



Arquitectura e Integración de Sistemas Software

Grado de Ingeniería del Software

Curso 2º

Ángel Delgado Luna (angeldelgadoluna@gmail.com)

Reyes Cabello Holgado (reyescabello97@gmail.com)

Tutor: Adela Del Rio

Número de grupo: "Sotano Apps" – G3 - ADR

Enlace de la aplicación: <http://spotygo2.appspot.com/>

Enlace de proyecto en GitHub: <https://github.com/angel96/ProyectoDeAISS20>

HISTORIAL DE VERSIONES

Fecha	Versión	Detalles	Participantes
3 de Julio de 2017	1.0	- Google Drive Implementado con Oauth2 - TMDB Implementado - Youtube Implementado	- Ángel Delgado Luna
5 de Julio de 2017	1.1	- Incorporado imports scripts JQuery	- Ángel Delgado Luna
7 de Septiembre de 2017	1.2	- Separar vistas con variables en sesión - Implementación API Rest y Junit4 - Inclusión de Diagrama de Despliegue - Inclusión de Diagrama de Componentes - Inclusión de pruebas escritas	- Ángel Delgado Luna - Reyes Cabello Holgado
8 de Septiembre	1.3	- Diagrama de clases - Diagrama de secuencias del GET TMDB	- Ángel Delgado Luna - Reyes Cabello Holgado
15 de Septiembre	1.4	- Resuelto el problema con la variable sesión	- Ángel Delgado Luna - Reyes Cabello Holgado
Septiembre – Octubre – Noviembre	1.5	- JavaScript de cada vista - Mejora de los controladores	- Reyes Cabello Holgado
Noviembre	1.6	- Implementación de la API REST – Google Drive	- Ángel Delgado Luna
Diciembre	1.7	- Despliegue de la aplicación y reconfiguración del Oauth2 en AppEngine - Documentación de la API REST	- Ángel Delgado Luna
Diciembre	1.8	- Mejora de la API REST	- Reyes Cabello Holgado
Diciembre	1.8	- Completar diagramas GET(TMDB Y YOUTUBE) - Mejora de la documentación API REST - Revisión de pruebas - Diagrama de clases MVC	- Ángel Delgado Luna
Diciembre	1.8	- Mejora de la aplicación (Visual y código)	- Reyes Cabello Holgado

Índice

1	Introducción	4
1.1	Aplicaciones integradas	4
1.2	Evolución del proyecto	4
2	Prototipos de interfaz de usuario	5
2.1	Vista X	¡Error! Marcador no definido.
2.2	Vista Y.....	¡Error! Marcador no definido.
3	Arquitectura	8
3.1	Diagrama de componentes.....	8
3.2	Diagrama de despliegue	8
3.3	Diagrama de secuencia de alto nivel	9
3.4	Diagrama de clases	9
3.5	Diagramas de secuencia	10
4	Implementación	11
5	Pruebas.....	14
6	Manual de usuario	17
6.1	Mashup	17
6.2	API REST	22
	Referencias	26

1 Introducción

¿Cuántas veces no hemos deseado de guardarnos algunas películas o series “en el bolsillo” para de alguna manera verlas en el cine u online a posteriori? Pues bien, este proyecto que implementa tres apis que facilitan esa labor vienen a dar solución a este problema.

TMDB (The Movie Database) es nuestro referente a la hora de consultar contenido sobre series o películas, donde al hacer una consulta podremos ver un tráiler o video relacionado con la API de Youtube y a posteriori realizar una colección e incluirlo en un fichero de texto plano en Google Drive.

1.1 Aplicaciones integradas

Describir cada una de las aplicaciones integradas dando detalles sobre cada una de ellas

Nombre aplicación	URL documentación API
Google Drive	https://developers.google.com/drive/v2/web/about-sdk
The Movie Database	https://developers.themoviedb.org/3
Youtube API v3	https://developers.google.com/youtube/documentation/

TABLA 1. APLICACIÓN INTEGRADAS

1.2 Evolución del proyecto

El proyecto basado en el ámbito cinematográfico como hemos descrito en la introducción, se hizo pensando en dar la facilidad a los usuarios que desearan darle uso.

Si bien es verdad, que este ha dado muchísimos problemas de implementación. Sobre todo, en lo que se refiere a generación de recursos Rest y obtención de los modelos de cada api. En Julio, se implementó las APIs para trabajar de manera troncal con el trabajo y evitar con tiempo los posibles odiosos fallos en servidor que suelen provocar.

Hasta septiembre que no hemos vuelto a tocar el trabajo, pese a los exámenes, donde hemos implementado las pruebas automáticas y además las vistas, junto con los repositorios. Ahora en Octubre – Noviembre – Diciembre, para presentarlo, hemos optimizado las vistas y terminado todo lo que se relaciona con el código y su

implementación como bien se detallará en esta memoria. Además de los cambios visibles en el enlace a github que tenemos en la cabecera de nuestro proyecto.

