Grado en ingeniería de sistemas audiovisuales

TRABAJO FIN DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA EL ANÁLISIS DE DATOS DE PÁGINAS DE FACEBOOK MEDIANTE TECNOLOGÍAS WEB

Autor: Patricia Callejo Pinardo

Tutor: Juan Miguel Carrascosa Amigo
Director: Rubén Cuevas Rumín

Leganés, Septiembre 2015

Agradecimientos

En primer lugar quiero agradecer a mis tutores, Juan Miguel Carrascosa y Rubén Cuevas por brindarme la gran oportunidad de poder realizar este proyecto y contar con su apoyo y ánimos constantes. Gracias a ellos he desarrollado este proyecto con una inmensa ilusión y optimismo.

Dar las gracias particularmente a Pablo y Christian, que sin sus consejos y ayuda no habría sido posible lograr esta meta.

También quiero expresar mi gratitud a toda mi familia, especialmente a mis padres, a mi hermano y a mi abuela que siempre han estado pendientes de mi vida académica y nunca han dejado de apoyarme y siempre me han animado a seguir adelante.

Gracias a Sergio, por su apoyo incondicional en esta etapa de mi vida, quien me ha ayudado en todo momento para llegar hasta aquí.

Agradecer también a mis compañeros de la Universidad, a aquellos que han estado conmigo desde el principio de la carrera a nivel académico como personal, nombrar concretamente a Alba, Marta, Ana, Alexandre y Andrés, gracias a ellos realizar esta carrera y este proyecto ha sido una gran experiencia.

Gracias a todos mis amigos que me han apoyado en cada momento difícil de este largo camino y me han tendido una mano sin dudarlo.

Para todas estas personas no tengo palabras suficientes para describir la admiración y afecto que siento por ellos.

Gracias a todos.

Resumen

Actualmente las redes sociales están en auge y en continuo crecimiento. Cada vez son más los usuarios que pertenecen a estas comunidades y hacen uso de ellas en su vida cotidiana. Revisar tus redes sociales en determinados momentos del día es un hábito que se ha creado de la necesidad de estar conectados. Este fenómeno ha aumentado con la expansión de los terminales móviles, que hacen posible tener al alcance de la mano su acceso en cualquier momento y lugar.

En este proyecto se presenta una aplicación centrada en una de las redes sociales de mayor extensión e importancia a día de hoy, Facebook. Se estima que más de un 80 % de la población tiene una cuenta en esta red social. Por eso, gracias a su gran extensión, es una fuente de información de la que se pueden realizar profundos análisis para describir patrones de los perfiles existentes.

La aplicación lleva a cabo un estudio de las páginas de Facebook, ofreciendo un análisis tanto de la información de las propias páginas como un perfilado de los usuarios que siguen dichas páginas.

El objetivo principal de la aplicación es dar acceso a aquellos individuos, usuarios o no de Facebook, que estén interesados en obtener información de un conjunto de páginas de un mismo sector elegido previamente.

El estudio se centra en un apartado específico dentro de las páginas de Facebook, llamado *Post to Page*. Esta sección la incluyen muchas empresas, marcas o asociaciones en sus páginas, con el fin de dar libertad de expresión a sus usuarios de una forma directa, facilitando a la entidad establecer una relación más cercana e interactiva con sus seguidores.

En este apartado los usuarios suelen expresar su agradecimiento o descontento con algún servicio ofrecido, también escriben mensajes de apoyo ante algún acontecimiento relacionado con la página o meras críticas que resultan ser constructivas para la empresa propietaria de la página en Facebook, ayudándoles a mejorar. Por este motivo, este trabajo de fin de grado se ha centrado en esta sección y ofrece un análisis de datos basado en la atención que reciben los usuarios por parte de las páginas ante sus comentarios y la popularidad de dichos comentarios. Por otra parte, también realiza un perfilado de los usuarios que más comentan en las páginas, con el fin de situarlas en un marco sociocultural.

Palabras clave: aplicación, crawler, Facebook, red social, minería de datos.

Abstract

Currently social networks are increasing very quickly. Each day there are more users that belong to those communities and use them regularly. Checking notifications in social networks several times a day has become a habit that generates the need to be connected to the net. This fact has increased with the expansion of mobile terminals that make the access possible anytime, anywhere.

This work introduces an application focused in one of the biggest and most important social network nowadays, Facebook. It has estimated more than 80 % of population has an account in this social network. Thanks to its big extension, it is a source of information from which can be obtained some data analysis and user patterns.

The proposed application performs a study of Facebook pages, providing an analysis of both the information of those pages, and a profile of the users following them.

The main objective of the application is to provide access to individuals, whether they are or not Facebook users, who are interested in obtaining a study of a set of pages in the same sector preselect by himself.

The study is focused on a particular section of the Facebook page called *Post to page*. This section is included by many companies, brands or associations in their websites in order to give free expression to their users on a direct way making easy an oncoming with their followers.

In this section, users often express their appreciation or dissatisfaction with any service offered. It is also used to show support to any related event in the page or constructive criticism for the company that owns the page. For this reason, this work is focused in that section, providing data analysis based on the attention to comments users by the pages and the popularity of those. In addition, it is also performed a profile of the users that make more comments in the pages in order to place them in a socio-cultural context.

Keywords: application, crawler, Facebook, social network, data mining.

Índice General

Índice de Figuras

Índice de Tablas

Glosario

La lista de los acrónimos utilizados en este proyecto es la siguiente:

AMQP Advanced Message Queuing Protocol

API Application Programming Interface

BOE Boletín Oficial del Estado

CPU Central Processing Unit (Unidad central de procesamiento)

CRUD Create, Read, Update and Delete (Crear, Obtener, Actualizar y Borrar)

HTTP Hypertext Transfer Protocol

IaaS Infrastructure as a Service

IP Internet Protocol

MVC Modelo-Vista-Controlador

Plugin Complemento

URL Uniform Resource Locator

VM Virtual Machine

XML eXtensible Markup Language (Lenguaje de marcas extensible)

WWW World Wide Web

Capítulo 1

Introducción

1.1. Motivación del proyecto

La comunicación es uno de los pilares fundamentales de la sociedad en la que vivimos, por ello es una de las vías de conocimiento y transmisión de información. Gracias a la evolución de la tecnología, han evolucionado también los medios de comunicación, abriendo paso a las redes sociales como un medio de comunicación abierto y cercano a la gente.

La acogida de las redes sociales ha sido tan grande que no dejan de crearse nuevas ideas para compartir el día a día. Están en continuo crecimiento, y más aún con la expansión del uso de los *smartphones*, que permiten tenerlas al alcance de la mano en cualquier momento y lugar. Se han convertido en un hábito de la sociedad, como medio de comunicación y como medio de expresión.

Centrándonos en las estadísticas, tal y como muestra *GlobalWebIndex* en su reporte anual de 2015 [10], hay varias plataformas como Instagram o Pinterest que han crecido exponencialmente en este último año, pero ninguna de ellas llega al nivel de Facebook. Esta web sigue siendo la red social más popular a nivel mundial, se estima que un 80 % de la población tiene una cuenta en esta red.

Estos hechos suponen una gran motivación en la realización de este proyecto, dado que el estudio realizado se basa en los datos proporcionados por Facebook, esperando obtener un gran volumen de datos fiables dada la gran dimensión de esta plataforma.

Se pretende crear una página web con un registro de usuarios, donde cada usuario pueda adquirir un estudio personalizado. Este estudio se dividirá en dos partes.

La primera parte realizará un análisis de las páginas de Facebook de su sección *Post to Page*. Este apartado recoge las publicaciones de los usuarios que han visitado la página para poner sus comentarios. Es una parte que no todas las páginas de Facebook tienen activas, por lo que se limitará el estudio a aquellas páginas que la contengan. Los datos se obtendrán consultando a la API de Facebook y se podrán obtener gráficas en relación a las contestaciones a dichos comentarios por parte de la página web, al tiempo de respuesta a los comentarios, y a la popularidad de los comentarios midiendo los "Me gusta" de cada uno.

La segunda parte realizará un estudio de los usuarios que más han publicado en la

sección comentada anteriormente, mediante un *crawler*[1] que visitará a cada uno de estos usuarios para coger su información pública. Una vez obtenidos los datos, se obtendrán las gráficas en relación a la edad de los usuarios, la localización y la ocupación profesional.

Con esta aplicación web se podrá definir las características de dichas páginas. Información muy útil de cara al diseño de patrones de empresas o asociaciones, que llevándolo al terreno de los propios integrantes de esta aplicación, podrán conocer los aspectos más destacables de las páginas.

Destacar que para la realización de la aplicación que hace posible este estudio, no sólo se debe realizar una aplicación web, si no que también hay que llevar a cabo la recopilación de los datos y su posterior procesado, para lo cual se tienen que desarrollar tareas de gran diversidad, que suponen un gran reto para la elaboración de este trabajo fin de grado.

1.2. Objetivos del proyecto

Con la realización de este proyecto se pretende conseguir una aplicación en la que los usuarios puedan llevar un registro de los análisis realizados a las diferentes páginas de Facebook, de forma que el propio usuario puede conocer las características de un conjunto de páginas de Facebook que deseen y cuál es el patrón de los usuarios que la siguen.

Se debe realizar una aplicación web que tenga las siguientes funcionalidades:

- Registro de nuevos usuarios y función *login/logout* de los usuarios.
- Opción de búsqueda de un conjunto de páginas de Facebook para su procesado.
- Procesado general de los datos de la página de Facebook.
- Procesado en profundidad del perfil de los usuarios.
- Historial de las búsquedas realizadas.
- Presentación de las gráficas de los resultados de cada una de las búsquedas.

Para poder realizar estas funcionalidades es necesario el desarrollo de varias partes totalmente distintas, que conformarían el backend de la aplicación. Se enumeran a continuación:

- Control del registro de usuarios y de cada una de sus búsquedas, almacenandolo en la base de datos.
- Consultas al API de Facebook para conseguir los datos de las páginas de Facebook.
- Desarrollo de un *crawler* que recorra cada uno de los usuarios para obtener los datos públicos que proporcionan.
- Distribución del trabajo en diferentes máquinas virtuales para optimizar el tiempo de procesado de los datos.
- Procesado de datos, para calcular las estadísticas y representar las gráficas.

[1] Un *crawler* es un robot informático que consulta de manera iterativa cierta información web disponible, con objeto de recopilar datos. Esto será definido más en detalle en la sección 2.3

1.3. Entorno socio-económico

La idea de este trabajo fin de grado es la creación de una aplicación útil para la sociedad. Con esta aplicación se puede beneficiar a las empresas a conocer mejor las características de sus competidores. También se puede utilizar para conocer la reacción de la sociedad ante un suceso importante.

Por ejemplo, suponiendo el hecho de que hay un corte en las líneas de teléfono móvil durante un día en toda España por un problema técnico. Una de las reacciones de los usuarios sería escribir en las páginas de Facebook para quejarse a sus compañías y esperar una solución acerca de lo sucedido. Con este proyecto se podría ver si realmente la sociedad expresa sus sentimientos en la redes sociales, qué compañía telefónica tiene mayor número de clientes, además con el análisis de los usuarios, se podría conocer el perfil de los usuarios más comunes que se han quejado o han comentado algo al respecto.

Los datos proporcionados por la aplicación, como consecuencia, pueden servir para aplicar tareas de marketing a raíz de los datos obtenidos. Por ejemplo, continuando con el ejemplo de las líneas telefónicas, identificar qué usuarios se quejan más de los servicios proporcionados con el fin de lanzar campañas de fidelización para los mismos, mejorando su calidad y el impacto de la empresa en cuestión.

En definitiva, esta aplicación hace un análisis de una de las redes sociales más importantes hoy en día. Puede llegar a tener un gran valor para las empresas ya que les permite tener *feedback* de sus clientes. Por tanto, este trabajo fin de grado podría llegar a ser un producto comercial. En un contexto socio-económico permitiría la creación de puestos de trabajo.

1.4. Estructura del documento

Tras el primer capítulo de Introducción, el contenido de la presente memoria se completa con 6 capítulos adicionales, los cuales se describen a continuación de forma resumida para facilitar su lectura.

- **Glosario**
- **Capítulo 1: Introducción** Breve introducción del proyecto en la que se expone la motivación del proyecto, los objetivos que se pretenden lograr, el entorno socio-económico que lo enmarca y la estructura de la memoria.
- **Capítulo 2: Estado del arte** Capítulo dedicado al planteamiento del problema, análisis del estado del arte presentando las tecnologías que se han aplicado para el diseño del sistema.
- **Capítulo 3: Descripción general del sistema** Visión general del sistema llevado a cabo. Se enumeran los requisitos de las capacidades, funcionalidades y restricciones de las tecnologías aplicadas. Además se plantean las diferentes alternativas y la solución elegida para resolver este proyecto. Por último se describe el entorno de desarrollo utilizado para desenvolver el sistema.
- **Capítulo 4: Funcionamiento del sistema** Explicación del funcionamiento del sistema, dividiéndolo en dos grandes bloques: Frontend, que conforma la aplicación web creada, y Backend, que desarrolla toda la parte técnica necesaria para este

proyecto.

- **Capítulo 5: Ejemplo de utilización de la aplicación** Presentación de un caso práctico para mostrar el funcionamiento de la aplicación.
- **Capítulo 6: Planificación del trabajo y presupuesto del proyecto** Planificación seguida para la realización del proyecto y el presupuesto económico necesario para el mismo.
- **Capítulo 7: Conclusiones** Conclusiones a las que se han llegado tras la finalización del trabajo y exposición de los posibles trabajos futuros que se podrían implementar para ampliar las funcionalidades de la aplicación.
- **Bibliografía**

Capítulo 2

Estado del arte

En este capítulo se expondrá la base teórica que sustenta este proyecto, explicando las tecnologías y técnicas aplicadas para su realización. Se explicará qué es Facebook y cómo utilizar su API mediante Facebook developers. También se explicará qué es un *crawler* y para qué son utilizados. Por otro lado, se introducirán los tipos de servidores y bases de datos utilizados necesarios para la realización del proyecto.

2.1. Facebook

Facebook se creó como una versión en línea de los *"facebooks"*, que son publicaciones que hacen las universidades americanas al inicio de cada curso académico, que contienen fotografías y nombres de todos los estudiantes y que tienen como objetivo ayudar a los estudiantes a conocerse mutuamente. Así nació Facebook en 2004. Creado por Mark Zuckerberg, un estudiante de Harvard en aquel momento, que abrió esta web como un hobby para dar servicio a los estudiantes de su universidad. Fundado junto a sus compañeros Eduardo Saverin, Chris Hughes y Dustin Moskovitz, que ayudaron a reforzar la programación, el negocio y el diseño.

El éxito de esta web se marcó desde sus inicios. En su primer mes de funcionamiento contaba con más de la mitad de los estudiantes de Harvard suscritos, poco después se expandió a las universidades e instituciones más prestigiosas de Estados Unidos. Un año después, Facebook tenía más de un millón de usuarios, una oficina en Palo Alto y había recibido financiación por parte de Peter Thiel, fundador de Pay-pal, y por parte de Accel Partners. Dada su gran acogida, ese mismo año incorporó a los alumnos de más de veinticinco mil escuelas secundarias y dos mil universidades, logrando un total de 11 millones de usuarios.

En el año 2006, Facebook se hizo público en Internet, dando acceso a todas las personas que tuvieran una cuenta de correo electrónico, y no sólo a los pertenecientes de ciertas universidades de Estados Unidos. De esta forma, Facebook se convirtió en una comunidad de comunidades, conectando estudiantes, empresas y gente que puede elegir sus redes de amistad en base a sus intereses. En definitiva, es una comunidad creada por y en función de sus miembros, lo que define el concepto de Web 2.0.

Ese mismo año, Facebook lanzó la Plataforma Facebook, un nueva herramienta que abría paso a los desarrolladores, permitiendo integrarse con la popular aplicación Facebook,

y también creando nuevas oportunidades de negocio. La idea era ampliar la utilización de Facebook integrando los productos y aplicaciones de otros con la red social de Facebook, dando acceso a los usuarios a utilizar esos productos desde Facebook.

A mediados de 2007 se lanzaron las versiones en francés, alemán y español para impulsar su expansión fuera de Estados Unidos, ya que sus usuarios se concentraban en Estados Unidos, Canadá y Reino Unido.

En 15 de diciembre de 2010 Mark Zuckerberg fue elegido como *"persona del año"* por la revista Time por *"haber conectado a más de 500 millones de personas en todo el mundo"*.

Los últimos datos obtenidos de Facebook en 2015 [1] indican que uno de cada cinco habitantes del mundo tiene una cuenta en la red social y la utiliza de manera activa. En concreto, el número de usuarios activos se sitúa en los 1390 millones en todo el mundo.

Una cifra que resulta más interesante para Facebook es el número de usuarios que utilizan la red diariamente. Unos 890 millones de usuarios revisan su perfil o las actualizaciones de sus contactos cada día, es decir, un 64 % de los usuarios. Este hecho se debe sin lugar a duda a la gran importancia que están adquiriendo los smartphones y *tablets* para esta red social.

En cuanto a su utilidad, Facebook ha estado y sigue estando orientado a las personas, pero, con el tiempo, también han ido adquiriendo un peso importante las páginas de Facebook. Una forma que las marcas y empresas han utilizado para atraer a los usuarios.

En resumen, Facebook es una de las redes sociales más populares a día de hoy, ya que promueve la interacción de las personas y la creación de nuevos proyectos para ser expandidos con la ayuda de sus millones de usuarios.

2.2. Facebook developers: Graph API

Uno de los grandes avances que tuvo Facebook fue el lanzamiento de la Plataforma de Facebook. Con esta herramienta los desarrolladores pueden crear aplicaciones integradas con Facebook.

La creación de aplicaciones se realiza mediante la Plataforma de Facebook. Una de las opciones de esta plataforma de interés para este proyecto son las peticiones a la API de Facebook.

El *Graph API* es una herramienta que te da acceso al API de Facebook, se pueden obtener datos dentro y fuera de la Plataforma de Facebook. Es una API basada en HTTP de bajo nivel que se puede utilizar para consultar datos, publicar nuevas historias, gestionar anuncios, subir fotos y gran variedad de tareas que una aplicación pueda necesitar.

Es un sistema muy intuitivo y fácil por el que Facebook pasa a trasmitir sus datos y por el que podemos enviárselos. La *Graph API* representa la información de Facebook de una forma organizada donde existen tipos de elementos con información que los compone (usuarios, posts, fotos, eventos, etc.) y conexiones entre todos ellos (amigos, feed, fotos y álbumes, etc.). Facebook adopta el modelo de datos Json y genera las peticiones mediante URLs muy simples.

