Universidad
Carlos III de Madrid
www.uc3m.es

ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA

EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

HERRAMIENTA AVANZADA PARA LA COLECCIÓN DE POST Y COMENTARIOS EN FACEBOOK

Nombre y apellidos del alumno: Francisco Javier Torrejón Castillo

Nombre y apellidos del tutor: Ángel Cuevas Rumin

Fecha de entrega: 23/Septiembre/2015

Contenido

1. Introducción y objetivos 1
   1.1. Introducción 1
   1.2. Objetivos 2
   1.3. Motivación 2
1. Introduction and goals 3
   1.1. Introduction 3
   1.2. Goals 3
   1.3. Motivation 4
2. Background 5
   2.1. Facebook para desarrolladores 8
   2.2. API de Facebook 11
   2.3. Librería Facebook4J 13
3. Almacenamiento de la información 18
   3.1. Gson, librería para manejar archivos json 18
   3.2. Base de datos 20
4. Desarrollo de la aplicación 22
   4.1. Aplicación con interfaz java 22
   4.2. Aplicación web 24
5. Funcionamiento interno 29
   5.1. Paquete Manager: 29
   5.2. Clases de la aplicación web 36
6. Evaluación de rendimiento 41
7. Planificación 43
8. Presupuesto 45
   8.1. Recursos utilizados 45
   8.2. Costes 46
9. Mejoras Futuras 47
   9.1. Obtener comentarios de comentarios 47
   9.2. Inclusión en un servidor físico 47
   9.3. Incluir Twitter a la hora de obtener datos 47
   9.4. Análisis de datos 47
10. Conclusiones 48
   10.1. Consecución de objetivos 48
   10.2. Reflexiones personales 48
10. Conclusions 50
   10.1. Achievement of objectives 50

10.2. Personal reflections 50
11. Referencias 52

Índice de Imágenes

Imagen 1. Interfaz de usuario 5
Imagen 2. Interfaz de una página..... 6
Imagen 3: Interfaz de página con código html de la página 7
Imagen 4: Post to Page de una página de Facebook. 8
Imagen 5: Página de Facebook para desarrolladores..... 9
Imagen 6: Instanciando la aplicación 9
Imagen 7: Aplicación ya instanciada..... 10
Imagen 8: Página de la aplicación 10
Imagen 9: Obteniendo el App ID y el App Secret..... 11
Imagen 10: Herramienta Graph API explorer 11
Imagen 11: Documentación del Graph API de Facebook 12
Imagen 12: Página de Facebook4J 13
Imagen 13: Sección Code Examples de Facebook4J 14
Imagen 14: API Support matrix de Facebook4J..... 15
Imagen 15: Javadoc de la librería Facebook4J..... 15
Imagen 16: Javadoc de Facebook4J 16
Imagen 17: Javadoc de gson,. 18
Imagen 18: Imagen de portada de esta versión..... 23
Imagen 19: Formulario de búsqueda avanzada 23
Imagen 20: Formulario de guardar el archivo 24
Imagen 21: Menu principal de la aplicación web..... 26
Imagen 22: Formulario de la opción de búsqueda avanzada..... 27
Imagen 23: Así se muestran los resultados al usuario..... 27
Imagen 24: Pantalla de Búsqueda avanzada de la versión corregida. 28
Imagen 25: Tabla de resultados de pruebas de Búsqueda básica. 42
Imagen 26: Tiempos de las diferentes opciones. 42
Imagen 27: Cuadro con fases y tareas del presente trabajo..... 43
Imagen 28: Diagrama de Gantt..... 44
Imagen 29: Coste de personal interno. 46
Imagen 30: Coste de personal externo. 46
Imagen 31: Coste de los Recursos Materiales. 46

1. Introducción y objetivos

1.1. Introducción

Facebook es, en la actualidad, la red social con mayor número de usuarios activos del mundo. Nos permite comunicarnos con nuestros amigos de siempre o bien conocer nuevas personas con los que compartimos aficiones o intereses. Sin embargo, no es sólo una herramienta para sociabilizar sino que, además, Facebook ha ido evolucionando y ha permitido nuevas herramientas como por ejemplo, las aplicaciones o los juegos. También ha permitido a las empresas crear páginas dentro de la red social. Éstas, pueden servir tanto para informar de productos o servicios que ofrecen las empresas como para informar de noticias relevantes de las mismas, o lo que quizás sea lo más interesante, un acercamiento directo de las empresas a los usuarios, permitiendo una interacción y ruptura de la línea que separa a la empresa del usuario, "humanizando" a esta primera y viendo una posible mejora de servicios los segundos.

Actualmente, las redes sociales han cambiado y revolucionado la manera de interactuar entre las personas, pero no solo han cambiado la forma de sociabilizar, sino que han traído la inmediatez en la información que ha hecho que los medios de comunicación usen las redes sociales como una herramienta y como una fuente más. Para las empresas las redes sociales han cambiado la forma de interactuar con el usuario, puesto que se lo acercan como nunca antes se había podido, convirtiéndose en una poderosa herramienta, dado que se puede obtener una gran cantidad de información y utilizarla para hacer publicidad específica para cada usuario, conocer si el target con el que se lanzó cierto producto acertó o acabó dirigiéndose a otro tipo de consumidores. Está información se puede personalizar para cada país, puede servir de ayuda para ver la evolución de la opinión de los clientes de las empresas de forma temporal. Para una ONG supone estar informado al instante de catástrofes y realizar una mejor gestión. En definitiva, una cantidad innumerable de posibilidades.

En la era de la información, saber utilizarla puede dar una gran ventaja competitiva. Desde hace unos años se creó el concepto de big data, es decir, recopilar grandes masas de datos, procesarlos y aprovechar los conocimientos que nos den, ya sean predicciones o información pura. Por tanto, las empresas que hagan un buen uso de la información tendrán una gran ventaja competitiva y poseer esa información tiene un gran valor. El vicepresidente de Oracle Europa, Malhar Kamdar, durante la presentación de un centro de big data en Barcelona dijo que: "*el big data es hoy lo que la electricidad fue en el siglo .XX: una gran oportunidad para que las empresas sean eficientes*" (Portolés, 2015).

1.2. Objetivos

El objetivo principal de este trabajo fin de Grado es la realización de una herramienta que obtenga de forma simple y rápida post y comentarios asociados de páginas de Facebook para su posterior procesado en diferentes aplicaciones.

Por consiguiente, para llegar a la realización del objetivo principal, antes debemos establecer unos objetivos parciales:

- Conocer la interfaz de las páginas de Facebook y qué información nos interesaría obtener.
- Conocer el Graph API de Facebook y que estructura tiene la información obtenida.
- Estudiar y aprender el uso de la librería de java Facebook4J que será el principal apoyo de nuestra aplicación.
- Buscar y aprender a usar librerías para crear archivos json y poder exportar la información que busquemos con nuestra aplicación.
- Hacer un uso correcto de una base de datos para almacenar nuestra información y saber gestionarla.
- Desarrollar una interfaz sencilla en una aplicación web para aplicarla a nuestro proyecto.
- Corregir errores y salvaguardar nuestra aplicación de posibles ataques informáticos.

1.3. Motivación

La motivación principal para realizar este trabajo fin de Grado es buscar implementar una aplicación externa a Facebook que permita hacer una búsqueda detallada en tiempo y en naturaleza de los comentarios, de la información de una o más páginas particulares, permitiendo además almacenar esta información en archivos externos o en una base de datos, facilitando su posterior uso y agilizando el acceso a la información. La aplicación contará con una interfaz simple e intuitiva para usuarios de Facebook. En definitiva, crear una solución sencilla a uno de los problemas de big data que es la obtención de datos.

Esta solución se integrará en el Proyecto Nacional “MECANICA ESTADISTICA PARA "BIG DATA": ADQUISICION, ANALISIS Y MODELIZACION”[1], por lo que también es una gran motivación para la realización del presente trabajo.

Otra motivación es aprender el uso de nuevas librerías para java y crear una aplicación que pueda tener un uso efectivo en un mundo donde el uso de la información está en auge y que con las mejoras pertinentes se pueda crear una aplicación muy útil.

[1] "Comunicación del programa estatal de fomento de la investigación científica y técnica de excelencia.".FIS2013-47532-C3-3-P Ministerio de Economía y Competitividad

1. Introduction and goals

1.1. Introduction

Today, Facebook is the social network with the largest number of active users of the world. It allows us to communicate with our friends or meeting new people with which we share hobbies or interests. However, it is not only a tool for socializing, Facebook has evolved and it has allowed new tools like applications or games. It also has allowed companies to create pages within the social network. These pages can serve both to inform about products or services offered by companies as to report relevant news of them, or, what is perhaps the most interesting thing, a direct approach of companies to users, allowing interaction between the company and the user, "humanizing" the first one and seeing a possible improvement of services in the second one.

Currently, social networks have changed and revolutionized the way to interact with people, but haven´t only changed the way of socializing, social networks have brought immediacy in information, that´s why the media are using them as a tool and as one source more of information. For companies, social networks have changed the way of interact with the user, becoming a powerful tool, now, they can get a lot of information and use it for specific advertising to each user, know whether the target with which certain product was launched was right or ended up going to another type of consumer. This information can be customized for each country and can serve as a support to analyze the evolution of the perception of business customers. For an NGO, this information means keep informed immediately about catastrophes. In short, a countless number of possibilities.

In the age of information, how to use it can give a great competitive advantage. Since a few years ago the concept of big data was created, that means, collect large masses of data, process them and seize the knowledge that give us, whether they are predictions or pure information. Therefore, companies that make good use of the information will have a great competitive advantage and own that information is valuable. The Vice-President of Oracle Europe, Malhar Kamdar, during the presentation of a big center data in Barcelona said that: "the big data is today what electricity was in the 20th century: a great opportunity to make companies efficient" (Portolés, 2015).

1.2. Goals

The main goal of this project is the realization of an easy to use tool that gets post and associated messages of Facebook´s pages quickly for subsequent processing in various applications.

Therefore, in order to achieve the main objective, we must establish partial goals:

- To know the interface of Facebook pages and what information we want obtain.
- To know Facebook´s Graph API and his information structure.

- Study and learn uses of java´s library Facebook4J.
- Search and learn how create json files and how use them into the application.
- Use properly a database and how we store our information.
- Develop an easy to use user interface for the application.
- Fix errors and prevent computer attacks to the application.

1.3. Motivation

The main motivation of this project is to implement an external application to Facebook that allow us to search detailed in time and nature of comments, the information of one or more particular pages, and store this information in external files or in a database. The application will have a simple and intuitive interface for Facebook users. In short, create a simple solution to one of the problems of big data, which is obtain the data.

This solution will be integrated in the national project “MECANICA ESTADISTICA PARA "BIG DATA": ADQUISICION, ANALISIS Y MODELIZACION”[2]. It is also a great motivation for the realization of this work.

Another motivation is to learn the use of new libraries for java and create an application that may have an effective use in a world where the use of information is growing and with relevant improvements, create a very useful application.

[2] "Comunicación del programa estatal de fomento de la investigación científica y técnica de excelencia.".FIS2013-47532-C3-3-P Ministerio de Economía y Competitividad

2. Background

Para entender la aplicación y qué datos queremos obtener y por qué. Primero, hay que entender el entorno del usuario de Facebook[3] y más específicamente su representación en las páginas de las diferentes empresas.

Dado que Facebook es la red social con mayor número de usuarios es muy posible que la interfaz ya sea muy trivial para las personas familiarizadas con la página.

En la siguiente imagen se muestra una captura de la interfaz de la página de perfil del usuario y explicaremos las partes más importantes para la aplicación:

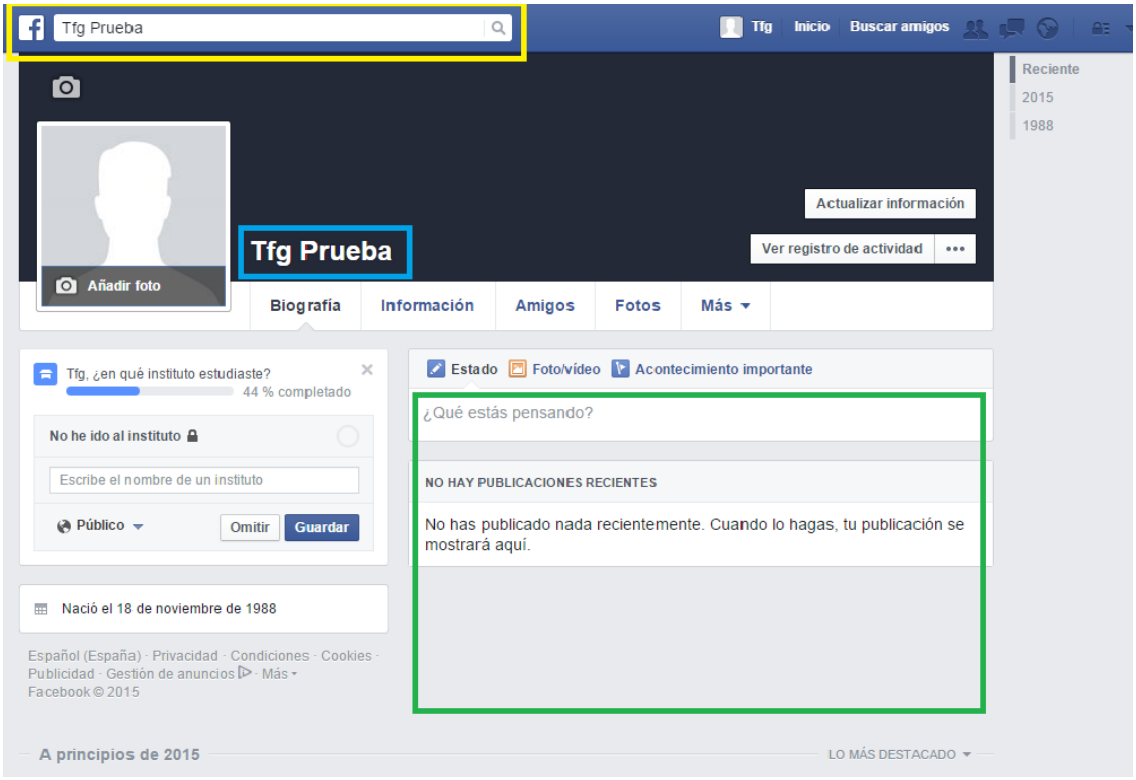


Imagen 1. Interfaz de usuario (FUENTE: Facebook)

En el cuadrado de color azul nos encontramos con el nombre de usuario, es el nombre por el que te ven los demás usuarios y por el que te ven las empresas en sus páginas. A cada nombre de usuario va anexado un número de identificación propio que es único en Facebook y desconocido para el propio usuario. Este identificador tiene la función de diferenciar usuarios con el mismo nombre.

En el cuadrado de color verde estamos ante el tablón de actividad reciente. En este apartado de la página se muestran las actividades del usuario sobre sí mismo como, por ejemplo, la actualización de los estados o los cambios de las fotos de perfil. Además, también se muestran post compartidos de otras páginas. Cuando un post hace referencia a una noticia, producto o servicio de una página de una empresa en el que el

[3] www.facebook.com

usuario lo haya compartido, en muchas ocasiones, es indicativo para la empresa de que el post ha sido relevante. Como veremos más adelante, este tipo de información también se puede obtener con la aplicación.

Finalmente, en el cuadrado de color amarillo nos encontramos con el buscador. Este apartado de la página nos sirve para buscar a otros usuarios, eventos, lugares y páginas, siendo esto último lo interesante para el propósito de este trabajo.

Ahora mostraremos las páginas de empresa y como utilizarlas para el proyecto y la información que podemos obtener.

Al buscar la página del equipo de fútbol Real Madrid nos encontramos con lo siguiente:

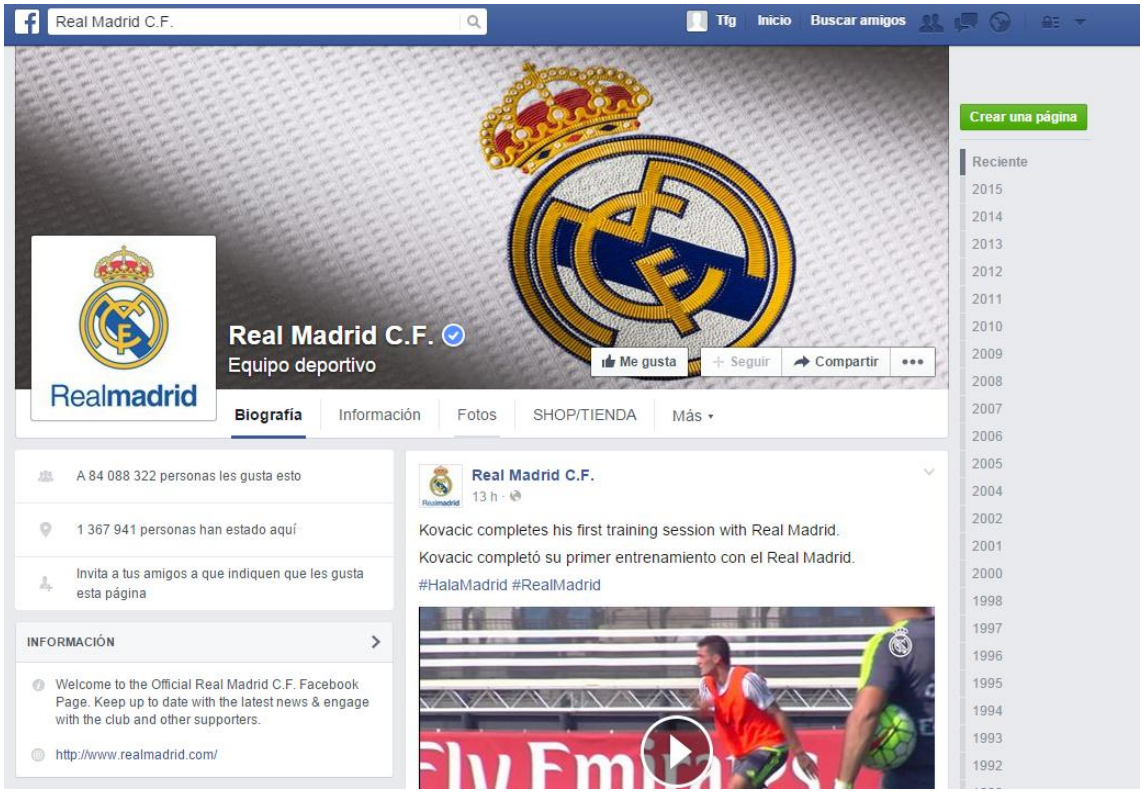


Imagen 2. Interfaz de una página (FUENTE: Facebook)

Podemos comprobar que la página se parece mucho a la página de perfil de usuario mostrada anteriormente. La página tiene su nombre propio, Real Madrid C.F., además de un tablón donde se muestran las actividades recientes de la página que se diferencia de la del perfil de usuario en que únicamente se muestran los post creados por la página. Como veremos más adelante estos post reciben un nombre específico.

Un dato muy importante para la aplicación es que, como se muestra en la siguiente imagen, el nombre de la página, además de tener un número de identificación único, también tiene un nombre único de uso interno en Facebook. Este nombre puede ser igual o distinto al nombre de la página y será el que utilizaremos para las peticiones en nuestra aplicación.

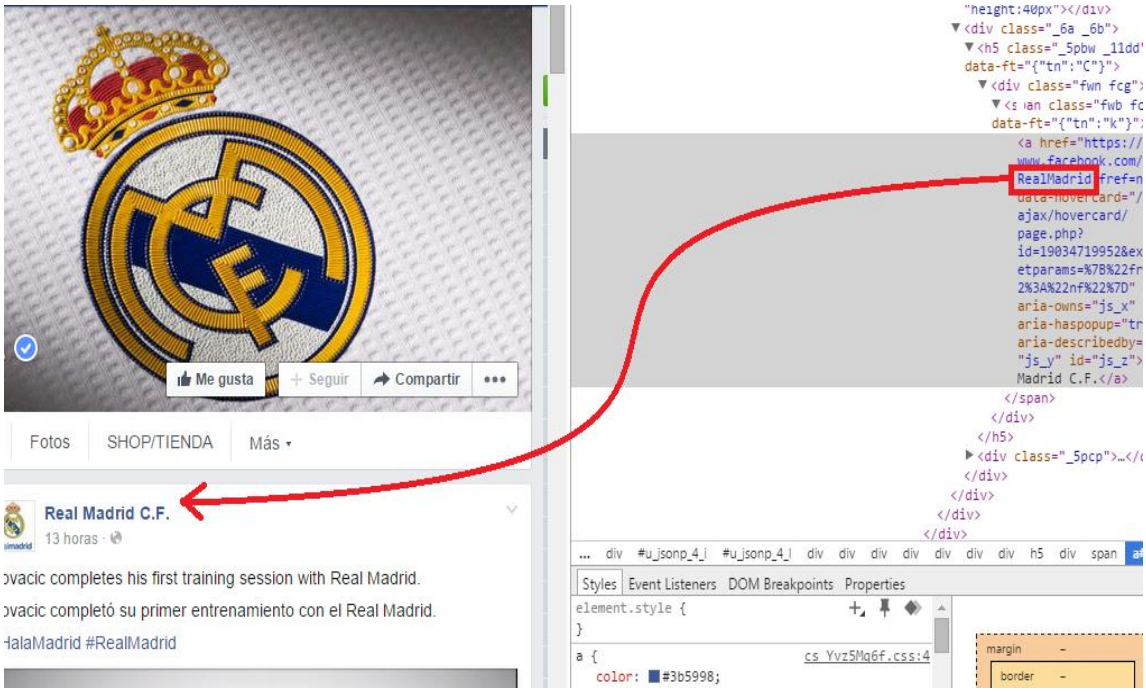


Imagen 3: Interfaz de página con código html de la página (FUENTE: Facebook)

Como vemos en la imagen, pese a que el nombre de la página es Real Madrid C.F., si inspeccionamos el código de la página vemos que el hipervínculo que tiene el nombre de la página hace referencia al nombre RealMadrid que a efectos internos es distinto pero es el nombre a utilizar para obtener información de la página a la hora de hacer las peticiones a Facebook.

Además del tablón de post propios como hemos visto, las páginas de Facebook también tienen un apartado donde personas externas pueden poner sus propios post. A los post propios de la página que se muestran en el tablón reciben el nombre de PagePost y los post de las personas externas reciben el nombre de Post to Page. Estos PagePost y Post to Page junto con las respuestas que reciben que tienen el nombre de Comment conforman la información más importante que podemos obtener para nuestros intereses y es el principal objetivo de la aplicación que tratamos en este trabajo.

