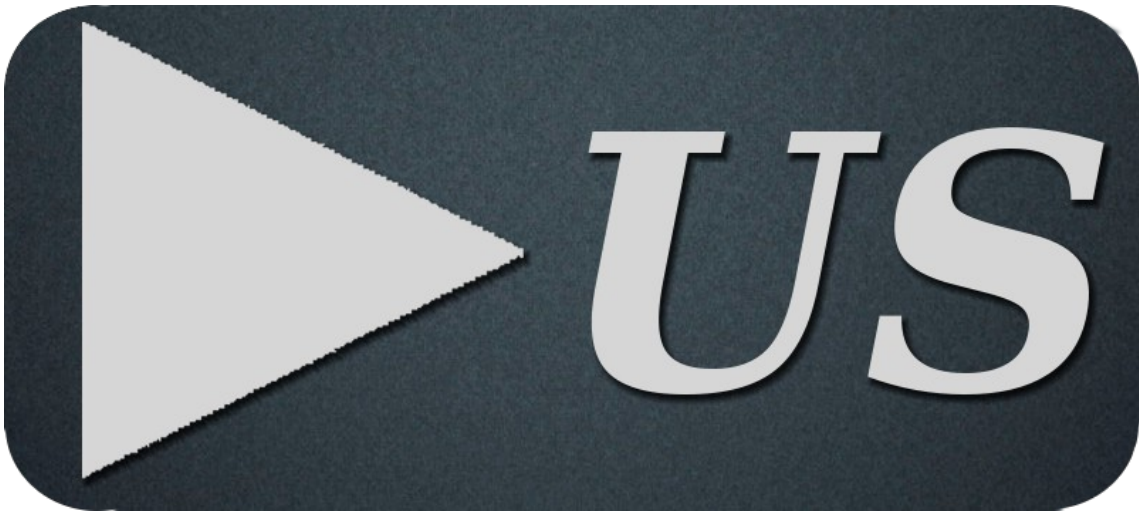
Play US



Arquitectura e Integración de Sistemas Software

Grado de Ingeniería del Software

García Sánchez, Alberto (albgarsanuni@yahoo.es)
Muñoz Moya, David (damu-cazalla@hotmail.es)
Sánchez Crespo, Juan Luis (juanlu-28a@hotmail.com)

Tutor: Ortega Rodríguez, Francisco Javier
Número de grupo: 6

Enlace de la aplicación: play-us.appspot.com
Enlace de proyecto en gitHub: https://github.com/Juanlu991/Play-US
Enlace de proyecto en Projetsii:
https://projetsii.informatica.us.es/projects/h36kfmvc9p3g2yyzyq4

Historial de versiones

Fecha	Versión	Detalles	Participantes
14/03/2014	1.0	- Incluye introducción, prototipos de las interfaces de usuario y diagramas UML de componentes y despliegue.	Muñoz Moya, David Garcia Sánchez, Alberto Sánchez Crespo, Juan Luis
25/04/2014	1.1	-Incluye introducción, prototipos de las interfaces de usuario, diagramas UML de componentes y despliegue, prototipo funcional y documentación API REST	Muñoz Moya, David Garcia Sánchez, Alberto Sánchez Crespo, Juan Luis
30/05/2014	2.0	-Se he hecho mejoras en todos lo módulos de la aplicación y ya funcionan correctamente.	Muñoz Moya, David Garcia Sánchez, Alberto Sánchez Crespo, Juan Luis

AUTOEVALUACIÓN

Alumno	Entregable 1	Entregable 2	Final
Sánchez Crespo, Juan Luis	100%	100%	100%
Muñoz Moya, David	100%	50%	100%
Garcia Sánchez, Alberto	100%	60%	80%

0% = No he podido participar y por tanto debo aceptar tener un cero en el entregable.

50% = He trabajado aproximadamente la mitad que mis compañeros y por tanto merezco la mitad de la nota del trabajo.

100% = He trabajado como el que más y merezco la nota dada al trabajo.

Índice

1 Introducción

Hay ocasiones en las que la canción que buscamos no se encuentra en nuestro reproductor de música online. Otras veces, tenemos varias canciones que nos gustan, pero hay una que no esta en nuestro reproductor preferido, por lo que no podemos hacer listas de reproducción de todas las canciones que deseamos. Algunas veces deseamos hacer un comentario en alguna red social, sobre la canción que estamos escuchando, y en ese momento no estamos en ella.

Para la solución de esos problemas y más, vamos a realizar una aplicación que englobe los 2 reproductores principales de música online y la opción de poder escuchar tu propia música con: Grooveshark, SoundCloud y las canciones que tengas en Google Drive. También vamos a conectar dos redes sociales: Twitter y Facebook. Podremos realizar listas de reproducción con canciones de diferentes reproductores, hacer comentarios mientras estamos escuchando nuestras canciones sin tener que abrir otra pestaña e incluso mencionar el título de la canción que estamos escuchando en nuestro comentario con el símbolo “|>” .

Con todo esto, agrupamos lo mejor de cada reproductor en un mismo sitio web, además con la posibilidad de realizar comentarios.

Como servicios vamos a ofrecer la posibilidad de realizar búsquedas, inserciones, modificaciones y eliminaciones de usuarios, sus listas de reproducción y las canciones que tiene la lista de reproducción que queramos. También se podrá realizar la búsqueda de canciones en servidores concretos.

1.1 Aplicaciones integradas

Grooveshark: Es una aplicación de búsqueda de música online y recomendación de la misma. También permite a los usuarios buscar y subir música de forma libre y gratuita.

SoundCloud: Es una plataforma de distribución de audio online en la que sus usuarios pueden colaborar, promocionar y distribuir sus proyectos musicales.

Google Drive: Es un servicio de alojamiento de archivos. Google Drive es un reemplazo de Google Docs que ha cambiado su dirección de enlace de docs.google.com por drive.google.com entre otras cualidades.

Facebook: Es un sitio web de redes sociales, en el que se puede dar de alta cualquier persona con una cuenta de correo electrónico.

Twitter: Es un servicio de microblogging. Ha sido apodado como el "SMS de Internet". Entre sus usuarios se destacan grandes figuras públicas.

Nombre	URL documentación API
Facebook	https://developers.facebook.com/docs/graph-api/using-graph-api/
SoundCloud	http://developers.soundcloud.com/docs/api/reference
Twitter	https://dev.twitter.com/docs/api/1.1
Google Drive	https://developers.google.com/drive/v2/reference/
Grooveshark	http://developers.grooveshark.com/docs/public_api/v3/

TABLA 1. APLICACIÓN INTEGRADAS

1.2 Evolución del proyecto

Este proyecto no ha sufrido transformaciones muy grandes a lo largo de su desarrollo.

