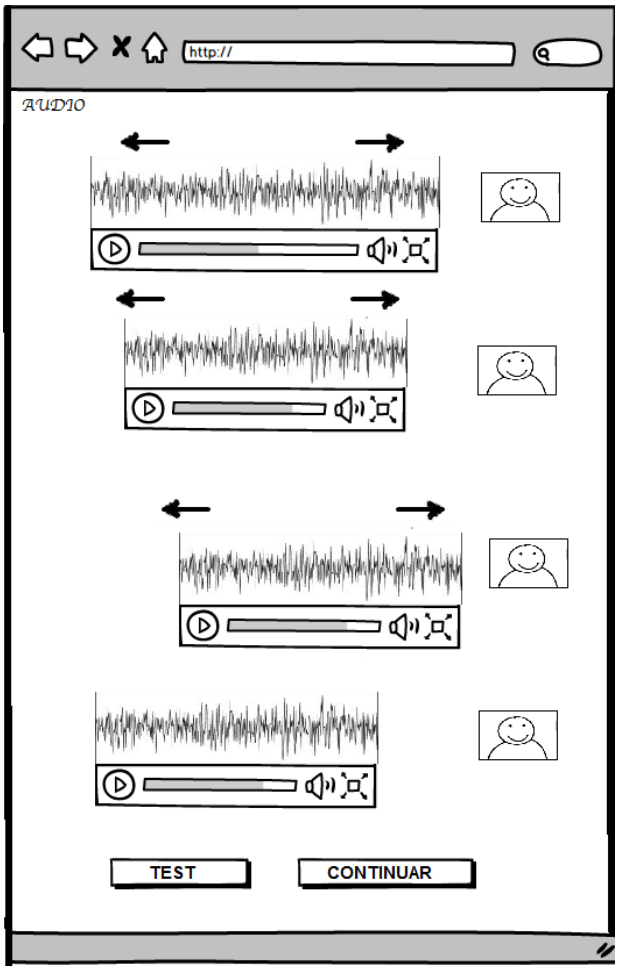
Rec Room



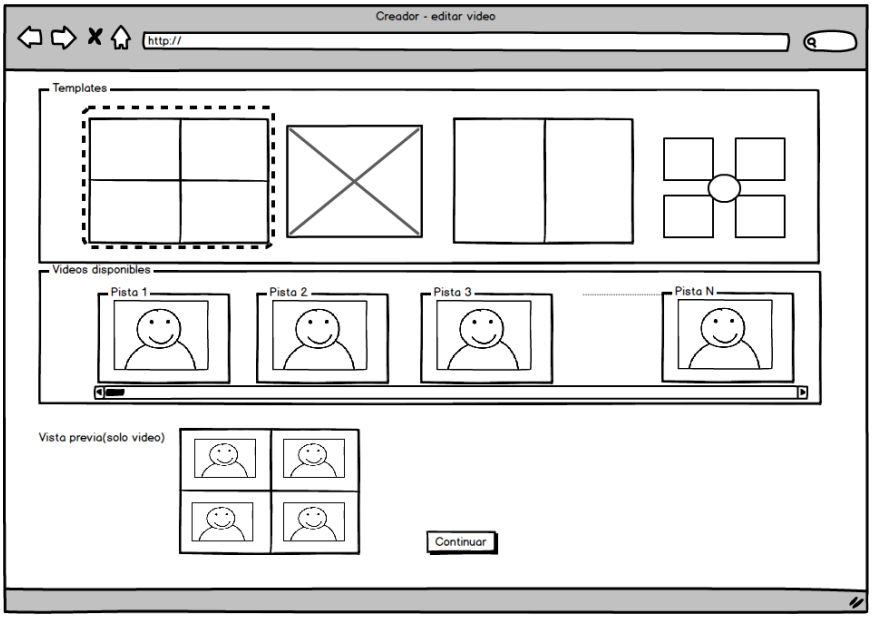
Formulario de creación de un proyecto



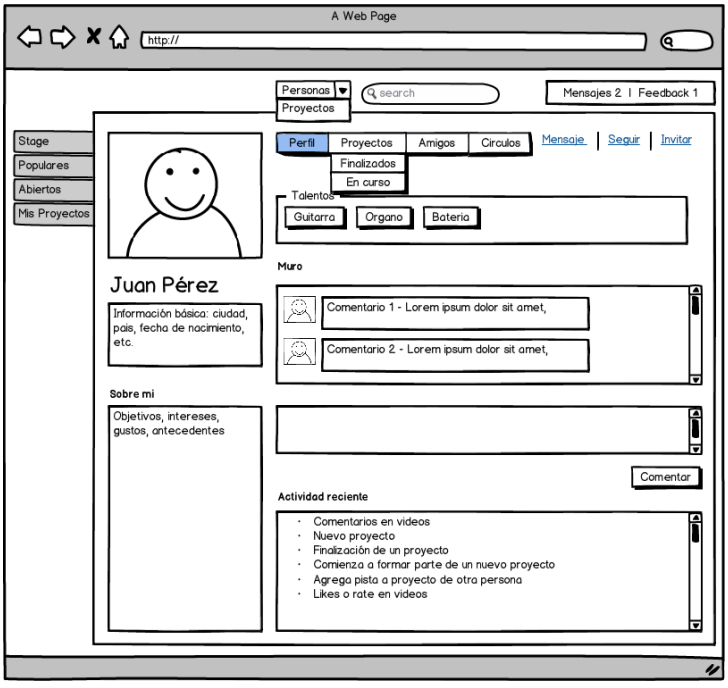
Sincronización de pistas



Selección de *template* de video



Perfil de usuario



10.2 Anexo 2 – Problema al agregar desfasajes con FFmpeg

Durante el desarrollo del prototipo 1 se presentó un problema al tratar de agregar desfasajes a los videos integrantes del mosaico. Se encontró un defecto en FFmpeg[17] que consistía en que cuando se agregaban desfasajes a los videos, solo se aplicaba el desfasaje al primer video de entrada. Para solucionar este problema se decidió separar el procesamiento de los mosaicos en varios videos intermedios y se utilizó una librería adicional para mezclar únicamente el audio. La librería seleccionada fue Sox. La misma, además de ofrecer funcionalidades de manejo de audio, es *open source* y se corre por línea de comando.

10.3 Anexo 3 – Mejoras relevadas de la validación del prototipo 1

En la validación del prototipo 1 se relevaron las siguientes nuevas funcionalidades correspondientes a mejoras sobre el proceso de sincronización:

- Poder hacer zoom en la onda de sonido.
- Poder bloquear una pista una vez sincronizada.
- Poder mover las pistas utilizando las flechas del teclado.
- Poder reproducir las pistas con la tecla *space*.
- Poder intercambiar la posición de las pistas mediante *drag and drop.*
- Poder sincronizar el cursor de todas las pistas con un solo clic.

10.4 Anexo 4 – Mejoras relevadas de la validación del prototipo 2

Lo más importante que se obtuvo sobre la validación del prototipo 2 fue lo siguiente:

- Se tenía que reproducir un compás de entrada del metrónomo antes de empezar a grabar. El motivo era que sino la onda de sonido aparecía cortada al principio y además le daba tiempo al músico para aprontarse antes de que empiece la grabación.
- Se tenía que encontrar un equilibrio entre la calidad de audio y video y el peso de los archivos generados de la grabación. El problema encontrado fue que la calidad de sonido era muy pobre en el video mosaico.
- El creador tenía que tener la opción de combinar dos o más pistas de audio para que el colaborador la escuche como pista base cuando colaboraba.
- Si un colaborador estaba grabando un video y no usaba auriculares para escuchar la pista del creador, se generaba un *delay* en la grabación. Esto es un problema de *hardware* que se identificó. Consistía que cuando se reproduce un sonido y se graba desde la misma computadora se le agrega un *delay* a la grabación. Este problema generaba que cuando el colaborador quería sincronizar la pista de audio con la del creador los picos de las ondas de audio no coincidan.

[17] https://trac.ffmpeg.org/ticket/1349

10.5 Anexo 5 – Elección de un *hosting* servicio en la nube.

Una de las metas planteadas para el prototipo 3 era la de evaluar un servicio que permitiera realizar el despliegue en la nube. El equipo evaluó dos opciones: Windows Azure y Amazon Web Services. De la comparación se obtuvo que Windows Azure era el más barato en cuanto a poder de computo. Sin embargo, se termino eligiendo AWS porque un integrante del equipo contaba con experiencia laboral en el mismo.