Imagen 4: Post to Page de una página de Facebook. (FUENTE: Facebook)

Como vemos en esta imagen, las Post to Page se colocan a la izquierda de la página. Dependiendo del trato que quiera dar la empresa a sus usuarios pueden tener mayor o menor importancia los Post to Page. Por ejemplo, en la página de Iberia Líneas Aéreas, los Post to Page tienen gran importancia porque sirven para responder a los usuarios las dudas sobre vuelos, horarios y precios, así como para la gestión de las reclamaciones o incidencias de los usuarios.

2.1. Facebook para desarrolladores

Para el desarrollo de la aplicación fue necesario crearse una cuenta como desarrollador en Facebook[4], para ello es necesario dar datos personales por lo que no se mostraran capturas de pantalla.

developers.facebook.com

Una vez creada la cuenta nos encontramos con la siguiente pantalla:

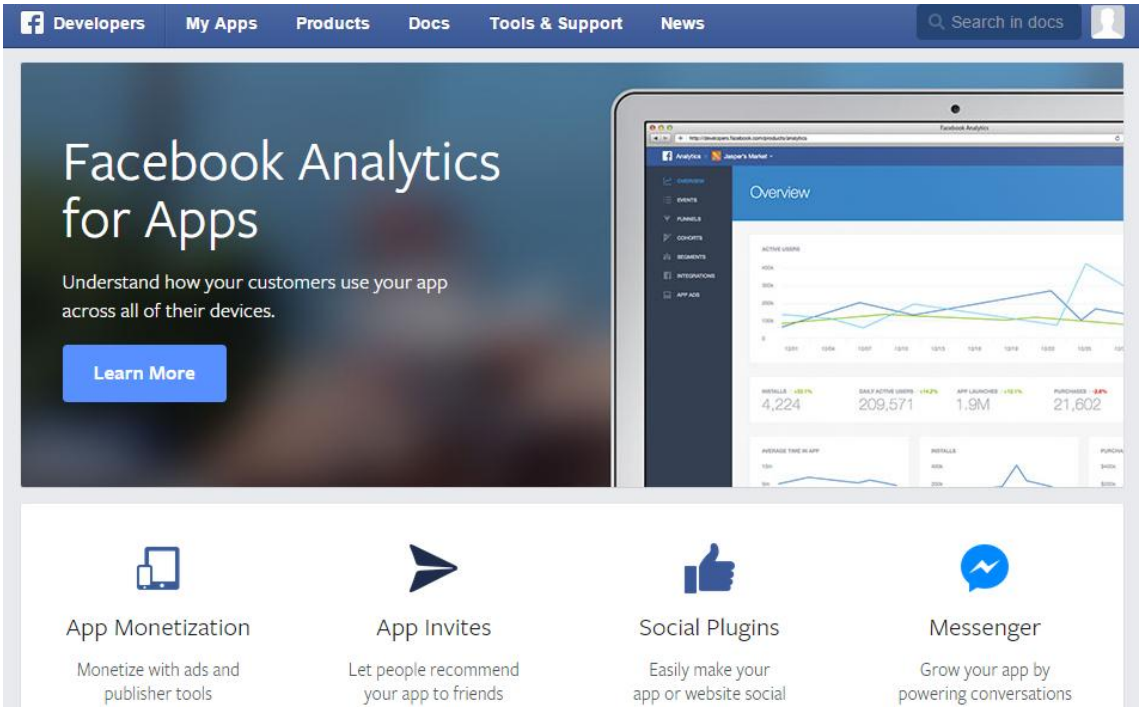


Imagen 5: Página de Facebook para desarrolladores (FUENTE: Facebook)

Para instanciar la aplicación hay que seleccionar My Apps y Add a new App, una vez seleccionado, nos aparecerá una pantalla para elegir donde se alojará la aplicación, en nuestro caso Sitio web.

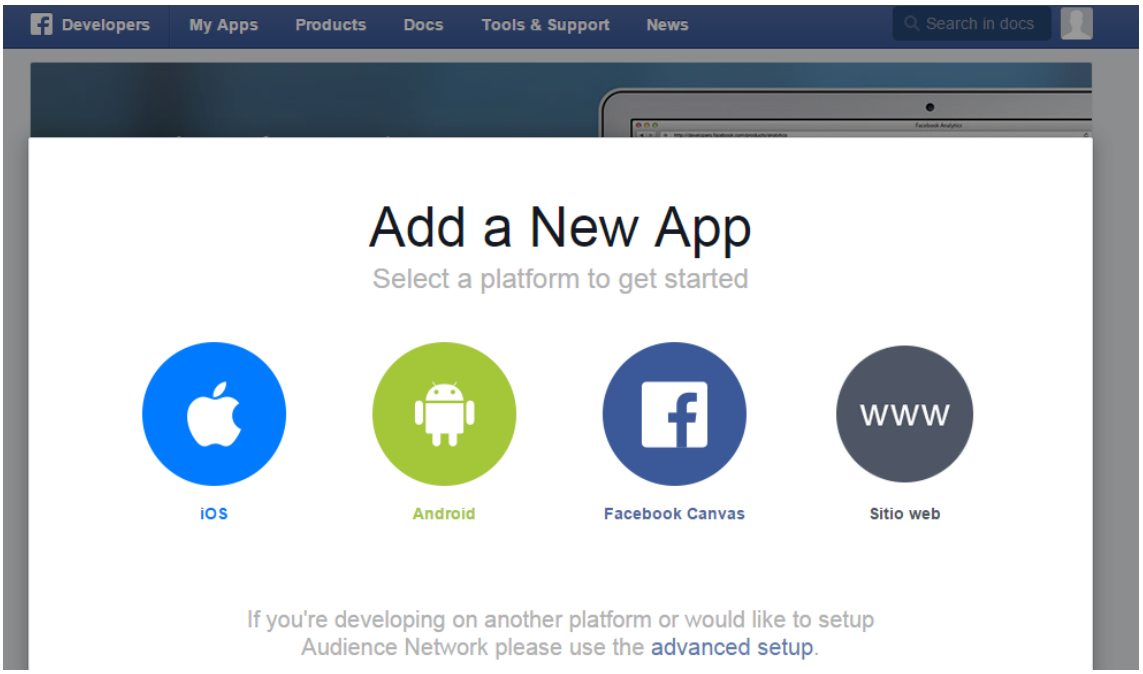


Imagen 6: Instanciando la aplicación (FUENTE: Facebook)

Le damos un nombre a nuestra aplicación y elegimos categoría. Para ésta aplicación se eligió el nombre de Big Page Application y en categoría Aplicaciones para Páginas.

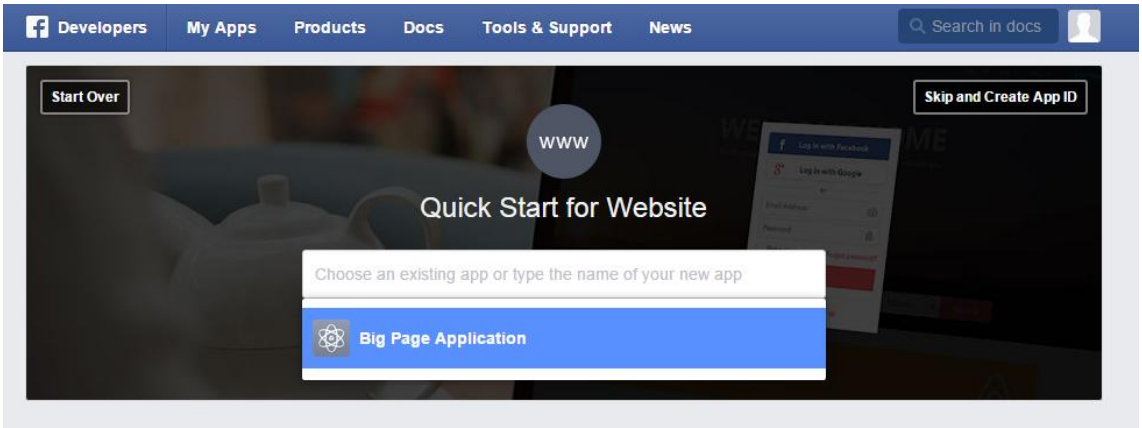


Imagen 7: Aplicación ya instanciada (FUENTE: Facebook)

Una vez instanciada, nos interesa obtener el token de aplicación, el App ID y el App Secret para poder empezar el desarrollo. Para obtener el token de aplicación seleccionamos Tools & Support y a continuación Access Token Tool. Aparecerán dos apartados, primero User Token que para obtenerlo es necesario dar permisos pero que no se va a utilizar en nuestra aplicación y segundo App Token, donde se muestra el token de nuestra aplicación y que será necesario para autenticar nuestra aplicación de cara a realizar peticiones a Facebook. Para obtener el App ID y el App Secret nos introducimos en la página principal de nuestra aplicación a través de My Apps. Una vez en la página principal de la aplicación hay que seleccionar el apartado Settings donde se visualizará el App ID de forma directa y el App Secret oculto, al pulsar el botón show del mismo y tras introducir la contraseña de la cuenta, se mostrará. Con estos tres elementos ya tenemos lo necesario para poder realizar peticiones a Facebook y, por consiguiente, comenzar el desarrollo de la aplicación.

Imagen 8: Página de la aplicación (FUENTE: Facebook)

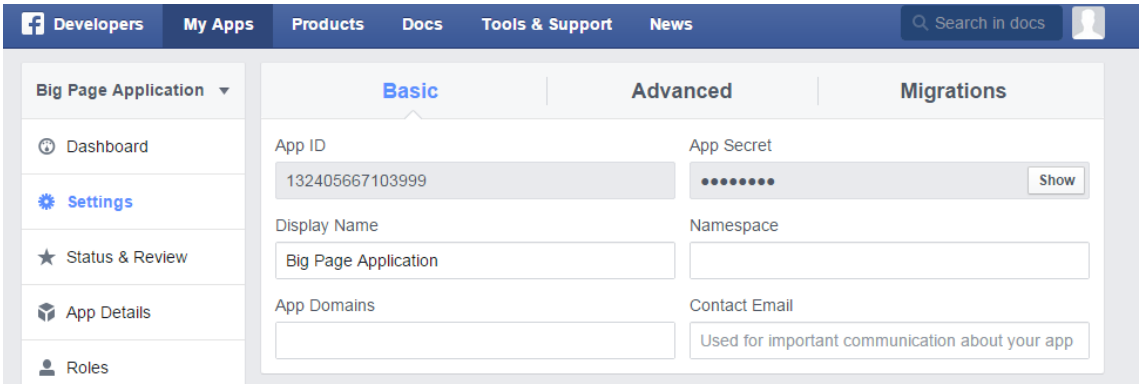


Imagen 9: Obteniendo el App ID y el App Secret (FUENTE: Facebook)

2.2. API de Facebook

Como hemos visto, previamente hay que crearse una cuenta como desarrollador para Facebook. En esta cuenta, puede crearse la instancia de una aplicación y desde este punto, Facebook proporciona tokens de aplicación y usuario para la autentificación a la hora de conectarse desde la aplicación en desarrollo, siendo necesario el token de usuario en caso de necesidad de escritura. En ese caso, en la página de Facebook vendrán reflejados las modificaciones realizadas desde la aplicación y qué usuario (dado por el token de usuario) ha realizado la acción.

Desde la página de Facebook para desarrolladores se puede acceder a una herramienta de gran utilidad desde la pestaña Tools & Support, la herramienta Graph API Explorer. Esta herramienta nos permite hacer peticiones a Facebook directamente utilizando el API de la misma red social.

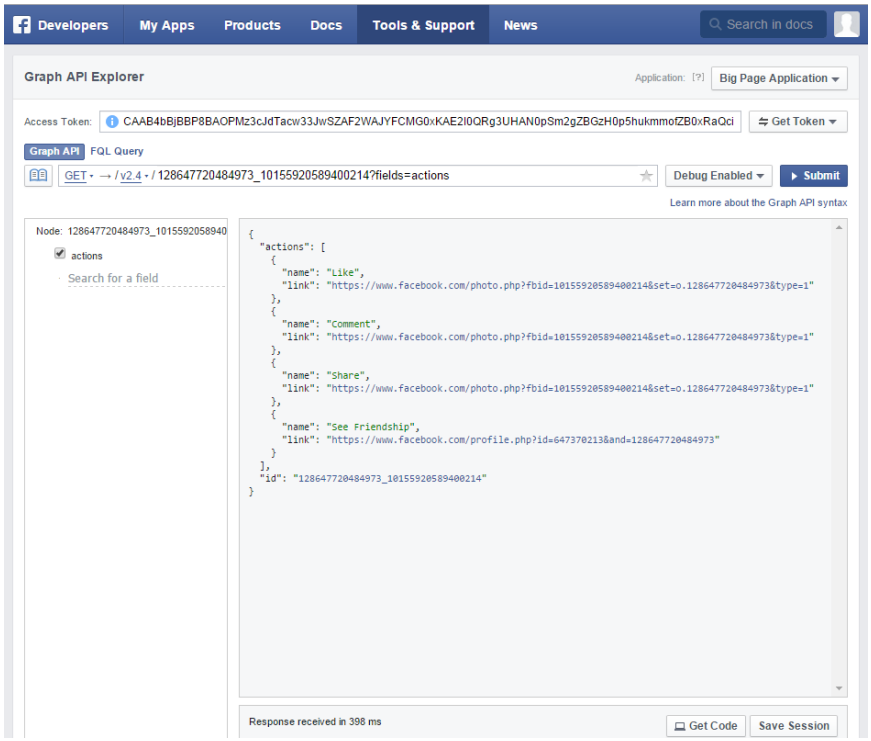


Imagen 10: Herramienta Graph API explorer (FUENTE: Facebook)

Para realizar las peticiones es necesario tener un token de usuario o de aplicación válido, que ya hemos visto en el apartado Facebook para desarrolladores como se obtienen. Las peticiones pueden ser de tipo GET, es decir, se pide información a Facebook; de tipo POST, en el que el usuario coloca la información en Facebook; o de tipo DELETE, que elimina la información de Facebook. Como sólo queremos obtener información, nos centraremos en las funciones GET, que únicamente necesitan el token de aplicación para su funcionamiento, siendo necesario un token de usuario para las funciones POST y DELETE. De esta forma, seremos invisibles para el resto de usuarios en Facebook.

Para conocer la sintaxis de las peticiones es necesario mirar la documentación que proporciona la página de Facebook para desarrolladores en la pestaña Docs y la sección Graph API[5]. Aquí, podemos encontrar como buscar la información de los elementos de Facebook que queremos obtener a partir de la herramienta.

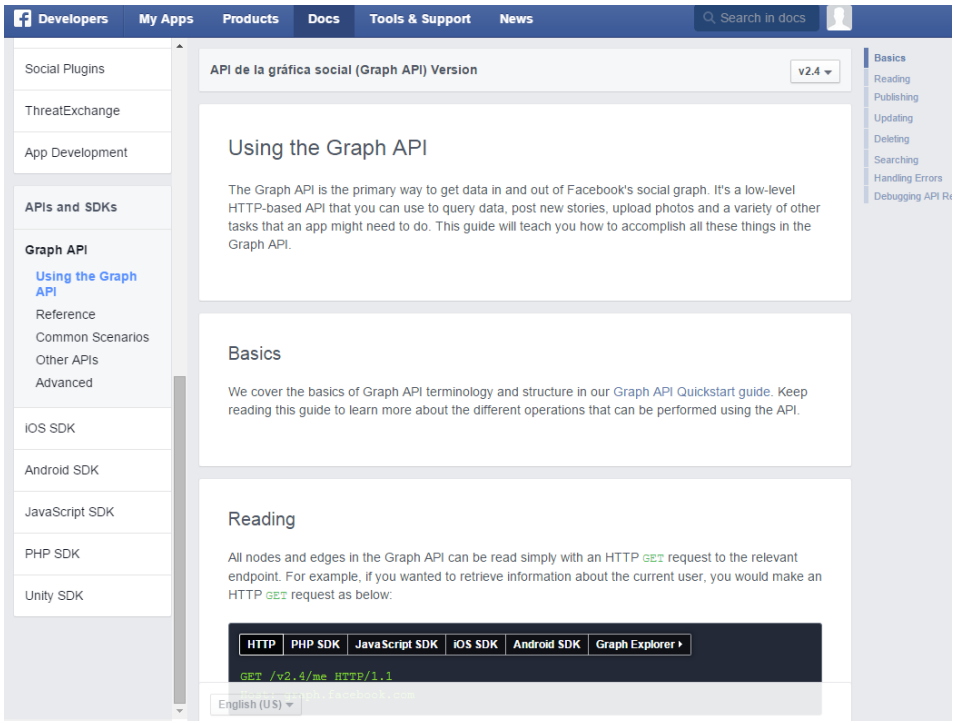


Imagen 11: Documentación del Graph API de Facebook (FUENTE: Facebook)

Como veremos en el siguiente apartado, saber utilizar la herramienta Graph API Explorer correctamente será muy importante para conocer que peticiones realizamos cuando programamos la aplicación, que como veremos, haremos uso de la librería Facebook4J.

[5]developers.facebook.com/docs/graph-api/using-graph-api/v2.4

2.3. Librería Facebook4J

Para el desarrollo de la aplicación se utilizó la librería no oficial Facebook4J para integrar el API de Facebook, dado que Facebook no da soporte oficial para java aunque lo dé para otros lenguajes como, por ejemplo, javascript. Esta librería reúne funciones de conexión con Facebook, así como de búsqueda y escritura.

Facebook4J[6] es una librería no oficial creada por Ryuji Yamashita cuya principal característica es que es 100% java y no necesita de otras librerías para funcionar. Además, como indica la página principal, utiliza software de otra librería llamada Twitter4J[7], similar a la que estamos tratando pero para la red social Twitter[8].

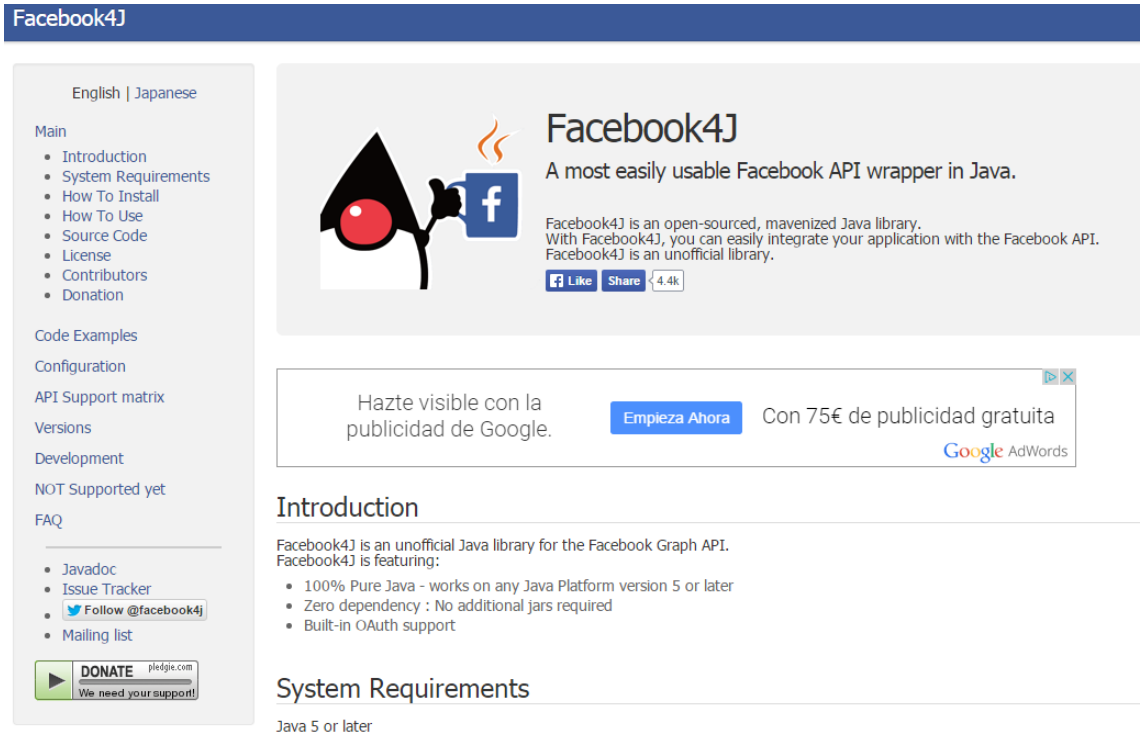


Imagen 12: Página de Facebook4J (FUENTE: Facebook4J)

La instalación es tan sencilla como agregar el archivo jar al classpath del proyecto o, en nuestro caso, como la aplicación acabó siendo una aplicación web instalada en un servidor Apache-Tomcat[9], el archivo jar se colocará en la carpeta lib de nuestra aplicación.

La página de la librería tiene varias secciones en las que se nos indica: donde se descarga la librería, como configurarla en caso de querer modificar los parámetros por defecto, las versiones de la librería y otras secciones. Sin embargo, las dos secciones más importantes son: Code Examples y API Support matrix.

[6] facebook4j.org/en/index.html
[7] http://twitter4j.org/en/index.html#license
[8] https://twitter.com
[9] http://tomcat.apache.org/

En Code Examples nos encontramos ejemplos de código java para realizar diferentes acciones en Facebook utilizando la librería. Esta sección es muy importante porque es muy complicado encontrar ejemplos de uso de esta librería y si se encuentran son muy similares a los que encontramos en este apartado, por lo que nos proporciona una base para poder investigar la librería.

Imagen 13: Sección Code Examples de Facebook4J (FUENTE: Facebook4J)

API Support matrix nos enseña varias tablas de elementos de Facebook que relaciona el API de Facebook con funciones de la librería de Facebook4J, además indica si la petición que realiza es de tipo GET, POST o DELETE, exactamente igual que en la herramienta Graph API Explorer de la página de Facebook para desarrolladores. Por último, debajo de cada elemento se encuentran enlaces al API de Facebook para un mayor entendimiento de cada función.