La transformación más importante que ha sufrido este proyecto, ha sido el remplazo de Spotify por Google Drive ya que Spotify no nos permite controlar las canciones como queríamos, y teníamos que adaptarnos a su reproductor.

Respecto al resto de aplicaciones y al desarrollo de la aplicación a sufrido cambios insignificantes.

2 Prototipos de interfaz de usuario

2.1 Vista Inicio

En esta vista tenemos a nuestro alcance todos los componentes que forman nuestro mashup.

En el panel de la izquierda, podemos encontrar las canciones específicas de cada reproductor de nuestra lista de reproducción actual.

En el panel central, tenemos primero un buscador de canciones, en el que especificamos el servidor en el que queremos buscarlo. A continuación aparecería la búsqueda realizada. Debajo de la búsqueda, hayamos un recuadro, en el cual podemos introducir nuestro comentario y elegir la red social en el que vamos a publicarlo.

En el último panel, encontramos la lista de reproducción actual, además tenemos la opción de guardar nuestra lista.

En la parte inferior de nuestra pantalla, siempre encontraremos una herramienta de reproducción con la cual podemos controlar la canción actual.

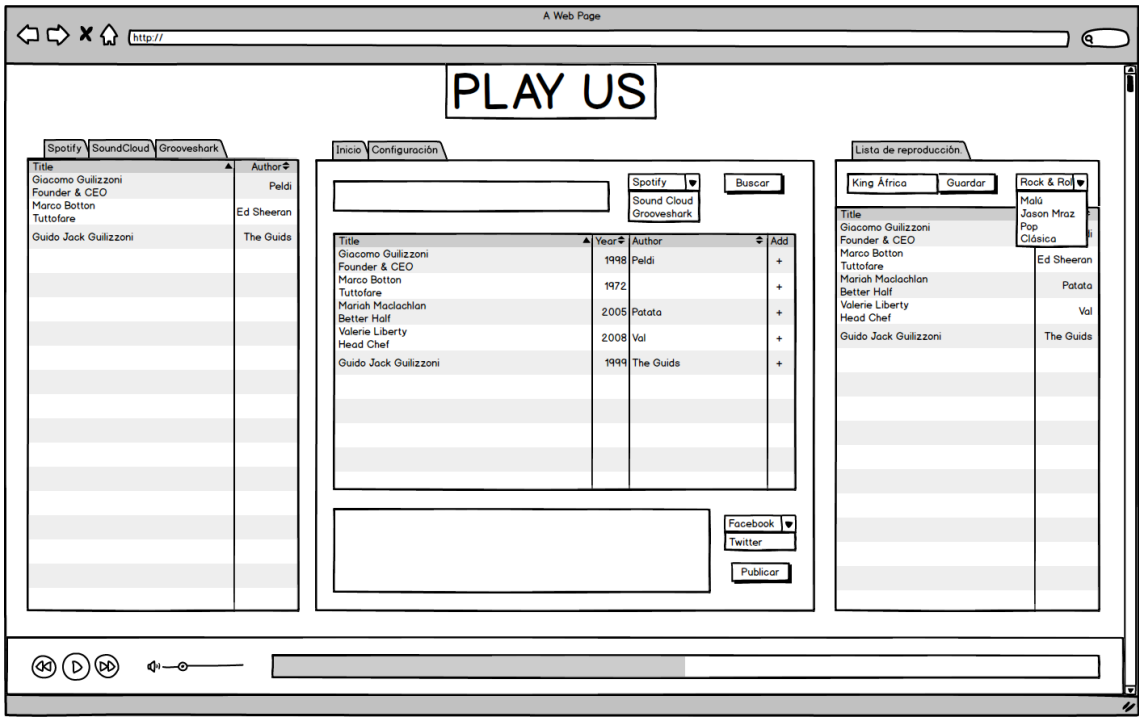


FIGURA 1. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA INICIO

2.2 Vista Configuración

En esta vista, el único cambio que encontramos es el del panel central, que ahora muestra una serie de opciones para autenticarnos en las distintas web.

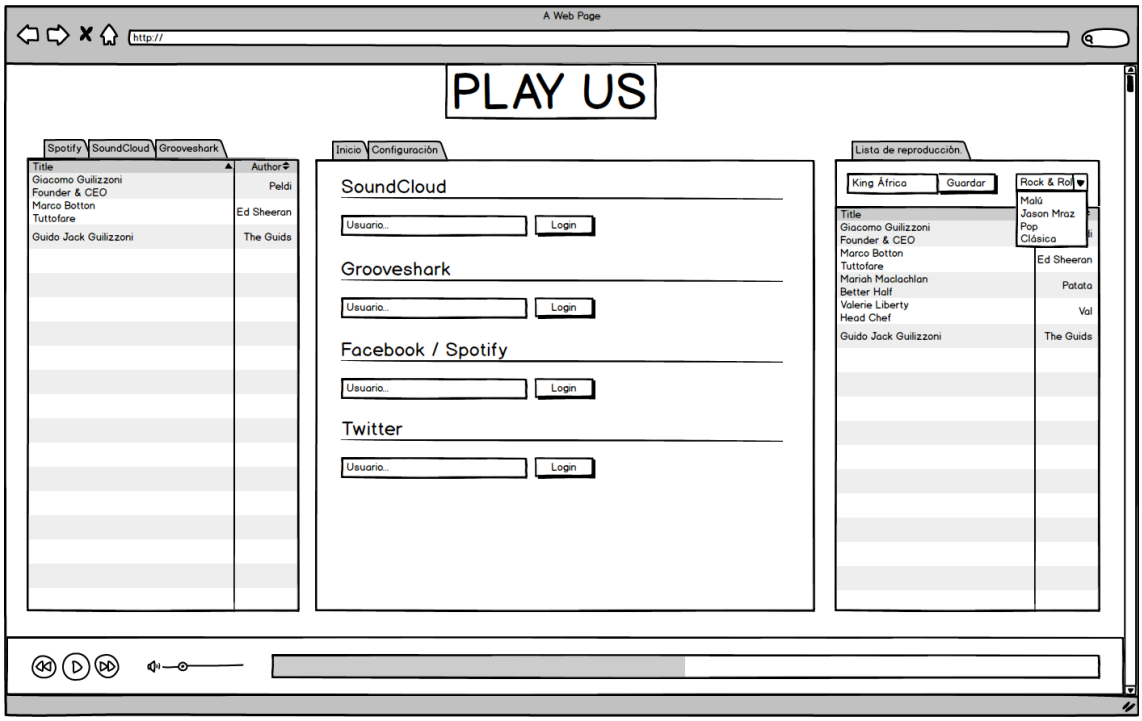


FIGURA 2. PROTOTIPO DE INTERFAZ DE USUARIO DE LA VISTA CONFIGURACIÓN.

