Resumen del Proyecto Fin de Carrera

[1]Sergio Ruiz Piulestán, [2]Iván Ruiz Rube

[1]Avenida Segunda Aguada n13, Cádiz,
Tlfno. 686406658,
Correo: sergioruiz5@gmail.com

[2]Escuela Superior de Ingeniería
Avenida de la Universidad nº 10
11519 Puerto Real
Cádiz

Extracto
jUCAbox es una herramienta web, orientada a la gestión de playlist para locales con ambiente musical. Los usuarios de la aplicación, podrán realizar peticiones de canciones a la persona encargada de la música del local, para una posterior validación de la misma.
jUCAbox pretende ser una herramienta social, que sirva como estudio de mercado de las canciones y artistas más solicitadas, para adecuar la música del local, dependiendo de los gustos de los usuarios allí presentes. La aplicación ha sido desarrollada con el conjuntos de herramientas propuestas por MEAN Stack, que son MongoDB, Express, Angular2 y Nodejs.

Palabras Clave: Mean Stack, Angular 2, MongoDB, Nodejs, jukebox.

1. Motivación y contexto

1.1. Introducción y motivación

En la actualidad, muchos locales tienen sistemas de reproducción musical. Cuando un cliente quiere solicitar alguna canción especial, tiene que ir al encargado de la música y decirle el nombre de la canción y el artista que desea. Conllevaría la interrupción de la labor desempeñada por el encargado de la música, así como la dificultad por el ruido y el ambiente en el local de poder transmitir que canción desea.

Por otro lado, está creciendo la necesidad de solicitar canciones antes de realizar un evento social, como una boda, una comunión, o simplemente una fiesta entre amigos. Por lo que el organizador, debería pedir, mediante correos, mensajes y/o personalmente, las canciones que le gustaría que sonaran en el evento.

La herramienta jUCAbox es un Jukebox social, una aplicación pensada para pubs, discotecas y eventos sociales.

Con jUCAbox, se pretende paliar estos problemas. El organizador del evento, o el encargado del sistema musical del local, registrarían su evento/local en la herramienta y automáticamente está disponible para que los usuarios puedan solicitar/registrar las canciones deseadas. Evitando el tráfico de información que pueda ser susceptible de perderse.

Se podrá interactuar con el DJ en tiempo real para solicitar canciones, intercambiar mensajes con otros usuarios de la herramienta, e incluso, el propio local podrá añadir promociones para favorecer a que el cliente use la herramienta.

A su vez, se podrán enviar canciones a una lista de reproducción futura, no teniendo que ser en el momento del evento, para su posterior validación. Este listado de canciones, son todas las disponibles en el ámbito de Spotify, ya que la herramienta se conecta a través de un API proporcionado por la misma.

Las motivaciones para realizar esta aplicación han sido diversas.
Conocer nuevos frameworks de desarrollo web, nuevos modelos de datos no convencionales, como son los modelos de base de datos no relaciones, y poder explotar toda esa información mediante API Rest full.

El sistema utilizado se define como SPA, Single Page Application, por el cual la navegación es muy fluida y la actualización de los datos es en tiempo real. Por el cual, es muy importante para el contexto en el que se basa la herramienta. Todo sistema con interacción social requiere de tiempos de respuesta muy cortos, uno de los puntos fuertes de los frameworks utilizados.

La principal motivación de este proyecto es poner en práctica los conocimientos adquiridos a lo largo de las asignaturas de la Titulación de Ingeniería en Informática así como conocer nuevos lenguajes de programación que nos permita ampliar nuestros conocimientos dentro del abanico de posibilidades que nos ofrece la informática hoy en día y aprender a desenvolverse en situaciones desconocidas hasta el momento por uno mismo.

1.2. Objetivos

El principal objetivo que se busca conseguir es el de proporcionar al cliente la posibilidad de solicitar una canción sin tener que moverse por todo el local. Y a su vez, facilitar al responsable de la música los gustos y preferencia de las personas que se encuentren en él.