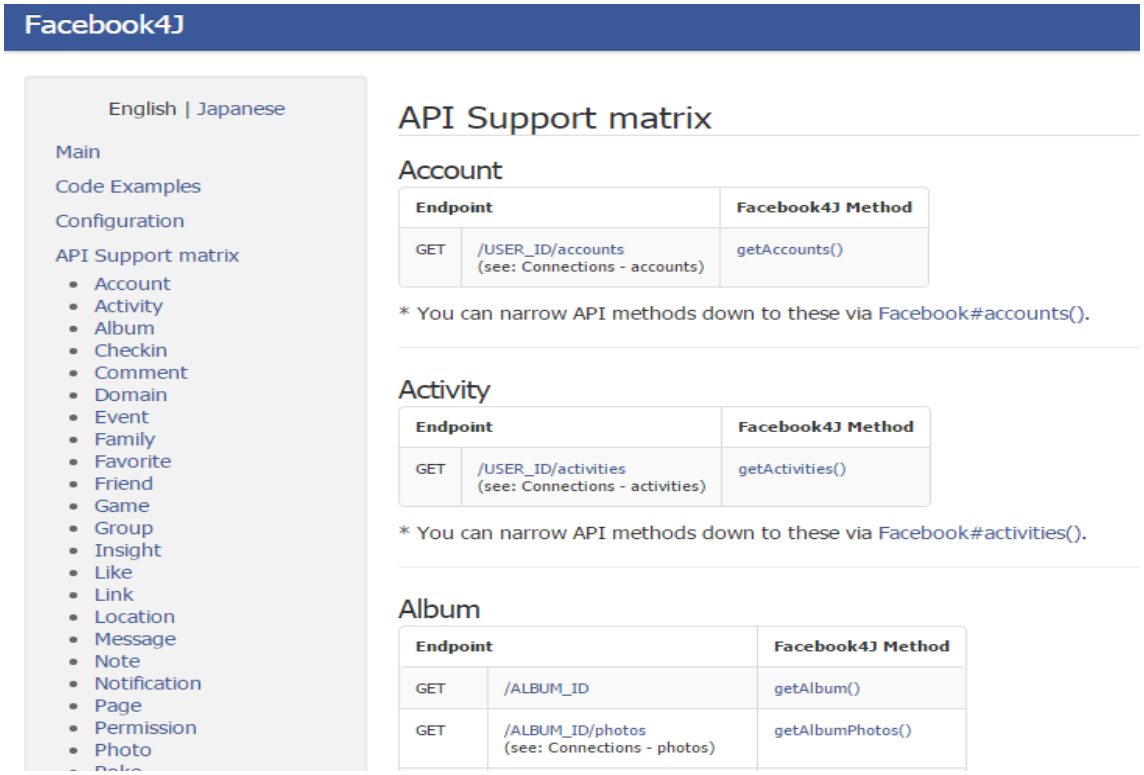


Imagen 14: API Support matrix de Facebook4J (FUENTE: Facebook4J)

Finalmente, la página da acceso a una página javadoc[10] con la librería desglosada por clases, funciones y objetos, por lo que es de mucha utilidad a la hora de programar. Un gran error es que varios de los métodos no tienen una descripción de lo que hacen. Si bien la mayoría de los métodos vienen explicados por encima para saber cuáles son los parámetros que leen y cuáles son los que devuelven, es necesario tener siempre el API de Facebook a mano y realizar varios intentos con el código hasta conseguir el objetivo que se pretende.

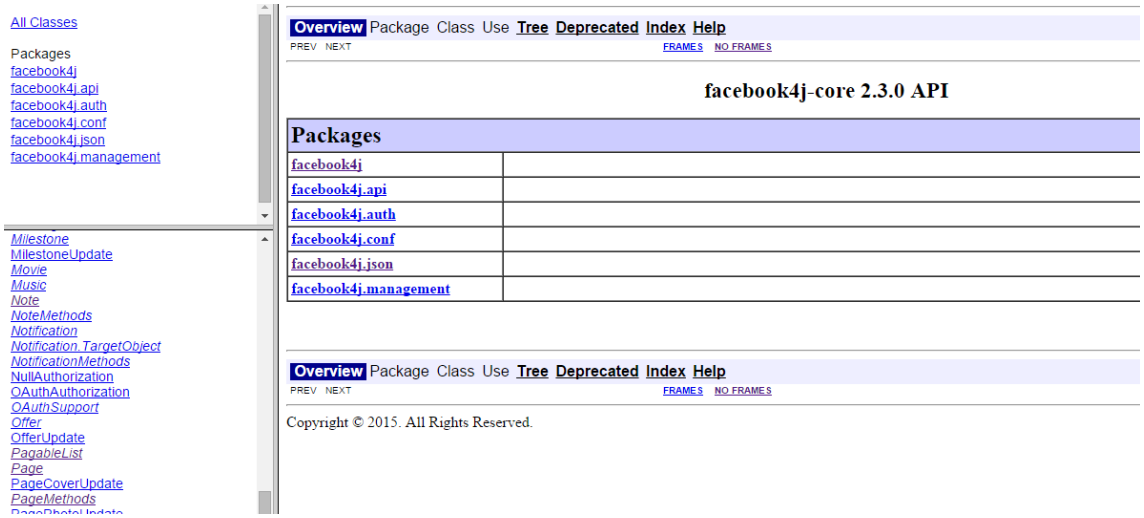


Imagen 15: Javadoc de la librería Facebook4J (FUENTE: Facebook4J)

[10] http://facebook4j.org/javadoc/index.html

Method Summary	
List<Post.Action>	getActions()
Application	getApplication()
String	getCaption()
PagableList<Comment>	getComments()
Date	getCreatedTime()
String	getDescription()
Category	getFrom()
URL	getFullPicture()
URL	getIcon()

Imagen 16: Javadoc de Facebook4J(FUENTE: Facebook4J)

En la imagen anterior podemos ver como los métodos de la clase Post no vienen explicados, siendo necesario buscar en el Graph API de Facebook.

A pesar de tener una información escasa, se realizaron varias pruebas en java como ensayo-error para ir observando cómo obtener la información que queríamos conseguir de las páginas, que como hemos visto en Background, son los Post to Page y los PagePost. Para las pruebas fue necesario usar la herramienta Graph API Explorer para analizar los elementos de la información a obtener que pudiese ser de ayuda. Algunos ejemplos de código de la sección Code Examples de la página de la librería sirvieron de punto de partida para las pruebas.

A raíz de las pruebas y aunque se emplearon varias funciones y clases proporcionadas por la librería; son cuatro clases las que, con sus métodos, son las encargadas de la mayoría de las funciones de la aplicación:

- Facebook: es la clase principal de la librería que incluye todas las funciones necesarias para la conexión y desconexión. Además, mantiene la sesión de la aplicación y a partir de ahí, realiza el resto de instrucciones de la aplicación.

- Post: es la clase que incluye las funciones necesarias para almacenar y tratar los post de las páginas que queremos buscar.

- Comment: es la clase necesaria para almacenar y tratar los comentarios de respuesta a los post anteriormente obtenidos.

- Reading: es la clase de apoyo a las dos anteriores. Guarda la información de los parámetros de búsqueda relacionados con el espaciado temporal y el número de post y comentarios a obtener.

Con estas cuatro clases obtenemos la información de las páginas que nosotros deseamos como: la fecha de creación de los mensajes, el id del mensaje, el creador de dicho mensaje entre otros. Por consiguiente, debemos abarcar el problema de cómo guardar dicha información.

3. Almacenamiento de la información

Necesitamos que la información buscada sea fácil de guardar, procesar y acceder. Por ello, se decidió almacenar la información de dos maneras: mediante archivos json (JavaScript Object Notation) y almacenando la información en una base de datos.

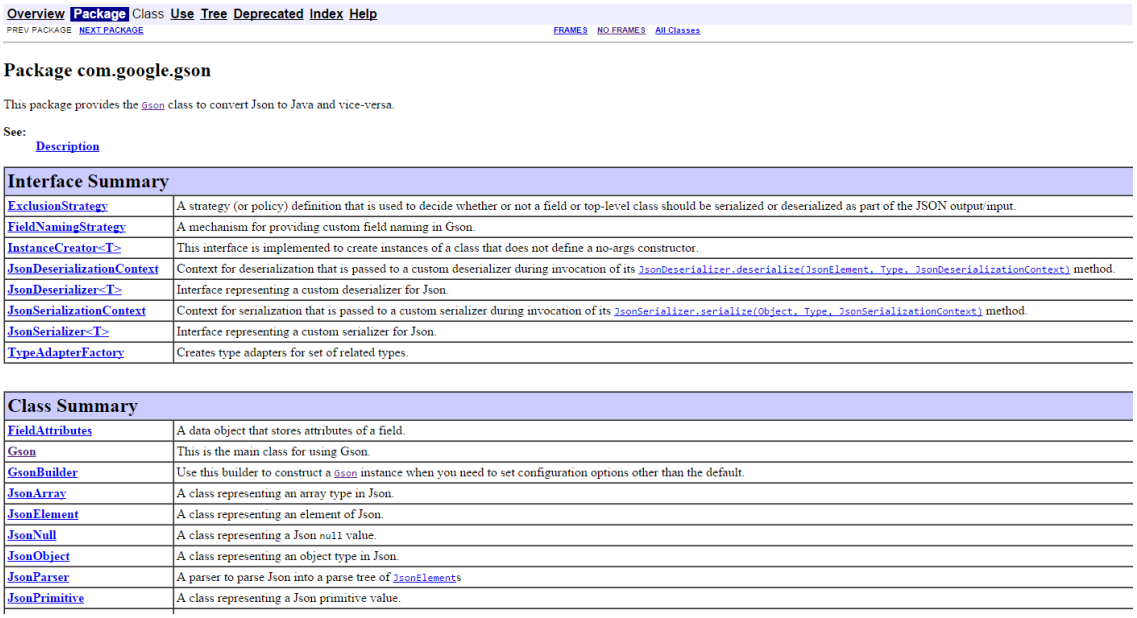
En los siguientes apartados razonaremos el uso de las dos formas de almacenamiento y como se han realizado.

3.1. Gson, librería para manejar archivos json

Se decidió el uso de json debido a la sencillez de su estructura, su aceptación y su fácil implementación en un analizador o en otro lenguaje de programación como javascript, que tiene soporte directo.

Para su implementación en el proyecto, utilizamos la librería gson de Google, una librería muy sencilla con funciones directas que transforman objetos java a texto en formato json y viceversa.

A través de su javadoc[11], vemos la sencillez de uso de sus métodos a través de una página bien estructurada y explicada.



Overview Package Class Use Tree Deprecated Index Help

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES All Classes

Package com.google.gson

This package provides the Gson class to convert Json to Java and vice-versa.

See:
Description

Interface Summary	
ExclusionStrategy	A strategy (or policy) definition that is used to decide whether or not a field or top-level class should be serialized or deserialized as part of the JSON output/input.
FieldNamingStrategy	A mechanism for providing custom field naming in Gson.
InstanceCreator<T>	This interface is implemented to create instances of a class that does not define a no-args constructor.
JsonDeserializationContext	Context for deserialization that is passed to a custom deserializer during invocation of its JsonDeserializer.deserialize(JsonElement, Type, JsonDeserializationContext) method.
JsonDeserializer<T>	Interface representing a custom deserializer for Json.
JsonSerializationContext	Context for serialization that is passed to a custom serializer during invocation of its JsonSerializer.serialize(Object, Type, JsonSerializationContext) method.
JsonSerializer<T>	Interface representing a custom serializer for Json.
TypeAdapterFactory	Creates type adapters for set of related types.

Class Summary	
FieldAttributes	A data object that stores attributes of a field.
Gson	This is the main class for using Gson.
GsonBuilder	Use this builder to construct a Gson instance when you need to set configuration options other than the default.
JsonArray	A class representing an array type in Json.
JsonElement	A class representing an element of Json.
JsonNull	A class representing a Json null value.
JsonObject	A class representing an object type in Json.
JsonParser	A parser to parse Json into a parse tree of JsonElements
JsonPrimitive	A class representing a Json primitive value.

Imagen 17: Javadoc de gson,. (FUENTE: Google)

Como veremos en el capítulo Funcionamiento interno guardaremos nuestra información en una lista de objetos java. Gracias a la librería gson simplemente con la

[11] google-gson.googlecode.com/svn/trunk/gson/docs/javadocs/com/google/gson/package-summary.html

función toJson obtenemos una cadena de texto directamente en formato json de esa lista de objetos. Después, para guardarlo en un archivo, utilizamos las funciones de java para el tratamiento de archivos.

Como también nos interesará poder leer la información de esos archivos emplearemos las funciones para transformar el texto en formato json, aplicando previamente las funciones FileReader características de java. Hay múltiples funciones para ir obteniendo los objetos necesarios en función de su naturaleza. En nuestro caso, cadenas de texto y números enteros.

Para entender mejor los archivos json y su creación, veremos un ejemplo de la estructura de uno de estos archivos:

[{"pagina":"Iberia",
"fechaCreacion":"2015-07-10 08:32:33",
"fechaActualizacion":"2015-07-10 10:37:41",
"tipoEstado":"shared_story",
"tamanoComentarios":2,
"id":"128647720484973_10153489934977220",
"mensaje":"Esto no me parece ... ",
"comentarios":
[{"pagina":"Iberia",
"fechaCreacion":"2015-07-0 10:37:41",
"idPropio":"10153489934977220_10153490237777220",
"idPost":"128647720484973_10153489934977220",
"likes":0,
"mensaje":"Hola Mar..."
}
{
//...otros comentarios...//
}
] //cierre de array de comentarios//
} //cierre de post//
,
{
//...otros post...//
}
] //cierre de archivo//

Se puede ver que un archivo json es un array, delimitado por los caracteres [ ], de elementos que son listas ordenadas de pares, nombre y valor, delimitadas por { } y que ese valor, a su vez, puede ser otro array. Cada par se separa por una coma al igual que sucede para separar cada elemento.

Independientemente del tipo de dato, los pasos de lectura son los siguientes:

- Abrir el archivo .json que queramos leer mediante FileReader.

- Pasar todo el archivo a cadena de texto y mediante una función parse, transformar la cadena a un objeto JsonElement donde extraeremos además el número de elementos json con la función size.
- Cerramos el archivo porque ya hemos leído los datos y ahora vamos a procesarlos.
- Obtenemos cada elemento uno a uno como si fuese un array de java mediante la función getAsJsonArray y getAsJsonObject para tener de forma individual un elemento json en java.
- Al objeto json en java le decimos el nombre del par que queramos obtener del objeto con una función get y por ultimo le decimos el tipo primitivo de objeto java que vamos a utilizar para almacenar el valor leído. Recordemos que vamos a guardar como cadena de texto y como número entero.

El poder cargar archivos json nos da más rapidez porque disminuye el tiempo de carga, además de poder exportarse y ser leídos por otros usuarios de la aplicación.

3.2. Base de datos

Una base de datos en una aplicación aporta una gran agilidad, portabilidad y nuevas funciones a la hora de filtrar o usar la información contenida en la misma. Todos los usuarios de la base de datos podrán extraer la información que ellos mismos adquirieron o que otros usuarios buscaron, creando una comunidad interna.

Se decidió el empleo de una base de datos mySql por la experiencia adquirida en su utilización en diferentes asignaturas del Grado.

El acceso a la información contenida en la base de datos es muy rápida, aunque no tanto como cargar un archivo json, lo que permite gran agilidad a los usuarios. Simplemente es necesario conectarse a través de la aplicación puesto que permite su uso en cualquier sitio sin necesidad de ningún tipo de archivo como pasa con json.

En las consideraciones del proyecto planteamos la posibilidad de cargar archivos json en la base de datos así como crear archivos json con los datos obtenidos de la misma, funciones que se implementaron con éxito. También, consideramos el no repetir información almacenada, y gracias a la gestión que hace mySql de claves primarias, resultó fácil conseguirlo gracias a que los identificadores tanto de posts como de comentarios son únicos en todo Facebook. Además, en las pruebas de la aplicación se observó cómo algunos mensajes tenían una longitud de caracteres muy extensa por lo que decidimos establecer un límite máximo de 20.000 caracteres en la base de datos al guardar mensajes empleando varchar, es decir, cadena de caracteres de tamaño variable, al crear las tablas porque ocupan memoria en función de la longitud del mensaje, ralentiza una parte cuando se realizan peticiones a la base de datos, pero la cantidad de memoria ahorrada es sustancial. Está forma de gestionar los mensajes también se empleó al guardar los ids de posts y comentarios, puesto que el número se incrementa de forma sustancial con el transcurso del tiempo.

Para una mayor seguridad en las transacciones y una mayor facilidad a la hora de manejarlas, entre la aplicación y la base de datos se decidió el uso de variables PreparedStatement en java por encima de Statement, para así evitar que los usuarios puedan romper la base de datos.

4. Desarrollo de la aplicación

En este capítulo trataremos las fases por las que ha pasado el proyecto desde el comienzo hasta su finalización.

Puesto que el lenguaje de programación principal era Java, se utilizó Eclipse como entorno de programación por ofrecer una mayor ayuda respecto a los errores de código y más facilidad para corregirlos. Teniendo la posibilidad de compilar y ejecutar el código que está siendo programado.

4.1. Aplicación con interfaz java

En primer lugar, la aplicación comenzó por el aprendizaje de la librería facebook4J, haciendo pruebas de conexión y de obtención de información y sobre todo aprendiendo el uso de los objetos Reading puesto que la información en la página de la librería es muy escasa.

La primera versión ejecutable mostraba por línea de comandos los 25 últimos post, configuración por defecto del objeto Reading, de una página escrita directamente en código.

La siguiente versión ya se mostraría por la interfaz gráfica de java. Con lo aprendido en las fases previas, se introdujeron dos opciones: una búsqueda básica donde se localizarían los últimos 25 post con un máximo de 25 comentarios de cada post de una página que se mostraría por parámetro y una búsqueda avanzada donde el usuario introduciría además del nombre de la página, dos fechas y la aplicación se encargaría de buscar los post y comentarios de esa página entre esas dos fechas. En esta versión los resultados se seguirían mostrando por línea de comandos.

Más adelante se planteó el problema de almacenar la información obtenida y se optó por el uso de archivos json que, como ya se ha explicado anteriormente, están muy aceptados y su estructura es sencilla. Se investigaron distintas librerías para obtener estos archivos hasta que definitivamente se eligió la librería gson de Google por la gran cantidad de información de la misma y por ser creada por una gran empresa.

Con la librería elegida y empleando la segunda versión como base, se desarrolló una tercera versión que incluía la función de guardar la búsqueda realizada en un archivo json con el nombre que eligiera el usuario. Además, se introdujo una función de filtro a la búsqueda avanzada para que sólo recogiera la información del tipo de post elegido. Debemos destacar que internamente el programa hacía dos búsquedas cuando se guardaba como archivo json, lo que lo hacía muy ineficiente, problema que se corregiría en los siguientes avances.

Imágenes de esta versión:

Imagen 18: Imagen de portada de esta versión

Esta era la visión principal de esta versión, directamente la búsqueda básica, donde se ponía el nombre de la página y el rango de búsqueda. Para ver los resultados, mostrados por línea de comandos, había que pulsar el botón Buscar. Además, si se deseaba guardar en un archivo json, había que pulsar el botón Guardar para que apareciera el formulario. También se mostraban por línea de comandos los resultados de la búsqueda. Por último, el botón Avanzado mostraba el formulario de búsqueda avanzada que veremos más adelante.

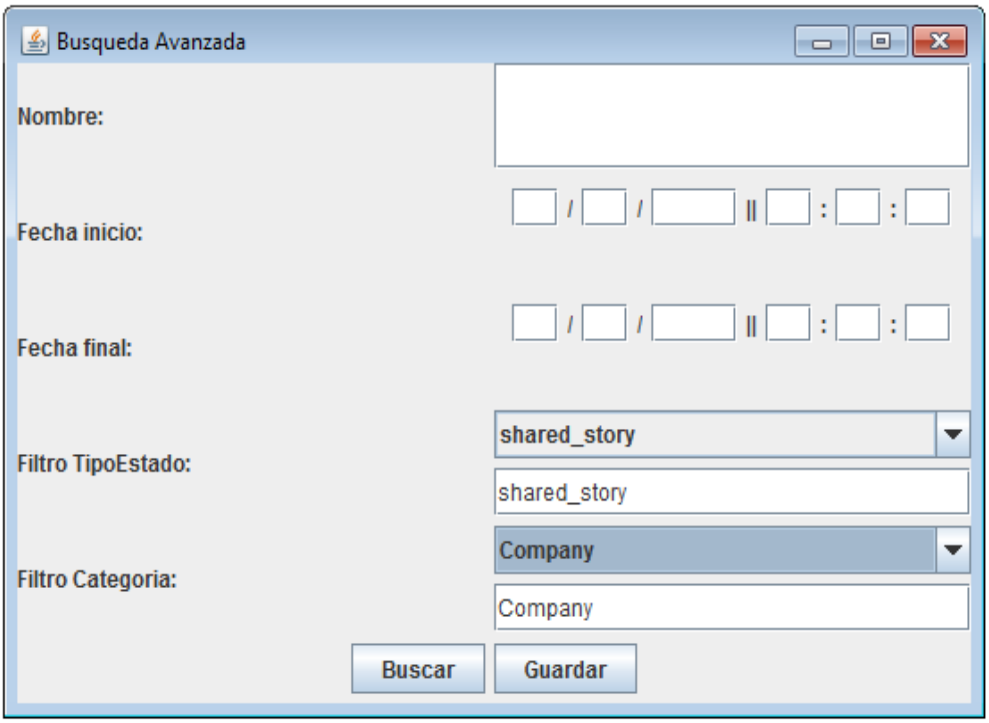


Imagen 19: Formulario de búsqueda avanzada

En el formulario de búsqueda avanzada nos encontramos los mismos campos que en el de búsqueda básica pero, ahora, con una precisión de horas, minutos y segundos para el rango temporal de búsqueda. Además, se agregan dos tipos de filtro: según la naturaleza del post en Filtro TipoEstado y según el tipo de categoría de quién escribía el post en Filtro Categoría.

En Filtro TipoEstado teníamos cinco opciones, que no filtrase con la opción –ninguno--, con wall_post si queríamos ver sólo los Post to Page y, finalmente, tres filtros para los PagePost, shared_story si era un post con texto, added_photos si era un post donde el elemento principal era la foto y added_video que es un caso igual al anterior pero donde el elemento principal era un video.

Los botones de Buscar y Guardar funcionaban exactamente igual que en búsqueda básica.

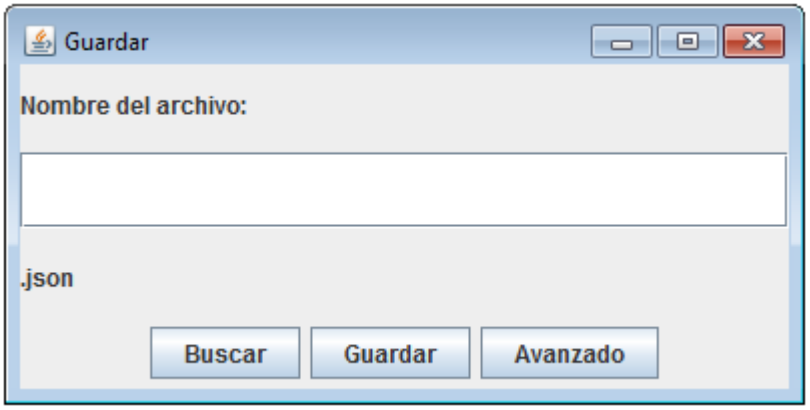


Imagen 20: Formulario de guardar el archivo

Esta imagen muestra el formulario para guardar la búsqueda en un objeto json, el usuario introducía el nombre y la aplicación añadía la extensión automáticamente. El archivo se guardaba en la misma carpeta donde se encontraba la aplicación.