3 Arquitectura

3.1 Diagrama de componentes.

Nuestra aplicación va ha utilizar diferentes servicios web, entre ellos SoundCloud, Google Drive y Grooveshark para reproducir canciones y Twitter y Facebook como redes sociales.

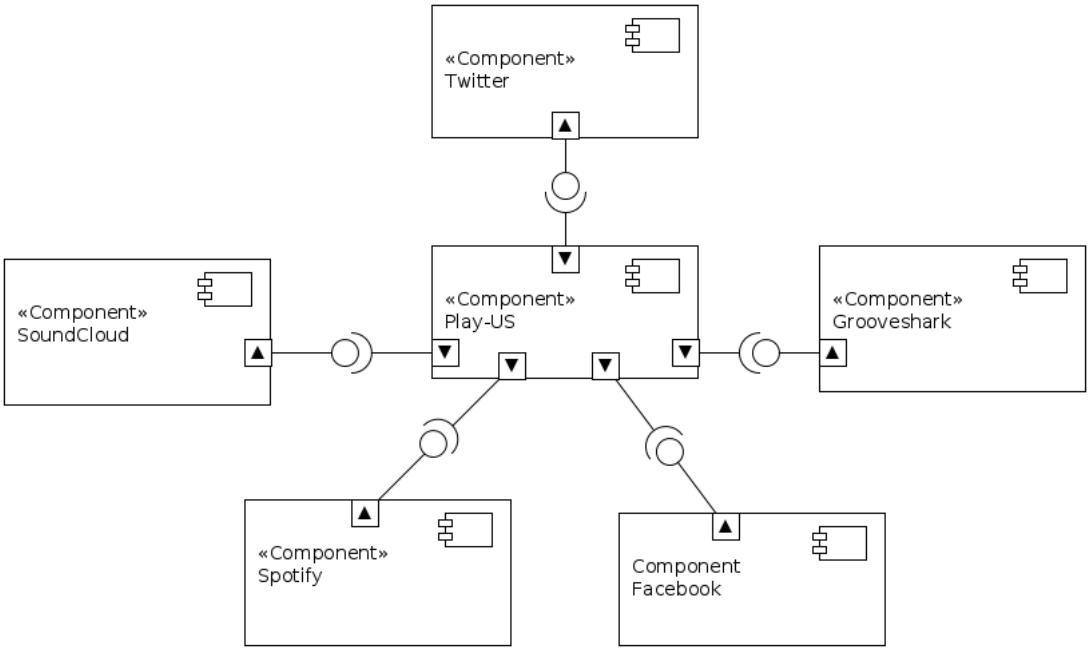


FIGURA 3. PROTOTIPO DE DIAGRAMA DE COMPONENTES.

3.2 Diagrama de despliegue

El mashup estará desarrollado en tecnología GWT y se ejecutará en AppEngine en un servidor de Google. Se podrá acceder a través de un navegador web.

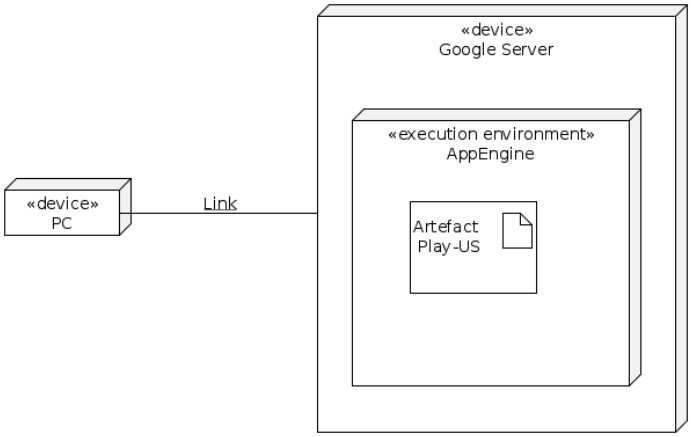


FIGURA 4 . PROTOTIPO DE DIAGRAMA DE DESPLIEGUE.

4 Implementación.

Como elementos destacados de este proyecto, esta la creación de un objeto llamado Song, en el cual mapeamos todas las canciones de los distintos sitios web y que después pasamos a nuestro propio reproductor, el cual dispone de un lista de reproducción y los botones que tiene todo reproductor.

Otro de los retos de este proyecto a sido la implementación de Twitter ya que la autenticación es con oauth1.

5 Pruebas.

Todas las pruebas están automatizadas con Junit. El ID de la prueba esta compuesto por: {nombre de la clase en el paquete de test}/ {nombre del método}.

ID	**`GoogleDriveResourceTest/testGetFiles`**
Descripción	Prueba para la detección de errores al implementar la búsqueda de archivos en Google Drive usando servicios RESTful.
Entrada	Como entrada tenemos un token(El cual podemos obtener de la página al conectarnos a Google Drive) y un filtro.
Salida esperada	Se espera que todo salga correcto y se imprime por pantalla las búsquedas realizadas.
Resultado	**ÉXITO**

ID	**`GoogleDriveResourceTest/testGetFile`**
Descripción	Prueba para la detección de errores al implementar el servicio para pedir información sobre un archivo en Google Drive usando servicios RESTful.
Entrada	Como entrada tenemos un token(El cual podemos obtener de la página al conectarnos a Google Drive) , un filtro y un id(El cual obtenemos de la prueba anterior).
Salida esperada	Se espera que todo salga correcto y se imprime por pantalla el nombre del archivo.
Resultado	**ÉXITO**

ID	**`GroovesharkResourceTest/testGetCountry`**
Descripción	Prueba para la detección de errores al implementar el servicio para pedir información sobre el país en el que estamos, en Grooveshark usando servicios RESTful.

