FACULTAD DE INGENIERÍA Y CIENCIAS AGROPECUARIAS

DESARROLLO DE UNA APLICACIÓN WEB PARA CREACIÓN DE DIAPOSITIVAS Y PRESENTACIÓN REMOTA A TRAVÉS DE DISPOSITIVOS MÓVILES INTELIGENTES.

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Sistemas de Computación e Informática

Profesora Guía

Ing. Anita Elizabeth Yánez Torres

Autor

Sebastián Puente Reshuán

Año

2016

DECLARACIÓN PROFESOR GUÍA

"Declaro haber dirigido este trabajo a través de reuniones periódicas con el estudiante, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación"

_____

Anita Elizabeth Yánez Torres
Ingeniera en Sistemas
C.I. 1802462216

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.”

_____

Sebastián Puente Reshuán

C.I. 1804068953

AGRADECIMIENTOS

Quiero agradecer profundamente a todos quienes han estado junto a mí en este y muchos otros retos dentro de mi carrera estudiantil.

A mi querida familia, gracias por la comprensión, cariño y apoyo que han sabido brindarme en todo momento.

A mi tutora Anita, gracias por sus consejos oportunos y motivarme para sacar este proyecto adelante.

A mis profesores y compañeros, gracias por recorrer este camino conmigo, los llevo en las más grandes muestras de amistad, enseñanzas y recuerdos que me deja esta etapa de la vida.

DEDICATORIA

Quiero dedicar este trabajo a mis padres y a mi ñaña.

Ustedes me han demostrado que lo más grande que puede existir es el amor y apoyo incondicional de la familia. Gracias por creer en mí y motivarme siempre a perseguir mis sueños.

Me siento muy afortunado al poder dedicarles este trabajo. Lo considero un reflejo de todas las oportunidades, que con mucho esfuerzo, cariño y dedicación, me han brindado a lo largo de la vida.

RESUMEN

Este proyecto consiste en el desarrollo de una aplicación web que permite a los usuarios crear presentaciones de diapositivas desde un navegador web y presentarlas utilizando un control remoto virtual, al cual se puede acceder desde el navegador web de un dispositivo móvil.

El principal beneficio que se desea obtener con este proyecto es brindar a los usuarios la posibilidad de acceder a sus presentaciones en cualquier momento y presentarlas de manera sencilla, utilizando los recursos tecnológicos que ya poseen.

Para construir la aplicación web se utilizaron las siguientes tecnologías: MongoDB como base de datos documental, Express como *framework web*, AngularJS como *framework* de presentación y Node.js como entorno de ejecución de JavaScript.

Para establecer la comunicación entre el control remoto y las presentaciones, se utilizó la librería Socket.io, que permite a los clientes web emitir y reaccionar ante eventos transmitidos en tiempo real. Gracias a esta capacidad, se logró que la presentación cambie de estado según las instrucciones dadas por el control remoto.

La aplicación fue diseñada para que se pueda acceder a cualquiera de las funcionalidades de las presentaciones y de control remoto directamente desde un navegador web, con lo cual ya no es necesario descargar aplicaciones nativas o comprar dispositivos de hardware externo. De esta manera la aplicación ayuda a los usuarios a reducir gastos y gestionar sus presentaciones de forma conveniente.

A través de la elaboración de este proyecto se evidenció que utilizar un mismo lenguaje para escribir el código de todas las capas de la aplicación permite optimizar el tiempo de desarrollo, ya que la sintaxis y el estilo de programación se mantienen consistentes. De igual manera, se apreció el gran impacto que puede tener la comunicación en tiempo real entre clientes web de una aplicación, ya que permite implementar herramientas y procesos innovadores de manera rápida y sencilla.

ABSTRACT

This project intends to explore the development of a web application that allows users to create slideshow presentations from a web browser and present them using a virtual remote control, which can be accessed from a mobile device through a web browser.

The main benefit this project offers is providing users the possibility to access their slideshows at any moment and present them in a simple manner by using the gadgets and technology they already have.
In order to build this web application, the following technologies were used: MongoDB as the documental database, Express as the web framework, AngularJS as the presentation framework and Node.js as the JavaScript runtime.

Socket.io is the library that enables communication between the remote control and the slideshow presentation. Socket.io allows web clients to emit and react to real-time events. By using this library, the application is able to change the slide being presented according to the instructions provided by the remote control.

The project was designed as a web application in order to make all of its functionalities accessible through a web browser, which means there is no need to download native applications or buy external hardware devices. In that way, the project helps users to spend less money and create and present their slideshow presentations in a more convenient way.

Throughout this project's development, it was noted that the use of a same programming language across all the application's layers provides important benefits like spending less time on development, not having to deal with different syntax issues and having a consistent programming style throughout the application. Likewise, it can be noted that real-time communication between clients can have a great impact on web applications, since it enables developers to implement innovative processes and tools in a fast and easy way.

ÍNDICE

1. CAPÍTULO I: INTRODUCCIÓN 1

1.1. Antecedentes 1

1.2. Alcance 2

1.3. Justificación 2

1.4. Objetivos 3

2. CAPÍTULO II: HERRAMIENTAS TECNOLÓGICAS 5

2.1. Desarrollo en el servidor 5

2.1.1. Node.js 5

2.1.2 Express 6

2.1.3. MongoDB 7

2.2. Desarrollo en el cliente 7

2.2.1. AngularJS 8

2.3. Comunicación entre clientes web 11

2.3.1. Socket.IO 11

3. CAPÍTULO III: DESARROLLO DEL APLICATIVO 13

3.1. Análisis y planificación 14

3.1.1. Iteraciones 14

3.1.2. Historias 14

3.1.3. Tareas técnicas 16

3.1.4. Corrección de defectos 17

3.2. Diseño e implementación 18

3.2.1. Visualización de presentaciones 18

3.2.2. Gestión de presentaciones 22
3.2.3. Control remoto de presentaciones 25
3.2.4. Autenticación de usuarios 26
3.3. Pruebas 27
3.3.1. Estructura de las pruebas 28
3.3.2. Ejecución de las pruebas 31
3.4. Procesos, configuraciones y despliegue 32
3.4.1. Control de versiones 32
3.4.2. Repositorio remoto 33
3.4.3. Despliegue de la aplicación 33
4. CONCLUSIONES Y RECOMENDACIONES 35
4.1. Conclusiones 35
4.2. Recomendaciones 36
REFERENCIAS 37
ANEXOS 38

ÍNDICE DE FIGURAS

Figura 1. Comunicación entre clientes web a través de un servidor 11
Figura 2. Diagrama de interacción entre cliente y servidor. 18
Figura 3. Estructura de carpetas y archivos generados por Express 19
Figura 4. Estructura y estilo de la página de presentación de ejemplo 21
Figura 5. Petición inicial de una SPA 22
Figura 6. Procesamiento de peticiones con una SPA 22
Figura 7. Procesamiento completo de una petición con SPA y base de datos 24
Figura 8. Diagrama de comunicación entre control remoto y presentación..... 25
Figura 9. Esquema de validación de credenciales y generación de token. 26
Figura 10. Esquema de petición de recursos protegidos enviando token 27
Figura 11. Bloque before de inicialización en la prueba. 29
Figura 12. Bloque it con las acciones y verificaciones en la prueba. 30
Figura 13. Bloque after con instrucciones de limpieza de la prueba 30
Figura 14. Estructura de archivos de prueba en el proyecto 31
Figura 15. Resultado de ejecución de una prueba de integración..... 32

ÍNDICE DE TABLAS

Tabla 1 Ejemplos de historias de usuario desarrolladas en el proyecto.....15
Tabla 2 Tareas técnicas implementadas en el proyecto 16
Tabla 3 Ejemplo de corrección de defecto implementada en el proyecto 17

CAPÍTULO I: INTRODUCCIÓN

1.1. Antecedentes

La presentación con diapositivas es una herramienta muy utilizada en la actualidad para exponer ideas frente a una audiencia. Existen varias alternativas de programas de presentación privativos o libres, de escritorio o en línea para crear presentaciones con diapositivas. Sin embargo, un problema común que experimentan los presentadores es el no poseer una forma sencilla de controlar el avance o retroceso de sus diapositivas mientras exponen.

Esta situación obliga a muchos presentadores a realizar su exposición cerca a sus ordenadores para cambiar las diapositivas manualmente o solicitar el apoyo de otra persona para realizar esta tarea. Esto resulta molesto e incómodo ya que limita al presentador en el uso del escenario y el nivel de control que tiene sobre el avance de su presentación.

Con el afán de tratar de solucionar este problema, algunas herramientas como Powerpoint, han optado por ofrecer la opción de definir una cantidad de tiempo para cada diapositiva, con lo cual el usuario no tiene la necesidad de pasarlas (Microsoft, 2014).

Esta funcionalidad no es útil para la mayoría de presentaciones, ya que la duración de cada diapositiva suele variar dependiendo de la audiencia y las preguntas que surjan en medio de la presentación. De esta manera, es común que los presentadores que han definido previamente el tiempo para cada diapositiva experimenten un desfase significativo entre la lámina que se está mostrando y el tema del cual están hablando.

Otra forma en que se intenta solventar este problema es comprando dispositivos de hardware externo como controles remotos o descargando aplicaciones nativas para ciertos dispositivos móviles. Estas alternativas pueden generar costos adicionales para los presentadores y problemas de incompatibilidad con ciertas versiones de sistemas operativos.

Todas las situaciones mencionadas anteriormente evidencian la necesidad de crear una solución más sencilla e innovadora para permitir a los presentadores adelantar y retroceder sus diapositivas mientras exponen.

1.2. Alcance

El alcance de este proyecto de titulación es desarrollar un prototipo de un sistema web para la creación y almacenamiento de presentaciones con diapositivas desde el navegador web de una computadora y permitir controlar el avance o retroceso de la presentación desde el navegador web de un dispositivo móvil. La elaboración de las diapositivas requerirá de poco esfuerzo por parte del usuario y contará con un formato pre establecido que le permitirá exponer sus ideas de forma simple y ordenada.

El sistema permitirá al usuario crear una nueva presentación, agregar diapositivas e ingresar y modificar campos con los contenidos que se desplegarán en cada diapositiva. El usuario tendrá la posibilidad de visualizar su presentación en el navegador web de una computadora y controlar el paso de las diapositivas desde cualquier dispositivo móvil que tenga un navegador web con soporte para *Web Sockets*.

El diseño de la interfaz de usuario para el control a través del dispositivo móvil será adaptativo, lo cual permite que los contenidos y estilos del sitio web se adapten al dispositivo del usuario, ya sea este de baja resolución como un móvil o de mayor resolución como tablets, dispositivos portátiles inteligentes, etc (Rubio, 2013).

1.3. Justificación

El rápido crecimiento en el acceso y disponibilidad del internet, así como la adquisición masiva de dispositivos móviles inteligentes evidencian la necesidad de adaptar los servicios y aplicaciones de software para que sean compatibles con estas nuevas tecnologías.

El sistema propuesto permitirá aprovechar de mejor manera estos recursos tecnológicos con que ya cuentan la mayoría de presentadores, brindándoles una forma fácil y conveniente de realizar sus presentaciones y controlarlas mientras exponen.

El proyecto ofrece una solución superior en varios aspectos frente a otras herramientas actuales:

Al tratarse de un sistema web, no se necesita de instalación y los usuarios tienen la posibilidad de acceder a los datos de sus presentaciones desde cualquier lugar en que cuenten con conexión a internet. Esto es una ventaja frente a programas de presentación de escritorio, que necesitan ser instalados y que generalmente almacenan las presentaciones localmente en la computadora del usuario, con lo cual la información tiene que ser transportada manualmente.

El permitir el avance y retroceso de las diapositivas desde el navegador de un móvil es otra ventaja frente a varios programas de presentación que no permiten controlar la presentación desde dispositivos móviles o que sólo proveen aplicaciones nativas para sistemas operativos móviles específicos (iOS y Android), como es el caso del servicio en línea Prezi.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar una aplicación web para creación de presentaciones y una aplicación web móvil para el control remoto de la misma, con el uso de tecnologías web para el desarrollo y la comunicación de los aplicativos, para permitir al usuario disponer de sus presentaciones en cualquier momento y presentarlas con facilidad.

1.4.2. Objetivos específicos

- Crear un aplicativo web con formularios básicos para la creación y almacenamiento de presentaciones, de manera que los usuarios puedan elaborar sus diapositivas de una forma estructurada y sencilla.

- Desarrollar una aplicación web para móviles adaptativa con una interfaz visual de fácil manejo que contenga los botones de avance y retroceso necesarios para manejar la presentación.

- Utilizar tecnología de comunicación como una estrategia que permita a la aplicación web móvil conectarse y manejar la presentación que se está desplegando en el navegador web de la computadora.

CAPÍTULO II: HERRAMIENTAS TECNOLÓGICAS

Para la elaboración de este proyecto se consideraron varias tecnologías y se seleccionaron aquellas que brindan mayor facilidad para el desarrollo de las funcionalidades requeridas.

A continuación se detalla de manera breve una explicación de cada herramienta y algunas razones de por qué se decidió utilizarlas en el desarrollo de la aplicación. Se las clasificará de manera general en herramientas utilizadas para: desarrollo en el servidor, desarrollo en cliente y comunicación entre clientes web.

2.1. Desarrollo en el servidor

La aplicación contará con algunas partes críticas en su lógica, que deberán ser implementadas en el lado del servidor ya que tienen que ser ejecutadas de forma centralizada y restringida para los usuarios. Las principales funcionalidades que se requiere implementar bajo este esquema son: autenticar usuarios, exponer recursos web, procesar peticiones, validar datos y almacenar información de manera persistente. En base a estos criterios, se seleccionaron las siguientes herramientas:

2.1.1. Node.js

Node.js es un entorno de ejecución para JavaScript, construido sobre el intérprete de JavaScript V8 de Chrome (Node.js Foundation, s.f.).

En el desarrollo de aplicaciones web, este entorno de ejecución puede ser utilizado para ejecutar lógica del lado del servidor. Es así que al utilizar Node.js, se puede escribir en JavaScript la lógica para servir páginas, procesar peticiones y exponer recursos de una aplicación.

JavaScript ya es utilizado en gran cantidad de aplicaciones web para proveer animaciones, validaciones y manipulación de datos en la capa de presentación. Sin embargo, Node.js ofrece la posibilidad de también utilizar JavaScript para escribir código que será ejecutado fuera del navegador, permitiendo así ocupar un mismo lenguaje a través de todas las capas de la aplicación.

Como desarrollador, una de las ventajas de escribir JavaScript tanto en el lado del servidor como en el lado del cliente es que se elimina la necesidad de cambiar de lenguaje o contexto cada vez que se trabaja en una capa distinta de la aplicación. En general, se puede decir que resulta más fácil y menos confuso mantener una sola sintaxis y estilo de programación a lo largo de todas las partes que componen el sistema.

2.1.2 Express

Express se define como “Infraestructura web rápida, minimalista y flexible para Node.js” (StrongLoop, s.f.).

Hann (2016) señala que Express es un *framework* que actúa como una ligera capa sobre el servidor web de Node.js, haciendo más agradable el desarrollo de aplicaciones web en Node.js.

Esta apreciación se evidencia en algunos beneficios que ofrece Express como mejorar la legibilidad del código y reducir la cantidad de líneas necesarias para desarrollar aplicaciones web. Si bien es posible crear toda la funcionalidad de una aplicación utilizando solo Node.js, este proceso sería largo y complicado. Express brinda varias abstracciones que ocultan los detalles de implementación irrelevantes, facilitando y agilizando el desarrollo de la lógica de negocio que es lo más importante dentro de una aplicación.

En el desarrollo de la aplicación se utilizará Express para exponer recursos de tipo *Representational State Transfer* (REST) que permitan a un cliente, a través del protocolo *Hypertext Transfer Protocol* (HTTP), solicitar y enviar información para que sea procesada en el servidor.

2.1.3. MongoDB

MongoDB es una base de datos documental de código abierto, que provee alto rendimiento, alta disponibilidad y escalamiento automático (MongoDB, Inc, s.f.).

Al ser una base de datos documental, MongoDB no exige la implementación de estructuras o relaciones rígidas que generalmente son requeridas en una base de datos relacional. Al tener mayor flexibilidad en la estructura del modelo, se logra almacenar y gestionar la información de los documentos de una manera más sencilla y directa.