2 Prototipos de interfaz de usuario



Índice

BUSCADOR

No Image

No Image

No Image

No Image

No Image

No Image

No Image

No Image

No Image

No Image

No Image

BUSCAR EN YOUTUBE

Buscador TMDB

GUARDAR EN GOOGLE DRIVE

No Image

No Image

No Image

No Image

No Image

Selector Youtube

Google Drive

TÍTULO

Películas seleccionadas

GUARDAR

Editor Google Drive

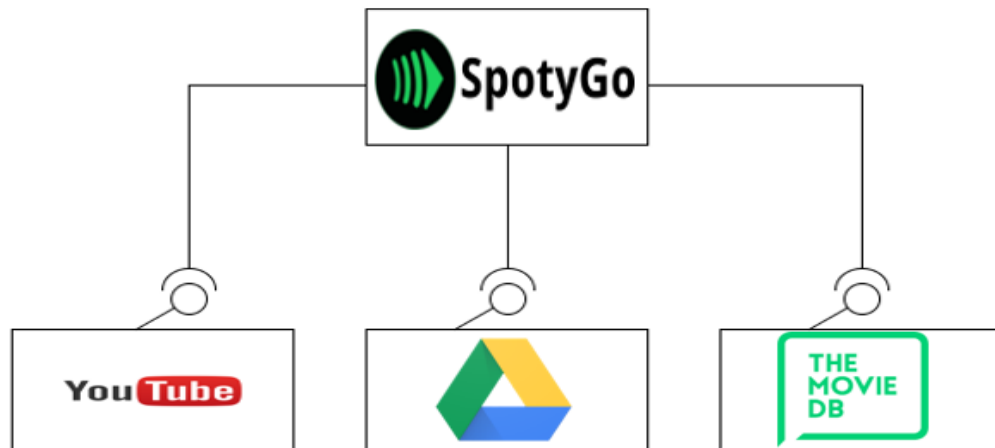
ARCHIVOS DE GOOGLE DRIVE

EDITAR	BORRAR	Archivo
		Archivo 1
		Archivo 2
		Archivo 3
		Archivo 4
		Archivo 5
		Archivo 6
		Archivo 7
		Archivo 8
		Archivo 9