4.2. Aplicación web

El siguiente paso puede que sea el más importante, debido a la limitación y los problemas de desarrollar interfaces gráficas en java y en la búsqueda de portabilidad de la aplicación. Finalmente, se decidió implementar la aplicación como una aplicación web.

En el mismo equipo de desarrollo se instaló un servidor Apache Tomcat donde se sostendría la nueva aplicación.

En la primera versión como aplicación web se introdujo todo lo de la última versión como aplicación puramente java pero con modificaciones. Como por ejemplo, introducir la búsqueda por fecha para la búsqueda básica y un aumento de hasta 500 comentarios por post. Ahora se mostraba el resultado de las búsquedas por pantalla en forma de tabla y se introducía una función para cargar archivos json. Es necesario puntualizar que los archivos html y jsp todavía no incluyen hojas de estilo css en esta versión y no se introducirían hasta la última versión.

El siguiente avance fue la introducción de una base de datos, puesto que la aplicación ahora se podría utilizar en cualquier lado y los archivos json no tienen esa ventaja. Se decantó por la convivencia de ambos sistemas de almacenamiento por su distinta naturaleza, el json es un archivo personal y puede ser procesado por más

programas, mientras que la información guardada en la base de datos puede ser utilizada por cualquier usuario pero es portable y el acceso a la información es más rápido que hacer peticiones a la página de Facebook, aunque eso sí, más lento que el acceso a un json.

Se utilizó una base de datos mySql por la experiencia adquirida durante el Grado. Se crearon dos tablas: una para post y otra para comentarios, con mecanismos de almacenamiento InnoDB por su fiabilidad y consistencia.

Para la creación de las tablas se utilizaron los siguientes algoritmos sql:

- Tabla de post:

```
create table Post(pagina varchar (30) not null,
 id varchar(40) not null,
 creador varchar(50) not null,
 fechaCreacion datetime not null,
 fechaActualizacion datetime not null,
 tipo varchar(20) not null,
 tamanhoComments int not null,
 likes int not null,
 mensaje varchar(20000) not null,
 primary key(id))Engine=innodb;
```

- Tabla de comentarios:

```
create table Comment(pagina varchar (30) not null,
 idPropio varchar(40) not null,
 idPost varchar(40) not null,
 creador varchar(50) not null,
 fechaCreacion datetime not null,
 likes int not null, mensaje varchar(20000),
 primary key(idPropio),
 constraint foreign key(idPost) refereences Post(id))Engine=innodb;
```

En la tabla de post se colocó el id del post como clave primaria para evitar post duplicados porque, como ya hemos visto, el id de un post es único. Para la tabla de comentarios se hizo lo mismo para sus identificadores en idPropio y se referenciaron los comentarios a su post correspondiente a través de la clave foránea idPost haciendo referencia al id de la tabla Post.

Tanto los creadores de los post como de los comentarios, se guardan en limpio, es decir, sin codificar. Se consideró que las empresas que utilicen la aplicación son las que deberían tener el deber de guardar y tratar la información personal de los usuarios tal y como se trata en las Directivas 95/46/EC de la Unión Europea[12].

[12] Directives 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data.

A partir de la primera versión como aplicación web se creó la segunda, añadiendo botones para guardar las búsquedas en la base de datos, así como pasar un archivo json a la base de datos y una nueva opción para buscar información en la base de datos y si se desea crear un archivo json de esa búsqueda.

La tercera y última versión añadía hojas de estilo, así como depuración de código y prevención de fallos o datos malintencionados por el usuario.

Imágenes de esta versión:

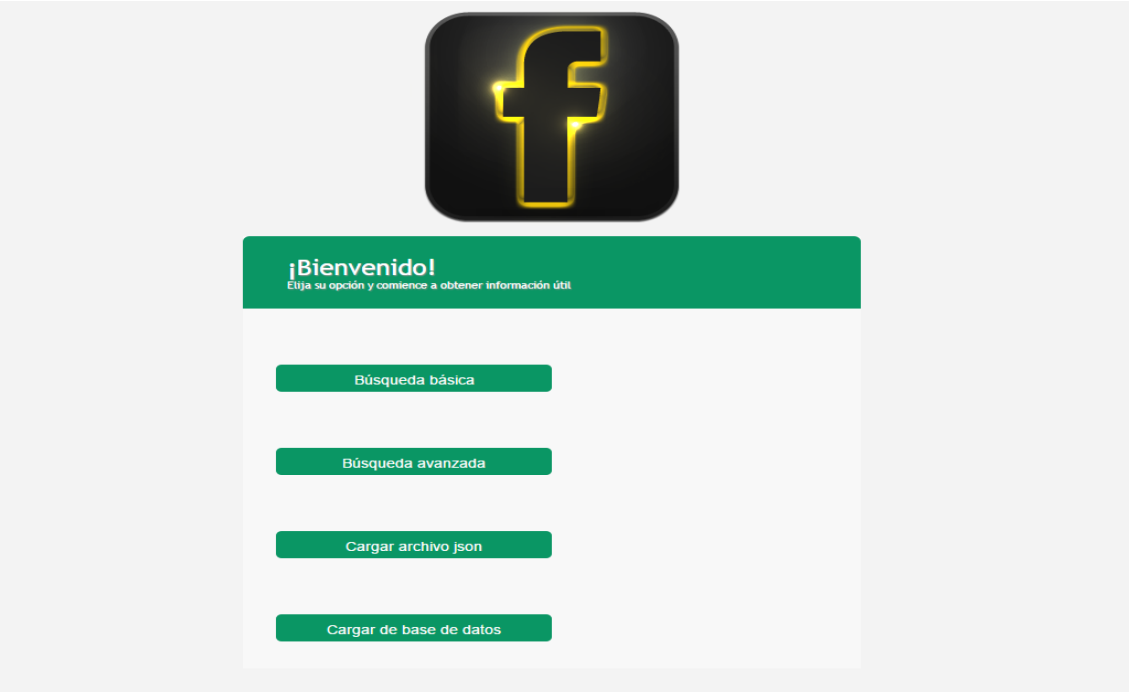


Imagen 21: Menu principal de la aplicación web

Se puede ver como el menú de inicio es muy distinto a la versión con interfaz java. Tenemos las opciones ya mencionadas que se añadieron, búsqueda básica y avanzada, cargar un archivo json y cargar de la base de datos.

Imagen 22: Formulario de la opción de búsqueda avanzada

La interfaz de búsqueda avanzada tiene las mismas opciones que tenía en la versión de interfaz java, pero ahora el filtro de tipo de estado es sustituido por un filtro en general y ahora tiene tres opciones: no poner filtro, filtrar los Post to Page o filtrar los PagePost. Además, se eliminó el filtro por categoría porque realmente no aportaba ninguna información útil. Así pues, debemos señalar que la interfaz de cargar desde la base de datos es prácticamente igual.

1176906585659076_1176907148992353	2015-09-01 09:30:44	-----	Hi Laura, we're sorry for the inconvenience. In order to forward your request along to the customer service could you please provide us with your claim reference number via private message? Kind regards.
1176906585659076_1176912118991856	2015-09-01 09:36:08	-----	My first name is clearly not Laura. I will message you
128647720484973_1176905902325811	2015-09-01 09:25:32	PagePost	#NuevaYork, anímate y toca el cielo :) http://ow.ly/m8hln
1176905875659147_1176961862320215	2015-09-01 11:15:32	-----	Donde puedo conseguir cupon descuento para nueva york?
1176905875659147_1176961862320215	2015-09-01 11:15:32	-----	Donde puedo conseguir cupon descuento para nueva york?
128647720484973_1176903742326027	2015-09-01 09:15:06	Post to Page	Dear Iberia, last night (10 hours ago) I booked a flight on your site but made a mistake. I would like to cancel this booking so I can then re-book a correct flight for my travel needs, as the booking was made less than 24 hours ago. My confirmation code is 2D8GSA. Thank you for looking into this.
	2015-		Hi Sara, in this case please contact the customer service on the phone for further

Imagen 23: Así se muestran los resultados al usuario

En esta imagen podemos ver como se muestran los post y los comentarios tras la búsqueda básica. Aunque de forma interna se guarden más datos como la fecha de actualización del post, se muestran el id del post o comentario, la fecha de cuando se creó, tipo de naturaleza en caso de los post, en comentarios se muestran como cinco

guiones y, por último, una columna donde se muestren los mensajes que contiene. Los comentarios se ponen a continuación del post al que hacen referencia.

Por último, sobre esta versión, se realizaron mejoras y corrección de errores. En búsqueda avanzada se añadió la opción para añadir el número de iteraciones que el usuario deseaba por post, obteniendo hasta 250 comentarios por post e iteración. Se realizó la misma operación para el número de likes. Estas opciones se verán en más detalle en el siguiente capítulo.

A continuación se muestra una imagen de cómo quedó la ventana de búsqueda avanzada:

Imagen 24: Pantalla de Búsqueda avanzada de la versión corregida.

5. Funcionamiento interno

En esta sección se explicará el funcionamiento interno de las partes de Java de la aplicación, comentando las diferentes partes de código que hacen posible conectarse a Facebook, hacer las peticiones necesarias y guardar la información buscada. El código mostrará la funcionalidad de lo que se pretende conseguir. Para más detalles, en los anexos de la memoria se agregará el código java de toda la aplicación.

Como ya hemos explicado, para poder realizar peticiones a Facebook es necesario un token de aplicación, además del App ID y el App Secret, que hemos visto como se obtenían en el apartado Facebook para desarrolladores.

5.1. Paquete Manager:

Para mayor facilidad a la hora de programar y con el fin de tener más orden a la hora de gestionar el código se creó un paquete con tres clases:

- Clase Manager: clase principal que incluye los métodos necesarios para conectarse y desconectarse tanto a Facebook como a la base de datos, además de métodos que devuelven objetos Reading preparados para las peticiones y métodos para buscar comentarios de post. Importante destacar que como atributos tiene un objeto de la clase Facebook de la librería facebook4J necesario para las conexiones a Facebook y un objeto de la clase Connection de la librería mysql-connector, incluida en el la carpeta de librerías del servidor Apache-Tomcat, necesario para mantener la conexión con la base de datos y realizar las transacciones necesarias.

Para autenticarnos en Facebook utilizaremos el siguiente código, que como veremos necesita el token de aplicación, el App ID y el App Secret. El método modifica el atributo de conexión del objeto Facebook a conectado:

```
Facebook facebook = new FacebookFactory().getInstance();
facebook.setOAuthAppId(appId,  appSecret);
facebook.setOAuthAccessToken( new AccessToken(appAccessToken,
```

Con las funciones que nos proporciona la librería mysql-connector podemos conectarnos y desconectarnos de la base de datos y también realizar las transacciones en las tablas de la misma.

Para conectarse es necesario tener la URL de la base de datos y un nombre de usuario y contraseña registrados en la misma. Los siguientes códigos muestran cómo se realizan la conexión y la desconexión a la base de datos:

Para conectarse:

```
Class.forName("com.mysql.jdbc.Driver");
Connection conn = DriverManager.getConnection(DB_URL,USER,PASS);
```

Para desconexión:

```
conn.close();
```

En la realización de las transacciones a la base de datos y, como ya se ha comentado en el apartado Base de datos, se utilizaron transacciones con PreparedStatement en vez de con Statement para evitar ataques malévolos de los usuarios a la base de datos.

El siguiente código muestra la transacción de escritura de un post en la tabla en la base de datos:

```
boolean sucess=false;
conn.setTransactionIsolation(Connection.TRANSACTION_REPEATABLE_READ);
conn.setAutoCommit(false);
String sql = "INSERT INTO Post(pagina, creador, id, fechaCreacion,
fechaActualizacion, tipo, tamanhoComments, likes, mensaje)"+"VALUES
(?,?,?,?,?,?,?,?,?)";
PreparedStatement stmt = conn.prepareStatement(sql);
stmt.setString(1, pagina);
stmt.setString(2, creador);
stmt.setString(3, id);
…
stmt.setString(9, mensaje);
int o = stmt.executeUpdate();
if( o > 0){
       sucess=true;
       }
if (sucess) {
          conn.commit();
} else {
          conn.rollback();
        }
conn.setAutoCommit(true);
```

El siguiente ejemplo muestra la lectura de información de la tabla de Post de la base de datos y como la almacena en una cadena de objetos infopost que se tratarán más adelante:

```
List<InfoPost> arch = new ArrayList<InfoPost>();
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
SimpleDateFormat sdh = new SimpleDateFormat("HH:mm:ss");
String query = "select * from Post where pagina=? and fechaCreacion between ?
and ?";
PreparedStatement stmt = conn.prepareStatement(query);
stmt.setString(1, pagina);
stmt.setString(2, ini);
stmt.setString(3, fin);
ResultSet rs = stmt.executeQuery();
while(rs.next()){
       String id = rs.getString("id");
       InfoPost po = new InfoPost(
       rs.getString("pagina"),
       rs.getString("creador"),
```

```
        sdf.format(rs.getDate("fechaCreacion"))+"
        "+sdh.format(rs.getTime("fechaCreacion")),
        sdf.format(rs.getDate("fechaActualizacion"))+" "
        +sdh.format(rs.getTime("fechaActualizacion")),
        rs.getString("tipo"),
        rs.getInt("tamanhoComments"),
        rs.getInt("likes"),
        rs.getString("mensaje"),
        comentarios(id)
        );
        arch.add(po);

}
stmt.close();
```

La mayor parte de las veces, una página es capaz de escribir y que le escriban numerosos post en un periodo corto de tiempo y se desea obtener información en un periodo largo. Facebook limita el número de post que devuelve por petición a 100, por lo que es necesario hacer una partición del rango temporal de búsqueda en la aplicación para hacer varias búsquedas en cada una de esas particiones y romper el límite de 100 post. Para ello se utilizó la clase calendar y se crearon dos métodos que hacían particiones cada diez horas para los post y cada hora para los comentarios, puesto que se reciben bastantes más comentarios que post. Estos métodos, además, devuelven una lista de cadenas de texto con las fechas partidas en un formato válido para los atributos de tiempo de un objeto Reading, que es el encargado de filtrar las búsquedas de información en las peticiones a Facebook. Los métodos reciben de entrada dos objetos Calendar:

```
public List<String> listadoFechas (Calendar ini, Calendar ult){

        List<String> lista_fechas = new ArrayList<String>();
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
                Calendar aux;
        aux = ini;
        Date ul = ult.getTime();

        while(aux.getTime().getTime() < ul.getTime()){
                lista_fechas.add(sdf.format(aux.getTime()));
                aux.add(Calendar.HOUR, 10);
                if(aux.getTime().getTime() >= ul.getTime()){
                        lista_fechas.add(sdf.format(ult.getTime()));
                }
        }
        return lista_fechas;

}
```

Los métodos para obtener un objeto Reading con los parámetros modificados son sencillos, pero para conocer su estructura fue necesario hacer pruebas con la herramienta Graph API Explorer. En el siguiente ejemplo se muestra un método que devuelve un objeto Reading para que en una búsqueda nos devuelva 250 comentarios (los post tienen un límite de 100) entre dos fechas:

```
public Reading actualizaReading (String inicio, String ultimo){

		Reading r = new Reading();
		r.since(inicio);
		r.until(ultimo);
		r.limit(250);
		return r;

	}
```

Esta clase incluye un método que devuelve los 250 comentarios de un post por cada iteración que desee el usuario (en Búsqueda básica una iteración por defecto). Como leemos directamente de Facebook, hay que modificar la fecha de los comentarios añadiendo dos horas para que el comentario se muestre y se guarde con hora española:

```
public List<InfoComment> listadoComentariosPost (Post p, int iteraciones,
String pagina) throws FacebookException, ParseException{

	String id = p.getId();
	int auxInt = 1;
	Calendar auxC = Calendar.getInstance();
	JsonParser parser = new JsonParser();
	SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
	String fechaCreacionPost = sdf.format(p.getCreatedTime());
	System.out.println("Fecha de creación del post: "+fechaCreacionPost);
	String fechaActualizacion = sdf.format(p.getUpdatedTime());
	System.out.println("Fecha de actualización del post:
"+fechaActualizacion);
	List<InfoComment> aux = new ArrayList<InfoComment>();
	String cursor1="";
	String cursor2="";
	if(iteraciones>0){
		try{
		while(auxInt<=iteraciones){
		BatchRequests<BatchRequest> batch = new
BatchRequests<BatchRequest>();
		int size = 0;
		if(auxInt==1){
		batch.add(new BatchRequest(RequestMethod.GET,
p.getId()+"?fields=comments.limit(250)"));
		batch.add(new BatchRequest(RequestMethod.GET,
p.getId()+"?fields=comments.limit(250){like_count}"));
		List<BatchResponse> results = facebook.executeBatch(batch);
		BatchResponse result1 = results.get(0);
		BatchResponse result2 = results.get(1);
		String jsonString1 = result1.asString();
		String jsonString2 = result2.asString();
		JsonElement arrayElement1 = parser.parse(jsonString1);
		JsonElement arrayElement2 = parser.parse(jsonString2);
		size =
arrayElement1.getAsJsonObject().get("comments").getAsJsonObject().get("data")
.getAsJsonArray().size();
		for(int i=0; i<size; i++){
			String creador =
arrayElement1.getAsJsonObject().get("comments").getAsJsonObject().get("data")
.getAsJsonArray().get(i).getAsJsonObject().get("from").getAsJsonObject().get(
"name").getAsString();
```

```
                    String idPropio =
arrayElement1.getAsJsonObject().get("comments").getAsJsonObject().get("data")
.getAsJsonArray().get(i).getAsJsonObject().get("id").getAsString();
                    String mensaje =
arrayElement1.getAsJsonObject().get("comments").getAsJsonObject().get("data")
.getAsJsonArray().get(i).getAsJsonObject().get("message").getAsString();
                    String f =
arrayElement1.getAsJsonObject().get("comments").getAsJsonObject().get("data")
.getAsJsonArray().get(i).getAsJsonObject().get("created_time").getAsString();
                    f = f.substring(0, 10)+" "+f.substring(11, 19);
                    auxC.setTime(sdf.parse(f));
                    auxC.add(Calendar.HOUR,2);
                    f = sdf.format(auxC.getTime());
                    int likes =
arrayElement2.getAsJsonObject().get("comments").getAsJsonObject().get("data")
.getAsJsonArray().get(i).getAsJsonObject().get("like_count").getAsInt();
                    InfoComment inf = new InfoComment(pagina, creador, f,
idPropio, id, likes, mensaje);
                    aux.add(inf);
                    }
                    cursor1 =
arrayElement1.getAsJsonObject().get("comments").getAsJsonObject().get("paging
").getAsJsonObject().get("cursors").getAsJsonObject().get("after").getAsStrin
g();
                    cursor2 =
arrayElement2.getAsJsonObject().get("comments").getAsJsonObject().get("paging
").getAsJsonObject().get("cursors").getAsJsonObject().get("after").getAsStrin
g();
                    }
                    else{
                          batch.add(new BatchRequest(RequestMethod.GET,
p.getId()+"/comments?limit=250&order=chronological&after="+cursor1));
                          batch.add(new BatchRequest(RequestMethod.GET,
p.getId()+"/comments?limit=250&fields=like_count&order=chronological&after="+
cursor2));
                    List<BatchResponse> results =
facebook.executeBatch(batch);
                    BatchResponse result1 = results.get(0);
                    BatchResponse result2 = results.get(1);
                    String jsonString1 = result1.asString();
                    String jsonString2 = result2.asString();
                    JsonElement arrayElement1 = parser.parse(jsonString1);
                    JsonElement arrayElement2 = parser.parse(jsonString2);
                    size =
arrayElement1.getAsJsonObject().get("data").getAsJsonArray().size();
                    for(int i=0; i<size; i++){
                          String creador =
arrayElement1.getAsJsonObject().get("data").getAsJsonArray().get(i).getAsJson
Object().get("from").getAsJsonObject().get("name").getAsString();
                          String idPropio =
arrayElement1.getAsJsonObject().get("data").getAsJsonArray().get(i).getAsJson
Object().get("id").getAsString();
                          String mensaje =
arrayElement1.getAsJsonObject().get("data").getAsJsonArray().get(i).getAsJson
Object().get("message").getAsString();
                          String f =
arrayElement1.getAsJsonObject().get("data").getAsJsonArray().get(i).getAsJson
Object().get("created_time").getAsString();
                          f = f.substring(0, 10)+" "+f.substring(11, 19);
```

```
                        auxC.setTime(sdf.parse(f));
                        auxC.add(Calendar.HOUR,2);
                        f = sdf.format(auxC.getTime());
                        int likes =
arrayElement2.getAsJsonObject().get("data").getAsJsonArray().get(i).getAsJson
Object().get("like_count").getAsInt();
                        InfoComment inf = new InfoComment(pagina, creador,
f, idPropio, p.getId(), likes, mensaje);
                        aux.add(inf);
                  }
                  cursor1 =
arrayElement1.getAsJsonObject().get("paging").getAsJsonObject().get("cursors"
).getAsJsonObject().get("after").getAsString();
                  cursor2 =
arrayElement2.getAsJsonObject().get("paging").getAsJsonObject().get("cursors"
).getAsJsonObject().get("after").getAsString();
            }
            auxInt++;
            }
            }
            catch(NullPointerException e){
                  System.out.println("Excepción cazada");
            }
            }
      return aux;
      }
```

Facebook no devuelve el número de likes que tiene un post, a diferencia de los comentarios, sino que envía una lista con hasta 1000 usuarios que han dado like a ese post y si se quieren obtener más de esos usuarios hay que hacer más peticiones, por lo que obtener el número de likes por post puede ser tedioso. El siguiente código muestra un método que dado un post y un número entero, devuelve el número hasta 1000 likes por post y por iteración:

```
public int likesDePost (Post p, int iteraciones) throws FacebookException{

      int likes = 0;
      int aux = 0;
      int cuenta= 1;
      String cursor ="";
      JsonParser parser = new JsonParser();
      BatchRequests<BatchRequest> batch = new BatchRequests<BatchRequest>();
      batch.add(new BatchRequest(RequestMethod.GET,
p.getId()+"?fields=likes.limit(1000)"));
      List<BatchResponse> results = facebook.executeBatch(batch);
      BatchResponse result1 = results.get(0);
      String jsonString = result1.asString();
      JsonElement arrayElement = parser.parse(jsonString);
      try{
            likes =
arrayElement.getAsJsonObject().get("likes").getAsJsonObject().get("data").get
AsJsonArray().size();
            cursor =
arrayElement.getAsJsonObject().get("likes").getAsJsonObject().get("paging").g
etAsJsonObject().get("cursors").getAsJsonObject().get("after").getAsString();

      }
```

```
		catch(NullPointerException e){
			likes = 0;
			System.out.println("Excepción cazada");
		}
		if(likes==1000 && iteraciones>1){
			try{
				while(cuenta < iteraciones){
					BatchRequests<BatchRequest> batch2 = new
BatchRequests<BatchRequest>();
					batch2.add(new BatchRequest(RequestMethod.GET,
p.getId()+"/likes?limit=1000&after="+cursor));
					List<BatchResponse> results2 =
facebook.executeBatch(batch2);
					BatchResponse result2 = results2.get(0);
					String jsonString2 = result2.asString();
					arrayElement = parser.parse(jsonString2);
					aux =
arrayElement.getAsJsonObject().get("data").getAsJsonArray().size();
					 cursor =
arrayElement.getAsJsonObject().get("paging").getAsJsonObject().get("cursors")
.getAsJsonObject().get("after").getAsString();
					likes = likes + aux;
					cuenta ++;
				}
			}
			catch(NullPointerException e){
				System.out.println("Excepción cazada");
			}
		}
return likes;
}
```

- Clases InfoPost e InfoComment: son las clases que almacenarán la información de los post y los comentarios respectivamente.