10.6 Anexo 6 – Capturas de pantalla del prototipo 3

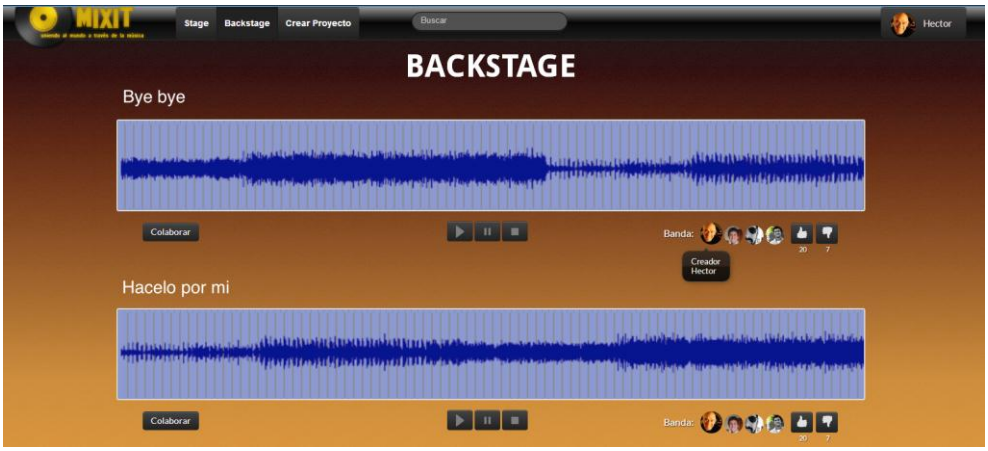
Las siguientes pantallas fueron realizadas con el *framework web* JSF.

Rec Room, donde se daba la opción de subir o grabar un video.

Rec Room, donde se visualizaba la pista de audio del creador y la aplicación Flash que grababa.

Sección en la que se sincronizaban las pistas de un proyecto.

Backstage



Las siguientes pantallas fueron realizadas con el *framework web* ASP.NET MVC.

Rec Room, explicando a través de mensajes los elementos de esta sección.

Finalizando un proyecto, luego de haber sincronizado las pistas de audio.

Rec Room grabando un video.

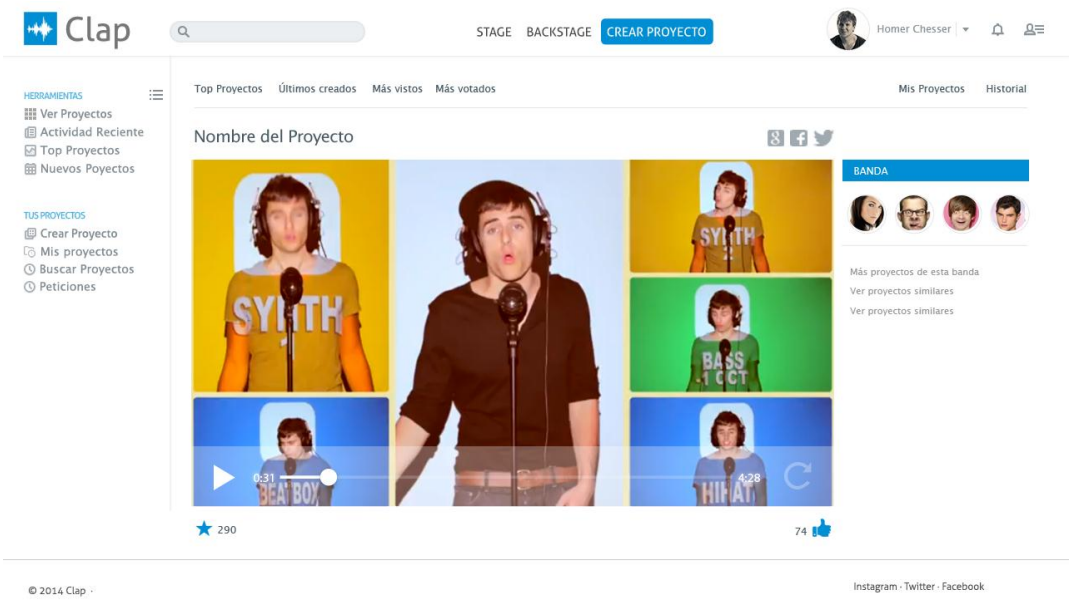
Backstage.

10.7 Anexo 7 – Primeros *mockups* realizados por el diseñador

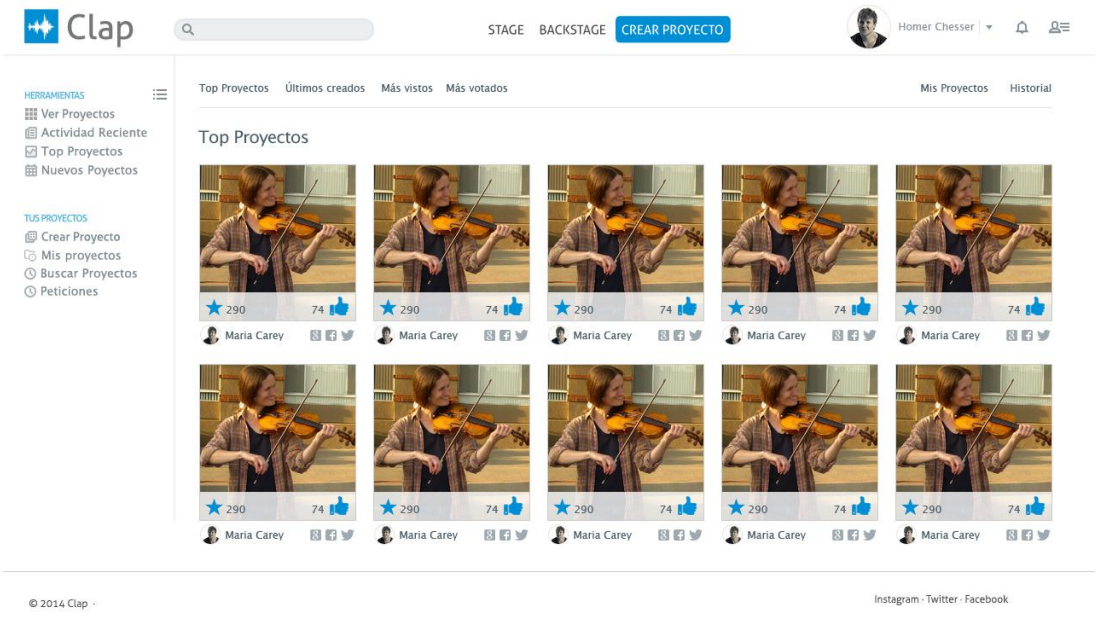
En las siguientes figuras se muestran los primeros *mockups* realizados por el diseñador. Estos corresponden a las principales secciones de la aplicación. Se puede observar el nombre comercial del proyecto que es Clap.

Pagina de inicio.

Stage.



Backstage.

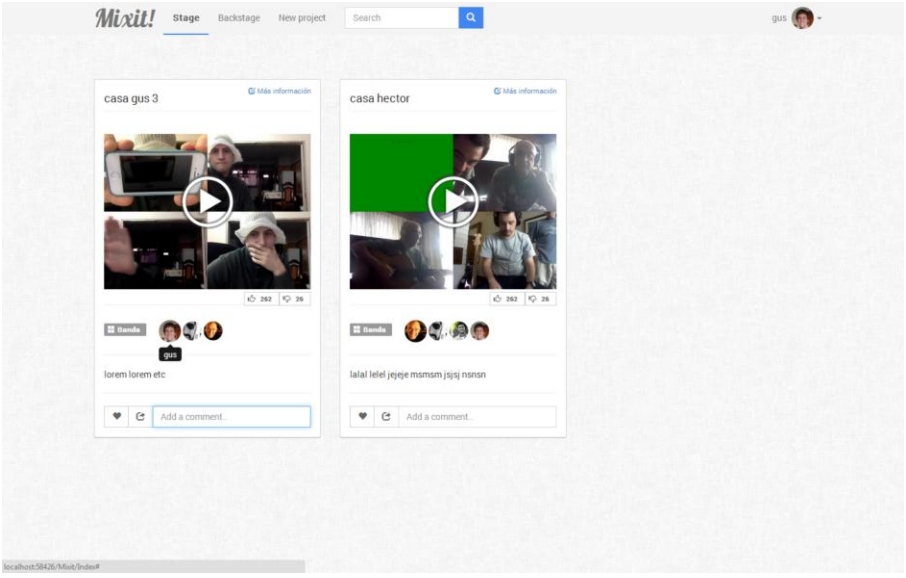

Clap
STAGE
BACKSTAGE
CREAR PROYECTO
Homer Chesser
HERRAMIENTAS
Ver Proyectos
Actividad Reciente
Top Proyectos
Nuevos Poyectos
TUS PROYECTOS
Crear Proyecto
Mis proyectos
Buscar Proyectos
Peticiones
Top Proyectos
Últimos creados
Más vistos
Más votados
Mis Proyectos
Historial
Top Proyectos
290
74
Maria Carey
© 2014 Clap ·
Instagram · Twitter · Facebook