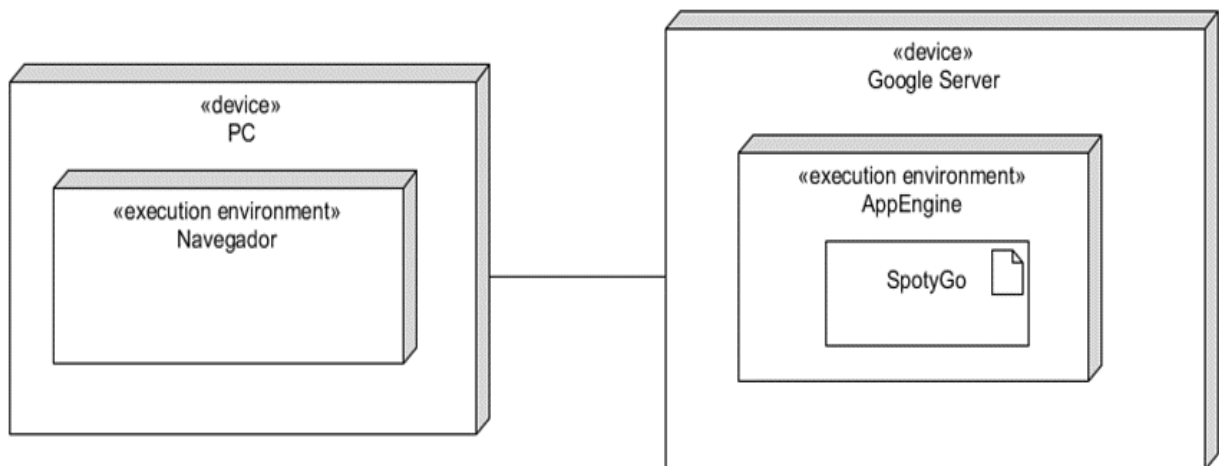
Selector de archivos de Google Drive

3 Arquitectura

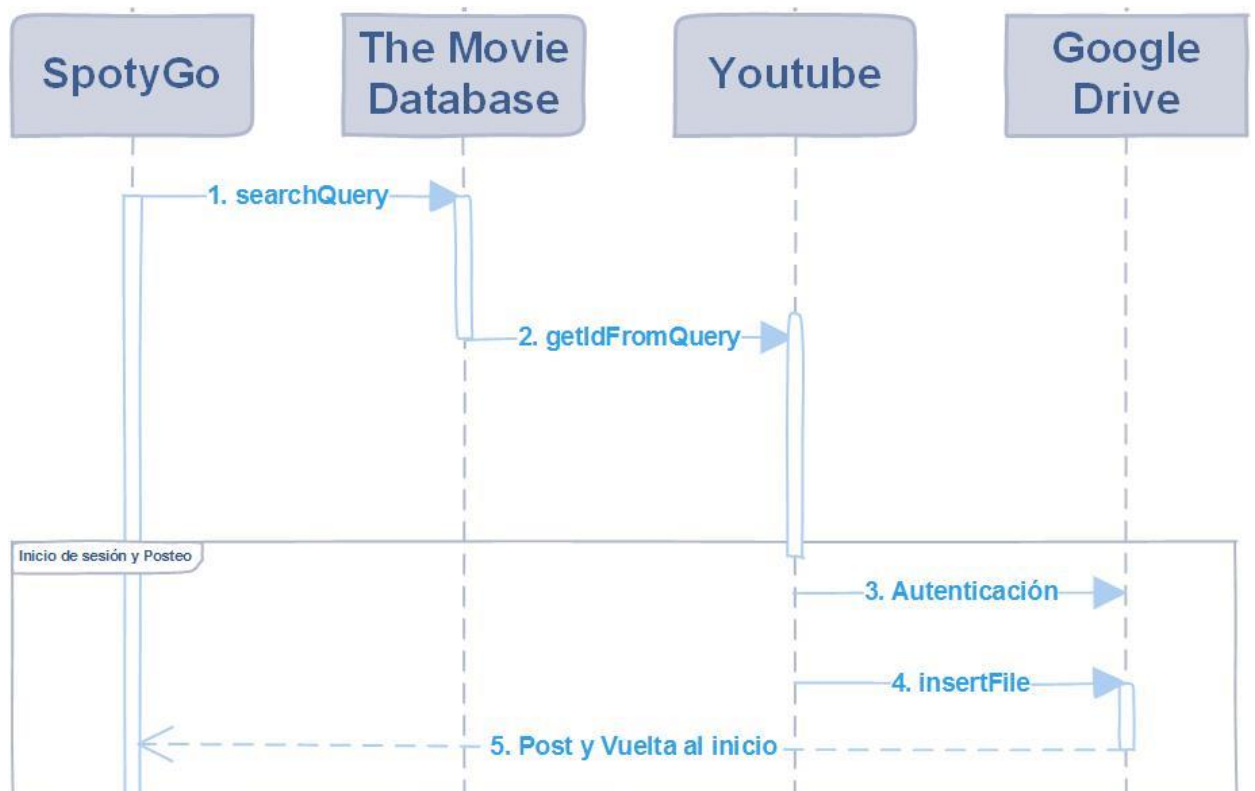
3.1 Diagrama de componentes



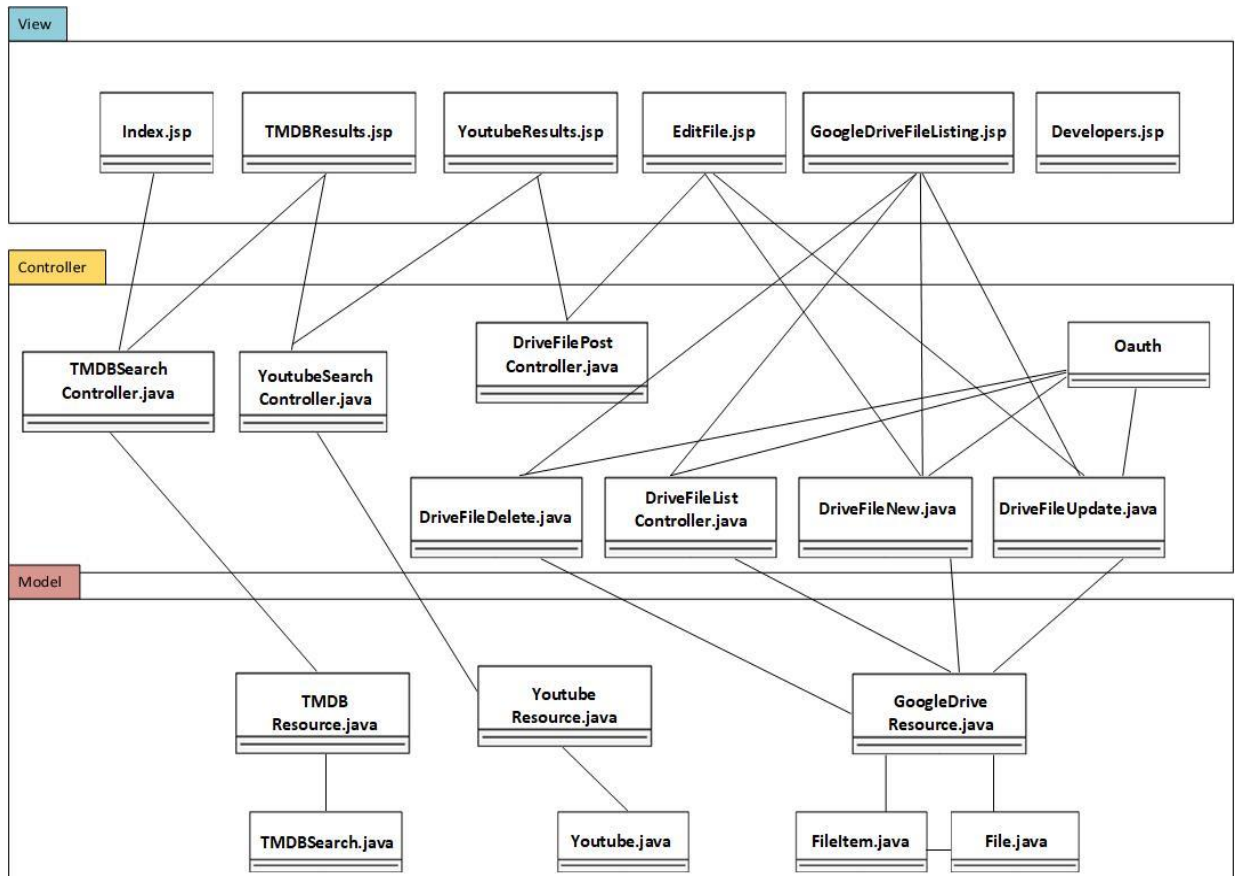
3.2 Diagrama de despliegue



3.3 Diagrama de secuencia de alto nivel



3.4 Diagrama de clases



3.5 Diagramas de secuencia

4 Implementación

(Detalles mas destacables)

Google Drive

- GET File

Recibe como parámetro el id de un determinado fichero para poder tratarlo a posteriori

```
public FileItem getFile(String id) {  
  
    ClientResource cr = null;  
    FileItem file = null;  
    try {  
        cr = new ClientResource(uri + "/" + id + "?access_token=" + access_token);  
        file = cr.get(FileItem.class);  
  
    } catch (ResourceException re) {  
        log.warning("Error when retrieving file: " + cr.getResponse().getStatus());  
    }  
  
    return file;  
  
}
```

- GET FileContent

Para poder editar el contenido, primero tenemos que conseguir los meta datos del fichero, usando este método.

```
public String getFileContent(FileItem item){  
    String result=null;  
    String contentURL=item.getDownloadUrl();  
    try{  
        ClientResource cr = new ClientResource(contentURL);  
        /*Map<String, Object> reqAttribs = cr.getRequestAttributes();  
        Series<Header> headers = (Series<Header>)reqAttribs.get("org.restlet.http.headers");  
        if (headers == null) {  
            headers = new Series<Header>(Header.class);  
            reqAttribs.put("org.restlet.http.headers", headers);  
        }  
        headers.add(new Header("Authorization:", "Bearer "+access_token));*/  
        ChallengeResponse chr = new ChallengeResponse(  
            ChallengeScheme.HTTP_OAUTH_BEARER);  
        chr.setRawValue(access_token);  
        cr.setChallengeResponse(chr);  
  
        result=cr.get(String.class);  
    } catch (ResourceException re) {  
        log.warning("Error when obtaining content of file: " + item.getId());  
    }  
    return result;  
}
```

○ PUT UpdateFileContent

```
public boolean updateFileContent(String id,String content){
    ClientResource cr=new ClientResource(uri_upload+"/"+id+ "?uploadType=media");
    try{
        ChallengeResponse chr = new ChallengeResponse(
            ChallengeScheme.HTTP_OAUTH_BEARER);
        chr.setRawValue(access_token);
        cr.setChallengeResponse(chr);
        StringRepresentation rep=new StringRepresentation(content,MediaType.TEXT_PLAIN);
        cr.put(rep);
    } catch (ResourceException re) {
        log.warning("Error when updating the content of file: " + id);
        log.warning(re.getMessage());
        return false;
    }
    return true;
}
```