Se intenta dar, a su vez, la posibilidad de crear listas de reproducción por un único canal, para que el organizador del evento sepa que música poner antes de la celebración del mismo.

Un objetivo colateral del resto, es hacer sentir al cliente escuchado, pudiendo tomar parte en la reproducción de canciones en el local.

En una primera fase se analizó los requisitos que debería cumplir una aplicación para que pudiera enviar y recibir canciones.

En primera instancia se pensó en crear una NAS de alojamiento de ficheros de audio. Un catálogo de canciones para poder relacionar esos ficheros con el artista, nombre de la canción etc…

En segundo lugar, para no tener que litigiar con derechos de autor, se pensó en utilizar herramientas de mercado que dieran conexión a terceros.

Entre las más utilizadas por la sociedad estaban, Spotify, Napster y SoundCloud.

Las herramientas ponen al servicio de terceros unas API de solicitud de información de datos que manejan.

Para el desarrollo de aplicación se pensó en realizar una herramienta móvil para la utilización del usuario y otra herramienta de escritorio para comunicar las canciones solicitadas.

La otra opción es realizar una Web responsive para el uso tanto de móvil como desde pc. Para el desarrollo de esta herramienta se barajó utilizar frameworks como Symphony2, basados en php, ASP .NET MVC o el conjunto de herramientas de Mean Stack.

La solución propuesta finalmente fue realizar una herramienta con conexión a Spotify, para la obtención de datos a través de su API. Para el desarrollo de la herramienta se realizaría con las herramientas propuestas por el MEAN stack, compuesto por MongoDB[7] y Express para los datos, Angular 2[4] para el Front y Node.js[5] para el Backend, puesto que son frameworks que se prevén que van a tener una gran comunidad de desarrolladores detrás, facilitando su mantenimiento futuro.

Además este paquete de herramientas, ofrece de manera implícita, el desarrollo responsive, por lo que el desarrollo será más sencillo. En un futuro si se quiere implementar una aplicación móvil, existe un componente llamado IONIC 2, que transforma el código de Angular 2 en una aplicación nativa de móvil, tanto para IOS como Android.

2. Planificación

La metodología utilizada para la elaboración del proyecto se basa en el Racional Unified Process (RUP) [1-2] que traducido es proceso racional unificado. Este proceso de desarrollo software unido al lenguaje unificado de modelado (UML) [3] componen la metodología habitual estándar para el análisis, implementación y documentación de sistemas orientados a objetos.

La planificación llevada para la realización del proyecto ha sido:

Iteración 0 – Elicitación de requisitos y prototipado

- Se realiza un estudio de los requisitos necesarios para elaborar el sistema. Se realizan varias reuniones con clientes y usuarios potenciales, elaborando una serie de condiciones para que la herramienta sea lo más funcional posible.
- Se realiza un prototipo de la herramienta, organizándose en bloques para su consiguiente desarrollo.

Iteración 1 – Usuarios/Roles

- Se desarrolla un sistema de autenticación y perfilado de la herramienta. Asignando una función a cada uno de los perfiles dados de alta.

Iteración 2 – Búsqueda de Canciones

- Se crea la conexión con la API de Spotify. Pudiendo realizar conexiones y extraer y parsear la información para que esté disponible en la herramienta en tiempo real.

Iteración 3 – Lugares

- Se crea un menú de administración para crear y editar distintos lugares, locales y eventos para que estén disponibles en la herramienta.
- Se crea el módulo de búsqueda de lugares, habilitando la posibilidad de que envíen canciones ya buscadas a un lugar concreto.
- Se crea el módulo de favoritos, añadiendo los usuarios con un local en concreto.

Iteración 4 – Envío y Reproducción de canciones

- Se crea la conexión entre la canción y el lugar, habilitando la validación de las mismas.
- Se crea la conexión de Spotify con el local. Añadiendo las canciones en tiempo real a Spotify
- Se desarrolla un módulo para la reproducción de canciones en Spotify directamente desde la herramienta con actualizaciones en tiempo real.