Para poder realizar peticiones a la API de Facebook desde una aplicación externa, existen múltiples librerías que facilitan este acceso. En la caso de Java existe la librería

Facebook4j, es una librería Java de código abierto, no oficial, para acceder al *Graph API* de Facebook. Soporta todas las peticiones al *Graph API* de Facebook, como se puede ver en el siguiente enlace [9].

La API de Facebook permite acceder a todos los datos públicos de sus usuarios. Centrándonos en el objetivo de este proyecto, se van a realizar peticiones sobre las páginas de Facebook. En este contexto el *Graph API* facilita la obtención de los posts de una página con una petición muy sencilla, tal y como se muestra en la siguiente figura (2.1)

Figura 2.1: Opciones de una petición a la API de Facebook sobre los posts de una página

Donde {user-id} se corresponde con el id o el nombre de la página bajo estudio. Además proporciona tres opciones para la obtención de estos datos:

- **feed**
  La opción *feed* devuelve todos los posts de una página.
- **posts**
  El filtro *posts* muestra sólo los posts publicados por la página, estos posts se corresponden con los que se muestran en el muro de Facebook, sin ningún tipo de filtro.
- **tagged**
  La opción *tagged* devuelve sólo los posts en los que la página ha sido etiquetada. Estos posts se corresponden con la sección *Post to Page*, donde los propietarios de las páginas habilitan esta sección para almacenar los comentarios de los usuarios.

Este proyecto realizará peticiones a la API sobre las páginas de Facebook, en concreto utilizando la opción *tagged*. Con esta petición se obtendrán los datos principales de los comentarios de los usuarios en las páginas de Facebook que contengan la sección comentada.

Existen muchas aplicaciones basadas en Facebook, que obtienen los datos de los usuarios para su propia aplicación, pero no existe ninguna aplicación que recoja los datos de una página de *Facebook* de manera automática.

2.3. Crawler

Un *crawler* o "Web Spider" es un programa que inspecciona las páginas de la World Wide Web (WWW, conjunto de protocolos que permite, de forma sencilla, la consulta remota de archivos de hipertexto) de forma metódica y automatizada. Es un robot de la Web (*webbot*) muy utilizado hoy en día, se encarga de recorrer páginas Web de Internet, las descarga al ordenador local, las parsea y las procesa.

Su principal utilidad viene otorgada por los buscadores, que utilizan dicha herramienta para rastrear nuevas webs y descargar automáticamente una copia que almacenan en un índice; una vez integradas en estas bases de datos, las webs podrán ser rápidamente localizadas entre la lista de resultados en las siguientes consultas efectuadas por los usuarios.

Un *crawler* trata de simular las acciones humanas recorriendo un sitio web. A diferencia de una persona, un *crawler* ve el código fuente de la página web que está rastreando, no el diseño ni las imágenes o el contenido. Como desventaja de este tipo de robots es que el código fuente de las páginas cambia cada cierto tiempo, por lo que dicha herramienta requiere mantenerse actualizada en todo momento. Para mucha gente, los *crawlers* son sinónimo de motores de búsqueda; sin embargo, las posibilidades que un *crawler* ofrece van mucho más allá. Por ejemplo, se puede desarrollar un *crawler* que haga lo que cualquier otro *webbot* puede realizar, con la ventaja de que el alcance logrado sería toda la Red.

La posibilidad de crear *crawlers* específicos, abre un nuevo mercado para los desarrolladores Web. Un mercado en el que cobra gran relevancia el diseño de *crawlers* con tareas concretas, es por eso que cada vez más empresas desean incluir *crawlers* en su negocio, con el fin de recolectar información concreta de ciertas páginas web.

A continuación se detalla el origen de los primeros *crawlers* y su evolución a lo largo de los años:

- RBSE: En 1994, Eichmann [21] publicó el primer *crawler* existente, RBSE, basado en dos programas: el primer programa, "spider" que mantiene una cola en una base de datos relacional, y el segundo programa, llamado "mite", un navegador modificado de WWW ASCII que descarga las páginas de la Web.

- World Wide Web Worm: También en el año 1994, McBryan [25] creó el Recolector usado para construir un índice simple de los títulos y de URLs del documento. Dicho índice podía ser buscado usando el comando grep del Unix.

- Web*crawler*: En 1994 Brian Pinkerton [26] comenzó un Web *crawler* en su tiempo libre, fue utilizado para construir el primer índice con texto completo público disponible de un subconjunto del World Wide Web Worm. Se desarrolló basado en una librería para acceder a varios tipos de contenido con diferentes protocolos, incluyendo HTTP y FTP. También se basó en otro programa para analizar y pedir URLs para la exploración del grafo del Web. Además incluyó un recolector en tiempo real que sigue la ruta de navegación basándose en la semejanza del texto del vínculo con la pregunta proporcionada por el usuario.

- *crawler* de Google: En 1998, Brin y Page [20] crean Google *crawler*, un *crawler* que fue integrado con el proceso de la indexación de direcciones, porque el texto que analizaba fue hecho para la indexación de direcciones con texto completo y también

para la extracción de URLs. Se basa en un servidor donde se almacenan todas las URLs, a este servidor se le envían las listas de URLs que son recogidas por varios *crawlers*. Durante el análisis, las URLs encontradas son comprobadas con el servidor si ya han sido vistas previamente. Si no, la URL es agregada a la cola del servidor de las URLs.

- Mercator: Creado en 1999 por Heydon y Najork [23]. Es un Recolector modular escrito en Java. Su modularidad radica en el uso de los "módulos intercambiables de protocolo" y de "los módulos de procesamiento". Los módulos de los protocolos se asocian con cómo adquirir los Web pages (e.g.: por el HTTP), y los módulos de procesamiento se relacionan con cómo procesar Web pages. El módulo de proceso estándar sólo analiza las páginas y extrae URLs nuevo, pero otros módulos de proceso se pueden utilizar para poner en un índice el texto de las páginas o también recopilar estadística del Web.

- PolyBot: En 2002 creado por Shkapenyuk y Suel. Es un Recolector distribuido escrito en C++ y en Python, que se compone de un "encargado del crawling", uno o más "downloaders o descargadores" y uno o más "Encargados de resolver DNS". Las URLs reunidas se agregan a una cola en disco, y se procesan posteriormente para buscar las nuevas URLs.

- WebRACE: También en el año 2002, Zeinalipour-Yazti y Dikaiakos crearon un módulo de crawling implementado en Java usado como una parte de un sistema más genérico catalogado como eRACE. El sistema recibe peticiones de los usuarios para descargar Web pages, así que los actos del *crawler* en parte funcionan como proxy server inteligente. El sistema además maneja los pedidos "suscripciones" a los Web pages que deben ser supervisados (cuando las páginas cambian, deben ser descargadas por el *crawler* y el suscriptor debe ser notificado). La característica de mayor importancia de WebRACE es que, mientras que la mayoría de los *crawlers* comienzan con un sistema de la "semilla" URLs, WebRACE está recibiendo continuamente URLs nuevas para comenzar.

- *crawler* FAST: En 2002, Risvik y Michelsen crean el recolector usado por el Search Engine FAST, cuya arquitectura es una arquitectura distribuida en la cual cada máquina sostiene un "planificador del documento" que mantiene una cola de documentos para ser descargado por un "procesador del documento". Este los almacena en un subsistema de almacenaje local. Cada *crawler* mantiene comunicación con los otros *crawlers* por medio de un módulo distribuidor que intercambia la información del hipervínculo

- Ubicrawler: En 2004, Boldi [19] crea un *crawler* distribuido, programado en Java y con todas sus funciones descentralizadas. Cada CPU reporta 660 páginas por segundo.

- En el año 2010 se presenta un proyecto de *crawler* focalizado en base de datos de tópicos (DTB) [30]. Una de las propuestas del trabajo es que se cuente con una base de tópicos estáticos, y una base de tópicos dinámicos con auto-aprendizaje.

- En el año 2011, Anbukodi [18] propone el uso de "agentes móviles" para reducir la sobrecarga de máquinas de los *crawlers* actuales. La arquitectura sugiere que cada agente cuente con un conjunto de páginas iniciales (llamadas "semilla") y rastree los vínculos que contenga dicha página recursivamente; cuando un vínculo tenga una gran cantidad de niveles debajo de él, el agente tiene la capacidad de pasar dicho vínculo a un nuevo agente, liberando recursos y balanceando su carga.

2.3.1. Crawlers sociales

Existen varios estudios de Web *crawlers* enfocados al contenido de las redes sociales. Uno de ellos es el propuesto por S. Ibrahim en el año 2008 [24], aunque su objetivo era el desarrollo de una red social, propone utilizar agentes multicrawlers. Se centra en la extracción de información importante para los negocios.

En el año 2009 se publica un artículo sobre técnicas de minería Web para redes sociales [27], los autores clasifican tres técnicas diferentes según su uso: Minería de contenido web (se analiza el contenido de la web como textos y gráficos y se enfoca en el procesamiento de texto), minería de estructuras web (se centra en el análisis de la estructura de los sitios web a partir de los enlaces de los sitios) y minería de uso web (se basa en analizar cómo son utilizados los sitios web, es decir, analizar el comportamiento de los usuarios cuando visitan cada sitio).

Ese mismo año Fard [22], introduce el concepto de "minería colaborativa", en un proyecto para descubrir patrones de grupos criminales dentro de las redes sociales; utiliza un sistema multiagentes, con unas reglas de asociación basadas en el algoritmo "a priori" (el algoritmo a priori es uno de los 10 algoritmos más conocidos en la minería de datos según la IEEE hasta el 2007 [29], y consiste en la creación de clases candidatas basadas en agrupamiento dinámico).

2.3.2. Selenium

Selenium es un entorno de pruebas de software para aplicaciones basadas en la web. Nos aporta una herramienta con funcionalidad de grabar/reproducir para crear pruebas sin usar un lenguaje de cd programación para pruebas (Selenium IDE). También nos provee de un lenguaje específico de dominio para pruebas (Selanese) para escribir pruebas en un amplio número de lenguajes de programación populares como Java, C#, Ruby, Groovy, Perl, Php y Python. Las pruebas pueden ejecutarse usando la mayoría de los navegadores web modernos en diferentes sistemas operativos como Windows, Linux y OSX.

Selenium fue inicialmente desarrollado por Jason Huggins en 2004 pero pronto se unieron al esfuerzo otras personas con gran experiencia en pruebas y programación. Es un software de código abierto bajo la licencia apache 2.0 que puede ser descargada y usada sin cargo. El nombre es resultado de una broma de Huggins burlándose de un competidor llamado Mercury (mercurio), argumentando que el envenenamiento por mercurio puede ser curado tomando Selenio.

Existen otros proyectos derivados de Selenium como Selenium Grid para probar la concurrencia de múltiples pruebas concurrentes de clientes remotos o locales, así como Flash Selenium para probar programas escritos en Adobe Flex o Selenium Silverlight.

Selenium IDE
Selenium IDE es un entorno de desarrollo integrado para pruebas con Selenium, conocido en sus orígenes como Selenium Recorder. Está implementado como una extensión de Firefox y permite grabar, editar y depurar pruebas. Es posible desarrollar automáticamente scripts al crear una grabación, de esa manera se puede editar manualmente con sentencias y comandos para que la reproducción de nuestra grabación sea adecuado. Los scripts se generan en Selanese, un lenguaje de scripting para Selenium. Este provee comandos que ejecutan acciones sobre elementos en el navegador, como hacer click en un enlace, seleccionar de

una lista de opciones, verificar la presencia de un texto en particular o para tomar la totalidad de la página producto de las acciones.

Las características principales de este entorno son:

- Grabación y reproducción fácil.
- Selección inteligente de campos usando ID, nombre o XPath según se requiera.
- Autocompletado de los comandos de Selenium más comunes.
- Pruebas de revisión cruzada.
- Depuración y puntos de verificación (breakpoint).
- Almacenar las pruebas como Selanese, Ruby, Java y otros formatos.
- Opción para insertar el título de la página.
- Opción de modificarle a la medida con el uso de complementos

Selenium Client API
Selenium Client API es una interfaz de programación de aplicaciones (API) de clientes. Como alternativa a escribir pruebas en Selanese, estas pueden escribirse en varios lenguajes de programación que se comunican con Selenium mediante llamadas a los métodos de Selenium Client API. En la actualidad Selenium provee API para Java, C#, Ruby y Python.

Con Selenium 2 se presentó una nueva API de clientes, con WebDriver como componente central, aunque la anterior API puede seguirse usando llamando a la clase Selenium. **Selenium Remote Control**
Selenium Remote Control (RC) es un servidor escrito en Java que acepta comandos al navegador vía HTTP. RC hace posible escribir pruebas automatizadas para aplicaciones web en cualquier lenguaje de programación, lo que permite una mejor integración de Selenium a entornos de prueba existentes. Para hacer la escritura de pruebas más sencilla, Selenium actualmente provee controladores de dispositivos para PHP, Python, Ruby,.NET, Perl y Java. El controlador de Java puede usarse para Javascript vía el motor Rhino.

Selenium WebDriver
Selenium WebDriver es el continuador de Selenium RC. Selenium WebDriver acepta comandos (enviados en Selenese o vía el API de cliente) y los envía a un navegador. Esto se implementa a través de un controlador del navegador específico para cada navegador que envía los comandos y trae los resultados de regreso. La mayoría de los controladores de navegador lanzan y acceden a la aplicación de navegador (como Mozilla Firefox o Internet Explorer), pero también hay un controlador para HtmlUnit que simula un navegador.

Como diferencia de Selenium 1, en el que el servidor Selenium RC era indispensable, en Selenium WebDriver no se necesita un servidor especial para ejecutar las pruebas; en vez de ello WebDriver inicia una instancia del navegador y lo controla; sin embargo puede usarse Selenium Grid para ejecutar pruebas en sistemas remotos.

Selenium-WebDriver está completamente implementado y soportado en Java, Ruby, Python y C#. Esto significa que en la practica la API de Selenium 2.0 tiene significativamente menos llamadas que el API de Selenium 1.0. Mientras Selenium 1.0 intentaba proveer una interfaz rica en muchas operaciones, Selenium 2.0 intenta proveer de los bloques de construcción básicos para que los desarrolladores puedan programar su propio lenguaje específico de dominio. Uno de ellos ya existe, es el proyecto Watir en Ruby

que tiene una historia rica en buen diseño. Watir-WebDriver implementa el API de Watir como un envolvente del Selenium-Webdriver en Ruby y se crea de forma completamente automática, basado en las especificaciones del WebDriver y HTML.

Selenium Grid
Selenium Grid se trata de un servidor que permite usar instancias de navegador ejecutándose en máquinas remotas. Con él uno de los servidores actúa como concentrador. El proceso consiste en que las pruebas contactan al concentrador para obtener acceso a instancias de navegadores y el concentrador lleva una lista de instancias de los navegadores (Nodos de WebDriver) que permiten a las pruebas usar estas instancias.

Selenium Grid nos permite ejecutar pruebas en paralelo en múltiples máquinas y manejar diferentes versiones y configuraciones de manera centralizada.

En este Trabajo Fin de Grado se ha utilizado la herramienta Selenium WebDriver para desarrollar un *crawler*, o robot, que recorra las páginas de los usuarios de Facebook recogiendo la información de dichos usuarios, almacenándolo en la base de datos para su posterior procesado y análisis con fines estadísticos.

2.4. Colas de mensajes: RabbitMQ

Para entender qué es RabbitMQ, primero se va a explicar que son las colas de mensajes. Una cola de mensajes es un contenedor que alberga mensajes mientras están en tránsito. Los emisores producen mensajes y para que estos lleguen a su destinatario deben ser entregados a un intercambiador, que los colocará en la cola del respectivo destinatario y dicho destinatario puede ir progresivamente desencolando y procesando los mensajes, o dejar que el intercambiador se los haga llegar, mediante distintos tipos de rutas. Las colas de mensajes son un intermediario entre los emisores y los destinatarios (entre clientes y servidores o publicadores y consumidores). Algunas de las ventajas de este tipo de intercambio de información son las siguientes:

- **Redundancia**
  Si la aplicación falla mientras se está procesando alguna petición, no se perderá la información, ya que el sistema de cola permite a la aplicación prescindir de esa información hasta que no sea procesada correctamente.
- **Naturaleza asíncrona**
  Dependiendo del funcionamiento del sistema, se puede querer acumular los mensajes en lotes. La cola almacenará los mensajes según como se haya programado y cómo vaya procesando
- **Garantía de entrega y ordenamiento**
  Se garantiza que el orden en el que llegan los mensajes será el orden en el que sean procesados, de igual manera un emisor puede estar seguro de que éste ha sido recibido y se procesará una única vez.
- **Disponibilidad**
  Si existen fallos, los mensajes no se perderán. Permanecerán en la cola hasta que se les indique lo contrario y al mismo tiempo la cola podrá seguir recibiendo mensajes para el procesamiento posterior a la recuperación del sistema.
- **Elasticidad**
  En situaciones donde tu sistema pudiera llegar al tope de su capacidad de recepción de

peticiones y volverse incapaz de responder por un anormal flujo de mensajes, el hecho de tener una cola o buffer de peticiones permitirá balancear, filtrar y normalizar el flujo, lo cual evitará que el sistema sea inundado de peticiones que no pueda responder causando perdida de los mismos o incluso el colapso del sistema.

- **Desacoplamiento**
  El hecho de tener una capa intermedia de comunicación entre procesos da la flexibilidad en la definición de arquitectura de cada uno de ellos de manera separada. Mientras cada uno se mantenga alineado a los mismos requerimientos de interfaz de la cola de mensajes, no existirán mayores problemas de compatibilidad.
- **Escalabilidad**
  El sistema es sencillo de escalar, agregando más unidades de procesamiento. De esta forma el sistema de colas se encargará de balancear la carga entre ellos.

RabbitMQ [28] está basado en el estándar AMQP (*Advanced Message Queuing Protocol*), y ofrece una arquitectura ligera de mensajería. El estándar AMQP es un protocolo de la capa de aplicaciones de un sistema de comunicación. Se caracteriza por la orientación a mensajes, encolamiento, enrutamiento, tanto punto-a-punto como publicación-subscripción, exactitud y seguridad. AMQP estipula el comportamiento tanto del servidor que provee los mensajes, como del cliente de la mensajería hasta el punto de que las implementaciones de diferentes proveedores son verdaderamente interoperables. AMQP es un protocolo a nivel de cable. Es decir, cualquier programa que cree e interprete mensajes con este formato de datos, puede interoperar con otra herramienta que cumpla con este protocolo, independientemente del lenguaje de implementación.

RabbitMQ ofrece las ventajas antes mencionadas de las colas de mensajes además de otras características:

- Soporta múltiples protocolos, muchas soluciones existentes sólo manejan uno, como AMQP.
- Dispone de librerías en casi todos los lenguajes de programación.
- Rastreo de mal comportamiento en las colas.
- Clusterización de varios servidores de colas para evitar que el sistema de encolamiento falle.
- Múltiples tipos de enrutamiento, incluso puedes crear tu propia ruta.
- Soporta plugins para extender su comportamiento
- Amigable interfaz gráfica.