Youtube

○ Get Youtube

```
public Youtube getIdFromQuery(String query) {
    // https://www.googleapis.com/youtube/v3/search?part=snippet&maxResults=1&order=relevance&q="
    // + keyword + "&key=YOUR_YOUTUBE_API_KEY"
    // TODO: REALIZAR METODO
    String URI =
        "https://www.googleapis.com/youtube/v3/search?part=snippet&type=video"
        + "&videoType=movie&maxResults="+maxResults()+"&order=relevance&regioncode=ES&q="+query
        + "&key=" + API_KEY;
    log.info("URI:"+URI);
    Youtube res = null;
    ClientResource cl = null;

    try{
        cl = new ClientResource(URI);
        res = cl.get(Youtube.class);
    } catch (ResourceException e){
        System.err.print(e.getStackTrace());
    }
    log.info("Total items:"+res.getItems().size());
    for(Item item:res.getItems())
        log.info("ItemID:"+item.getId().getVideoId());
    return res;
}
```

1. Toma en la URI un método llamado maxResults (Devuelve un número determinado de videos a buscar), query (búsqueda deseada) y API_KEY (Clave de nuestra api)
2. A partir de lo anterior y con el modelo de nuestro JSON construye nuestro objeto youtube, gracias a la librería ClientResource.
3. Devuelve el objeto si no es nulo, en caso contrario salta excepción.

API Google

La api de Google funciona con una variable llamada token que permite el inicio de sesión para acceder a nuestras cuentas.

Nuestra api rest esta basada principalmente en ella. Al programar el código pertinente, para evitar excepciones tipo 500 causadas por el valor a null predeterminado del token (ya que es una variable temporal) hemos tenido que pasarle el parámetro como parte de la Query para acceder al servicio.

```
umes("application/json")
uces("text/plain")
c Response addFile(@Context UriInfo uriInfo, @QueryParam("filename") String file, String content, @QueryParam("oauth") String oauth)
```

Si nos centramos en el caso mas difícil de implementación, para nosotros ha sido el del post. Ya que estamos creando un fichero en drive y no generando un simple JSON como en otras aplicaciones.

```
public String newFiles(String file, String content ,String token) {
    FileItem fileItem = new FileItem();
    fileItem.setTitle(file + ".txt");
    fileItem.setMimeType("text/plain");
    String id = new GoogleDriveResource(token).insertFile(fileItem, content);
    return id;
}
```

JavaScript

También es sorprendente la gran variedad de código JavaScript para poder trabajar con los elementos en las vistas.

Además de destacar la validación en cliente de los campos de textos.

```
document.getElementById("array").value = sessionStorage.getItem("names");

function tableToText() {
    var i, text;
    text = "";
    var info = document.getElementsByTagName("td");

    var a = new Array();
    for (i = 0; i < info.length; i++) {
        if (info[i]) {
            a.push(info[i]);
        }
    }
    info = a;

    for (i = 0; i < info.length; i += 4) {
        text += info[i].innerText + "-#" + info[i + 1].innerText
            + "-#" + info[i + 2].innerText + "-#"
            + info[i + 3].innerText + "-#";
    }
    document.getElementById("textcontent").value = text;
}
```

5 Pruebas

Resumen	
Número total de pruebas realizadas	7
Número de pruebas automatizadas	7 (100 %)

ID	Prueba 1
Descripción	Prueba para la detección de errores al implementar búsquedas en The movie Database
Entrada	Se hace uso de los modelos de The Movie Database para consultar con el recurso existente en la clase aiss.model.resource.TMDBResource.
Salida esperada	Se obtiene a partir de la búsqueda una serie de resultados. Se espera que la búsqueda no sea nula.
Resultado	EXITO
Automatizada	Sí
Código	<pre>@Test public void getSearchNotNull() throws UnsupportedEncodingException { for (i = 0; i < ls.size(); i++) { assertNotNull("Comprobando que la lista del repositorio no sea nula", ls.get(i).getResults()); } }</pre>

ID	Prueba 2
Descripción	Prueba para la detección de errores al implementar búsquedas en The movie Database.
Entrada	Se hace uso de los modelos de The Movie Database para consultar con el recurso existente en la clase aiss.model.resource.TMDBResource.
Salida esperada	Se obtiene a partir de la búsqueda una serie de resultados. Se espera que la búsqueda de elementos sea mayor que 1.
Resultado	EXITO
Automatizada	Sí
Código	<pre>@Test public void getSearchResultsMayorThan() throws UnsupportedEncodingException { assertTrue("Comprobar que se obtienen los objetos", ls.get(i).getResults().size() > 1);}</pre>

ID	Prueba 3
Descripción	Prueba para la detección de errores al implementar búsquedas de videos en Youtube.
Entrada	Se hace uso de los modelos de Youtube para consultar con el recurso existente en la clase aiss.model.resource.YoutubeResource. Por el que se obtiene el id de una serie de videos relacionados con la Query de entrada.
Salida esperada	Se obtienen los videos relacionados con la búsqueda de la Query. Se espera que la búsqueda no sea nula.
Resultado	EXITO
Automatizada	Sí
Código	<pre> @Test public void testGetIdFromQuery() throws UnsupportedEncodingException { for (i = 0; i < ls.size(); i++) { assertNotNull("Comprobando que lo obtenido no sea nulo: ", ls.get(i)); } } </pre>

*Denotar en los casos de google drive que es necesario generar un token temporal para que los casos de JUnit4 y la generación de la api rest funcionen correctamente. El token se obtiene en la siguiente url: <https://developers.google.com/oauthplayground/?code=4/ChhEuAj8HP4DGM1bKTcYLuf41C7Eg2OrkOIWE-mltul#>

*DriveRepositoryTest.java ira adjunto en el pdf. No se añade al proyecto ya que provoca fallo en el deploy.

