UNIVERSIDAD
REY JUAN CARLOS

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Curso Académico 2014/2015

Trabajo Fin de Grado

FOUNBEP

Autor: Manuel Aranda Rosales

Tutor: Felipe Alonso Atienza

AGRADECIMIENTOS

Escribo este apartado como la guinda al proyecto, y no por ser el último, carece de importancia, sino todo lo contrario. Esta sección va dedicada a todas esas personas que desde el minuto 0 hasta el día de la entrega, han estado junto a mí, directa o indirectamente en la realización del proyecto.

Este proyecto ha pasado por etapas complicadas, etapas de dejadez, y etapas de trabajo muy intenso y es por ello por lo que quiero hacer hincapié en mis amigos, los cuales han estado y van a estas siempre, aunque con una manera peculiar de animar, pero es su manera. Digo peculiar, porque he tenido que oír frases como "el Escorial tardo menos en hacerse" o "el TFG no existe, son los padres". Viajes, risas, cervezas de por medio, han sido los pilares fundamentales que han ayudado a construir los cimientos de estas cuantas líneas de código. Aquellos a los que hablo, saben de sobra quienes son.

Por otra parte, en mi casa, el apoyo familiar, día tras día aguantando mis frustraciones y celebrando los avances, son los que más sufren pero estoy seguro que son los que más orgullosos están que cierre este proyecto, y esta etapa de mi vida de manera satisfactoria.

No olvidarme de mi tutor, Felipe Alonso, que desde primera instancia me guió y me aconsejó de la mejor manera, teniendo la mayor paciencia del mundo conmigo y con la templanza necesaria haciendo que recuerde de manera muy positiva al último tutor del grado.

Por todo ello, a todas personas, gracias de corazón.

Os quiero.

RESUMEN

El uso de las redes sociales está en auge, y bien sabemos todos que forman parte del día a día, de casi todos los ciudadanos. El concepto red social se extiende cada vez más, y en la actualidad van surgiendo diferentes tipos de las mismas, en las que podemos conversar, subir fotos, conocer gente, e incluso encontrar trabajo. La motivación original surge para poder aglutinar en una sola aplicación la información de todas las redes sociales y no depender así de las aplicaciones individuales.

Ahí es donde surge la motivación de este proyecto. Es el intento de unión de dos de estas redes más utilizadas. Hablamos de Twitter y LinkedIn. Esta aplicación, obtiene información resumida de un usuario, en la que se puede ver, sus aficiones, gustos, y sus últimos *tweets*, y *retweets*, mezclado todo ello con unos datos más técnicos y profesionales, en los que observaremos el empleo actual (en caso de que lo haya) y un resumen de su curriculum vitae (CV). Es decir, obtenemos en un "click" una visión general del usuario en cuestión. Todo esto contribuye al nacimiento de *Founbep*.

Founbep es una aplicación web, implementada en Python, mediante el *framework* Django, usando además, HTML, CSS y SQLite como modelo de base de datos. El motor básico de esta aplicación es el acceso a las API tanto de Twitter como de LinkedIn, donde accederemos a los datos de un usuario en concreto.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

INTRODUCCION

La finalidad de este Trabajo Fin de Grado (TFG) es la creación de una aplicación web, la cual se encargará de unificar información sobre un usuario, de datos pertenecientes a varias redes sociales. Se tendrá un híbrido entre información de gustos y aficiones de dicho usuario, y por otro lado, datos profesionales, técnicos, experiencia laboral…Todo esto se conseguirá accediendo a las APIs de LinkedIn y de Twitter, herramientas gratuitas para desarrolladores que aportan información pública y útil de los *users* registrados.

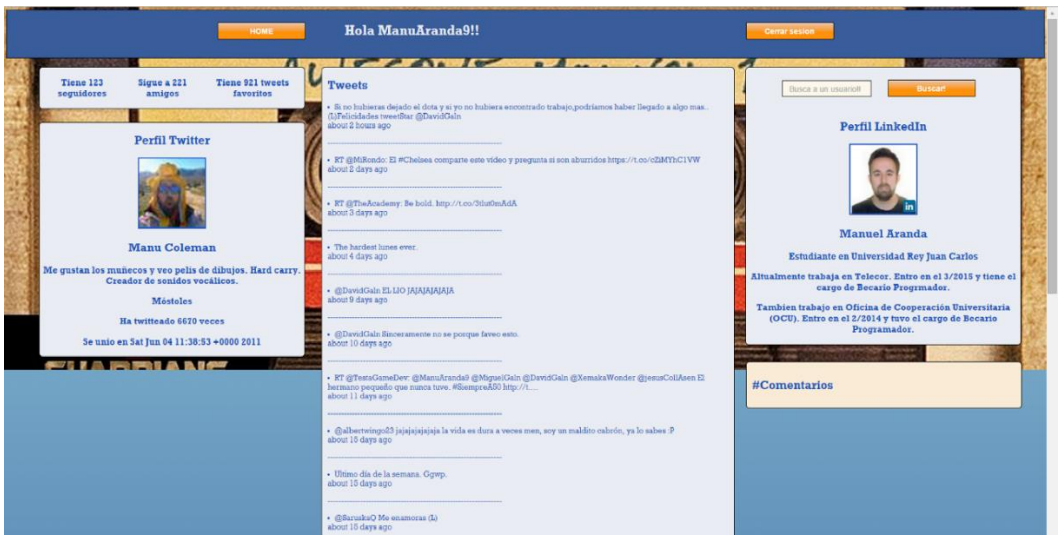

Figura1. Apariencia principal de *FounBep*.

1.1 Contexto

Aprovechando el auge de las redes sociales, y del potencial de estas herramientas para conectar mundos y personas, surge una primera idea de la creación de *FounBep.* Estas aplicaciones se han incorporado a nuestras vidas formando parte de ellas y ahora podemos compartir información de nuestras rutas en bicicleta, jugar contra nuestros amigos y superarles en puntuación, realizar compras desde casa sin ningún tipo de esfuerzo, y mil usos cotidianos que nos hacen ejercitar cuerpo y mente.

Estas redes cumplen de cierta forma la falta de roce que trae consigo el desarrollo de la vía moderna. Gracias a la facilidad que ofrecen para conocer personas y de establecer relaciones de amistad con gente que posee gustos e intereses similares a los nuestros, las redes sociales son un medio por el cual podemos estar en contacto con diversas personas alrededor del mundo. Y lo que es más importante, son fáciles de consultar y posibilitan la obtención de la información más relevante de nuestro usuario en tiempo real.

A pesar de existir gran cantidad de redes sociales, el desarrollo de aplicaciones que se encarguen de mostrarnos todos nuestros perfiles en uno solo no ha sido encauzado y es por ello, por lo que *Founbep* cobra mayor protagonismo.

1.2 Motivación

Todas estas redes sociales, tienen en común, que conectan miles de usuarios mundialmente, ya sea mediante fotos, comentarios, *likes,* pero en muchas de ellas se echa en falta un perfil más generalizado de un usuario. Se habla de general en cuanto a saber cuáles son sus aficiones, gustos, y saber por otro lado donde trabaja, cuáles son sus estudios… En el caso de Twitter, se puede conocer a un usuario por sus aficiones y gustos en cualquier ámbito social, desde política, hasta deportes, y por el contrario no se suele saber nada de ese mismo usuario en cuanto a su vida profesional. Podemos observar algún *tweet* aislado, pero con exactitud no sabrás cuál es su curriculum vitae, ni en que se especializó al salir de la carrera…cosa que por ejemplo sí podrás ver en LinkedIn.

Las redes sociales, son aplicaciones que sacian la sed del usuario en un aspecto concreto, ya sea socializarse, desconectar, buscar sitios de ocio cerca de su ciudad, encontrar trabajo…es decir hay infinidad de usos, pero a veces nos gustaría encontrar varios de esos perfiles en una misma red social. Es cierto que en alguna de ellas se puede ver algo más generalizado, como quizás en el perfil de Facebook, pero sigue careciendo de poseer más detalles.

Ahí es donde entra en acción *FounBep*, mostrando de manera clara, y muy visual, los pilares que forman al usuario, agrupando en esta plataforma toda la información de las redes sociales sin depender de aplicaciones individuales.

1.3 Estructura de la memoria

La memoria del proyecto está estructurado en los siguientes capítulos:

1. **Introducción**
   En este capítulo se llevará a cabo una introducción al contexto de las redes sociales en la sociedad actual, para tener una idea general del proyecto.

2. **Objetivos**
   Se concretará los objetivos principales que tiene que cumplir la aplicación desde que el usuario se registra.

3. **Descripción informática**
   Resumen de las tecnologías utilizadas en el proyecto, y donde se expondrán todas las herramientas y el porqué de su elección frente a otras.

4. **Pruebas**
   Este capítulo contiene las pruebas unitarias ejecutadas sobre el proyecto.

5. **Conclusiones**
   Apartado dedicado a las conclusiones que han surgido al realizar el proyecto.

6. **Trabajo Futuro**
   En este capítulo se describen las funcionalidades adicionales que podrían hacerse en versiones posteriores del proyecto, siendo factibles.

Capítulo 2

OBJETIVOS

2.1 Descripción del problema

Se creará una aplicación web, que permitirá al usuario registrarse con sus credenciales de Twitter y de LinkedIn, para más adelante, mostrar la información obtenida de manera resumida, y sencilla en pantalla. La aplicación adicionalmente permitirá a dicho usuario buscar otros perfiles en la aplicación, los cuales se encuentren registrados también.

Se añade más adelante la funcionalidad extra de poder dejar comentarios en los distintos perfiles, haciendo de *Founbep* algo más parecido a una red social, que una aplicación meramente de visualización.

2.2 Objetivos

El objetivo principal de *FounBep* es presentar de manera intuituva y clara la información del usuario. Esta herramienta sigue siendo un ejercicio sencillo y demostrativo, pero la idea de poseer información de alguien de manera más completa es la raíz principal en la que se basa este proyecto.

2.3 Estudio de alternativas

Para el desarrollo de una aplicación web se necesita la elección de un lenguaje principal de programación, un *framework* el cual ayudará a la implantación, y un gestor de base de datos, por lo que en este apartado se meterá de lleno en la comparación de varios de los mismos, y el porqué de su elección final.

2.3.1 Lenguajes de programación

2.3.1.1 Java

La principal característica de Java es la de ser un lenguaje compilado e interpretado. Todo programa en Java ha de compilarse y el código que se genera, es interpretado por una máquina virtual. De este modo se consigue la independencia de la máquina, el código compilado se ejecuta en máquinas virtuales que si son independientes de la plataforma.

Java es un lenguaje orientado a objetos, de propósito general. Aunque Java comenzara a ser conocido como un lenguaje de programación de *applets* que se ejecutan en el entorno de un navegador web, se puede utilizar para construir cualquier tipo de proyecto. [1]

Figura 2. Java - Spring

2.3.1.2 Python

Es un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa, y en menor medida programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Su filosofía reside en que su sintaxis favorezca a un código legible.

Es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel. Es un lenguaje ideal para el scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas. [2]

Figura 3. Python – Django

2.3.1.3 PHP

PHP es un lenguaje de programación de uso general de código del lado servidor originalmente diseñado para el desarrollo de contenido web dinámico. Fue uno de los primeros lenguajes de programación del lado servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos.

PHP se utiliza para generar páginas web dinámicas. Recordar que llamamos página estática a aquella cuyos contenidos permanecen siempre igual, mientras que llamamos páginas dinámicas a aquellas cuyo contenido no es el mismo siempre. Por ejemplo, los contenidos pueden cambiar en base a los cambios que haya en una base de datos, de búsquedas o aportaciones de los usuarios, etc [3]

Figura 4. PHP – Symfony2

2.3.1.4 Framework Django

Django es un *framework* web de código abierto escrito en Python que permite construir aplicaciones web más rápido y con menos código.

Django fue inicialmente desarrollado para gestionar aplicaciones web de páginas orientadas a noticias de World Online, más tarde se liberó bajo licencia BSD. Django se centra en automatizar todo lo posible y se adhiere al principio DRY (*Don't Repeat Yourself*).

También se adscribe al diseño *MVC* (Modelo-Vista-Controlador), por lo que las diferentes partes del sitio están claramente separadas. Por ejemplo, el código de acceso a los datos es completamente independiente al que gobierna el aspecto externo de la página.

Al tratarse de Python, Django permite que el desarrollador escriba código ágilmente. El resultado son menos líneas de código y, en consecuencia, menos probabilidades de que haya bugs. De ahí que digan que "fomenta el desarrollo rápido y el diseño limpio y pragmático". [4]

2.3.1.5 HTML

HTML es un lenguaje de programación que se utiliza para el desarrollo de páginas de Internet. Se trata de la sigla que corresponde a *HyperText Markup Language*, es decir, Lenguaje de Marcas de Hipertexto, que podría ser traducido como Lenguaje de Formato de Documentos. Este lenguaje es el que se utiliza para especificar los nombres de las etiquetas que se utilizarán al ordenar, no existen reglas para dicha organización, por eso se dice que es un sistema de formato abierto.

EL HTML se encarga de desarrollar una descripción sobre los contenidos que aparecen como textos y sobre su estructura, complementando dicho texto con diversos objetos (como fotografías, animaciones, etc).

Es un lenguaje muy simple y general que sirve para definir otros lenguajes que tienen que ver con el formato de los documentos. El texto en él se crea a partir de etiquetas, también llamadas tags, que permiten interconectar diversos conceptos y formatos. [5]

Figura 5. Logo HTML 5

2.3.1.6 CSS

Al igual que HTML nos permite definir la estructura de nuestra página web, CSS es el encargado de establecer el diseño y estilo de la página en los diferentes dispositivos. Por tanto, las hojas de estilo nos permiten definir la representación de la página web de manera eficiente logrando separar el contenido de la forma.

El lenguaje CSS se basa en una serie de reglas que rigen el estilo de los elementos en los documentos estructurados, y que forman la sintaxis de las hojas de estilo. Cada regla consiste en un selector y una declaración, esta última va entre corchetes y consiste en una propiedad o atributo, y un valor separados por dos puntos.

Ejemplo:

h2 {color: green;}

h2 ---> es el selector

{color: green;} ---> es la declaración

color ---> es la propiedad o atributo

green ---> es el valor

El *Selector* especifica que elementos HTML van a estar afectados por esa declaración, de manera que hace de enlace entre la estructura del documento y la regla estilística en la hoja de estilo.

La *Declaración* que va entre corchetes es la información de estilo que indica cómo se va a ver el selector. En caso de que haya más de una declaración se usa punto y coma para separarlas.

El uso de CSS permite conseguir efectos visuales que antes solo eran posibles con el uso de tecnologías adicionales. Además ahorramos tiempo y trabajo al poder seguir varias técnicas (bordes redondeados, sobras en el texto, sobras en las cajas, gradientes…) sin necesidad de usar un editor gráfico. [6]

Figura 6. Logo CSS 3

Cualquiera de los lenguajes anteriores, habría servido perfectamente para el desarrollo de nuestro proyecto, pero al final nos decantamos por el lado de Python. Python es un lenguaje muy sencillo de comprender, y tiene una curva de aprendizaje muy suave e implementa gran cantidad de bibliotecas adicionales, características importantes para la final elección del mismo. Adicionalmente, hubo un trabajo de investigación respecto al acceso a las Apis, tanto de Twitter como de LinkedIn.

En el primer caso, hay librerías en casi todos los lenguajes principales, pero en el caso de LinkedIn, Python contaba con más documentación, foros, y centros de ayuda, con los cuales podría resolver los futuros problemas que surgieran, y pensando en el futuro, debido a la poca o nula experiencia en Python, se valoró el gran contenido de apoyo y documentación como algo fundamental.

2.3.2 Bases de datos

2.3.2.1 SQLite

Django soporta de manera predeterminada la conexión con *postgresql*, *mysql*, *sqlite3* y *oracle*. En nuestro proyecto usaremos sqlite3.

Con toda esta filosofía en mente, vamos a comenzar a explorar la capa de la base de datos de Django. Primero, necesitamos tener en cuenta algunas configuraciones iniciales: necesitamos indicarle a Django qué servidor de base de datos usar y cómo conectarse con el mismo.

Asumimos que se ha configurado un servidor de base de datos, se ha activado, y se ha creado una base de datos en este (por ej. usando la sentencia *CREATE DATABASE*). *SQLite* es un caso especial; es este caso, no hay que crear una base de datos, porque *SQLite* usa un archivo autónomo sobre el sistema de archivos para guardar los datos.

SQLite merece especial atención como herramienta de desarrollo. Es un motor de base de datos extremadamente simple y no requiere ningún tipo de instalación y configuración del servidor. Es por lejos el más fácil de configurar si sólo quieres jugar con Django, y viene incluido en la librería estándar de Python 2.5. [7]

Configuración	Base de datos	Adaptador requerido
`postgresql`	PostgreSQL	psycopg versión 1.x
`postgresql_psycopg2`	PostgreSQL	psycopg versión 2.x
`mysql`	MySQL	MySQLdb
`sqlite3`	SQLite	No necesita adaptador si se usa Python 2.5+. En caso contrario, pysqlite
`ado_mssql`	Microsoft SQL Server	adodbapi version 2.0.1+
`oracle`	Oracle	cx_Oracle

Figura 7. Diferentes tipos de BD que se pueden conectar a Django.