Existen algunas alternativas a RabbitMQ muy interesantes en el ámbito cloud oSaaS (software como servicio) como IronMQ y Amazon SQS pero no son de código abierto como lo es RabbitMQ, siendo una de las soluciones más robustas de manejo de colas de mensajes.

Si hay un flujo de mensajes muy alto, hay que levantar más consumidores y RabbitMQ se encargará de balancear la carga entre ellos, siempre manteniendo un orden lógico en los mensajes y evitando que estos se pierdan o sean procesados más de una vez. Gracias a esto se pueden soportar altos volúmenes de peticiones.

RabbitMQ es de naturaleza asíncrona y ya se han comentado las ventajas que ofrece. El enfoque productivo más común es aquel que se basa en suscripciones a contenidos mejor conocido como el modelo Pub/Sub. éste por medio de un intercambiador distribuirá el

mensaje para ser consumido por múltiples consumidores (como los feeds). Otro enfoque importante es aquel basado en un procedimiento que se compone de varias tareas pesadas, las cuales suelen ser síncronas, lo cual obliga al usuario a esperar a que una tarea termina para proseguir. Tareas como la gestión de imágenes y programación de tareas prolongadas.

2.5. Bases de datos

Una base de datos es un banco de información que contiene datos relativos a diversas temáticas y categorizados de distinta manera, pero que comparten entre sí algún tipo de vínculo o relación que busca ordenarlos y clasificarlos en conjunto.

Actualmente se dividen en dos grandes tipos, las tradicionales, que son bases de datos SQL de tipo relacional, como son MySQL o PostgresSQL, y las que reciben el nombre de NoSQL (Not only SQL) y que han entrado en el mercado hace relativamente poco, con la intención de hacer frente a las bases de datos relacionales utilizadas por la mayoría de los usuarios.

Una base de datos relacional se caracteriza por la definición de tablas estáticas en las que se almacena toda la información. Estas tablas necesitan una llave única para identificarlas, denominada "Primary key". Además entre las distintas tablas pertenecientes a una misma base de datos existen relaciones de diferentes tipos:

- Relación de uno a varios (1,n). Se crea una relación de uno a varios si uno de los campos relacionados es una clave principal. Esta relación es la más común. Cada registro de una tabla puede estar enlazado con varios registros de una segunda tabla, pero cada registro de la segunda sólo puede estar enlazado con un único registro de la primera.

- Relación de uno a uno (1,1). Se creará una relación de este tipo si ambos campos relacionados son claves principales. En este tipo de relación, un registro de la tabla uno sólo puede estar relacionado con un único registro de la tabla dos y viceversa. No es muy usada.

- Relación de varios a varios (n,m). En este caso, ninguno de los campos relacionados son claves principales. Cada registro de la primera tabla puede estar enlazado con varios registros de la segunda y viceversa. Este tipo de relación implica la repetición de los campos de cada tabla; esto es lo que Access pretende evitar. Para establecer relaciones de este tipo, es necesario crear una tabla intermedia que esté relacionada con las dos de uno a varios.

Aunque todas se parecen en cuanto a definición, existen distintos tipos de bases de datos relacionales. A continuación se definen las más comunes:

TIPO	DESCRIPCIÓN
MySQL	Base de datos más usada y mejor documentada hasta ahora. Se considera la más rápida de las bases de datos relacionales, aunque no es recomendable para grandes volúmenes de datos.
Microsoft SQL Server	Desarrollada por Microsoft. Suele ser utilizada por desarrolladores de la plataforma ASP.NET
PostgresSQL	Alternativa a MySQL, se parecen en cuanto a escalabilidad y versatilidad y destaca por la robustez que ofrece.
SQLite	Es una biblioteca muy ligera. Implementa un sistema autónomo sin servidor ni apenas configuración.
Microsoft Acces	Desarrollada también por Microsoft para uso personal o para pequeños proyectos y organizaciones.

Tabla 2.1: Tipos de bases de datos relacionales

En cuanto a las bases de datos no relacionales, han abierto paso a un nuevo concepto de bases de datos. A continuación se detallan sus principales características:

- Se ejecutan en máquinas con pocos recursos: Estos sistemas, a diferencia de los sistemas basados en SQL, no requieren de apenas computación, por lo que se pueden montar en máquinas de un coste más reducido.

- Escalabilidad horizontal: Para mejorar el rendimiento de estos sistemas simplemente se consigue añadiendo más nodos, con la única operación de indicar al sistema cuáles son los nodos que están disponibles.

- Pueden manejar gran cantidad de datos: Esto es debido a que utiliza una estructura distribuida, en muchos casos mediante tablas Hash.

- No genera cuellos de botella: El principal problema de los sistemas SQL es que necesitan transcribir cada sentencia para poder ser ejecutada, y cada sentencia compleja requiere además de un nivel de ejecución aún más complejo, lo que constituye un punto de entrada en común, que ante muchas peticiones puede ralentizar el sistema.

- No utilizan estructuras fijas como tablas para el almacenamiento de los datos. Permiten hacer uso de otros tipos de modelos de almacenamiento de información como sistemas de clave-valor, objetos o grafos.

En la siguiente tabla se describen brevemente las bases de datos no relacionales más conocidas.

TIPO	DESCRIPCIÓN
MongoDB	Base de datos orientada a documentos, de esquema libre. Es bastante rápido a la hora de ejecutar sus operaciones. Utiliza un sistema propio de almacenamiento llamado BSON, conocido como la evolución de JSON.
Cassandra	Creada por Apache. Dispone de un lenguaje propio para realizar consultas CQL (Cassandra Query Language).
Redis	Se trata de una base de datos tipo clave-valor. No permite realizar consultas, sólo se puede insertar y obtener datos.
CouchDB	Tambien creado por Apache. Uso Restfull HTTP API como interfaz y JavaScript como principal lenguaje de interacción.

Tabla 2.2: Tipos de bases de datos no relacionales

2.6. Servidores

Un servidor es un ordenador, con sus programas, que está al servicio de otros ordenadores. El servidor atiende y responde a las peticiones que le hacen los otros ordenadores llamados "clientes".

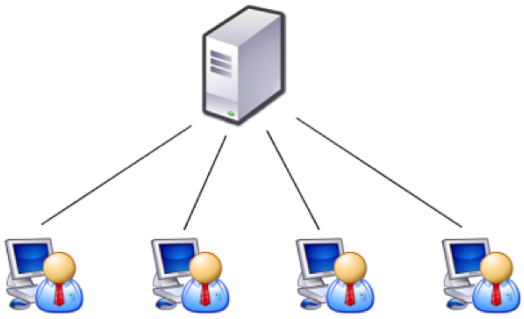

Figura 2.2: Ejemplo funcionamiento básico de un servidor

El modelo o arquitectura que siguen los servidores es el de cliente-servidor, es decir el cliente/s pide y el servidor proporciona los recursos o servicios. La red más conocida y más grande es Internet, y está llena de servidores.

Un servidor deberá estar siempre encendido, ya que si se apaga dejará de dar servicio a los demás. Cuando un servidor falla (se apaga o tiene errores) hace que los demás usuarios de la red tengan problemas, porque no disponen de los servicios que proporciona ese servidor.

Dependiendo del servicio que de el servidor, tiene que disponer de:

- Software: programas específicos capaces de ofrecer esos servicios.
- Hardware: ordenador que da respuesta a las peticiones lo más rápido posible. En la siguiente imagen vemos el apilamiento de los servidores de una empresa que se dedica

a proporcionar almacenamiento de información.

Algunos tipos de servidores que existen son:

1. **Servidores de archivos**
   Estos servidores son los encargados de almacenar distintas clases de archivos para después enviárselas a otros clientes en la red.
2. **Servidores de correo**
   Son los que hacen todas las operaciones relacionadas con e-mails para los clientes de la red: enviar, almacenar, recibir, enrutar, etcétera.
3. **Servidor de impresión**
   Estos controlan una o varias impresoras y son los que se encargan de poner en cola de impresión aquello que solicitan los clientes de la red. Por medio de este servidor se puede trabajar con la impresora como si esta estuviese directamente conectada a la computadora.
4. **Servidor de base de datos**
   Proveen servicios de base de datos a otros programas u ordenadores, aunque también pude hacerse referencia a aquellas computadoras (servidores) dedicadas a ejecutar esos programas, prestando el servicio.
5. **Servidor web**
   Este servidor provee de contenidos estáticos a los navegadores. Este le envía los archivos que carga por medio de la red al navegador del usuario. Los archivos pueden ser imágenes, escrituras, documentos HTML y cualquier otro material web.

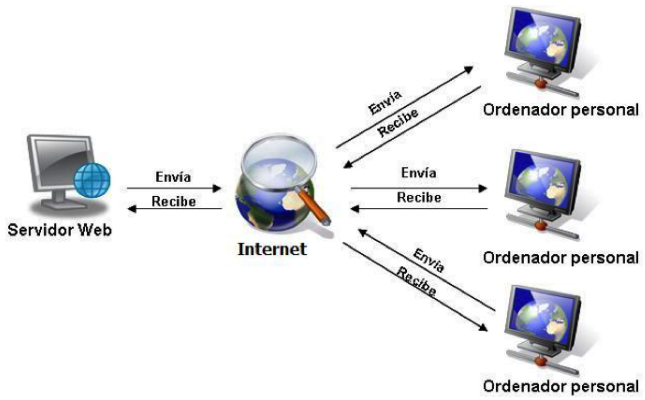

Figura 2.3: Ejemplo funcionamiento de un servidor web

6. **Servidor de fax**
   Estos servidores realizan todas las actividades necesarias para que los faxes sean transmitidos, recibidos y distribuidos. Aquí se incluyen las tareas de envío, almacenamiento y recepción, entre otras.
7. **Servidor del acceso remoto (RAS)**
   Permiten la administración del acceso a internet en una determinada red. De esta forma, se puede negar el acceso a ciertos sitios web. Por otro lado, ofrece servicios de seguridad y controla las líneas de módem de los canales de comunicación de las redes para que las peticiones sean conectadas con las redes cuya posición es remota.
8. **Servidores de telefonía**
   Realizan funciones obviamente relacionadas con la telefonía como es la de contestador

automático, las funciones para la respuesta de la voz, almacenan los mensajes de voz, encaminan llamadas y controlan también la red o el Internet.

9. **Servidor proxy** Se encarga de ciertas funciones a nombre de otros clientes de una red para aumentar el funcionamiento de ciertas operaciones, también proporciona seguridad (incluye firewall) y permite administrar el acceso a Internet en una red de ordenadores, permitiendo o denegando el acceso a ciertos sitios web.
10. **Servidores de uso**
Realizan la parte lógica del negocio de un uso del cliente, realizando operaciones ordenadas por un sitio de trabajo y dando los resultados a éste sitio. El sitio de trabajo realiza la lógica de la presentación para trabajar correctamente.
11. **Servidor de reserva**
Tiene un software de reserva de la red instalado y tiene cantidades grandes de almacenamiento de la red en discos duros u otras formas de almacenamiento para asegurar que la pérdida de un servidor principal no afecte a la red.
12. **Servidor de autenticación**
Son aquellos que verifican que un cliente se pueda conectar a la red desde cualquier punto en el que se encuentre (sea por cable o inalámbrico).

Para este proyecto se utilizarán varios tipos de servidores, uno de ellos, motor principal de este trabajo, un servidor web, que es imprescindible que soporte HTTP Post, requisito indispensable impuesto por *Facebook*.

Por otro lado, también se necesitará un servidor de correo para enviar notificaciones a los usuarios. En el caso de este trabajo fin de grado utilizara el servidor de correo de la universidad.

El resto de funcionalidades necesarias por parte de los servidores, se manejarán desde un servicio de *cloud computing*[1] llamado OpenStack, se explica a continuación en la sección (2.6.1).

2.6.1. OpenStack

Una infraestructura de *cloud computing* [14] permite al proveedor de contenidos o servicios en la nube privarse de instalar cualquier tipo de *hardware*, puesto que éste es provisto por el proveedor de la infraestructura. El gran beneficio del *cloud computing* es la simplicidad que ofrece. Requiere mucho menor tiempo a la hora de empezar a trabajar con el servicio, puesto que ya está equipado con el *hardware* necesario.

En referencia al *cloud computing* se definen tres capas, las cuales son:

- **Software as a Service (SaaS)**
Aplicación completa ofrecida como servicio en la nube (Servicios de Google, Salesforce.com, Microsoft Office 365, etc.)
- **Platform as a Service (PaaS)**
Aplicación completa para el desarrollo ofrecida como servicio en la nube (Google App Engine, Windows Azure, RedHat OpenShift, etc)

[1] *Cloud computing*, en español, Computación en la nube. Es un sistema informático que permite ofrecer servicios de computación a través de una red, que usualmente es Internet.

- **Infrastructure as a Service (IaaS)**
  Almacenamiento (también denominado Storage as a Service) y capacidades de cómputo (máquinas completas) ofrecida como servicio en la nube.

Además puede ser de tres tipos:

- **Público**
  Una empresa ofrece IaaS a terceros, encargándose de toda la gestión del *Cloud*. El caso más conocido es Amazon Elastic Compute Cloud.
- **Privado**
  Una organización configura sus propios recursos como IaaS para tener más flexibilidad y control total sobre sus recursos.
- **Híbrido**
  Algunos servicios se gestionan en el *Cloud* privado y otros se transfieren a uno público, normalmente utilizan una API común que permita una buena integración.

OpenStack es un proyecto de infraestructura de código abierto como servicio (IaaS) para la creación y gestión de grandes grupos de servidores privados virtuales en un centro de datos.

Los objetivos de esta iniciativa son soportar la interoperabilidad entre los servicios en la nube y que permita a las empresas construir servicios en la nube en sus propios centros de datos.

OpenStack tiene una arquitectura modular formada por varios componentes, a continuación se describen los principales:

- **OpenStack Nova**
  Nova conforma el componente principal de OpenStack. Es el software que controla la plataforma IaaS. Proporciona Máquinas Virtuales (VM) bajo demanda.
  Una instancia es una VM aprovisionada por OpenStack. Se permite seleccionar que tipo de instancia se quiere lanzar, eligiendo la interfaz gráfica, el tamaño de la memoria ram y del disco duro de almacenamiento. Además en conceptos de Red cada VM tiene una IP fija, que se asigna al crear la instancia, y además se cuenta con un servicio de IP flotantes, que se asocian dinámicamente a una instancia.
- **OpenStack Glance**
  Este componente ofrece un catálogo y repositorio de imágenes de disco virtuales. Es una simple API REST que contiene los metadatos de las imágenes.
- **OpenStack Swift**
  Proporciona un sistema de almacenamiento escalable que soporta almacenamiento de objetos.
- **Keystone**
  Suministra la autenticación y autorización de todos los servicios de OpenStack.
- **OpenStack Horizon**
  Este componente, facilita la utilización de OpenStack a los usuarios. Proporciona una interfaz gráfica basada en la web de usuario para los servicios de OpenStack. Implementa las funcionalidades básicas de los componentes principales de OpenStack: Nova, Glance, Swift, etcétera.

OpenStack es una herramienta muy útil para este proyecto, puesto que se necesitan VM para lanzar el *crawler* en servidores con interfaz gráfica con gran capacidad. Además en este servicio se han montado las bases de datos utilizadas para tener acceso desde la aplicación web en local y desde el *crawler* en las diferentes instancias.

2.7. Marco regulador: Leyes de Protección de datos

Dado que en este trabajo de fin de grado se van a manejar datos de usuarios que serán cedidos por *Facebook*. En este apartado, se describirá la principal ley de protección de datos española que actualmente se encuentran en vigor.

La privacidad en España está regulada por la Ley Orgánica 15/1999, de 13 de Diciembre, de Protección de Datos de Carácter Personal (LOPD). Según se cita en el BOE-A-1999-23750 [3], "la presente Ley Orgánica tiene por objeto garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar".

A continuación se citan los apartados de los artículos de interés para este trabajo fin de grado definidos en la LOPD:

- **Artículo 4. Calidad de los datos**

  1. Los datos de carácter personal sólo se podrán recoger para su tratamiento, así como someterlos a dicho tratamiento, cuando sean adecuados, pertinentes y no excesivos en relación con el ámbito y las finalidades determinadas, explícitas y legítimas para las que se hayan obtenido.
  2. Los datos de carácter personal objeto de tratamiento no podrán usarse para finalidades incompatibles con aquellas para las que los datos hubieran sido recogidos. No se considerará incompatible el tratamiento posterior de éstos con fines históricos, estadísticos o científicos.

- **Artículo 5. Derecho de información en la recogida de datos**

  1. Los interesados a los que se soliciten datos personales deberán ser previamente informados de modo expreso, preciso e inequívoco.

- **Artículo 6. Consentimiento del afectado**

  1. El tratamiento de los datos de carácter personal requerirá el consentimiento inequívoco del afectado, salvo que la ley disponga otra cosa.

En este proyecto se recogerán datos de los usuarios y páginas pertenecientes a la comunidad de *Facebook*. Estos usuarios han aceptado la política de privacidad de Facebook para hacer sus datos públicos, todos los datos que hagan públicos en su cuenta, serán de libre acceso para todos los internautas. Además, los datos obtenidos sólo se utilizan para el tratamiento de los mismos con fines estadísticos.

Capítulo 3

Descripción general del sistema

En este capítulo se van a describir las características principales de la aplicación, las herramientas utilizadas para su desarrollo y los requisitos necesarios para poder llevar a cabo las funcionalidades descritas en la sección 3.4.2.

3.1. Presentación

El objetivo principal de este trabajo fin de grado es la creación de una aplicación web que sea capaz de analizar un conjunto de páginas de *Facebook* de manera automática.