ID	Prueba 4
Descripción	Consulta de Ficheros en Google Drive
Entrada	Se hace uso de los modelos de Google Drive para consultar con el recurso existente en la clase aiss.model.resource.GoogleDriveResource.
Salida esperada	Se obtiene a partir de la búsqueda una serie de resultados. Se espera que la búsqueda no sea nula.
Resultado	EXITO
Automatizada	Sí
Código	<pre> @Test </pre>

	<pre>public void test01GetFiles() { assertNotNull("Comprobando que se devuelven bien los ficheros con el token del repositorio", ls);} </pre>
--	---

ID	Prueba 5
Descripción	Creación de un fichero nuevo en Google Drive
Entrada	Recibe como entrada dos cadenas. La primera se refiere al titulo que toma como extensión “.txt” y se guarda en la nube como un fichero de texto plano. La segunda es el contenido que llevara el fichero.
Salida esperada	Se obtiene el id del fichero, el cual se espera que no sea nulo.
Resultado	EXITO
Automatizada	Sí
Código	<pre>@Test public void test02NewFile() { String title = "prueba"; item.setTitle(title + ".txt"); item.setMimeType("text/plain"); String content = "probando junit4"; id = rep.newFiles(title, content); assertNotNull(id); } </pre>

ID	Prueba 6
Descripción	Actualizar ficheros de Google Drive
Entrada	Recibe como parámetros en la consulta al recurso el id de un fichero y el contenido del mismo (Nos referimos a modificar ficheros de texto plano).
Salida esperada	Se conserva el id del fichero creado en la anterior prueba y se prueba actualizar. Si se actualiza el contenido correctamente se devuelve un true.
Resultado	EXITO
Automatizada	Sí
Código	<pre>@Test public void test03UpdateFile() { String id = "1V0S2xAFxbjDqCKj3rwOxTl4t6mbMbJ51"; boolean var = rep.updateFile(id,"testing"); assertEquals(var, true); } </pre>

ID	Prueba 7
Descripción	Eliminar algún fichero de Google Drive
Entrada	Recibe como parámetro el id de un determinado fichero
Salida esperada	Se obtiene true o false en caso de que la eliminación se haya o no cumplido.
Resultado	EXITO
Automatizada	Sí
Código	<pre>@Test public void test04DeleteFile() { boolean var = rep.deleteFile(id); assertEquals(var, false); }</pre>

6 Manual de usuario

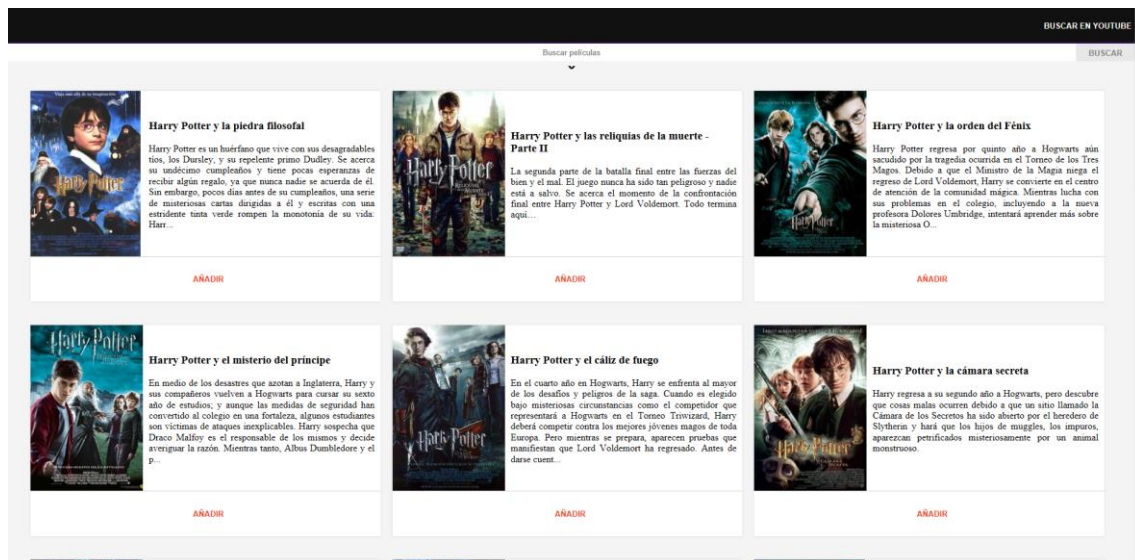
6.1 Mashup



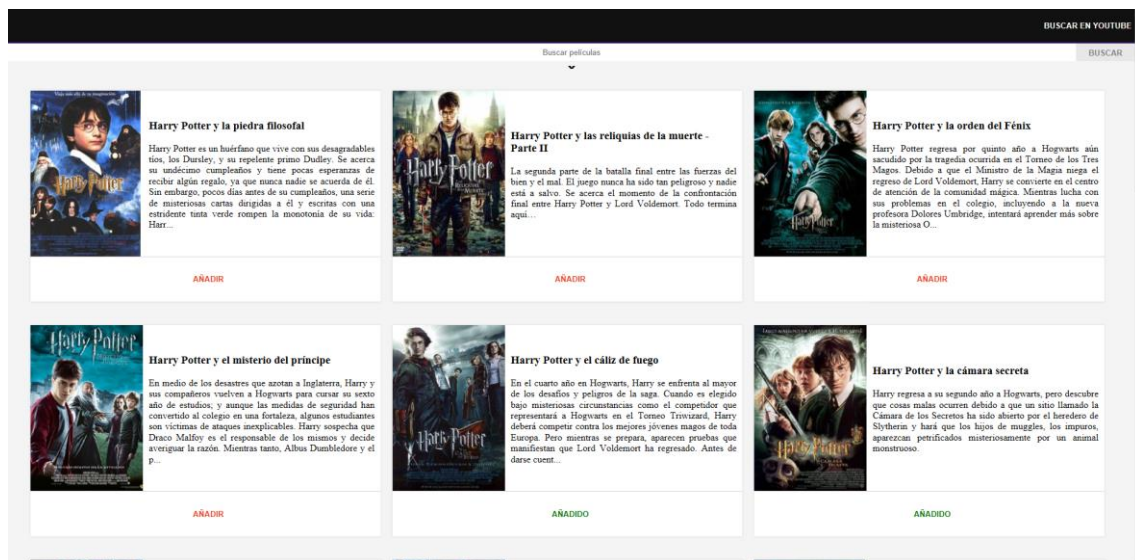
Página de inicio de la aplicación.