Entrada	No tenemos ningún parámetro de entrada.
Salida esperada	Se espera que todo salga correcto.
Resultado	**ÉXITO**

ID	**GroovesharkResourceTest/testGetSongSearchResults**
Descripción	Prueba para la detección de errores al implementar el servicio para buscar canciones, en Grooveshark usando servicios RESTful.
Entrada	Como parámetros de entrada tenemos el número de canciones que queremos que nos imprima(limit), un country(obtenido en la prueba anterior) y la palabra a buscar(search).
Salida esperada	Se espera que todo salga correcto y que se imprima por pantalla la lista de los títulos de las canciones encontradas.
Resultado	**ÉXITO**

ID	**GroovesharkResourceTest/testGetStartSession**
Descripción	Prueba para la detección de errores al implementar el servicio para obtener un identificador de la sesión, en Grooveshakr usando servicios RESTful.
Entrada	No tenemos parámetros de entrada.
Salida esperada	Se espera que todo salga correcto y que se imprima por pantalla el identificador obtenido.
Resultado	**ÉXITO**

ID	**GroovesharkResourceTest/testGetGetStreamKeyStreamServer**
Descripción	Prueba para la detección de errores al implementar el servicio para obtener la url de streaming de una canción, en Grooveshakr usando servicios RESTful.

Entrada	Como parámetros de entrada tenemos el songID(que obtenemos en prueba de búsqueda), un objeto country(que obtenemos en la prueba testGetCountry) y una sesionID(Obtenida en la prueba testGetStartSession)
Salida esperada	Se espera que todo salga correcto y que se imprima por pantalla la url obtenida.
Resultado	**ÉXITO**

ID	**`PlayUSResourceTest/testUser`**
Descripción	Prueba para la detección de errores al implementar el servicio para la administración de usuarios.
Entrada	Como parámetros de entrada tenemos un email y una password.
Salida esperada	Se espera que todo salga correcto y que se imprima por pantalla el usuario insertado.
Resultado	**ÉXITO**

ID	**`PlayUSResourceTest/testPlaylist`**
Descripción	Prueba para la detección de errores al implementar el servicio para la administración de las listas de reproducción.
Entrada	Como parámetros de entrada tenemos un email y el nombre de una lista de reproducción.
Salida esperada	Se espera que todo salga correcto y que se imprima por pantalla la lista de reproducción creada.
Resultado	**ÉXITO**

ID	**`PlayUSResourceTest/testSong`**
Descripción	Prueba para la detección de errores al implementar el servicio para la administración de una lista de

	reproducción.
Entrada	Como parámetros de entrada tenemos un email y el nombre de una lista de reproducción.
Salida esperada	Se espera que todo salga correcto y que se imprima por pantalla la lista de canciones de esa lista de reproducción.
Resultado	**ÉXITO**

ID	**`SoundcloudResourceTest/testGetTracks`**
Descripción	Prueba para la detección de errores al implementar el servicio para buscar canciones, en SoundCloud usando servicios RESTful.
Entrada	Como parámetros de entrada tenemos la palabra a buscar(filter), la página a mostrar y el número de canciones por páginas.
Salida esperada	Se espera que todo salga correcto y que se imprima por pantalla la lista de los títulos de las canciones encontradas en la búsqueda.
Resultado	**ÉXITO**

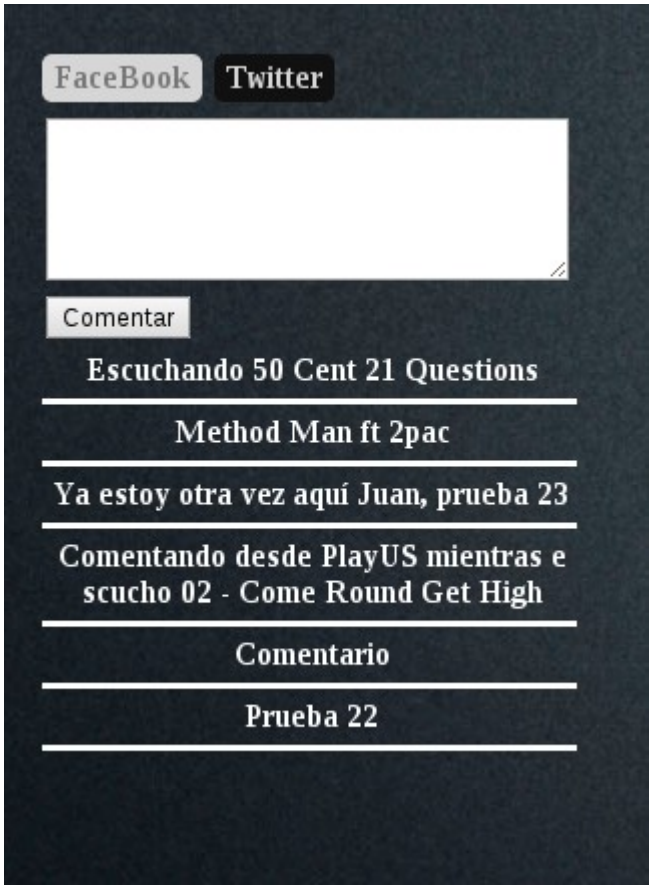
ID	**`SoundcloudResourceTest/getTrackId`**
Descripción	Prueba para la detección de errores al implementar el servicio para obtener información sobre una canción, en SoundCloud usando servicios RESTful.
Entrada	Como parámetros de entrada tenemos el ID de la canción (Se coge el primer ID de la búsqueda de canciones).
Salida esperada	Se espera que todo salga correcto y que se imprima por pantalla el título de la canción.
Resultado	**ÉXITO**