2.4 Metodología empleada

Para la creación de nuestro proyecto, seguiremos el Proceso Unificado de Desarrollo Software, un marco de desarrollo guiado por casos de uso, centrado en la arquitectura y con un ciclo de vida iterativo e incremental.

Está compuesto de cuatro fases denominadas *Inicio, Elaboración, Construcción y Transición*. Cada una de estas fases es a su vez dividida en una serie de iteraciones. [8]

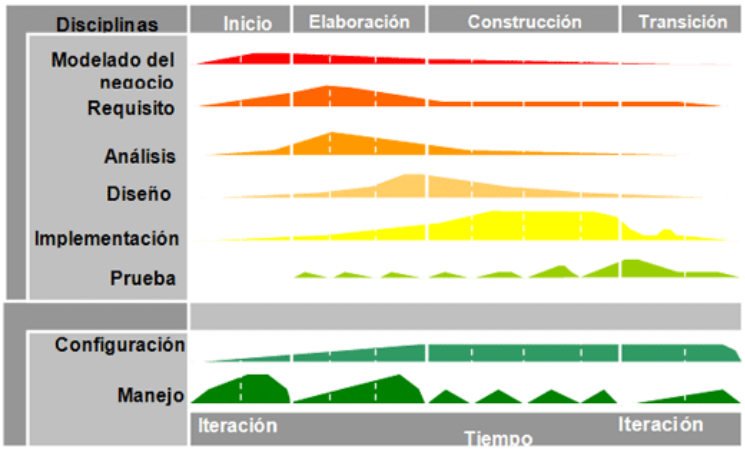

Figura 8. Proceso Unificado de Desarrollo SW

En cada iteración se llevaran a cabo las siguientes etapas:

- Análisis de requisitos

  En esta fase se describen los objetivos y necesidades del sistema y poner a los desarrolladores y al cliente de acuerdo en esta descripción.

- Análisis y diseño

  Describe como el SW será realizado en la fase de implementación. Se plasma en un modelo de diseño que consiste en una serie de clases con interfaces bien definidos.

- Implementación

  En esta etapa se realiza la codificación del sistema siguiendo todos los objetivos y documentos creados en las etapas anteriores.

- Pruebas

  Se comprueba que el funcionamiento es correcto analizando diversos aspectos: los objetos como unidades, la integración entre objetos, la implementación de todos los requisitos, entre otros.

- Evaluación

  Se evalúan la facilidad de manejo y la utilidad de la interfaz a través de pruebas con varios usuarios.

Capítulo 3

DESCRIPCIÓN INFORMÁTICA

3.1 Captura de requisitos

En este apartado se llevara a cabo la definición de requisitos de SW del sistema. Tiene por objeto averiguar qué se debe construir. Ello implica que el cliente y los desarrolladores tienen que tener claro lo que debe y no debe hacer el sistema.

El propósito de esta especificación es ayudar a tener una idea general del producto y ofrecer una guía para pasos posteriores. Además, servirá para ayudar a entender las necesidades y funcionalidades de SW de cara a la implementación.

3.1.1 Requisitos funcionales

Los requisitos funcionales describen los requisitos a tener en cuenta en cuanto a la funcionalidad del sistema. Será necesario que el sistema satisfaga estos requisitos.

Mediante casos de uso. El caso de uso es una funcionalidad que proporciona el sistema. Para el usuario, el caso de uso representa la forma de usar el sistema. También se especificara la forma de la interfaz de usuario para la ejecución de los casos de uso.

Lista de requisitos funcionales

RF1 – El usuario podrá darse de alta en la aplicación. Para ello tendrá que dejar su Twitter y su URL publica de LinkedIn para poder enlazar con sendas *apis.*

RF2 – El usuario una vez registrado, podrá autenticarse en la herramienta, con su *user* y su *password.*

RF3 – El usuario podrá ver información de su perfil.

RF4 – El usuario podrá buscar en el sistema a otros usuarios ya registrados anteriormente para ver sus perfiles.

RF5 – El usuario podrá dejar comentarios en los demás perfiles.

RF6 – El usuario podrá volver a su perfil.

RF7 – El usuario podrá cerrar sesión y volver a la pantalla de login/registro.

3.1.2 Requisitos no funcionales

Especifican propiedades del sistema como restricciones, rendimiento, dependencias de la plataforma, facilidad de mantenimiento, fiabilidad, etc.

Lista de requisitos no funcionales

RNF 1 – Nuestro sistema tendrá una interfaz intuitiva y sencilla, llegando así a un mayor número de usuarios. (Principios Interacción Persona-Ordenador)

RNF 2 – El sistema deberá responder en tiempo real a las búsquedas del usuario, evitando respuestas lentas.

RNF 3 – El sistema podrá usarse desde cualquier ordenador y cualquier sistema operativo.

3.2 Casos de uso

Los diagramas de casos de uso representan las diferentes operaciones que debe realizar el sistema, y la manera de actuar con su entorno, más concretamente, los usuarios y otras aplicaciones que interactúan con él.

Cada manera en que los usuarios emplean el sistema representara un caso de uso y uniendo todos los casos tendremos la especificación total del sistema.

3.2.1 Diagrama de casos de uso

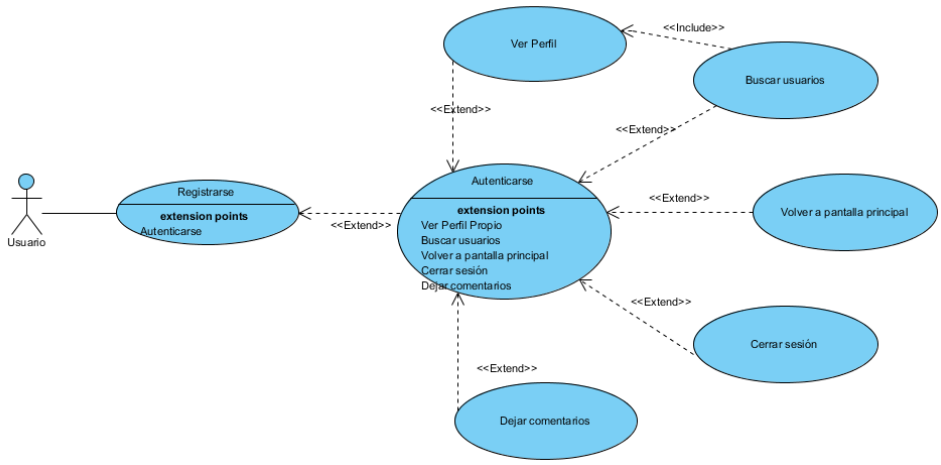

Figura 9. Diagrama de casos de uso.

3.2.2 Descripción de los casos de uso

- Caso de uso: RF1- **"Darse de alta"**

  - Objetivo: Dar de alta al usuario en la herramienta.
  - Actores: Nuevo usuario de la aplicación.
  - Precondición: La única precondición para registrarse es que el usuario acceda a la herramienta desde un navegador web.
  - Descripción: El usuario introduce sus datos (Twitter, LinkedIn, correo, password) y quedan almacenados para el posterior acceso a la aplicación.

USUARIO	SISTEMA
1. El usuario accede a la app y decide registrarse, con lo que introduce sus datos. Una vez completado el formulario, pulsa "Registrar".	2. El sistema corrobora que dicho *user* no está duplicado en la BBDD, y que todos los datos son correctos. OK.
	3. La aplicación registra dichos datos en la BBDD, y avisa al usuario con un mensaje por pantalla de que todo ha ido correctamente.

Tabla 1. Tabla Caso de uso "Darse de Alta".

-Casos alternativos: 3. La aplicación ve que alguno de los datos introducidos son erróneos, y avisa al usuario por pantalla de que compruebe los mismos.

- Caso de uso: RF2- **"Autenticarse"**

- Objetivo: *Loguear* a un usuario en la aplicación.
- Actores: Usuario de la aplicación.
- Precondición: El usuario debe estar registrado en la herramienta.
- Descripción: El usuario introduce su usuario y contraseña para poder acceder al sistema.

USUARIO	SISTEMA
1. El usuario una vez registrado en la app, introduce sus datos y pulsa el botón "LogIn"	2. El sistema comprueba que esos datos son correctos y que corresponden con un *user* registrado anteriormente en la app.
	3. Una vez realizada dicha comprobación, el sistema redirige a la pantalla principal de la aplicación.

Tabla 2. Tabla Caso de uso "Autenticarse".

- Casos alternativos: 2. La aplicación no encuentra coincidencias para los datos introducidos, con lo avisa al usuario por pantalla del error.

- Caso de uso: RF3- **"Ver perfil"**

- Objetivo: El usuario visualiza un perfil, ya sea suyo o de otra persona.
- Actores: Usuario de la aplicación.
- Precondición: El usuario debe *loguearse* con anterioridad para que la aplicación le redirija a su página.
- Descripción: El usuario podrá visualizar los datos en pantalla correspondientes al perfil propio o de otro usuario.

USUARIO	SISTEMA
1. El usuario autenticado es redirigido a la página principal de la aplicación.	2. El sistema se encarga de mostrar los datos que se han obtenido.
3. El usuario visualiza los datos mostrados.	

Tabla 3. Tabla Caso de uso "Ver Perfil".

- Casos alternativos: 2. La aplicación no puede obtener ningún dato respecto al usuario registrado con lo que no podrán visualizarse dicho perfil.

- Caso de uso: RF4- **"Buscar perfil"**

- Objetivo: El usuario busca a otra persona registrada en la aplicación para visualizar su perfil
- Actores: Usuario de la aplicación.
- Precondición: El usuario debe *loguearse* con anterioridad para que la aplicación le redirija a su página.
- Descripción: El usuario introduce en el buscador un nombre de otro *user* para que el sistema muestre su perfil.

USUARIO	SISTEMA
1. El usuario autenticado es redirigido a la página principal de la aplicación.	
2. El usuario introduce en el buscador el nombre de otro usuario.	3. El sistema comprueba que dicho usuario existe.
	4. La herramienta redirige a la página principal del usuario buscado y muestra su perfil.

Tabla 4. Tabla Caso de uso "Buscar Perfil".

- Casos alternativos: 3. La aplicación no puede obtener ningún usuario de la barra de búsqueda, porque los datos introducidos no coinciden con la BBDD. Se muestra un mensaje de error por pantalla.

- Caso de uso: RF5- **"Escribir comentario"**

- Objetivo: El usuario deja un comentario en el perfil de otro usuario.
- Actores: Usuario de la aplicación.
- Precondición: El usuario debe buscar en la aplicación a otro usuario para poder dirigirse a su perfil.
- Descripción: El usuario una vez redirigido al perfil ajeno, introduce un comentario en la caja de texto, y una vez terminado, pulsa "Enviar".

USUARIO	SISTEMA
1. El usuario autenticado es redirigido a la página principal de la aplicación.	
2. El usuario introduce en el buscador el nombre de otro usuario.	3. El sistema comprueba que dicho usuario existe.
	4. La herramienta redirige a la página principal del usuario buscado y muestra su perfil.
	5. El usuario introduce un comentario en la caja de texto que aparece a la derecha y la envía.
	6. Se actualiza el listado de comentarios y se muestran.

Tabla 5. Tabla Caso de uso "Escribir comentario".

- Caso de uso: RF6- **"Volver a pantalla principal"**

- Objetivo: El usuario regresa a la pantalla principal donde se muestra su perfil.
- Actores: Usuario de la aplicación.
- Precondición: El usuario debe *loguearse* con anterioridad para que la aplicación le redirija a su página.

- Descripción: El usuario una vez autenticado en la aplicación, puede volver a su perfil principal pulsando el botón "HOME". Puede hacerlo en cualquier parte de la aplicación.

USUARIO	SISTEMA
1. El usuario autenticado es redirigido a la página principal de la aplicación.	2. El sistema se encarga de mostrar los datos que se han obtenido.
3. El usuario presiona el botón "Home" y es redirigido a su perfil nuevamente.	

Tabla 6. Tabla Caso de uso "Volver a pantalla principal".

- Casos alternativos: 3. El usuario en lugar de presionar dicho botón, busca en la aplicación otros usuarios y se muestra su perfil. Vuelva al paso 3.

- Caso de uso: RF7- **"Cierre de sesión "**

- Objetivo: El usuario cierra su sesión y vuelve a la pantalla de "*login/signup*"
- Actores: Usuario de la aplicación.
- Precondición: El usuario debe *loguearse* con anterioridad para que la aplicación le redirija a su página.
- Descripción: El usuario una vez autenticado en la aplicación, puede cerrar sesión donde se volverá a la pantalla de carga inicial. Vuelta al RF1.

USUARIO	SISTEMA
1. El usuario autenticado es redirigido a la página principal de la aplicación.	2. El sistema se encarga de mostrar los datos que se han obtenido.
3. El usuario presiona el botón "Cierre sesion" y es redirigido a la pantalla inicial.	

Tabla 7. Tabla Caso de uso "Cierre de sesión".

3.3 Arquitectura de alto nivel.

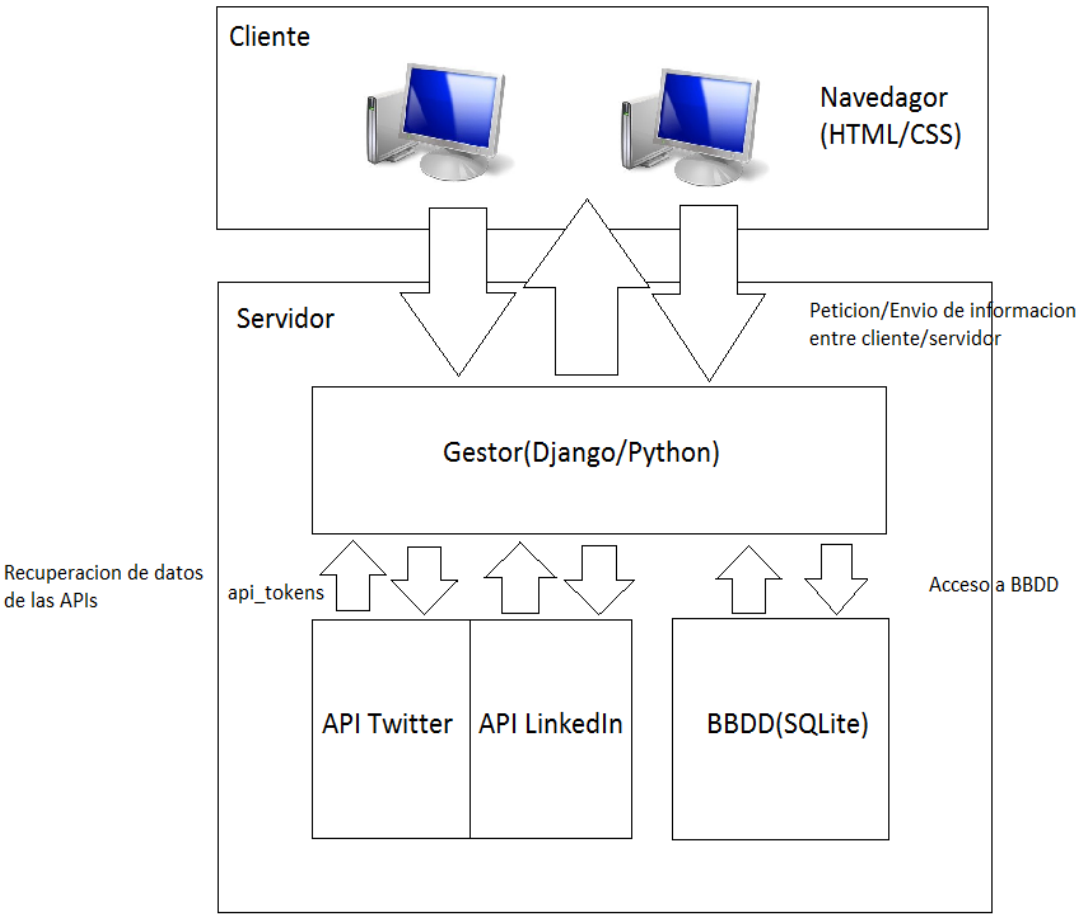

Figura 10. Diagrama alto nivel.

- Capa de cliente: En la que se encuentra nuestro navegador cliente, que envía y acepta peticiones HTTP a la capa servidor. Esta capa debe interpretar el código HTML y las hojas de estilos (CSS). Nuestros formularios y páginas se mostraran aquí.

- Capa servidor: Esta capa se encarga de recibir los datos de los formularios HTML para procesarlos y enviar peticiones a las Apis de Twitter y de LinkedIn. Por otra parte, se encarga de almacenar usuarios y sus comentarios en nuestra base de datos. Todo esto bajo la tutela de Django, y SQLite que es nuestro gestor de bases de datos.

Para definir mejor la arquitectura del sistema, la Figura 11 muestra el diagrama de alto nivel según los módulos de la aplicación.