Puede realizar búsquedas en TMDb desde el cuadro de búsqueda o seleccionar un archivo de una cuenta de Google Drive desde el enlace que hay bajo el cuadro de búsqueda.

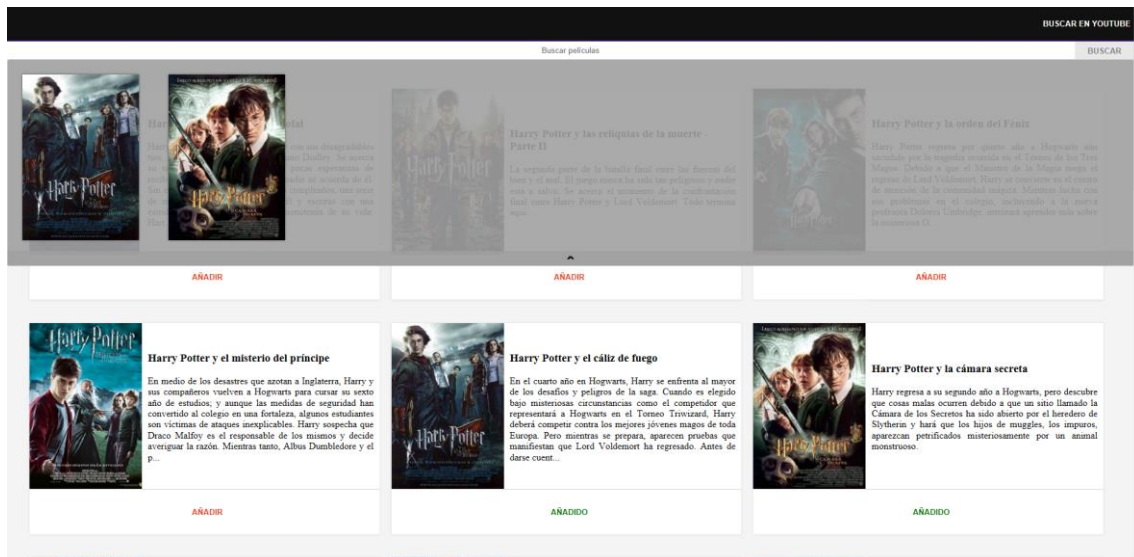
BÚSQUEDA EN TMDB



Al realizar una búsqueda en TMDB aparece la siguiente vista, en la cual se muestran los resultados disponibles. Puede seleccionar una película haciendo click en “AÑADIR”. El botón de selección cambiará su texto entonces a “AÑADIDO”, como se puede ver en la siguiente imagen. Para deseleccionar una película, vuelva a pulsar el botón.

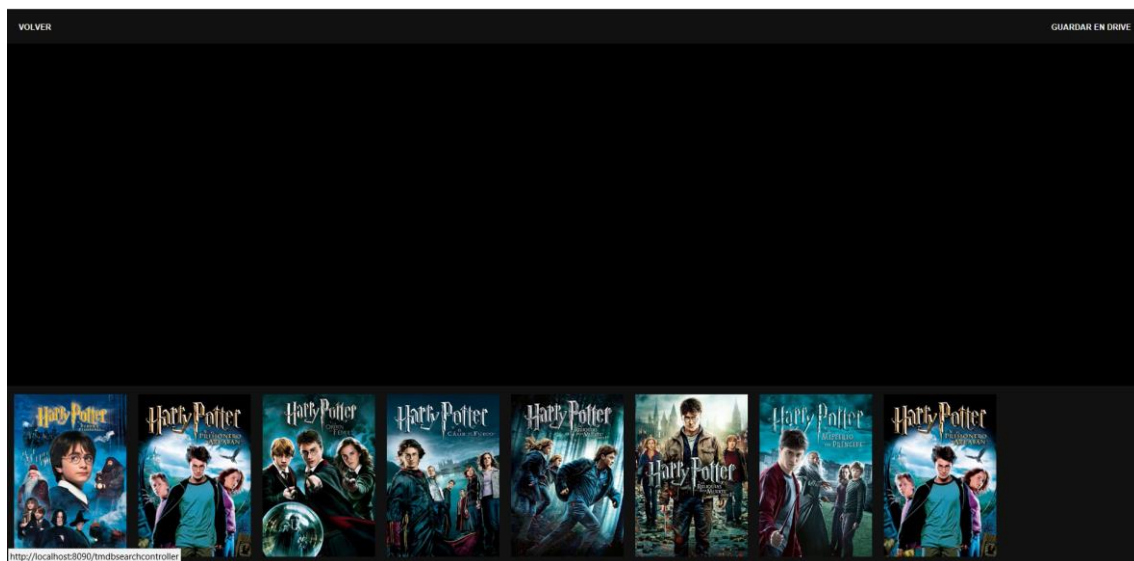


Puede realizar distintas búsquedas en el cuadro de búsqueda superior. Haciendo click en la pestaña que se encuentra justo bajo el cuadro de búsqueda se pueden consultar las películas seleccionadas.