Para el caso del presente proyecto, resulta adecuado el uso de una base documental ya que cada presentación puede ser vista como un documento que contiene una cantidad variable de diapositivas, las cuales a su vez contienen una cantidad variable de campos. El implementar este mismo modelo en una base de datos relacional implicaría tener los datos de una sola presentación dispersos alrededor de varias tablas y mantener las relaciones entre ellas. Esto agregaría complejidad al modelo y no permitiría representar el concepto de presentación de forma tan clara.

Otra característica de MongoDB es que guarda la información como JSON binario (BSON). Al utilizar una variación de JSON, resulta fácil leer y guardar objetos en MongoDB desde código JavaScript (Bretz e Ihrig, 2014). Es por lo tanto conveniente utilizar MongoDB, debido a que en la aplicación se utilizará JavaScript para escribir la lógica del lado del servidor.

2.2. Desarrollo en el cliente

La responsabilidad más importante en el desarrollo de una aplicación del lado del cliente es permitir la comunicación con el servidor a través de una interfaz de usuario amigable. Para ello el código del lado del cliente, que se ejecuta en el navegador, debe poder comunicarse con el servidor para transferir datos y presentar los resultados al usuario.

Tanto la comunicación con el servidor, como la presentación de datos pueden ser realizadas utilizando HTML y JavaScript puro. Sin embargo, esto implicaría tener que escribir mucho código y detalles de implementación que suelen ser tediosos e irrelevantes en la mayoría de aplicaciones. Es por ello que existen algunas alternativas de *frameworks* de presentación y librerías que pueden ser utilizadas para este propósito.

Como enfoque inicial para el desarrollo de esta aplicación se consideró utilizar la librería jQuery. jQuery es una librería pequeña, rápida y con muchas funcionalidades, la cual permite recorrer y manipular HTML, manejar eventos, animaciones y llamadas asíncronas al servidor de manera sencilla (The jQuery Foundation, s.f.).

jQuery puede parecer una buena opción, sin embargo todavía requiere que se escriba una gran cantidad de código en JavaScript para manipular las interacciones entre los elementos HTML y el modelo.
Debido a esta razón, se opta por utilizar AngularJS como una solución más completa y sencilla que jQuery para desarrollar la aplicación.

2.2.1. AngularJS

AngularJS es un *framework* estructural del lado del cliente para crear aplicaciones web dinámicas. Permite utilizar HTML y extender la sintaxis de HTML para expresar los componentes de la aplicación de manera concreta y clara. AngularJS cuenta con enlazamiento de datos e inyección de dependencias que facilitan el desarrollo y reducen la cantidad de código que se necesita escribir (Google, s.f.).

El enlazamiento de datos de AngularJS permite una sincronización entre los datos que se imprimen en pantalla y los objetos que representan un modelo. Esto implica que si el modelo cambia, la vista también cambiará para reflejar los contenidos del modelo. Es así que resulta sencillo adaptar una página para

mostrar información dinámica que se ha obtenido como respuesta del servidor. De igual manera, esta característica es útil cuando se presentan formularios con campos variables que deben ser llenados o editados por los usuarios.

AngularJS también cuenta con la funcionalidad de inyección de dependencias, la cual ayuda a crear una solución más modular. Al inyectar dependencias, se puede separar las responsabilidades que cada parte del código está realizando. Esto permite tener archivos y funciones más pequeñas, logrando una mejor legibilidad y mantenibilidad de la base de código.

Si bien AngularJS provee una abstracción útil para desarrollar aplicaciones, esto limita su flexibilidad y por lo tanto existen ciertos tipos de aplicaciones para las cuales no sería una buena elección. AngularJS fue construido pensando en las aplicaciones *Create, Read, Update, Delete* (CRUD) (Google, s.f.).

En este proyecto se construirá una aplicación que centra mucha de su lógica alrededor de la manipulación de presentaciones, diapositivas e ideas mediante un CRUD, que es la creación, listado, edición y eliminación de entidades. Por lo tanto se reafirma que las características de AngularJS se acoplan bastante bien a las necesidades del proyecto.

AngularJS también brinda ciertas características importantes en cuanto a estructura y funcionalidades debido a que las aplicaciones que crea son de tipo *Single Page Application* (SPA). Una SPA es una aplicación que se ejecuta en el lado del cliente y se encarga de procesar toda la lógica relacionada a la generación y manipulación de los elementos que se presentan al usuario a través del navegador. Por lo tanto, cada vez que el usuario realiza una petición al servidor, este no responde con una nueva página sino que únicamente envía la información relacionada al modelo de datos y la SPA se encarga de alterar la vista. Es así que a diferencia de una aplicación web tradicional en donde toda la página se recarga por completo, en una SPA solo ciertas partes de la página cambian o se refrescan.

Los usuarios de la aplicación de presentaciones necesitarán editar y crear contenido constantemente, por lo tanto es deseable tener una SPA para que solo las partes de la página que han cambiado sean alteradas. Con esto se busca reducir las interrupciones en el flujo de trabajo, que generalmente son ocasionadas al refrescar páginas completas para reflejar pequeños cambios.
Implementar una aplicación estilo SPA presenta también otras ventajas. Por un lado se libera la cantidad de carga de procesamiento en el servidor, ya que este no se responsabiliza por procesar la lógica de la vista sino que únicamente se encarga de exponer recursos con información del modelo. Por otra parte, la cantidad de datos que se envía como respuesta al cliente también es menor, lo cual permite que la transferencia sea más rápida.

A pesar de las ventajas mencionadas, una SPA puede también presentar ciertas desventajas frente a una aplicación web tradicional. Uno de los principales problemas de una SPA es la demora que se suele experimentar al momento de cargar el sitio por primera vez en el cliente. Esto se debe a que durante la primera petición, el servidor deberá enviar todo el código de la aplicación al navegador. Este incluye el *framework*, librerías, lógica de presentación y plantillas para todas las páginas que potencialmente pueden ser generadas por la aplicación.

En un página web informativa o blog se debería priorizar el tiempo de carga inicial, ya que los usuarios muchas veces solo entran a una o dos páginas del sitio y generalmente no interactúan mucho, solo leen el contenido y comentan. En este proyecto, por el contrario, los usuarios no van a acceder solo a una página inicial y salir, sino que van a interactuar con el sistema de muchas maneras y probablemente accederán a varias vistas. Por lo tanto se puede tolerar la demora que experimentan las SPA durante la carga inicial del sitio, ya que las próximas peticiones e interacciones serán más rápidas.

2.3. Comunicación entre clientes web

Como parte importante del proyecto, se requiere conectar el control remoto con la presentación que se está proyectando. Para ello es necesario implementar un mecanismo que permita comunicar en tiempo real un mensaje desde un cliente hacia otro. Por lo general una aplicación web es una aplicación desconectada, lo cual quiere decir que el cliente no está permanentemente escuchando al servidor. Sin embargo para este proyecto será necesario agregar la capacidad de escuchar eventos externos en el cliente, a través de una librería basada en el uso de *web sockets*:

2.3.1. Socket.IO

Socket.IO permite comunicación bidireccional basada en eventos en tiempo real (Socket.io, s.f.).

La comunicación en dos direcciones resulta útil para desarrollar funcionalidades que requieren enviar mensajes al cliente web. Esto permite que los clientes web escuchen permanentemente peticiones y reaccionen o cambien su estado en base a ello.

Mediante la comunicación bidireccional descrita anteriormente, se puede tener una interacción en donde un cliente envía un mensaje al servidor y este lo retransmite a otro cliente, como en el siguiente diagrama:

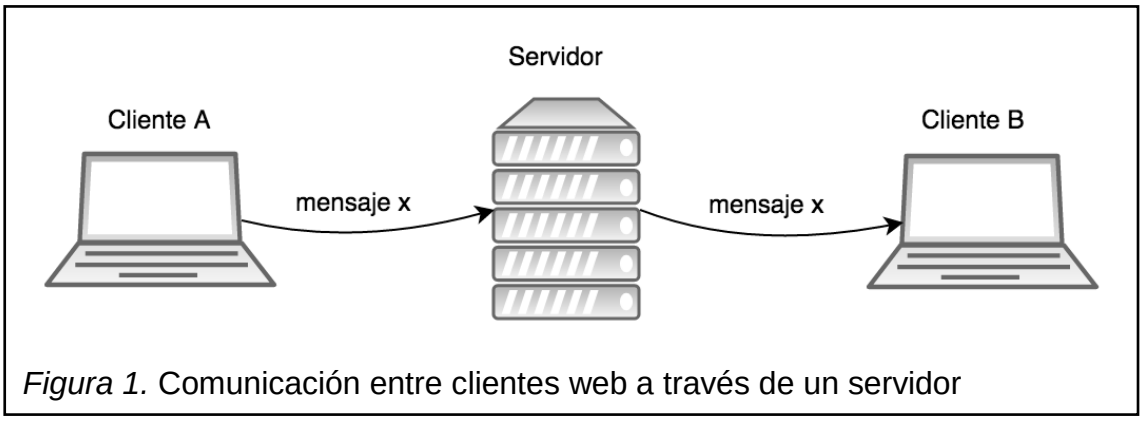


Figura 1. Comunicación entre clientes web a través de un servidor

Este es el escenario que se desea tener entre el cliente que despliega el control remoto y el cliente que muestra las diapositivas. El contacto a través del servidor es importante, ya que este es el punto central de control que debe decidir cómo y a quién retransmitir el mensaje que recibió.

CAPÍTULO III: DESARROLLO DEL APLICATIVO

Para la planificación, análisis y desarrollo del aplicativo se tomaron como base algunos de los valores y prácticas de *Extreme Programming* (XP). Por lo tanto se utilizó un enfoque incremental e iterativo, en el cual las experiencias y descubrimientos obtenidos en cada etapa son incorporados como aprendizaje para la siguiente.

XP se adhiere a cinco valores para guiar el desarrollo: comunicación, simplicidad, retroalimentación, coraje y respeto (Beck, 2004).

En la elaboración del proyecto, se tomó en cuenta el valor de la simplicidad para tratar de escribir el código más sencillo que cumpliera con los objetivos propuestos. Se realizaron además varias modificaciones de manera continua a la estructura del proyecto para simplificar el desarrollo y obtener una base de código más clara, simple y limpia.

La retroalimentación temprana y continua fue otro de los valores que se evidenció a lo largo del proyecto. Las reuniones semanales con la profesora guía y el profesor corrector permitieron ajustar las funcionalidades a medida que se iban desarrollando. Las sesiones de retroalimentación también facilitaron la identificación de defectos de forma temprana.

Los valores de comunicación, coraje y respeto están relacionados de manera más directa con las interacciones entre miembros de un equipo de desarrollo. Por lo tanto estos valores no tuvieron mucho impacto en este proyecto, ya que el equipo de desarrolladores consta de una sola persona.

Además de la guía que proveen los valores de XP, se utilizaron algunas prácticas de XP que se identificaron como relevantes para la elaboración de este proyecto. Entre las prácticas utilizadas están la creación de historias, la planificación en ciclos cortos y el diseño incremental. Estas prácticas permitieron organizar de mejor manera el trabajo y facilitaron el proceso de elaboración del proyecto.

3.1. Análisis y planificación

3.1.1. Iteraciones

En base al plazo con que se dispone para realizar y entregar este proyecto, se decidió dividir el tiempo en cuatro ciclos o iteraciones de tres semanas cada una. Se definieron las iteraciones y los resultados esperados para cada una de ellas, separándolas en base a conjuntos de funcionalidades de la siguiente manera:

- Iteración 1: Visualización de presentaciones
- Iteración 2: Gestión de presentaciones
- Iteración 3: Control remoto de presentaciones
- Iteración 4: Autenticación de usuarios

La planificación y análisis se realizó al comienzo de cada iteración, definiendo un plan de las funcionalidades que se desea implementar para lograr el objetivo de la iteración.

Durante cada iteración se realizaron algunas revisiones con la profesora guía, en las cuales se verificaron los avances en el desarrollo del proyecto para identificar correcciones, mejoras y nuevas funcionalidades que se deberían implementar.

A lo largo de este proceso, se clasificó el trabajo a realizar según sus características en: historia de usuario, tarea técnica o corrección de defecto.

3.1.2. Historias

Para llevar un registro de las nuevas funcionalidades que se desea desarrollar en el sistema, se elaboró un listado de historias enfocadas en el usuario. Cada historia detalla una funcionalidad que los usuarios requieren. Para describir cada historia se utilizó el siguiente formato:

- *Título: <Explicación general de la funcionalidad>*
- *Descripción: <Explicación completa y detalles de la funcionalidad>*
- *Complejidad: <Valor estimado de puntos para la historia>*

Los puntos de complejidad que se utilizan son una estimación realizada por el desarrollador respecto a qué tan complejo será el desarrollo de cada historia. Se optó por utilizar los primeros números en la serie matemática de Fibonacci: 1, 2, 3, 5 y 8 para asignar los puntos. En caso de contar con una historia cuya estimación sea mayor a 8 puntos, se la descompuso en historias más pequeñas de menor complejidad.

Se consideró más adecuado estimar las historias en puntos de complejidad en vez de horas, ya que los puntos no se enfocan tanto en tiempo sino en proveer una idea aproximada y relativa de la complejidad que tienen ciertas funcionalidades con respecto a otras.

A continuación se muestran algunas historias de usuario del proyecto:

Tabla 1. Ejemplos de historias de usuario desarrolladas en el proyecto

#	**Título**	**Descripción**	**Complejidad (Puntos)**
1	Creación de cuenta	Permitir al usuario crear una cuenta con información personal básica para identificarlo en el sistema	3
2	Acceso privado a presentaciones	Cada usuario puede acceder únicamente a las presentaciones que han sido creadas desde su cuenta	2
3	Cerrar sesión	Proveer a los usuarios la opción de salir de su cuenta, evitando que los siguientes usuarios del navegador accedan a su contenido	2
4	Crear nueva presentación	El usuario podrá crear una nueva presentación con título y descripción	2
5	Crear diapositiva con título e ideas	Dentro de una presentación permitir agregar nuevas diapositivas con título e ideas	5

Referirse al Anexo 1 para una lista completa de todas las historias de usuario

3.1.3. Tareas técnicas

Se catalogó como tarea técnica a cualquier ítem de trabajo que tiene como objetivo mejorar una parte del código o de la infraestructura, pero no agrega una nueva funcionalidad o mejora tangible para el usuario final.

En ocasiones, a medida que se implementan nuevas funcionalidades bastante similares a las ya existentes, se duplican algunas porciones de código. Se evidencia entonces la necesidad de crear una tarea técnica para extraer el código común y tener así reutilización de lógica en vez de duplicación. Las tareas técnicas constan de los siguientes campos:

- *Título: <Mejora técnica que se desea realizar>*
- *Descripción: <Explicación completa de cómo implementar los cambios>*
- *Dificultad: <Dificultad general del trabajo técnico>*
- *Impacto: <Cuán beneficioso va a ser para el proyecto>*

Algunos ejemplos de tareas técnicas con este formato son:

Tabla 2. Tareas técnicas implementadas en el proyecto

#	Título	Descripción	Dificultad	Impacto
1	Separar la lógica de controlador de la lógica de rutas.	Extraer la funcionalidad a un nuevo archivo de controladores con funciones separadas, ya que actualmente se encuentra todo el código bajo las funciones de que definen rutas.	Medio	Medio
2	Crear directivas para código HTML común	Utilizar directivas en vez de código HTML que está repetido en varias plantillas. Esto permite reutilizar código y brinda mejor legibilidad a las plantillas.	Medio	Alto

Para ver una lista completa de todas las tareas técnicas, referirse al Anexo 2.

3.1.4. Corrección de defectos

Se catalogaron como defectos a todos los problemas o comportamientos no deseados que el usuario o revisor de la aplicación encontró mientras la utilizaba. Los defectos se evidencian después de que una funcionalidad ha sido desarrollada y considerada como terminada. Una vez identificado un defecto, se debe crear un ítem para corrección de defecto, que cuente con el siguiente formato:

Defecto:

<Descripción del error que presenta actualmente el sistema>

¿Cómo reproducirlo?:

<Estado de la aplicación y acciones que el usuario realizó para evidenciar el defecto>

Comportamiento esperado:

<Descripción del comportamiento que el sistema debería presentar para considerar que el problema ha sido solucionado>

Durante el desarrollo de este proyecto, los defectos fueron identificados junto con la profesora guía durante las sesiones de revisión de funcionalidades. En base a ellos se crearon ítems de corrección de defectos.