En InfoPost almacenamos los datos del post: el nombre de la página donde proviene el post, el creador del post, la fecha de creación del post, la fecha de la última actualización del post, la naturaleza del post, el número de comentarios, el id del post, el mensaje que incluye y todos los comentarios de respuesta al post:

```
		public class InfoPost {
		public String pagina;
		public String creador;
		public String fechaCreacion;
		public String fechaActualizacion;
		public String tipoEstado;
		public int tamanoComentarios;
		public String id;
		public int likes;
		public String mensaje;
		public List<InfoComment> comentarios;

		public InfoPost(String pagina,String creador, String
fechaCreacion,String fechaActualizacion,String tipoEstado, int
tamanoComentarios, String id, int likes, String mensaje, List<InfoComment>
comentarios){
			this.pagina = pagina;
```

```
            this.creador = creador;
            this.fechaCreacion = fechaCreacion;
            this.fechaActualizacion = fechaActualizacion;
            this.tipoEstado = tipoEstado;
            this.comentarios = comentarios;
            this.tamanoComentarios = tamanoComentarios;
            this.id = id;
            this.likes = likes;
            this.mensaje = mensaje;
            this.comentarios = comentarios;
      }

}
```

En InfoComment almacenamos la información de los comentarios: la página de donde proviene el comentario, el creador del comentario, la fecha de creación del comentario, el id del post al que el comentario responde, el id del comentario, el número de likes y el mensaje que contiene el comentario:

```
public class InfoComment {
      public String pagina;
      public String creador;
      public String fechaCreacion;
      public String idPropio;
      public String idPost;
      public int likes;
      public String mensaje;

      public InfoComment(String pagina,String creador,String fechaCreacion,
String idPropio, String idPost, int likes, String mensaje){
            this.pagina = pagina;
            this.creador = creador;
            this.fechaCreacion = fechaCreacion;
            this.idPropio = idPropio;
            this.idPost = idPost;
            this.likes = likes;
            this.mensaje = mensaje;
      }



}
```

5.2. Clases de la aplicación web

En este apartado se hablará de las clases que hacen posible la aplicación web y se tratara el código conforme a: como obtenemos la información del usuario, como crear un archivo json y como enviar la información procesada para mostrársela al usuario.

Como hemos visto en la última parte de la sección Desarrollo de la aplicación, en la aplicación web se introducía un menú con cuatro opciones. Cada opción conducía a una página htlm con un formulario donde el usuario introduce los datos de búsqueda y

se envían a un servlet java que procesará los datos y una vez procesados los devolverá al usuario donde se visualizarán a través de una página JSP (JavaServer Pages), que será otra página html con los datos mostrados.

Todas las clases que gestionan la información de las páginas html utilizan respuestas doPost para no enseñar información importante a través de la URL y así evitar ataques de usuario.

Para tener unicidad en toda la aplicación, para todas las opciones se intentará gestionar la información como texto json, es decir, con la información encontrada, guardarla en formato json y luego mostrarla al usuario, guardarla en un archivo o en la base de datos.

La aplicación trabaja con hora española por lo que es necesario modificar el rango de búsqueda que quiere el usuario para adaptarlo a la hora de servidor de Facebook. Tras investigar con la herramienta Graph API se llegó a la conclusión de que el servidor tiene una diferencia de dos horas menos que lo que se muestra a un usuario en España, por lo que es necesario quitarle dos horas a los datos que introduzca el usuario.

En el siguiente código se enseña cómo se obtiene la información y se guarda en una lista de objetos InfoPost y como se crea un String con formato json:

```
Gson gson = new Gson();
List<InfoPost> arch = new ArrayList<InfoPost>();
Manager m = new Manager();
m.autenticar();
Page pagina = m.facebook().getPage(nombre);
String comprobar = pagina.getName();
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
Calendar inicio = Calendar.getInstance();
Calendar ultimo = Calendar.getInstance();
inicio.setTime(sdf.parse(ini));
inicio.add(Calendar.HOUR, -2);
ultimo.setTime(sdf.parse(fin));
ultimo.add(Calendar.HOUR, -2);

	String idPagina = "";
	Reading god = new Reading();
	List<Comment> aux = new ArrayList<Comment>();
	List<String> lista_fechas = new ArrayList<String>();
	lista_fechas = m.listadoFechas (inicio, ultimo);
	for(int ll = 0; ll<lista_fechas.size()-1; ll++){
	god= m.actualizaReading(lista_fechas.get(ll), lista_fechas.get(ll+1));
	ResponseList<Post> k = m.facebook().getGroupFeed(nombre, god);
		for(int w=0; w < k.size(); w++){
		Post p = k.get(w);
		aux = m.listadoComentariosPost (p);
		List<InfoComment> cm = new ArrayList<InfoComment>();
		for(int mmm=0; mmm<aux.size(); mmm++){
			Comment c = aux.get(mmm);
			String fechaCreacion = sdf.format(c.getCreatedTime());
			String idPropio = c.getId();
			String idPost = p.getId();
```

```
                int likesC = c.getLikeCount();
                String mensaje = c.getMessage();
                String creadorC = c.getFrom().getName();
                InfoComment inf = new InfoComment(nombre, creadorC,
                fechaCreacion, idPropio, idPost, likesC, mensaje);
                cm.add(inf);
            }
            String fechaCreacionPost = sdf.format(p.getCreatedTime());
            String fechaActualizacion = sdf.format(p.getUpdatedTime());
            String tipoEstado = "nulo";
            String creadorP = p.getFrom().getName();
            int likesP = p.getLikes().size();
            if(creadorP.equals(comprobar)){
                tipoEstado ="PagePost";
            }
            else{
                tipoEstado ="Post to Page";
            }
            String id = p.getId();
            String mensajePost = p.getMessage();
            InfoPost inf = new InfoPost(nombre, creadorP, fechaCreacionPost,
            fechaActualizacion, tipoEstado, cm.size(),  id,
            likesP,mensajePost, cm);
            arch.add(inf);
            }
        }
        String formatoJSON = gson.toJson(arch);
```

En el código anterior se puede comprobar cómo se obtienen los post de Facebook a partir de la función getGroupFeed de la librería Facebook4J, pasando el nombre de la página y el objeto Reading con el rango de búsqueda. Además, se ve como gracias a la librería gson es muy sencillo obtener un String en formato json a partir de una lista de objetos java, donde del par nombre-valor lo obtiene del atributo del objeto java y su valor. Por último, la aplicación diferencia los Post to Page de los PagePost y lo hace comprobando el creador del post con el método getFrom a un objeto Post, que devuelve un objeto Category y, a su vez, a ese objeto se le emplea la función getName para obtener el nombre. Si el creador es la propia página se considera PagePost y si es distinto se considera Post to Page.

Para crear el fichero json basta con utilizar las clases características de java de tratamiento de archivos para crear un archivo e introducir la cadena de texto en formato json. Por ejemplo:

```
FileWriter fw = new FileWriter(nombre+".json");
fw.write(stringJson);
fw.close();
```

Para leer de un archivo json es más complicado. Primero hace falta leer el archivo y crear un String con toda la cadena de texto que lleva el archivo.

Código para leer un archivo json e introducir el texto en un objeto String, en este caso el objeto aux:

```
File archivo = new File (archivoJson);
```

```
FileReader fr = new FileReader (archivo);
BufferedReader br = new BufferedReader(fr);
String aux="";
String linea;
while((linea=br.readLine())!=null){
aux=aux+linea;
}
```

Una vez tenemos el String con la cadena de texto en formato json, es necesario crear un objeto JsonElement que es un array con toda la información del archivo y de ahí, ir obteniendo y creando objetos InfoPost e InfoComment a partir de los pares nombre-valor de json.

En el siguiente ejemplo se enseña cómo se obtiene una lista de objetos InfoPost a partir del String obtenido con el código anterior:

```
Gson gson = new Gson();
JsonParser parser = new JsonParser();
JsonElement arrayElement = parser.parse(aux);
int tamanho = arrayElement.getAsJsonArray().size();
List<InfoPost> arch = new ArrayList<InfoPost>();
for(int i = 0; i<tamanho; i++){
String pagina =
e.getAsJsonArray().get(i).getAsJsonObject().get("pagina").getAsString();
 String creador =
e.getAsJsonArray().get(i).getAsJsonObject().get("creador").getAsString();
 String fechaCreacion =
e.getAsJsonArray().get(i).getAsJsonObject().get("fechaCreacion").getAsString(
);
 String fechaActualizacion =
e.getAsJsonArray().get(i).getAsJsonObject().get("fechaActualizacion").getAsSt
ring();
 String id =
e.getAsJsonArray().get(i).getAsJsonObject().get("id").getAsString();
 String tipoEstado =
e.getAsJsonArray().get(i).getAsJsonObject().get("tipoEstado").getAsString();
 String mensaje =
e.getAsJsonArray().get(i).getAsJsonObject().get("mensaje").getAsString();
 int likes =
e.getAsJsonArray().get(i).getAsJsonObject().get("likes").getAsInt();
int tamanhoComentarios =
e.getAsJsonArray().get(i).getAsJsonObject().get("tamanoComentarios").getAsInt
();
List<InfoComment> cm = new ArrayList<InfoComment>();
if(tamanhoComentarios>0){
for(int j = 0; j<tamanhoComentarios; j++){
      pagina2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("pagina").getAsString();
      String creador2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("creador").getAsString();
      String fechaCreacion2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("fechaCreacion").getAsString();
```

```
            String idPropio =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("idPropio").getAsString();
            String idPost =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("idPost").getAsString();
            int likes2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("likes").getAsInt();
            String mensaje2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("mensaje").getAsString();
            InfoComment c = new InfoComment(pagina2, creador2,
fechaCreacion2, idPropio, idPost, likes2, mensaje2);
            cm.add(c);
            }
      }
      InfoPost p = new InfoPost(pagina, creador, fechaCreacion,
fechaActualizacion, tipoEstado, tamanhoComentarios, id, likes,  mensaje, cm);
      arch.add(p);
      }
```

Hemos concluido la sección de Funcionamiento interno, como se ha visto en el apartado de Paquete Manager, ya venía explicado los pasos de lectura y escritura de la base de datos.

6. Evaluación de rendimiento

El límite de comentarios y de post lo establece la clase Reading. Por defecto, la clase Reading tiene un límite de 25 post o comentarios, pero se puede modificar para obtener hasta 250. Es posible que 250 no sean suficientes, así que en la última fase del desarrollo se cambió el sistema de obtención de post, pasando de usar la clase Facebook en combinación con la clase Reading, a la clase BatchRequest. Con esta clase se pueden realizar peticiones a Facebook como si se tratase de la herramienta Graph API, recibiendo como respuesta texto en formato json que será tratado para almacenar los comentarios en una lista de objetos InfoComment. Gracias al tratamiento de Facebook de los comentarios de los post, se puede iterar las veces que sean necesarias para obtener todos los comentarios. En cada iteración Facebook devuelve 250 comentarios y se decidió que en Búsqueda básica se realizase una sola iteración, y en Búsqueda avanzada, el usuario introduciría por parámetro el número de iteraciones que desease obtener.

Para la obtención de los likes de los post, se realizó una solución similar a la de los comentarios. En cada iteración Facebook devuelve hasta 1000 likes por post, por lo que se decidió que para Búsqueda básica se realizase una sola iteración y para Búsqueda avanzada, el usuario decidirá el número de iteraciones.

Se realizaron pruebas para monitorizar el tiempo de post y comentarios con la opción Búsqueda básica (hasta 250 comentarios por post y hasta 1000 likes por post), para comprobar el tiempo de obtención de información en un mes, una semana y un día para una página de Facebook. Hay que tener en cuenta que la aplicación depende de Facebook para la obtención de información, es decir, si Facebook está muy saturado recibiremos más lentamente los post y comentarios que si la página está menos solicitada, por tanto puede variar mucho el tiempo.

Para obtener el tiempo de obtención de la información, se utilizó el método System.currentTimeMillis() en java, que da el tiempo actual de la aplicación en milisegundos, colocando una de estas funciones al principio de la ejecución y otra al final y mostrando la diferencia de tiempo. La información a obtener sería la de la página as.com durante el mes de Agosto.

Para la prueba de obtención de un día, se le pidió a la aplicación que obtuviera los post entre el día 1 de Agosto a las 00:00:00 y el día 2 a las 00:00:00, para una semana entre los días 1 de Agosto a las 00:00:00 y el 8 de Agosto a las 00:00:00, y para un mes, entre los días 1 de Agosto a las 00:00:00 y el 1 de Septiembre a las 00:00:00. Se obtuvieron los siguientes resultados:

	Búsqueda durante 1 día	Búsqueda durante 1 semana	Búsqueda durante 1 mes
	40,06 s	3 m 8,33 s	33 m 56 s
	40,78 s	5 m 56,53 s	13 m 59,26 s
	40,79 s	5 m 26,62 s	13 m 24,83 s
	40,41 s	5 m 31,22 s	13 m 18,63 s
	40,51 s	5 m 13,62 s	13 m 24,93 s

Media	40,51 s	5m 3,26 s	17 m 36,73 s
Nº post	33	284	1313
Nº comentarios	1791	13454	54856

Imagen 25: Tabla de resultados de pruebas de Búsqueda básica.

Medimos ahora el tiempo de obtención de datos para la información entre el día 1 de Agosto a las 00:00:00 y el 2 de Agosto a las 00:00:00 con diferentes opciones: Búsqueda básica como se ha realizado anteriormente, Búsqueda avanzada con 4 iteraciones para comentarios y 4 iteraciones para likes por post; cargar esa información almacenada en un archivo json y cargarla desde la base de datos:

Opción	Tiempo
Búsqueda básica	42,91 s
Búsqueda avanzada	1m 38,66 s
Carga desde archivo Json	5 ms
Carga desde la base de datos	247 ms

Imagen 26: Tiempos de las diferentes opciones.

Se puede comprobar que la carga del archivo json es la más rápida con un tiempo de 5 ms, mínimo en comparación con los otros tiempos. Para la base de datos un tiempo de 247 ms, que también es muy escaso, hay que tener en cuenta que el servidor sql está en el equipo de ejecución, por lo que sería mayor si se encontrase en un equipo externo. Por último, la búsqueda básica se queda como la más lenta dado que tiene que realizar varias peticiones a Facebook y tiene que gestionar las respuestas pero, sin embargo, es la opción primaria de obtención de información que luego almacenaremos para su uso y para una mayor rapidez de acceso. Aquí es donde vemos mayor contraste, con un tiempo de 42,91 segundos para Búsqueda básica (tiempo muy similar a las pruebas realizadas anteriormente) y de un minuto con 38,66 segundos para la búsqueda avanzada con los parámetros citados en el párrafo anterior, prácticamente el doble de tiempo entre ellas.

Por último destacar que si en Búsqueda básica se obtuvieron 1791 comentarios, en Búsqueda avanzada se obtuvieron 2132 comentarios.

7. Planificación

En este capítulo trataremos la planificación del proyecto y si evolución temporal, desglosando las tareas en las que se dividió.

Para ello emplearemos un diagrama de Gantt. En la siguiente imagen se muestran las fases y sus fechas de inicio y finalización:

Nombre	Fecha de inicio	Fecha de fin
Fase de análisis	28/01/15	29/01/15
Propuesta del proyecto	28/01/15	29/01/15
Estudio del estado del arte	28/01/15	29/01/15
Fase de estudio de Facebook y la librería a utilizar	30/01/15	24/02/15
Aprendizaje de la interfaz de las páginas de Facebook	30/01/15	2/02/15
Aprendizaje Graph API de Facebook	2/02/15	10/02/15
Aprendizaje de la librería Facebook4J	11/02/15	24/02/15
Fase de pruebas	11/02/15	31/03/15
Pruebas de aprendizaje de la librería Facebook4J	11/02/15	27/02/15
Desarrollo de la primera versión java	2/03/15	20/03/15
Desarrollo de la segunda versión java	23/03/15	31/03/15
Fase de archivos Json	1/04/15	1/05/15
Estudio de la estructura de un archivo json	1/04/15	1/04/15
Estudio de la librería gson de Google	2/04/15	6/04/15
Pruebas con librería gson	7/04/15	10/04/15
Desarrollo de la tercera versión java	13/04/15	1/05/15
Fase de desarrollo aplicación web	4/05/15	2/07/15
Instalación en servidor Apache-Tomcat	4/05/15	5/05/15
Desarrollo de la primera versión de la aplicación web	6/05/15	26/05/15
Instalación y creación de la base de datos mysql	27/05/15	28/05/15
Desarrollo de la segunda versión de la aplicación web	29/05/15	25/06/15
Creación de hojas de estilo para la aplicación web	26/06/15	2/07/15
Fase de correción de errores	3/07/15	24/08/15
Pruebas de rendimiento de la aplicación actual	3/07/15	8/07/15
Correción de errores, prevención de ataques y optimización de código	9/07/15	29/07/15
Desarrollo de la tercera versión de la aplicación web	30/07/15	19/08/15
Testeo de la aplicación	20/08/15	24/08/15
Fase de desarrollo de la memoria	20/07/15	9/09/15
Reuniones	28/01/15	11/09/15
Primera reunión	28/01/15	28/01/15
Segunda reunión	20/02/15	20/02/15
Tercera reunión	11/06/15	11/06/15
Cuarta reunión	3/07/15	3/07/15
Quinta reunión	11/09/15	11/09/15

Imagen 27: Cuadro con fases y tareas del presente trabajo

A la hora de contabilizar las horas de desarrollo se tuvo en cuenta que cada día correspondía a 4 horas de trabajo y tomando los sábados y domingos como días de descanso y aparte, días que no se avanzó en el proyecto por motivos personales.

A continuación se muestra el diagrama de Gantt resultante de la planificación:

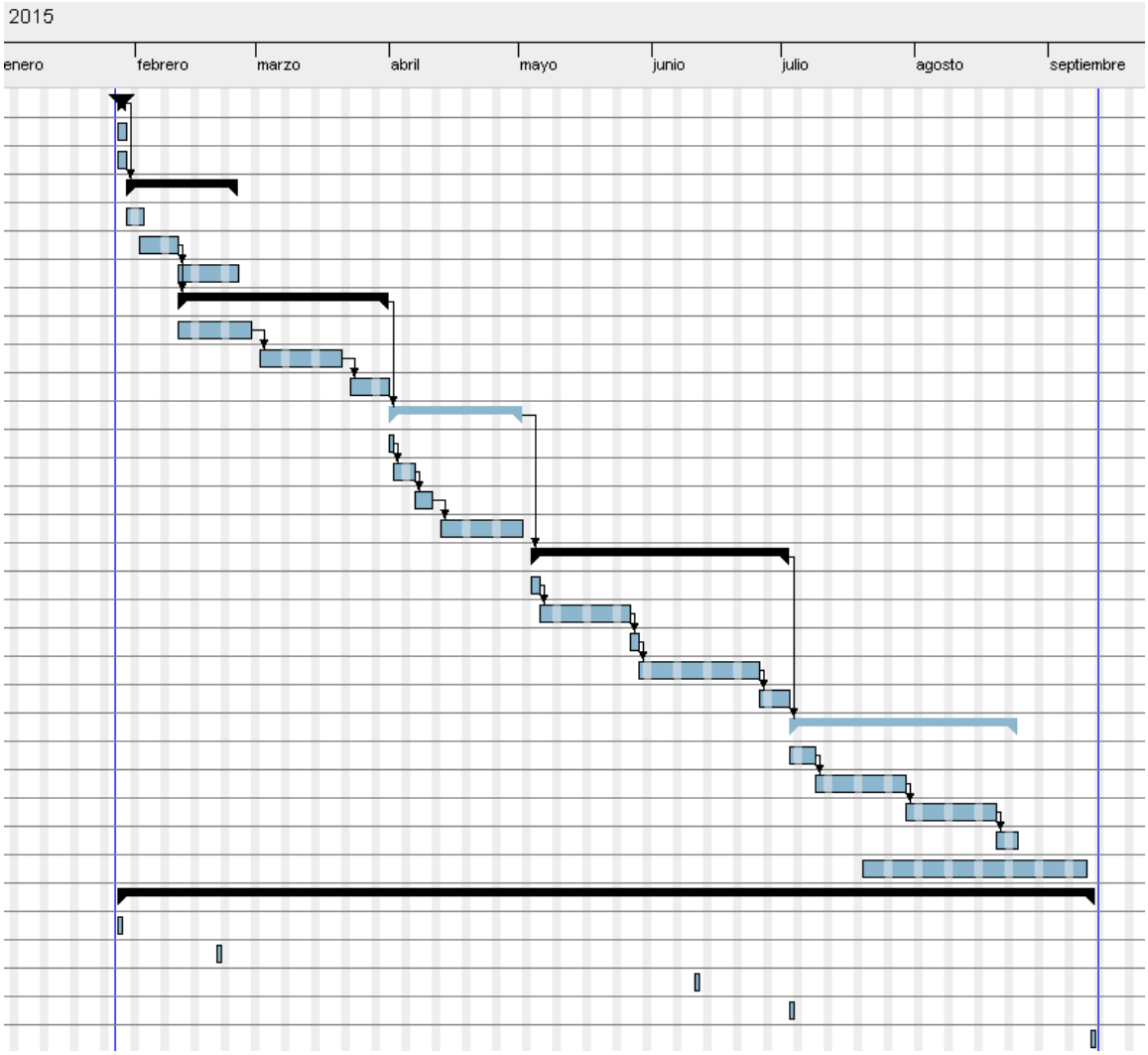


Imagen 28: Diagrama de Gantt

8. Presupuesto

A continuación se justifican los costes del presente trabajo realizado.