La figura (3.1) muestra un esquema general de la aplicación:

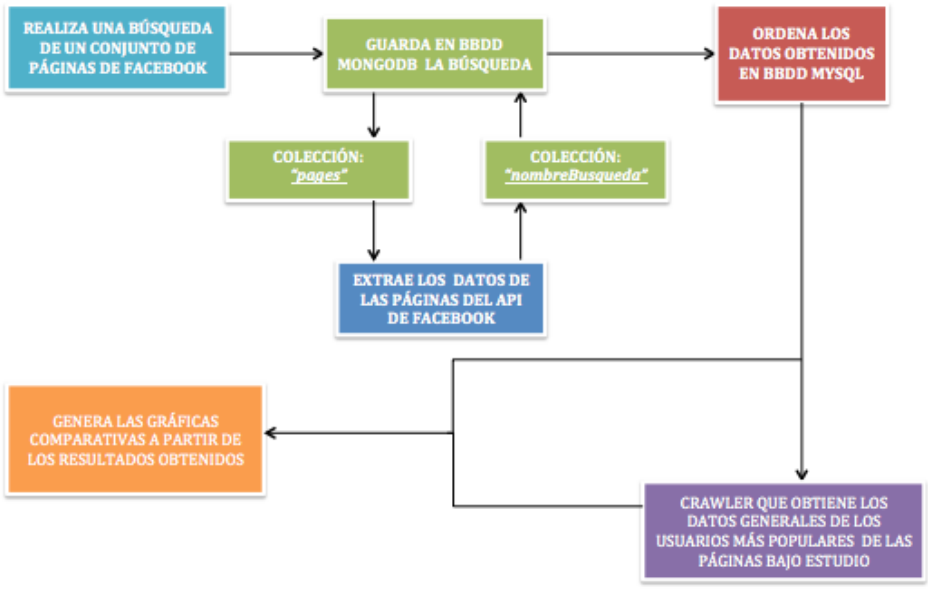

Figura 3.1: Esquema general del proyecto

Siguiendo este esquema, primero el usuario debe indicar el nombre del estudio que va a realizar, el nombre de las páginas y el periodo de tiempo que lo engloban, y una dirección de correo electrónico para enviarle una notificación cuando la recopilación de datos haya

finalizado. También debe seleccionar si desea que el estudio se haga sólo de las páginas de *Facebook* o también de los usuarios más comunes. Una vez realizada la petición por parte del usuario, la aplicación web internamente se encarga de la recopilación de datos.

Para la recolección de los datos, primero guarda la búsqueda del usuario en una base de datos en MongoDB, en una colección llamada "pages".

Acto seguido, realiza una petición al API de *Facebook*, solicitando los datos contenidos en la sección *Post to Page* de cada una de las páginas dentro del periodo de tiempo solicitado por el usuario. Esta petición devuelve los datos que son automáticamente guardados en la base de datos de MongoDB, en una colección creada con el mismo nombre que ha indicado el usuario para su estudio.

A continuación, se procesan los datos en una base de datos relacional, MySQL, que permite estructurar los datos obtenidos de la Plataforma de *Facebook*, guardando sólo aquellos campos que nos interesan y ordenándolos para facilitar su posterior procesado.

Una vez finalizado este paso, si el usuario ha seleccionado la opción del estudio de usuarios, por un lado se procede a la extracción de sus datos y por otro lado, se manda un primer informe al usuario de la aplicación con las gráficas de los datos obtenidos de *Facebook*. Proceso que se repite cuando la recopilación de datos de los usuarios haya finalizado.

Se trata de una aplicación clara y sencilla en la que los usuarios puedan comparar el análisis de diferentes páginas de *Facebook* dentro de su sesión, llevando un historial que almacena todas las peticiones realizadas.

3.2. Requisitos necesarios

En esta sección se van a definir los requisitos necesarios para el desarrollo técnico de la aplicación propuesta. También se definen los requisitos y restricciones de las tecnologías utilizadas.

3.2.1. Requisitos de la aplicación

A continuación se enumeran todos los requisitos y necesidades que se considera que debe realizar la herramienta a desarrollar, que afectan sólo al portal web.

1. **Accesible desde distintos navegadores**
   Los navegadores más comunes, "Mozilla Firefox", "Google Chrome", "Safari", "Internet Explorer", podrán utilizar la aplicación.
2. **Idioma**
   El idioma de la aplicación será el español.
3. **Registro de usuarios**
   El usuario deberá registrarse previamente para usar la aplicación.
4. **Inicio de sesión**
   El usuario deberá haber iniciado sesión con una cuenta registrada para acceder a la aplicación.

5. **Nueva búsqueda: Nombre estudio**
   El usuario deberá indicar el nombre del estudio que quiere realizar.
6. **Nueva búsqueda: Nombre páginas de Facebook**
   El usuario deberá indicar el nombre de las páginas que quiere analizar, asegurándose primero de que estas páginas contengan la sección *Post to Page*.
7. **Nueva búsqueda: Periodo del estudio**
   El usuario deberá indicar la fecha de inicio del estudio, es decir, desde cuando quiere que sea el primer comentario recogido en el estudio hasta la fecha actual.
8. **Nueva búsqueda: Correo Electrónico**
   El usuario deberá indicar una cuenta de correo electrónico donde se le enviará una notificación cuando la recopilación de datos haya acabado.
9. **Nueva búsqueda: Opción informe de las páginas**
   Si el usuario selecciona esta opción se procederá a la recolección de datos de las páginas indicadas obteniéndolos del API de *Facebook*, y se representarán las gráficas comparativas con los resultados obtenidos.
10. **Nueva búsqueda: Opción informe de los usuarios**
    Si el usuario selecciona esta opción se procederá a la extracción de datos de los usuarios que más comentan en dicha sección de *Facebook*. Se obtendrán los datos de los usuarios de su fecha y lugar de nacimiento, edad y ocupación profesional mediante un *crawler*.
11. **Historial de búsquedas**
    Los usuarios tendrán acceso a todas las búsquedas que hayan realizado en la aplicación, se mostrarán en una tabla.
12. **Ver análisis**
    Los usuarios dentro del historial de búsqueda podrán acceder a ver el análisis de cada uno de los estudios, se representarán las gráficas más representativas de los datos obtenidos.
13. **Cerrar sesión**
    Los usuarios podrán cerrar sesión cuando quieran, para mantener seguros sus estudios.
14. **Web responsiva**
    La interfaz web deberá adaptarse a cualquier tamaño de pantalla.
15. **Lenguajes de programación**
    Los lenguajes de programación para el desarrollo de la aplicación serán: Java, JSP, HTML, CSS, XML, JavaScript y jQuery.

3.2.2. Requisitos de Facebook

En este epígrafe se describirán los requisitos que Facebook impone para el correcto desarrollo de la aplicación que se pretende seguir.

1. **Soporte HTTP**
   El servidor donde se aloje debe soportar peticiones HTTP Get y Post.
2. **Cuenta en Facebook**
   Para poder acceder a *Facebook*, se debe tener creada una cuenta en dicha red social.
3. **Cuenta de desarrollador**

Para poder acceder a la Plataforma de *Facebook* y a su API, se debe crear crear una cuenta de desarrollador, esto es, aceptar los permisos y condiciones que indican desde una cuenta de *Facebook* de usuario.

4. **Crear una aplicación en Facebook**
   Para poder hacer peticiones a la API se necesita crear una aplicación dentro de la Plataforma, que proporcionará las credenciales necesarias para dichas peticiones.

3.2.3. Requisitos del crawler

En este apartado se definirán las condiciones necesarias para poder ejecutar el *crawler* en el entorno adecuado.

1. **Máquinas virtuales con interfaz gráfica**
   Para poder simular la acción de un usuario normal de *Facebook* que recopile los datos de los usuarios a los que visita, se necesitará acceso a máquinas virtuales con entorno gráfico, donde se ejecutará el *crawler* que lanzará instancias de un navegador para simular esta acción.
2. **Java**
   El lenguaje de programación será Java, por lo que habrá que instalar el entorno de Java en la máquinas que se vaya a ejecutar.
3. **Número de identificación de los usuarios de *Facebook***
   Se necesitarán los *ids* de aquellos usuarios que se vayan a analizar mediante el *crawler*.
4. **Distribución del crawler**
   Se utilizará un software de distribución de paquetes, RabbitMQ, para optimizar el tiempo de ejecución del *crawler*. Desde la aplicación se mandará a cada máquina virtual, mediante este mecanismo, el trabajo que debe realizar cada una.

3.2.4. Requisitos de desarrollo

Por último se expondrán los requisitos necesarios para la elaboración de este proyecto.

1. **Espacio para el trabajo**
   Se dispondrá de una espacio lo suficientemente amplio para poder albergar el equipo de trabajo.
2. **Ordenador**
   Se necesitará un ordenador para el desarrollo del trabajo fin de grado.
3. **Conexión a internet**
   Se necesitará conexión a Internet para la completa realización del proyecto.
4. **Servidor**
   Se necesitará un servidor donde alojar las máquinas virtuales.
5. **Servidor web y bases de datos** Se requerirá un servidor web donde alojar la aplicación web y deberá soportar el almacenamiento de los datos en bases de datos.

3.3. Alternativas de implementación del sistema

3.3.1. Alternativas del lenguaje

Una vez planteadas las tecnologías a desarrollar y los requisitos necesarios, se tuvieron en cuenta las posibilidades de los lenguajes de programación para aplicar al proyecto. Para el desarrollo de una aplicación web se utilizan tres lenguajes indispensables, HTML, CSS y JavaScript, para el diseño de las vistas. Sin embargo, para integrar la aplicación con la API de *Facebook* y para el manejo de las bases de datos se pueden utilizar varios lenguajes como Java, Python o Ruby, que disponen de librerías para facilitar el acceso a estas tecnologías. También se analizaron las posibilidades para el desarrollo del "*crawler*" basado en la herramienta de pruebas de Selenium-WebDriver. Esta herramienta está completamente implementada y soportada en Java, Ruby, Python y C#.

3.3.2. Evaluación de las alternativas de lenguaje y elección de la solución

Como las alternativas disponibles para el desarrollo de las tecnologías que conforman el núcleo de este trabajo fin de grado se podían desarrollar indistintamente en varios lenguajes, finalmente se decidió el lenguaje de programación Java. Una de las ventajas de decidir Java es que es programación orientada a objetos, simplifica mucho el código y el desarrollo definiendo los objetos necesarios y reutilizándolos. Otra de las ventajas es que es multiplataforma, funciona en todos los entornos disponibles. Es completamente gratis y no depende de ninguna licencia.

3.3.3. Alternativas de las bases de datos

A la hora de decidir el tipo de bases de datos a utilizar, se tuvo que tener en cuenta el motor principal de este trabajo fin de grado, que es la API de Facebook. Las consultas que se realizan a Facebook, la API las devuelve en formato JSON, y con una estructura variable, dependiendo de los datos que proporciona cada página. Por este motivo, se estudió la posibilidad de usar una base de datos NoSQL, que permitiera almacenar los datos sin un esquema predefinido. Existen varias posibilidades de bases de datos no relacionales como son Cassandra, Redis, MongoDB y CouchDB. Pero por otro lado, de cara al análisis de datos que esta aplicación recogerá, se consideró que sería conveniente tener una base de datos organizada, almacenando los datos de interés para el análisis. Varias de las alternativas planteadas fueron: MySQL, Access, PostgreesSQL, Microsoft SQL.

3.3.4. Evaluación de las alternativas de bases de datos y elección de la solución

Dentro de los tipos de bases de datos relacionales, se eligió una base de datos en MySQL, porque aunque todos los tipos cumplían los requisitos necesarios para la implementación de este trabajo fin de grado, MySQL se caracteriza por la rapidez de acceso a la información, punto interesante de cara a cargar los datos en una página web. Además el volumen de datos almacenado no será muy grande, otro aspecto por los que se eligió MySQL.

En cuanto a las opciones de bases de datos no relacionales la elección fue más fácil. Se

eligió MongoDB porque se pueden almacenar los datos BSON directamente, que es una evolución de JSON, esto facilita las peticiones realizadas a la API de Facebook, pudiendo almacenar el JSON que devuelve, sin tener que comprobar si están todos los campos, o si añade uno nuevo. Además de que es un bastante rápido a la hora de ejecutar las operaciones.

3.3.5. Alternativas del diseño web

El Frontend que conformará la interfaz de usuario, para esta parte se utilizará HTML, CSS para definir los estilos y JavaScript y jQuery para añadir efectos y funcionalidades a la página. Por otro lado, para el Backend, se estudiaron las distintas posibilidades para generar las páginas de forma dinámica. En esta parte, hubo que decidir el lenguaje de programación adecuado. Estos lenguajes, en el lado del servidor, buscarán la información en una base de datos para mostrarla en la interfaz. Existen varias posibilidades como son PHP, Python, Ruby, ASP.NET o Java.

3.3.6. Evaluación de las alternativas de diseño web y elección de la solución

El diseño web fue el último punto de este proyecto que se decidió, debido a que la importancia del trabajo era recopilar los datos de Facebook. Por tanto, una vez definido como lenguaje de programación Java, y teniendo tanto el *crawler* como el programa que accede a la API de Facebook en Java, se estudiaron las opciones disponibles de hacer una aplicación que implementara sin ningún tipo de problema los scripts comentados. Por esta misma razón se decidió realizar la aplicación web en la plataforma Java, utilizando Spring, que es un framework para el desarrollo de aplicaciones para la plataforma Java. Concretamente. Spring-MVC, módulo que implementa la arquitectura de Modelo-Vista-Controlador.

Destacar que el diseño de la aplicación se ha llevado acabo utilizando dos plantillas de Bootstrap. Bootstrap es un *framework* CSS, que permite dar forma a un sitio web utilizando librerías CSS ya definidas. Una de las plantillas se ha utilizado para los formularios de Inicio de Sesión y de Registro, esta plantilla se llama "Sign In Bootstrap Template" y se puede encontrar en el enlace [13]. Estas plantillas se han modificado acorde a la estética de la aplicación. La otra plantilla se ha utilizado para el formulario de una nueva búsqueda, y ha definido el estilo que sigue la aplicación, se llama "Multi Step Registration Form" y se puede encontrar en el enlace [12].

3.4. Entorno de desarrollo utilizado

3.4.1. Eclipse IDE for Java EE Developers

El programa utilizado para implementar el código de la aplicación es Eclipse [7]. Es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma.

Para desarrollar este proyecto se han utilizado dos plugins disponibles en el entorno de

desarrollo. Un plugin es un complemento que se instala en el entorno de desarrollo, facilitan y agilizan el desarrollo.

Un plugin utilizado es el de Maven [7], Maven es una herramienta de gestión de proyectos. Se basa en un fichero central pom.xml, donde se definen todas las dependencias necesarias para el proyecto. Maven maneja las dependencias del proyecto, se las descarga y las añade al *classpath*. Otro puglin necesario es el de Spring, Spring es un *framework* para el desarrollo de aplicaciones de código abierto para la plataforma Java. En este proyecto se ha utilizado la versión 3 del *framework*, que permite definir una arquitectura de software Modelo-Vista-Controlador. Esta arquitectura define por un lado los componentes para la representación de la información, y por otro lado para la interacción del usuario.

En las figuras (3.2) y (3.3) se presenta el esquema seguido para el desarrollo del trabajo de fin de grado en Eclipse.

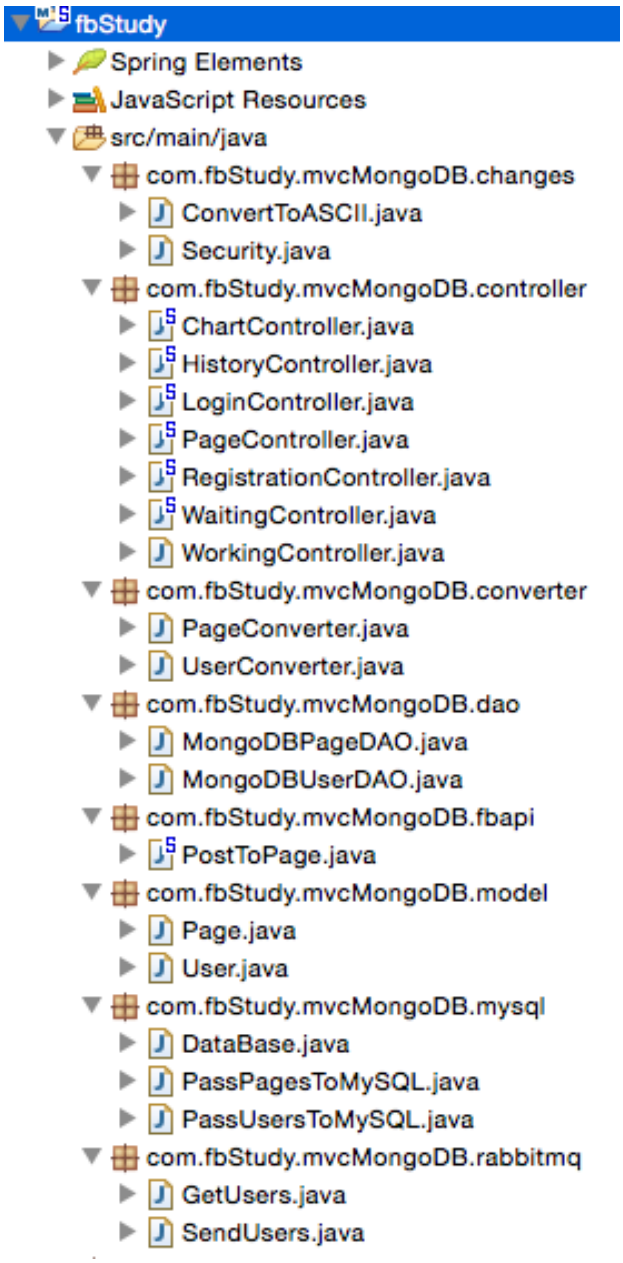

Figura 3.2: Esquema TFG en Eclipse. Parte 1

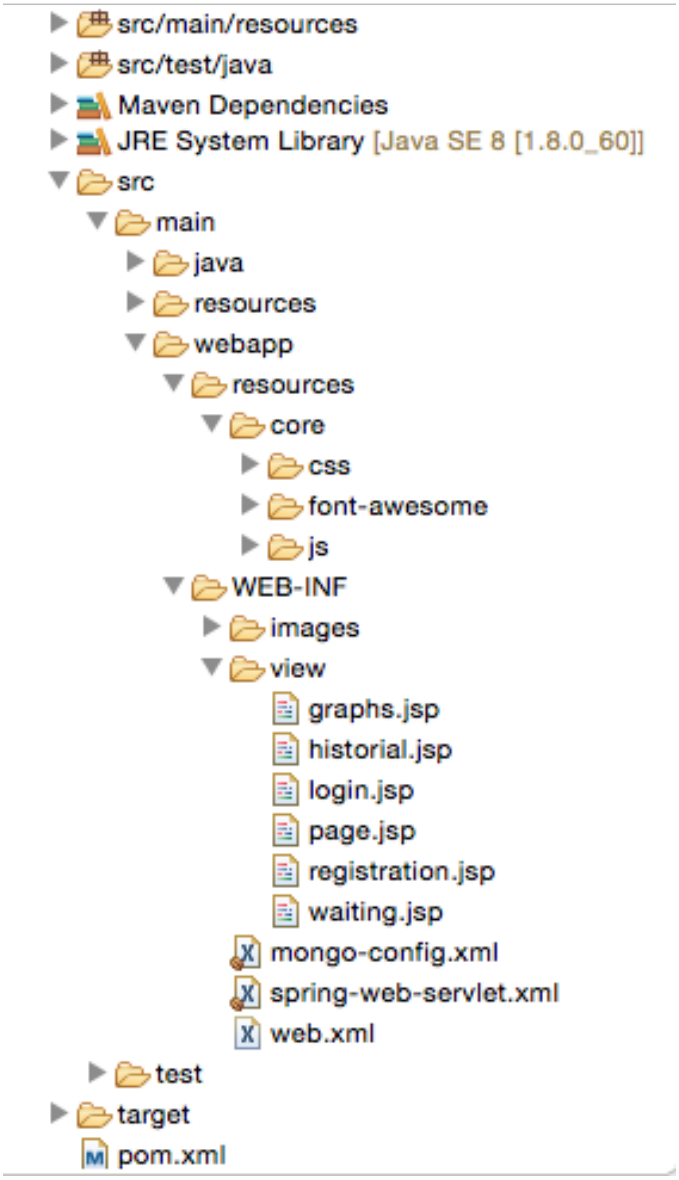

Figura 3.3: Esquema TFG en Eclipse. Parte 2

Como se puede observar en los esquemas anteriormente mostrados, dentro de la carpeta "src/main/java" se ha definido un paquete para cada una de las funcionalidades implementadas, además de las propias necesarias para el correcto funcionamiento de la aplicaciones web siguiendo la estructura MVC.