10.8 Anexo 8 – Capturas de pantalla del prototipo 4

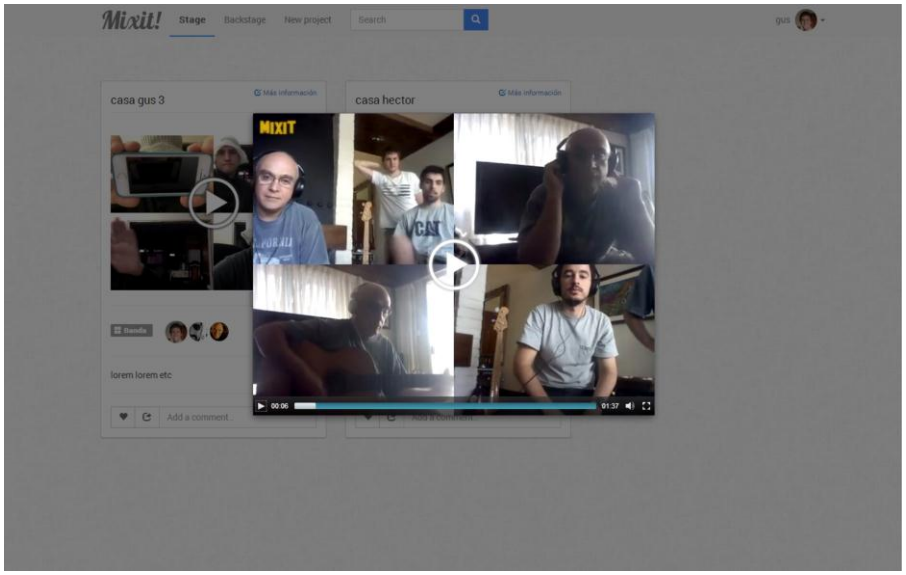
Las siguientes pantallas fueron realizadas con el *framework* Twitter Bootstrap.

Home.

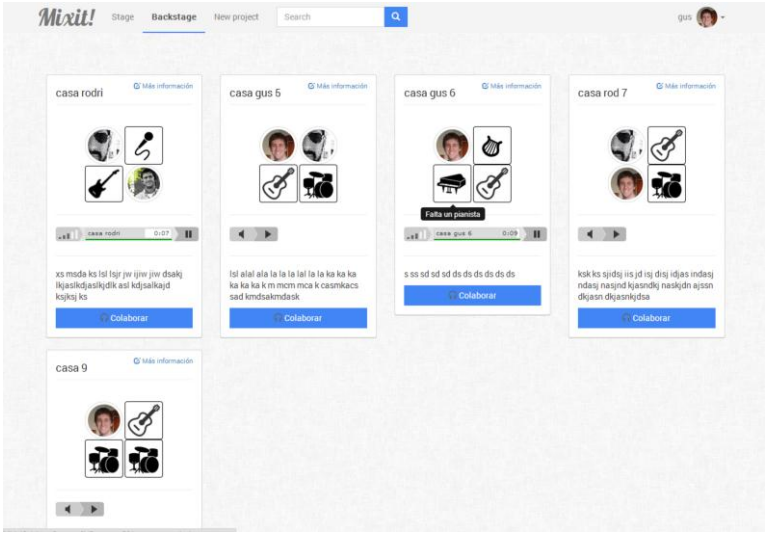
Stage con dos proyectos finalizados.



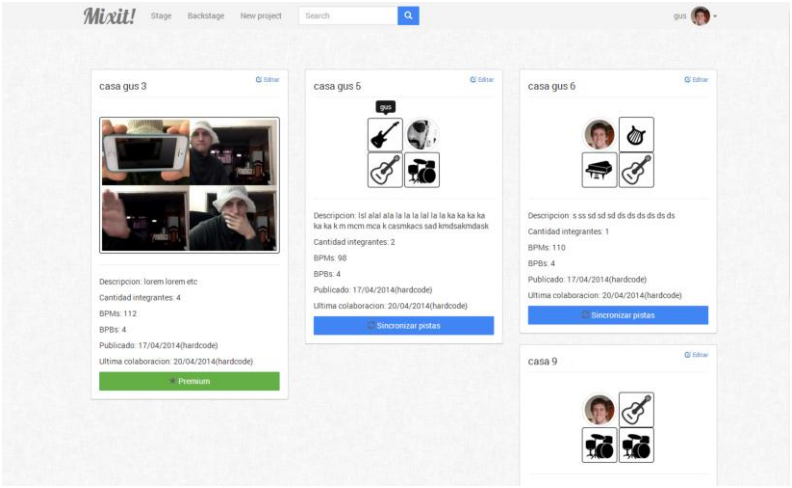
Reproduciendo el video mosaico en el Stage. Se puede observar la marca de agua de Mixit! en la esquina superior izquierda.



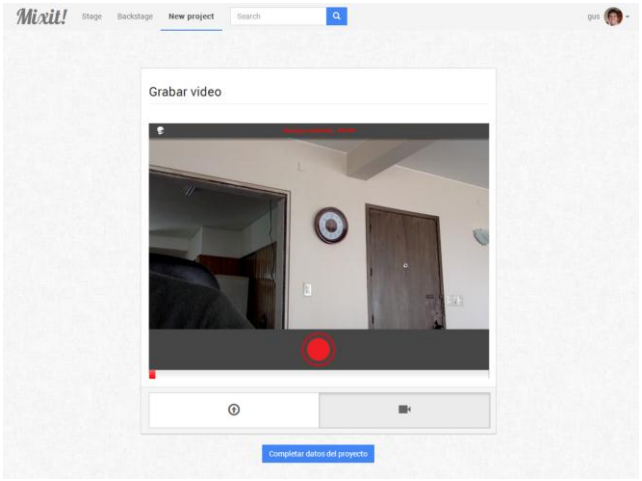
BackStage con algunos proyectos en curso.



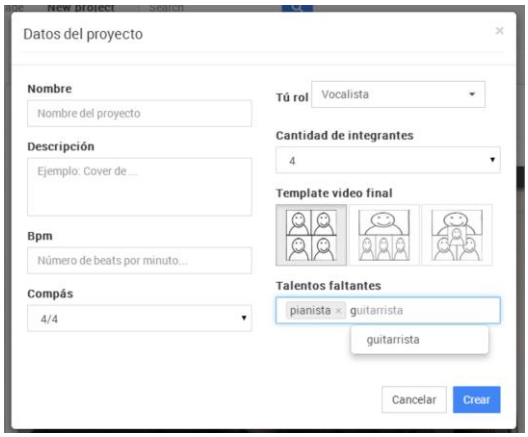
Mostrando los proyectos de un usuario en la sección Mis Proyectos.



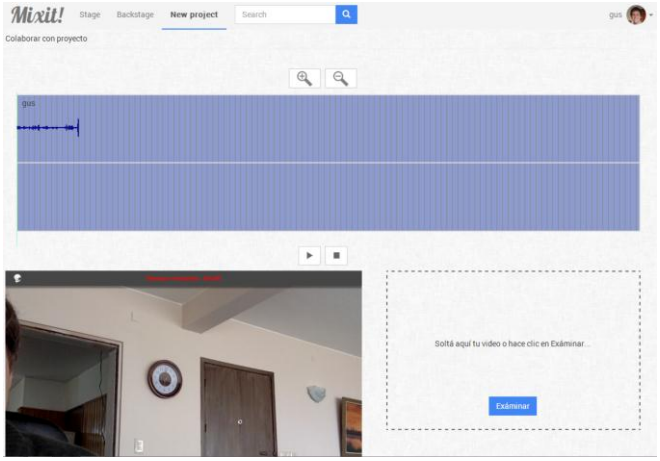
Rec Room, grabando un proyecto.



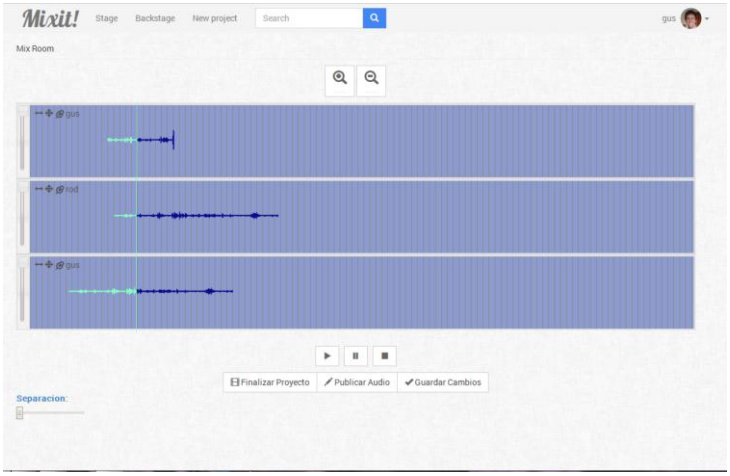
Formulario de creación de un proyecto.