La estimación de los costes se calculará en base de los recursos utilizados y las horas estimadas de trabajo.

8.1. Recursos utilizados

En este apartado trataremos los recursos utilizados para el proyecto con una descripción de su función. Se pueden dividir los recursos en dos tipos:

- Recursos Humanos: Son las personas encargadas de la organización y realización del proyecto:
    - Jefe del proyecto: Encargado de la organización del proyecto y dirección del programador.
    - Programador: Persona responsable de dar forma a la aplicación y llevarla a cabo. Necesita conocimientos en java, aplicaciones web y mysql.
    - Testers: Personal externo encargado de probar la aplicación y de buscar errores y posibles ataques a la misma. Deben informar al programador en caso de encontrar alún error.
- Recursos Materiales: En este proyecto se tratan de los recursos informáticos necesarios para terminar el proyecto. Tenemos dos tipos de Recursos Materiales:
    - Software: Programas utilizados durante el proyecto. Se empleó el siguiente software:
        - Sistema Operativo Windows 7 Professional 64 bits.
        - Microsoft Office 2013 para la realización de la memoria.
        - 7zip para la descomprensión de archivos.
        - Eclipse como entorno de desarrollo.
        - Apache-Tomcat 8.0.24 como servidor web.
        - MySQL 5.6 como base de datos.
    - Hardware: Parte física de los Recursos Materiales. Se utilizó el siguiente hardware:
        - PC Intel Core i5 3570k a 3.4GHz con 8GB de memoria ram.
        - Disco duro interno Western Digital de 250GB
        - Conexión a internet de 10 megas.
        - Pen-drive Toshiba de 16GB.

8.2. Costes

En base al capítulo de Planificación, haremos el reparto de los costes en Recursos Humanos dependiendo de la cantidad de horas de trabajo en el proyecto en caso de personal interno y de días trabajados en caso de personal externo:

Para el personal interno:

Puesto	Precio/hora	Número de horas	Total
Jefe del proyecto	40€/h	35h	1.400 €
Programador	15€/h	450h	6.750 €

Imagen 29: Coste de personal interno.

Para el personal externo:

Puesto	Precio/dias	Número de personal	Número de días	Total
Tester	20€/d	5 personas	4d	400 €

Imagen 30: Coste de personal externo.

Los Recursos Humanos suponen un coste total de 8550€.

El coste de los recursos materiales viene mostrado en la siguiente imagen, los recursos que no se muestran implican coste cero en el proyecto:

Material	Coste
Windows 7 Professional 64 bits	145 €
Microsoft Office 2013	130 €
PC completo + disco duro 250GB	1.000 €
Pen Drive 16GB	6 €

Imagen 31: Coste de los Recursos Materiales.

Los Recursos Materiales suponen un coste de 1281€.

Por lo tanto, el coste estimado total del proyecto es de 9831€.

9. Mejoras Futuras

Las posibilidades del big data y el uso correcto de la información como se ha visto en el capítulo de introducción pude dar una gran ventaja competitiva a las empresas con respecto a sus competidores. La aplicación que hemos tratado en este trabajo es una herramienta para obtener una gran cantidad de información y almacenarla en base de datos, en nuestro caso, mysql y en archivos json, estructura de datos nombre-valor no sql, que no deja de ser otra cosa que las tres primeras fases del big data. Por lo que el trabajo futuro a raíz de esta aplicación iría acorde a ir aplicando el resto de fases, además de mejoras a la hora de obtener datos.

9.1. Obtener comentarios de comentarios

Una de las posibles mejoras es obtener comentarios de respuesta a otros comentarios, dado que es posible que ese comentario si ha recibido muchas respuestas tenga información valiosa de cara a ser analizada.

9.2. Inclusión en un servidor físico

Poder incluir la herramienta en un servidor de gran capacidad y que sea capaz de forma rutinaria de guardar diariamente la información de, por ejemplo, 1000 páginas daría una capacidad de obtener información ya procesada de una forma muy ágil. Esta información podría ser usada o vendida a empresas para mejorar sus servicios.

9.3. Incluir Twitter a la hora de obtener datos

La librería Facebook4J toma de base otra librería parecida pero, para la red social Twitter, poder incluir búsquedas de información para esta red social podría considerarse para una futura mejora de la aplicación.

9.4. Análisis de datos

Como ya se ha comentado, esta herramienta contempla únicamente los primeros pasos del concepto de big data, por lo que sería muy interesante añadir funciones que implementen los siguientes pasos. Uno de ellos es analizar los datos que, en el caso de Facebook, los análisis irían en la dirección del clustering o agrupamiento, en busca de agrupar usuarios por gustos o necesidades, esto sería de gran relevancia en páginas de empresas en busca de necesidades no cubiertas por sus servicios o productos. Otra dirección sería analizar la información a través del análisis de texto, lo que ayudaría a visualizar la opinión de los usuarios de las páginas o encontrar información relevante. Este tipo de análisis además de para páginas de empresa, sería importante para ONGs o páginas de servicios públicos.

Sería obligatorio mostrar los resultados de estos análisis para dar una visión más gráfica y estadística de los datos analizados por lo que añadir alguna forma de enseñar los resultados sería interesante.

10. Conclusiones

10.1. Consecución de objetivos

Al comienzo de la presente memoria se marcaron unos objetivos parciales y a raíz de realización de los mismos, la consecución del objetivo principal, siendo este el motivo de realización del trabajo.

- Se ha aprendido la interfaz de las páginas de Facebook y hemos conocido los Post to Page y los PagePost, la información que nos interesaba obtener.
- Hemos aprendido el Graph API de Facebook y hemos comprendido la estructura de los datos que obtenemos para poder usarlos más adelante.
- Se han estudiado y aprendido las clases y métodos de la librería de java Facebook4J que son interesantes para la aplicación.
- Se ha hecho uso de la librería gson de Google que nos ha permitido aprender la estructura de los archivos json y su uso en el proyecto.
- Se ha creado y empleado una base de datos mysql en la aplicación, lo que nos permite el acceso a la información buscada en cualquier lugar.
- Se ha desarrollado una aplicación web y se ha instalado con éxito en un servidor Apache-Tomcat.
- Se han corregido errores de inserción de datos y se ha tenido en cuenta posibles intenciones malévolas de usuarios de la aplicación.

Tras la realización de estos objetivos parciales se puede concluir que se ha desarrollado con éxito una herramienta que permite obtener de forma fácil y sencilla datos de Facebook como desee el usuario.

10.2. Reflexiones personales

Para la realización del trabajo fue necesario el estudio de una librería no oficial de java, así como el uso de archivos json que, si bien sí se dieron durante el estudio del Grado, no se usaron o no se les dio demasiada importancia. Estos dos puntos me han aportado experiencia en algo que desconocía o tenia pocos conocimientos y la poca información que existe sobre la librería facebook4J ha hecho durante el desarrollo que realizase muchas pruebas y junto con el Graph API de Facebook ir aprendiendo e ir consiguiendo a base de ensayo y error, la información que se proponía obtener.

Me di cuenta a medida que programaba como se iban añadiendo nuevos objetivos. Comprobé lo escalable que era el proyecto y con más tiempo y, sobre todo, con más conocimientos se puede hacer una herramienta muy profesional ya que puede tener un uso efectivo en empresas.

Trabajar en una tecnología que se considera que va a ser la que va a marcar el futuro ha sido una gran motivación ya que, desde antes de comenzar con el trabajo, big data era de mi interés ya que considero que realmente puede ser útil viendo cómo

evoluciona la tecnología en un mundo donde todos estamos conectados y generamos una gran cantidad de datos que pueden ser de interés.

10. Conclusions

10.1. Achievement of objectives

At the beginning of this report, some partial goals have been established, and as a result of them, achieve the main objective, the reason for carrying out the work.

- The interface of Facebook pages has been learned and we have known the Post to Page and the PagePost, the information that we had interest to obtain.
- We have learned Facebook´s Graph API and have understood the structure of the data that we collect in order to use them later.
- Facebook4J java library methods and classes that are interesting for the application have been learned.
- Google gson java library methods and classes that has allowed us to learn the structure of the files json and its use in the project have been used.
- A mysql database has been created and used for the application.
- A web application has been developed and installed successfully on an Apache-Tomcat server.
- Inclusion of data errors have been corrected, and users possible attacks have been prevented.

After the completion of these partial goals, it can be concluded that it has successfully developed a tool that allows to obtain quickly and easily information from Facebook as the user wants.

10.2. Personal reflections

For the realization of this project, it was necessary to study an unofficial java library, as well as json files. These two points have given me experience in something that I ignored or I had little knowledge and little information that exists about the facebook4J library. These things made me realize many test during development and together with the Facebook Graph API, to learn and go getting based on trial and error, the information that is intended to obtain.

I realized that as I programmed and added new goals, I checked the scalable was the project and with more time and more knowledge, the application can be much more professional and that it can have a real use in companies.

Working on a technology that is considered that it will mark the future has been a highly motivation. Before start the project, big data was something of interest to me and now I see that it can really be useful watching how evolving technology to a world where we are all connected and we generate a large amount of data that may be of interest.

11. Referencias

Apache Tomcat. (s.f.). Obtenido de http://tomcat.apache.org/

Facebook. (s.f.). Obtenido de www.facebook.com

Facebook for developers. (s.f.). Obtenido de https://developers.facebook.com/

Facebook4J. (s.f.). Obtenido de http://facebook4j.org/en/index.html

Facebook4J. (s.f.). *Librería Facebook4J*. Obtenido de http://facebook4j.org/javadoc/facebook4j/Facebook.html

Google. (s.f.). *Librería gson*. Obtenido de https://google-gson.googlecode.com/svn/trunk/gson/docs/javadocs/overview-summary.html

Ohlhorst, F. (2013). *Big Data Analytics: Turning Big Data in Big Money*. Hoboken, New Jersey: John Wiley & Sons, Inc.

Portolés, A. (09 de 02 de 2015). *http://www.dirigentesdigital.com/*. Obtenido de http://www.dirigentesdigital.com/articulo/economia-y-empresas/219695/oracle/barcelona/digital/abren/centro/big/data/referencia/europea.html

Twitter. (s.f.). Obtenido de https://twitter.com/

Twitter4J. (s.f.). Obtenido de http://twitter4j.org/en/index.html

"Comunicación del programa estatal de fomento de la investigación científica y técnica de excelencia.".FIS2013-47532-C3-3-P Ministerio de Economía y Competitividad

Directives 95/46/EC of the European Parliament and of the Council of 24 October 1995 *on the protection of individuals with regard to the processing of personal data and on the free movement of such data*.

ANEXO I: Resumen en inglés.

Abstract

At the moment, Facebook is the social network with the largest number of active users worldwide. It offers to chat with our old friends and find new people who have as the same hobbies or interests as you. It is allowed to create personal profiles and even private companies make them to post their products or services as well by Facebook. These pages can be used to inform about products or services, and also can be used to report important news, or what is perhaps most interesting, a direct approach to users.

From this pages, companies can obtain precious info and use it them in their favor. This information can be a competitive advantage over other companies.

Motivation and partial goals

The main motivation of this project is looking for implement an external application from Facebook. This application must be able to do a detailed search in one or more particular pages to find good information with which we can save it in external files or in a database. The meaning to doing this is to prepare information for a next phase in which information is processed and used for other purposes. Definitely, we want to make an easy solution to get a lot of data as outlined in big data.

Some partial goals were scored to be solved:

- To know the interface of Facebook pages and what information we want obtain.
- To know Facebook´s Graph API and his information structure.
- Study and learn uses of java´s library Facebook4J.
- Search and learn how create json files and how use they into the application.
- Use properly a database and how we store our information.
- Develop an easy to use user interface for the application.
- Fix errors and prevent computer attacks to the application.

Facebook´s interface

First we explain Facebook´s user interface and how we find the data we what want to use from Facebook´s pages. This information may be a post or a comment. There are two different types of post, it is a Post to Page if the post has been created by a user, and it is a PagePost if the post has been created by the page itself. Comments are responses to the post.

It was necessary to create a Facebook developer´s account from Facebook for developer´s page to begin the development. Once the account has been created, Facebook for developer´s page gives us a unique identifier for the application, a secret

number for the application and a token for the application. These three things are necessary to identify us and make request to Facebook.

Facebook4J library

Before starting the application development, it was necessary to learn how to use Facebook4J´s library. The page from the library provides useful examples in Code Examples. This examples are a good idea to be a first contact with the library and a starting point for testing code.

The library´s javadoc is not well explicated, that's why it was necessary to test code to learn which methods and classes are main for the development.

The getGroupFeeds method allows us to obtain the post we need from Facebook. In addition to the above method, there are four classes that are mainly responsible for everything related to making request to Facebook:

- Facebook class: It is the main class of the library. It includes all the methods needed for connection and disconnection. It also keeps the Facebook session open that is necessary for the rest classes can make their functions.

- Post class: It is the class that includes methods to store and treat the post of the pages we want looking for.

- Comment class: It is the class that includes methods to store and treat the comments from the post we have looked before.

- Reading class: It is a support class for the methods of the application. It includes attributes related to the search of information like the time spacing or the number of elements to obtain.

Once the methods and classes to obtain post and comments have been known, we encounter the problem of saving information.

Saving information

For this problem we think in two solutions, save the information in json files or save it in a mysql database.

To save the information in json files, Google´s gson libray was used. This library is very easy to use and his javadoc page, in contrast to the Facebook4J´s library, is very well explained. The library includes methods that convert a list of java objects directly into a string in json format. Later, this string is inserted in a file with extension .json by java typical FileWriter methods to create and write files.

Java´s interface version

With the investigated methods, a new test version with java´s interface was created. In this version, the application search the last 25 post and 25 comments for each post and if the user wants, save the data in a json file. In this version, the results had been showed by command line.

Web application

After this release, it was decided to change radically the way that the user saw the application. It was decided to create a web application. This web application would be installed on an Apache-Tomcat server and also would show the results orderly and would have a function to load json files and view the information they contain.

Whatever type of data, steps of Reading are the follows:

- Open json file we want read.
- Pass the entire file to string and using a parse function, transform the string to a JsonElement object where we draw the number of json elements with the size function.
- We close the file because we have already read data and now let's process them.
- We get each item one by one as if were a java array by using the function getAsJsonArray and getAsJsonObject to have individually a json element in java.
- We tell to json java object the info we want get and finally store it in a java primitive type object. We are going to save as text string and integer.

Load json files is faster than other methods such as get data from Facebook in addition, json files can be exported and be read by other users of the application.

The next step in the application was to add options to save and load information from a database. As we have already explained we have used a mysql database with InnoDB storage mechanisms for their reliability and consistency. Two boards were created, one to store post and another to store reviews.

Mysql-connector library included in Apache-Tomcat server files was used to connect and disconnect the application to the database. Access to the information contained in the database is very fast, but not so much how to load a json file, which allows great agility to users. You simply need to connect through the application since it permits its use on any website without the need for any type of file as with json.

We sought to create boards in the database that the post IDs were primary keys, this way when we want store information into the database we avoid duplicate post because IDs are unique to each post.

For greater security in transactions and greater ease in handling them, between the application and the database it was decided using PreparedStatement variables in java above Statement to avoid attacks form the users.

From this point, thanks to a better utilization of the Reading class in java, we can now save 500 comments per post. In addition, a new version of the web application was created, which included the following:

- Basic Search: performs a lookup on a temporal range and shows all the post and 500 comments for post for that search. The information is displayed to the user on the screen and it can decide if stored in the database or in a json file.

- Advanced Search: similar to Basic Search but the temporary range is now more accurate and you can choose the nature of the post that you want to obtain. You can get post of the type Post to Page, PagePost or both. Once shown the search to the user, as in the basic search, you can choose to save it in a json file or database.

- Load json file: loads a json file created in a previous search. Once shown to the user, this can choose to turn over information to the database.

- Load the database: similar to Advanced Search, search the database for stored post and comments. Once shown the user can be stored in a json file.

After this release and correcting errors and procurer that the users can´t make attacks. Now, the application is be able to store any number of comments. The application is terminated.

Internal working

After trying the application options, we explained how the java code was created and makes possible the operation of the application. They can be separated in two:

- Package Manager: includes classes to store information about the post and the comments as well as another class with the functions of connection and disconnection to Facebook and to the database. In addition, this class also includes methods to modify the class Reading, making transactions to the database, get comments from post and help effectively get a temporary search range.

- Classes for the web application. It includes classes responsible of obtain and manage the users data through the application, make requests to Facebook and also includes the classes responsible for creating the json files and save the information in them. He is also responsible for interpreting the json files.

The application includes html pages that navigates the user and writes the information that the user wants and jsp pages by showing the information managed by the web application classes.

Evaluation of internal operation

Internal operation test were carried out to check the time that took the application to obtain post and comments depending on the different options. Function System.currentTimeMillis of java was used to measure the difference of time in milliseconds that took the application request information and obtain it in each of application´s options.

As a result, on equal footing, was obtained that load a json file was much faster than look up the information on Facebook or in the database. But to have a json file is necessary to do a search on Facebook previously.

Improvements

As already mentioned, this tool provides only the first steps of the concept of big data, so it would be interesting to add functions that implement the following steps. Another improvement would be to use this tool in other social networks like Twitter and reduce the response time of the application

Conclusions

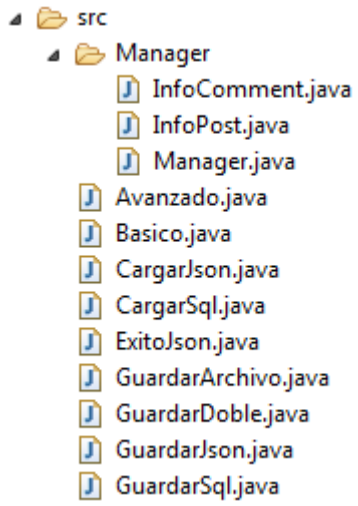
Following the completion of the partial objectives set at the beginning, it can be concluded that it has successfully developed a tool that allows to obtain quickly and easily information from Facebook as the user wants.

It has been a great experience and a great satisfaction seeing finished this work.

ANEXO 2: Código Java

Código java

Estructura:



Paquete Manager:

-InfoComment.java :

```
package Manager;
//Clase apoyo a Json para la colocación de los comentarios respuesta de los
post
public class InfoComment {

	public String pagina;
	public String creador;
	public String fechaCreacion;
	public String idPropio;
	public String idPost;
	public int likes;
	public String mensaje;

	public InfoComment(String pagina,String creador,String fechaCreacion,
String idPropio, String idPost, int likes, String mensaje){
		this.pagina = pagina;
		this.creador = creador;
		this.fechaCreacion = fechaCreacion;
		this.idPropio = idPropio;
		this.idPost = idPost;
		this.likes = likes;
		this.mensaje = mensaje;
	}
}
```

-InfoPost.java :

```
package Manager;
```

```
import java.util.List;
//Clase apoyo a Json para la colocación de los post
public class InfoPost {

	public String pagina;
	public String creador;
	public String fechaCreacion;
	public String fechaActualizacion;
	public String tipoEstado;
	public int tamanoComentarios;
	public String id;
	public int likes;
	public String mensaje;
	public List<InfoComment> comentarios;

	public InfoPost(String pagina,String creador, String
fechaCreacion,String fechaActualizacion,String tipoEstado, int
tamanoComentarios, String id, int likes, String mensaje, List<InfoComment>
comentarios){
		this.pagina = pagina;
		this.creador = creador;
		this.fechaCreacion = fechaCreacion;
		this.fechaActualizacion = fechaActualizacion;
		this.tipoEstado = tipoEstado;
		this.comentarios = comentarios;
		this.tamanoComentarios = tamanoComentarios;
		this.id = id;
		this.likes = likes;
		this.mensaje = mensaje;
		this.comentarios = comentarios;
	}
}
```