Iteración 5 – Mensajería

- Se crea un módulo para publicar mensajes en el tablón. Dando la opción al administrador del lugar de enviar un mensaje a los usuarios que tienen agregado al local como favorito.

Iteración 6 – Reporting

- Se desarrollan una serie de informes para que el administrador del local pueda tener constancia de las peticiones que recibe, así como el género de las mismas, para adecuar la música al gusto de los clientes.

3. Análisis

3.1. Requisitos

A continuación se describen los requisitos funcionales y no funcionales del sistema, para conocer las características que debe cumplir la aplicación una vez finalizada.

Requisitos funcionales

En esta sección se van a describir los requisitos funcionales del sistema. Es necesario recalcar, que el perfil de usuario, se comportaría como administrador en el caso de gestionar algún lugar. Se puede acceder a la herramienta sin ser un usuario registrado. Siempre que se hable de usuario, se hablará de usuario registrado. Si el usuario no está registrado se indicará como invitado.

- **Se deberá tener un apartado para el registro de usuarios**
    - El invitado se podrá registrar con Google, Facebook, Twitter o registro interno
    - El usuario podrá buscar a usuarios y agregarlos como amigo
    - El usuario podrá ver los datos de otros usuarios si los agrega como amigos
    - El usuario podrá modificar sus datos
    - El usuario verá un log de actividad
    - El usuario podrá chatear con otros usuarios si son amigos
- **Se deberá tener un apartado para la gestión de lugares**
    - El usuario podrá crear lugares. Pasará a ser administrador para ese lugar.
    - El usuario podrá ver lugares agregados como favoritos

- El usuario/invitado podrá buscar lugares
- El administrador podrá publicar mensajes de sus lugares
- El administrador podrá validar o rechazar las canciones enviadas
- El administrador podrá crear distintas playlist asociadas al lugar
- El administrador podrá reproducir la lista de canciones desde la herramienta

- **Se deberá tener un apartado de búsqueda de canciones y artistas**
  - El usuario/invitado podrá buscar cualquier canción/artista de Spotify
  - El usuario/invitado podrá enviar una canción a cualquier lugar registrado en la herramienta
  - El usuario podrá agregar como favorito a artistas

Requisitos no funcionales

La aplicación deberá cumplir los requisitos no funcionales listados a continuación:

- **Disponibilidad**
  - La aplicación debe estar disponible las 24 horas del día, los 365 días del año. Para garantizar una mayor experiencia al usuario.
- **Usabilidad**
  - La herramienta debe ser de fácil uso. No restringiendo solo a personas con alto conocimiento tecnológico, sino de sencilla comprensión, y con funcionalidades intuitivas.
- **Portabilidad**
  - La aplicación debe poder usarse en cualquier dispositivo,

independientemente del sistema operativo usado y/o el navegador.

- **Seguridad**
    - La herramienta ante todo tendrá implementada una capa de seguridad, en la que impediría a cualquier usuario externo acceder a datos internos. Será necesario evitar cualquier ataque a los datos, estableciendo unas medidas extremas en el lado del servidor.
- **Interfaz**
    - La interfaz de usuario debe ser sencilla a la vez que completa. Pudiendo permitir al usuario de forma ágil y veloz acometer su acción de la forma más sencilla posible. Siendo, a su vez, una interfaz atractiva y correctamente consultable desde cualquier dispositivo.
- **Mantenibilidad**
    - La herramienta podrá ser mantenida por cualquier desarrollador previamente formado en los frameworks utilizados. Por lo que es necesario mantener unas buenas prácticas a la hora de desarrollar la aplicación, evitando duplicidad de código o creando funcionalidades demasiado complejas, pudiendo desglosarse en varias.
- **Rendimiento**
    - Al ser una herramienta con conexión de datos con sistemas terceros, los tiempos de respuestas deben de ser cortos, implementando la transmisión asíncrona de datos siempre que sea posible.