Tabla 3. Ejemplo de corrección de defecto implementada en el proyecto

#	Defecto	¿Cómo reproducirlo?	Comportamiento esperado
1	La aplicación no valida que una diapositiva tenga título al momento de guardarla	1. Ingresar a cualquier presentación. 2. Agregar o editar una diapositiva. 3. Llenar los campos de ideas. 4. Dejar el campo de título en blanco. 5. Guardar la diapositiva. 6. La diapositiva se guarda con título vacío.	Al intentar guardar una diapositiva con título en blanco, se debe mostrar un mensaje explicando que el título es un campo obligatorio y no se debe almacenar.

Para ver una lista completa de todas las correcciones de defectos, referirse al Anexo 3.

3.2. Diseño e implementación

El desarrollo e implementación de la aplicación fue separado en base a las cuatro iteraciones que se definieron en la planificación. Cada una de ellas agrupa un conjunto independiente de funcionalidades que el sistema ofrece.

3.2.1. Visualización de presentaciones

En esta primera parte se construirán las bases de la aplicación y se trabajará en la parte visual referente a cómo se desplegarán las diapositivas en el navegador del usuario. Por el momento los contenidos de las presentaciones no serán almacenados en una base de datos, sino que se incluirán en el código del lado del servidor. Esto permitirá enfocarse en los aspectos de presentación y conexión entre la parte del cliente y el servidor, dejando a un lado la persistencia y manipulación de los datos, que será implementada en la iteración destinada a gestión de presentaciones.

El primer diseño planteado para mostrar las presentaciones es:

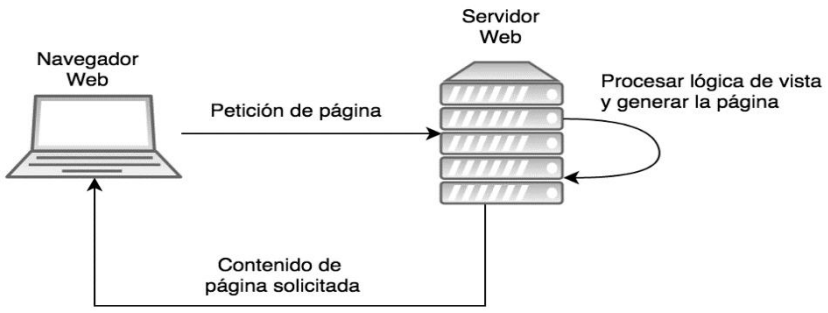


Figura 2. Diagrama de interacción entre cliente y servidor.

a) El servidor se encarga de procesar la lógica de la vista. El cliente solo pide la página y presenta en el navegador el contenido que el servidor le envíe como respuesta.

Bajo este diseño, es necesario que el servidor procese una plantilla de presentación e incorpore en ella el contenido de las presentaciones. El contenido de las presentaciones son datos dinámicos que por el momento se encuentran definidos en el servidor, pero posteriormente serán consultados desde una base de datos.

Para implementar este diseño en código, se requiere solamente de Express y Node.js. Se puede generar una nueva aplicación de Express en el directorio del proyecto utilizando el comando: express

Figura 3. Estructura de carpetas y archivos generados por Express

La aplicación base creada con Express, contiene todos los archivos y código necesario para servir una página web básica. Las rutas, vistas y recursos públicos son configurados en app.js. El archivo bin/www crea y levanta un nuevo servidor http, en el cual se despliega la aplicación definida en app.js.

El archivo package.json contiene información acerca de la aplicación, dependencias y scripts de ejecución. Para instalar las dependencias se ejecuta desde la línea de comandos: `npm install`. Para ejecutar la aplicación se ejecuta: `npm start`

Para trabajar en la visualización de las presentaciones es necesario crear nuevas rutas a las que los usuarios puedan acceder, plantillas para desplegar la información y hojas de estilo para controlar la apariencia de los elementos dentro del navegador.

Para crear nueva rutas, se debe agregar sus valores a alguno de los archivos que se encuentran en la carpeta routes. Para agregar plantillas con la estructura de las páginas, se debe crear archivos dentro de la carpeta views. Las hojas de estilo son contenido estático y por tanto deben ir bajo la carpeta public. Por motivos de organización es preferible colocarlos bajo una subcarpeta denominada stylesheets, como la que provee el proyecto generado por express.

Por el momento, las plantillas son interpretadas por el motor de plantillas Jade, el cual convierte elementos en etiquetas HTML y reemplaza los nombres de variables por contenido dinámico que proviene del archivo de rutas.

Además de la estructura de las páginas, es necesario controlar la apariencia, posicionamiento y tamaño de los elementos. Para ello se incluyó la librería de estilos *Bootstrap* y se crearon varias definiciones de estilos en CSS dentro del archivo style.css.

A continuación se muestra una pantalla de presentación con una diapositiva de ejemplo. En ella se puede apreciar la organización de los elementos y la estética general de la página:

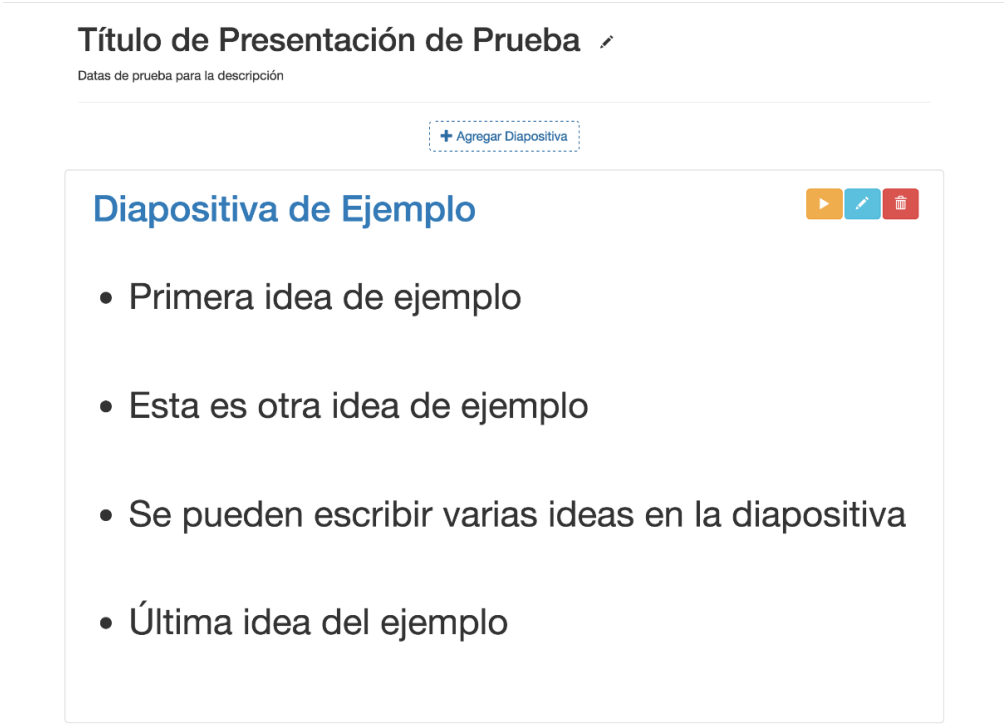


Figura 4. Estructura y estilo de la página de presentación de ejemplo

Para visualizar una secuencia más completa de todas las páginas con que cuenta la aplicación, referirse al Anexo 4.

3.2.2. Gestión de presentaciones

Para la gestión de presentaciones, se evidencia la necesidad de crear una aplicación SPA, ya que se debe brindar respuestas rápidas y no interrumpir a los usuarios cada vez que realicen una acción de edición dentro de la página. Por lo tanto, se alteró el diseño planteado para desarrollar la parte de la aplicación relacionada con presentaciones. A continuación se muestra la nueva forma de interacción entre el cliente y el servidor para ocupar un estilo de aplicación SPA.

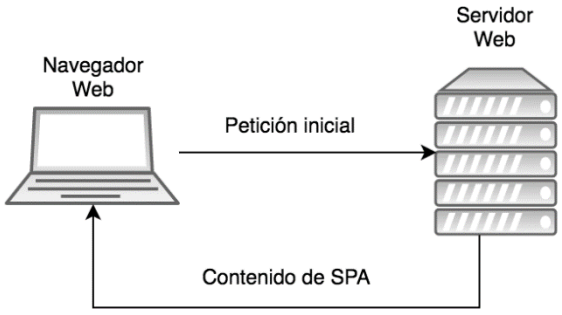


Figura 5. Petición inicial de una SPA

a) El navegador recibe desde el servidor todo el contenido de la aplicación SPA.

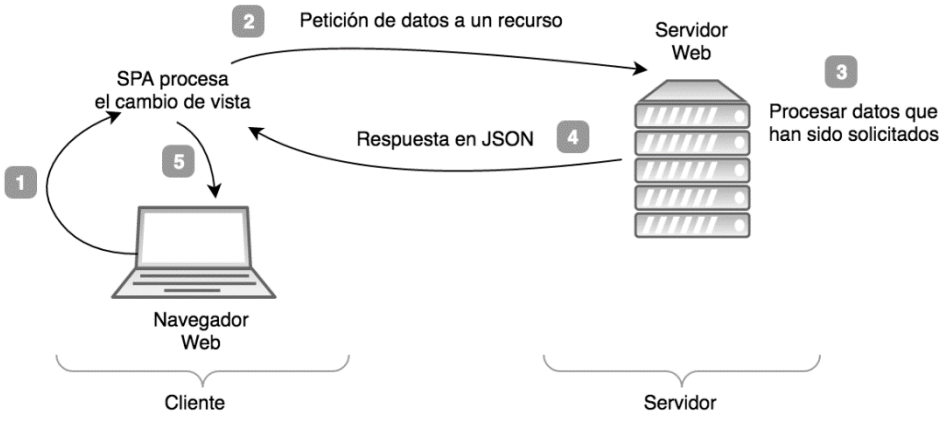


Figura 6. Procesamiento de peticiones con una SPA

a) Una vez que el cliente ya tiene la aplicación SPA, esta se encarga de procesar y presentar las nuevas vistas, realizando peticiones a recursos web, sólo cuando requiere información dinámica del servidor.

Esta nueva estructura implica un cambio radical en la forma en que se procesa la información. Toda la lógica relacionada con la presentación que se encontraba en las plantillas interpretadas por Jade y las rutas y controladores de Express, deberán moverse a una nueva aplicación SPA de AngularJS.

AngularJS provee vistas y plantillas de directivas para agregar todo el código HTML que la aplicación requiera. La lógica de peticiones al servidor se maneja desde los servicios, los cuales utilizan una librería propia de angular llamada $http para realizar las peticiones y obtener los datos en formato JSON. Los controladores proveen la lógica de presentación, alterando la vista y pidiendo información a los servicios en caso de requerir datos dinámicos del servidor.

AngularJS facilita considerablemente la manipulación de los elementos que se muestran en pantalla, especialmente cuando se tienen arreglos de longitud variable como en el caso de las ideas de las diapositivas. Otra ventaja que tiene es la facilidad para definir en la vista las condiciones bajo las cuales se desea ocultar o mostrar ciertos elementos. Esto resulta útil por ejemplo para ocultar el botón de agregar idea una vez que se ha alcanzado el número máximo de ideas permitidas en una diapositiva. Para implementar estas funcionalidades se utilizan los atributos ng-hide o ng-show desde los archivos HTML.

Una vez que la lógica de presentación ha sido migrada a la nueva estructura SPA con AngularJS, es necesario actualizar la lógica del servidor para exponer recursos que transmitan datos en formato JSON. Además se requiere conectarse a la base de datos, con lo cual el diseño de la aplicación será:

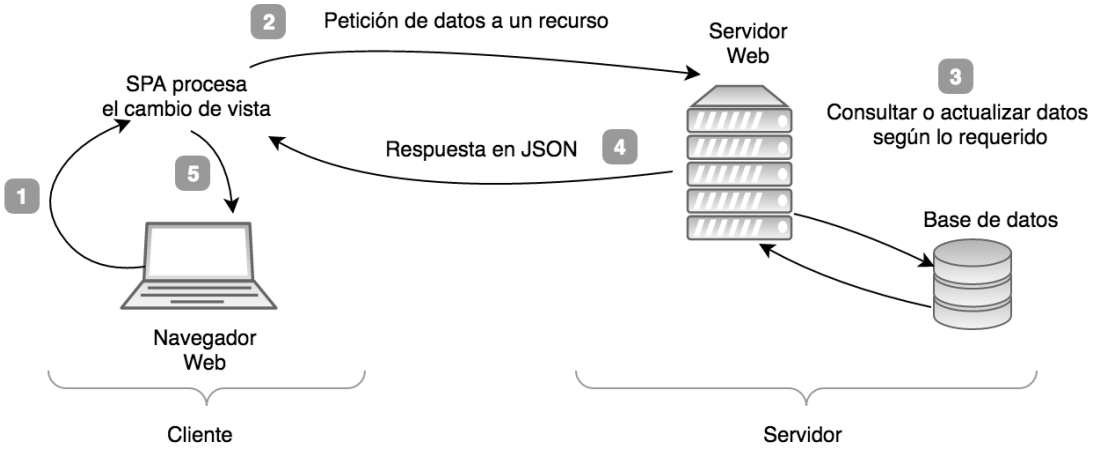


Figura 7. Procesamiento completo de una petición con SPA y base de datos

Se utiliza la base de datos documental MongoDB, que utiliza el formato BSON para almacenar los documentos, por lo tanto la comunicación desde JavaScript hacia la base de datos es relativamente sencilla.

El primer paso es definir las configuraciones de conexión, desconexión, credenciales y dirección para acceder a la base de datos. Posteriormente se define el esquema de una presentación, apoyándose en la librería Mongoose. Este esquema será utilizado desde los controladores para buscar, agregar, editar o eliminar datos de presentaciones en la base. Finalmente los controladores retornarán datos en formato JSON para transmitir un resultado a la petición que el cliente solicitó.

3.2.3. Control remoto de presentaciones

Para la conexión entre el control remoto y la diapositiva que se está presentando, se utilizará Socket.IO, el cual debe ser instalado tanto en el lado del cliente como en el servidor. Este permitirá tener comunicación en tiempo real entre clientes específicos a través del envío y recepción de mensajes con el servidor.

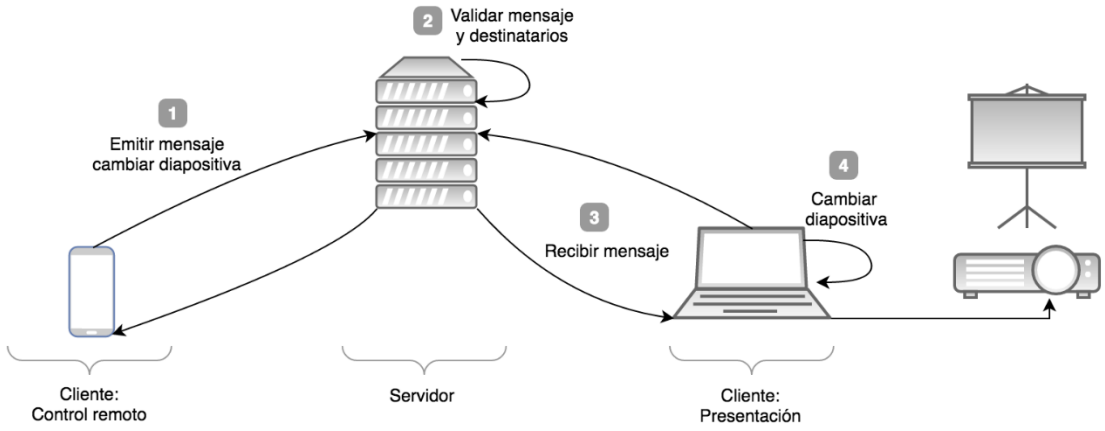


Figura 8. Diagrama de comunicación entre control remoto y presentación

a) Esquema de conexión y transmisión de mensajes desde el control remoto hasta el cliente que proyecta la presentación

El cliente móvil tiene abierta la página del control remoto de una presentación específica y emite un mensaje con el identificador de la diapositiva que quiere cambiar. El servidor transmite el mensaje a todos los clientes que tengan abierta la misma presentación. Los clientes reaccionan ante el mensaje, cambiando la diapositiva actual por aquella especificada en el mensaje. En caso de que existan clientes que tienen abiertas otras presentaciones, estos no se verán afectados.