-Manager.java :

```
package Manager;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import facebook4j.BatchRequest;
import facebook4j.BatchRequests;
import facebook4j.BatchResponse;
import facebook4j.Facebook;
import facebook4j.FacebookException;
import facebook4j.FacebookFactory;
import facebook4j.Post;
import facebook4j.Reading;
import facebook4j.auth.AccessToken;
import facebook4j.internal.http.RequestMethod;
import java.sql.*;
import com.google.gson.Gson;
```

```
import com.google.gson.JsonElement;
import com.google.gson.JsonParser;

public class Manager {

	private static String appId = "*****";
	private static String appSecret = "******";
	private static String appAccessToken = "*****";
	private Facebook facebook;
	static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
	static final String DB_URL =
"jdbc:mysql://adscom03.it.uc3m.es/prueba";
	static final String USER = "****";
	static final String PASS = "****";
	public Connection conn = null;

	public void autenticar(){
		facebook = new FacebookFactory().getInstance();
		facebook.setOAuthAppId(appId,  appSecret);
		facebook.setOAuthAccessToken( new AccessToken(appAccessToken,
null));
	}

	public void conectar() throws ClassNotFoundException, SQLException{
		 Class.forName("com.mysql.jdbc.Driver");
		conn = DriverManager.getConnection(DB_URL,USER,PASS);
	}

	public void desconectar() throws ClassNotFoundException, SQLException{

		try{

			if(conn!=null)
			   conn.close();
		}

		catch(SQLException se){
			   se.printStackTrace();
		}
	}

	public Facebook facebook (){
		return facebook;
	}

	//metodo que comprueba que no exista ya un post con el id dado
	public boolean posiblePost (String id) throws SQLException{
		boolean sucess=false;

		if (conn != null){
			String query = "select id from Post where id=?";
			PreparedStatement stmt = conn.prepareStatement(query);
			stmt.setString(1, id);
			ResultSet rs = stmt.executeQuery();

			if(rs.next()==false){
				sucess=true;
			}
```

```
                stmt.close();

            }
            return sucess;
      }

      //metodo que comprueba que no exista ya un Comment con el idPropio
dado
      public boolean posibleComment (String idPropio) throws SQLException{
            boolean sucess=false;

            if (conn != null){
                  String query = "select idPropio from Comment where
idPropio=?";
                  PreparedStatement stmt = conn.prepareStatement(query);
                  stmt.setString(1, idPropio);
                  ResultSet rs = stmt.executeQuery();

                  if(rs.next()==false){
                        sucess=true;
                  }

                  stmt.close();
            }
            return sucess;
      }

      public int tamanhoCommentPost (String id) throws SQLException{
            int tamanho = 0;

            if (conn != null){
                  String query = "select count(*) from Comment where
idPost=?";
                  PreparedStatement stmt = conn.prepareStatement(query);
                  stmt.setString(1, id);
                  ResultSet rs = stmt.executeQuery();

                  while(rs.next()){
                        tamanho = rs.getInt(1);
                  }

                  stmt.close();
            }
            return tamanho;
      }

      public boolean actualizaTamanhoPost(String id, int likes) throws
SQLException{
            int tamanho = tamanhoCommentPost (id);
            boolean sucess=false;
            if (conn != null ){

      conn.setTransactionIsolation(Connection.TRANSACTION_REPEATABLE_READ);
            conn.setAutoCommit(false);
            String sql = "Update Post set tamanhoComments = ?, likes = ?
where id = ?";
          PreparedStatement stmt = conn.prepareStatement(sql);
          stmt.setInt(1, tamanho);
          stmt.setInt(2, likes);
```

```
            stmt.setString(3, id);
            int o = stmt.executeUpdate();

            if( o > 0){
                  sucess=true;
            }

             if (sucess) {
               conn.commit();
            }

             else {
               conn.rollback();
            }

           conn.setAutoCommit(true);
      }
      return sucess;
      }

      //metodo que introduce un post en la base de datos
      public boolean introducirPost(String pagina, String creador, String
fechaCreacion, String fechaActualizacion, String id, String tipoEstado, int
tamanhoComentarios, int likes, String mensaje) throws SQLException{
            boolean sucess=false;

            if (conn != null ){

                  if((mensaje.length() < 20000) && (creador.length() < 50)){

                  if(posiblePost(id)==true){

      conn.setTransactionIsolation(Connection.TRANSACTION_REPEATABLE_READ);
                        conn.setAutoCommit(false);
                        String sql = "INSERT INTO Post(pagina, id, creador,
fechaCreacion, fechaActualizacion, tipo, tamanhoComments, likes,
mensaje)"+"VALUES (?,?,?,?,?,?,?,?,?)";
                     PreparedStatement stmt = conn.prepareStatement(sql);
                     stmt.setString(1, pagina);
                        stmt.setString(2, id);
                        stmt.setString(3, creador);
                        stmt.setString(4, fechaCreacion);
                        stmt.setString(5, fechaActualizacion);
                        stmt.setString(6, tipoEstado);
                        stmt.setInt(7, tamanhoComentarios);
                        stmt.setInt(8, likes);
                        stmt.setString(9, mensaje);
                        int o = stmt.executeUpdate();

                        if( o > 0){
                              sucess=true;
                        }

                        if (sucess) {
                           conn.commit();
                       }

                        else {
                           conn.rollback();
```

```
                        }

                        conn.setAutoCommit(true);

                  }
                  }
            }
            return sucess;
      }

      //metodo que introduce un comment en la base de datos
      public boolean introducirComment(String pagina,String creador, String
fechaCreacion, String idPropio, String idPost, int likes, String mensaje)
throws SQLException{
            boolean sucess=false;

            if (conn != null ){

                  if((mensaje.length() < 20000) && (creador.length() < 50)){

                  if(posibleComment(idPropio)==true){

      conn.setTransactionIsolation(Connection.TRANSACTION_REPEATABLE_READ);
                        conn.setAutoCommit(false);
                        String sql = "INSERT INTO Comment(pagina, idPropio,
idPost, creador, fechaCreacion, likes, mensaje)"+"VALUES (?,?,?,?,?,?,?)";
                     PreparedStatement stmt = conn.prepareStatement(sql);
                  stmt.setString(1, pagina);
                      stmt.setString(2, idPropio);
                      stmt.setString(3, idPost);
                      stmt.setString(4, creador);
                      stmt.setString(5, fechaCreacion);
                      stmt.setInt(6, likes);
                      stmt.setString(7, mensaje);
                        int o = stmt.executeUpdate();

                        if( o > 0){
                              sucess=true;
                        }

                        if (sucess) {
                           conn.commit();
                        }

                        else {
                           conn.rollback();
                        }

                        conn.setAutoCommit(true);
                  }
                  }
            }
            return sucess;
      }

      public List<InfoComment> comentarios (String idPost) throws
SQLException{
            List<InfoComment> cm = new ArrayList<InfoComment>();
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
```

```
            SimpleDateFormat sdh = new SimpleDateFormat("HH:mm:ss");

            if (conn != null){
                  String query = "select * from Comment where idPost=?";
                  PreparedStatement stmt = conn.prepareStatement(query);
                  stmt.setString(1, idPost);
                  ResultSet rs = stmt.executeQuery();

                  while(rs.next()){
                    InfoComment co = new InfoComment(
                    rs.getString("pagina"),
                    rs.getString("creador"),
                    sdf.format(rs.getDate("fechaCreacion"))+"
"+sdh.format(rs.getTime("fechaCreacion")),
                    rs.getString("idPropio"),
                    rs.getString("idPost"),
                    rs.getInt("likes"),
                    rs.getString("mensaje")
                                );
                    cm.add(co);
                   }

                  stmt.close();
            }
            return cm;
      }

      public List<InfoPost> listarSql(String pagina, String ini, String fin,
String filtro) throws SQLException{
            List<InfoPost> arch = new ArrayList<InfoPost>();
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
            SimpleDateFormat sdh = new SimpleDateFormat("HH:mm:ss");

            if (conn != null){

                  if(filtro.equals("--ninguno--")){
                        String query = "select * from Post where pagina=?
and fechaCreacion between ? and ?";
                        PreparedStatement stmt =
conn.prepareStatement(query);
                        stmt.setString(1, pagina);
                        stmt.setString(2, ini);
                        stmt.setString(3, fin);
                        ResultSet rs = stmt.executeQuery();

                        while(rs.next()){
                              String id = rs.getString("id");
                              InfoPost po = new InfoPost(
                              rs.getString("pagina"),
                              rs.getString("creador"),
                              sdf.format(rs.getDate("fechaCreacion"))+"
"+sdh.format(rs.getTime("fechaCreacion")),

      sdf.format(rs.getDate("fechaActualizacion"))+"
"+sdh.format(rs.getTime("fechaActualizacion")),
                              rs.getString("tipo"),
                              rs.getInt("tamanhoComments"),
                              id,
                              rs.getInt("likes"),
```

```
                                rs.getString("mensaje"),
                                comentarios(id)
                                );
                                arch.add(po);

                            }

                    stmt.close();
                    }

                    else{
                            String query = "select * from Post where pagina=?
and tipo=? and fechaCreacion between ? and ?";
                            PreparedStatement stmt =
conn.prepareStatement(query);
                            stmt.setString(1, pagina);
                            stmt.setString(2, filtro);
                            stmt.setString(3, ini);
                            stmt.setString(4, fin);
                            ResultSet rs = stmt.executeQuery();

                            while(rs.next()){
                                String id = rs.getString("id");
                                InfoPost po = new InfoPost(
                                rs.getString("pagina"),
                                rs.getString("creador"),
                                sdf.format(rs.getDate("fechaCreacion")),
                                sdf.format(rs.getDate("fechaActualizacion")),
                                rs.getString("tipo"),
                                rs.getInt("tamanhoComments"),
                                id,
                                rs.getInt("likes"),
                                rs.getString("mensaje"),
                                comentarios(id)
                                );
                                arch.add(po);

                                }

                            stmt.close();
                    }
            }
            return arch;
        }

        public String leerSql(List<InfoPost> arch) throws SQLException{
            String vuelta="vacio";
            Gson gson = new Gson();

            if (conn != null){
                    vuelta = gson.toJson(arch);
            }

            return vuelta;
        }
```

```
	public Reading actualizaReading (String inicio, String ultimo){

		Reading r = new Reading();
		r.since(inicio);
		r.until(ultimo);
		r.limit(250);
		return r;

	}

	public Reading commentReading (){

		Reading r = new Reading();
		r.limit(250);
		return r;

	}



	public List<String> listadoFechas (Calendar ini, Calendar ult){

		List<String> lista_fechas = new ArrayList<String>();
		SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
		Calendar aux;
		aux = ini;
		Date ul = ult.getTime();

		while(aux.getTime().getTime() < ul.getTime()){
			lista_fechas.add(sdf.format(aux.getTime()));
			aux.add(Calendar.HOUR, 10);

			if(aux.getTime().getTime() >= ul.getTime()){
				lista_fechas.add(sdf.format(ult.getTime()));
			}

		}

		return lista_fechas;
	}

	public List<String> listadoFechasComment (Calendar ini, Calendar ult){
		List<String> lista_fechas = new ArrayList<String>();
		SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
		Calendar aux;
		aux = ini;
		Date ul = ult.getTime();

		while(aux.getTime().getTime() < ul.getTime()){
			lista_fechas.add(sdf.format(aux.getTime()));
			aux.add(Calendar.HOUR, 1);

			if(aux.getTime().getTime() >= ul.getTime()){
				lista_fechas.add(sdf.format(ult.getTime()));
			}

		}
```

```
            return lista_fechas;
      }



      public List<InfoComment> listadoComentariosPost (Post p, int
iteraciones, String pagina) throws FacebookException, ParseException{
            String id = p.getId();
            int auxInt = 1;
            Calendar auxC = Calendar.getInstance();
            JsonParser parser = new JsonParser();
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
            String fechaCreacionPost = sdf.format(p.getCreatedTime());
            String fechaActualizacion = sdf.format(p.getUpdatedTime());
            List<InfoComment> aux = new ArrayList<InfoComment>();
            String cursor1="";
            String cursor2="";

            if(iteraciones>0){

                  try{

                  while(auxInt<=iteraciones){
                        BatchRequests<BatchRequest> batch = new
BatchRequests<BatchRequest>();
                        int size = 0;

                        if(auxInt==1){
                              batch.add(new BatchRequest(RequestMethod.GET,
p.getId()+"?fields=comments.limit(250)"));
                              batch.add(new BatchRequest(RequestMethod.GET,
p.getId()+"?fields=comments.limit(250){like_count}"));
                              List<BatchResponse> results =
facebook.executeBatch(batch);
                              BatchResponse result1 = results.get(0);
                              BatchResponse result2 = results.get(1);
                              String jsonString1 = result1.asString();
                              String jsonString2 = result2.asString();
                              JsonElement arrayElement1 =
parser.parse(jsonString1);
                              JsonElement arrayElement2 =
parser.parse(jsonString2);
                              size =
arrayElement1.getAsJsonObject().get("comments").getAsJsonObject().get("data")
.getAsJsonArray().size();

                              for(int i=0; i<size; i++){
                                    String creador =
arrayElement1.getAsJsonObject().get("comments").getAsJsonObject().get("data")
.getAsJsonArray().get(i).getAsJsonObject().get("from").getAsJsonObject().get(
"name").getAsString();
                                    String idPropio =
arrayElement1.getAsJsonObject().get("comments").getAsJsonObject().get("data")
.getAsJsonArray().get(i).getAsJsonObject().get("id").getAsString();
```

```
                                        String mensaje =
arrayElement1.getAsJsonObject().get("comments").getAsJsonObject().get("data")
.getAsJsonArray().get(i).getAsJsonObject().get("message").getAsString();
                                        String f =
arrayElement1.getAsJsonObject().get("comments").getAsJsonObject().get("data")
.getAsJsonArray().get(i).getAsJsonObject().get("created_time").getAsString();
                                        f = f.substring(0, 10)+"
"+f.substring(11, 19);
                                        auxC.setTime(sdf.parse(f));
                                        auxC.add(Calendar.HOUR,2);
                                        f = sdf.format(auxC.getTime());
                                        int likes =
arrayElement2.getAsJsonObject().get("comments").getAsJsonObject().get("data")
.getAsJsonArray().get(i).getAsJsonObject().get("like_count").getAsInt();
                                        InfoComment inf = new
InfoComment(pagina, creador, f, idPropio, id, likes, mensaje);
                                        aux.add(inf);
                                }

                                cursor1 =
arrayElement1.getAsJsonObject().get("comments").getAsJsonObject().get("paging
").getAsJsonObject().get("cursors").getAsJsonObject().get("after").getAsStrin
g();
                                cursor2 =
arrayElement2.getAsJsonObject().get("comments").getAsJsonObject().get("paging
").getAsJsonObject().get("cursors").getAsJsonObject().get("after").getAsStrin
g();
                        }

                        else{
                                batch.add(new BatchRequest(RequestMethod.GET,
p.getId()+"/comments?limit=250&order=chronological&after="+cursor1));
                                batch.add(new BatchRequest(RequestMethod.GET,
p.getId()+"/comments?limit=250&fields=like_count&order=chronological&after="+
cursor2));
                                List<BatchResponse> results =
facebook.executeBatch(batch);
                                BatchResponse result1 = results.get(0);
                                BatchResponse result2 = results.get(1);
                                String jsonString1 = result1.asString();
                                String jsonString2 = result2.asString();
                                JsonElement arrayElement1 =
parser.parse(jsonString1);
                                JsonElement arrayElement2 =
parser.parse(jsonString2);
                                size =
arrayElement1.getAsJsonObject().get("data").getAsJsonArray().size();

                                for(int i=0; i<size; i++){
                                        String creador =
arrayElement1.getAsJsonObject().get("data").getAsJsonArray().get(i).getAsJson
Object().get("from").getAsJsonObject().get("name").getAsString();
                                        String idPropio =
arrayElement1.getAsJsonObject().get("data").getAsJsonArray().get(i).getAsJson
Object().get("id").getAsString();
                                        String mensaje =
arrayElement1.getAsJsonObject().get("data").getAsJsonArray().get(i).getAsJson
Object().get("message").getAsString();
```

```
                                        String f =
arrayElement1.getAsJsonObject().get("data").getAsJsonArray().get(i).getAsJson
Object().get("created_time").getAsString();
                                        f = f.substring(0, 10)+"
"+f.substring(11, 19);
                                        auxC.setTime(sdf.parse(f));
                                        auxC.add(Calendar.HOUR,2);
                                        f = sdf.format(auxC.getTime());
                                        int likes =
arrayElement2.getAsJsonObject().get("data").getAsJsonArray().get(i).getAsJson
Object().get("like_count").getAsInt();
                                        InfoComment inf = new
InfoComment(pagina, creador, f, idPropio, p.getId(), likes, mensaje);
                                        aux.add(inf);
                                }

                                cursor1 =
arrayElement1.getAsJsonObject().get("paging").getAsJsonObject().get("cursors"
).getAsJsonObject().get("after").getAsString();
                                cursor2 =
arrayElement2.getAsJsonObject().get("paging").getAsJsonObject().get("cursors"
).getAsJsonObject().get("after").getAsString();

                        }

                        auxInt++;
                }
                }

                catch(NullPointerException e){
                        System.out.println("Excepción cazada");
                }

        }
        return aux;
        }

        public int likesDePost (Post p, int iteraciones) throws
FacebookException{
                int likes = 0;
                int aux = 0;
                int cuenta= 1;
                String cursor ="";
                JsonParser parser = new JsonParser();
                BatchRequests<BatchRequest> batch = new
BatchRequests<BatchRequest>();
                batch.add(new BatchRequest(RequestMethod.GET,
p.getId()+"?fields=likes.limit(1000)"));
                List<BatchResponse> results =
facebook.executeBatch(batch);
                BatchResponse result1 = results.get(0);
                String jsonString = result1.asString();
                JsonElement arrayElement = parser.parse(jsonString);

                try{
                        likes =
arrayElement.getAsJsonObject().get("likes").getAsJsonObject().get("data").get
AsJsonArray().size();
```

```
                        cursor =
arrayElement.getAsJsonObject().get("likes").getAsJsonObject().get("paging").g
etAsJsonObject().get("cursors").getAsJsonObject().get("after").getAsString();
                  }

                  catch(NullPointerException e){
                        likes = 0;
                        System.out.println("Excepción cazada");
                  }

                  if(likes==1000 && iteraciones>1){

                        try{

                              while(cuenta < iteraciones){
                              BatchRequests<BatchRequest> batch2 = new
BatchRequests<BatchRequest>();
                              batch2.add(new
BatchRequest(RequestMethod.GET,
p.getId()+"/likes?limit=1000&after="+cursor));
                              List<BatchResponse> results2 =
facebook.executeBatch(batch2);
                              BatchResponse result2 = results2.get(0);
                              String jsonString2 = result2.asString();
                              arrayElement = parser.parse(jsonString2);
                              aux =
arrayElement.getAsJsonObject().get("data").getAsJsonArray().size();
                              cursor =
arrayElement.getAsJsonObject().get("paging").getAsJsonObject().get("cursors")
.getAsJsonObject().get("after").getAsString();
                              likes = likes + aux;
                              cuenta ++;
                              }

                        }

                        catch(NullPointerException e){
                              System.out.println("Excepción cazada");
                        }

                  }

                  return likes;
            }

}
```

Clases encargadas de las peticiones de la Aplicación web:

-Basico.java :

```
import Manager.Manager;
import Manager.InfoComment;
import Manager.InfoPost;

import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
```

```
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.google.gson.Gson;
import facebook4j.FacebookException;
import facebook4j.Page;
import facebook4j.Post;
import facebook4j.Reading;
import facebook4j.ResponseList;

	public class Basico extends HttpServlet{


		@Override
		protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
				throws ServletException, IOException {

			String nombre = req.getParameter("nombre");
			String diaIni = req.getParameter("diaIni");
			String mesIni = req.getParameter("mesIni");
			String anoIni = req.getParameter("anoIni");
			String diaFin = req.getParameter("diaFin");
			String mesFin = req.getParameter("mesFin");
			String anoFin = req.getParameter("anoFin");
			String ini = anoIni+"-"+mesIni+"-"+diaIni+" 00:00:00";
			String fin = anoFin+"-"+mesFin+"-"+diaFin+" 00:00:00";

			try {
				response(req, resp, nombre, ini, fin);
			}

			catch (ParseException e) {
				e.printStackTrace();
				resp.sendRedirect("fallo.html");
			}

			catch (FacebookException e) {
				e.printStackTrace();
				resp.sendRedirect("fallo.html");
			}
		}

		private void response(HttpServletRequest req, HttpServletResponse
resp, String nombre, String ini, String fin)
				throws IOException, ParseException,
ServletException, FacebookException {

			long inic = System.currentTimeMillis();
			int numeroPost = 0;
			int numeroComments = 0;
			Gson gson = new Gson();
			List<InfoPost> arch = new ArrayList<InfoPost>();
			Page pagina = null;
```

```
                    Manager m = new Manager();
                    m.autenticar();

                    try {
                          pagina = m.facebook().getPage(nombre);
                    }

                    catch (FacebookException e1) {
                          e1.printStackTrace();
                    }

                    String comprobar = pagina.getName();
                    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
                    Calendar inicio = Calendar.getInstance();
                    Calendar ultimo = Calendar.getInstance();
                    inicio.setTime(sdf.parse(ini));
                    inicio.add(Calendar.HOUR, -2);
                    ultimo.setTime(sdf.parse(fin));
                    ultimo.add(Calendar.HOUR, -2);

                    if(inicio.before(ultimo)==true){
                          String vuelta="\n";
                          String idPagina = "";
                          Reading god = new Reading();
                          List<String> lista_fechas = new
ArrayList<String>();
                          lista_fechas = m.listadoFechas (inicio, ultimo);

                          for(int ll = 0; ll<lista_fechas.size()-1; ll++){
                                god= m.actualizaReading(lista_fechas.get(ll),
lista_fechas.get(ll+1));
                                ResponseList<Post> k = null;

                                try {
                                      k = m.facebook().getGroupFeed(nombre,
god);

      idPagina=m.facebook().getPage(nombre).getId();
                                }

                                catch (FacebookException e) {
                                      e.printStackTrace();
                                      resp.sendRedirect("fallo.html");
                                }

                                for(int w=0; w < k.size(); w++){

                                Post p = k.get(w);
                                boolean fg = true;

                                try{

                                      if(p.getMessage().isEmpty()){
                                            fg = false;
                                      }

                                      if(p.getMessage() == null){
                                            fg = false;
```

```
                                        }

                                }

                                catch (NullPointerException e) {
                                        fg = false;
                                        e.printStackTrace();
                                }

                                if(fg){
                                        vuelta = vuelta + "- " +
p.getMessage() + "\n";
                                        List<InfoComment> cm = new
ArrayList<InfoComment>();
                                        cm = m.listadoComentariosPost(p, 1,
nombre);
                                        String fechaCreacionPost =
sdf.format(p.getCreatedTime());
                                        String fechaActualizacion =
sdf.format(p.getUpdatedTime());
                                        String tipoEstado = "nulo";
                                        String creadorP =
p.getFrom().getName();
                                        int likesP = m.likesDePost(p, 1);

                                        if(creadorP.equals(comprobar)){
                                                tipoEstado ="PagePost";
                                        }

                                        else{
                                                tipoEstado ="Post to Page";
                                        }

                                        String id = p.getId();
                                        String mensajePost = p.getMessage();

                                        if(cm.size() != 0){
                                                numeroComments = numeroComments
+ cm.size();
                                                InfoPost infpIf = new
InfoPost(nombre, creadorP, fechaCreacionPost, fechaActualizacion, tipoEstado,
cm.size(),  id, likesP,mensajePost, cm);
                                                arch.add(infpIf);
                                        }

                                        else{
                                                InfoPost infpElse = new
InfoPost(nombre, creadorP, fechaCreacionPost, fechaActualizacion, tipoEstado,
cm.size(), id, likesP, mensajePost, cm);
                                                arch.add(infpElse);
                                        }

                                        numeroPost++;

                                }
                                }
                        }

                        String formatoJSON = gson.toJson(arch);
```

```
                        System.out.println("fin");
                        long finis = System.currentTimeMillis();
                        long resul = finis-inic;
                        System.out.println("Tiempo de ejecución de búsqueda
básica en milisegundos: " + resul);
                        System.out.println("Número de post: " +
numeroPost);
                        System.out.println("Número de comentarios: " +
numeroComments);
                        req.setAttribute("data", formatoJSON);
                        req.setAttribute("lista", arch);
                        RequestDispatcher rd =
req.getRequestDispatcher("guardarJson.jsp");
                        rd.forward(req, resp);

                  }

                  else{
                        resp.sendRedirect("fallofecha.html");
                  }

       }
      }
```