Por otro lado en la carpeta "webapp" se definen todas las vistas necesarias para la aplicación web, además de los estilos y plantillas utilizadas, y la configuración del servidor web, que se describe en el siguiente apartado 3.4.2.

A continuación en las tablas (3.1) y (3.2), se detalla la funcionalidad de las clases mostradas en los esquemas anteriores:

CLASE	FUNCIONALIDAD
Paquete changes	
ConvertToASCII.java	Esta clase convierte el nombre del estudio introducido por el usuario para poder almacenarlo correctamente en las bases de datos sin caracteres especiales.
Security.java	Esta clase encripta los ID de los usuarios y sus contraseñas para utilizarlos y almacenarlos de forma segura.
Paquete controller	
ChartController.java	Controlador que gestiona la información de las páginas de Facebook y de los usuarios bajo estudio y la adapta para representarla en las gráficas en la vista "graphs.jsp".
HistoryController.java	Controlador que obtiene todas las búsquedas del usuario que ha iniciado sesión y las envía en una Lista para mostrarlos en una tabla en la vista "historial.jsp"
LoginController.java	Controlador que verifica si un usuario está registrado en la aplicación o no, o si ha metido los datos correctamente, para poder acceder a la web. Vista "login.jsp"
PageController.java	Controlador que gestiona el formulario de una nueva búsqueda realizada por el usuario. Guarda los datos introducidos por el usuario en la base de datos. Vista "page.jsp"
RegistrationController.java	Controlador encargado de registrar a nuevos usuarios almacenándolos en la base de datos. Vista "resgistration.jsp"
WaitingController.java	Controlador que muestra la vista "waiting.jsp" para indicar al usuario que el estudio solicitado está en curso.
WorkingController.java	Controlador encargado de manejar las tareas asíncronas, solicita a la clase PostToPage coger los datos del API de Facebook, al finalizar envía un correo al usuario para avisarle de que ha acabado. Además si el usuario ha solicitado también el estudio de los usuarios se repite el mismo proceso asíncronamente y se envía otro correo al usuario al teminar esta segunda parte.
Paquete converter	
PageConverter.java	Clase que convierte una variable de tipo Document, estraida de la base de datos de MongoDB a tipo Page para poder manejar los datos dentro de la aplicación. Y al contrario, para añadirlo el elemento a la base de datos.
UserConverter.java	Clase que convierte una variable de tipo Document, extraída de la base de datos de MongoDB a tipo User para poder manejar los datos dentro de la aplicación. Y al contrario, para añadirlo el elemento a la base de datos.

Tabla 3.1: Clases de java definidas en el proyecto. Parte 1

Paquete dao (Data Access Object)	
MongoDBPageDAO.java	Clase de acceso a datos almacenados en MongoDB para diferentes operaciones CRUD. Para objetos de tipo Page
MongoDBUserDAO.java	Clase de acceso a datos almacenados en MongoDB para diferentes operaciones CRUD. Para objetos de tipo User
Paquete fbapi	
PostToPage	Clase que realizar de manera asíncrona a la aplicación web las consultas de las páginas al API de Faacebook.
Paquete model	
Page.java	Define el objeto Page del modelo de esta aplicación. Este objeto contiene todos los campos indicados por el usuario en una nueva búsqueda.
User.java	Define el objeto Page del modelo de esta aplicación. Este objeto contiene todos los campos del registo de un usuario.
Paquete mysql	
DataBase	Clase que define las credenciales para conectarse a la base de datos de MySQL
PassPagesToMySQL.java	Clase que ordena los datos de las páginas guardados en MongoDB obtneidos del API de Facebook y los almacena en una tabla en MySQL
PassUsersToMySQL.java	Clase que ordena los datos de los usuarios guardados en MongoDB obtenidos del crawler de Facebook y los almacena en una tabla en MySQL
Paquete rabbitmq	
GetUsers.java	Clase que obtiene los usuarios, de las páginas de Facebook bajo estudio, más comunes en los datos recogidos de la tabla creada en MySQL en la clase PassPagesToMySQL.java descrita anteriormente.
SendUsers.java	Clase que implementa el protocolo de envío de mensajes de RabbitMQ. Envía una lista de usuarios y el nombre de una página de Facebook a cada máquina virtual disponible. Cada una de las máquinas lanzaran el crawler para recoger la información de los usuarios dados.

Tabla 3.2: Clases de java definidas en el proyecto. Parte 2

3.4.2. Servidor web Jetty

Figura 3.4: Logo Servidor Web Jetty

Jetty es un Servidor Web y Contenedor de *Servlet*s preparado para ser embebido en las aplicaciones. Es un proyecto de software libre y está completamente basado en Java. Los motivos por los que se ha usado este servidor en concreto son los siguientes:

- Es ligero y eficiente
- Está alojado en Eclipse
- Basado en Java
- Soporta peticiones HTTP POST y GET
- Procesa peticiones asíncronas, capaz de manejar el mecanismo de consulta larga

Además de cumplir todos los requisitos necesarios para este proyecto, otro de los motivos por los que se ha utilizado es por la simplicidad de su configuración, basta con definirlo en un fichero XML o en una clase java e incluir sus dependencias en el fichero pom.xml. Con estos dos pasos, la aplicación se ejecuta directamente desde este servidor web.

```
INFO  XmlBeanDefinitionReader - Loading XML bean definitions from ServletContext resource [/WEB-INF/spring-web-
INFO  AutowiredAnnotationBeanPostProcessor - JSR-330 'javax.inject.Inject' annotation found and supported for a
INFO  DefaultListableBeanFactory - Pre-instantiating singletons in org.springframework.beans.factory.support.De
INFO  RequestMappingHandlerMapping - Mapped "{[/charts],methods=[GET],params=[],headers=[],consumes=[],produces
INFO  RequestMappingHandlerMapping - Mapped "{[/historial],methods=[GET],params=[],headers=[],consumes=[],produ
INFO  RequestMappingHandlerMapping - Mapped "{[/ || /login || /logout],methods=[GET],params=[],headers=[],consu
INFO  RequestMappingHandlerMapping - Mapped "{[/ || /login || /logout],methods=[POST],params=[],headers=[],cons
INFO  RequestMappingHandlerMapping - Mapped "{[/page],methods=[GET],params=[],headers=[],consumes=[],produces=[
INFO  RequestMappingHandlerMapping - Mapped "{[/page],methods=[POST],params=[],headers=[],consumes=[],produces=
INFO  RequestMappingHandlerMapping - Mapped "{[/registration],methods=[GET],params=[],headers=[],consumes=[],pr
INFO  RequestMappingHandlerMapping - Mapped "{[/registration],methods=[POST],params=[],headers=[],consumes=[],p
INFO  RequestMappingHandlerMapping - Mapped "{[/waiting],methods=[GET],params=[],headers=[],consumes=[],produce
INFO  Version - Hibernate Validator 4.2.0.Final
INFO  SimpleUrlHandlerMapping - Mapped URL path [/resources/**] onto handler 'org.springframework.web.servlet.r
INFO  SimpleUrlHandlerMapping - Mapped URL path [/images/**] onto handler 'org.springframework.web.servlet.reso
INFO  DispatcherServlet - FrameworkServlet 'spring-web': initialization completed in 2256 ms
[INFO] Started o.e.j.m.p.JettyWebAppContext@79add732{/fbAPP,file:/Users/Patri/Desktop/TFG/fbStudy/src/main/weba

[INFO] Started ServerConnector@79fc6c3d{HTTP/1.1}{0.0.0.0:8080}
[INFO] Started @20189ms
[INFO] Started Jetty Server
[INFO] Starting scanner at interval of 10 seconds.
```

Figura 3.5: Registro del servidor web Jetty

Capítulo 4

Funcionamiento del sistema

Para explicar en profundidad el funcionamiento de la aplicación web, se va a describir dividiéndolo en dos grandes grandes bloques, por un lado la parte del cliente, lo que el usuario ve, denominado frontend, y por otro lado la parte del servidor, cómo está distribuido el sistema para poder realizar todas las funcionalidades, denominado backend.

4.1. Frontend

En el frontend se define la interfaz de usuario. Como ya se ha descrito en la sección (3.3.6), la aplicación web está basada en Spring MVC. Se trata de un Modelo-Vista-Controlador.

En esta sección se va a definir la Vista, que depende directamente del Modelo que se ha definido para que el usuario pueda interactuar con la aplicación. Además de indicar qué acciones realiza el Controlador para el correcto funcionamiento de cada una de las vistas detallas.

En el Modelo de esta aplicación se han diseñado dos objetos:

- Modelo User: define todos los campos de información relacionados con el usuario.
- Modelo Page: define todos los campos de información relacionados con la búsqueda de páginas.

Una vez definido el Modelo, la Vista ya puede interactuar con ellos mediante el Controlador que los maneja. Un Controlador responde a eventos, acciones del usuario, e invoca peticiones al Modelo cuando se hace alguna solicitud sobre la información. Cada Controlador se encarga de una vista, definidos en la tabla (3.1).

Para facilitar el desenlace de este apartado en la figura (4.1) representa un diagrama del funcionamiento de la web básico.

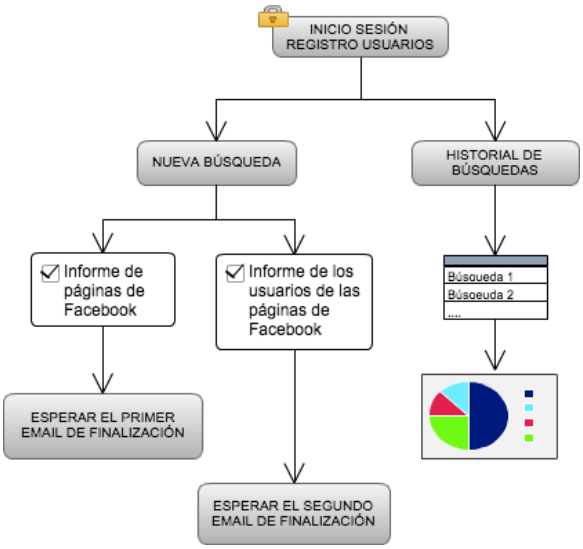

Figura 4.1: Esquema funcionamiento frontend

Cada una de las pantallas está definida en una vista distinta que se definen a continuación por separado, explicando en detalle cada una de las acciones que el usuario puede realizar.

4.1.1. Registration

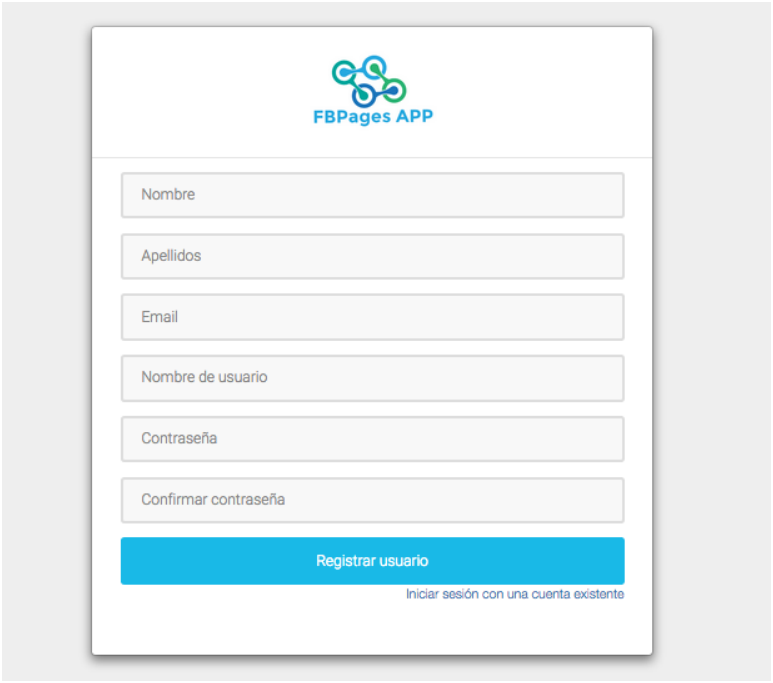

Figura 4.2: Captura de pantalla del formulario de Registro

En esta vista se define un formulario con un objeto de tipo "User", donde el usuario debe introducir todos los datos relacionados con su perfil, para poder crear una cuenta nueva en la aplicación.

El controlador "RegistrationController.java" se encarga de comprobar que el usuario haya rellenado todos los campos del formulario, y que los datos introducidos sean correctos, en concreto que el campo email, se corresponda con una dirección de correo electrónico, y que la contraseña introducida coincida con la de verificación.

Si el formulario es correcto, se crea un nuevo usuario que se almacena en la base de datos en la colección "users".

4.1.2. Login

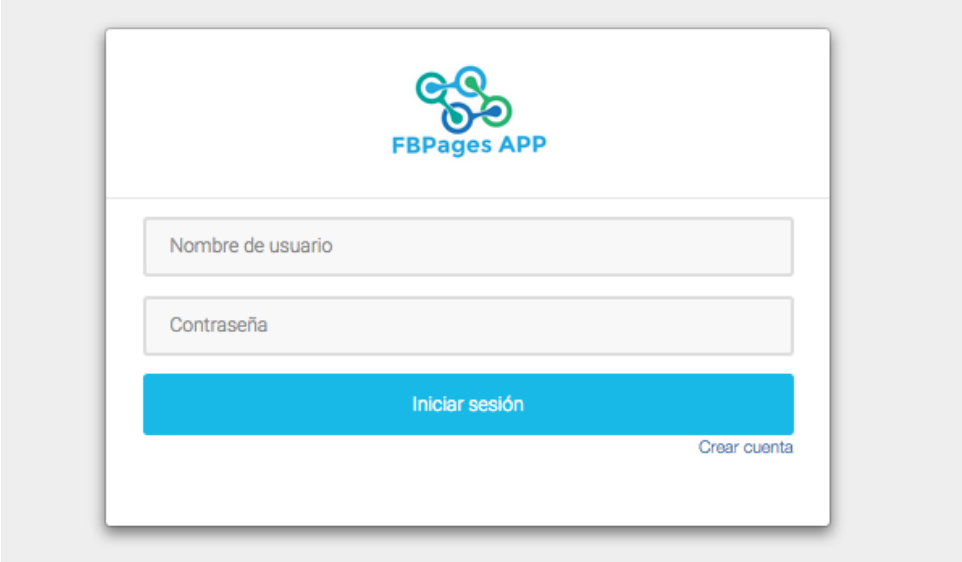

Figura 4.3: Captura de pantalla del formulario de Inicio de Sesión

En esta vista se presenta el formulario de inicio de sesión del usuario. Para poder acceder a la aplicación se requiere estar previamente registrado.

De forma similar a la anterior vista, el controlador "LoginController.java" se encarga de que el usuario haya introducido bien el nombre de usuario y la contraseña, de ser así, se conecta con la base de datos y comprueba que exista dentro de la colección "users". Si existe, le da acceso a la aplicación, y si no le indica que no existe el usuario.

4.1.3. Búsqueda

Esta es la pantalla principal de la aplicación, aquí el usuario definirá las características del estudio que quiere realizar. Este formulario se define mediante el modelo "Page". Para facilitar la búsqueda al usuario, el formulario se ha dividido en cuatro pasos. Cada paso, está formado por un formulario sencillo, incluyen una breve explicación de los datos que se deben introducir.

A continuación se explica cada uno de los pasos brevemente, los campos necesarios en cada uno y además se muestra su aspecto visual:

- Paso 1/4

Figura 4.4: Captura de pantalla del Paso 1

El usuario debe indicar el nombre del estudio que va a realizar

- Paso 2/4

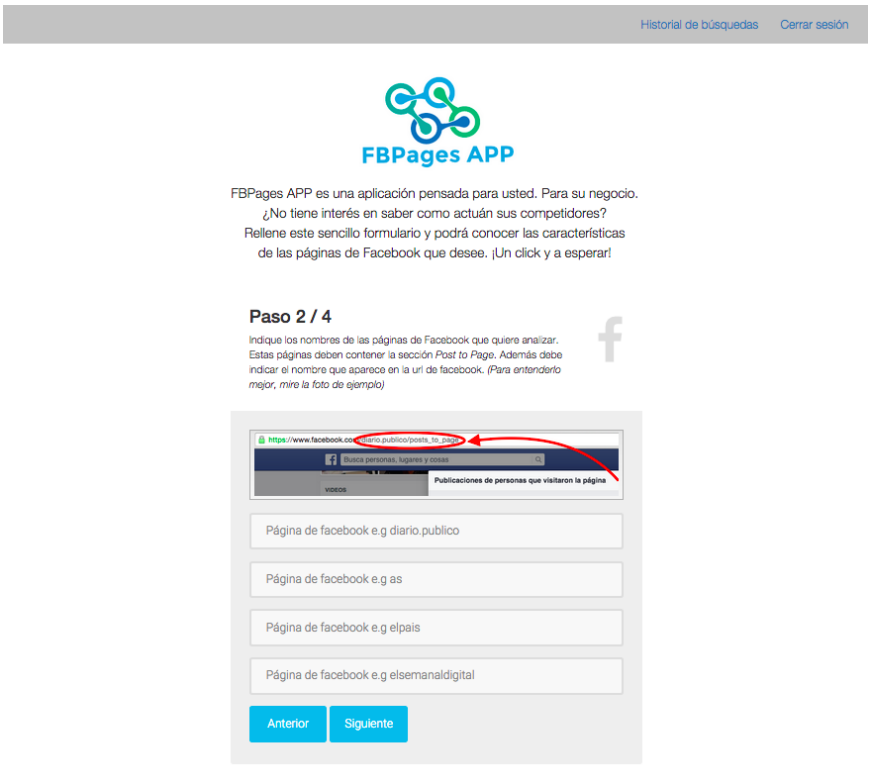

Figura 4.5: Captura de pantalla del Paso 2

El usuario debe introducir el nombre de las cuatro páginas de Facebook que quiere

analizar. Como se puede observar en la figura (4.5), se ha introducido una imagen explicativa de la comprobación que debe hacer el usuario antes de añadir una página al formulario, para saber si dicha página contiene la sección *Post To Page*, explicada anteriormente en el presente documento. Si contiene dicha sección, el nombre que debe introducir el usuario en el formulario, es el que aparece en la url de Facebook, para asegurarse de que es la página correcta la que se va a analizar.