En la implementación del código se utiliza el concepto de *rooms* de Socket.io, con el cual el servidor puede registrar a los clientes en salas distintas. Para el caso de esta aplicación, cada presentación es una sala distinta. Se escribió la lógica para que tanto el cliente del control remoto como el cliente de la presentación estén conectados a la misma sala, que lleva por nombre el

identificador único de cada presentación. De esta manera, los mensajes emitidos por el control remoto llegan al servidor y son transmitidos al resto de clientes de la sala, que son los clientes que se encuentren proyectando esa presentación.

3.2.4. Autenticación de usuarios

La parte referente a cuentas de usuarios es un módulo que se desarrolló dentro del aplicativo web de gestión de presentaciones utilizando las mismas herramientas tecnológicas. Sin embargo, se necesitó introducir un mecanismo para identificar de forma repetible si un usuario ya se ha autenticado en el sistema. Para ello se utilizaron *tokens* de autorización, los cuales permiten al servidor identificar quién está realizando la petición y si es un usuario válido o no. Los *tokens* cuentan con un tiempo de expiración y se generan después de que el usuario ingresa al sistema con sus credenciales correctas para nombre de usuario y contraseña.

El proceso de autenticación del usuario sigue el siguiente esquema:

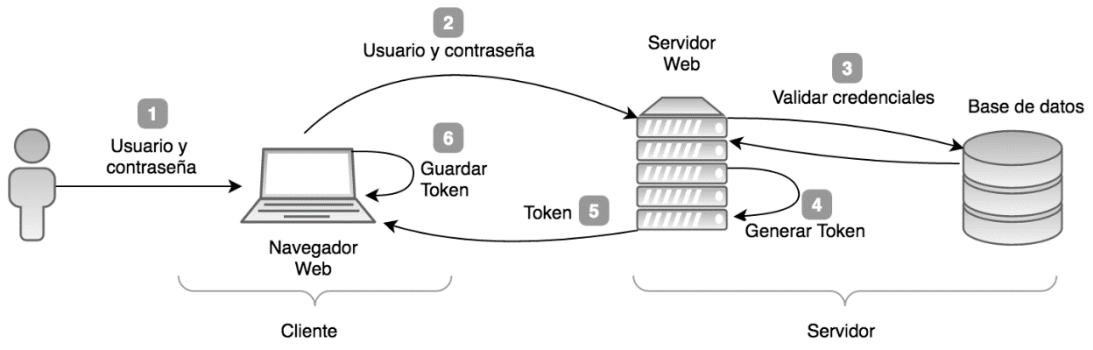


Figura 9. Esquema de validación de credenciales y generación de token.

a) Las credenciales del usuario son enviadas desde el navegador al servidor para que las valide con la base de datos y devuelva un token de autorización, el cual es almacenado en el navegador

Una vez que el usuario ha recibido un token, ya puede realizar peticiones a recursos que requieren de autorización. Para acceder a los recursos protegidos, el cliente envía el token al servidor en vez de ingresar cada vez usuario y contraseña.

El siguiente diagrama muestra este tipo de interacciones:

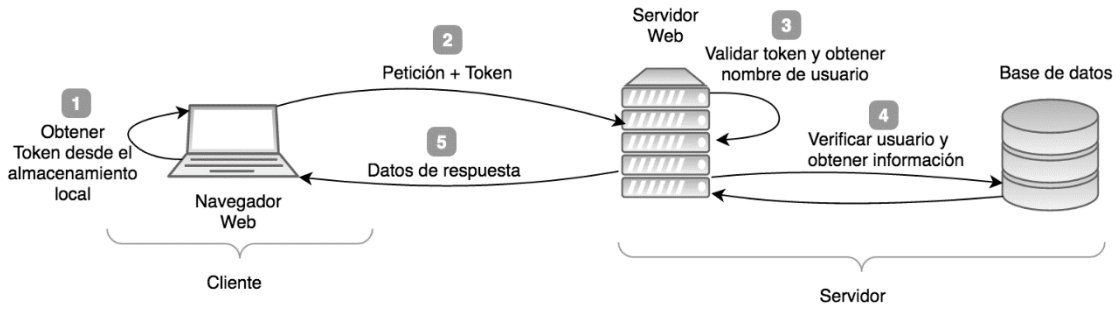


Figura 10. Esquema de petición de recursos protegidos enviando token

a) Cuando el cliente envía un token de autorización, el servidor lo verifica. Se extrae la información del usuario y se valida con la base de datos. En caso de una validación exitosa, se realiza la consulta solicitada y se envía una respuesta al cliente

3.3. Pruebas

Como parte del desarrollo del aplicativo, se realizaron varias pruebas para verificar que el código y las funcionalidades del sistema presenten un comportamiento adecuado.

Durante las etapas de desarrollo y revisión del proyecto, se realizaron múltiples interacciones y pruebas manuales en el sistema Esto permitió agregar algunas nuevas funcionalidades e identificar algunos errores que luego fueron corregidos como parte de las correcciones de defectos. Sin embargo, la parte más relevante de pruebas en el sistema la componen las pruebas automáticas de integración. Estas pruebas validan las acciones y respuestas que el servidor envía cuando se utilizan los recursos REST a través de peticiones HTTP. Estas pruebas garantizan que el servidor tenga la lógica adecuada y que pueda responder de manera correcta a cualquier interfaz de usuario o cliente web que haga uso de sus servicios.

Una de las principales ventajas que presentan este tipo de pruebas es que permiten obtener retroalimentación rápida al desarrollador cuando realiza cambios en el código del proyecto. Al tratarse de pruebas automáticas, estas requieren una cantidad de esfuerzo considerable al momento de escribirlas pero luego se las puede ejecutar de manera fácil, rápida y repetible. En esta sección se detalla la implementación de estas pruebas en el proyecto.

3.3.1. Estructura de las pruebas

Las pruebas automáticas de integración realizan llamadas HTTP para validar las respuestas de los recursos REST que la aplicación del lado del servidor expone. Todas las pruebas se escribieron en JavaScript utilizando el framework de pruebas *Mocha*, acompañado de las librerías *supertest* y *assert* de Node.js. Para realizar las pruebas, se utilizaron algunas estructuras y funciones que permiten verificar que las llamadas a la aplicación retornen los resultados esperados. A continuación se muestra un ejemplo del proyecto con los distintos elementos que componen una prueba de este tipo.

La función `describe` sirve para agrupar un conjunto de pruebas bajo un nombre que las categorice y separe del resto de pruebas. La función recibe como primer argumento el nombre del conjunto de pruebas y como segundo argumento una función que contendrá el desarrollo de las mismas. Dentro de dicha función, se tienen varios bloques de código como `before`, `it` y `after`.

Al inicio de la prueba existe una fase de inicialización o preparación, en la cual se crean y definen todas las variables y datos que se necesitarán para la prueba. Estas sentencias suelen ir dentro del bloque de código `before`. A continuación se muestra un ejemplo:

```
before(function () {
  presentationData = {
    title: 'Sample Title',
    description: 'Sample Description',
    author: process.env.EMAIL_FOR_TESTS,
    slides: []
  }
})
```

Figura 11. Bloque before de inicialización en la prueba.

a) Código en tests/integration/presentations/presentationsCreate.js

El código que realiza llamadas a los recursos que se desean probar y verifica sus respuestas se encuentra dentro del bloque `it`. La función `it` al igual que `describe`, recibe como primer argumento el nombre que se dará a la prueba y como segundo argumento la función que contiene el código de prueba. Para levantar los recursos de la aplicación y hacer los llamados HTTP, se utiliza un objeto de la librería supertest, que llama al método HTTP adecuado. En caso de requerirse también se puede enviar datos en la petición utilizando el método `send`. La verificación se realiza a través del método `expect`. Se puede verificar el código HTTP que retorna la petición, así como el tipo y contenido de los datos de respuesta.

```
it('should create a new presentation', function (done) {
  request(app)
    .post('/api/presentations')
    .set('Accept', 'application/json')
    .send(presentationData)
    .expect('Content-Type', /json/)
    .expect(function (res) {
      mockPresentationAutoGeneratedValues(res.body)
    })
    .expect(201, expectedPresentation, done)
})

var mockPresentationAutoGeneratedValues = function (presentation) {
  presentation._id = 'AnyId'
  presentation.createdAt = 'Yesterday'
  presentation.updatedAt = 'Now'
}
```

Figura 12. Bloque it con las acciones y verificaciones en la prueba.

a) Código en tests/integration/presentations/presentationsCreate.js

Finalmente, la función `after` permite realizar las tareas de limpieza una vez que la prueba ha terminado. Esta etapa es importante, ya que ayuda a mantener los datos de prueba consistentes en la aplicación y disminuye las posibilidades de que la prueba genere resultados engañosos o falsos positivos.

```
after(function (done) {
  request(app)
    .del('/api/presentations/author/' + process.env.EMAIL_FOR_TESTS)
    .expect(204)
    .end(done)
})
})
```

Figura 13. Bloque after con instrucciones de limpieza de la prueba

a) Código de tests/integration/presentations/presentationsCreate.js

De manera general, esos son los elementos que fueron más importantes en la elaboración de las pruebas. Para una inspección más detallada del resto de pruebas, estas se encuentran en los archivos bajo la ruta `tests/integration/`

3.3.2. Ejecución de las pruebas

Para ejecutar las pruebas automáticas del proyecto, se necesita utilizar el comando `mocha` seguido de la dirección en donde se encuentran todas las pruebas. El programa mocha se encuentra dentro de la dirección `node_modules/.bin`. La ubicación de todos los archivos de prueba dentro del proyecto se muestra a continuación:

Figura 14. Estructura de archivos de prueba en el proyecto

Por lo tanto, el comando completo que se debe ejecutar desde el directorio raíz de la aplicación es: `node_modules/.bin/mocha tests/integration/*`

Para facilitar la ejecución de pruebas desde la consola, se asignó el comando completo a la instrucción `test` dentro de la sección `scripts` del archivo `package.json` del proyecto. De esta manera, se pueden ejecutar todas las pruebas desde la línea de comandos utilizando la instrucción `npm test`

```
➜  puente-sistemas-presentaciones git:(master) npm test

> presentations-app@0.0.0 test /Users/spuente/puente-sistemas-presentaciones
> mocha tests/integration/*



  Presentations Create
Mongoose connected to mongodb://127.0.0.1/Presentations
POST /api/presentations 201 38.718 ms - 225
    ✓ should create a new presentation (67ms)
DELETE /api/presentations/author/emailForTest@sample.com 204 5.297 ms - -
```

Figura 15. Resultado de ejecución de una prueba de integración

3.4. Procesos, configuraciones y despliegue

A continuación se detallan algunos de los procesos y elementos importantes que intervinieron en la implementación y despliegue de las funcionalidades de la aplicación.

3.4.1. Control de versiones

Para llevar un control de los cambios que se introdujeron en el código durante el proceso de implementación, se utilizó la herramienta de control de versiones Git.

Al trabajar sobre un repositorio con control de versiones, es posible ver en qué punto se introdujeron ciertos cambios y cómo la base de código ha ido evolucionando. En caso de requerirse, también se puede regresar a un punto específico en la historia del código, en el cual se hayan guardaron los cambios en el repositorio.

3.4.2. Repositorio remoto

Si un proyecto se almacena únicamente en un repositorio local de la máquina de desarrollo, existe el riesgo de perder el código si la computadora se daña o se pierde. Es importante por lo tanto respaldar el código y poder acceder a este desde cualquier lugar. Para ello se creó un repositorio remoto en Bitbucket, accediendo a través del sitio *https://www.bitbucket.org*.

Al utilizar git, es posible sincronizar el repositorio local con el repositorio remoto. De esta manera se tiene acceso al código desde cualquier máquina y se elimina el riesgo de perder el trabajo realizado.

3.4.3. Despliegue de la aplicación

La aplicación fue desplegada en un ambiente local y en un ambiente de producción de acceso público.

El despliegue local requiere levantar la base de datos MongoDB a la cual se conectará la aplicación. Posteriormente se debe ejecutar el comando `npm start` desde el directorio raíz del proyecto para iniciar la aplicación. Como alternativa, durante la etapa de desarrollo se puede instalar y utilizar el paquete `nodemon`, que reinicia la aplicación cada vez que identifica que se han realizado cambios sobre cualquiera de sus componentes.

Para desplegar la aplicación al ambiente de producción, se utilizó el servicio en línea Heroku a través del sitio *https://www.heroku.com/*. Heroku provee una forma fácil de desplegar una aplicación que corre sobre el entorno de ejecución Node.js.

Para utilizar una base de datos MongoDB en producción, se utilizó el proveedor mLab, al cual se accede mediante https://mlab.com/. Este servicio ofrece una base de datos en la nube, que puede ser accedida mediante la aplicación.

Para desplegar localmente o a producción, es necesario configurar las variables de entorno. En el ambiente local, se puede crear un archivo `.env` en el directorio raíz de la aplicación. Este archivo deberá incluir las configuraciones de ambiente según se detalla en el archivo `README.txt` y no deberá ser almacenado en el repositorio `git`, ya que contendrá valores secretos. Para el despliegue a producción, las variables de entorno deben ser configuradas en el servidor a través de comandos o la interfaz web de Heroku. Además, se debe crear un archivo `Procfile` con los comandos que Heroku requiere para levantar la aplicación.

Para desplegar nuevas versiones del código a producción, es necesario configurar en el repositorio una nueva dirección remota que apunte a Heroku. Cuando se sube nuevo código a la rama principal de ese repositorio remoto, Heroku intentará iniciar la aplicación utilizando los comandos detallados en el archivo `Procfile`.

CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

El aplicativo desarrollado en este proyecto permitió demostrar que es posible presentar diapositivas en un navegador web y controlarlas desde otro navegador mediante el uso de tecnologías web. Se evidenció por lo tanto que es posible desarrollar aplicaciones con interacciones en tiempo real entre clientes web sin necesidad de construir aplicaciones nativas o comprar dispositivos externos de hardware.

Las funcionalidades implementadas para que los usuarios gestionen el contenido de sus presentaciones siguen una estructura coherente y sencilla de utilizar. Esto se puede apreciar en el uso de formularios, botones, barras e íconos contextuales que permiten a los usuarios navegar intuitivamente por la aplicación. Además, como se evidenció en la elaboración del manual de usuario y en las interacciones de los docentes con la aplicación, se requiere de pocas acciones para ingresar y gestionar la información.

Se logró crear una aplicación que se adapta adecuadamente a los distintos tipos de pantallas y dispositivos. Esto se demuestra cuando al abrir las páginas en un dispositivo pequeño, sus elementos se ajustan al tamaño de la pantalla y cargan solamente botones y acciones sencillas para presentar a través de control remoto.

La aplicación requiere que el usuario pueda interactuar constantemente con ella. Por lo tanto se valida que fue adecuado utilizar un *Single Page Application* (SPA), ya que este provee respuestas rápidas en las páginas web sin necesidad de refrescarlas completamente, disminuyendo el tiempo de espera y evitando interrupciones en el flujo de trabajo del usuario.

4.2. Recomendaciones

Se recomienda utilizar este proyecto como base para implementar nuevas funcionalidades y características que se requieran a futuro, tales como agregar pie de página a las diapositivas con imágenes, ofrecer el servicio de carga de imágenes en el servidor y permitir publicar ciertas presentaciones para acceso público.

Previo a la publicación de la aplicación para consumo masivo, es recomendable realizar pruebas de carga y monitorear el rendimiento que el servidor tendrá con un número similar o superior de peticiones a las que se espera tener por parte de los usuarios.

En base a la experiencia de este proyecto, se recomienda que para implementar tipos de funcionalidades que el desarrollador no ha realizado antes, primero se elaboren pruebas de concepto o ejemplos sencillos. Estos brindarán información acerca de la complejidad de la funcionalidad y permitirán evaluar cuál es el camino más óptimo que se debe tomar para la implementación real.