En el Rec Room grabando un proyecto y escuchando la pista de audio del creador.



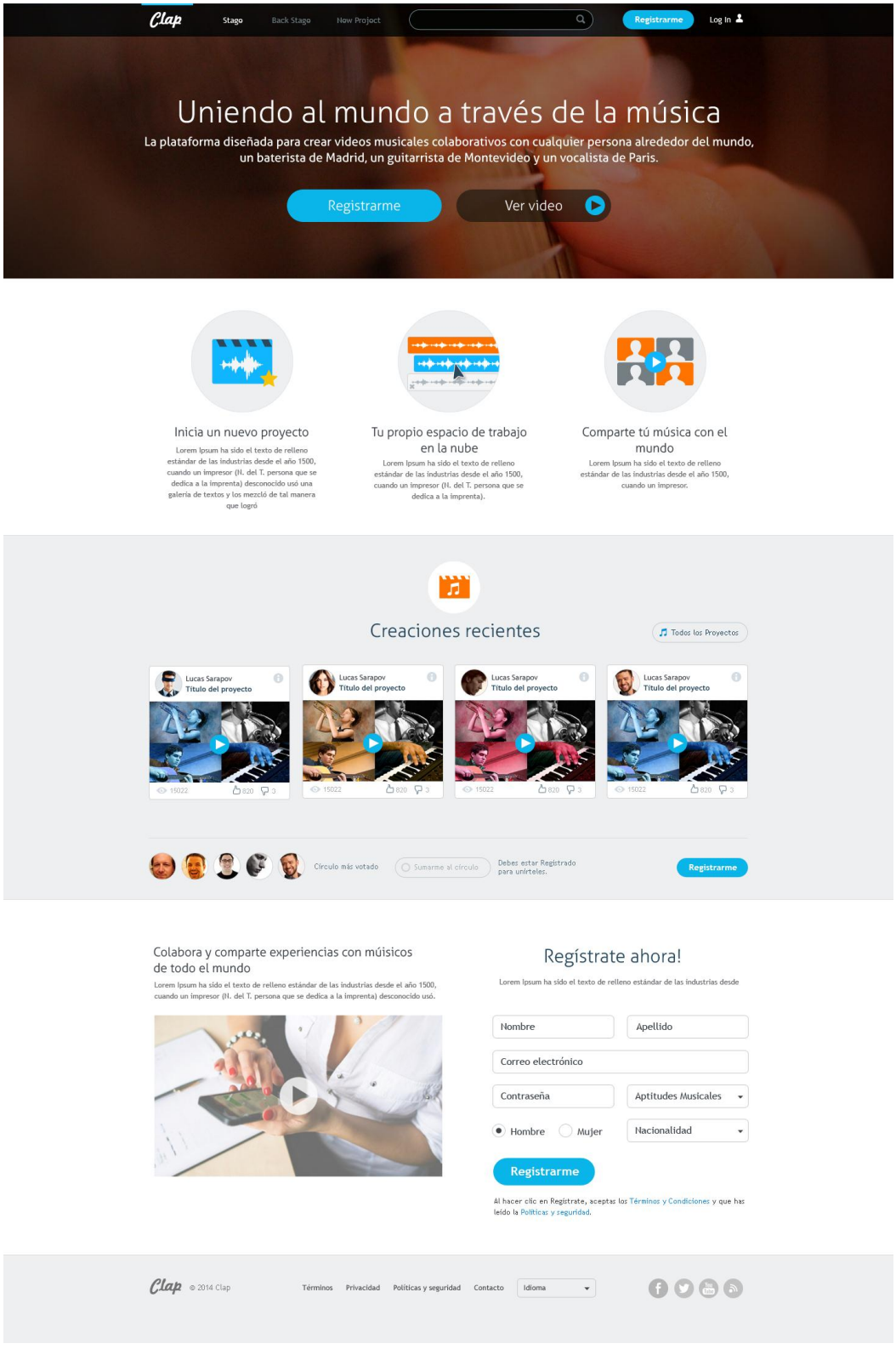
Sincronizando las pistas de varios colaboradores.



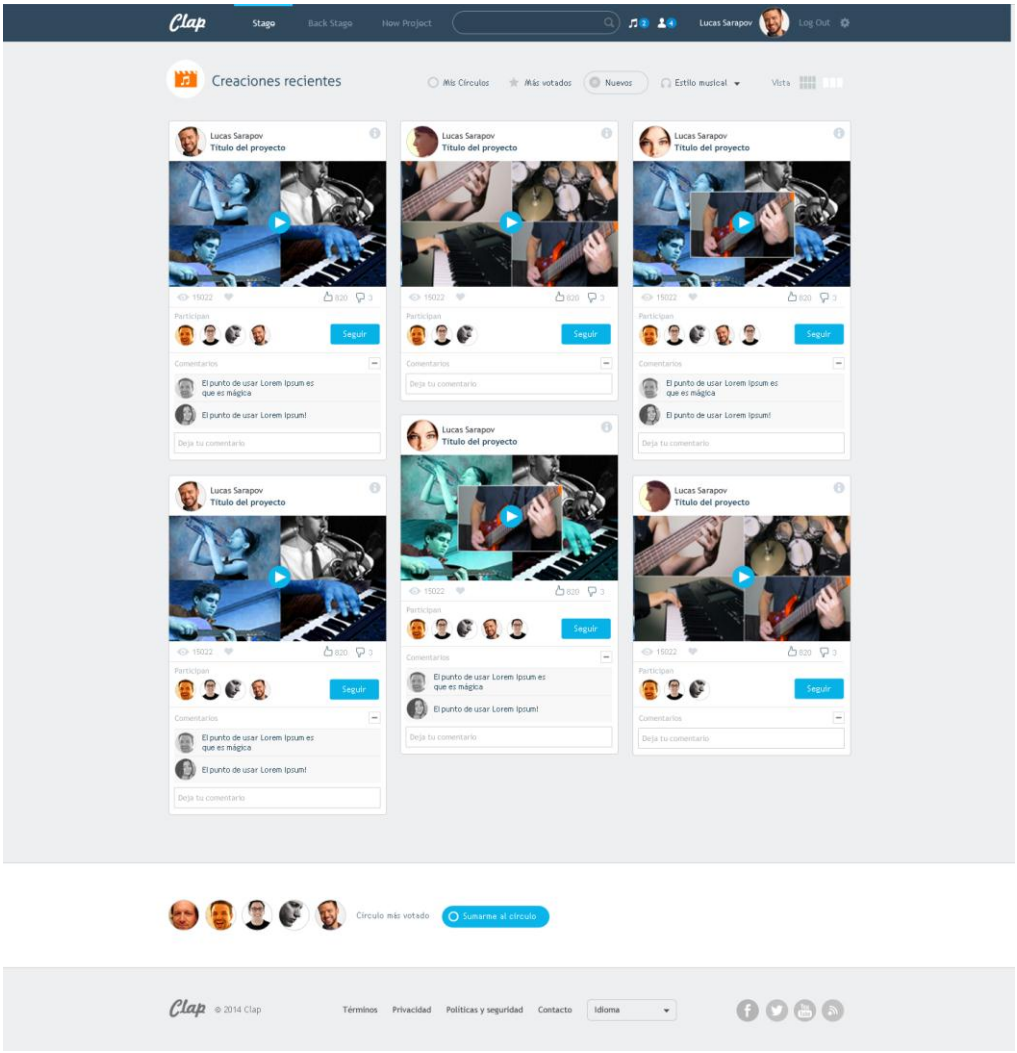
10.9 Anexo 9 – Imágenes del diseñador basadas en prototipo 4

Las siguientes imágenes fueron provistas por el diseñador.