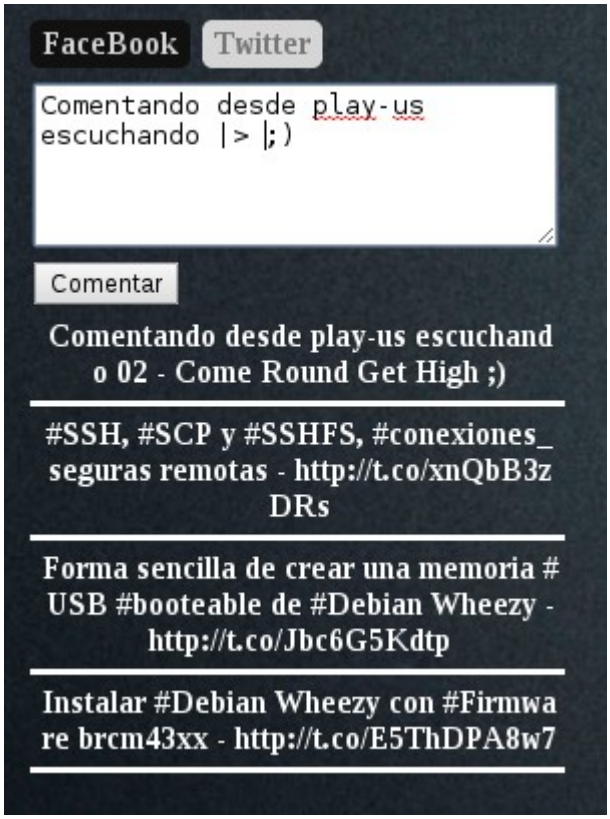
ID	**facebookResourceTest / testPublicarEstado**
Descripción	Prueba para la detección de errores al implementar la publicación de estados en Facebook a través de su Graph API.
Entrada	La única entrada será un token, tanto con permisos para publicar como para la petición de información, que será necesario para la invocación de los métodos de publicación y de petición de estados.
Salida esperada	Una vez hecha la publicación, pedimos el último estado que será el que se supone que hemos publicado y lo comparamos con el inicial, si son iguales, la publicación se ha resuelto correctamente, en caso contrario no.
Resultado	**ÉXITO**

ID	**facebookResourceTest / testGetEstados**
Descripción	Prueba para la detección de errores al implementar la petición de estados en Facebook a través de su Graph API.
Entrada	La única entrada será un token con permisos de petición de información, que será necesario para el método de petición de estados.
Salida esperada	Los datos devueltos en formato JSON son mapeados a una clase Java, concretamente Status, que contendrá una lista de Datum. Cada Datum contendrá un atributo message, que serán los mensajes de estados devueltos, de ahí que recorramos la lista de Datum en nuestro Status devuelto para formar la lista de mensajes. Finalmente comprobamos si es nula. Si realmente lo es la salida esperada será un fallo, si no lo es es un éxito.
Resultado	**ÉXITO**

6 Manual de usuario.

6.1 Mahup.

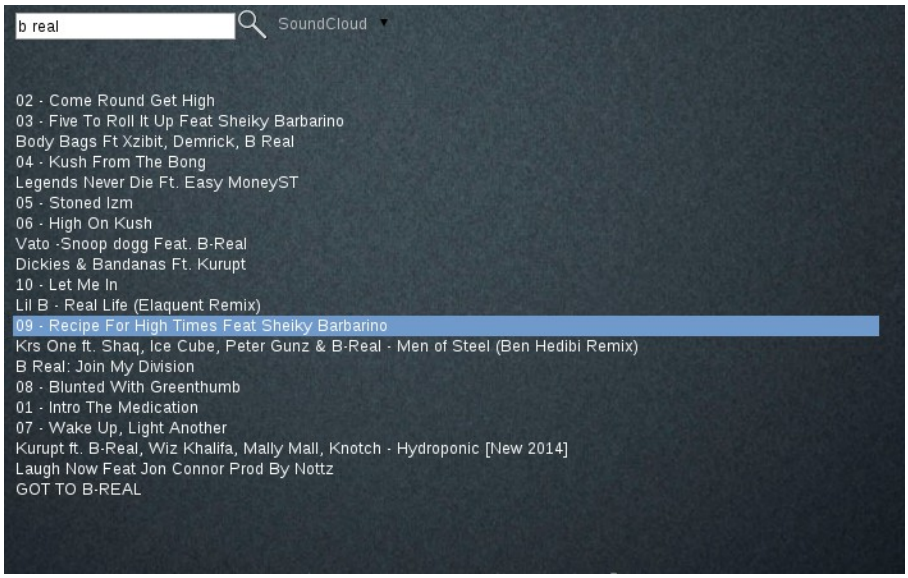




Esta parte de la aplicación está reservada para las redes sociales, será desde donde se pueden crear los comentarios y ver tus últimos comentarios.

Para poder realizar un comentario o poder visualizar los que has realizado, primero tendrás que conectar tu Facebook o Twitter con Play-US mediante el panel de configuración.

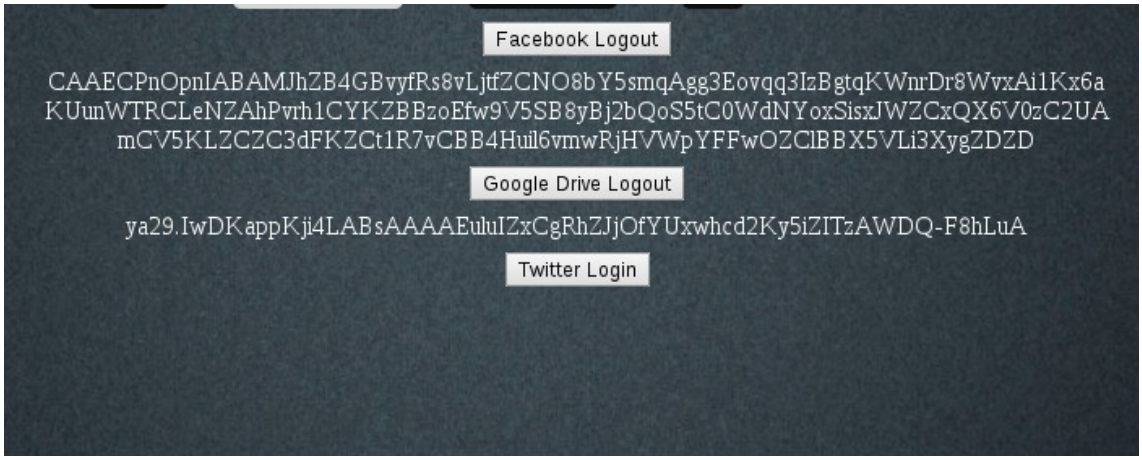
Al realizar un comentario podemos poner el símbolo “|>” y si estamos escuchando una canción nos lo cambiará por el nombre de dicha canción.