Es importante analizar e identificar cuáles son las herramientas adecuadas para desarrollar una aplicación, basándose en el tipo de aplicación y funcionalidades que se requiere. Al utilizar herramientas que faciliten la resolución de los problemas del proyecto, se mejorará la velocidad de desarrollo y se podrá escribir código más compacto, sencillo y mantenible.

Cuando se realiza el diseño de las interfaces de usuario, se recomienda pensar siempre en las formas en que los usuarios van a utilizar la aplicación. En base a ello se puede jugar con los elementos visuales para crear una experiencia de usuario intuitiva y sencilla de utilizar.

Se recomienda la utilización de pruebas automáticas, ya que permiten obtener retroalimentación más rápida y continua durante el desarrollo.

REFERENCIAS

Beck, K. (2004). *Extreme Programming Explained: Embrace Change.* MA: Addison-Wesley. Recuperado el 22 de agosto de 2016, de https://www.safaribooksonline.com/library/view/extreme-programming-explained/0321278658/ch04.html

Bretz, A., & Ihrig, C. (2014). *Full Stack JavaScript Development With MEAN.* Australia: Sitepoint. Recuperado el 8 de julio de 2016, de https://www.safaribooksonline.com/library/view/full-stack-javascript/9781457174377/ch01.html

Google. (s.f.). *What is Angular?* Recuperado el 7 de julio de 2016, de https://docs.angularjs.org/guide/introduction

Hahm, E. (2016). *Express in Action: Writing, building, and testig Node.js applications.* New York: Manning Publications. Recuperado el 8 de julio de 2016, de https://www.safaribooksonline.com/library/view/express-in-action/9781617292422/kindle_split_009.html

Microsoft. (2014). *Set the timing and speed of a transition*. Recuperado el 8 de marzo de 2016, de https://support.office.microsoft.com/en-us/article/Set-the-timing-and-speed-of-a-transition-c3c3c66f-4cca-4821-b8b9-7de0f3f6ead1?CorrelationId=15ab40c4-806e-4083-b708-9fd56fd3fb45&ui=en-US&rs=en-US&ad=US#__toc242693353

MongoDB, Inc. (s.f.). *Introduction to MongoDB*. Recuperado el 1 de julio de 2016, de https://docs.mongodb.com/manual/introduction/

Node.js Foundation. (s.f.). *Node.js.* Recuperado el 25 de junio de 2016, de https://nodejs.org/en/

Rubio, M. (1 de septiembre de 2013). *¿Qué es Diseño Adaptativo?* Recuperado el 8 de marzo de 2016, de The Joomla! Community Magazine: http://magazine.joomla.org/es/ediciones-anteriores/sept-2013/item/1514-que-es-diseno-adaptativo

Socket.io. (s.f.). *Socket.io 1.0 is here*. Recuperado el 7 de julio de 2016, de http://socket.io/

StrongLoop, Inc. (s.f.). *Express.* Recuperado el 24 de agosto de 2016, de http://expressjs.com/es/

The jQuery Foundation. (s.f.). *jQuery*. Recuperado el 1 de julio de 2016, de https://jquery.com/

ANEXOS

Anexo 1: Listado de historias de usuario

#	Título	Descripción	Complejidad (puntos)
1	Creación de cuenta	Permitir al usuario crear una cuenta con información personal básica para identificarlo en el sistema	3
2	Acceso privado a presentaciones	Cada usuario puede acceder únicamente a las presentaciones que han sido creadas desde su cuenta	2
3	Cerrar sesión	Proveer a los usuarios la opción de salir de su cuenta, evitando que los siguientes usuarios del navegador accedan a su contenido	2
4	Crear nueva presentación	El usuario podrá crear una nueva presentación con título y descripción	2
5	Crear diapositiva con título e ideas	Dentro de una presentación permitir agregar nuevas diapositivas con título e ideas	5
6	Agregar diapositiva en cualquier posición de presentación	La nueva diapositiva debe almacenar su posición y puede agregarse encima o debajo de cualquier otra diapositiva de la presentación	3
7	Crear diapositiva con título e imagen	Para mostrar contenido de manera gráfica, se debe permitir agregar un tipo de diapositiva con título e imagen grande que ocupe la mayor parte del cuerpo de la diapositiva	5
8	Tener distintos tipos de diapositivas bajo una misma presentación	Se requiere poder crear y visualizar los distintos tipos de diapositivas dentro de una misma presentación. Esto ayudará a que la presentación sea variada y no muy monótona	5
9	Visualizar fecha de creación de las presentaciones	Para identificar rápidamente la época y contexto en que cierta presentación fue creada, se requiere mostrar la fecha de creación junto a cada presentación en el listado principal	2
10	Visualizar la lista de presentaciones en orden descendente	Para acceder rápidamente a las presentaciones que han sido creadas recientemente, se debe ordenar a las presentaciones de manera descendente en base a fecha de creación	1

11	Editar título y descripción de la presentación	Para reflejar los cambios de la presentación, es necesario que los campos título y edición sean editables desde la pantalla de detalle de la presentación	2
12	Longitud mínima para creación de contraseña de usuario	Para promover el uso de contraseñas más fuertes y mitigar el riesgo de acceso no autorizados, se impondrá una longitud mínima de 8 caracteres al crear la contraseña de la cuenta	2
13	Validar que se registre una cuenta de correo única y válida	Para identificar a cada usuario de manera única y verificar su identidad, se enviará un email de verificación al momento de creación. El usuario deberá acceder a un link desde su email para activar la cuenta. Así se espera mitigar el riesgo de creación masiva de cuentas falsas.	8
14	Verificar email sin distinguir mayúsculas ni minúsculas	Para eliminar confusiones y facilitar el inicio de sesión al usuario, se validará la dirección de email sin tomar en cuenta mayúsculas o minúsculas	2
15	Eliminar una presentación	Permitir remover presentaciones que ya no consideren relevantes o necesarias. Debe contar con un mensaje de confirmación antes de proceder	1
16	Eliminar una diapositiva	Permitir remover contenido que ya no es necesario o relevante dentro de una presentación. Debe contar con un mensaje de confirmación antes de proceder	1
17	Editar una diapositiva	Poder alterar su contenido (título, ideas o imagen) sin necesidad de eliminar y crear una nueva diapositiva. Debe permitir cambiar los contenidos e incluso el tipo de diapositiva.	3
18	Visualizar cada diapositiva en una página completa	Permitir a la audiencia una visualización clara y sin distracciones del contenido de esa diapositiva. La barra de navegación debe estar presente en la parte superior de la pantalla, pero no debe mostrarse cuando se presenta una diapositiva sino solamente cuando el usuario está navegando dentro de la página	2
19	Acceder al modo control remoto	Implementar botones con íconos adecuados al lado de cada presentación para que los usuarios puedan acceder al modo de control remoto, especialmente cuando la pantalla es pequeña	1

20	Listar las presentaciones y seleccionar desde el control remoto	Para indicar al sistema cual es la presentación que se desea controlar, se podrá dar click sobre el título de la presentación o sobre el botón de control remoto	2
21	Desde el control remoto, listar las diapositivas de la presentación	Mostrar el listado de las diapositivas en el control remoto, para indicar al sistema cual es la diapositiva que se desea proyectar	1
22	Tener flechas claras y grandes en el control remoto	Permitir la navegación hacia adelante o atrás en una presentación con facilidad, mediante botones grandes que se acoplen al tamaño de pantalla	2
23	Tener un botón para ir al inicio de la presentación desde el control remoto	Empezar la proyección desde el inicio, independientemente de la diapositiva que esté siendo presentada en ese momento	1
24	Link para creación de usuario	Agregar un link en la pantalla de inicio de sesión hacia "creación de usuario" para facilitar el registro de un nuevo usuario que entra por primera vez al sitio	1
25	Controlar únicamente la presentación que ha sido seleccionada	Se necesita desplazar los contenidos de la presentación actual, sin alterar otras presentaciones del usuario o de otros usuarios que se encuentren presentando en ese momento	3

Anexo 2: Listado de tareas técnicas

#	Título	Descripción	Dificultad	Impacto
1	Separar la lógica de controlador de la lógica de rutas.	Extraer la funcionalidad a un nuevo archivo de controladores con funciones separadas, ya que actualmente se encuentra todo el código bajo las funciones de que definen rutas.	Medio	Medio
2	Crear directivas para código HTML común	Utilizar directivas en vez de código HTML que está repetido en varias plantillas. Esto permite reutilizar código y brinda mejor legibilidad a las plantillas.	Medio	Alto
3	Utilizar métodos y elementos de AngularJS en vez de métodos nativos	Cambiar a elementos que provee AngularJS en vez de los elementos nativos, ya que estos brindan mejor compatibilidad y abstraen la implementación de bajo nivel, por ejemplo utilizar ng-href en vez de href para links	Bajo	Alto
4	Limpiar código antiguo	Durante la transición a SPA, existen algunas variables, métodos y archivos que ya no son necesarios pero que aún se encuentran en la aplicación. Removerlos para evitar futuras confusiones.	Bajo	Alto
5	Clasificar el código en carpetas por contexto del proyecto y no por tipo de archivos	Separar el código en base a carpetas que representen conceptos del proyecto como presentations, slides, authentication, etc. En vez de separarlas por funcionalidades como servicios, controladores, vistas, etc.	Medio	Medio
6	Remover estilos y atributos innecesarios	Las vistas han ido evolucionando y algunos estilos ya no existen o ya no tienen efecto en la apariencia visual de la página. Quitarlos para remover código muerto y mejorar la legibilidad de las plantillas en HTML	Bajo	Medio

Anexo 3: Listado de corrección de defectos

#	Defecto	¿Cómo reproducirlo?	Comportamiento esperado
1	La aplicación no valida que una diapositiva tenga título al momento de guardarla	1. Ingresar a cualquier presentación. 2. Agregar o editar una diapositiva. 3. Llenar los campos de ideas. 4. Dejar el campo de título en blanco. 5. Guardar la diapositiva. 6. La diapositiva se guarda con título vacío.	Al intentar guardar una diapositiva con título en blanco, se debe mostrar un mensaje explicando que el título es un campo obligatorio y no se debe almacenar nada.
2	Problema al ingresar ideas repetidas en una misma diapositiva	1. Ingresar a cualquier presentación. 2. Agregar o editar una diapositiva. 3. Agregar título. 4. Agregar dos ideas iguales. 5. Guardar la diapositiva. 6. La diapositiva no se guarda.	El sistema debería permitir guardar ideas duplicadas en una misma diapositiva
3	Mensaje de validación de usuario demasiado explícito	1. Ingresar a la aplicación. 2. Ingresar nombre de usuario correcto 3. Ingresar contraseña incorrecta 4. Entrar. 5. Se muestra un mensaje de nombre de usuario incorrecto	La aplicación debería decir nombre de usuario o contraseña incorrecta, para no revelar a posibles atacantes que un nombre de cuenta existe

Anexo 4: Manual de usuario

La aplicación ha sido diseñada para que su uso sea bastante intuitivo para usuarios finales. Para conseguir este propósito, se han implementado varios elementos como botones con íconos y colores contextuales, mensajes de validación en los formularios y navegación sencilla dentro del sitio. La aplicación también se acopla a distintos tamaños de pantallas y en base a ello decide las funcionalidades y elementos que mostrará.

Es por ello que de manera general, se espera que los usuarios puedan descubrir el funcionamiento del sistema en poco tiempo y les sea sencillo utilizarlo.

Sin embargo, este manual de usuario es un complemento que puede ayudar a los usuarios a comprender mejor cómo utilizar ciertas funcionalidades que les parezcan complejas.

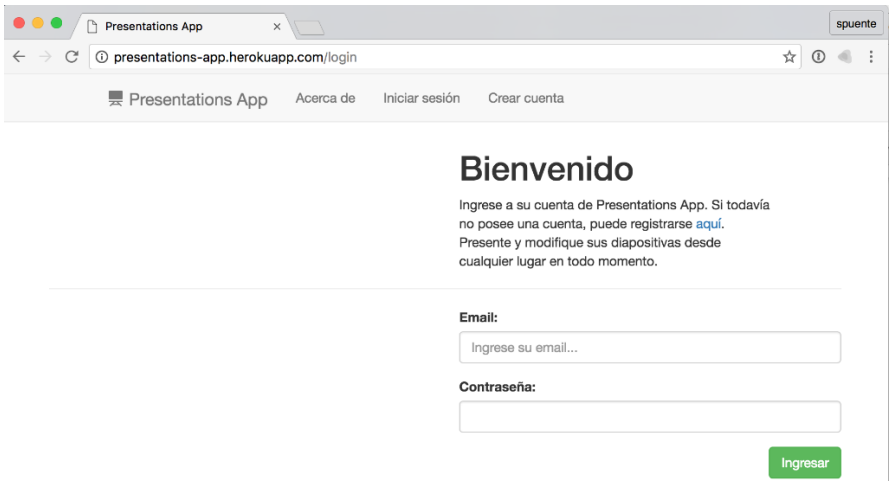
Manual de Usuario

A continuación se detallan cada una de las funcionalidades del aplicativo agrupadas por áreas de funcionalidad

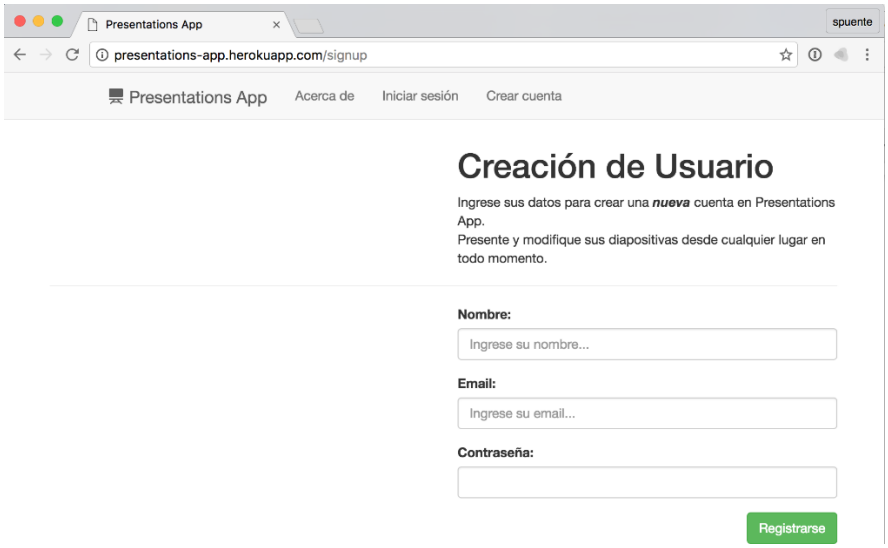
Autenticación

Crear nueva cuenta:

1. Acceder en el navegador a http://presentations-app.herokuapp.com/

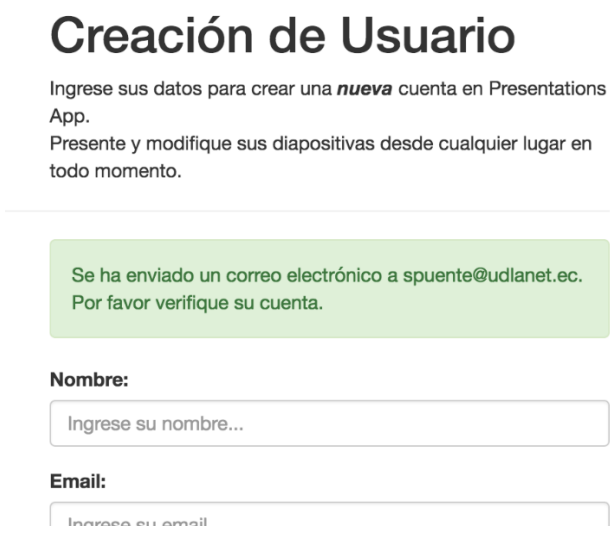


2. Seleccionar el botón **Crear cuenta.** Se mostrará la siguiente pantalla:

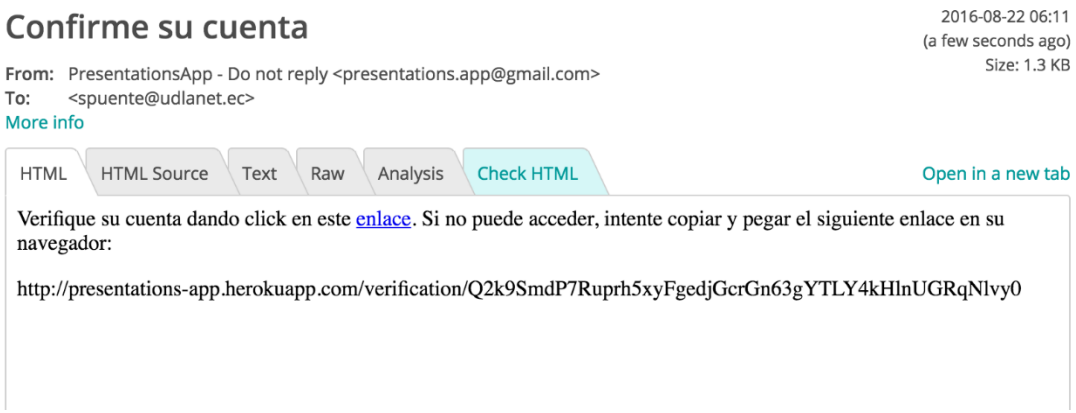


3. Ingresar los datos solicitados. Tomar en cuenta las siguientes validaciones:
    a. Se debe ingresar un email válido, ya que el sistema enviará un correo de confirmación a esa dirección.
    b. La contraseña debe tener al menos 8 caracteres.