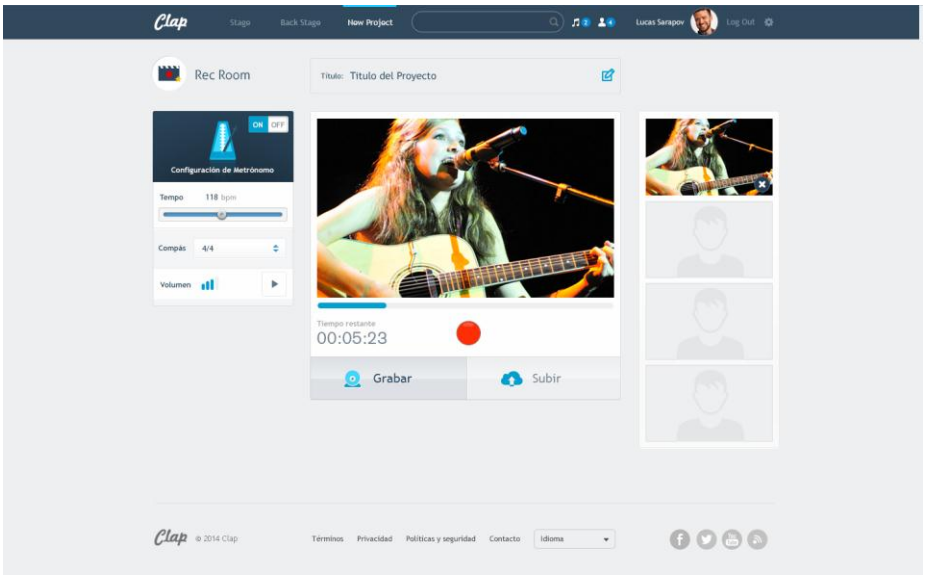
Home o pagina principal.



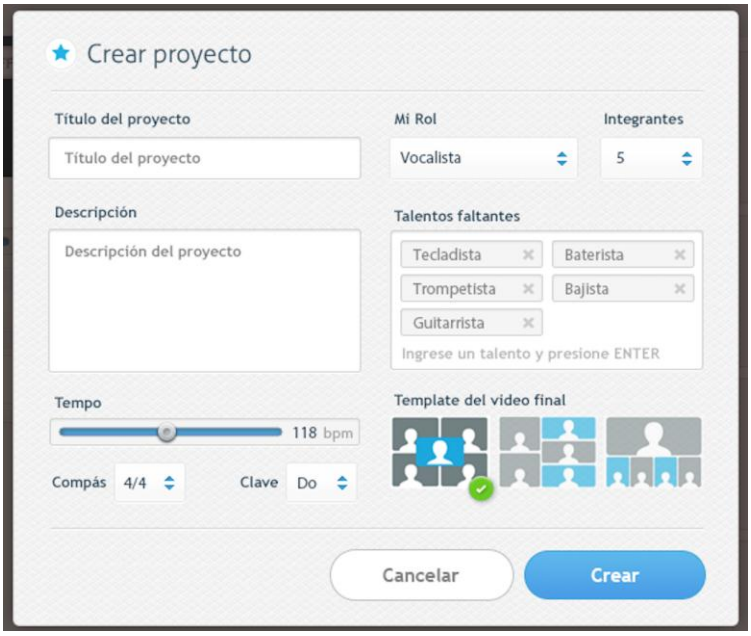
Stage con varios proyectos finalizados.



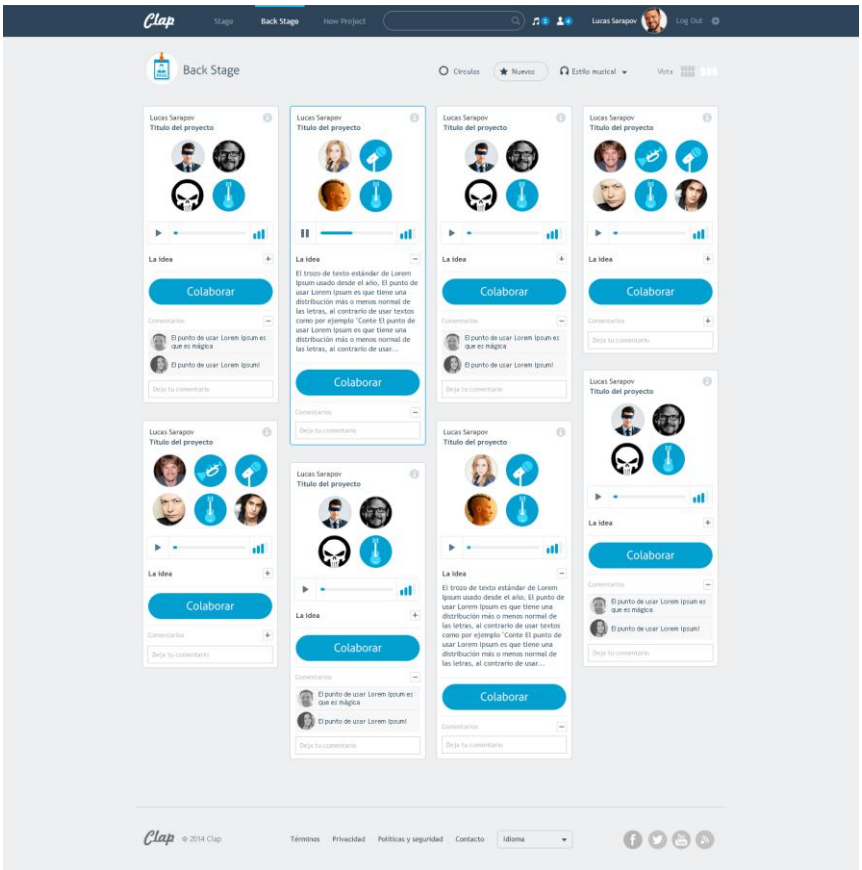
Rec Room.



Formulario de creación de un proyecto.



Backstage, mostrando varios proyectos en curso.



10.10 Anexo 10 – Problemas relevados del prototipo 5

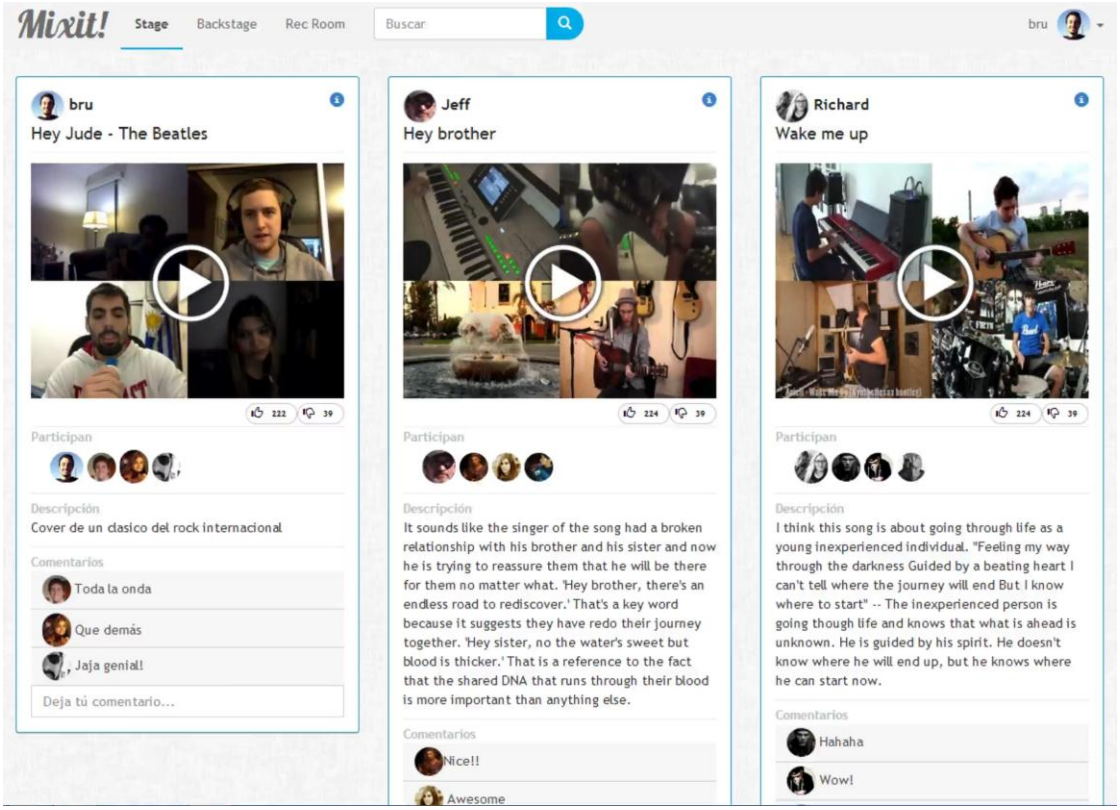
Los problemas que encontró el músico amateur se trataron de textos en la aplicación que no eran los correctos. Por ejemplo, en el Rec Room los botones de “Subir” y “Grabar”. Estos permitían: grabar el video de la página o subir un archivo de video grabado previamente. Lo que el músico interpreto fue que “Grabar” comenzaba la grabación y “Subir” subía la grabación. Otra observación fue en la sección de sincronizar pistas. La misma carecía de la opción que tienen muchas aplicaciones, el “Control + Z”. Que permitía si se equivocaba en la sincronización volver para atrás.