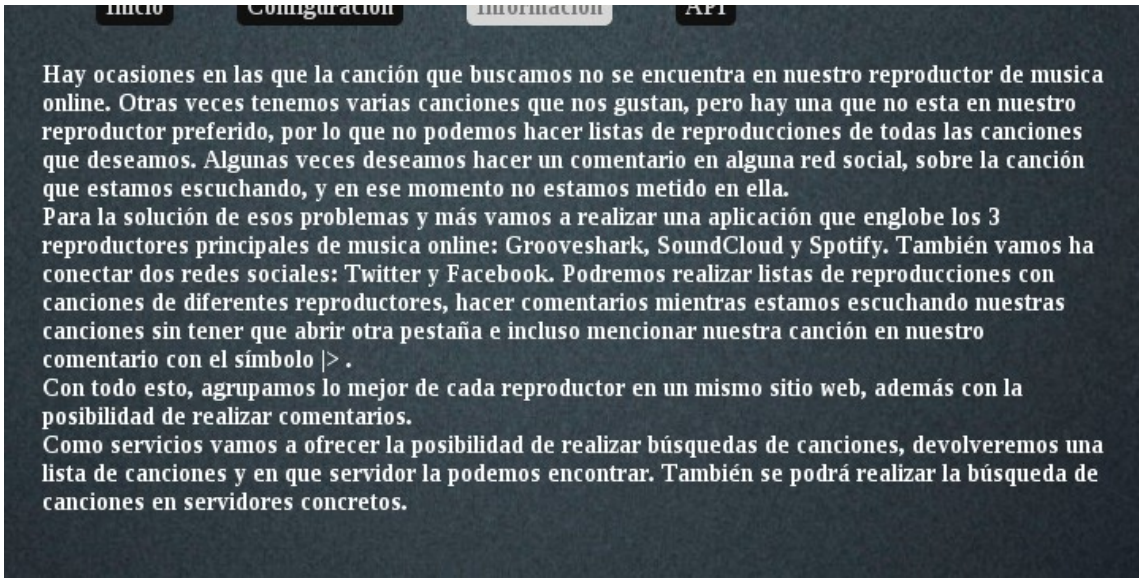
Esta parte está en la parte central de nuestra aplicación, en la pestaña inicio. En ella, podemos buscar canciones en diferentes servidores y añadirlas a nuestra lista de reproducción con doble click.

Para seleccionar el servidor picamos en el seleccionador que hay al lado de la lupa y aparecerán los distintos servidores. Si queremos realizar una búsqueda desde Google Drive, primero tendremos que pasar por la pestaña de configuración y conectar nuestra cuenta.

Cuando conectemos nuestra cuenta de Google Drive con Play-US, podremos buscar las canciones que se encuentra en nuestro Google Drive. Si no ponemos ninguna palabra aparecerán todas. Al introducir una palabra aparecerán las que contengan esa cadena.

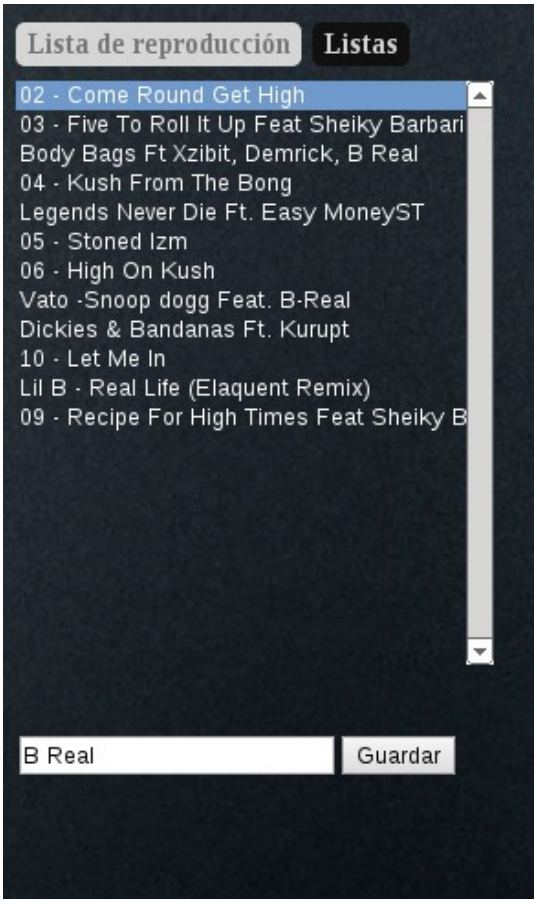


En la parte central también encontramos la pestaña de configuración, donde podremos iniciar sesión en los servidores que creamos necesario. Al iniciar sesión en google o facebook, nos aparecerá el token que nos ha asignado, por si queremos utilizarlo en otro sitio. Con twitter nos aparecerá en la URL.



En la siguiente pestaña de la parte central, Información, podemos encontrar una descripción del proyecto.

Por último, pero no menos importante, tenemos en la parte central de nuestra aplicación la pestaña API. En ella aparece un manual detallado con los métodos que se pueden usar y como se pueden usar, acompañados de ejemplos. Podemos darle al botón “Ampliar” y nos ampliará la página.



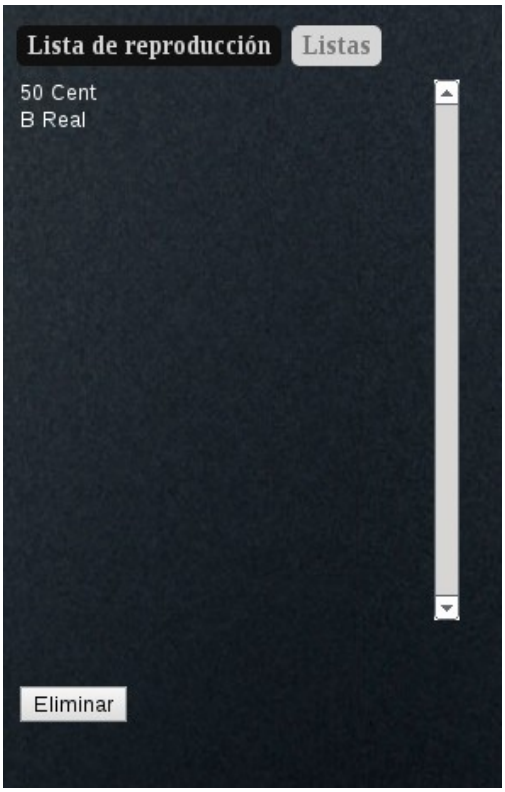
En la parte derecha de nuestra aplicación, en la primera pestaña, tenemos la lista de reproducción.

En esta lista podemos encontrar las canciones que hemos añadido tras nuestra búsqueda.

Esta lista podemos guardarla poniendo el nombre que deseamos en el recuadro de abajo y pulsando sobre el botón guardar.

Esta lista detecta diferentes acciones:

- Doble Click: Inicia la reproducción de la canción seleccionada.
- D: Elimina la canción seleccionada.
- P: Reproduce/Pausa la canción seleccionada.
- N: Pasa a la siguiente canción de la lista.
- L: Pasa a la canción anterior de la lista.