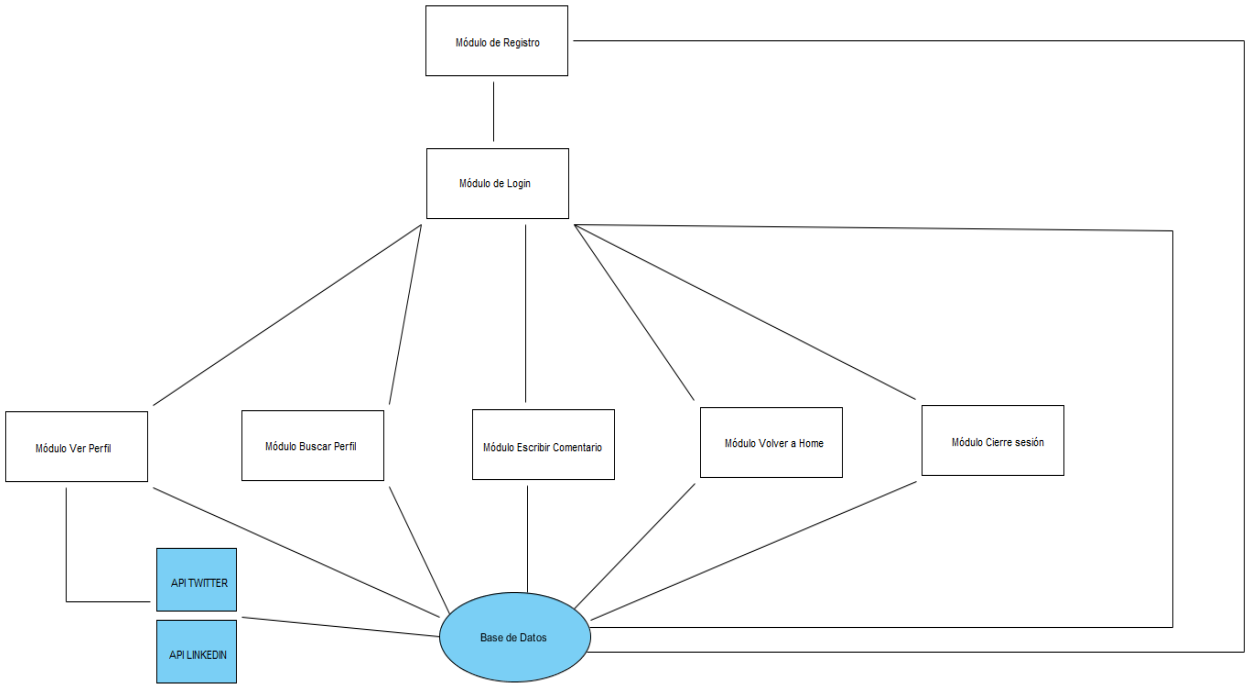

Figura 11. Diagrama alto nivel según módulos.

-Módulo "Registro": módulo encargado de almacenar nuestro nuevo usuario en la base de datos.

-Módulo "Login": módulo encargado de conectar al usuario previamente registrado, con nuestra aplicación, para posteriormente acceder con dichos datos a las apis.

-Módulo "Ver Perfil": módulo encargado de conectar al usuario previamente registrado, con nuestra aplicación, para posteriormente acceder con dichos datos a las apis.

-Módulo "Buscar Perfil": módulo encargado de buscar en nuestra BBDD coincidencias con el *user*, que metemos en el buscador, y en caso de encontrarlo, mostrar su perfil.

-Módulo "Escribir Comentario": módulo encargado de capturar los comentarios de los usuarios y almacenarlos en la base de datos.

-Módulo "Volver a Home": Módulo encargado de volver a la pantalla principal, en la cual se muestra el perfil del usuario *logueado* actualmente.

-Módulo "Cerrar Sesión": Módulo encargado de cerrar la sesión del usuario activo para que otros usuarios no puedan acceder a su información.

Todos los módulos se conectan con la base de datos, ya que necesitan mostrar, consultar, y comprobar que los datos introducidos en los distintos formularios de la app, corresponden con datos.

3.4 Diagrama E-R.

Para comprender la estructura de la BBDD, mostraremos en la figura 12, el diagrama entidad-relación correspondiente a la aplicación, y posteriormente explicaremos cada una de sus partes.

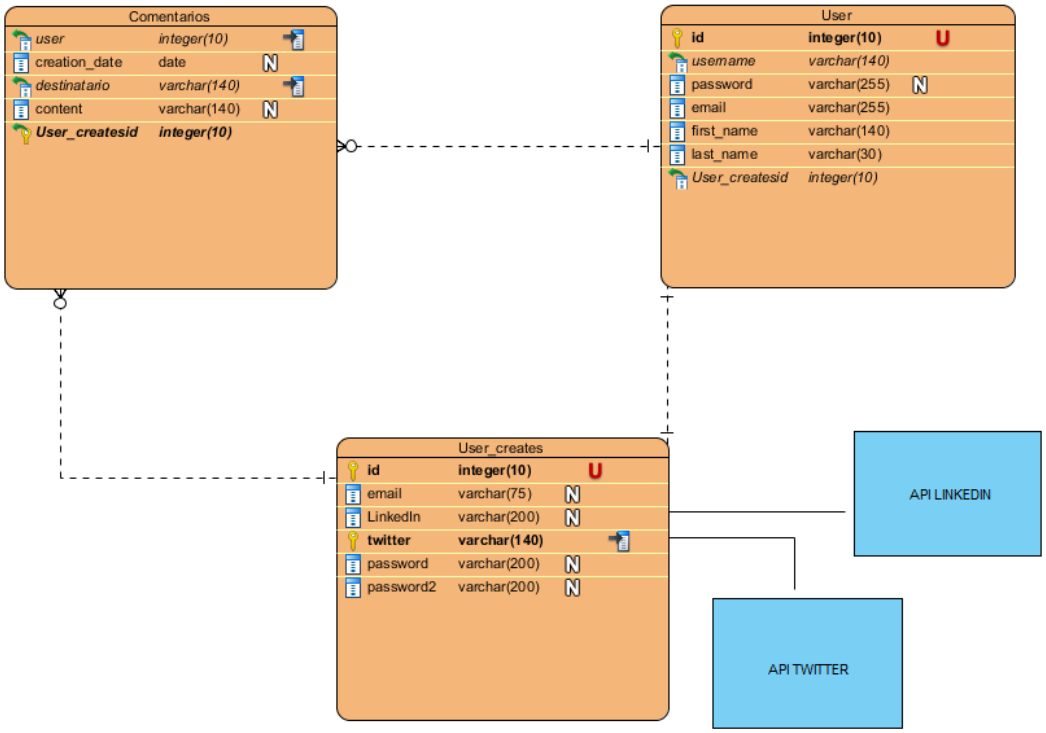

Figura 12. Diagrama ER.

En el modelo anterior podemos observar 3 entidades distintas. Por un lado, tenemos la entidad "*user*", la cual proporciona Django. Se utiliza para facilitar la autenticación al sistema, para asociar contenido al propietario, para registros de perfiles… Sus principales atributos son los mostrados en la vista de su entidad.

Ya que Django nos proporciona esta funcionalidad, no la desperdiciaremos y usaremos esta tabla para registrar al usuario y manejarle por todo el sistema, sin tener que implementar funcionalidad adicional.

La entidad "*User_Creates*" es una variante de la tabla anterior, en la cual añadimos los campos "LinkedIn" y "twitter" que serán los que usaremos para conectarnos a las APIs. Cuando el usuario se registre por primera vez, crearemos una instancia para cada una de estas tablas. Por un lado tendremos un objeto "*user*" creado de esta manera:

```
username = formulario.cleaned_data['username']
password = formulario.cleaned_data['password2']
email = formulario.cleaned_data['email']
user = User.objects.create_user(username, email, password)
```

Figura 13. Código crear "user".

Y por otro lado, la otra instancia se forma con los datos recuperados del formulario principal de registro:

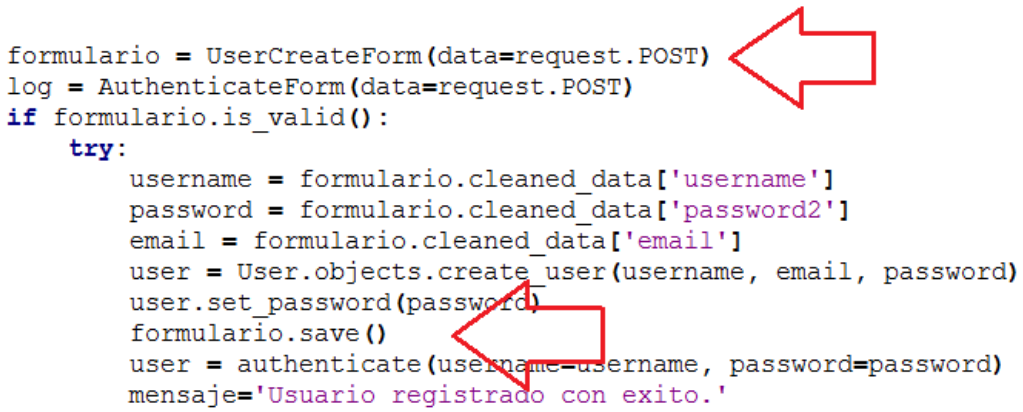

Figura 14. Código crear "User_Create".

Por otro lado tenemos la tabla "*comentarios*" en la cual un usuario podrá tener varios de estos, dirigidos hacia otros "*destinatarios*" que serán a su vez, otros usuarios.

Como se puede observar, es un modelo de base de datos sencillo, el cual solo se encarga de gestionar los usuarios, y sus comentarios hacia otros de los mismos. Esto se debe a que el acceso a la información no se almacena en nuestra BBDD, si no que se recupera de manera íntegra de las apis, para mostrarla directamente.

Descripción y ejemplos de las tablas.

Las siguientes capturas de pantalla están tomadas desde el sitio "Django/admin" reservado para *superusuarios*, en el que vemos la arquitectura de la base de datos.

- Tabla User.

  Esta tabla es la sumistrada por Django, como hemos dicho anteriormente, en la que se pueden ver los campos "*Username*", "*email*", "*first_name*" y "*last_name*". El campo "contraseña" se encuentra oculto, pero se puede ver entrando en cada uno de los usuarios.

Nombre de usuario	Dirección de correo electrónico	Nombre	Apellidos	Es staff
alumnoURJC	mj.arandar@alumnos.urjc.es			
miguelgaln	miguel.galan@gmail.com			
ManuAranda9	manu.aranda.9@gmail.com			

Figura 15. Vista de Admin/Django para "user".

- Tabla User_creates.

Esta tabla tiene nuestra versión personalizada del usuario del sistema, que accederá a las apis. Vemos como el campo "twitter" de esta tabla, coincide con el campo "*username*" de la tabla anterior.

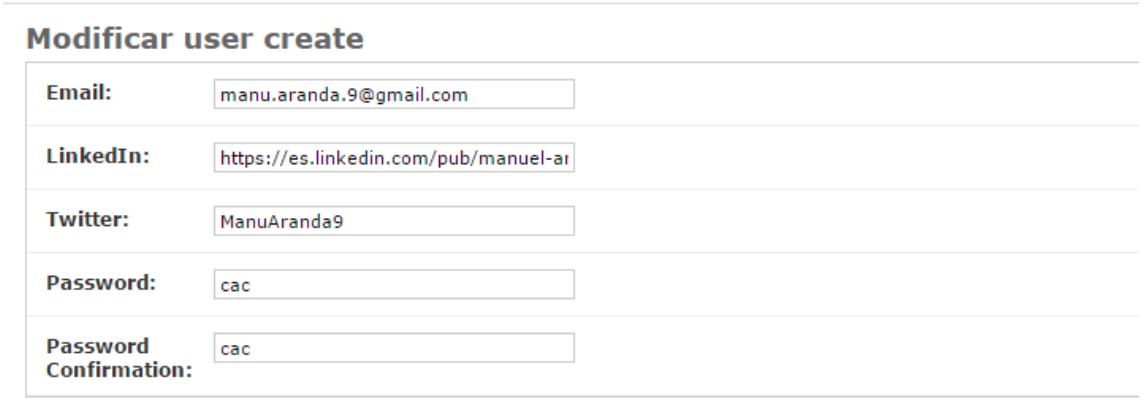

Figura 16. Vista de Admin/Django para "User_create".

- Tabla Comentario

Podemos observar en la figura X los campos que forman esta tabla. Anotar, que el campo "*creation_date*" no aparece ya que es una propiedad que no se puede gestionar desde admin. En la figura +1 comprobamos que el campo User (*foreign_key*), es un desplegable que referencia a todos los usuarios registrados en la aplicación.

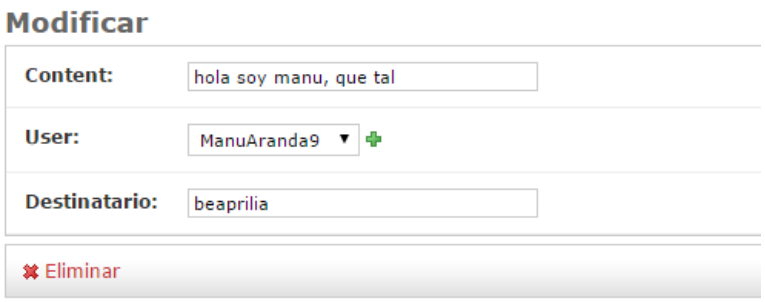

Figura 17. Vista de Admin/Django para "Comentario" (1).

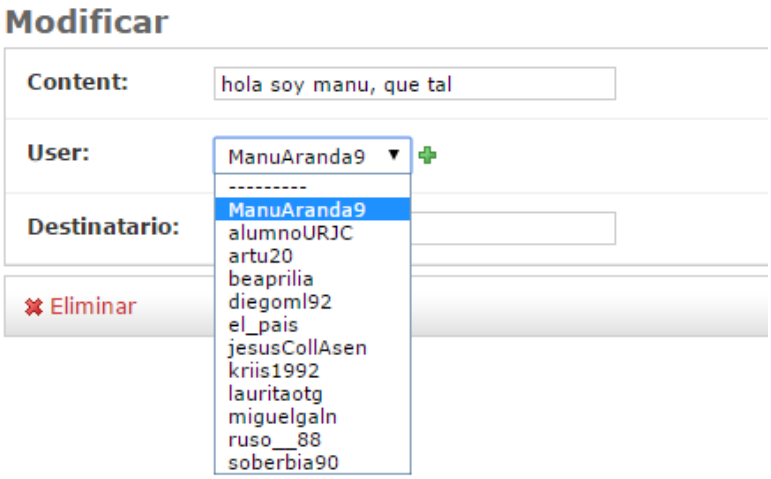

Figura 18. Vista de Admin/Django para “Comentario” (2).

3.5 Algoritmo utilizado

En este apartado realizaremos la descripción de los algoritmos utilizados para implementar cada uno de los requisitos funcionales de la aplicación.

Mostrar datos en pantalla de autenticación;

Mostrar datos en pantalla de registro;

Si usuario quiere registrarse {

Usuario introduce datos en formulario de registro;

Si los datos introducidos son correctos {

Mostrar mensaje de éxito;

Guardar datos en la BD;

} Si no {

Mostrar mensaje de error;

}

Mostrar datos en pantalla de autenticación;

Mostrar datos en pantalla de registro;

}

```
//una vez el usuario se haya registrado, podrá logearse
Si usuario quiere loguearse {
        Usuario introduce datos en formulario de login;
        Si los datos introducidos coinciden en la BD con un usuario registrado {
            mostrarPerfil (usuario);
        } Si no {
            Mostrar mensaje de error;
        }
}

// Función que se encarga de mostrar un perfil, dado un usuario
Función mostrarPerfil (usuario) {
        Redirección a pantalla principal;
        Acceso al api de twitter;
        Acceso al api de linkedin;
        Se muestra la información de ese usuario en twitter;
        Se muestra la información de ese usuario en linkedin;
        Mostrar comentarios;
        Si usuario!= usuarioLogueado{
            Mostrar caja para introducir comentario;
        }
        Mostrar botón "volver a pantalla principal";
        Mostrar botón "cierre sesión";
        Mostrar buscador de usuarios;

        Si usuario quiere volver a pantalla principal {
            Pulsa botón Home;
            Redirección a pantalla principal;
        }
```

Si usuario quiere cerrar sesión {

Pulsa botón Cerrar Sesión;

Mostrar datos en pantalla de autenticación;

Mostrar datos en pantalla de registro;

}

Si usuario quiere buscar {

Introduce un usuario en el buscador;

Si usuario existe {

mostrarPerfil (usuario);

} Si no {

Mostrar mensaje de error;

}

}

3.6 Descripción de los módulos.

En esta sección, se explican los principales módulos de los que se compone foundbep:

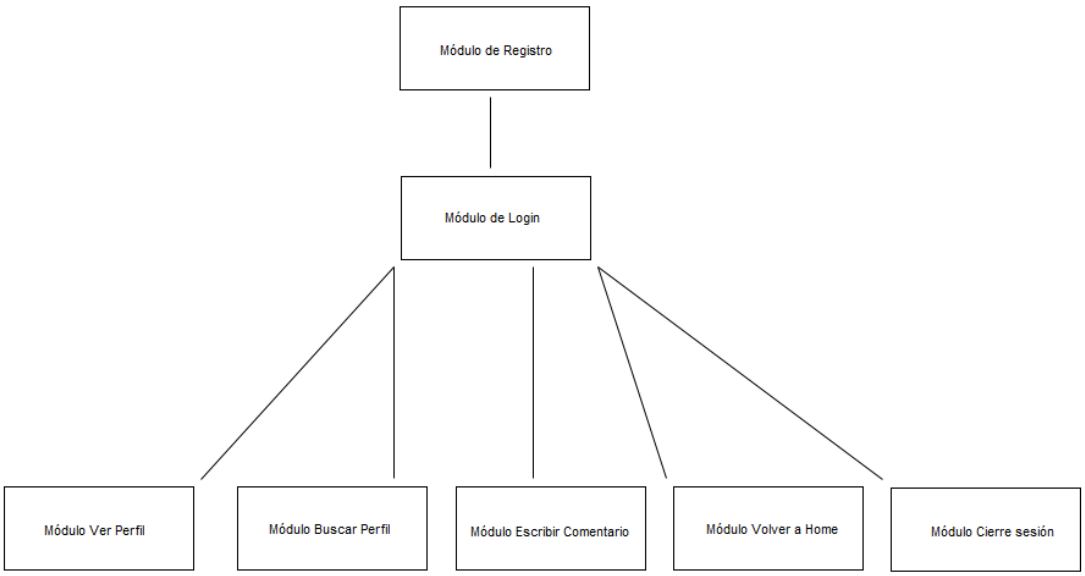

Figura 19. Diagrama de Módulos.