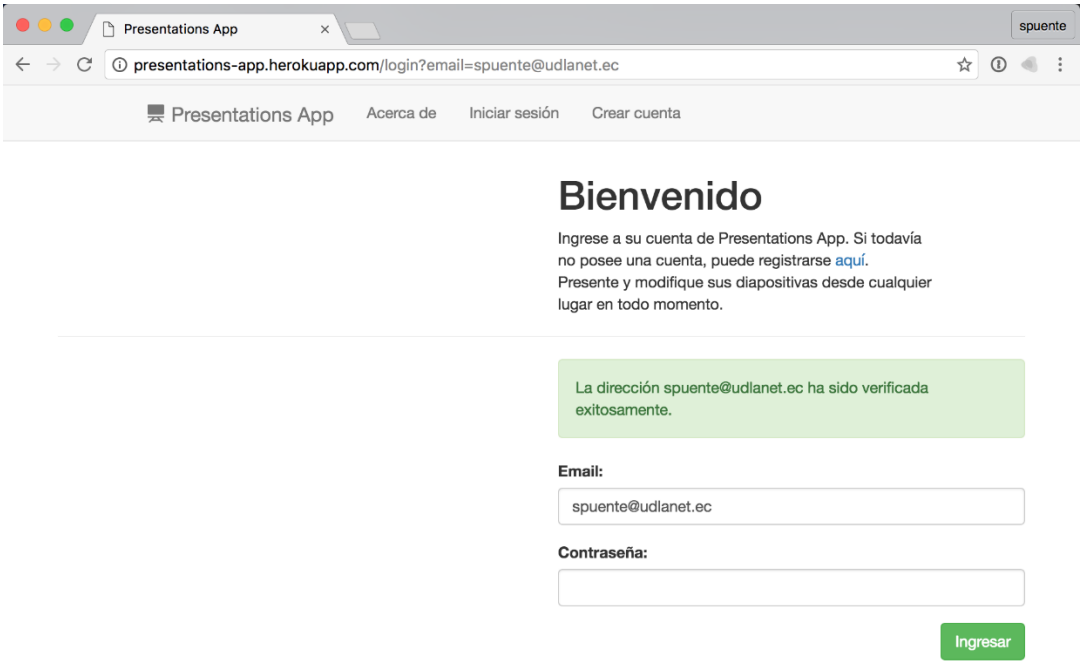
4. Dar click en el botón **Registrarse**. Se mostrará el siguiente mensaje:



5. Acceder al correo electrónico proporcionado en el formulario y verificar que se ha recibido un email de confirmación como este:



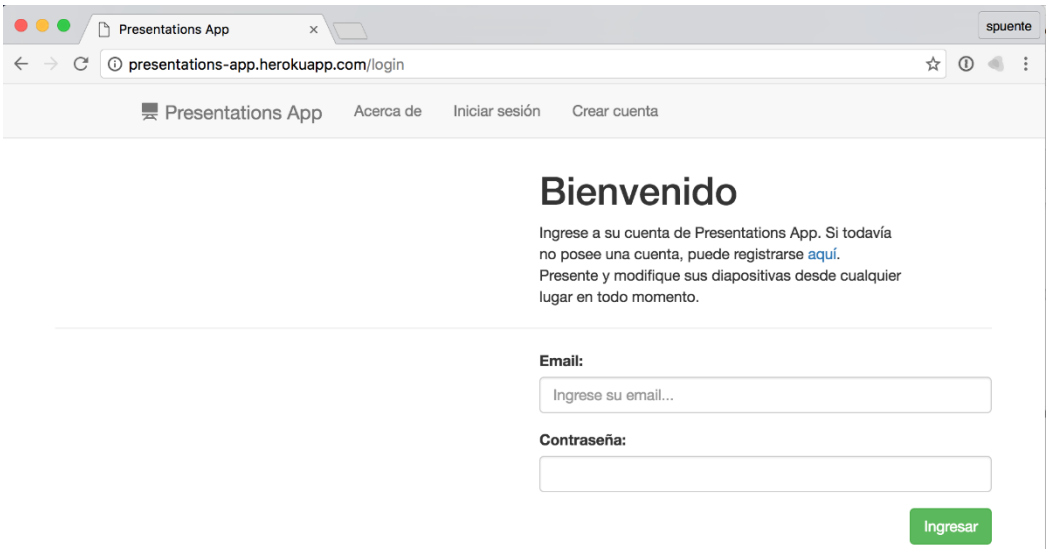
6. Dar click en el **enlace** o copiar y pegar la **URL** en el navegador. Se mostrará la siguiente pantalla:



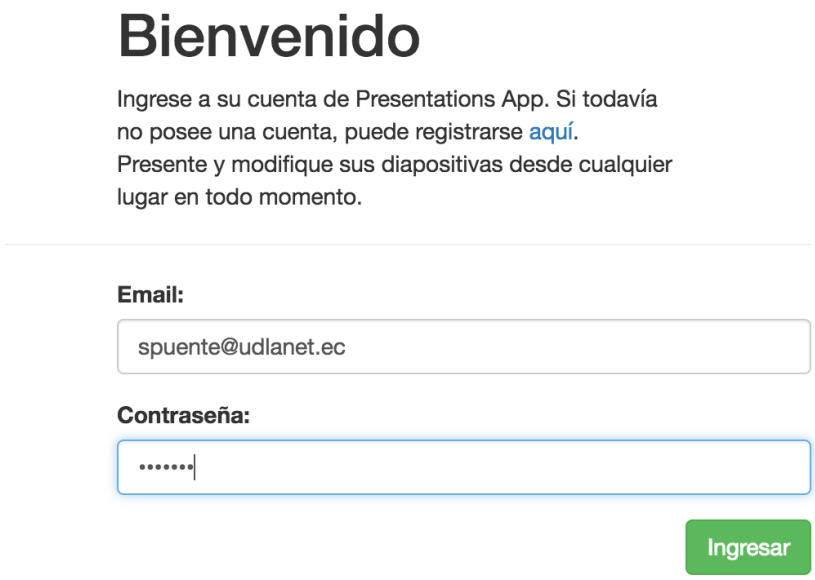
7. La cuenta ha sido exitosamente creada. Para ingresar, proporcionar la contraseña y dar click en **Ingresar**. Se mostrará la página principal de su cuenta:

Iniciar sesión:

1. Acceder en el navegador a http://presentations-app.herokuapp.com/ o si se encuentra en alguna otra página dentro de la aplicación, puede presionar el botón Iniciar sesión de la barra de navegación superior.



2. Llenar los datos de email y contraseña con que se registró en la aplicación:

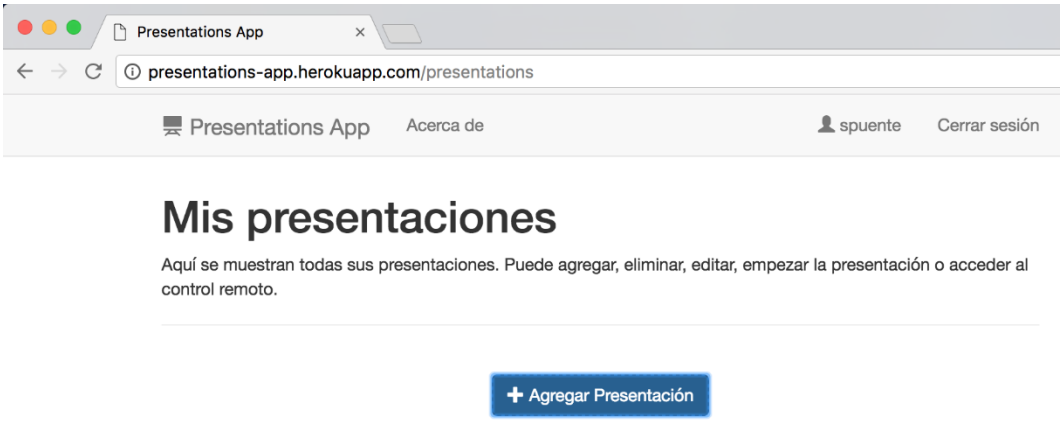


3. Presionar el botón **Ingresar**. Se mostrará la página principal de su cuenta:

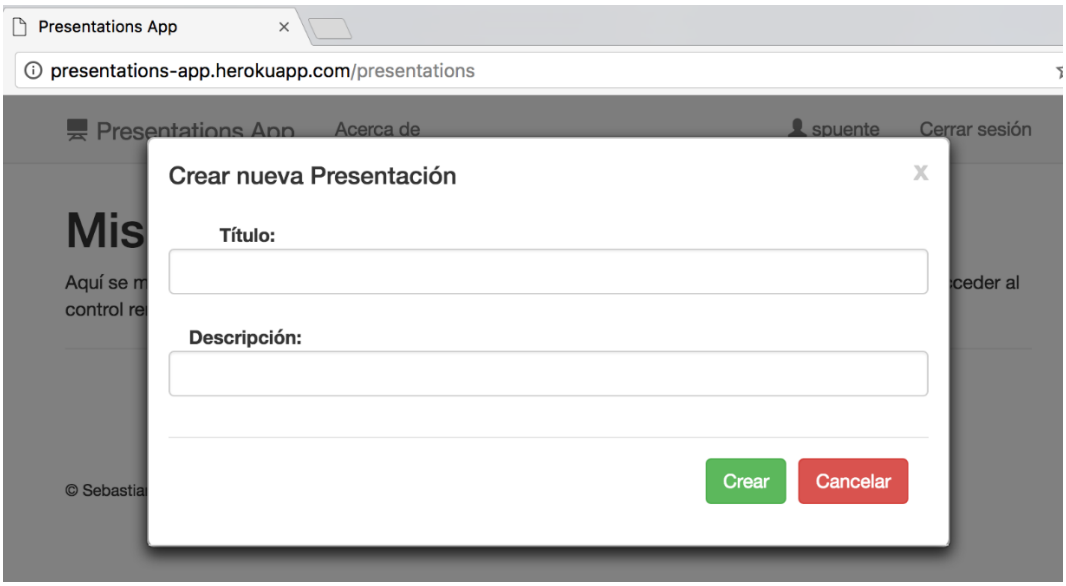
Gestión de presentaciones

Crear nueva presentación:

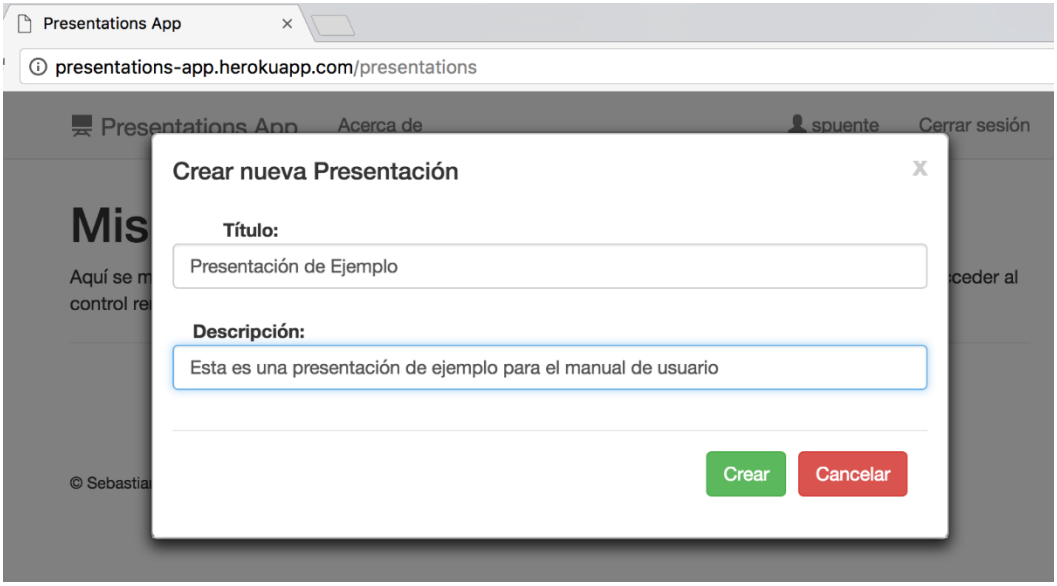
1. Para crear una nueva presentación, una vez que haya iniciado sesión, acceder a http://presentations-app.herokuapp.com/



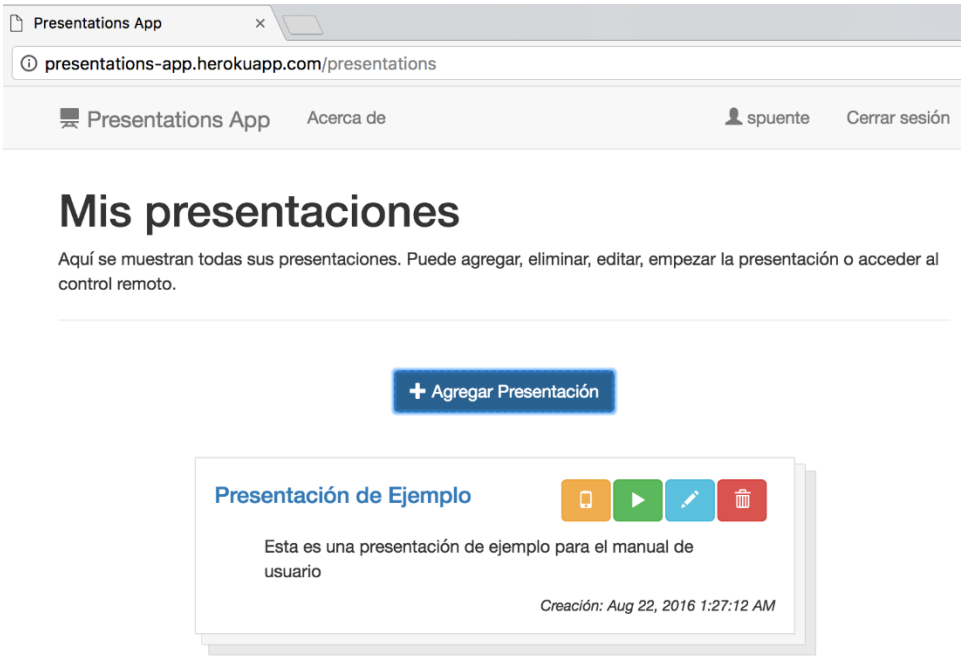
2. Al presionar el botón **Agregar Presentación** se mostrará el siguiente diálogo:



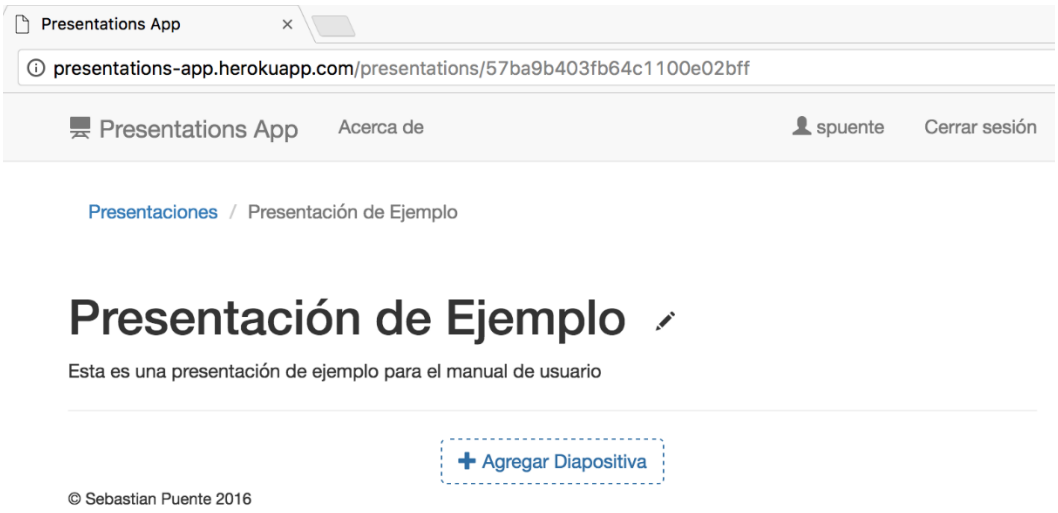
3. Llenar los datos de **Título** y **Descripción** de la presentación:



4. Presionar el botón **Crear**. Si la presentación fue creada exitosamente, se desplegará en pantalla una lista de todas sus presentaciones, incluyendo la recién creada y una lista de botones que tienen acciones que se pueden ejecutar en esa presentación:



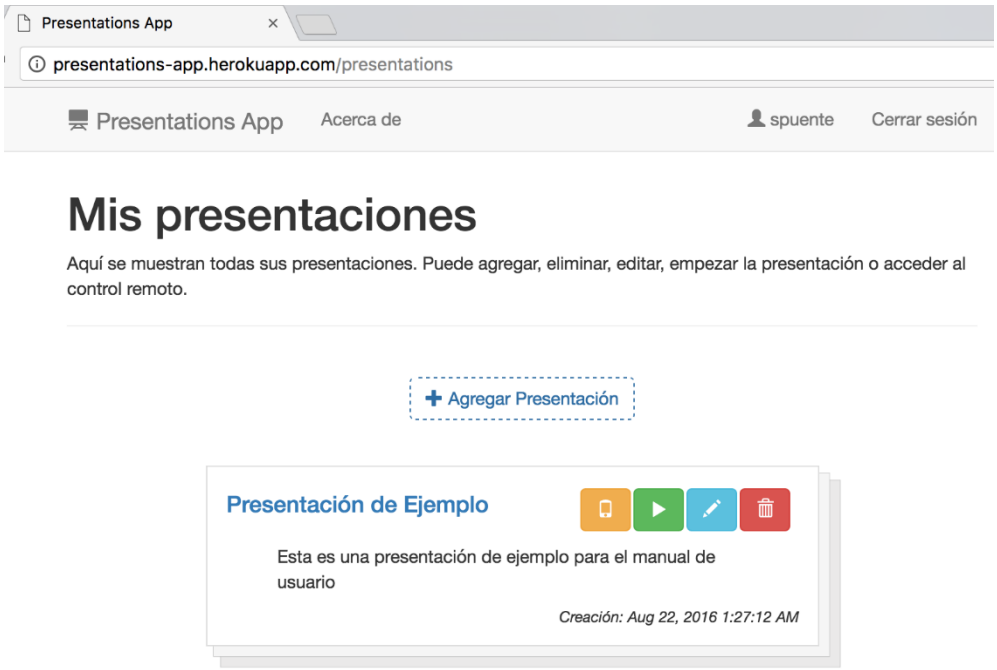
5. Para ver todo el contenido que almacena la presentación, seleccionar el **título** de la presentación o el botón celeste con ícono de **lápiz**, el cual representa edición:



6. Por el momento la presentación está vacía. El siguiente deberá ser **Crear diapositivas**, que se encuentra en la sección **Gestión de diapositivas** del manual.

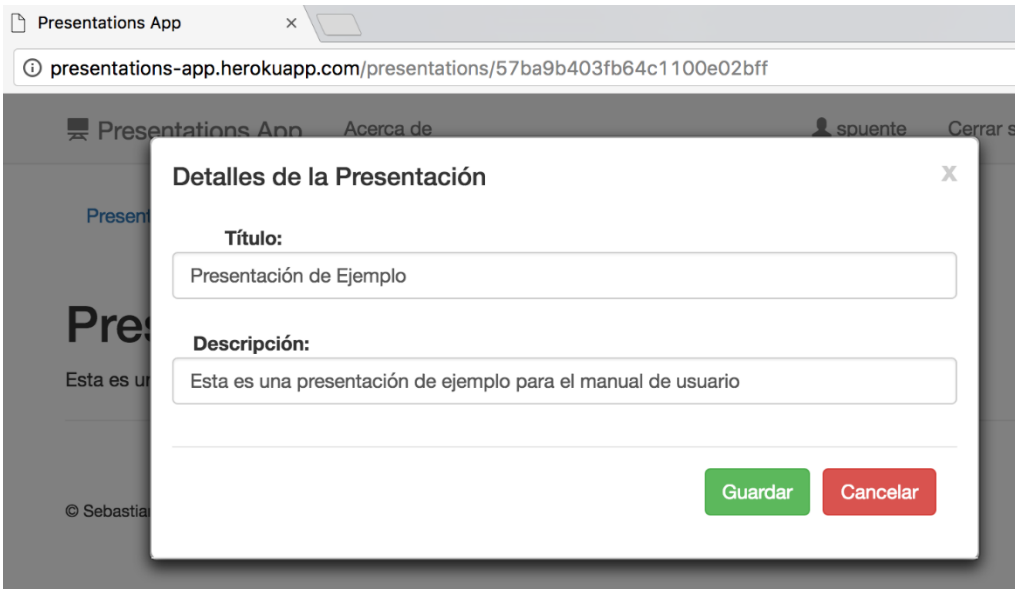
Editar presentación:

1. Para editar una presentación existente, una vez que haya iniciado sesión, acceder a http://presentations-app.herokuapp.com/

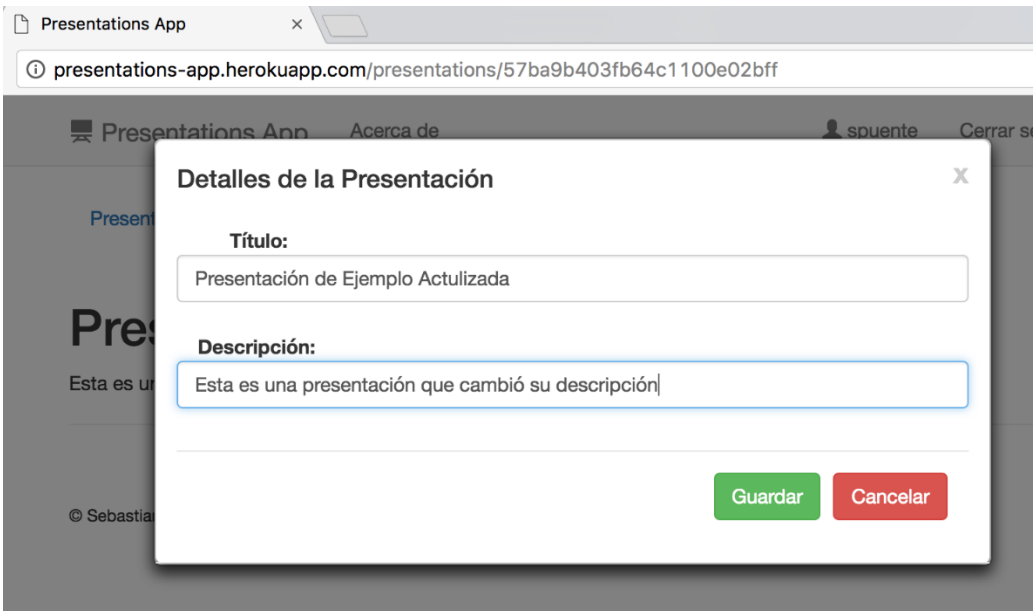


2. Dar click en el **título** de la presentación o en el botón celeste con ícono de **lápiz**, el cual representa edición. Se mostrará la siguiente pantalla:

3. Desde aquí es posible editar el título y descripción de la presentación. Presionar el botón con ícono de **lápiz** que se encuentra a la derecha del título de la presentación. Se mostrará el siguiente diálogo:



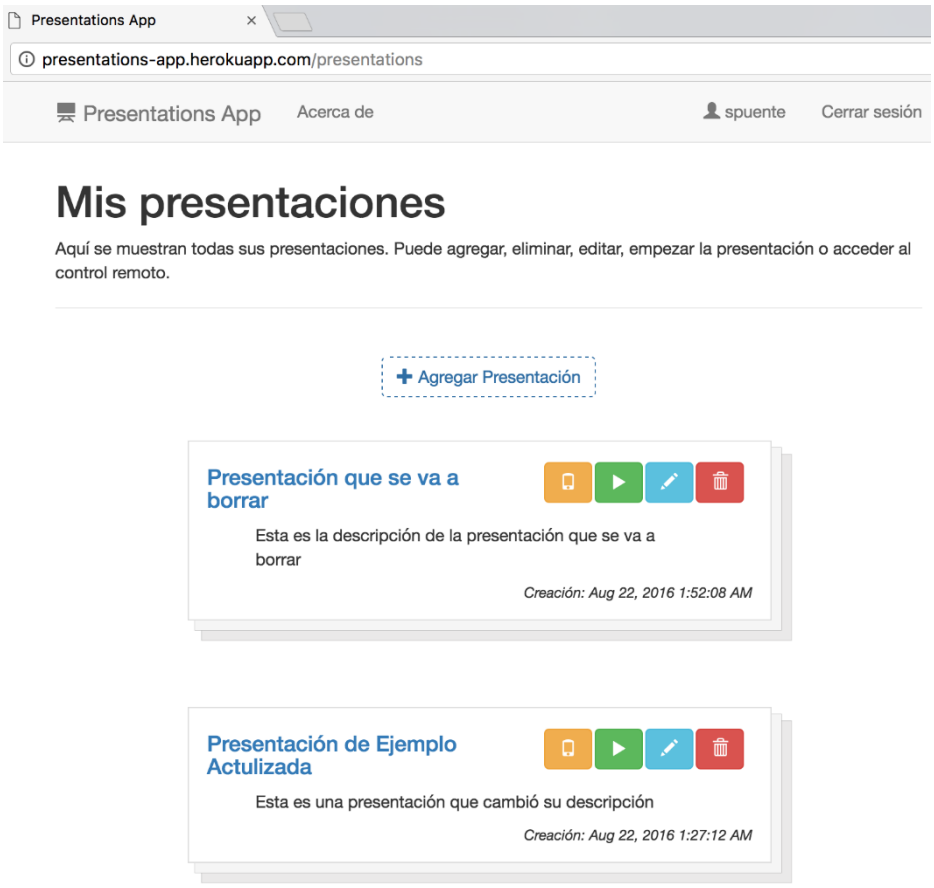
4. Realizar los cambios deseados en el **título** y la **descripción**



5. Presionar el botón **Guardar**. Se regresará a la página de la presentación, en la cual se visualizarán el título y descripción con los valores que se actualizaron.

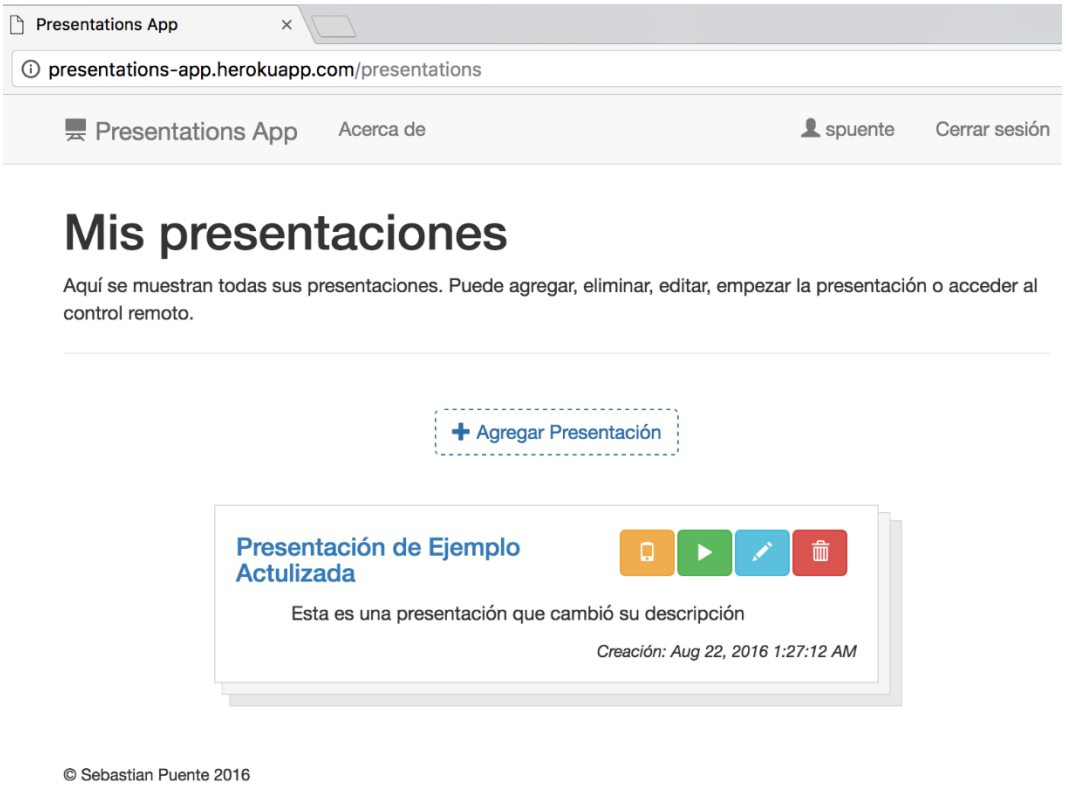
Eliminar presentación:

1. Para eliminar una presentación, una vez que haya iniciado sesión, acceder a la página principal http://presentations-app.herokuapp.com/



2. Dar click en el botón rojo con un **tacho de basura** que se encuentra junto a la presentación que se desea eliminar. En este caso se desea eliminar la primera presentación. Aparecerá el siguiente diálogo:

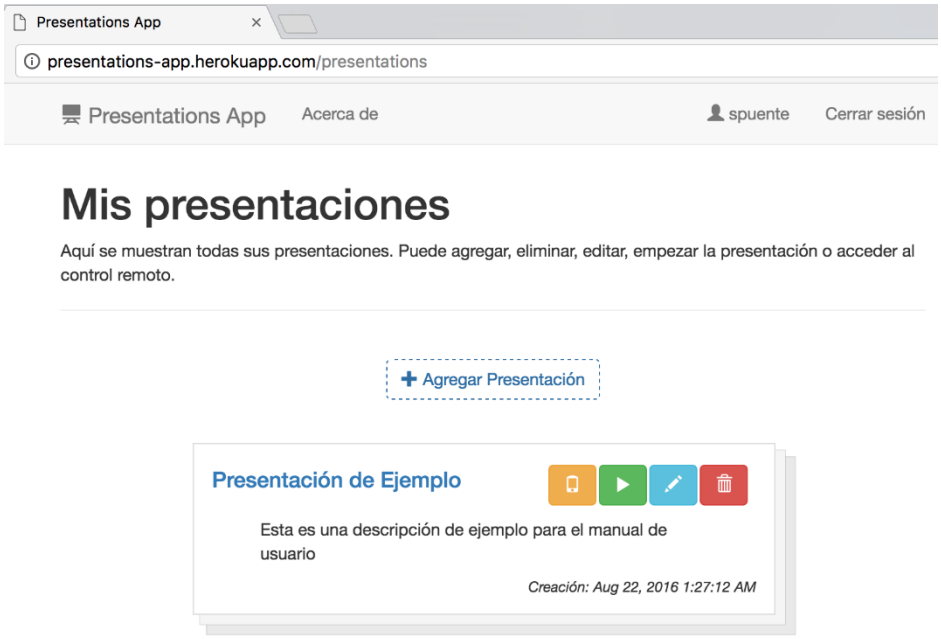
3. Presionar el botón **Eliminar**. Se mostrará la lista de todas las presentaciones del usuario y ya no aparecerá la presentación que fue eliminada.