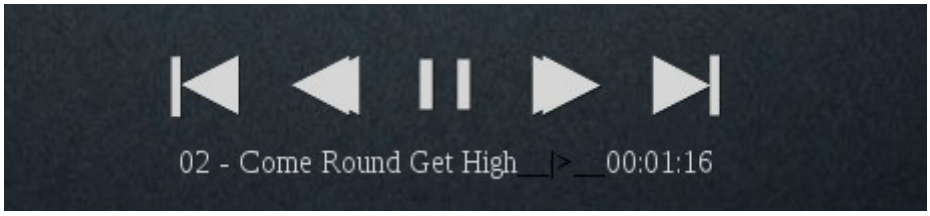
La segunda pestaña de la parte derecha contiene las listas de reproducción guardadas.

Para poder acceder a esta parte tendremos que estar registrados como usuario en Play-US.

Para cargar una lista simplemente tenemos que hacer doble clic en la lista que deseemos.

Si queremos eliminar una lista solo tenemos que seleccionarla y darle a el botón eliminar.

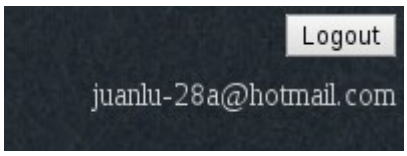
Para modificarlas, primero las cargamos y después la guardamos con el mismo nombre.



Este es el reproductor que podemos encontrar en la parte baja de la aplicación. En él tenemos los botones básicos de manejo de canciones, el título de la canción que se esta reproduciendo y el tiempo que lleva reproduciendo.

Para entrar o registrarnos, tenemos estos dos botones.

Cuando le demos a alguno de los dos nos aparecerá este cartel naranja en el cual tenemos que introducir el nombre y la contraseña de nuestro usuario.



Una vez estemos registrados y entremos nos aparecerá nuestro usuario y un botón para cerrar la sesión.

6.2 API REST

La API Play-US ofrece recursos propios tales como listas de reproducción y otros ajenos como canciones mediante llamadas a otras API´s haciendo peticiones HTTP, como GET, POST, DELETE o PUT.

Tanto las peticiones como las respuestas se harán en formato Json.

Todos los recursos son accedidos y manipulados de manera similar:

```
/[nombre del recurso]
```

Estos recursos pueden tener otros que dependan de él, de la forma:

```
/[nombre del recurso]/[nombre del subrecurso]
```

La URI de punto de entrada a la API es:

http://play-us.appspot.com/api

Para tener acceso a los datos no es necesario nada, solo hacer la petición. Si queremos ver todos los datos del usuario o modificar los datos tenemos que pasar el parámetro md5, con la contraseña del usuario encriptada en md5.

6.2.1 /users

Devuelve una lista de usuarios.

Métodos

Método	Ruta	Descripción
GET	/users	Retorna una lista de usuarios.
POST	/users	Retorna el usuario creado con una respuesta 201.

Parámetros.

Parámetro	Tipo	Descripción
q	String	Cadena de filtrado.

Ejemplos.

GET

URL : http://play-us.appspot.com/api/users/

POST

URL: http://play-us.appspot.com/api/users/

Json:
{"email":"prueba@email.com","tokenF":null,"tokenG":null,"password":"202cb962ac59075b964b07152d234b70","playlists":[]}

6.2.2 /users/{id}

Provee de la información general de un usuario.

Propiedades

Parámetro	Tipo	Descripción
email	String	Email del usuario.
playlists	Playlist[]	Listas de reproducción del usuario.

Métodos

Método	Ruta	Descripción
GET	/users/{id}	Devuelve el usuario.
PUT	/users/{id}	Modifica información de usuario y devuelve un estado 201.
DELETE	/users/{id}	Eliminar el usuario y devuelve un estado 200.

Parámetros.

Parámetro	Tipo	Descripción
md5	String	Contraseña del usuario.

Ejemplos.

GET

URL: http://play-us.appspot.com/api/users/prueba@email.com

PUT

URL: http://play-us.appspot.com/api/users/prueba@email.com?md5=202cb962ac59075b964b07152d234b70

Json:
{"email":"prueba@email.com","tokenF":null,"tokenG":null,"password":"202cb962ac59075b964b07152d234b71","playlists":[]}

DELETE

URL: http://play-us.appspot.com/api/users/prueba@email.com?md5=202cb962ac59075b964b07152d234b71

6.2.3 /users/full/{id}

Devuelve toda la información de un usuario.

Propiedades

Parámetro	Tipo	Descripción
email	String	Email del usuario.
tokenF	String	Token(si lo hay) de facebook.
tokenT	String	Token(si lo hay) de twitter.
password	String	Contraseña encriptada en MD5.
playlists	Playlist[]	Listas de reproducción del usuario.

Métodos

Método	Ruta	Descripción
GET	/users/full/{id}	Devuelve el usuario.

Parámetros.

Parámetro	Tipo	Descripción
md5	String	Contraseña del usuario.

Ejemplos.

GET.

URL: http://play-us.appspot.com/api/users/full/prueba@email.com?md5=202cb962ac59075b964b07152d234b70

6.2.4 /users/{id}/playlists

Provee información de una lista con las listas de reproducción de un usuario.

Métodos

Método	Ruta	Descripción
GET	/users/{id}/playlists	Información de las listas de reproducción.
POST	/users/{id}/playlists	Retorna la playlist creada con un estado 201.

Filtros

Parámetro	Tipo	Descripción
q	String	Cadena de filtrado.
md5	String	Contraseña del usuario.

Ejemplos.

GET

URL: http://play-us.appspot.com/api/users/prueba@email.com/playlists

POST

URL: http://play-us.appspot.com/api/users/prueba@email.com/playlists?md5=202cb962ac59075b964b07152d234b70

Json: {"playlist_name":"50 Cent","size":3,"song_list":[]}

6.2.5 /users/{id}/playlists/{playlist_name}

Provee información de una lista de reproducción de un usuario concreto.

Propiedades

Parámetro	Tipo	Descripción
playlist_name	String	Nombre de la lista de reproducción.
size	Integer	Número de canciones de la lista de reproducción.
song_list	Song[]	Lista de canciones de la lista de reproducción.