- Módulo de Registro.

En este módulo, el cual se encuentra dentro de **views.py,** se implementa dentro de la función **nuevo_usuario.** Su funcionalidad reside en registrar al usuario en el sistema. El siguiente pseudocódigo muestra la estructura del módulo.

Recuperar datos del formulario de registro;

Si los datos son válidos {

Intentar {

Crear objeto de la tabla "user";

Crear objeto de la tabla "user_create";

Guardar datos en BD;

Mostrar mensaje de éxito;

} Excepción {

Mostrar mensaje de error;

}

}

- Módulo Login.

En este módulo, el cual se encuentra dentro de **views.py,** se implementa dentro de la función **login_view.** Su funcionalidad reside en autenticar al usuario en el sistema una vez registrado. El siguiente pseudocódigo muestra la estructura del módulo.

Recuperar datos del formulario de autenticación;

Si los datos son válidos {

Intentar {

Acceso con "user" y "password";

Recupera los comentarios que tiene ese "user";

} Excepción {

Mostrar mensaje de error;

Acceso = error;

}

```
        Si acceso = error {
            Autenticación (acceso);
        } Si no {
            Mensaje de error;
        }
    }
```

- Módulo Ver Perfil

En este módulo, el cual se encuentra dentro de **views.py,** se implementa dentro de las funciones **getTweets()** y **getBiografia().** Su funcionalidad es la de acceder a las distintas apis, y con los datos proporcionados, recuperar información para mostrarla. El siguiente pseudocódigo muestra la estructura del módulo.

```
//función getTweets()
    Intentar {
        Importar Api Twitter;
        Obtener datos de la api(nombre, descripción, timeline, ...);

    } Excepción {
        Mostrar error;
    }

//----------------------------------------------------------------
//función getBiografia()
    Intentar {
         Importar Api LinkedIn;
        Obtener datos de la api (nombre, centro de estudios, CV, ...);

    } Excepción {
        Mostrar error;
    }
```

- Módulo Buscar Perfil.

En este módulo, el cual se encuentra dentro de **views.py,** se implementa dentro de la función **buscar ().** Su funcionalidad reside en buscar en nuestra BD coincidencias con el usuario introducido por el buscador. El siguiente pseudocódigo muestra la estructura del módulo.

Recuperar datos del buscador;

Si los datos son válidos {

Recupera los comentarios que tiene ese "user";

Para "user" recupera el LinkedIn asociado en la tabla "User_Create";

getTweets();

getBiografía();

}

- Módulo Escribir Comentario.

En este módulo, el cual se encuentra dentro de **views.py,** se implementa dentro de la función **submit ().** Su funcionalidad reside en escribir un comentario y guardarlo en la BD con la información de quien lo escribe, y para quien va destinado. El siguiente pseudocódigo muestra la estructura del módulo.

Recuperar datos del formulario;

Si los datos son válidos {

Recupera el usuario que envía dicho comentario;

Recupera el usuario al que va destinado el comentario;

Guarda toda la información en una objeto instanciado de la tabla "Comentario"

}

- Módulo volver a "Home"

En este módulo, el cual se encuentra dentro de **views.py,** se implementa dentro de la función **cargarPantalla ().** Su funcionalidad reside en cargar la información del usuario autenticado en la app y mostrar su perfil. El siguiente pseudocódigo muestra la estructura del módulo.

Si usuario quiere regresar a la pantalla principal {

Recupera información del usuario actualmente logueado en la app;

getTweets ();

getBiografía ();

}

- Módulo Cerrar Sesión.

En este módulo, el cual se encuentra dentro de **views.py,** se implementa dentro de la función **logout_view ().** Su funcionalidad reside en cerrar la sesión actual y volver a la pantalla de login/registro. El siguiente pseudocódigo muestra la estructura del módulo.

Si usuario quiere cerrar sesión {

Logout();

Re direccionar a página inicial;

}

Capítulo 4

PRUEBAS

Para comprobar el correcto funcionamiento de la aplicación se han realizado distintos tipos de pruebas que se describen a continuación.

4.1 Pruebas Unitarias

Con este tipo de pruebas se comprueba el correcto funcionamiento de cada módulo de forma independiente. Se distingue entre pruebas de **caja blanca** que son aquellas que verifican el funcionamiento interno del sistema. Las pruebas se planifican observando la estructura del algoritmo del sistema, sus condicionales, bucles, casos especiales… En las pruebas de **caja negra** no se tiene en cuenta el funcionamiento interno del sistema, solo se analizan entradas y salidas. [9]

4.1.1 Pruebas de Caja Blanca

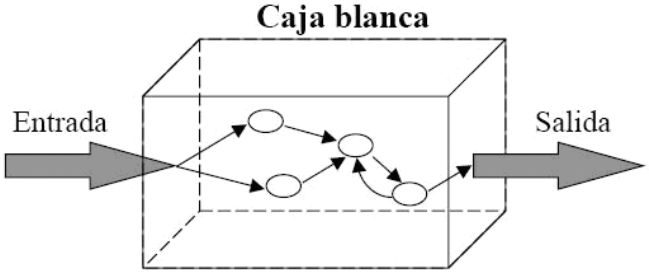

Figura 20. Prueba de Caja Blanca.

En estas pruebas se mostrará el pseudocódigo de cada módulo seguido del diagrama que muestra los caminos del módulo. Cada círculo representado en el diagrama contiene un número que coincide con la línea del pseudocódigo a la que nos referimos.

A continuación, se mostrará la complejidad de la prueba (número de caminos independientes) y los caminos.

Una vez obtenidos los caminos, se mostrarán entradas de la prueba, así como las salidas de la prueba y la salida esperada (la que debería haber dado la prueba). Si ambas coincidan, el módulo funcionará correctamente.

Módulo "Registro"

1. *Recuperar datos del formulario de registro;*
2. *Si los datos son válidos {*
3. *Intentar {*
4. *Crear objeto de la tabla "user";*
5. *Crear objeto de la tabla "user_create";*
6. *Guardar datos en BD;*
7. *Mostrar mensaje de éxito;*
8. *} Excepción {*
9. *Mostrar mensaje de error;*
10. *}*
11. *}*

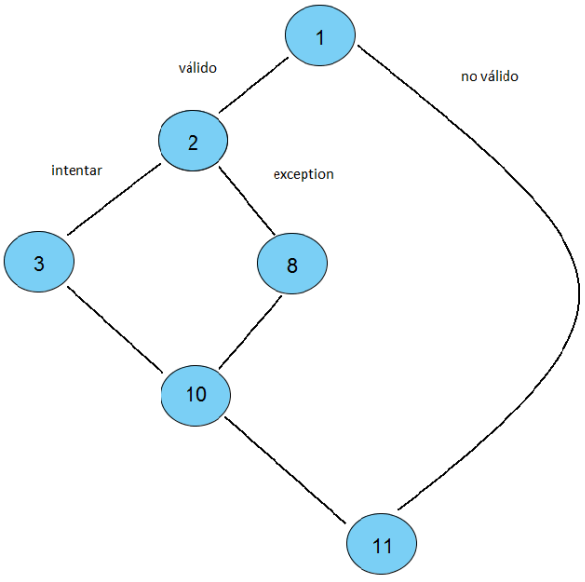

Figura 21. Prueba Caja blanca caso de uso Registro

Complejidad = 3

Caminos:

- Camino 1: 1, 2, 3, 10, 11
- Camino 2 :1, 2, 8, 10, 11
- Camino 3: 1, 11

	Valido	Intentar	Salida prueba	Salida esperada
Camino 1	Si	Si	-Guardar datos en BBDD -Mensaje éxito	-Guardar datos en BBDD -Mensaje éxito
Camino 2	Si	No	-Mostrar mensaje de error	-Mostrar mensaje de error
Camino 3	No	-	-Seguir en pantalla	-Seguir en pantalla

Tabla 8. Resultados de prueba caja blanca caso de uso Registro.

Módulo "Autenticación"

```
1.  Recuperar datos del formulario de autenticación;
2.  Si los datos son válidos {
3.      Intentar {
4.          Acceso con "user" y "password";
5.          Recupera los comentarios que tiene ese "user";
6.      } Excepción {
7.          Mostrar mensaje de error;
8.          Acceso = error;
9.      }
10.     Si acceso!= error{
11.         Autenticación (acceso);
12.     } Si no {
13.         Mensaje de error;
14.     }
15. }
```

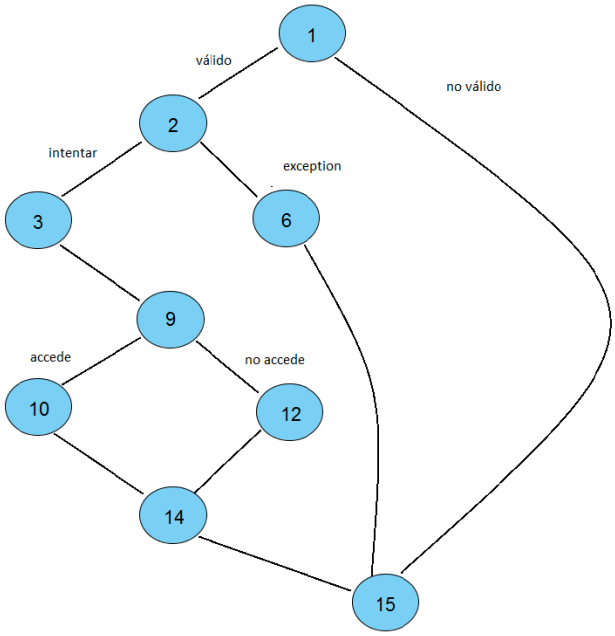

Figura 22. Prueba Caja blanca caso de uso Autenticación

Complejidad = 4

Caminos:

- Camino 1: 1, 2, 3, 9, 10, 14, 15
- Camino 2: 1, 2, 3, 9, 12, 14, 15
- Camino 3: 1, 2, 6, 16
- Camino 4: 1, 15

	Valido	Intentar	Accede	Salida prueba	Salida esperada
Camino 1	Si	Si	Si	-Recuperar comentarios del “user” -Autenticación(acceso)	-Recuperar comentarios del “user” -Autenticación(acceso)
Camino 2	Si	Si	No	-Recuperar comentarios del “user” -Mensaje de error	-Recuperar comentarios del “user” -Mensaje de error
Camino 3	Si	No	-	-Mensaje de error	-Mensaje de error
Camino 4	No	-	-	-Mensaje de error	-Mensaje de error

Tabla 9. Resultados de prueba caja blanca caso de uso Autenticación.

Módulo "Ver Perfil"

//función getTweets ()

1. *Intentar {*
2. *Importar Api Twitter;*
3. *Obtener datos de la api (nombre, descripción, timeline,…);*
4. *} Excepción {*
5. *Mostrar error;*
6. *}

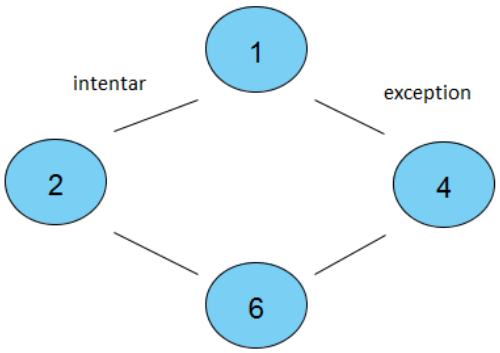

Figura 23. Prueba Caja blanca caso de uso Ver Perfil (1)

Complejidad = 2

Caminos:

- Camino 1: 1, 2, 6
- Camino 2: 1, 4, 6

	Intentar	Salida prueba	Salida esperada
Camino 1	Si	-Importar API Twitter -Obtener Datos Api	-Importar API Twitter -Obtener Datos Api
Camino 2	No	-Mensaje de error	-Mensaje de error

Tabla 10. Resultados de prueba caja blanca caso de uso Ver perfil (1).

//función getBiografía ()

1. *Intentar {*
2. *Importar Api LinkedIn;*
3. *Obtener datos de la api (nombre, centro de estudios, CV,…);*
4. *} Excepción {*
5. *Mostrar error;*
6. *}

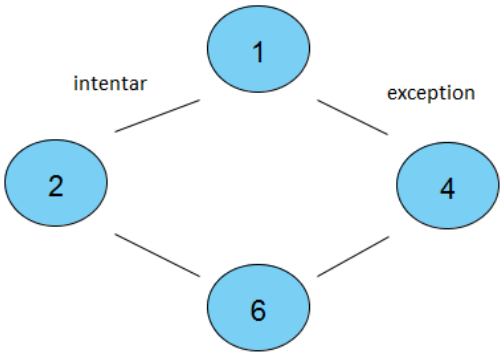

Figura 24. Prueba Caja blanca caso de uso Ver Perfil (2)

Complejidad = 2

Caminos:

- Camino 1: 1, 2, 6
- Camino 2: 1, 4, 6

	Intentar	Salida prueba	Salida esperada
Camino 1	Si	-Importar API LinkedIn -Obtener Datos Api	-Importar API LinkedIn -Obtener Datos Api
Camino 2	No	-Mensaje de error	-Mensaje de error

Tabla 11. Resultados de prueba caja blanca caso de uso Ver perfil (2).

Módulo "Buscar Perfil"

1. *Recuperar datos del buscador;*
2. *Si los datos son válidos {*
3. *Recupera los comentarios que tiene ese "user";*
4. *Para "user" recupera el LinkedIn asociado en la tabla "User_Create";*
5. *getTweets ();*
6. *getBiografía ();*
7. *}*

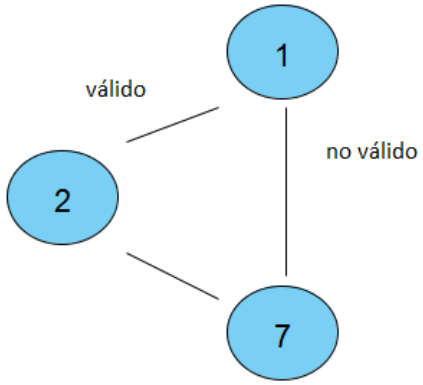

Figura 25. Prueba Caja blanca caso de uso Buscar Perfil

Complejidad = 2

Caminos:

- Camino 1: 1, 2, 7
- Camino 2: 1, 7

	Válido	Salida prueba	Salida esperada
Camino 1	Si	-Recuperar datos del user -GetTweets, GetBiografia	-Recuperar datos del user -GetTweets, GetBiografia
Camino 2	No	-Seguir en pantalla	-Seguir en pantalla

Tabla 12. Resultados de prueba caja blanca caso de uso Buscar Perfil.

Módulo "Escribir Comentario"

1. *Recuperar datos del formulario;*
2. *Si los datos son válidos {*
3. *Recupera el usuario que envía dicho comentario;*
4. *Recupera el usuario al que va destinado el comentario;*
5. *Guarda toda la información en una objeto instanciado de la tabla "Comentario"*
6. *}*

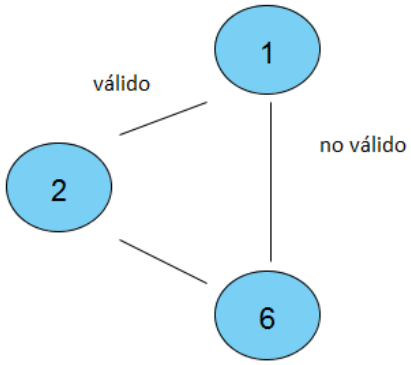

Figura 26. Prueba Caja blanca caso de uso Escribir Comentario

Complejidad = 2

Caminos:

- Camino 1: 1, 2, 6
- Camino 2: 1, 6

	Válido	Salida prueba	Salida esperada
Camino 1	Si	-Recuperar user -Recuperar destinatario -Almacenar comentario en BD	-Recuperar user -Recuperar destinatario -Almacenar comentario en BD
Camino 2	No	-Seguir en pantalla	-Seguir en pantalla

Tabla 13. Resultados de prueba caja blanca caso de uso Escribir Comentario.