- **Fiabilidad**
    - La aplicación debe de ser lo suficientemente robusta para evitar errores en su ejecución.

Requisitos de información

En esta sección se describen los requisitos de información que debe cumplir la herramienta.

- Se almacenará la dirección de los lugares a registrar. Para poder realizar búsquedas dependiendo de la ciudad o provincia.
- Se almacenarán los datos referentes a las canciones enviadas al local. Será necesario para la explotación a futuro de los datos, conocer la temática o autores más relevantes.
- Será necesario guardar en un log de actividad toda la actividad de los usuarios. En una siguiente fase se podrá hacer un estudio de mercado con la información obtenida.

3.2. Perfiles

Para poder utilizar el sistema, se ha pensado en tres perfiles.

- Invitado:
    - Son las personas que acceden a la herramienta sin registro previo. Podrán utilizar la parte esencial de la aplicación, como es enviar canciones. Pero no podrán añadir como favoritos lugares ni artistas, ni tampoco conocer los mensajes publicados de los lugares. Podrán ver los lugares y canciones enviadas, a su vez podrán filtrar el top de canciones del lugar entre fechas.

- Usuario:
    - Son las personas que acceden a la herramienta habiéndose registrados. Podrán realizar todas las funciones descritas para el invitado. Además de añadir lugares y artistas como favoritos. Intercambiar mensajes con otros usuarios, ver mensajes publicados por los lugares que tenga añadido como favorito. Así como crear nuevos lugares en la aplicación.
- Administrador:
    - Son las personas que acceden a la herramienta habiéndose registrado y que gestionan algún lugar. Podrán hacer las mismas gestiones que los usuarios además de validar las canciones enviadas y publicar mensajes en el tablón del lugar.

4. Diseño y Construcción del Sistema

4.1. Arquitectura Lógica

Para la arquitectura de la aplicación web se utilizó el patrón de diseño MVC (Modelo – Vista – Controlador) pues es en el que se basa MEAN Stack [6]. En este patrón de diseño, se separa la lógica de negocio, de la lógica visual. El patrón de diseño se divide en tres partes anteriormente mencionadas:

- Modelo
    - Es el encargado de controlar los datos, así como su acceso y modificación. En el cual se implementan los roles de acceso y permisos para dichas consultas. En nuestra aplicación el encargado es MongoDB, que a través de Express.js y Node.js, hacen que el acceso a los datos sean seguro.

- Vista
    - Es el encargado de renderizar los datos proporcionado por el controlador, es la capa visual que puede ver y utilizar el usuario. Es el que muestra la interfaz de usuario. En nuestra aplicación es controlado por Angular2, mediante el ViewModel.
- Controlador
    - Es el encargado de invocar peticiones y responde a eventos cuando se solicita una petición a través de la vista. Esta función la cubre también Angular2, a través de su interacción con los Templates de html5.

4.2. Arquitectura Física

En el lado del servidor, vamos a necesitar una infraestructura necesaria para albergar MongoDB 3.2.11, que será nuestro sistema de base de datos. A su vez debe tener instaladas las dependencias de Express 4.15.13 para poder comunicarse con Node.js 6.11.10 que a su vez incorpora el inyector de dependencias npm 3.10.10.

4.3. Construcción del Sistema

Como ya hemos comentado antes, para el desarrollo de la herramienta se utiliza Mean Stack, Las siglas MEAN [6] significan:

- MongoDB: Sistema de base de datos no relacionales basada en JSON. Utiliza una versión propia llamada BSON.
- Express: Framework encargado de conectar MongoDB con el servidor WEB. Es el responsable de gestionar la API Rest. Capaz de manejar las solicitudes y peticiones Http.
- Angular2: Framework basado en MVC utilizado para la parte Cliente de la aplicación WEB. Se utiliza para el desarrollo de páginas SPA

(Single Page Application). Se desarrolla con typescript, una versión libre de JavaScript creada por Microsoft.

- Node.js: Framework utilizado como servidor WEB. Se utiliza JavaScript para el desarrollo de las reglas del servidor.