Métodos

Método	Ruta	Descripción
GET	/users/{id}/playlists/{playlist_name}	Información de la lista de reproducción.
PUT	/users/{id}/playlists/{playlist_name}	Modifica una lista de reproducción y devuelve un estado 201.
DELETE	/users/{id}/playlists/{playlist_name}	Elimina una lista de reproducción y devuelve un estado 200.

Filtros

Parámetro	Tipo	Descripción
md5	String	Contraseña del usuario.

Ejemplos.

GET

URL: http://play-us.appspot.com/api/users/prueba@email.com/playlists/50+Cent

PUT

URL:http://play-us.appspot.com/api/users/prueba@email.com/playlists/50+Cent?md5=202cb962ac59075b964b07152d234b70

Json: {"playlist_name":"50 Cent 2","size":0,"song_list":[]}

DELETE

URL: http://play-us.appspot.com/api/users/prueba@email.com/playlists/50+Cent+2?md5=202cb962ac59075b964b07152d234b70

6.2.6 /users/{id}/playlists/{playlist_name}/songs

Retorna una lista de canciones.

Métodos

Método	Ruta	Descripción
GET	/users/{id}/playlists/{playlist_name}/songs	Información de las canciones de una lista de reproducción.
POST	/users/{id}/playlists/{playlist_name}/songs	Retorna la canción añadida, con un estado 201.

Filtros

Parámetro	Tipo	Descripción
q	String	Cadena de filtrado.
md5	String	Contraseña del usuario.
p	Integer	Posición en la que se añade la canción.

Ejemplos.

GET

URL: http://play-us.appspot.com/api/users/prueba@email.com/playlists/50+Cent/songs

POST

URL: http://play-us.appspot.com/api/users/prueba@email.com/playlists/50+Cent/songs?md5=202cb962ac59075b964b07152d234b70

Json: {"id_song":null,"name_song":"21 Questions","server_song":"Grooveshark","length_song":null,"artist_song":"null","url_song":null}

6.2.7 /users/{id}/playlists/{playlist_name}/songs/{index}

Provee información de una canción.

Propiedades

Parámetro	Tipo	Descripción
id_song	String	Identificador de la canción.
name_song	String	Nombre de la canción.
server_song	String	Servidor al que pertenece dicha canción.
length_song	Integer	Duración en segundos de la canción.
artist_song	String	Nombre del autor de la canción.
url_song	String	URL con la que se puede escuchar la canción.

Métodos

Método	Ruta	Descripción
GET	/users/{id}/playlists/{playlist_name}/songs/{index}	Información de la canción.
PUT	/users/{id}/playlists/{playlist_name}/songs/{index}	Modifica una canción y devuelve un estado 201.
DELETE	/users/{id}/playlists/{playlist_name}/songs/{index}	Elimina una canción y devuelve un estado 200.

Filtros

Parámetro	Tipo	Descripción
md5	String	Contraseña del usuario.

Ejemplos.

GET

URL: http://play-us.appspot.com/api/users/prueba@email.com/playlists/50+Cent/songs/0

PUT

URL: http://play-us.appspot.com/api/users/prueba@email.com/playlists/50+Cent/songs/0?md5=202cb962ac59075b964b07152d234b70

Json: {"id_song":null,"name_song":"We Up","server_song":"Grooveshark","length_song":null,"artist_song":"null","url_song":null}

DELETE

URL: http://play-us.appspot.com/api/users/prueba@email.com/playlists/50+Cent/songs/0?md5=202cb962ac59075b964b07152d234b70

6.2.8 /songs

Retorna una lista de canciones.

Métodos

Método	Ruta	Descripción
GET	/songs	Lista de canciones.

Filtros

Parámetro	Tipo	Descripción
q	String	Cadena de filtrado.
limit	Integer	Numero de canciones por servidor(en Grooveshark no ofreceremos mas de 20).
server	SoundCloud o Grooveshark	Filtrado por servidor.
offset	Integer	Pagina de la búsqueda, que se quiere mostrar.

Ejemplos.

GET

URL: http://play-us.appspot.com/api/songs?q=50&limit=5&offset=0

6.2.9 /songs/{server}/{id}

Provee información sobre una canción de un servidor concreto.

Propiedades

Parámetro	Tipo	Descripción
id_song	String	Identificador de la canción.
name_song	String	Nombre de la canción.
server_song	String	Servidor al que pertenece dicha canción.
length_song	Integer	Duración en segundos de la canción.
artist_song	String	Nombre del autor de la canción.
url_song	String	URL con la que se puede escuchar la canción.

Métodos

Método	Ruta	Descripción
GET	/songs/{server}/{id}	Información de la canción

Ejemplos.

GET

URL: http://play-us.appspot.com/api/songs/SoundCloud/50380380?q=50&limit=5&offset=0

6.2.10 Tabla de errores.

Error	Cuando puede ocurrir.
Error 404 This user not exist	Intento acceder a un usuario que no existe.
Error 404 You must enter the password in MD5 with the parameter md5	Es necesario el parámetro md5 y no se ha introducido o el valor no tiene el formato de una cadena en MD5
Error 404 User or password incorrect	La contraseña no coincide con la

	contraseña de ese usuario.
Error 404 Do not match emails	Al intentar modificar el email de un usuario.
Error 404 This song not exist	El índice es mayor o igual al numero de canciones.
Error 404 This playlist not exist	No se encuentra esa lista de reproducción.
Error 404 The user not exist or the md5 key is incorrect	El usuario no existe o la contraseña que se ha introducido no existe con la del usuario.
Error 404 The user already exists	El usuario que intentas crear ya existe.
Error 404 Send a user	No se ha enviado un usuario en el Json
Error 404 The email of the user must not be null	El email del usuario que se intenta crear es nulo.
Error 404 The password of the user must be a hash MD5	El password del usuario que se intenta crear, no coincide con la estructura de un hash MD5.
Error 404 The playlist already exists	La lista de reproducción que intentas crear tiene el mismo nombre que otra ya existente.
Error 404 The playlist not found	No se ha encontrado la lista de reproducción.
Error 404 The position exceeds	La posición en la que se quiere meter la canción excede el tamaño de la lista de canciones.

Referencias

[1] *Balsamiq*. http://balsamiq.com/

[2] J. Webber, S. Parastatidis y I. Robinson. *REST in Practice: Hypermedia and Systems Architecture.* O'Reilly Media. 2010.

[3] UMLet. http://www.umlet.com/

[4] http://www.gwtproject.org/
[5] http://stackoverflow.com/

[6] http://es.wikipedia.org/

[7] http://www.mkyong.com/