-Avanzado.java :

```
import Manager.InfoComment;
import Manager.InfoPost;
import Manager.Manager;
import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.google.gson.Gson;
import facebook4j.FacebookException;
import facebook4j.Page;
import facebook4j.Post;
import facebook4j.Reading;
import facebook4j.ResponseList;

      public class Avanzado extends HttpServlet{


       @Override
       protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                        throws ServletException, IOException {

                  String nombre = req.getParameter("nombre");
                  String diaIni = req.getParameter("diaIni");
```

```
                String mesIni = req.getParameter("mesIni");
                String anoIni = req.getParameter("anoIni");
                String horaIni = req.getParameter("horaIni");
                String minIni = req.getParameter("minIni");
                String segIni = req.getParameter("segIni");
                String diaFin = req.getParameter("diaFin");
                String mesFin = req.getParameter("mesFin");
                String anoFin = req.getParameter("anoFin");
                String horaFin = req.getParameter("horaFin");
                String minFin = req.getParameter("minFin");
                String segFin = req.getParameter("segFin");
                String filtro = req.getParameter("filtro");
                int iteracionesLikes =
Integer.parseInt(req.getParameter("IteracionesLikes"));
                int iteracionesComments =
Integer.parseInt(req.getParameter("IteracionesPost"));
                String ini = anoIni+"-"+mesIni+"-"+diaIni+"
"+horaIni+":"+minIni+":"+segIni;
                String fin = anoFin+"-"+mesFin+"-"+diaFin+"
"+horaFin+":"+minFin+":"+segFin;

                try {
                        response(req, resp, nombre, ini, fin, filtro,
iteracionesLikes, iteracionesComments);
                }

                catch (ParseException e) {
                        e.printStackTrace();
                        resp.sendRedirect("fallo.html");

                }

                catch (FacebookException e) {
                        e.printStackTrace();
                        resp.sendRedirect("fallo.html");
                }
        }

        private void response(HttpServletRequest req, HttpServletResponse
resp, String nombre, String ini, String fin, String filtro, int itLikes, int
itComments)
                        throws IOException, ParseException,
ServletException, FacebookException {

                long inic = System.currentTimeMillis();
                int numeroPost = 0;
                int numeroComments = 0;
                Gson gson = new Gson();
                Page pagina = null;
                List<InfoPost> arch = new ArrayList<InfoPost>();
                Manager m = new Manager();
                m.autenticar();

                try {
                        pagina = m.facebook().getPage(nombre);
                }

                catch (FacebookException e1) {
                        e1.printStackTrace();
```

```
                    }

                    String comprobar = pagina.getName();
                    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
                    Calendar inicio = Calendar.getInstance();
                    Calendar ultimo = Calendar.getInstance();
                    inicio.setTime(sdf.parse(ini));
                    inicio.add(Calendar.HOUR, -2);
                    ultimo.setTime(sdf.parse(fin));
                    ultimo.add(Calendar.HOUR, -2);

                    if(inicio.before(ultimo)==true){
                          String vuelta="\n";
                          String idPagina = "";
                          Reading god = new Reading();
                          List<String> lista_fechas = new
ArrayList<String>();
                          lista_fechas = m.listadoFechas (inicio, ultimo);

                          for(int ll = 0; ll<lista_fechas.size()-1; ll++){
                                god= m.actualizaReading(lista_fechas.get(ll),
lista_fechas.get(ll+1));
                                ResponseList<Post> k = null;

                                try {
                                      k = m.facebook().getGroupFeed(nombre,
god);

      idPagina=m.facebook().getPage(nombre).getId();
                                }

                                catch (FacebookException e) {
                                      e.printStackTrace();
                                      resp.sendRedirect("fallo.html");
                                }

                                for(int w=0; w < k.size(); w++){
                                Post p = k.get(w);
                                boolean fg = true;

                                try{

                                      if(p.getMessage().isEmpty()){
                                            fg = false;
                                      }

                                      if(p.getMessage() == null){
                                            fg = false;
                                      }
                                }

                                catch (NullPointerException e) {
                                      fg = false;
                                      e.printStackTrace();
                                }

                                if(fg){
                                      if(filtro.equals("--ninguno--") ||
```

```
                                        (filtro.equals("Post to Page") &&
      p.getFrom().getName().equals(comprobar) == false )||
                                        (filtro.equals("PagePost") &&
p.getFrom().getName().equals(comprobar))
                                        ){
                                                vuelta = vuelta + "- " +
p.getMessage() + "\n";
                                                List<InfoComment> cm =
new ArrayList<InfoComment>();
                                                cm=
m.listadoComentariosPost(p, itComments, nombre);
                                                String fechaCreacionPost
= sdf.format(p.getCreatedTime());
                                                String fechaActualizacion
= sdf.format(p.getUpdatedTime());
                                                String tipoEstado =
"nulo";
                                                String creadorP =
p.getFrom().getName();
                                                int likesP =
m.likesDePost(p, itLikes);


      if(creadorP.equals(comprobar)){
                                                      tipoEstado
="PagePost";
                                                }


                                                else{
                                                      tipoEstado ="Post
to Page";
                                                }


                                                String id = p.getId();
                                                String mensajePost =
p.getMessage();


                                                if(cm.size() != 0){
                                                      numeroComments =
numeroComments + cm.size();
                                                      InfoPost infpIf =
new InfoPost(nombre, creadorP, fechaCreacionPost, fechaActualizacion,
tipoEstado, cm.size(),  id, likesP,mensajePost, cm);
                                                      arch.add(infpIf);
                                                }


                                                else{
                                                      InfoPost infpElse =
new InfoPost(nombre, creadorP, fechaCreacionPost, fechaActualizacion,
tipoEstado,  cm.size(), id, likesP, mensajePost, cm);
                                                      arch.add(infpElse);
                                                }


                                        }
                                  }

                                  numeroPost++;

                                  }
```

```
                        }

                        String formatoJSON = gson.toJson(arch);
                        long finis = System.currentTimeMillis();
                        long resul = finis-inic;
                        System.out.println("Tiempo de ejecución de búsqueda
avanzada en milisegundos: " + resul);
                        System.out.println("Número de post: " +
numeroPost);
                        System.out.println("Número de comentarios: " +
numeroComments);
                        req.setAttribute("data", formatoJSON);
                        req.setAttribute("lista", arch);
                        RequestDispatcher rd =
req.getRequestDispatcher("guardarJson.jsp");
                        rd.forward(req, resp);

                  }

                  else{
                        resp.sendRedirect("fallofecha.html");
                  }

       }
      }
```

-GuardarJson.java :

```
import java.io.IOException;
import java.text.ParseException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


public class GuardarJson extends HttpServlet{

      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
            throws ServletException, IOException {
             String tete = req.getParameter("datos");

             try {
                   response(req, resp, tete);
             }

             catch (ParseException e) {
                   e.printStackTrace();
             }

      }

      private void response(HttpServletRequest req, HttpServletResponse
resp, String tete)
                  throws IOException, ParseException, ServletException {
            req.setAttribute("data", tete);
```

```
            RequestDispatcher rd =
req.getRequestDispatcher("formuGuardarJson.jsp");
            rd.forward(req, resp);
      }

}
```

-GuardarSql.java :

```
import java.io.IOException;
import java.sql.SQLException;
import java.text.ParseException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.google.gson.Gson;
import com.google.gson.JsonElement;
import com.google.gson.JsonParser;
import Manager.Manager;

public class GuardarSql extends HttpServlet{
      protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
            throws ServletException, IOException {
             boolean exito = false;
             Manager m = new Manager();
             Gson gson = new Gson();
             JsonParser parser = new JsonParser();
             String tete = req.getParameter("datos");
        JsonElement e = parser.parse(tete);
        int tamanho = e.getAsJsonArray().size();

        try {
                    m.conectar();

                    for(int i = 0; i<tamanho; i++){
                       String pagina =
e.getAsJsonArray().get(i).getAsJsonObject().get("pagina").getAsString();
                       String creador =
e.getAsJsonArray().get(i).getAsJsonObject().get("creador").getAsString();
                            String fechaCreacion =
e.getAsJsonArray().get(i).getAsJsonObject().get("fechaCreacion").getAsString(
);
                            String fechaActualizacion =
e.getAsJsonArray().get(i).getAsJsonObject().get("fechaActualizacion").getAsSt
ring();
                            String id =
e.getAsJsonArray().get(i).getAsJsonObject().get("id").getAsString();
                            String tipoEstado =
e.getAsJsonArray().get(i).getAsJsonObject().get("tipoEstado").getAsString();
                            int tamanhoComentarios =
e.getAsJsonArray().get(i).getAsJsonObject().get("tamanoComentarios").getAsInt
();
                            int likes =
e.getAsJsonArray().get(i).getAsJsonObject().get("likes").getAsInt();
                            String mensaje =
e.getAsJsonArray().get(i).getAsJsonObject().get("mensaje").getAsString();
```

```
                        m.introducirPost(pagina, creador, fechaCreacion,
fechaActualizacion, id, tipoEstado, tamanhoComentarios, likes, mensaje);

                        if(tamanhoComentarios>0){

                              for(int j = 0; j<tamanhoComentarios; j++){
                                    String pagina2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("pagina").getAsString();
                                    String fechaCreacion2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("fechaCreacion").getAsString();
                                    String idPropio =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("idPropio").getAsString();
                                    String idPost =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("idPost").getAsString();
                                    String creador2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("creador").getAsString();
                                    int likes2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("likes").getAsInt();
                                    String mensaje2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("mensaje").getAsString();
                                    m.introducirComment(pagina2, creador2,
fechaCreacion2, idPropio, idPost,  likes2, mensaje2);
                              }

                        }

                         m.actualizaTamanhoPost(id, likes);
                    }

                  exito=true;
                  m.desconectar();
            }

        catch (ClassNotFoundException | SQLException e1) {
                  e1.printStackTrace();
            }

        finally{

            try {
                        response(req, resp, exito);
                  }

            catch (ParseException e1) {
                        e1.printStackTrace();
                  }

        }

      }
```

```
	private void response(HttpServletRequest req, HttpServletResponse
resp, boolean exito)
			throws IOException, ParseException, ServletException {

		if(exito){
			resp.sendRedirect("exito.html");
		}

		else{
			resp.sendRedirect("fallo.html");
		}

	}
}
```

-GuardarDoble.java :

```
import java.io.IOException;
import java.sql.SQLException;
import java.text.ParseException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import Manager.Manager;
import com.google.gson.Gson;
import com.google.gson.JsonElement;
import com.google.gson.JsonParser;

public class GuardarDoble extends HttpServlet{protected void
doPost(HttpServletRequest req, HttpServletResponse resp)
	  throws ServletException, IOException {
	boolean exito = false;
	Manager m = new Manager();
	Gson gson = new Gson();
	JsonParser parser = new JsonParser();
	String tete = req.getParameter("datos");
    JsonElement e = parser.parse(tete);
    int tamanho = e.getAsJsonArray().size();

    try {
		m.conectar();

		for(int i = 0; i<tamanho; i++){
		   String pagina =
e.getAsJsonArray().get(i).getAsJsonObject().get("pagina").getAsString();
		   String creador =
e.getAsJsonArray().get(i).getAsJsonObject().get("creador").getAsString();
		   String fechaCreacion =
e.getAsJsonArray().get(i).getAsJsonObject().get("fechaCreacion").getAsString(
);
		   String fechaActualizacion =
e.getAsJsonArray().get(i).getAsJsonObject().get("fechaActualizacion").getAsSt
ring();
		   String id =
e.getAsJsonArray().get(i).getAsJsonObject().get("id").getAsString();
		   String tipoEstado =
e.getAsJsonArray().get(i).getAsJsonObject().get("tipoEstado").getAsString();
```

```
                    int tamanhoComentarios =
e.getAsJsonArray().get(i).getAsJsonObject().get("tamanoComentarios").getAsInt
();
                    int likes =
e.getAsJsonArray().get(i).getAsJsonObject().get("likes").getAsInt();
                    String mensaje =
e.getAsJsonArray().get(i).getAsJsonObject().get("mensaje").getAsString();
                    m.introducirPost(pagina, creador, fechaCreacion,
fechaActualizacion, id, tipoEstado, tamanhoComentarios, likes, mensaje);

                  if(tamanhoComentarios>0){

                        for(int j = 0; j<tamanhoComentarios; j++){
                              String pagina2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("pagina").getAsString();
                              String fechaCreacion2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("fechaCreacion").getAsString();
                              String idPropio =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("idPropio").getAsString();
                              String idPost =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("idPost").getAsString();
                              String creador2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("creador").getAsString();
                              int likes2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("likes").getAsInt();
                              String mensaje2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("mensaje").getAsString();
                              m.introducirComment(pagina2, creador2,
fechaCreacion2, idPropio, idPost,  likes2, mensaje2);
                        }

                  }

                    m.actualizaTamanhoPost(id, likes);
              }

            exito=true;
            m.desconectar();

      }

    catch (ClassNotFoundException | SQLException e1) {
            e1.printStackTrace();
      }

    finally{

      try {
                  response(req, resp, exito, tete);
            }

      catch (ParseException e1) {
```

```
                    e1.printStackTrace();
             }

    }

}

private void response(HttpServletRequest req, HttpServletResponse resp,
boolean exito, String formatoJSON)
             throws IOException, ParseException, ServletException {

       if(exito){
             req.setAttribute("data", formatoJSON);
             RequestDispatcher rd =
req.getRequestDispatcher("formuGuardarJson2.jsp");
             rd.forward(req, resp);
       }

       else{
              resp.sendRedirect("fallo.html");
       }

}
}
```

-GuardarArchivo.java :

```
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.text.ParseException;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


public class GuardarArchivo extends HttpServlet{

       protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
            throws ServletException, IOException{
             String nombre =req.getParameter("nombre");
             String fichero = nombre+".json";
             ServletOutputStream sOutStream = resp.getOutputStream();

             try {
                    response(req, resp, sOutStream, fichero);
             }

             catch (ParseException e) {
                    e.printStackTrace();
             }

       }
```

```
      private void response(HttpServletRequest req, HttpServletResponse
resp, ServletOutputStream sOutStream, String fichero)
                    throws IOException, ParseException, ServletException {
        BufferedInputStream bis = null;
        BufferedOutputStream bos = null;
        int tam = (int) new File(fichero).length();
        bis = new BufferedInputStream(new FileInputStream(fichero));

        try {
              resp.setContentType("application/json");
              resp.setContentLength(tam);
              resp.setHeader("Content-Disposition", "attachment;filename=\""
+ fichero + "\"");
              bos = new BufferedOutputStream(sOutStream);
              byte[] array = new byte[1000];
              int leidos = bis.read(array);

              while (leidos > 0) {
                  bos.write(array, 0, leidos);
                  leidos = bis.read(array);
              }

       }

        catch (Exception e) {
            e.printStackTrace();
            String ErrorStr = "Error Streaming the Data";
            sOutStream.print(ErrorStr);
        }

        finally {

            if (bis != null) {
                bis.close();
            }

            if (bos != null) {
                bos.close();
            }

            if (sOutStream != null) {
             sOutStream.flush();
             sOutStream.close();
            }

        }

      }
}
```

-CargarJson.java :

```
import Manager.InfoComment;
import Manager.InfoPost;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.text.ParseException;
```

```
import java.util.ArrayList;
import java.util.List;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import com.google.gson.Gson;
import com.google.gson.JsonElement;
import com.google.gson.JsonParser;

       public class CargarJson extends HttpServlet{

        protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
                        throws ServletException, IOException {

                   Gson gson = new Gson();
                   JsonParser parser = new JsonParser();
                   String aux="";
                   String fin;
                   String tete = req.getParameter("archivo");
                   File archivo = null;
                FileReader fr = null;
                BufferedReader br = null;

                try {
                      archivo = new File (tete);
                      fr = new FileReader (archivo);
                      br = new BufferedReader(fr);
                      String linea;

                      while((linea=br.readLine())!=null){
                          aux=aux+linea;
                      }

                      JsonElement arrayElement = parser.parse(aux);
                      fin=aux;
                      int tamanho = arrayElement.getAsJsonArray().size();
                      response(req, resp, arrayElement, tamanho, fin);
                   }

                   catch(Exception e){
                      e.printStackTrace();
                   }

                finally{

                      try{

                         if( null != fr ){
                            fr.close();
                         }

                      }

                      catch (Exception e2){
                         e2.printStackTrace();
                      }
```

```
                    }

        }

         private void response(HttpServletRequest req, HttpServletResponse
resp, JsonElement e, int tamanho, String fin)
                              throws IOException, ParseException,
ServletException {
               long inic = System.currentTimeMillis();
               List<InfoPost> arch = new ArrayList<InfoPost>();

               for(int i = 0; i<tamanho; i++){
                   String pagina =
e.getAsJsonArray().get(i).getAsJsonObject().get("pagina").getAsString();
                   String creador =
e.getAsJsonArray().get(i).getAsJsonObject().get("creador").getAsString();
                     String fechaCreacion =
e.getAsJsonArray().get(i).getAsJsonObject().get("fechaCreacion").getAsString(
);
                     String fechaActualizacion =
e.getAsJsonArray().get(i).getAsJsonObject().get("fechaActualizacion").getAsSt
ring();
                     String id =
e.getAsJsonArray().get(i).getAsJsonObject().get("id").getAsString();
                     String tipoEstado =
e.getAsJsonArray().get(i).getAsJsonObject().get("tipoEstado").getAsString();
                     String mensaje =
e.getAsJsonArray().get(i).getAsJsonObject().get("mensaje").getAsString();
                     int likes =
e.getAsJsonArray().get(i).getAsJsonObject().get("likes").getAsInt();
                     int tamanhoComentarios =
e.getAsJsonArray().get(i).getAsJsonObject().get("tamanoComentarios").getAsInt
();
                     List<InfoComment> cm = new ArrayList<InfoComment>();

                    if(tamanhoComentarios>0){

                          for(int j = 0; j<tamanhoComentarios; j++){
                                String pagina2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("pagina").getAsString();
                                String creador2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("creador").getAsString();
                                String fechaCreacion2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("fechaCreacion").getAsString();
                                String idPropio =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("idPropio").getAsString();
                                String idPost =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("idPost").getAsString();
                                int likes2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("likes").getAsInt();
```

```
                                String mensaje2 =
e.getAsJsonArray().get(i).getAsJsonObject().get("comentarios").getAsJsonArray
().get(j).getAsJsonObject().get("mensaje").getAsString();
                                InfoComment c = new InfoComment(pagina2,
creador2, fechaCreacion2, idPropio, idPost, likes2, mensaje2);
                                cm.add(c);
                            }

                    }

                      InfoPost p = new InfoPost(pagina, creador,
fechaCreacion, fechaActualizacion, tipoEstado, tamanhoComentarios, id, likes,
mensaje, cm);
                      arch.add(p);
              }

                    long finis = System.currentTimeMillis();
                    long resul = finis-inic;
                    System.out.println("Tiempo de ejecución de cargar json en
milisegundos: " + resul);
                    req.setAttribute("data", fin);
                    req.setAttribute("lista", arch);
                    RequestDispatcher rd =
req.getRequestDispatcher("guardarSqlJson.jsp");
                    rd.forward(req, resp);

        }
       }
```

-CargarSql.java :

```
import java.io.IOException;
import java.sql.SQLException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import Manager.InfoPost;
import Manager.Manager;

public class CargarSql extends HttpServlet{

       protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
             throws ServletException, IOException {
       String nombre = req.getParameter("nombre");
       String diaIni = req.getParameter("diaIni");
       String mesIni = req.getParameter("mesIni");
       String anoIni = req.getParameter("anoIni");
       String horaIni = req.getParameter("horaIni");
       String minIni = req.getParameter("minIni");
       String segIni = req.getParameter("segIni");
```

```
        String diaFin = req.getParameter("diaFin");
        String mesFin = req.getParameter("mesFin");
        String anoFin = req.getParameter("anoFin");
        String horaFin = req.getParameter("horaFin");
        String minFin = req.getParameter("minFin");
        String segFin = req.getParameter("segFin");
        String filtro = req.getParameter("filtro");
        String ini = anoIni+"-"+mesIni+"-"+diaIni+"
"+horaIni+":"+minIni+":"+segIni;
        String fin = anoFin+"-"+mesFin+"-"+diaFin+"
"+horaFin+":"+minFin+":"+segFin;

                try {
                        response(req, resp, nombre, ini, fin, filtro);
                }

                catch (ClassNotFoundException | ParseException | SQLException e)
{
                        e.printStackTrace();
                }

        }

         private void response(HttpServletRequest req, HttpServletResponse
resp, String nombre, String ini, String fin, String filtro)
                                throws IOException, ParseException,
ServletException, ClassNotFoundException, SQLException {

                        Calendar inicio = Calendar.getInstance();
                        Calendar ultimo = Calendar.getInstance();
                        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
                        inicio.setTime(sdf.parse(ini));
                        ultimo.setTime(sdf.parse(fin));

                        if(inicio.before(ultimo)==true){
                            long inic = System.currentTimeMillis();
                              Manager m = new Manager();
                              m.conectar();
                              List<InfoPost> arch = new ArrayList<InfoPost>();
                              arch = m.listarSql(nombre, ini, fin, filtro);
                              String Mandanga = m.leerSql(arch);
                              m.desconectar();
                              long finis = System.currentTimeMillis();
                              long resul = finis-inic;
                              System.out.println("Tiempo de ejecución de búsqueda
básica en milisegundos: " + resul);
                              req.setAttribute("data", Mandanga);
                              req.setAttribute("lista", arch);
                              RequestDispatcher rd =
req.getRequestDispatcher("guardarJsonDB.jsp");
                              rd.forward(req, resp);
                        }

                        else{
                              resp.sendRedirect("fallofecha.html");
                        }

         }
```

```
}
```

-Exito.java :

```
import java.io.FileWriter;
import java.io.IOException;
import java.text.ParseException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ExitoJson extends HttpServlet{

	protected void doPost(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
			 String tete = req.getParameter("datos");
			 String nombre =req.getParameter("nombre");

			 try {
				response(req, resp, tete, nombre);
			 }

			 catch (ParseException e) {
				e.printStackTrace();
			 }

	}

	private void response(HttpServletRequest req, HttpServletResponse
resp, String tete, String nombre)
				throws IOException, ParseException, ServletException {
			 FileWriter fw = new FileWriter(nombre+".json");
			 fw.write(tete);
			 fw.close();
			 req.setAttribute("nombre", nombre);
			 RequestDispatcher rd =
req.getRequestDispatcher("guardarArchivo");
			 rd.forward(req, resp);
	}

}
```