Módulo "Volver a Home (pantalla principal)"

1. *Si usuario quiere regresar a la pantalla principal {*
2. *Recupera información del usuario actualmente logueado en la app;*
3. *getTweets ();*
4. *getBiografía ();*
5. *}*

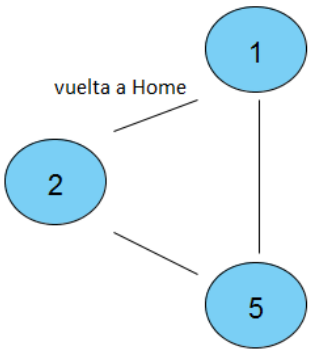

Figura 27. Prueba Caja blanca caso de uso Volver a pantalla Principal

Complejidad = 2

Caminos:

- Camino 1: 1, 2, 5
- Camino 2: 1, 5

	Vuelta Atrás	Salida prueba	Salida esperada
Camino 1	Si	-Recuperar datos del user logueado -GetTweets, GetBiografia	-Recuperar datos del user logueado -GetTweets, GetBiografia
Camino 2	No	-Seguir en pantalla	-Seguir en pantalla

Tabla 14. Resultados de prueba caja blanca caso de uso Volver a pantalla Principal.

Módulo "Cierre sesión"

1. *Si usuario quiere cerrar sesión {*
2. *Logout ();*
3. *Re direccionar a la pantalla inicial;*
4. *}*

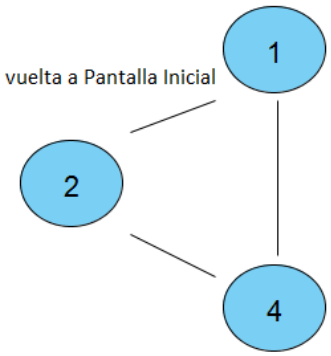

Figura 28. Prueba Caja blanca caso de uso Cierre sesión.

Complejidad = 2

Caminos:

- Camino 1: 1, 2, 4
- Camino 2: 1, 4

	Vuelta Atrás	Salida prueba	Salida esperada
Camino 1	Si	-LogOut; -Pantalla Inicial;	-LogOut; -Pantalla Inicial;
Camino 2	No	-Seguir en pantalla	-Seguir en pantalla

Tabla 15. Resultados de prueba caja blanca caso de uso Cierre sesión.

Conclusiones

Ya que todas las salidas de las pruebas son iguales a las esperadas, y no hay caminos de los que no se pueda salir, concluimos que la aplicación cumple todas las pruebas de funcionamiento.

4.1.2 Pruebas de Caja Negra

En el método de la **caja negra** se decide no tener en cuenta el funcionamiento interno de un sistema y solo se analizan sus entradas y salidas. Se aplica tanto como estrategia de testeo, fijándose más en el exterior (usuario) o en la conexión entre diferentes sistemas (interfaz), que como necesidad cuando no es accesible o no es practico estudiar el funcionamiento interno del sistema en análisis.

Para Caja Negra o pruebas de caja opaca se requiere menos habilidad técnica, menos tiempo y menos herramientas. Por ende, menos costo. Pero solo te permite detectar errores y fallos pero no te acerca a la solución de éstos.

Para realizar estas pruebas se elegirá la técnica de partición en clases de equivalencia, que consiste en identificar las condiciones de entrada de los módulos y obtener sus reglas, así como las clases válidas e inválidas obtenidas mediante dichas reglas. Una vez obtenidas estas clases, se buscan los casos de pruebas correspondientes. [10]

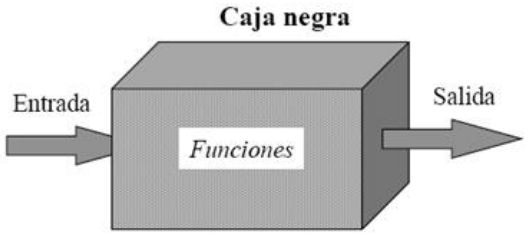

Figura 29. Prueba de Caja Negra

"home.html"

- Registro de la aplicación

Entrada	Reglas a seguir	Clases Válidas	Clases No Válidas
E-Mail	-alfanumérico@alfanumérico.com - No vacío	**1**. alfanumérico@alfanumérico.com que tenga menos de 200 caracteres y que sea No vacío.	**2.** noalfanumérico@noalfanumérico.com **3.** "" (vacío)
LinkedIn	- No vacío - No acentos	**4.** Que no sea vacío y que no tenga acentos en la URL.	**5.** Vacío **6.** Contenga acentos.
Twitter	- No vacío	**7.** Que no sea vacío.	**8.** Vacío
Contraseña	- <200 - No vacío	**9.** Con menos de 200 caracteres y que no sea vacío.	**10.** >200 caracteres **11.** Vacío
Repetir contraseña	- <200 - No vacío	**12.** Con menos de 200 caracteres y que no sea vacío.	**13.** >200 caracteres **14.** Vacío

Tabla 16. Clases de entrada para "registro"

NOTA: Para crear a nuestro nuevo usuario, tanto el campo "Twitter" como "LinkedIn" dependen directamente tanto del *user* de Twitter que tenga el usuario, como de la *url* de su perfil público, que posea. Es decir, que en dichos campos, solo podemos controlar que no sea vacío, ya que las demás reglas, son implícitas a la hora de escribir sobre los campos. Cuando el usuario, introduzca alguno de los campos, siguiendo las clases correctamente, pero no coincidan con usuarios creados en las 2 apis, no dará error el cual se pueda observar en pantalla, pero no se mostrara información alguna, lógicamente.

Caso válido:

E-mail	LinkedIn	Twitter	Contraseña	Repetir contraseña	Clases
Manu.aranda.9@gmail.com	https://es.linkedin.com/pub/manuel-aranda/b0/436/689	ManuAranda9	1234	1234	1,4,7,9,12

Tabla 17. Clase válida para "registro"

Casos no válidos:

E-mail	LinkedIn	Twitter	Contraseña	Repetir contraseña	Clases
….@------	https://es.linkedin.com/pub/manuel-aranda/b0/436/689	ManuAranda9	1234	1234	2
(vacío)	https://es.linkedin.com/pub/manuel-aranda/b0/436/689	ManuAranda9	1234	1234	3
Manu.aranda.9@gmail.com	(vacío)	ManuAranda9	1234	1234	5
Manu.aranda.9@gmail.com	http://es.linkedin.com/pub/mánuel-árándá/b0/436/689 (acento en la a)	ManuAranda9	1234	1234	6
Manu.aranda.9@gmail.com	https://es.linkedin.com/pub/manuel-aranda/b0/436/689	(vacío)	1234	1234	8
Manu.aranda.9@gmail.com	https://es.linkedin.com/pub/manuel-aranda/b0/436/689	ManuAranda9	12345…(>200)	1234	10
Manu.aranda.9@gmail.com	https://es.linkedin.com/pub/manuel-aranda/b0/436/689	ManuAranda9	(vacío)	1234	11
Manu.aranda.9@gmail.com	https://es.linkedin.com/pub/manuel-aranda/b0/436/689	ManuAranda9	1234	12345…(>200)	13
Manu.aranda.9@gmail.com	https://es.linkedin.com/pub/manuel-aranda/b0/436/689	ManuAranda9	1234	(vacío)	14

Tabla 18. Clases no válidas para "registro"

- Autenticación en la aplicación

Entrada	Reglas a seguir	Clases Válidas	Clases No Válidas
Twitter	- No vacío	**1.** Que no sea vacío.	**2.** Vacío
Contraseña	- <200 - No vacío	**3.** Con menos de 200 caracteres y que no sea vacío.	**4.** >200 caracteres **5.** Vacío

Tabla 19. Clases de entrada para "autenticarse"

Caso válido:

Twitter	Contraseña	Clases
ManuAranda9	1234	1,3

Tabla 20. Clase válida para "autenticarse"

Casos no válidos:

Twitter	Contraseña	Clases
(Vacío)	1234	2
ManuAranda9	1234…(>200)	4
ManuAranda9	(Vacío)	5

Tabla 21. Clase no válida para "autenticarse"

Para este módulo, de tan solo 2 casillas, además de comprobar que nuestros datos cumplen las clases, el sistema comprueba que existen en el sistema un usuario que corresponda con el introducido, y que además contenga esa contraseña. En caso negativo, aunque cumplan las clases anteriormente mencionadas, se mostrará en pantalla un error correspondiente a fallo en los datos introducidos.

"inicio.html"

- Búsqueda de perfil

Entrada	Reglas a seguir	Clases Válidas	Clases No Válidas
User/Twitter	- No vacío	**1**. No vacío.	**2.** Vacío

Tabla 22. Clases de entrada para "Buscar Perfil"

Caso válido:

User/Twitter	Clases
ManuAranda9	1

Tabla 23. Clase válida para "Buscar Perfil"

Casos no válidos:

User/Twitter	Clases
(Vacío)	2

Tabla 24. Clase no válida para "Buscar Perfil"

El funcionamiento de este módulo es similar al de "autenticarse" ya que los datos introducidos en el input, serán comprobados internamente en el sistema para ver si hay coincidencias de usuarios para mostrar. Tan solo podemos "controlar" que dicho campo no este vacío cuando se pulse el botón de "Buscar".

- Ver perfil

En este módulo no hay que introducir ningún tipo de dato, con lo que lo obviaremos lógicamente.

- Escribir comentario

Entrada	Reglas a seguir	Clases Válidas	Clases No Válidas
Comentario	- No vacío - <200 caracteres	**1.** Que no sea vacío y que contenga menos de 200 caracteres.	**2.** Vacío **3.** >200 caracteres

Tabla 25. Clases de entrada para "Escribir comentario"

Caso válido:

Comentario	Clases
Hola usuario, ¿qué tal estás?	1

Tabla 26. Clase válida para "Escribir comentario"

Casos no válidos:

Comentario	Clases
(Vacío)	2
Hola usuario, ¿qué tal estás?, soy tu admin…. (>200)	3

Tabla 27. Clase no válida para "Escribir comentario"

- Cierre de sesión

Para el cierre de sesión, tan solo tenemos que pulsar el botón situado arriba a la derecha, con lo que tampoco interfieren datos de entrada ni de salida.

- Volver a pantalla principal

Al igual que el apartado anterior, en este módulo no hay interacción de datos.

4.2 Pruebas de integración

En este caso probamos cómo es la interacción entre dos o más unidades del software. Este tipo de pruebas verifican que los componentes de la aplicación funcionan correctamente actuando en conjunto.

Este tipo de pruebas son dependientes del entorno en el que se ejecutan. Si fallan, puede ser porque el código esté bien, pero haya un cambio en el entorno.

En la siguiente figura, se muestra la relación entre cada uno de los módulos. Esta figura se ha tomado como referencia para realizar las pruebas y comprobar que cada una de las relaciones indicadas mediante las flechas son correctas y que los módulos funcionan correctamente.

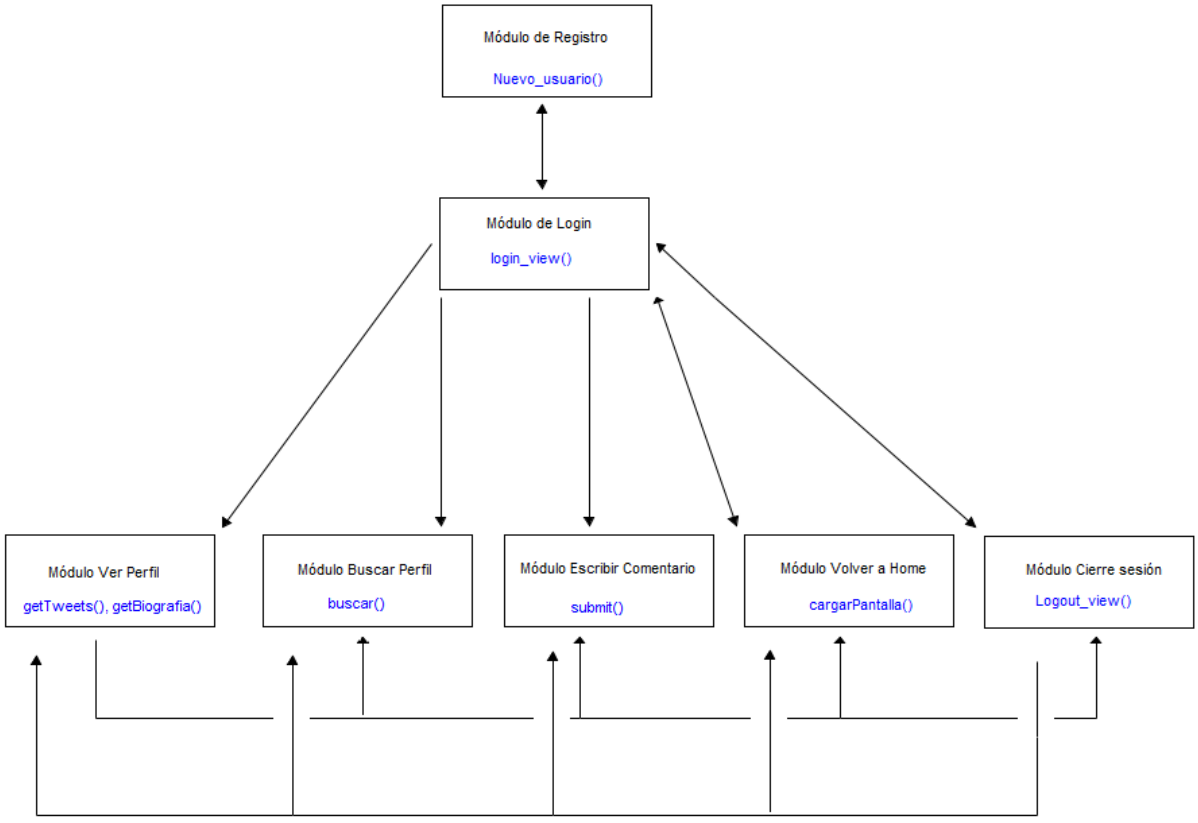

Figura 30. Diagrama pruebas de interacción

4.3 Pruebas de validación

En estas pruebas se comprueba el correcto funcionamiento de la aplicación respecto a los requisitos descritos en la fase de especificación. Para ello, dividiremos el apartado según los diferentes requisitos e iremos mostrando que se cumplen mediante capturas de pantallas.

RF1 – Registro en la aplicación

Como podemos observar en la figura 31, nuestra aplicación te muestra en la pantalla inicial, el módulo "registro" al cual accederemos para crear un usuario nuevo.

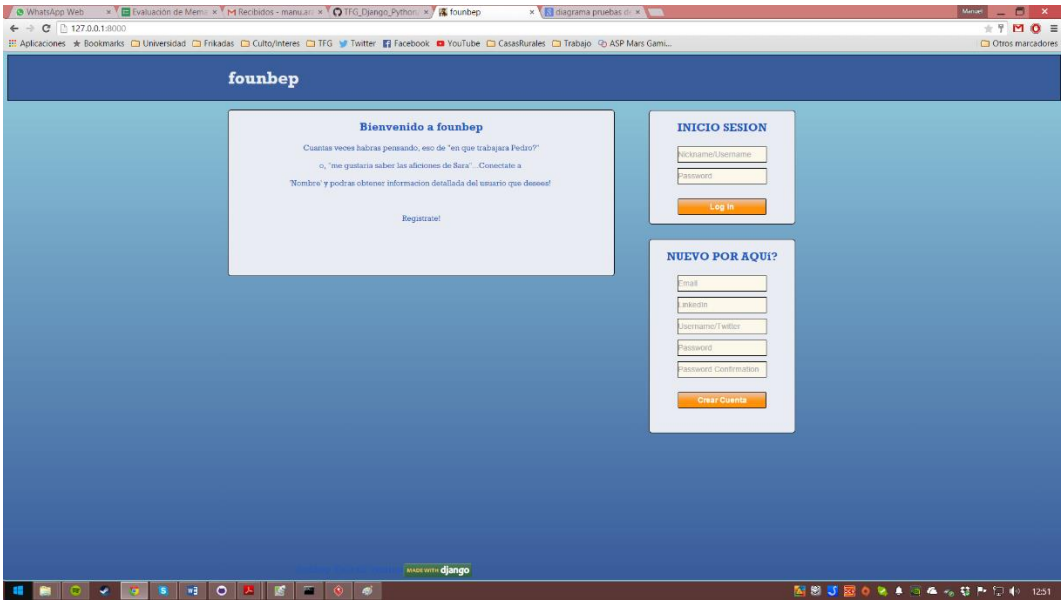

Figura 31. Pantalla inicial de la aplicación.

Figura 32. Pantalla módulo registro.

Una vez introducidos los campos, correctamente, como se explica en el apartado de pruebas anterior, pulsaremos el boton “crear cuenta” como se muestra en la figura X.