Gestión de diapositivas

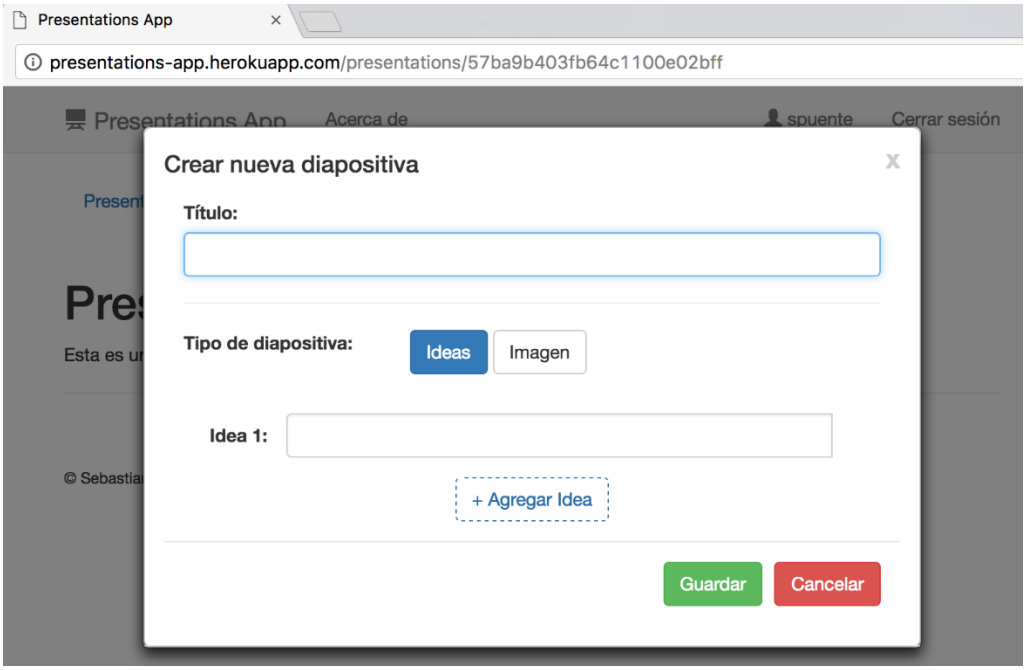
Crear nueva diapositiva:

1. Para crear diapositivas, una vez que haya iniciado sesión, acceder a la página principal http://presentations-app.herokuapp.com/



2. Escoger la diapositiva a la cual se desea agregar diapositivas. Presionar el **título** de la presentación o el botón celeste con ícono de **lápiz**. Se mostrará la pantalla:

3. Presionar el botón de **Agregar Diapositiva**. Se mostrará el siguiente diálogo:

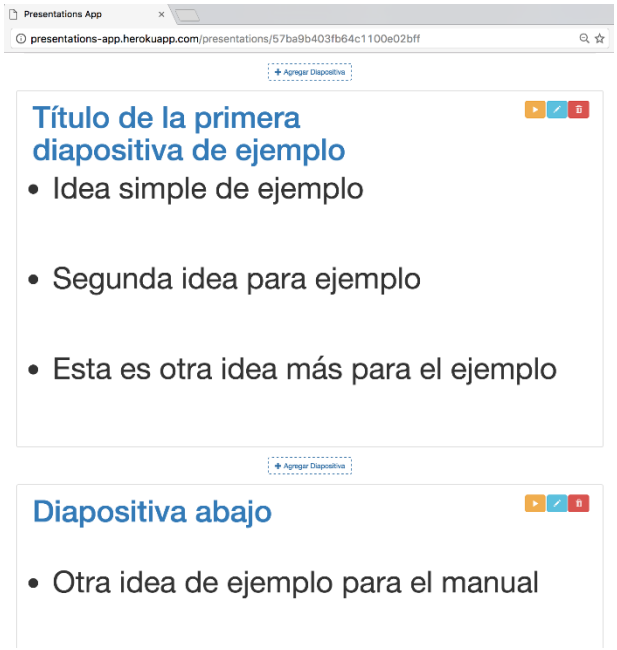


4. Ingresar el **título** e **idea** que tendrá la diapositiva.

5. Al presionar el botón **Agregar Idea** se puede agregar más campos para ideas. Se puede tener entre 1 y 5 ideas en una diapositiva.

6. Al presionar el botón **Guardar**, se almacenará la diapositiva y se mostrará la pantalla de la presentación con todas las diapositivas que contiene:

7. Es posible agregar otra diapositiva utilizando los botones de Agregar Diapositiva y repitiendo los **pasos 4 al 6**. La nueva diapositiva se ubicará en la posición en que se encuentra el botón **Agregar Diapositiva** seleccionado. Para este ejemplo se agregó una nueva diapositiva debajo de la anteriormente creada:



8. Para agregar una diapositiva con **imagen**, de igual manera se debe seleccionar el botón **Agregar Diapositiva** y luego el botón **Imagen** que aparece en el diálogo:

9. Ingresar la información de **título** y la **URL** de la cual se cargará la imagen:

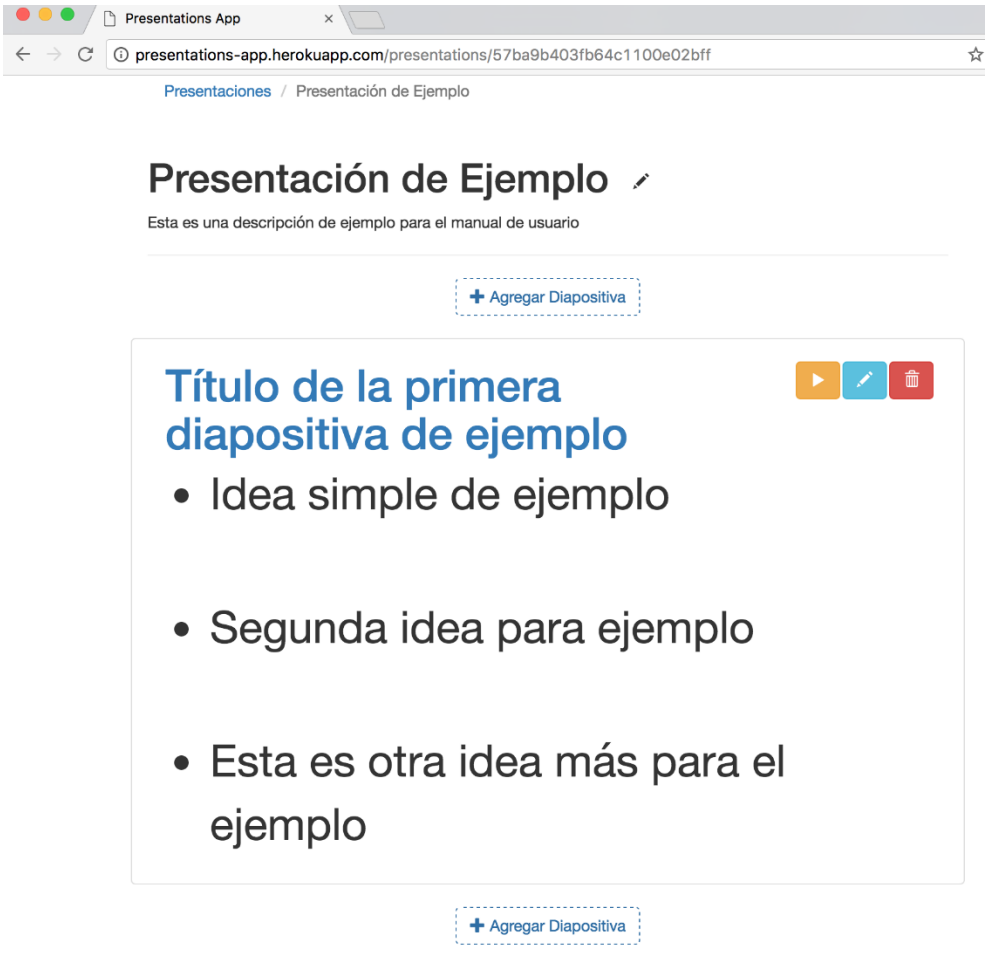
10. Presionar el botón **Guardar** y la nueva diapositiva será agregada y se mostrará junto al resto de diapositivas de la presentación:

Editar y eliminar diapositivas:

- Para editar diapositivas, se pueden seguir los mismos pasos que se utilizaron para crearlas, con la única diferencia que en vez de dar click sobre Agregar Diapositiva, se deberá seleccionar el **botón celeste con ícono de lápiz** que se encuentra al lado de la diapositiva que se desea editar:

- Para eliminar diapositivas, se deberá dar click sobre el **botón rojo con tacho de basura.** Posteriormente se debe aceptar la confirmación para eliminar la diapositiva.

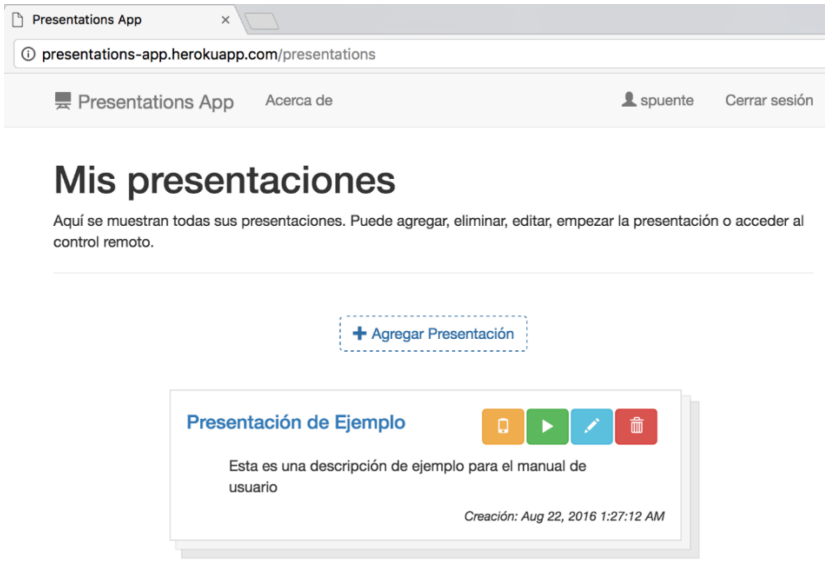
- A continuación se muestra la página de edición de una presentación con una diapositiva de ejemplo, la cual podrá ser editada o eliminada al seleccionar los botones con íconos que aparecen en la parte superior derecha.



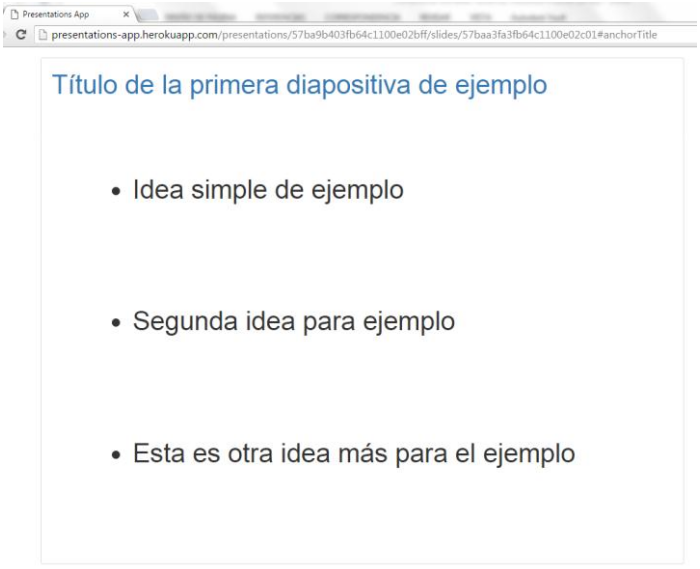
Presentación y Control Remoto

Iniciar una presentación:

1. Para iniciar una presentación, una vez que haya iniciado sesión, acceder a la página principal http://presentations-app.herokuapp.com/

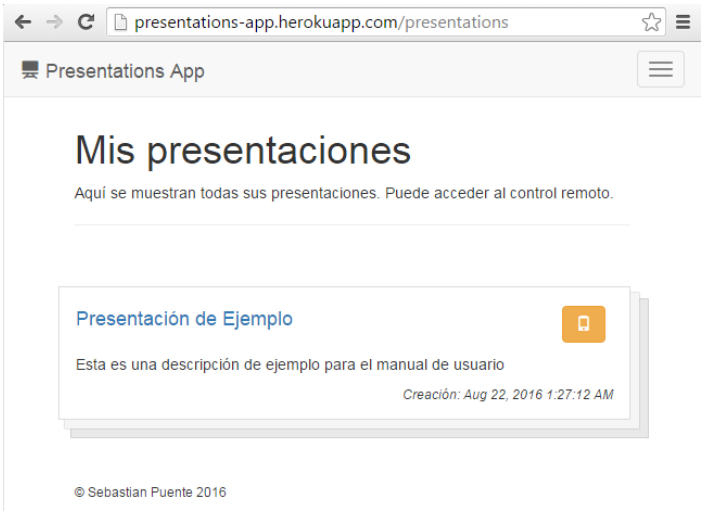


2. Escoger el ícono que tiene un **triángulo verde** apuntado hacia la derecha. Se mostrará la primera diapositiva en pantalla completa

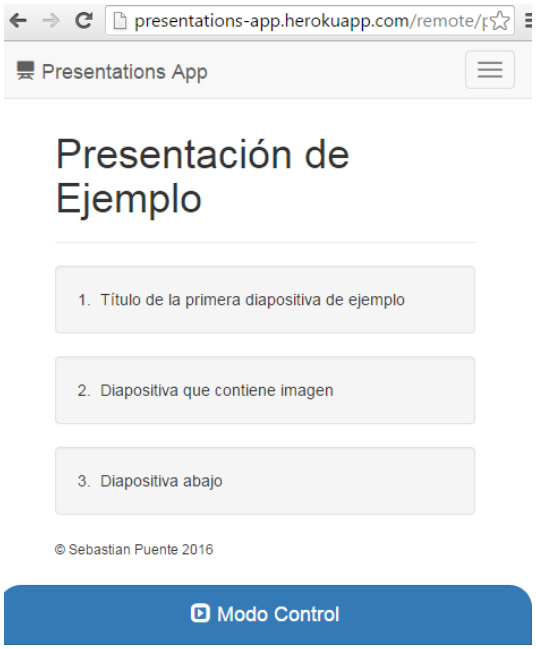


Control Remoto:

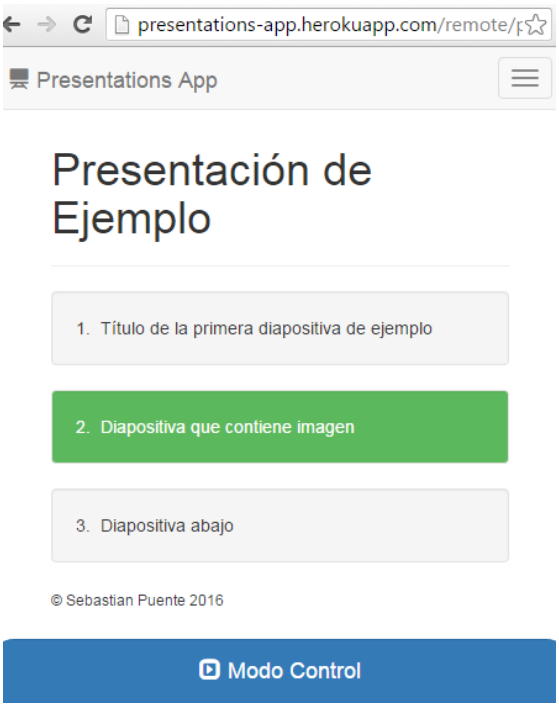
1. Para cambiar de diapositivas utilizando el control remoto, una vez que haya iniciado sesión, acceder a la página principal http://presentations-app.herokuapp.com/ desde su teléfono



2. Escoger el **ícono amarillo** con una imagen de teléfono móvil o dar click sobre el título de la presentación. Se mostrará la siguiente pantalla:



3. Seleccionar la diapositiva a la cual se desea que la presentación cambie.



4. Presionar el botón de **Modo Control** para ir al control remoto, desde ahí podrá adelantar, retroceder o ir al inicio de la presentación.