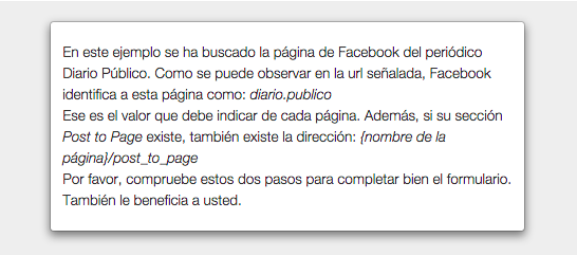

Figura 4.6: Captura de pantalla del diálogo de ayuda del paso 2

Este paso es necesario ya que en Facebook existen muchas páginas con nombres similares pero de contenido totalmente opuesto, de esta forma, el usuario se asegura de que la página que se va a analizar es la correcta.

- Paso 3/4

Figura 4.7: Captura de pantalla del Paso 3

En este paso el usuario debe indicar la fecha de inicio del estudio. Con este dato, la petición realizada a la API de Facebook se realizará desde la fecha de inicio hasta la fecha actual. Con este dato, el usuario puede decidir si quiere realizar un análisis de las últimas semanas, meses o años.

- Paso 4/4

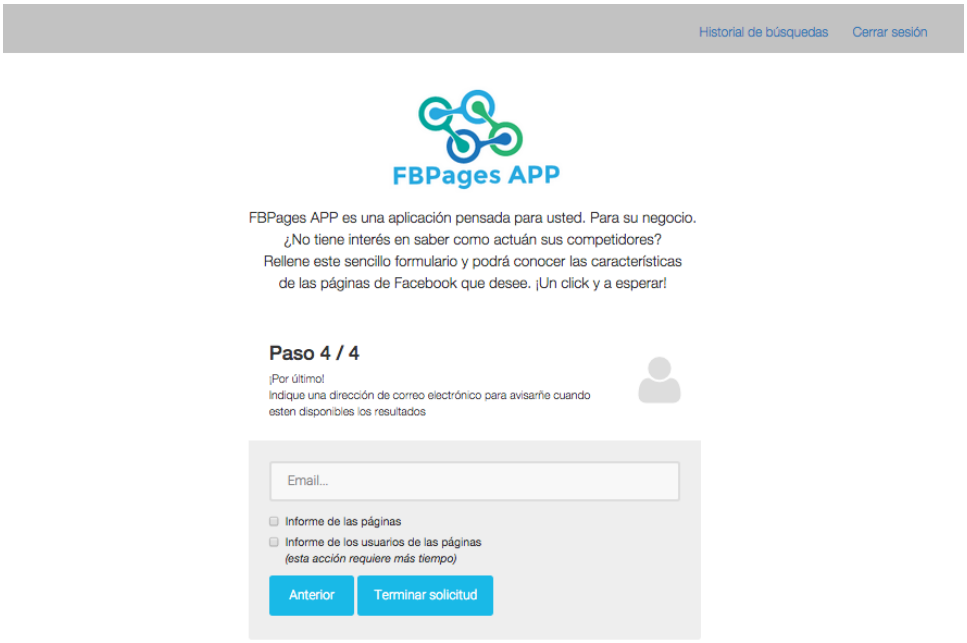

Figura 4.8: Captura de pantalla del Paso 4

Por último, el usuario debe indicar la dirección de correo donde quiere que se le avise cuando el estudio haya terminado de recopilar los datos.

Una vez completado el formulario, el controlador "PageController.java" se encarga de almacenar esta búsqueda en la base de datos en la colección "pages", además de incluir los datos de la búsqueda introducidos por el usuario, también se añade al documento de la base de datos el "ID" del usuario, de forma que se pueda relacionar cada búsqueda con su usuario correspondiente.

Acto seguido, el controlador se encarga de llamar a la siguiente acción. De cara al usuario, aparece una pantalla en la que se le indica que se están recogiendo los datos y que una vez haya acabado este proceso se le enviará un correo electrónico para notificárselo, tal y como se muestra en la siguiente figura (4.9).

Figura 4.9: Captura de pantalla de la página de espera

No es tan sencillo como el usuario puede apreciar. Del controlador "PageController.java" se llama al controlador "WorkingController.java". Este controlador lo primero que hace es redirigir la página web a la vista "waiting.jsp" mostrada en la anterior figura. Esta vista está manejada por el controlador "WaitingController.java" que sólo se encarga de mostrar dicha vista.

Por otro lado, el controlador "WorkingController.java" inicia una tarea asíncrona. Una tarea asíncrona es aquella que es realiza en segundo plano para no alterar el flujo de la aplicación. Para manejar tareas asíncronas en una aplicación web en Spring MVC, se ha creado un *Servlet* asíncrono. Un *Servlet* es una clase de lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor. Son utilizados para manejar las páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador. Un *Servlet* asíncrono tiene además la función de manejar hilos de tareas que requieren un largo tiempo de espera, de esta forma el navegador no está cargando hasta que se finalice la tarea.

En la siguiente figura 4.10 se muestra el esquema de funcionamiento de un *Servlet* asíncrono.

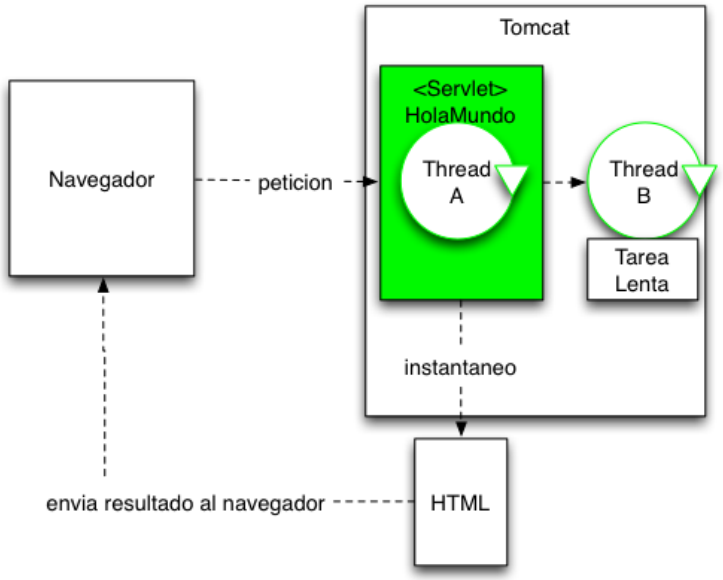

Figura 4.10: Ejemplo de funcionamiento de un *Servlet* asíncrono

La tarea asíncrona que maneja este controlador es la conexión con la API de Facebook para sacar los datos de las páginas solicitadas, esta función se explica en detalle más adelante en la sección (4.2.1).

Una vez completada esta tarea, el controlador realiza dos pasos. Por un lado, envía un correo electrónico al usuario para notificarle que el estudio de las páginas de Facebook ha terminado, enviando el enlace donde puede acceder para ver los resultados. Por otro lado, envía un correo al administrador de la aplicación, en este caso la propia autora del trabajo fin de grado, para indicar que se ha solicitado el estudio de usuarios de las páginas, esta parte del proyecto se ha definido así porque el *crawler* requiere ciertos servicios que deben estar correctamente activados antes de lanzarlo automáticamente, el funcionamiento de esta parte se explica más adelante en el sección (4.2.3).

4.1.4. Historial

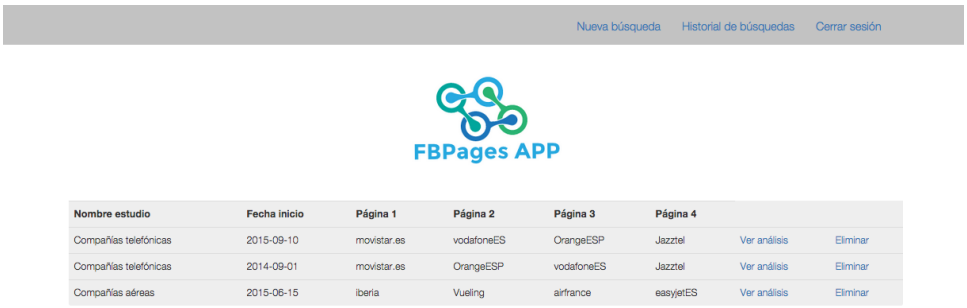

Figura 4.11: Captura de pantalla de la página Historial

Otra de las funciones a las que tiene acceso el usuario es a ver todos los estudios que ha solicitado en la aplicación. Esta vista muestra una tabla con todos los datos relacionados con el usuario. Desde esta tabla el usuario tiene dos acciones: "Ver el análisis" del estudio que elija o "Eliminar" esa búsqueda.

Internamente el controlador "HistoryController.java" solicita a la base de datos "pages" que le devuelva todos los objetos que contienen el "ID" de usuario que corresponden con el que ha iniciado sesión en la aplicación, y devuelve la lista de páginas a la vista que se renderiza en la tabla que se observa en la figura (4.11).

Además, si el usuario pulsa en la opción "Ver análisis", el controlador redirecciona la página a la vista "graphs.jsp" que es la encargada de mostrar los gráficos diseñados para la búsqueda seleccionada.

Si por el contrario el usuario selecciona la opción "Eliminar", el controlador mediante una operación sencilla del CRUD definido para las páginas, eliminará esa entrada de la base de datos.

4.1.5. Resultados

Por último, la vista que muestra los resultados de un estudio concreto. Los usuarios pueden acceder a esta vista pulsando en el link que se les envía por correo electrónico o desde el historial, pulsando en la opción "Ver análisis". Pulsando en dicha opción, el usuario vería la siguiente pantalla que se muestra en la siguiente figura (4.12)

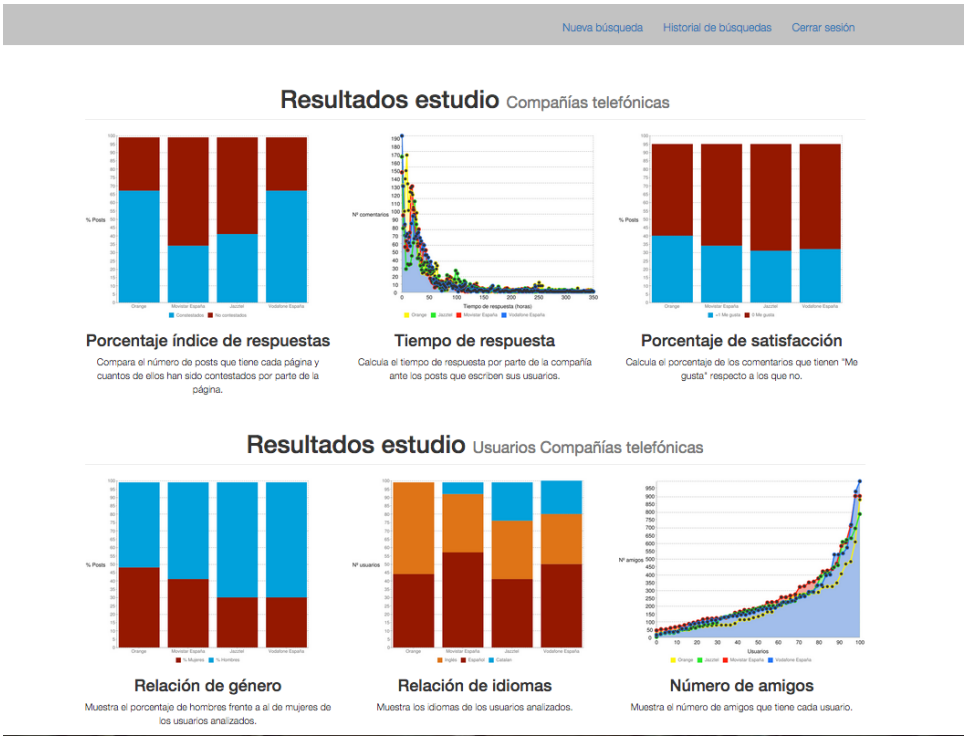

Figura 4.12: Captura de pantalla del resultado del análisis

Esta vista se maneja por el controlador "ChartController.java". Se encarga de procesar los datos recogidos en la base de datos de MySQL con el nombre concreto de la búsqueda y los representa en gráficas mediante Google Chart.

Google Chart es un herramienta para desarrolladores que permite la creacción de gráficas en forma de imágenes PNG para poder utilizarlas en las páginas web. Para utilizar esta herramienta se ha utilizado una librería de java "charts4j" [4] que permite integrar las funcionalidades de Google Chart en la aplicación.

Se han definido las gráficas por defecto, lo único que cambian son los datos que las definen, que son cogidos de la base de datos. En el siguiente listado se muestran todas las gráficas que se han realizado y un ejemplo de cada una de ellas:

1. **ESTUDIO BASADO EN LAS PÁGINAS DE FACEBOOK**
   - **Porcentaje índice de respuesta**

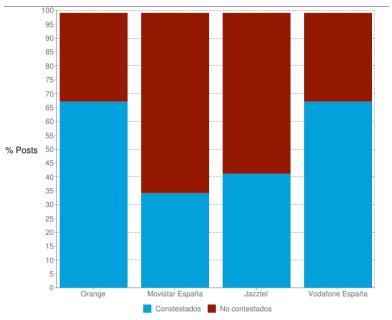

Figura 4.13: Gráfica porcentaje índice de respuesta

Esta gráfica presenta los resultados de los datos obtenidos, obteniendo el porcentaje del total de los comentarios que ha contestado la página bajo estudio frente a los que no.

- **Tiempo de respuesta**

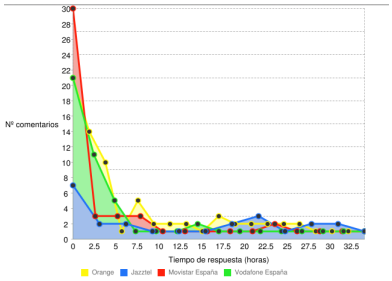

Figura 4.14: Gráfica tiempo de respuesta

Esta gráfica muestra el tiempo de respuesta por parte de la página a los comentarios que escriben los usuarios. Este tiempo está calculado en horas.

- **Porcentaje de satisfacción**

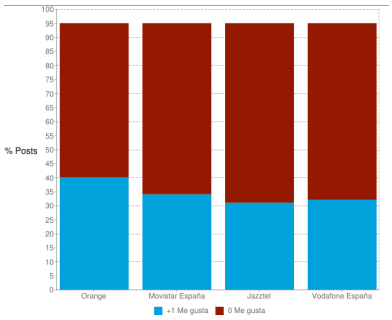

Figura 4.15: Gráfica porcentaje de satisfacción

Este gráfico representa el porcentaje de comentarios que han obtenido algún "Me gusta" (*Likes*, comúnmente conocido) frente a los que no.

2. **ESTUDIO BASADO EN LOS USUARIO DE LAS PÁGINAS DE FACEBOOK**

- **Relación de género**

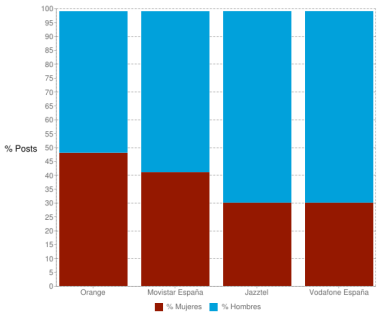

Figura 4.16: Gráfica relación de genero

Esta gráfica muestra el porcentaje de hombres y mujeres dentro de los usuarios analizados de cada página.

- **Relación de idiomas**

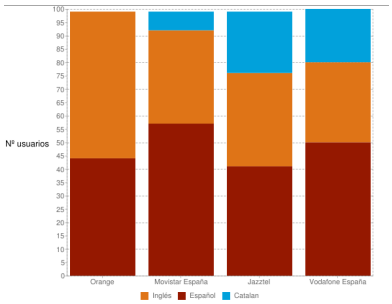

Figura 4.17: Gráfica relación de genero

Esta gráfica representa el porcentaje de usuarios de habla ingles, española o catalana. Aunque no es un dato muy fiable, dado que muchos usuarios dominarán más del idioma materno, por norma general, aquellos usuarios que especifican un idioma, indican su propia lengua materna.

- **Número de amigos**

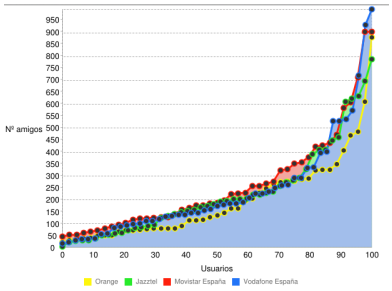

Figura 4.18: Gráfica número de amigos

En esta gráfica se muestra el número de amigos que tiene cada uno de los usuarios analizados. El fin de esta gráfica es considerar si son cuentas activas o no, ya que si el porcentaje de usuarios con cero amigos es muy elevado, podría indicar que son usuarios falsos.

4.2. Backend

En el lado del servidor tenemos una estructura más compleja, que se va a explicar dividiéndolo en cada una de las herramientas utilizadas o creadas para el correcto funcionamiento de la aplicación. Primeramente, se va a mostrar el diagrama completo del backend, tal y como se muestra en la figura (4.19).

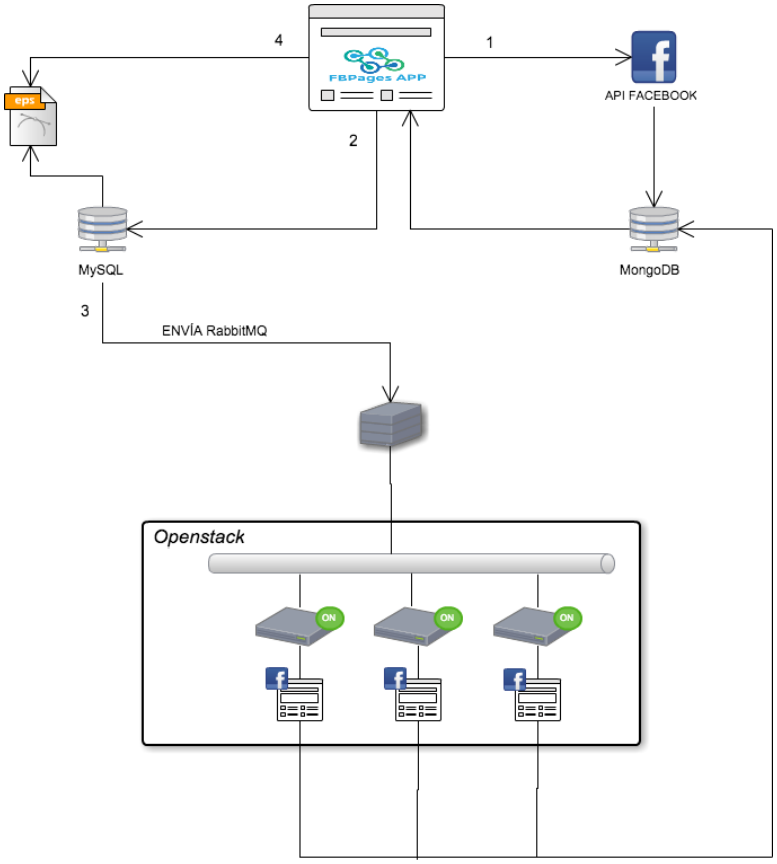

Figura 4.19: Esquema funcionamiento backend

Como se puede observar en la anterior figura, hay varios pasos implementados, que tienen que seguir el orden indicado para poder obtener todos los datos requeridos.