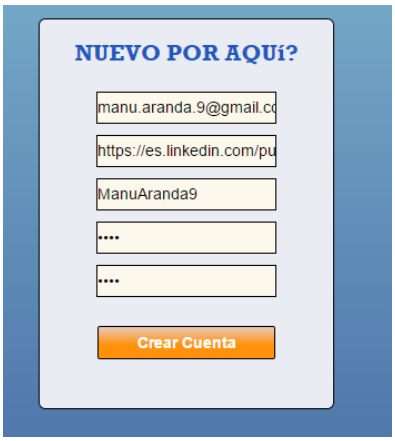

Figura 33. Pantalla campos de registro completados

El sistema comprobará que los campos son correctos, en cuanto a las clases descritas en “pruebas unitarias” y nos mostrara un mensaje aclaratorio en la parte superior de la pantalla.

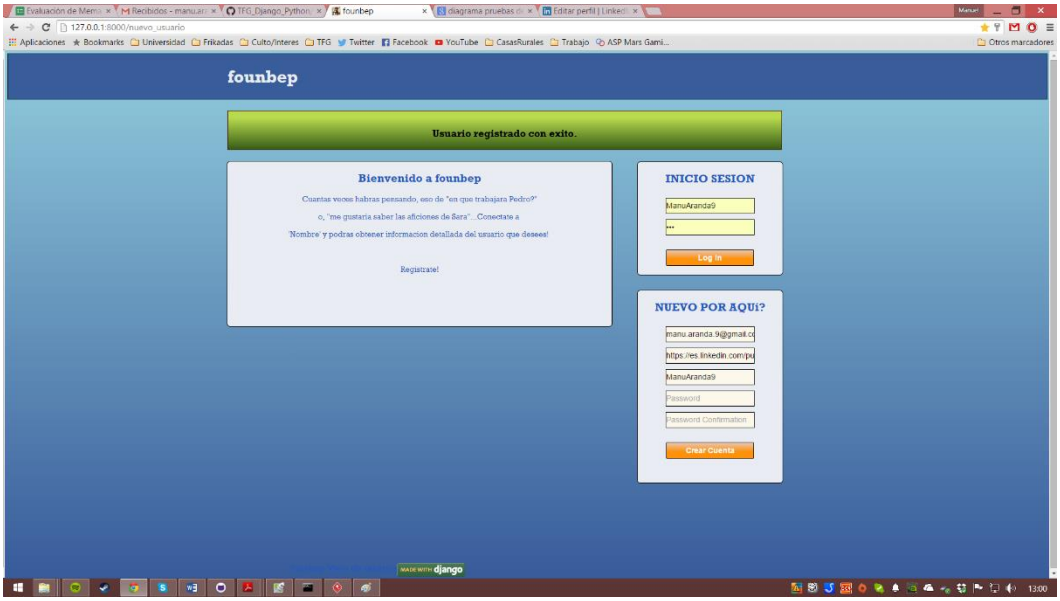

Figura 34. Pantalla éxito al registrar el usuario.

Por el contrario, si alguno de los campos están mal, o el *user* está repetido, el mensaje será de color rojo pero se nos mostrara en la misma parte de la pantalla.

RF2 – Autenticación.

Este es el modulo encargado de conectar al usuario creado anteriormente con el sistema, accediendo a su información almacenada en las APIs.

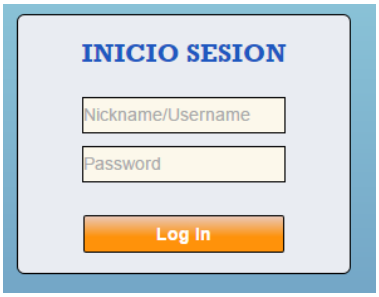

Figura 35. Pantalla módulo autenticación.

Introducimos los datos del usuario como se muestra en la figura 36, y pulsamos el botón "Log In".

Figura 36. Pantalla módulo autenticación con datos correctos.

Si los datos son erróneos se mostrara lo siguiente:

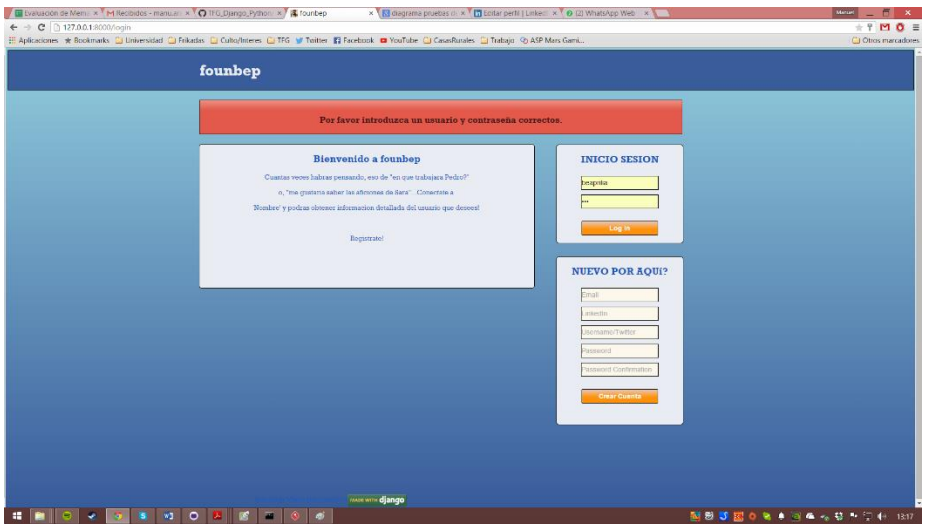

Figura 37. Mensaje de error al autenticarse

RF3 –Ver perfil.

Una vez autenticados satisfactoriamente, el sistema re-direcciona a la página principal de nuestro usuario, en la cual observamos información de Twitter como su nombre, descripción, tweets, favoritos, timeline… y su perfil profesional ligado a LinkedIn.

Debajo de este, están los comentarios que los demás usuarios nos han dejado, y que podremos leer en tiempo real.

Desde aquí podremos navegar, y buscar otro usuario registrado, volver a nuestra pantalla en caso de que estemos viendo otro usuario y cerrar sesión. No podremos escribir comentarios a nuestro propio perfil, lógicamente, por eso no nos muestra el área correspondiente a dicho módulo.

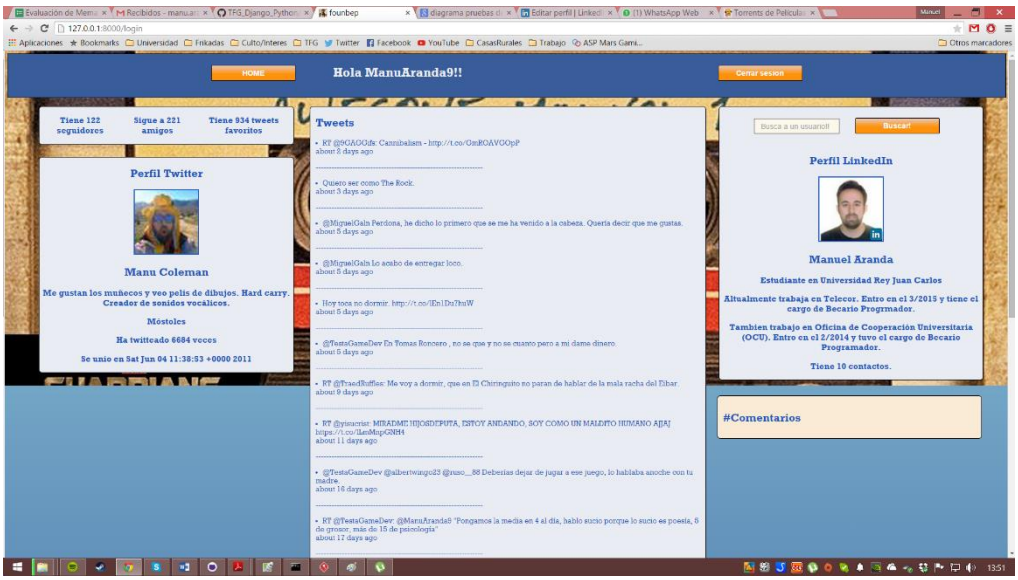

Figura 38. Ver perfil

RF4 – Buscar Perfil

Introducimos un nombre de usuario en el buscador, antes de darle al botón “Buscar” para que nos redirija a la vista del mismo. Si no hay coincidencias de usuarios registrados, nos mostrara un mensaje de error en la parte superior de la pantalla.

En la figura x vemos donde está situado este módulo.

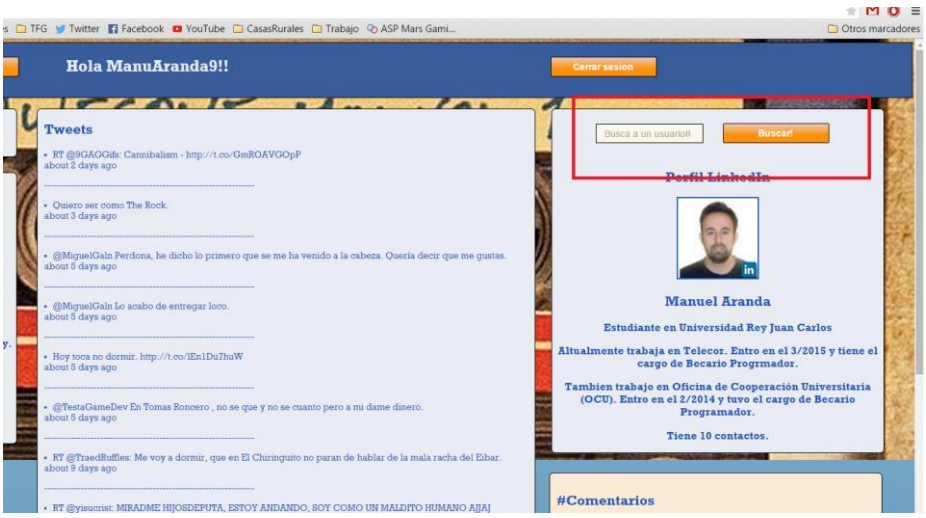

Figura 39. Buscar perfil

RF5 – Escribir comentarios.

Para poder escribir comentarios, es indispensable que estemos visualizando otro usuario, ya que como hemos dicho anteriormente, no podemos escribirnos a nosotros mismos. Como vemos en la figura X, estamos ya en el perfil de otro usuario (‘kriis1992’). Ahora si nos aparece el módulo de “escribir comentario”.

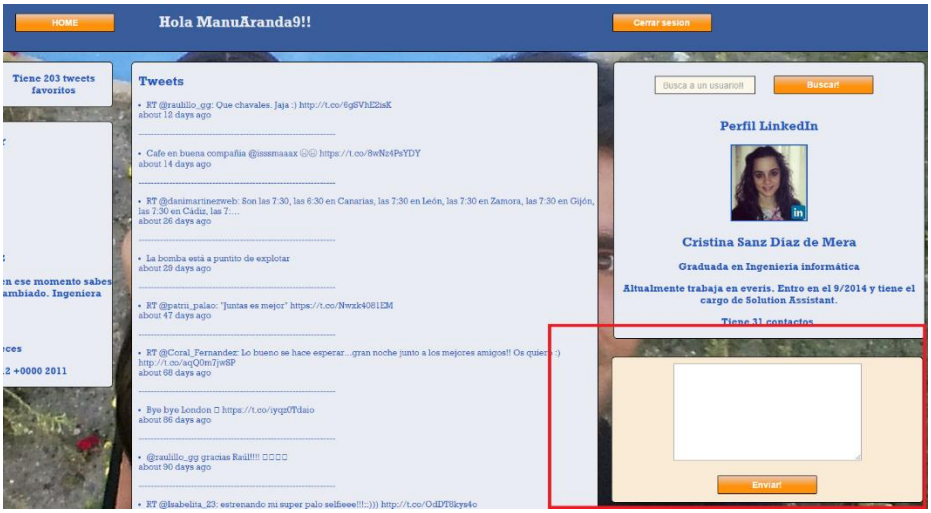

Figura 40. Escribir comentario

Arriba se puede ver, en el mensaje “Hola ManuAranda9”, que estamos *logueados* actualmente con ese usuario, aunque estemos viendo el perfil de otro.

RF6 –Volver al perfil.

Pulsando este botón, volveremos a la pantalla principal del usuario autenticado.

Figura 41. Volver a Home.

RF7 – Cerrar sesión.

Y por último, el módulo de cierre de sesión, cuyo funcionamiento es el mismo que el anterior, salvo que en este caso, nos vuelve a redirigir hacia la pantalla de log/registro.

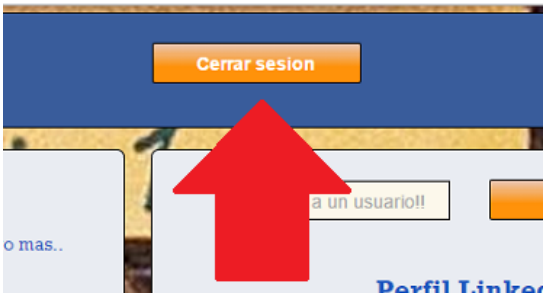

Figura 42. Cierre de sesión.

4.4 Pruebas de aceptación

Este tipo de pruebas son realizadas por los usuarios sobre el sistema completo con el fin de ayudar a corregir errores que no se hayan detectado, y poder mejorar la aplicación en aspectos relacionados con la interacción *Persona-Ordenador*. Se han realizado las pruebas a 5 usuarios de distinto perfil y se han anotado los detalles de cada prueba.

Se muestra las preguntas iniciales realizadas a los usuarios, y en la tabla x se recoge el resto de información solicitada en la escala de 0-10. Estos resultados son fruto de un uso aproximado de 15 minutos, tiempo más que suficiente para probar la funcionalidad completa de la aplicación.

Edad: _____

Ocupación: _____

¿Utilizas las redes sociales a diario? En caso afirmativo, ¿cuáles y con qué frecuencia?:_____

Valorar de 1-10:

¿Le ha gustado la aplicación?:_____

¿Le ha resultado intuitivo el manejo de la aplicación?:_____

¿Ve útil este tipo de aplicación?:_____

Valora la rapidez: _____

¿Has necesitado ayuda en algún momento de la interacción?:_____

¿Qué mejorarías de ella?:_____

Figura 43. Cuestionario

Ocupación	Edad	Redes sociales	Gusto	Intuitivo	Útil	Rapidez	General
Informático	25	Twitter, Facebook, LinkedIn,Instagram...	9	7	8	7	8
Informático	25	Twitter, Facebook, Instagram...	8	9	9	8	9
No-Informático	51	-	9	8	10	9	9
No-Informático	24	Twitter, Facebook, LinkedIn,Instagram...	7	8	7	8	7
No-Informático	18	Twitter, Facebook…	8	9	7	7	8
			8.2	8.2	8.2	7.8	8.2

4.4.1. Gráficas

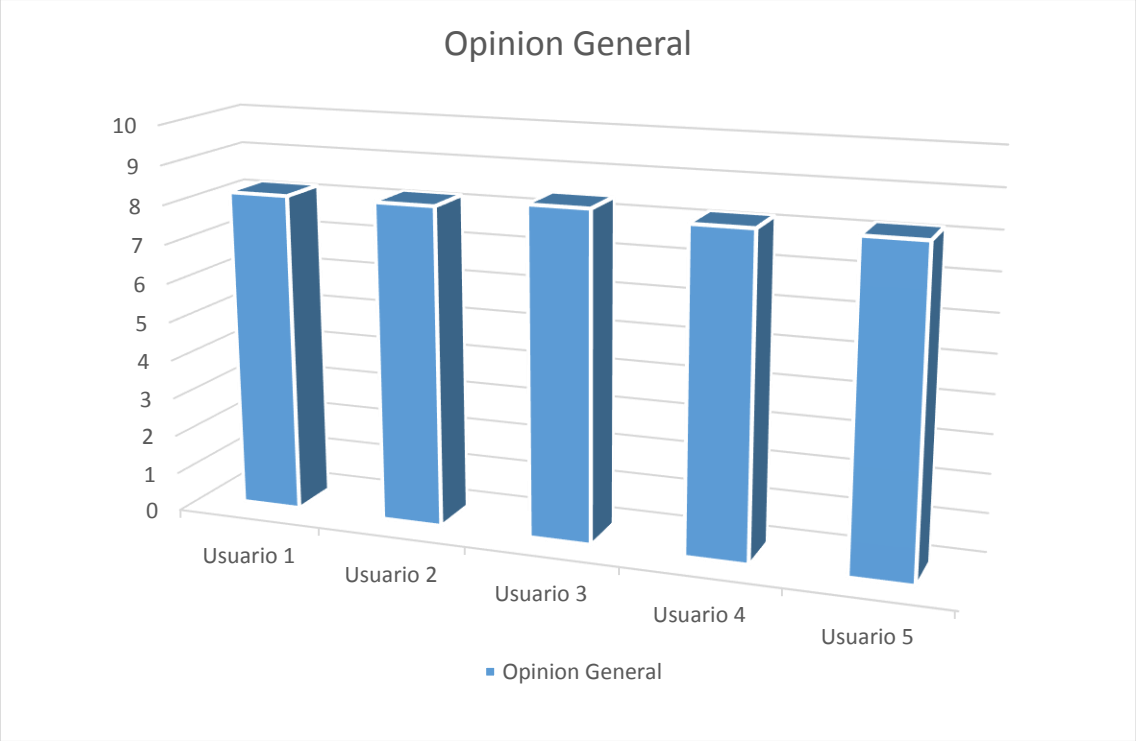

Figura 44. Gráfica “opinión general”.