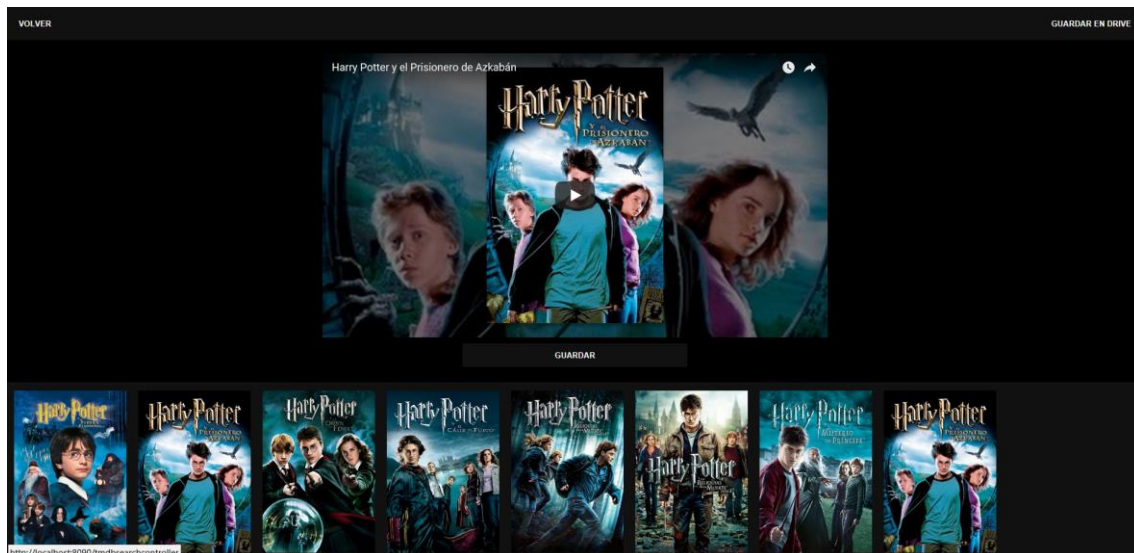


El máximo de películas que se pueden seleccionar son 7, en caso de sobrepasar ese número la aplicación lanza una alerta.

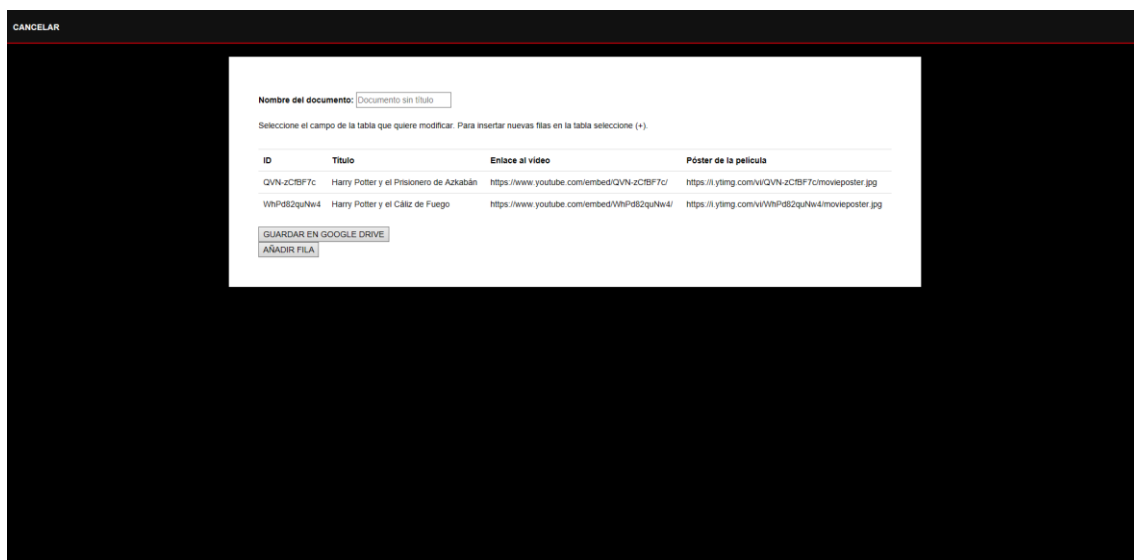
Para realizar las búsquedas en Youtube, pulsar en “BUSCAR EN YOUTUBE” en la esquina superior derecha.



En la vista de selección de Youtube tiene las distintas películas relacionadas con la búsqueda de TMDB. Para visualizar y guardar una, pulse sobre su imagen.



Para guardar la película, pulse el botón “GUARDAR”. Para deseleccionar una película, vuelva a pulsar el botón. Para guardar las películas seleccionadas en Google Drive, pulse “GUARDAR EN GOOGLE DRIVE” en la esquina superior derecha.



Las películas seleccionadas se muestran en una tabla. Cada fila de la tabla contiene:

1. ID de la película. Identificador único de la película en Youtube.
2. Título
3. Enlace al vídeo. Redirige a la página de Youtube.
4. Póster. Enlace al póster asociado a cada película en Youtube.

Todos los campos de la tabla son editables (puede no tener esa opción en su navegador). Para hacerlo seleccione los campos que desee editar arrastrando el ratón.

En caso de tener la opción de editar los campos de texto, también puede añadir otros nuevos seleccionando “AÑADIR FILA”.

Finalmente, para guardar el archivo, introduzca título y seleccione “GUARDAR EN GOOGLE DRIVE”.





CONSULTA DE FICHEROS

[VOLVER](#)

Seleccione el archivo que desea editar

Mostrar todo

Archivos SpotyGo

Nombre	Tamaño	Última modificación	Editar	Eliminar
Star Wars.txt	145	2017-12-11T23:41:56.801Z		
harry potter.txt	160	2017-12-11T23:24:49.446Z		

[Crear un nuevo archivo de texto plano](#)

Desarrolladores

Google Drive - API - Ficheros

TMDB - API

YouTube - API

En la consulta de archivos a Google Drive hay dos opciones de visualización.

1. Mostrar todos los archivos. Muestra todos los archivos almacenados en la cuenta de Google Drive.
2. Archivos SpotyGo. Muestra únicamente los archivos creados en la aplicación.

6.2 API REST

Nota: Hecho desde advanced rest client. En el propio proyecto desplegado damos a mostrar el oauth en la URL para que el cliente pueda hacer uso del mismo para el consumo de la API Rest.