5. Pruebas del sistema

Para el testeo de la herramienta se ha seguido con un método de pruebas de manera manual, pues la cantidad de funcionalidades de jUCAbox no requerían el desarrollo de un módulo específico para pruebas.

Al ser una herramienta orientada al mercado, y no a algún cliente en particular, se ha realizado pruebas de aceptación con distintos perfiles potenciales de la herramienta, desde un usuario básico, a un administrador de un local de música.

Para probar la dependencia entre componentes, se realizaron pruebas de integración. Cuando se hacía alguna modificación en algún componente que interactuaba con otros, se comprobaban que cumplían con las necesidades propuestas. Cuando la herramienta ya estuvo lo suficientemente madura, las pruebas se realizaban en la capa visual de la herramienta. Era más sencillo realizar las pruebas conforme se iban desarrollando funcionalidades, que buscar los fallos según la dependencia de los componentes.

6. Conclusiones

6.1. Objetivos alcanzados

Se han cumplido con todos los objetivos propuestos en la fase de análisis, profundizando aún más de lo propuesto en las funciones propuestas para la conexión de la herramienta con la API de Spotify. Además, consiguiendo que

la herramienta sea más social de lo previsto, añadiendo una interacción entre usuarios.

Se ha conseguido hacer una aplicación sencilla pero robusta, pudiendo ser utilizada fácilmente desde cualquier dispositivo con conexión a internet.

6.2. Lecciones aprendidas

En lo referente al proyecto uno de los principales objetivos ha sido aplicar los conocimientos obtenidos a lo largo de los años en la carrera de Ingeniera Informática. Gracias a estos conocimientos, se ha sido capaz de crear una herramienta desde cero, realizando paso por paso todas las funciones y procedimientos necesarios.

Además añadiendo el valor de la realización de la documentación, necesaria para todos los proyectos. Siendo útil para futuros mantenimientos y ampliaciones del software, incluso para posibles incidencias que puedan ocurrir. Por lo que resultado muy importante el aprendizaje obtenido a la hora de elaborar diagramas, casos de usos, diseño, así como la correcta redacción de un documento técnico.

Por el lado del software utilizado, ha sido muy enriquecedor aprender nuevos lenguajes de programación, así como todas las herramientas utilizadas a lo largo del proyecto. Siendo un trabajo fundamental investigar sobre nuevas técnicas y soluciones.

6.3. Trabajo futuro

Una vez concluido con los objetivos propuestos en el proyecto, para trabajos futuros sería interesante crear una aplicación móvil nativa. Aunque la herramienta se adapta de forma adecuada al tamaño de cualquier dispositivo,

es mucho más sencillo y ágil acceder directamente desde una aplicación instalada en el móvil.

De cara a nuevas funcionalidades propuestas, sería la posibilidad de crear distintos roles par un mismo lugar, es decir, varios administradores a distinto nivel, o poder crear lugares privados, que solo los usuarios puedan acceder y enviar canciones con invitaciones previas.

Referencias

[1] Philippe Kruchten, The Rational Unified Process: An Introduction (3rd Edition), Booch Jacobson Rumbuagh (1999).

[2] Grady Booch, The unified modeling language user guide, Booch Jacobson Rumbuagh (1999).

[3] Russ Miles, Kim Hamilton, Learning UML 2.0, O'Reilly (2006).

[4] Ari Lerner, Nate Murray, Felipe Coury, Carlos Taborda, Gistia, Ng-Book 2: The Complete Book on Angular 2. (2016)

[5] Nathan Rajlich, Mike Cantelon, T. J. Holowaychuk, Marc Harter, Manning, Node.js in Action (2013).

[6] Simon Holmes, Manning, Getting MEAN With Mongo, Express, Angular, and Node (2015).

[7] Tim Hawkins, Eelco Plugge, Peter Membrey, Apress, The Definitive Guide to MongoDB: A Complete Guide to Dealing with Big Data (2010).