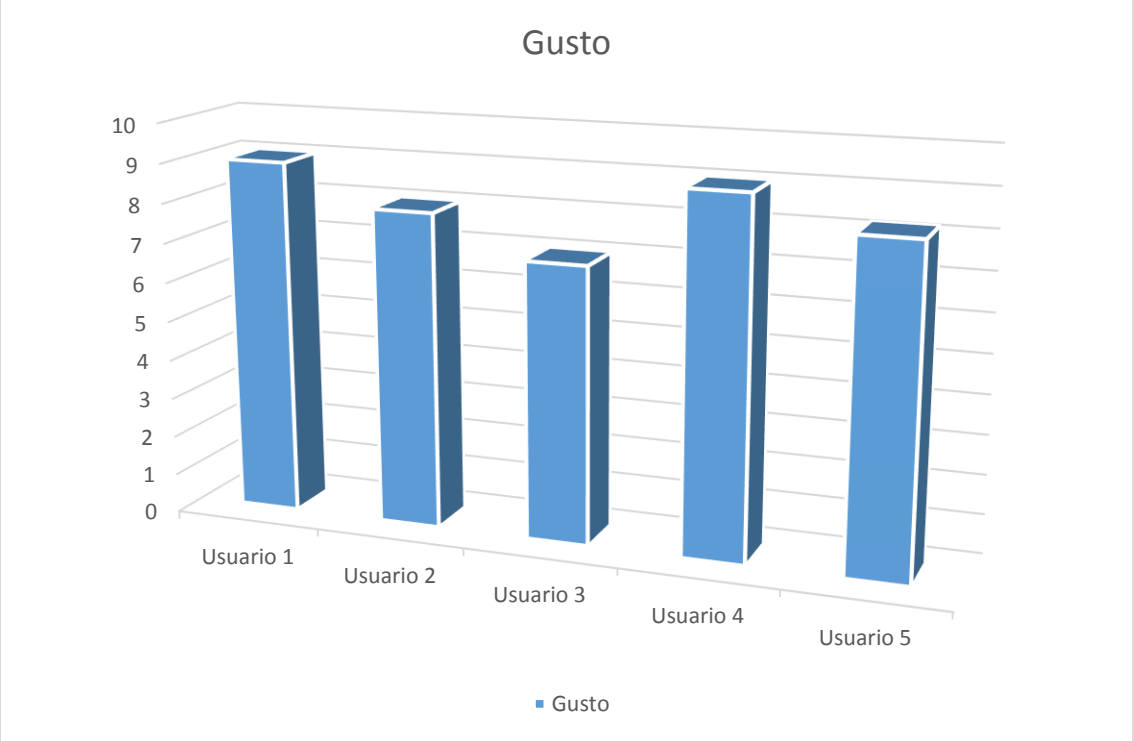

Figura 45. Gráfica “gusto”.

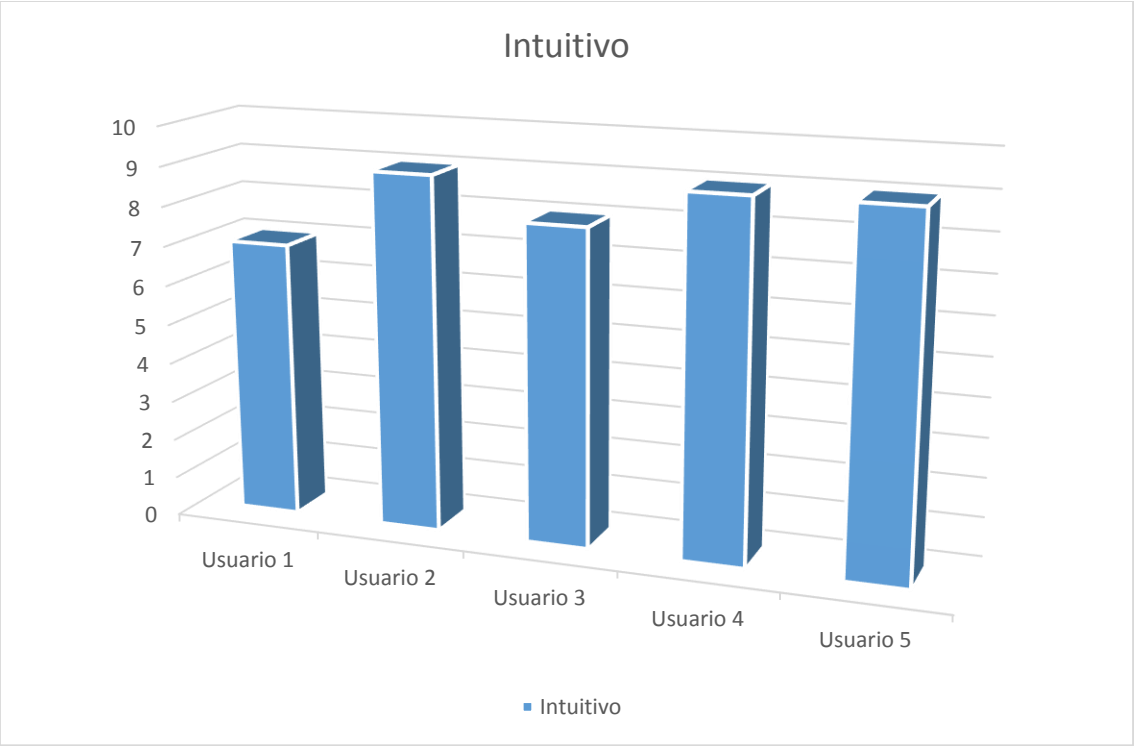

Figura 46. Gráfica “intuitivo”.

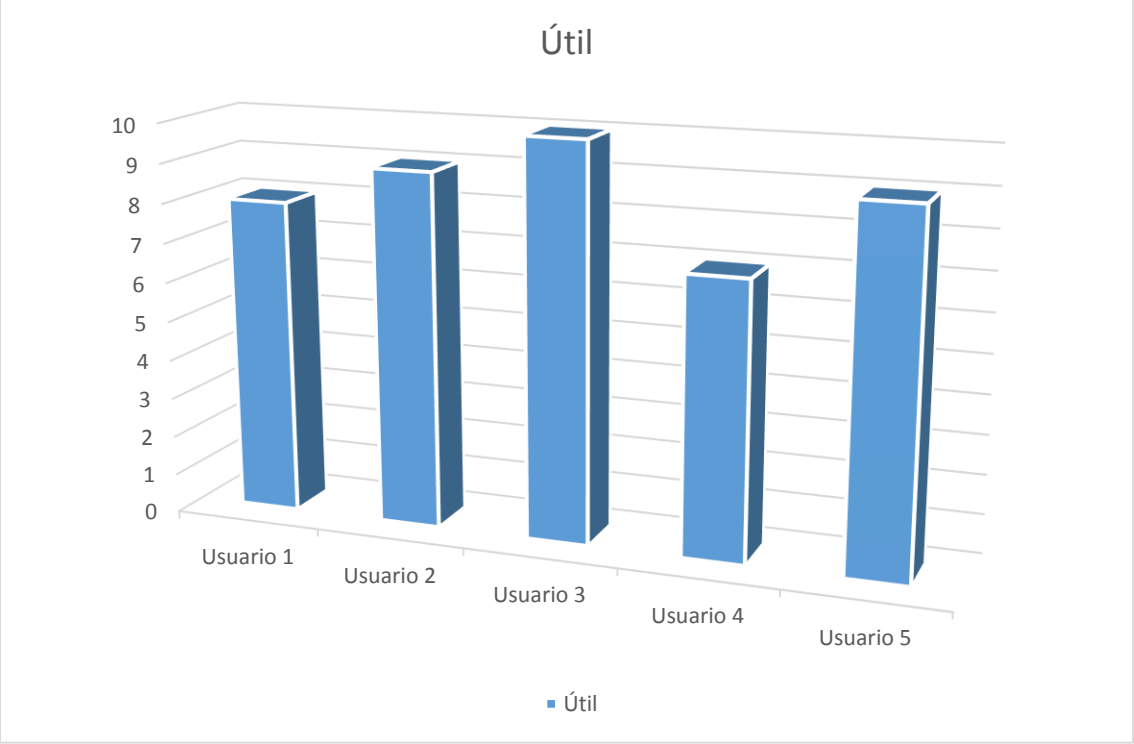

Figura 47. Gráfica “útil”.

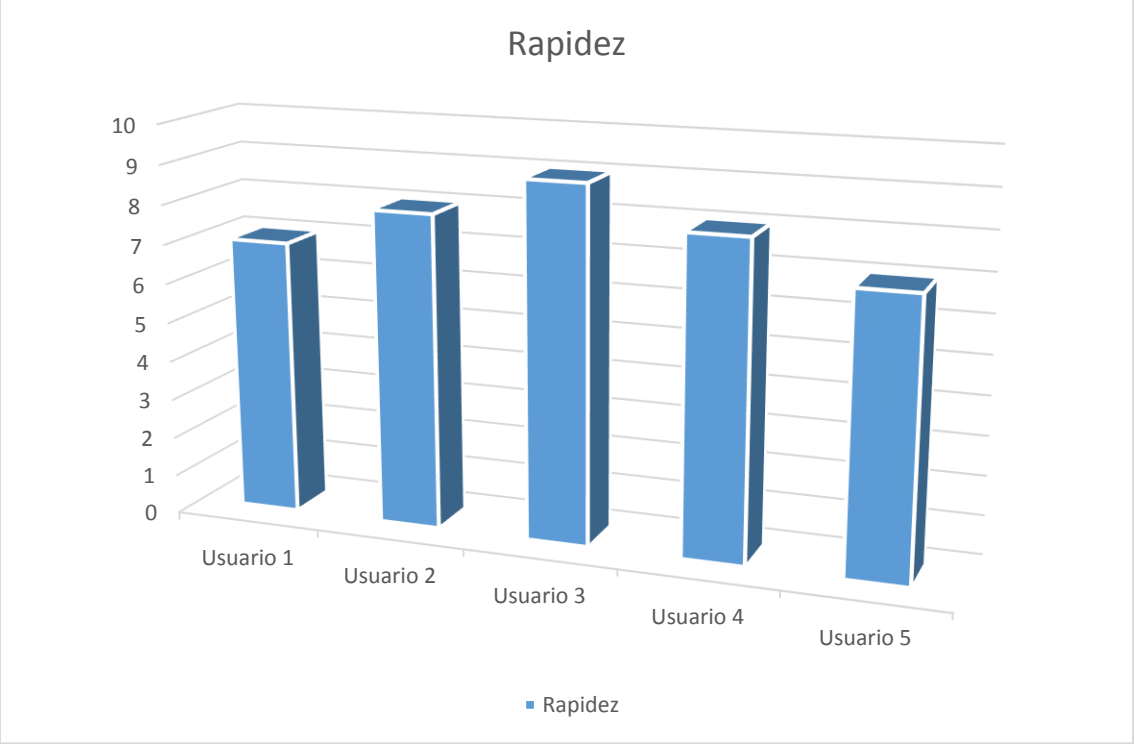

Figura 48. Gráfica “rapidez”.

Con las anteriores gráficas, y la tabla donde se muestran los datos, podemos comprobar que los requisitos no funcionales de la aplicación han quedado solventados:

RNF 1 – Nuestro sistema tendrá una interfaz intuitiva y sencilla, llegando así a un mayor número de usuarios. (Principios Interacción Persona-Ordenador)

Podemos observar que la nota media en este aspecto es de 8,2. El rango de edades es variado, con lo que podemos concluir que es una aplicación notablemente intuitiva y sencilla. Esto es debido a que es una aplicación meramente de visualización y su única complejidad consistiría en registrar al usuario proporcionando correctamente sus datos.

RNF 2 – El sistema deberá responder en tiempo real a las búsquedas del usuario, evitando respuestas lentas.

La valoración de este apartado es bastante notable, al igual que la anterior, por lo que se considera un requisito cumplido. Tener en cuenta, que una posible lentitud en el sistema, seria causada por alguna irregularidad en las apis, con lo que indirectamente, la herramienta se ralentizaría.

RNF 3 – El sistema podrá usarse desde cualquier ordenador y cualquier sistema operativo.

Los usuarios han probado la aplicación desde Windows XP, Windows 8, Windows 8.1 y Ubuntu 15.04. Y desde Google Chrome, Firefox e Internet Explorer.

Los usuarios han aportado las siguientes sugerencias sobre las mejoras de la aplicación:

- Experimentar con la interfaz y probar cosas más actuales.
- Añadir alguna funcionalidad extra.
- Añadir más redes sociales.
- En la sección central donde podemos visualizar el *timeline*, en los tweets que tengan referencias a contenido multimedia, hacer que se puedan visualizar.
- Intentar acceder de manera completa a la API de LinkedIn

Capítulo 5

CONCLUSIONES

El objetivo de este proyecto era crear una aplicación, que unificara en una misma plantilla, datos de las APIs de Twitter y de LinkedIn dando visión general de un usuario.

Las pruebas realizadas han verificado el correcto funcionamiento del sistema, y también se han validado los requisitos expuestos en el análisis como se describe a continuación:

Lista de requisitos funcionales:

RF1 – El usuario podrá darse de alta en la aplicación. Para ello tendrá que dejar su Twitter y su URL publica de LinkedIn para poder enlazar con sendas apis.

RF2 – El usuario una vez registrado, podrá autenticarse en la herramienta, con su user y su password.

RF3 – El usuario podrá ver información de su perfil.

RF4 – El usuario podrá buscar en el sistema a otros usuarios ya registrados anteriormente para ver sus perfiles.

RF5 – El usuario podrá dejar comentarios en los demás perfiles.

RF6 – El usuario podrá volver a su perfil.

RF7 – El usuario podrá cerrar sesión y volver a la pantalla de login/registro.

Es decir, que nuestro usuario podrá registrarse en la aplicación aportando sus datos, para más adelante poder hacer uso de la misma, autenticándose.

Podrá visualizar su perfil, buscar a otros usuarios, a los que adicionalmente podrá dejarle algún comentario en su perfil. Y por supuesto, cerrar sesión y volver a la pantalla principal.

Por tanto, todos los requisitos funcionales han sido validados.

Lista de requisitos no funcionales:

RNF 1 – Nuestro sistema tendrá una interfaz intuitiva y sencilla, llegando así a un mayor número de usuarios. (Principios Interacción Persona-Ordenador)

RNF 2 – El sistema deberá responder en tiempo real a las búsquedas del usuario, evitando respuestas lentas.

RNF 3 – El sistema podrá usarse desde cualquier ordenador y cualquier sistema operativo.

Vistas las notas medias de las pruebas de validación, hechas a los 5 usuarios, podemos concluir que los requisitos no funcionales pasan la nota mínima, al ser una aplicación sencilla, intuitiva, que responde ágilmente a nuestras peticiones y vista desde cualquier sistema operativo y/o distinto navegador web.

Capítulo 6

TRABAJO FUTURO

Durante el análisis y el desarrollo del proyecto, han ido surgiendo ideas, que más adelante se han corroborado en las pruebas de aceptación, ya que los usuarios que han probado la aplicación, hacían un *feedback* similar a las posibles ideas futuras.

Esta aplicación se pensó principalmente como una herramienta que unificara varias redes sociales, que ayuden a visualizar de un mismo usuario, datos diversos como su experiencia laboral, y sus aficiones. Pero el desarrollo e investigaciones posteriores, hacen pensar, en que *founbep* podría tener un enfoque empresarial y laboral, en lugar de algo tan social. Es decir, de cara a las empresas, en RRHH, cuando se proponga un proceso de selección entre varios posibles empleados, con esta aplicación podrían saber el perfil completo de una persona, y las posibles orientaciones en su vida cotidiana. Y así saber si dicho perfil se ajusta a la filosofía de la empresa, con solo una ojeada en dicha nuestra aplicación.

Existen seguramente, cientos de opciones, y enfoques diferentes, ya que es un tipo de red social, al fin y al cabo, pero quiero destacar el anterior, como una de las ideas más atractivas.

A continuación se expondrán brevemente todas las ideas que han surgido, y que los usuarios que han probado *Founbep* han aportado:

- Dentro del apartado estético, se ha podido ver, que en una aplicación de carácter social, el impacto visual es imprescindible y fundamental. Por eso, surgen nuevas plataformas y herramientas que pueden facilitar la realización de un *front* más intuitivo, sencillo e innovador. Dos de estas herramientas son **Material Design**, y **Boostrap.** Ambas propuestas tienen un diseño más limpio, en el que predominan animaciones y transiciones de respuesta, el relleno y los efectos de profundidad tales como la iluminación y las sombras.

- En el entorno de usabilidad, para el futuro, seria vital implementar la funcionalidad completa que nos aportan las APIs. Por ejemplo, cuando un usuario se registre en el sistema, preguntarle explícitamente, si da permiso para que se acceda al API tanto de Twitter como de LinkedIn para acceder a la totalidad de sus datos, y poder modificarlos. Eso haría que *Founbep* no solo fuera una aplicación de visualización, sino una plataforma en la que poder twittear, modificar tu perfil, actualizar tu experiencia laboral en este último año…etc.

- A especie de actualización más a corto plazo, existe la posibilidad de introducir otra u otras redes existentes más a la aplicación, como podría ser **Instagram, Facebook**...