4.2.1. API Facebook Developers

El primer paso indicado es la API de Facebook. El usuario en la aplicación indicará los datos necesarios para realizar la consulta a la API, una vez obtenidos esos datos por parte del usuario, se lanza un script, de manera asíncrona, tal y como se ha explicado en la sección (4.1.3). En este punto se va a explicar más en detalle como funciona este programa.

La API de Facebook es la principal forma de obtener los datos dentro y fuera del gráfico social de Facebook. Es una API basada en HTTP que se ha utilizado para consultar datos. A continuación se muestra un ejemplo de una petición a la API de Facebook, mediante su herramienta *Graph API*. Esta herramienta permite ver los resultados de la consulta que se desea hacer.

Figura 4.20: Esquema petición API Facebook

Para poder obtener los datos de las páginas de Facebook indicadas por el usuario de forma automática, se ha creado un programa que realiza las peticiones a la API de Facebook con los parámetros necesarios, en nuestro caso de la sección *tagged*, que se corresponde con los posts de los usuarios en las páginas.

La API de Facebook devuelve en formato JSON los resultados, este fichero es almacenado directamente en la base de datos de MongoDB, para su posterior procesado.

Esta acción se repite para cada una de las páginas indicadas por el usuario, hasta completar la obtención de datos necesaria.

El segundo paso consiste en leer los datos obtenidos y almacenados en MongoDB, coger sólo aquellos datos de interés para este trabajo fin de grado y ordenarlos en una tabla en MySQL. De esta forma, es más fácil acceder a ellos y representarlos.

4.2.2. Distribución: RabbitMQ

Una vez guardados los datos en MySQL, si el usuario ha seleccionado la opción de "Informe de los usuarios de las páginas", el siguiente paso es lanzar el *crawler* para recoger los datos de los usuarios.

Antes de explicar el funcionamiento del *crawler*, se va a explicar la arquitectura desplegada mediante RabbitMQ para la correcta distribución del *crawler*. Como se ha explicado en el Estado del Arte, RabbitMQ es un sistema de intercambio de mensajes. La utilización en este trabajo fin de grado se basa en el envío de mensajes a cuatro máquinas virtuales en las que se ejecutará el *crawler*.

El sistema es muy sencillo, se configuran las cuatro máquinas virtuales con el entorno necesario para ejecutar el *crawler*. Una vez hecho esto, se lanza el script de RabbitMQ que tiene la función de recibir. Cada VM recibe mensajes de una cola de mensajes distinta, por

lo que el emisor es el encargado de decidir que envía a cada VM, o lo que es lo mismo a cada cola.

Por otro lado, desde la aplicación se maneja el script encargado de enviar los mensajes. Este script manda un mensaje a cada una de las colas de mensajes activas indicando el número de usuarios a analizar y el nombre de la página de donde han sido extraídos. Una vez enviado el mensaje, se queda esperando para recibir la confirmación de que el trabajo ha finalizado. De esta forma, se distribuye el *crawler* en cada máquina virtual de forma automática, sin necesidad de ir a cada una de las VMs para lanzar el *crawler*.

4.2.3. Crawler de Facebook: Selenium WebDriver

El *crawler* es un script escrito en Java, que tiene como función principal recoger los datos de un usuario de Facebook. Se basa en reproducir los pasos que haría un humano si mirase el perfil de un usuario en Facebook.

Este *crawler* funciona lanzando una instancia del navegador Firefox, indicando la url de Facebook. Como toda persona que pertenece a esta comunidad, lo primero que debe hacer es iniciar sesión con una cuenta existente, y una vez dentro, procede a visitar los perfiles de cada uno de los usuarios que se le han indicado. El número de usuarios estipulado para este proyecto han sido 100, número suficiente para poder perfilar los patrones de los usuarios más comunes de una página web.

La instancia que lanza en navegador recibe el nombre de WebDriver, permite configurar y ejecutar los comandos necesarios para parsear la web y coger los datos deseados.

Los datos recogidos de los usuarios en Facebook, son almacenados de igual forma que los de la API de Facebook, se crea un objeto JSON con todos los campos obtenidos del perfil y se almacena directamente en MongoDB. Se ha definido de esta forma por si en un futuro cambian los campos del perfil de un usuario de Facebook, añadiendo o eliminando algún campo, solo habría que modificar el fichero JSON que se va a introducir en la base de datos, sin necesidad de modificar nada más.

4.2.4. Análisis gráfico: Google Chart

Por último, el cuarto paso consiste en analizar todos los datos obtenidos tanto de la API de Facebook como del *crawler* para representarlos gráficamente y que el usuario pueda verlo en la interfaz de la aplicación.

Para representar los datos se ha utilizado la herramienta de Google Chart, que facilita la integración de las gráficas con una aplicación web. Esta herramienta es utilizada en este proyecto mediante una librería de java, ya comentada anteriormente en la sección (4.1.5). Se define el tipo de gráfica que se quiere representar, y se le pasa como parámetro los datos ya procesados, Google Chart se encarga de dibujarlos en las gráficas.

Hay dos grandes bloques de datos que se han almacenado en la base de datos para ser procesados, los de los usuarios y los de las páginas de Facebook.

Por un lado, los datos relacionados con los usuarios, extraídos mediante el *crawler*. De estos datos se obtiene la información de los usuarios en relación a su género, número de amigos e idiomas hablados y se calcula el porcentaje de cada una de las variables para cada

página. Una vez calculados los porcentajes, se representan en una tabla conjunta para ver la diferencia de porcentaje entre cada página.

Por parte de los datos obtenidos de la API de Facebook se han tenido en cuenta la fecha de creación de un mensaje y la fecha de actualización del mismo, si no existe la fecha de actualización significa que la página de Facebook no ha contestado al comentario, y si por el contrario existe, se calcula la diferencia de tiempo, pudiendo así obtener el tiempo de respuesta de la página de Facebook ante un comentario. Con estos dos datos, se obtienen las gráficas del porcentaje de respuesta y del tiempo de respuesta explicadas en la sección (4.1.5). También se extrae el dato de número de "Me gusta" que tiene cada comentario para calcular el porcentaje de satisfacción de los usuarios ante las publicaciones recogidas.

Capítulo 5

Ejemplo de utilización de la aplicación

Para mostrar el funcionamiento de este trabajo fin de grado gráficamente, a continuación se explica el ejemplo de un caso práctico con todos los pasos seguidos y los resultados obtenidos.

5.1. Caso práctico

Se supone un usuario que va a solicitar un estudio sobre cuatro páginas de Facebook relacionadas con compañías telefónicas.

- **PASO 1: Registro en la aplicación**
  Se registra en la aplicación como un nuevo usuario. Se llama usuarioPrueba. En la siguiente figura se muestra este paso una vez completado y creado satisfactoriamente.

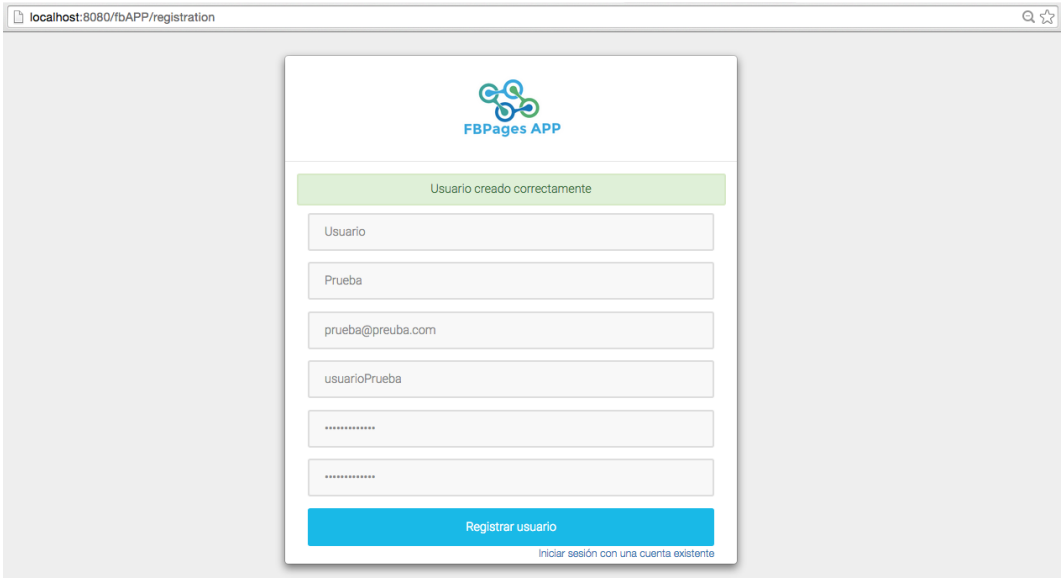

Figura 5.1: Captura de pantalla ejemplo Registro de usuario

- **PASO 2: Inicio de sesión**
  Una vez creado el usuario, ya puede acceder a la aplicación con el nombre de usuario y contraseña establecidos. A continuación se presenta un ejemplo de este paso gráficamente.

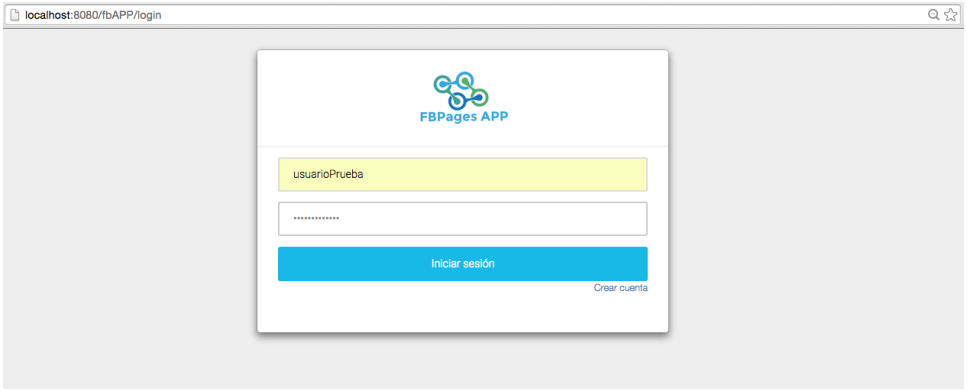

Figura 5.2: Captura de pantalla ejemplo Inicio de sesión

- **PASO 3: Solicitud del análisis a realizar** El siguiente paso se divide en cuatro. Consiste en definir las características del análisis de que va a realizar.

  1. En el primer formulario hay que indicar el nombre que va a englobar el estudio. En este caso, Compañías telefónicas.

Figura 5.3: Captura de pantalla ejemplo solicitud de análisis. Paso 1

  2. En el segundo hay que indicar el nombre de las páginas de Facebook que se van a analizar. En este caso práctico, se quien analizar las páginas de las compañías telefónicas Movistar, Orange, Vodafone y Jazztel.

     No es suficiente con poner el nombre de la compañía, hay que introducir el nombre que Facebook indica en la dirección URL. Para ello, se ha buscado la página de Movistar, a modo de ejemplo. Como es una compañía internacional, hay varias páginas posibles para esta búsqueda, en este caso, se va a elegir Movistar España. En la siguiente figura se muestra la búsqueda en Facebook para entenderlo mejor.

Figura 5.4: Captura de pantalla ejemplo búsqueda en Facebook

Además de elegir la página adecuada para los requisitos esperados del estudio, es conveniente comprobar que existe la sección *Post to Page* de la página seleccionada. Este paso se realiza añadiendo al final de la URL el texto "/posts_to_page", y si se carga una pantalla mostrando las publicaciones de los usuarios, significa que existe. A continuación se muestra gráficamente.

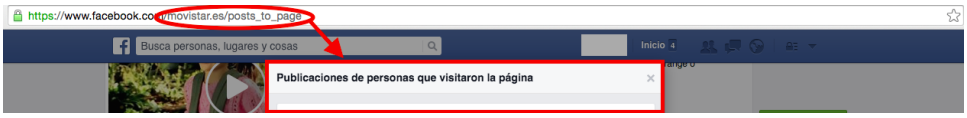

Figura 5.5: Captura de pantalla ejemplo comprobación sección *Post to Page*

Una vez realizadas las comprobaciones necesarias para las cuatro compañías telefónicas se introducen con el nombre adecuado en el formulario, tal y como se muestra en la siguiente figura.

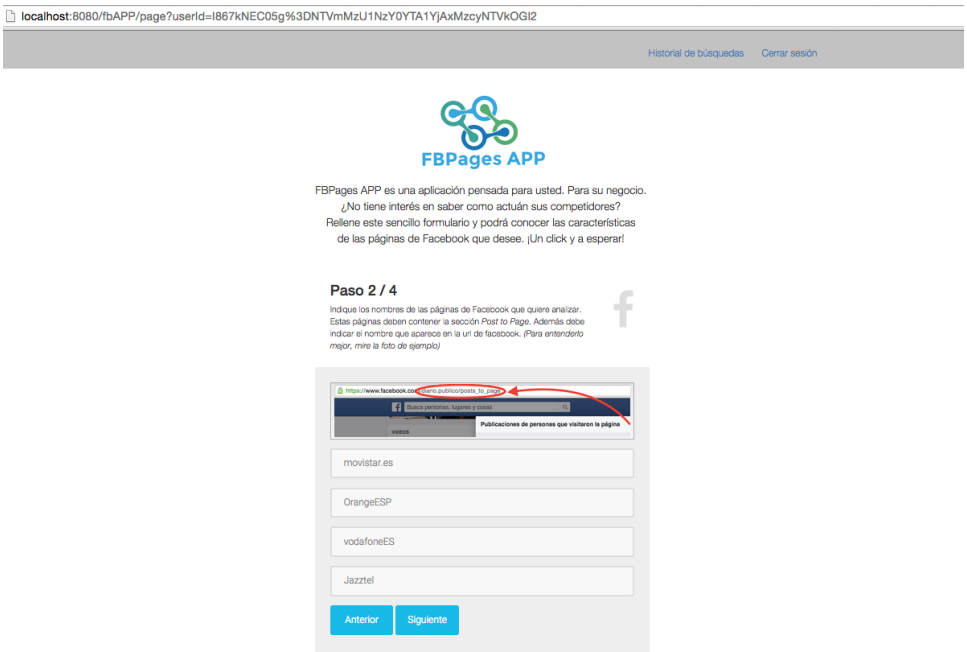

Figura 5.6: Captura de pantalla ejemplo solicitud de análisis. Paso 2

3. En el tercer paso, hay que indicar el periodo temporal del estudio a realizar. En

concreto se quiere realizar del último año, por lo que se introducirá la fecha de inicio el uno de septiembre de 2014, en el formato indicado por la aplicación.

Figura 5.7: Captura de pantalla ejemplo solicitud de análisis. Paso 3

4. Por último, se debe indicar una dirección de correo electrónico a la que se avisará una vez finalizado el estudio de Facebook, además de seleccionar las características del análisis. En este caso se van a seleccionar ambas, "Informe de las páginas" e "Informe de los usuarios de las páginas".

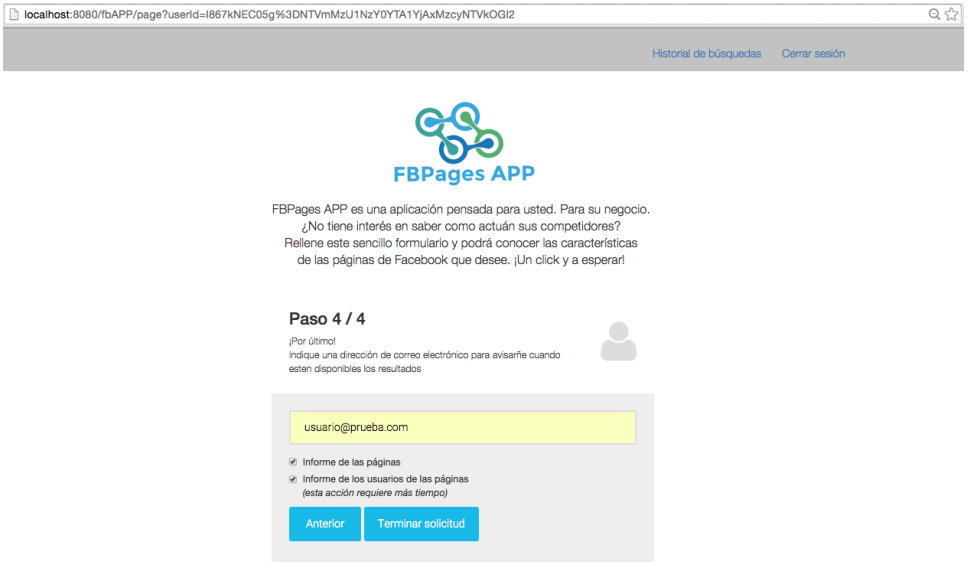

Figura 5.8: Captura de pantalla ejemplo solicitud de análisis. Paso 4

Una vez completados todos los formularios necesarios para realizar una solicitud de un análisis, se pulsa en el botón "Terminar solicitud".

- **PASO 4: Tiempo de espera hasta que llegue la notificación por correo**
  Después de pulse el botón "Terminar solicitud", se visualiza la pantalla que se muestra a continuación. En la que se indica al usuario que debe esperar a que se haya realizado el análisis de las páginas de Facebook.

Figura 5.9: Captura de pantalla ejemplo pantalla de espera

Pasado el tiempo necesario para obtener los datos del análisis, el usuario recibe un correo donde se le avisa de que ya puede ver los resultados del estudio. En la siguiente imagen, se muestra un ejemplo del correo recibido.