10.11 Anexo 11 – Capturas de pantalla del prototipo 5

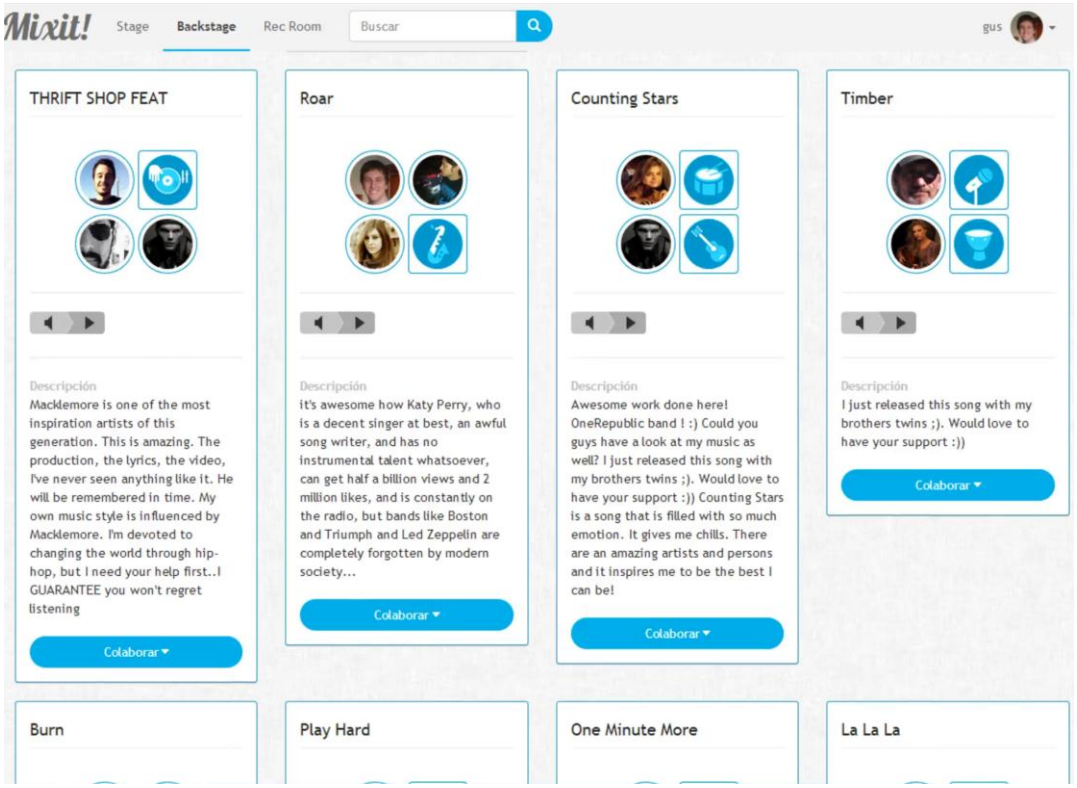
Una vez integradas las imágenes y recomendaciones del diseñador, se obtuvieron las siguientes pantallas.

Home.

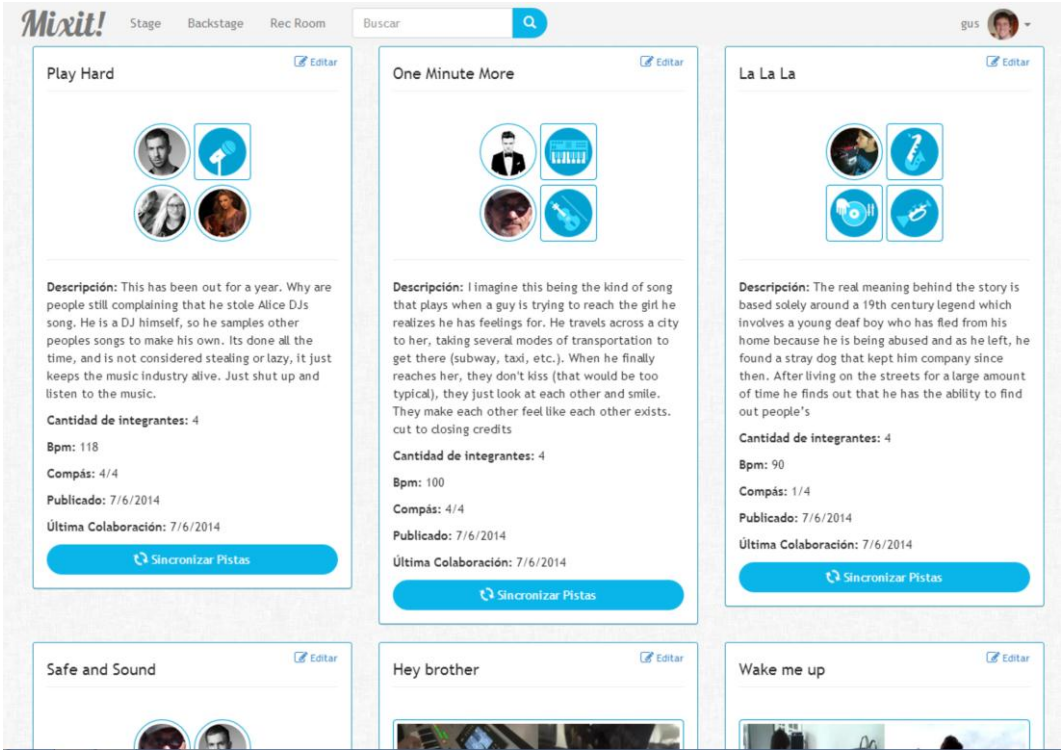
Stage.



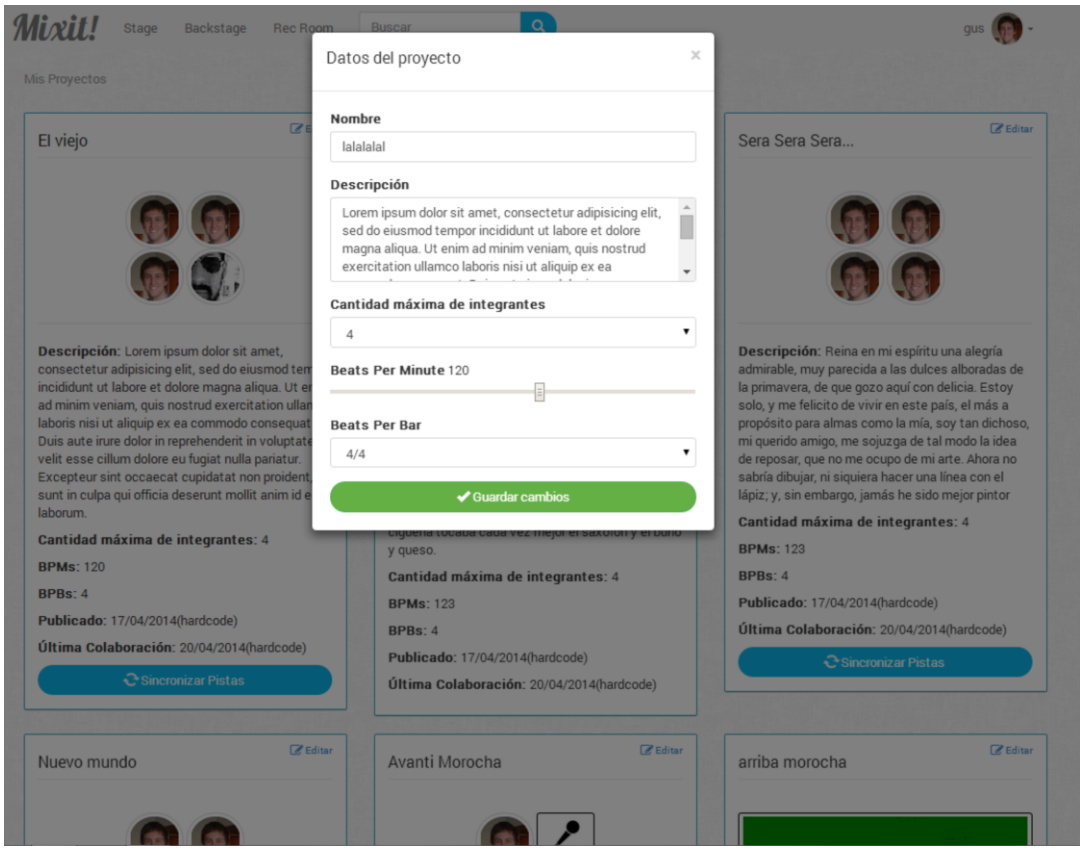
Backstage.