- En el apartado de seguridad, es imprescindible bajo un punto de vista objetivo, la implementación adicional en el módulo de registro. Ésta constaría de mandar un e-mail, al correo que el usuario introduzca, para que complete desde el mismo el registro, y así poder controlar el intrusismo de cualquier usuario al sistema.

BIBLIOGRAFIA

Referencias web

[1]http://webplusplus.blogspot.com.es/2011/11/comparativa-python-vs-php-vs-java.html

[2]http://librosweb.es/libro/django_1_0/

[3]http://es.wikipedia.org/wiki/PHP

[4]http://django.es/

[5]http://es.slideshare.net/rosamariamondragongomez/definicin-de-html-42015911

[6]https://www.masadelante.com/faqs/css

[7]http://www.maestrosdelweb.com/curso-django-instalacion-y-primera-aplicacion/

[8]http://informatica-iutll.blogspot.com.es/2013/03/proceso-unificado-de-desarrollo.html

[9]https://books.google.es/books?id=rXU-WS4UatYC&pg=PA335&lpg=PA335&dq=proceso+unificado+de+desarrollo+de+software+captura+de+requisitos&source=bl&ots=vvsEyb2i_Z&sig=csYJel9vjJnShvhCEZfkUplR_-s&hl=es&sa=X&ei=L4ZPVbtGitZRjoyAkA0&ved=0CCYQ6AEwATgK#v=onepage&q=proceso%20unificado%20de%20desarrollo%20de%20software%20captura%20de%20requisitos&f=false

[10]https://sites.google.com/site/pruebasdesofwareunitarias/tarea

[11]http://django.es/docs/intro/tutorial01/

APÉNDICES

Apéndice A

PUBLICACIÓN DEL CÓDIGO

El código desarrollado y esta memoria se encuentran en el repositorio GitHub:

https://github.com/manuaranda/TFG_Django_Python

Las APIs utilizadas para Python han sido descargadas desde:

Django: *http://django.es/*

API Twitter: *https://code.google.com/p/python-twitter/*

API LinkedIn: *https://pypi.python.org/pypi/python-linkedin/4.0*

Apéndice B

PLANIFICACION DEL PROYECTO

En esta sección de la memoria se detallara el tiempo realizado en el proyecto; de esta manera quedará descrito el tiempo que ha sido necesario para poder realizar la aplicación y por tanto el volumen de trabajo que ha sido necesario emplear para poder llevarlo a cabo. Es necesario explicar, que mientras se realizaba el proyecto, tuve que trabajar en una beca. Todo ello ha supuesto un enorme trabajo y dedicación que sin embargo también han provocado mayor satisfacción al haber logrado terminar en este año 2015.

B.1 Elección del proyecto

En Marzo-Abril de 2013, se me asignó este proyecto, de un listado que elegí como posibles candidatos. Hablé con Felipe Alonso, por primera vez y hablamos de las posibles orientaciones que podría tomar la futura herramienta. Le dimos varias vueltas, y en la siguiente reunión, a Felipe se le ocurrió darle una vertiente más de futuro, que es la que actualmente tiene la aplicación. Comentamos la posible arquitectura, las herramientas a utilizar, los *frameworks*, y buscamos en común bibliografía para adentrarnos en los conocimientos que utilizaría de allí en adelante.

B.2 Estudio de los objetivos

Desde que acabó el curso, y durante todo el verano estuve indagando de lleno en todo lo que habíamos hablado. Todo de lo que me había hablado Felipe era nuevo, tanto la tecnología Django, como el lenguaje PHP y por supuesto el acceso a las APIs tanto de Twitter como de LinkedIn, por ello supuso un inicio un poco complicado.

B.3 Desarrollo de la aplicación

Desde finales del año 2013 que fue cuando acabé todas las asignaturas de la carrera, empecé a desarrollar las primeras líneas de código, interactuando en primera instancia con las APIs, de un modo muy rudimentario. En Febrero de 2014, encontré trabajó, como becario para una empresa afiliada a la universidad, y hasta septiembre de ese mismo año deje el TFG a un 10%.

No fue hasta Octubre de 2014 cuando retomé las líneas, y me puse al 100% de mis recursos en el desarrollo. Después de varios meses de 2015, en marzo volví a encontrar trabajo, con lo que las mañanas volvía a tenerlas ocupadas. Pero tenía el proyecto muy avanzado, y quería presentarlo en la convocatoria de Junio, y es aquí donde tuve la mayor carga de esos 2 años de desarrollo.

B.4 Desarrollo de la memoria

La memoria la empecé a hacer en mayo de 2015, con el desarrollo del proyecto a un 90%. Es una memoria minuciosa, con detalles, y al ser la última de la carrera, le he dedicado muchas horas para que el resultado sea bastante notable.

APÉNDICE C

MANUAL DE USUARIO

Al acceder a *Founbep*, lo primero de todo, será crearnos un usuario para poder ver nuestro perfil, y el de los usuarios registrados que busquemos.

Para ello, los campos a introducir serán los siguientes:

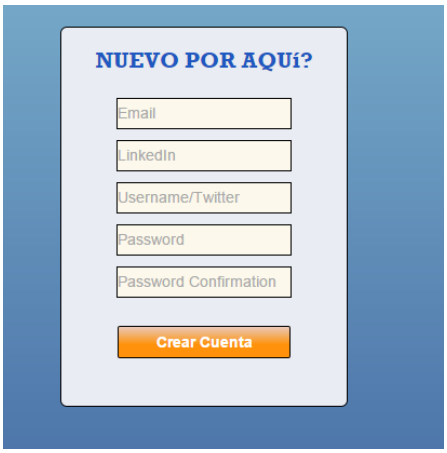

Figura 49. Campos de Registro.

En "LinkedIn", pondremos la URL que encontramos en nuestro perfil público de la página, situada en mitad de la misma, como se muestra en la figura 50.

NOTA: esta URL no tiene que tener acentos/tildes, ya que da conflicto en el acceso a la API. Por ejemplo, si la URL fuera "https://es.linkedin.com/pub/arturo-gonzález-díaz/a4/669/527" deberíamos sustituir el "gonzález" por "gonzalez", para que funcione.

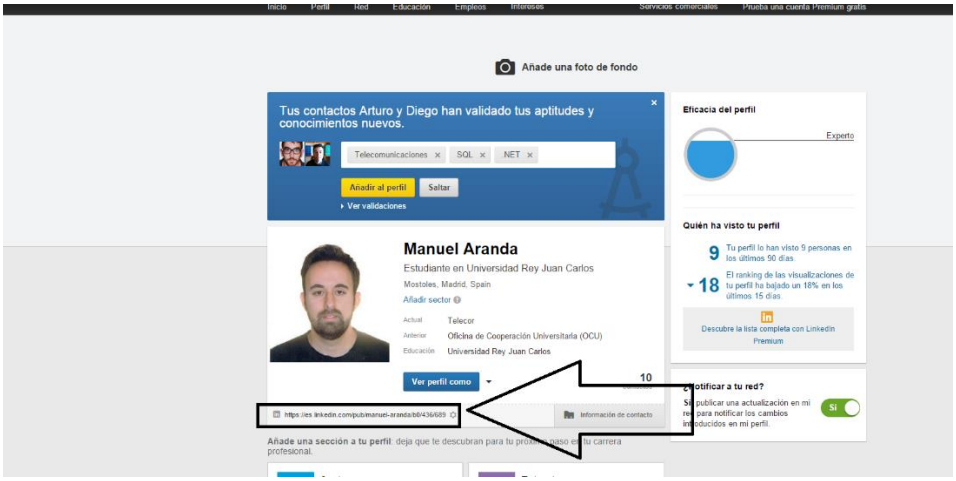

Figura 50. Pantalla de linkedin.es.

En “Twitter”, pondremos nuestro nombre de usuario, el cual no acarrea ningún problema.

Completamos el formulario introduciendo nuestro e-mail y la contraseña, dos veces.

Si todo ha salido correctamente, nos saldrá un mensaje de éxito, por pantalla y ya podremos acceder por primera vez al sistema. En caso contrario, volveremos a repetir la acción introduciendo los datos correctos.

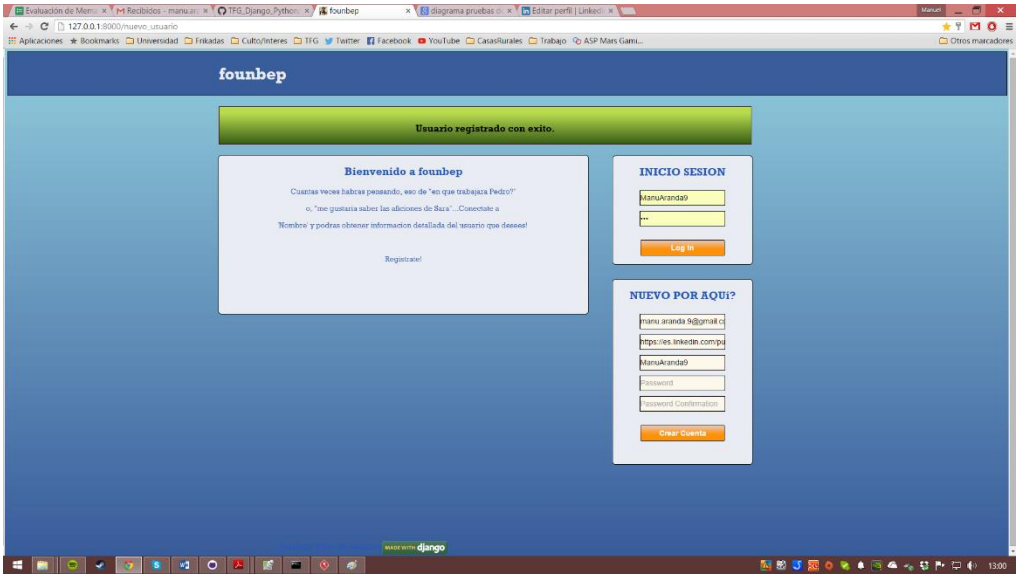

Figura 51. Registro con éxito.

Una vez registrado, accederemos con el *user* de Twitter y la contraseña. Se nos muestra la pantalla principal y la visión de los datos que las APIs pueden aportar acerca de las 2 redes sociales.

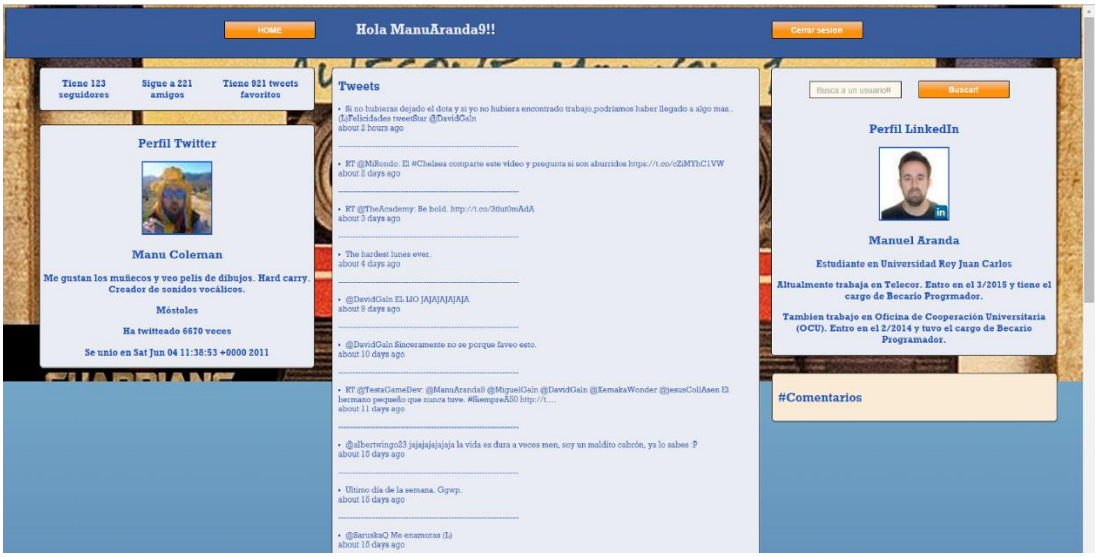

Figura 52. Pantalla principal.

Una vez en la pantalla principal podremos buscar utilizar el buscador para encontrar a otros usuarios, y así comentar en sus perfiles.

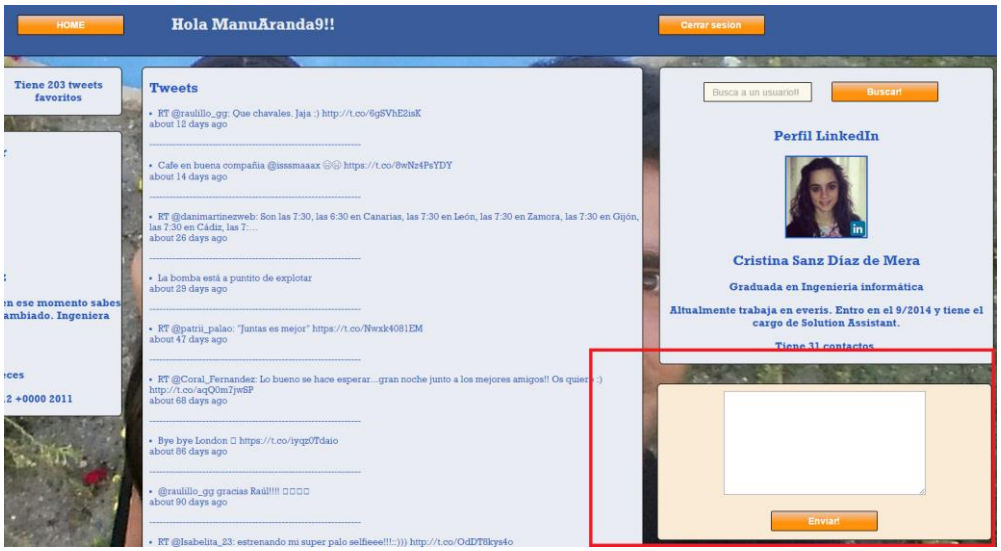

Figura 53. Después de buscar a otro usuario

Podremos cerrar sesión, y/o volver a la pantalla principal en cualquier momento, tan solo pulsando los botones que nos encontramos en la parte superior de la pantalla.

APÉNDICE D

MANUAL DE INSTALACIÓN/ DESPLIEGUE

Al ser una aplicación web, no necesita una instalación previa, pero vamos a explicar en las líneas siguientes el desarrollo de la instalación para su implementación y el despliegue del servidor local.

Django no funciona con Python 3.0 actualmente, debido a incompatibilidades con el intérprete de Python.

Para crear nuestro primer proyecto, abrimos una terminal (o ventana de comandos si así lo conoces en Windows), nos ubicamos en la carpeta en donde queremos crear nuestro proyecto y digitamos:

django-admin.py startproject nombreCarpeta

Esta instrucción creará dos directorios con el nombre del proyecto (en este caso: carpetaPrueba) y 5 archivos distribuidos de la siguiente manera:

-manage.py

-nombreCarpeta

__init__.py

settings.py

urls.py

wsgi.py

Para ver que el proyecto está funcionando en la terminal debemos escribir:

python manage.py runserver

Con esto, lo que acabamos de hacer es arrancar el servidor de desarrollo de Django, un servidor web liviano escrito completamente en Python. Así desarrollaremos de manera rápida sin que tengamos que vérnoslas con un servidor de producción.

Es bueno recordar que este servidor es meramente de desarrollo, no se debe usar en un entorno de producción, ya que no es su objetivo.

Introducimos en nuestro navegador http://127.0.0.1:8000 para comprobar que nuestro servidor está andando. Veremos una página de bienvenida de Django con "welcome to Django" escrito en ella.

Ahora procederemos a configurar y sincronizar nuestra base de datos. Como estamos usando SQLite, la base de datos será un archivo de datos en nuestro PC, el cual se creará cuando sincronicemos la base con la aplicación por primera vez. Aclarar que usamos SQLite en lugar de otras porque viene incluido en el paquete Python y es de un uso muy sencillo, entre otras cosas.

Como hemos mencionado líneas arriba, es necesario sincronizar nuestra aplicación con la base, por tanto necesitaremos crear las tablas en la base antes de poder usarlas. Para hacerlo, ejecutaremos el siguiente comando:

python manage.py syncdb

El comando "syncdb" revisa la variable "INSTALLED_APPS" y crea las tablas necesarias de acuerdo a la configuración de la base registrada en "settings.py".

Ahora que tenemos el entorno del proyecto listo, tendremos que crear nuestro modelo, para ello nos situamos en la carpeta raíz y tecleamos:

python manage.py startapp nombreAplicacion

Esto creará un directorio "nombreAplicacion" que contiene lo siguiente:

-nombreAplicación

__init__.py

models.py

test.py

views.py

El primer paso para codificar una aplicación web Django es definir nuestros modelos, esencialmente es la estructura de la base de datos, con metadatos adicionales.

Ahora ejecutamos de nuevo el comando “syncdb” para crear esas tablas y sincronizar nuestro modelo con ellas.

Para terminar el despliegue, cada vez que queramos ver nuestros progresos, teclearemos “syncdb” seguido de “runserver”. [11]