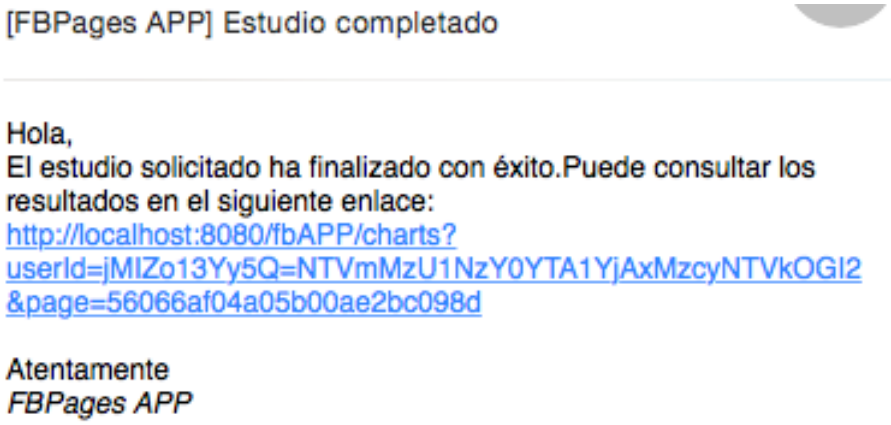

Figura 5.10: Captura de pantalla ejemplo correo recibido

- **PASO 5: Visualización de los resultados** Pinchando en el enlace indicado en el correo recibido (Figura 5.10) se pueden ver los resultados. Otra opción es acceder de nuevo a la aplicación y pulsar sobre la opción "Historial de búsquedas", donde aparece una tabla que recoge todos los análisis solicitados y se selecciona la opción "Ver análisis". En la siguiente figura se muestra el Historial de Búsquedas del usuarioPrueba.

Figura 5.11: Captura de pantalla ejemplo historial de búsquedas del usuario

Por último, para finalizar la explicación de este caso práctico, se presentan los resultados que ha obtenido el usuario después de todo el proceso.

Dado que el análisis de los usuarios requiere más tiempo, si el usuario accede automáticamente después de recibir el correo, sólo ve el análisis de las páginas de Facebook, y el análisis de los usuarios en espera. Tal y como se muestra en la siguiente figura.

Figura 5.12: Captura de pantalla ejemplo análisis obtenido. Sólo las páginas

Si transcurrido un tiempo vuelve a acceder a la aplicación, ya estarán disponibles los resultados de los usuarios, visualizando el análisis completo.

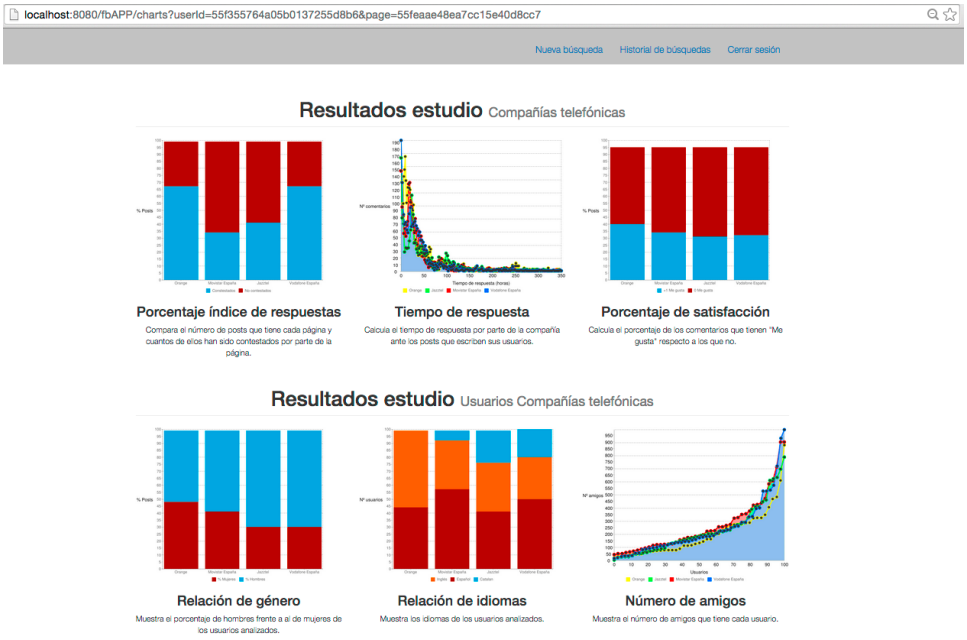

Figura 5.13: Captura de pantalla ejemplo análisis obtenido. Completo

Capítulo 6

Planificación del trabajo y presupuesto

6.1. Planificación del trabajo

6.1.1. Definición de tareas

La elaboración del presente proyecto se ha dividido en tres fases, que se enumeran a continuaón:

- FASE 1: Planificación
  En todo proyecto antes de empezar a trabajar es necesario un estudio previo de los requisitos necesarios y de las posibles tecnologías para desarrollar el trabajo.
  - *Planteamiento del trabajo:* La primera tarea consistió en estudiar cuales eran los objetivos necesarios para la realización de este proyecto, su finalidad y una previa organización del mismo.
  - *Definición de requisitos:* Una vez examinados los objetivos indispensables, se realizó un listado de los requisitos tanto técnicos como materiales necesarios para llevar a cabo este proyecto.
  - *Estudio de sistemas de desarrollo:* El siguiente paso fue el análisis de las diferentes alternativas para desarrollar este trabajo fin de grado, definiendo que lenguaje de programación utilizar, las tecnologías adecuadas, y la elección de la base de datos.
  - *Estudio de las tecnologías web:* Por último, se estudiaron las diferentes tecnologías web disponibles para desarrollar el portal web y las opciones que más convenían para este proyecto.
- FASE 2: Desarrollo

  2.1 BACKEND
  - *Creación de las bases de datos:* Definido el proyecto, se empezó creando las bases de datos necesarias, una en MongoDB y otra en MySQL. Los requisitos de las bases de datos creadas se fueron incrementando en función de las necesidades de los programas que acceden a ellas.

- *Creación script API de Facebook:* Después de crear un primer esquema de las bases de datos, el siguiente paso fue la realización del script que accedía a la API de Facebook. Hubo que analizar las diferentes posibilidades de acceder a la API de Facebook, los parámetros necesarios y la configuración adecuada.
- *Creación del crawler de Facebook:* Realizadas las primeras pruebas del script de la API de Facebook y teniendo los primeros datos para poder seguir avanzando con el proyecto, se procedió a desarrollar el *crawler*, tarea que requería mayor tiempo de dedicación.
- *Distribución del crawler en distintas máquinas virtuales:* Desarrollado el *crawler* casi por completo, se decidió montar una estructura para dividir el *crawler* automáticamente en varias máquinas virtuales. Para esta parte se desarrolló un sistema de colas de mensajes con RabbitMQ que supuso también mucho tiempo de pruebas pero ayudó a definir mejor los requisitos del *crawler*.

2.2 FRONTEND

- *Implementación del diseño de la aplicación web:* En cuanto al *frontend*, lo primero que se realizó fue una primera plantilla sencilla con un formulario que fuera capaz de introducir los datos indicados por un usuario en una base de datos. Una vez conseguido esto, se decidió el diseño de la web y se mejoraron aspectos de estilo.
- *Implementación del registro de usuarios:* Definidas las funcionalidades básicas de la aplicación web, se creó un sistema de registro/autenticación, para que sólo los usuarios registrados pudieran tener acceso a la aplicación.
- *Implementación de las funcionalidades de la aplicación:* Una vez desarrolladas todas las partes necesarias para la creación de este proyecto, sólo faltaba unir los script creados en el *backend* con la aplicación para que automáticamente parseará y obtuviera los datos de Facebook, los procesara y se los mostrara al usuario en la interfaz gráfica.

- FASE 3: Evaluación y documentación
  El último bloque para finalizar el desarrollo de este trabajo fin de grado consistía en realizar pruebas de la aplicación y la redacción de la presente memoria, así como la preparación de la presentación del mismo.

  - *Pruebas aplicación:* Finalizado el sistema completo, se realizaron pruebas para comprobar el correcto funcionamiento y se llevaron a cabo un par de casos prácticos para explicar su funcionamiento en el presente documento.
  - *Redacción de la memoria:* Redacción del presente documento.
  - *Preparación de la presentación.*

6.1.2. Planificación: Diagrama de Gantt

Una vez establecidas las tareas principales que definen el proyecto, se enmarcan dichas tareas en un intervalo temporal para establecer el tiempo estimado en la realización del mismo. Este periodo de tiempo se estima teniendo en cuenta el plazo máximo de entrega del proyecto, el 27 de Septiembre, y la disponibilidad del autor para la realización del trabajo. En este caso concreto, se ha requerido más tiempo ya que no se realizó a jornada completa.

En el diagrama de Gantt (6.1) se representan cada una de las fases definidas con el periodo de tiempo estimado. Se ha considerado el tiempo máximo disponible, 9 meses.

	ENERO				FEBRERO				MARZO				ABRIL				MAYO			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
FASE 1: PLANIFICACIÓN																				
Planteamiento del trabajo																				
Definición de requisitos																				
Estudio de sistemas de desarrollo																				
Estudio de tecnologías web																				
FASE 2: DESARROLLO																				
FASE 2.1: Backend																				
Creación de bases de datos																				
Creación script API Facebook																				
Creación crawler																				
Distribución crawler																				

	MAYO				JUNIO				JULIO				AGOSTO				SEPTIEMBRE			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Creación crawler																				
Distribución crawler																				
FASE 2.2: Frontend																				
Implementación diseño																				
Implementación usuarios																				
Implementación funcionalidades																				
FASE 3: EVALUACIÓN Y DOCUMENTACIÓN																				
Pruebas aplicación																				
Redacción de la memoria																				
Preparación de la presentación																				

Tabla 6.1: Planificación del proyecto. Diagrama de Gantt

6.2. Presupuesto

En el siguiente apartado se procede al calculo del presupuesto del proyecto realizado, detallando dos tipos de costes: costes materiales y costes de personal.

6.2.1. Costes materiales

En cuanto a costes materiales se han tenido en cuenta las siguientes variables:

- Portátil: Ordenador portátil de uso personal, marca Apple y modelo MacBook Air.
- Software: El software utilizado para la realización del proyecto es de código abierto. Sólo se tiene en cuenta la utilización del paquete Microsoft Office para la realización de la documentación del proyecto.
- Servidor: Donde alojar OpenStack.
- Material de oficina: Varios.
- Alquiler de oficina: Estudio estimado de unos 20 metros cuadrados en Madrid.
- Gastos de luz y de internet.

6.2.2. Costes de personal

En cuanto a los costes de personal, se ha tenido en cuenta las horas dedicadas por parte del tutor, considerando el precio de la hora como Ingeniero Senior, y las horas dedicadas por parte de la autora del presente proyecto, contando las horas como Ingeniero Junior.

Para estimar el precio de estas horas, según el Colegio de Ingenieros Técnicos de Telecomunicación (COITT) [5]: *"los honorarios son libres y responden al libre acuerdo entre el profesional y su cliente"*.

6.2.3. Costes totales

En la siguiente tabla (6.2) se calcula el presupuesto total del proyecto, teniendo en cuenta los costes comentados anteriormente.

CONCEPTO	CANTIDAD	COSTE	TIEMPO (meses)	DEPRECIACIÓN (meses)	TOTAL
COSTES MATERIALES					
Portatil	1	1800	9	96	168,75
Software Microsoft Office	1	120	9	/	120
Servidor	1	12	9	/	108
Material de oficina	VARIOS	30	9	/	30
Alquiler Oficina	1	250	9	/	2250
Gastos Luz	1	23,5	9	/	211,5
Gastos Internet	1	54	9	/	486
					3374,25
COSTES DE PERSONAL					
Ingeniero Junior	1	30€/h * 80h/mes	9	/	21.600 €
Ingeniero Senior	1	60€/h * 4h/mes	9	/	8.640 €
					30.240 €
IVA (21%)					7.058,99 €
COTES TOTALES					40.673 €

Tabla 6.2: Presupuesto del proyecto

Capítulo 7

Conclusiones

Se presentan a continuación las conclusiones derivadas de la realización de este trabajo fin de grado. A su vez, se detallarán posibles mejoras aplicables para trabajos futuros.

7.1. Conclusión

En este trabajo fin de grado se ha desarrollado una aplicación web basada en el análisis estadístico de las páginas de Facebook. Esta aplicación ofrece al usuario la posibilidad de elegir las páginas que desea analizar y las características del estudio a realizar. Como se tratan de datos personales, el acceso a la aplicación está protegido de forma que sólo los usuarios registrados en la aplicación podrán tener acceso. Además de que sólo podrán ver sus datos y no los de ningún otro usuario.

Gracias a la aplicación desarrollada, cualquier usuario puede conocer las características de una página de Facebook de manera automática, sin necesidad de recorrer cada una de las páginas e intuir en función de la actividad de la página cuáles son sus cualidades, o si hay algún factor que las haga distintivas respecto a otras. Se considera una aplicación pionera en ofrecer estos datos a usuarios externos, ya que, actualmente, no existe ninguna aplicación conocida que sea capaz de realizar esto con los datos relacionados de Facebook.

Se trata de una aplicación de gran utilidad para las empresas, si lo que desean es ampliar su mercado mediante las redes sociales, distinguiéndose del resto. Los datos proporcionados por la aplicación, pueden servir para aplicar tareas de marketing a raíz dichos datos. Mediante dichos datos se puede conocer cómo actúan sus mayores competidores y mejorar los aspectos que consideren oportunos.

Para la realización de las funcionalidades de esta aplicación se ha tenido que desarrollar un programa capaz de recoger y almacenar todos los datos necesarios de las páginas de Facebook a través de su API, además de la realización de un *crawler* capaz de recoger todos los datos de los usuarios de Facebook.

Destacar también un mayor grado de dificultad de este trabajo manejando dos tipos de bases de datos distintos, lo que ha supuesto un reto más. Además de tener que aplicarse diferentes tecnologías para su realización.

Concluyendo, a pesar de la complejidad de alguna de las tareas realizadas, los objetivos del presente proyecto han sido concluidos por completo. Además el trabajo realizado ha

servido de gran aprendizaje para la autora del mismo. Por lo tanto se puede concluir el trabajo con una valoración muy positiva del mismo.

7.2. Desarrollos futuros

Para concluir con la documentación de este trabajo fin de grado se proponen varias mejoras posibles para un trabajo futuro.

- **Mejorar la accesibilidad de los usuarios:** Integrar la aplicación en diferentes plataformas como *Android* e *iOS*, para permitir su accesibilidad desde dispositivos móviles.
- **Añadir más funcionalidades a la aplicación:** Añadir más gráficas al estudio, para obtener un análisis más completo. También se puede mejorar el formulario de búsqueda ofreciendo a los usuarios el resultado de las búsquedas de páginas en Facebook, eliminando el paso de comprobación del nombre y de la existencia de la sección *Post To Page*.
- **Definir mejor los resultados realizando análisis de sentimiento:** Incluir análisis de texto para identificar y extraer información subjetiva de los comentarios de los usuarios, este procesado de los datos intenta determinar la actitud de un usuario ante un suceso. Con esta mejora se podrían calificar los comentarios como positivos o negativos, pudiendo deducir si son quejas o expresan gratitud.
- **Mejorar la concurrencia del sistema:** Dividir en hilos las consultas a la API de Facebook, de forma que se reduzca el tiempo de espera del usuario.

Bibliografía

[1] Artículo sobre el reciente crecimiento de facebook. http://www.tuexperto.com/2015/02/01/facebook-se-acerca-a-los-1400-millones-de-usuarios-activos/.

[2] Bases de datos nosql. http://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf.

[3] Boe–a-1999-23750. http://www.boe.es/buscar/doc.php?id=BOE-A-1999-23750.

[4] Charts4j. https://code.google.com/p/charts4j/.

[5] Coitt: Colegio oficial de ingenieros, honorarios profe. http://www.coitt.es/res/libredocs/Honorarios.pdf.

[6] Computación aplicada al desarrollo, historia de facebook. http://www.cad.com.mx/historia_de_facebook.htm.

[7] Eclipse. http://es.wikipedia.org/wiki/Eclipse_(software).

[8] Evolución y tendendias actuales de los web crawlers. dialnet.unirioja.es/descarga/articulo/4797426.pdf.

[9] Facebook4j. http://facebook4j.org/en/api-support.html.

[10] Global web index. https://app.globalwebindex.net/products/report/15-trends-for-2015.

[11] Maven. https://maven.apache.org/.

[12] Plantilla de bootstrap formulario por pasos. http://azmind.com/2015/05/31/multi-step-registration-form-bootstrap-css3-jquery/.

[13] Plantilla de bootstrap login. http://www.bootstrapzero.com/bootstrap-template/sign-in.

[14] Proyecto cloud computing. http://informatica.gonzalonazareno.org/plataforma/course/view.php?id=47.

[15] Qué es un servidor y tipos de servidores. http://www.areatecnologia.com/informatica/servidor-y-tipos.html.

[16] Spring. http://projects.spring.io/spring-framework/.

[17] Tipos de servidores. http://www.tiposde.org/informatica/131-tipos-de-servidores/.

[18] S Anbukodi and K Muthu Manickam. Reducing web crawler overhead using mobile crawler. In *Emerging Trends in Electrical and Computer Technology (ICETECT), 2011 International Conference on*, pages 926–932. IEEE, 2011.

[19] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. Ubicrawler: A scalable fully distributed web crawler. *Software: Practice and Experience*, 34(8):711–726, 2004.

[20] Sergey Brin and Lawrence Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 56(18):3825–3833, 2012.

[21] David Eichmann. The rbse spider-balancing effective search against web load. In *Proc. 1st WWW Conf.* Citeseer, 1994.

[22] Amin Milani Fard and Martin Ester. Collaborative mining in multiple social networks data for criminal group discovery. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 4, pages 582–587. IEEE, 2009.

[23] Allan Heydon and Marc Najork. Mercator: A scalable, extensible web crawler. *World Wide Web*, 2(4):219–229, 1999.

[24] Siti Nurkhadijah Aishah Ibrahim and Ali Selamat. Scalable e-business social network using multicrawler agent. In *Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on*, pages 702–706. IEEE, 2008.

[25] Oliver A McBryan. Genvl and wwww: Tools for taming the web. In *Proceedings of the first international world wide web conference*, volume 341. Geneva, 1994.

[26] Brian Pinkerton. Finding what people want: Experiences with the webcrawler. In *Proceedings of the Second International World Wide Web Conference*, volume 94, pages 17–20. Chicago, 1994.

[27] I Ting, Hui-Ju Wu, Pei-Shan Chang, et al. Analyzing multi-source social data for extracting and mining social networks. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 4, pages 815–820. IEEE, 2009.

[28] Alvaro Videla and Jason JW Williams. *RabbitMQ in action*. Manning, 2012.

[29] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

[30] Ming-sheng Zhao, Peng Zhu, and Tian-chi He. An intelligent topic web crawler based on dtb. In *Web Information Systems and Mining (WISM), 2010 International Conference on*, volume 1, pages 84–86. IEEE, 2010.