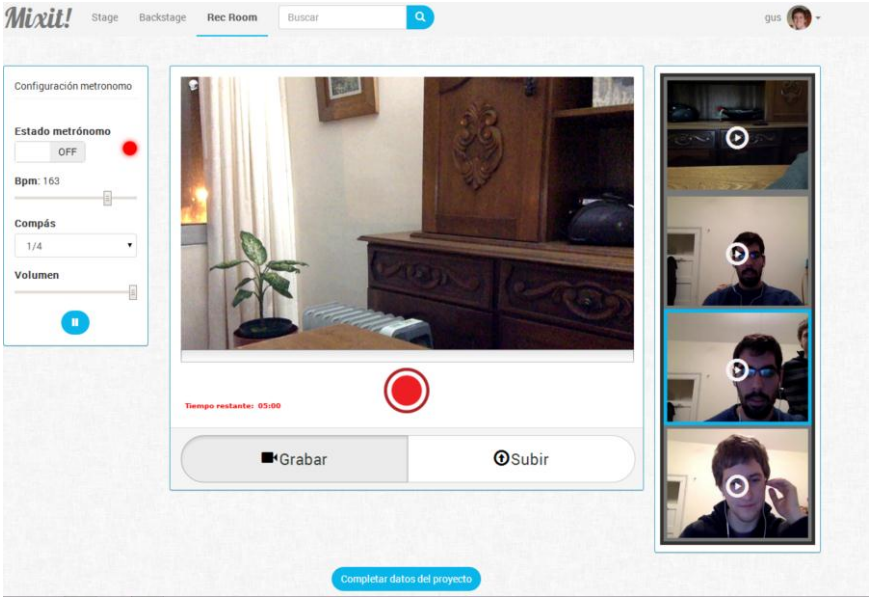
Mis Proyectos.



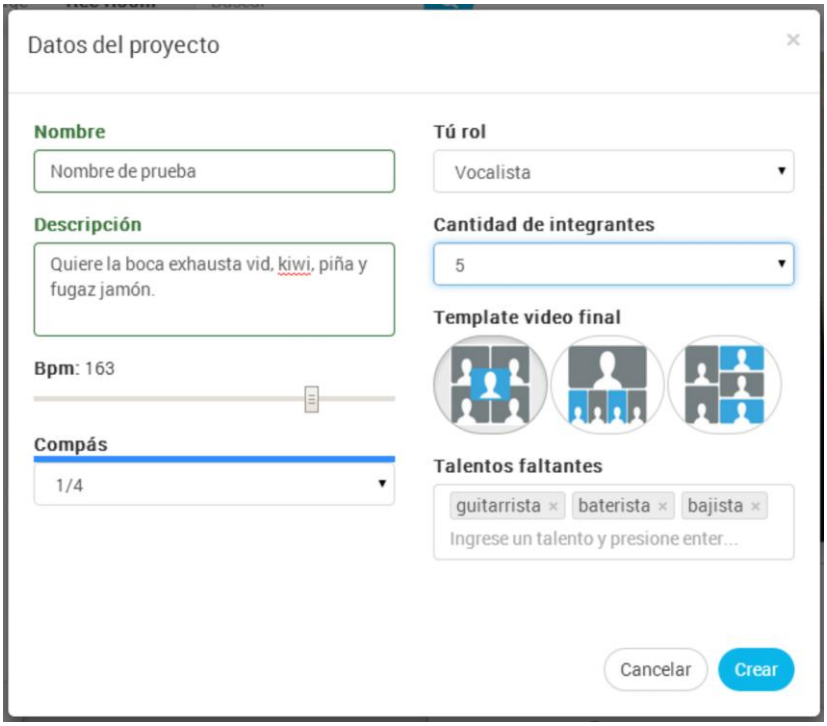
Editando uno de los proyectos creados.



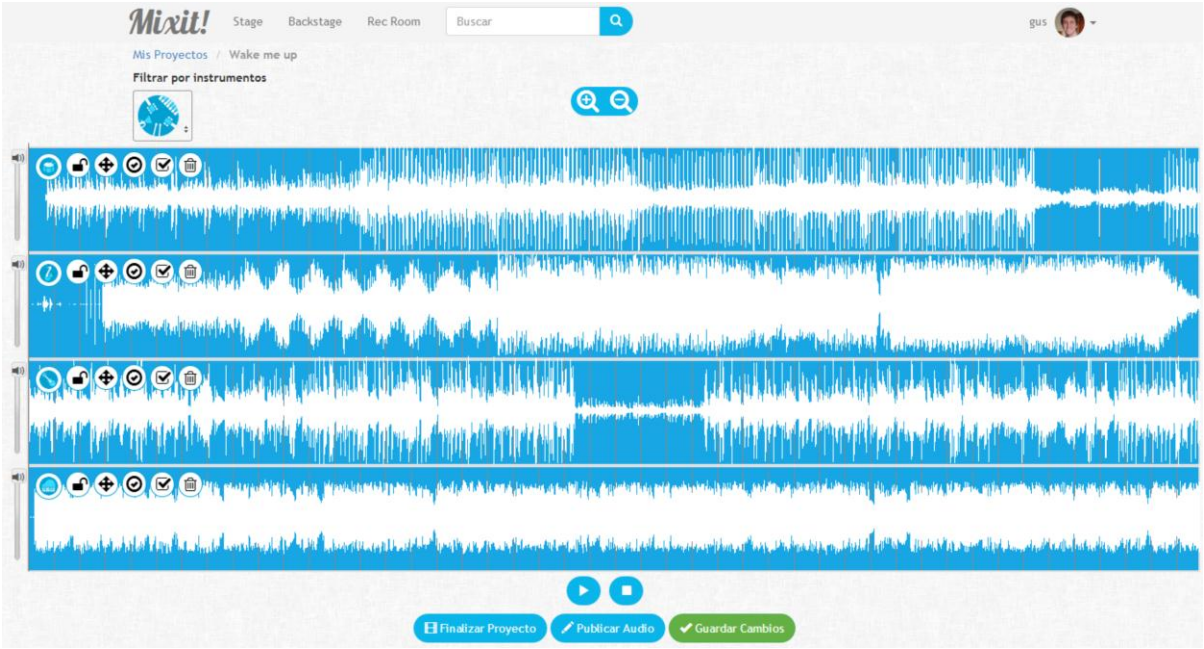
Rec Room.



Formulario de creación de un proyecto.



Sincronizando pistas.



10.12 Anexo 12 – Funcionalidades y sus prioridades

Nro.	Prioridad	Funcionalidades
1	Alta	Como *guest*(usuario no registrado), deseo poder ver las creaciones de los músicos sin tener que estar registrado en el sistema.
2	Baja	Como usuario logueado/no logueado, deseo poder ver los videos más populares por género o estilo, más vistos, ratings, top 10 semanales, tener la posibilidad de filtrar(fecha, género, estilo, país, músicos, etc.), posibilidad de que cuando termine el video arranque otro automáticamente del mismo género y dar "share" en redes sociales con el fin de que me sea más fácil encontrar videos que me gusten y que mis amigos sepan cual música me gusta.
3	Baja	Como usuario logueado , deseo poder dar "me gusta", o "estrellas", comentarios a los videos , ver cuáles fueron los videos subidos por mis amigos, tener información de contacto para el autor del video, con el fin de valorar lo que me gusta y poder contactarme con otros músicos.
4	Baja	Como usuario logueado, deseo tener una página personalizada de la banda en la que estoy, similar a YouTube, con el fin de tener todos los videos centralizados y de fácil acceso a terceros.
5	Media	Como cliente de Mixit!, deseo que los usuarios puedan registrarse utilizando diferentes cuentas (Google, Facebook ,etcétera) con el fin de obtener la mayor cantidad de información del usuario, lo más rápido posible para luego analizarla.
6	Baja	Como colaborador, deseo tener un perfil con toda mi información, con el fin de que otros músicos puedan ver mis colaboraciones, proyectos creados e información personal (instrumentos que toco, descripción