Protocolo GET – GET Files (O Get con paginación)

- URI
 - [http://spotygo2.appspot.com/api/drive/all?oauth=""](http://spotygo2.appspot.com/api/drive/all?oauth=)
- Formato empleado para las representaciones de los recursos
 - @Produces("application/json")
- Codigos de Estado
 - 200 OK
 - 500. Si no recibe correctamente el Access_token.
- Código de programación

```
@GET
@Path("/all")
@Produces("application/json")
public Collection<FileItem> getAll(@QueryParam
("oauth") String oauth, @QueryParam("title")
String title) {
    List<FileItem> res = new ArrayList<>();
    for(FileItem file :
repository.init(oauth).getItems()) {
        if(title==null || title.equals("") ||
title.equals(file.getTitle())) {
            res.add(file);
        }
    }
    return res;
}
```

Protocolo POST – POST Files (Creación de ficheros)

- URI (Recibe un QueryParam para nombrar al fichero, en el campo body de advanced rest client se le indica que su contenido es json y se escribe una prueba en el cuadro gris)
 - [http://spotygo2.appspot.com/api/drive?filename=""&oauth=""](http://spotygo2.appspot.com/api/drive?filename=)
- Formato empleado para las representaciones de los recursos
 - @Consumes("application/json")
 - @Produces("text/plain")
- Codigos de Estado
 - 201 Created
 - 400 Fichero nulo o Contenido del fichero vacio
- Código de programación

```
@POST
@Consumes("application/json")
@Produces("text/plain")
public Response addFile(@Context UriInfo uriInfo,
@QueryParam("filename") String file, String content,
@QueryParam("oauth") String oauth)
{

    if (file == null) {
        return Response.status(400).entity("Fichero nulo!!!").build();
    }

    if (content == null || content == "") {
        return Response.status(400).entity("Contenidovacio!!!").build();
    }

    String id = repository.newFiles(file, content, oauth);
    UriBuilder ub =
    uriInfo.getAbsolutePathBuilder().path(this.getClass(),
    "getAll");
    URI uri = ub.build();
    Response resp = Response.created(uri).entity(id).build();
    return resp;
}
```


Protocolo PUT – PUT Files (Actualización de ficheros)

- URI (Recibe un id por el path)
 - [http://spotygo2.appspot.com/api/drive/id/?oauth="](http://spotygo2.appspot.com/api/drive/id/?oauth=)
- Formato empleado para la representación de los recursos
 - No procede
- Códigos de estado
 - 204 No content (Se actualizó correctamente)
 - 404 Si el id no se encuentra
- Código del método

```
@PUT
@Path("/{id}")
public Response updateFile(@PathParam("id") String id,
    @QueryParam("oauth") String oauth) {
    FileItem oldFile = repository.getFile(id, oauth);
    String oldContent =
repository.getFileContent(oldFile , oauth);
    if (oldFile == null) {
        throw new NotFoundException("El fichero que
se ha intentado modificar no se ha encontrado");
    } else {
        if (oldContent != null) {
            oldContent = "probando";
            repository.updateFile(oldFile.getId(),
oldContent, oauth);
        }
    }

    return Response.noContent().build();
}
```

Protocolo Delete – Delete Files (Actualización de ficheros)

- URI (Recibe un id por el path)
 - [http://spotygo2.appspot.com/api/drive/id/?oauth="](http://spotygo2.appspot.com/api/drive/id/?oauth=)
- Formato empleado para la representación de los recursos
 - No procede
- Códigos de estado
 - 204 No content (Se elimino correctamente)
 - 404 Si el id no se encuentra
- Código del método

```
@DELETE
@Path("/{id}")
public Response removeFile(@PathParam("id") String id,
    @QueryParam("oauth") String oauth) {
    FileItem fileremoved = repository.getFile(id , oauth);
    if (fileremoved == null) {
        throw new NotFoundException("El archivo con el id
deseado no fue encontrada");
    } else {
        repository.deleteFile(id , oauth);
    }
}
```



```

    }
    return Response.noContent().build();
}

```

Request

Method Request URL SEND

GET http://spotygo2.appspot.com/api/drive/all?oauth=ya29.GlwgBRqtm3vmUx8tKtF82IYIB4Kp2_zvppVIW6d8N5N-1P2r

Parameters ^

Headers

Variables



☒ Headers sets

Use predefined sets below to insert headers into the editor

Default headers set

```

accept: application/json
accept-encoding: gzip, deflate
accept-language: en-US,en;q=0.8
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.84 Safari/537.36

```

USE THIS SET



220 bytes

200 OK 3282.07 ms

DETAILS

Method Request URL SEND

POST http://spotygo2.appspot.com/api/drive?filename=prueba&oauth=ya29.GlwgBRqtm3vmUx8tKtF82IYIB4Kp2_zvppVIW

Parameters ^

Headers

Body

Variables



☐ Headers sets

Header name	Header value	
accept-encoding	gzip, deflate	×
accept-language	en-US,en;q=0.8	×
content-type	application/json	×
user-agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.84 Safari/537.36	×
Accept	text/plain	×

ADD HEADER



239 bytes

201 Created 2122.48 ms

DETAILS

Request

Method

Request URL

PUT

http://spotygo2.appspot.com/api/drive/1QFQi7CLuOFNxSsCSP5k9QA9Eb5SD-y7i?oauth=ya29.GlwgBRqtm3vmUx8

SEND

Parameters

Headers

Body

Variables

<>

Headers sets

Use predefined sets below to insert headers into the editor

Default headers set

accept: application/json
accept-encoding: gzip, deflate
accept-language: en-US,en;q=0.8
content-type: application/json
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.84 Safari/537.36

USE THIS SET

220 bytes

204 No Content

1666.17 ms

DETAILS

Request

Method

Request URL

DELETE

http://spotygo2.appspot.com/api/drive/1QFQi7CLuOFNxSsCSP5k9QA9Eb5SD-y7i?oauth=ya29.GlwgBRqtm3vmUx8

SEND

Parameters

Headers

Body

Variables

<>

Headers sets

Use predefined sets below to insert headers into the editor

Default headers set

accept: application/json
accept-encoding: gzip, deflate
accept-language: en-US,en;q=0.8
content-type: application/json
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.84 Safari/537.36

USE THIS SET

220 bytes

204 No Content

1230.40 ms

DETAILS

Referencias

- [1] *Balsamiq*. <http://balsamiq.com/>. Accedido en Enero 2014.
- [2] Microsoft Visio 2016. Microsoft Imagine, Escuela Técnica Superior de Ingeniería Informática
- [3] StackOverFlow. API REST
- [4] Advanced Rest Client – Google Plugin