		sobre mí, actividades recientes, email de contacto, etc.).
7	Alta	Como creador, deseo poder dar de alta un proyecto, indicando nombre, descripción, género, integrantes, instrumentos, *tags*, definir un *template* para los videos, definir los BPM del metrónomo, subir mi pista de música o grabarla en el momento, habilitar que otro suba la primer pista , habilitar que otro edite, con el fin de crear un video colaborativo.
8	Alta	Como editor de un proyecto, deseo poder escuchar las pistas y ver los videos con el fin de elegir cuáles se queda y cuáles no en el video final, teniendo la opción de publicarlo en el Stage(finalizar) o dejarlo en el Backstage (*work in progress*).
9	Alta	Como editor de un proyecto, deseo poder sincronizar las pistas (por BPM o milisegundos), con el fin de obtener un video de mayor calidad auditiva.
10	Baja	Como editor profesional, deseo poder descargar los videos para editarlos con un software profesional, con el fin de publicarlo en la página web.
11	Media	Como editor, deseo poder dar *feedback* de las pistas del proyecto con el fin de que los colaboradores mejoren las pistas brindadas.
12	Media	Como editor, deseo poder seleccionar diferentes *templates* de videos, con el fin de personalizar el video a publicar.
13	Alta	Como colaborador, deseo poder ver una descripción de todos los proyectos abiertos que se encuentran en el Backstage y poder filtrarlos según mis características musicales, con el fin de poder colaborar.
14	Baja	Como colaborador, deseo que aparezca en mi perfil y me avise cuando se publicó en el Stage un video en el que participe, con el fin de compartirlo con mis amigos en las redes sociales.
15	Baja	Como colaborador de un proyecto en proceso, deseo que después de cierto plazo tenga la posibilidad de continuar la edición, con el fin de no descartar la pista que había colaborado.
16	Alta	Como colaborador, deseo escuchar la pista base del proyecto para grabar o subir mi pista, con el fin de poder colaborar con el proyecto.
17	Alta	Como colaborador, deseo poder sincronizar mi pista con la pista base, con el fin de que el editor seleccione mi pista.
18	Media	Como colaborador, deseo poder comunicarme con el editor o creador, con el fin de transmitirle mi opinión sobre el proyecto.
19	Baja	Como cliente de Mixit!, deseo que los proyectos no finalizados tengan un tiempo de vida de x meses, con el fin de borrar información innecesaria en el servidor.
20	Baja	Como cliente de Mixit!, deseo que las pistas individuales de los colaboradores sean borradas una vez que termina el proyecto , con el fin de tener la menor información en el servidor.
21	Baja	Como cliente de Mixit!, deseo que la página soporte publicidad patrocinada con el fin de recaudar plata.
22	Baja	Como cliente de Mixit!, deseo que los usuario tengan que pagar para agregar más *tags* en la creación del proyecto
23	Baja	Como cliente de Mixit!, deseo que se puedan ceder los derechos de autor a una discografía, con el fin de retener un porcentaje.
24	Baja	Como cliente de Mixit!, deseo poder vender instrumentos vía acuerdos con proveedores para tener otro canal de ingresos(estilo pedidos ya).
25	Baja	Como cliente de Mixit!, deseo que los usuario tengan funcionalidades

		limitadas como un máximo de 5 minutos para el video, crear un proyecto una vez por mes, 5 colaboradores por proyecto, tener *template* básicos para los videos, calidad del sonido de no más de 128 bits y tener publicidad, con el fin que el usuario se pase a una cuenta Premium.
26	Baja	Como cliente de Mixit!, deseo que los usuario tengan la posibilidad de contratar editores profesionales o músicos profesionales para sus proyectos, con el fin de retener un porcentaje.
27	Baja	Como cliente de Mixit!, deseo poder vender las pistas de audio en ITunes, con el fin de recaudar plata.
28	Baja	Como cliente de Mixit!, deseo tener una versión pro(sin limitaciones) para los usuario que estén suscriptos y pagan por mes, con el fin de recaudar plata.
29	Baja	Como cliente de Mixit!, deseo que los usuario *premium* puedan: tener más de un proyecto abierto y hasta x integrantes, subir más de una pista en proyectos de otros, tener una duración mayor a 5 minutos, tener más *tags*, saber quien entro a ver mi proyecto, tener acceso a listado de editores y utilizar *templates* de video más sofisticados, con el fin de mejorar la satisfacción del usuario.
30	Baja	Como usuario *premium* deseo tener una sección que me muestre información de los pagos que voy realizando, historial , etc.
31	Baja	Como usuario *premium*, deseo tener planes con diferentes opciones de pagos.
32	Alta	Como usuario visitante/registrado quiero poder ver los video finalizados.
33	Alta	Como usuario registrado quiero poder ver la lista de los proyectos de los cuales soy creador de forma tal de poder editar alguno de ellos.
34	Alta	Como usuario registrado (creador o colaborador) quiero poder habilitar o no el metrónomo mientras grabo una pista.
35	Alta	Como accionista del proyecto Mixit! Quiero poder ver en el producto al menos una potencial forma de rentabilización para el mismo.
36	Baja	Como usuario visitante/registrado quiero poder compartir un video en Facebook y/o Twitter.
37	Media	Como usuario colaborador quiero poder visualizar una lista de los proyectos a los cuales estoy postulado.
38	Media	Como usuario registrado quiero poder enviar/recibir mensajes privados a otros usuarios.
39	Media	Como usuario registrado (colaborador) quiero recibir una notificación de si mi pista fue o no aceptada en un proyecto.

10.13 Anexo 13 – Evolución de los riesgos

En las siguientes tablas se muestran los valores de la probabilidad de ocurrencia, el impacto y la magnitud de los riesgos en cada prototipo.

Riesgo1 - Requerimientos poco claros y/o difusos

Riesgo2 - Desconocimiento del dominio del problema

Riesgo3 - Selección de las librerías para grabación y mezcla de audio y video

Riesgo4 - Baja disponibilidad de alguno de los integrantes del equipo

Riesgo5 - Producto poco usable
Riesgo6 - Pérdida de interés por parte del cliente

Prototipo 0

Riesgo	Probabilidad de ocurrencia (0 - 1)	Impacto (1 - 5)	Magnitud (P x I)
Riesgo1	0,8	5	4
Riesgo2	0,6	5	3
Riesgo3	0,2	2	0,4
Riesgo4	0,4	5	2
Riesgo5	0,2	1	0,2
Riesgo6	0,2	5	1

Prototipo 1

Riesgo	Probabilidad de ocurrencia (0 - 1)	Impacto (1 - 5)	Magnitud (P x I)
Riesgo1	0,8	4	3,2
Riesgo2	1	4	4
Riesgo3	1	5	5
Riesgo4	0,4	4	1,6
Riesgo5	0,2	1	0,2
Riesgo6	0,4	5	2

Prototipo 2

Riesgo	Probabilidad de ocurrencia (0 - 1)	Impacto (1 - 5)	Magnitud (P x I)
Riesgo1	0,6	3	1,8
Riesgo2	0,8	4	3,2
Riesgo3	0,8	5	4
Riesgo4	0,6	4	2,4
Riesgo5	0,2	1	0,2
Riesgo6	0,4	5	2

Prototipo 3

Riesgo	Probabilidad de ocurrencia (0 - 1)	Impacto (1 - 5)	Magnitud (P x I)
Riesgo1	0,4	3	1,2
Riesgo2	0,2	4	0,8
Riesgo3	0,2	1	0,2

Riesgo4	0,6	4	2,4
Riesgo5	0,8	3	2,4
Riesgo6	0,6	4	2,4

Prototipo 4

Riesgo	Probabilidad de ocurrencia (0 - 1)	Impacto (1 - 5)	Magnitud (P x I)
Riesgo1	0,2	1	0,2
Riesgo2	0,2	3	0,6
Riesgo3	0,2	1	0,2
Riesgo4	0,2	4	0,8
Riesgo5	0,8	4	3,2
Riesgo6	0,2	4	0,8

Prototipo 5

Riesgo	Probabilidad de ocurrencia (0 - 1)	Impacto (1 - 5)	Magnitud (P x I)
Riesgo1	0,2	1	0,2
Riesgo2	0,2	3	0,2
Riesgo3	0,2	1	0,2
Riesgo4	0,2	5	1
Riesgo5	0,8	4	0,2
Riesgo6	0,2	3	0,4

10.14 Anexo 14 – Escenarios definidos

A continuación se muestran algunos de los escenarios utilizados por los músicos para las pruebas de usabilidad sobre los prototipos.

Escenario 1

- Contexto: sos un músico y queres crear un proyecto, para eso necesitas grabar un video con el fin de que otros músicos colaboren.
- Tarea: grabar un video y publicar la pista para que otros usuarios puedan colaborar.

Escenario 2

- Contexto: sos un músico y queres participar en un proyecto con el fin de aportar tu instrumento.
- Tarea: escuchar pista publicada por el creador y grabar un video (escuchándolo para verificar que quedo bien).

Escenario 3

- Contexto: creaste un proyecto, varios músicos colaboraron y queres darlo por

finalizado con el fin de generar el video mosaico.

- Tarea: escuchar pistas de los músicos colaboradores y sincronizarlas para después crear un video mosaico mediante Mixit!.