UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

Plataforma escalable para la captura y gestión de datos en Twitter (PARSE4U)

Dirigido por:
Anibal Bregón Bregón
Miguel Angel Martínez Prieto

"Aunque nuestra visión hacia delante es muy corta, podemos darnos cuenta de que hay mucho por hacer."

Alan Mathison Turing

"*Si ves a alguien ahogarse y sabes nadar,*
tienes el deber moral de salvarlo, a no ser que sea Bush o Aznar."

Richard Stallman

Agradecimientos

Quiero dar las gracias a mis tutores Anibal y Miguel Ángel, por la incalculable ayuda y dedicación que me han brindado a lo largo de este tiempo; Álvaro por todo el trabajo y sufrimiento en común por el que hemos pasado; a todos las personas involucradas en el desarrollo de Twitter y Java, sin ellos PARSE4U no existiría; a mi familia, por todo el apoyo y el animo recibido a lo largo de estos años; especialmente, gracias a mi pareja por apoyarme y soportar mi humor en los momentos duros.

Resumen

Este proyecto quiere proporcionar a los usuarios una herramienta que permita recuperar y almacenar, sin la necesidad de trabajar directamente con la API de Twitter. El propósito es ofrecer una plataforma que permita filtrar, conservar y ofrecer la información capturada mediante la API Streaming de Twitter. De esta manera, ante la necesidad de desarrollar un análisis de un evento, o el deseo de elaborar un estudio o investigación basándose en tweets, PARSE4U Tracking posibilita el acceso a los datos obtenidos en bruto.

Abstract

This project wants to provide to the users a tool that it allows to recover and to store, without the need to work directly with Twitter's API. The intention is to offer a platform that allows to filter, to preserve and to offer the information captured by means of the API Streaming de Twitter. Hereby, before the need to develop an analysis of an event, or the desire to elaborate a study or investigation being based in tweets, PARSE4U Tracking makes possible the access to the information obtained in raw.

Índice general

Capítulo 1

Introducción

En la actualidad nos encontramos en una época donde los medios de comunicación y publicidad tradicionales se están quedando obsoletos. Estamos en el comienzo de una era digital donde la sociedad está acostumbrada a un constante bombardeo de información y publicidad por parte de las empresas en los diferentes medios de comunicación, especialmente, utilizando las redes sociales como medio de comunicación. Pero, ¿qué son las redes sociales? Las redes sociales son sitios en Internet formados por comunidades de individuos con intereses actividades o profesiones en común y que permiten la comunicación entre los miembros, de esta manera se pueden intercambiar información y opiniones. Las redes sociales son una fuente inagotable de información, y proporcionan un mecanismo de comunicación entre personas y empresas.

Cualquier empresa actual debería tener perfiles en redes sociales tanto para comunicarse con sus clientes, como para recuperar información relevante para su actividad. El estudio de las redes sociales puede definirse como la captura de datos de diferentes redes sociales para posteriormente ser analizados y tomar ciertas decisiones comerciales, publicitarias, obtener información de clientes o ver la evolución de determinados perfiles de usuarios. Este seguimiento y posterior análisis de los datos no es algo genérico, sino que dependiendo de los objetivos que se deseen evaluar serán necesarias diferentes técnicas, herramientas y enfoques para llegar a unos resultados concluyentes. Llevar a cabo un análisis de los datos en la red social incluye dos procesos que son: captura de datos y el análisis de los datos; en este caso nos centraremos especialmente en la captura de datos, dejando la puerta abierta a posibles clientes que deseen realizar un análisis de los datos.

Si deseamos capturar datos de las redes sociales, uno de los primeros problemas a los que nos enfrentaremos consiste en que los datos están almacenados en los servicios de las redes sociales, pero, ¿cómo podemos acceder a estos datos? Muchas de las redes sociales actuales proporcionan APIs públicas para que los desarrolladores puedan enviar y recibir datos de las redes sociales, por lo que el problema estaría solucionado. Una vez se conoce la existencia de las APIs nos damos cuenta de que depender del estado de los servicios externos, de las conexiones a Internet y de la saturación de los servicios, puede no ser una buena idea, lo que nos hace pensar que la mejor solución sería que los datos que se desean analizar estén en nuestra posesión y dentro de nuestro dominio; y no en servicios externos.

A la hora de trabajar con datos provenientes de una red social, debido al gran volumen de datos del que se dispone podemos considerar que se trabajará con BigData, por lo que nuestra herramienta deberá ser escalable y permitir almacenar grandes volúmenes de datos. Debido a esto nuestra herramienta estará desarrollada basándose en una arquitectura empresarial y hará uso de

una base de datos no relacional. De esta manera se garantiza la escalabilidad y la posibilidad de trabajar con BigData.

El uso de las APIs públicas y la necesidad de poseer los datos generados en las redes sociales son las motivaciones principales de este proyecto centrado en el desarrollo y exposición de servicios para la captura y recuperación de estos datos. De esta manera se capturarán los datos que se deseen en función de determinados filtros y se obtendrá la información concreta, se almacenarán los datos en nuestra propia base de datos, y se crearán unos servicios para permitir que clientes externos consulten nuestros datos y puedan realizar sus propios análisis.

1.1. Motivación

Tras un estudio del mercado, comprobamos que existen infinidad de aplicaciones web o herramientas que permiten realizar análisis de redes sociales, pero todas ellas tienen una gran carencia que este trabajo quiere suplir; ninguna de estas herramientas permite el acceso a los datos obtenidos de las redes sociales. Estas herramientas permiten realizar estadísticas y gráficas basándose en datos provenientes de redes sociales, pero no permiten el acceso a los datos utilizados para realizar los análisis, ni permiten almacenar los datos para poder cruzarlos con datos de otros análisis. Esta herramienta busca suplir las carencias que todas las herramientas disponibles en la web tienen.

En la actualidad, se utilizan las redes sociales como medio de obtención de datos para multitud de estudios. En este caso, dada la potencia de Twitter para obtener información de sus usuarios, así como, la facilidad que existe para recuperarla a través de la API, permite programar algoritmos encargados de recuperar dicha información.

Aunque esta idea suene algo confusa, a través del sentimiento de un "Tweet", podemos determinar el futuro de ese usuario y de sus seguidores. Cuando decimos que podemos determinar o predecir su futuro y el de sus "Followers", nos referimos a que, en base a la manera en que está escrito el "Tweet", se puede determinar la tendencia política, anímica o economice de los usuarios.

Pero, ¿qué es lo que pasa cuando en vez de analizar personas, analizamos "Tweets" empresariales?. Con este tipo de "Tweets", no se pueden predecir elecciones políticas ni posibles ataques terroristas, pero si se pueden predecir los diferentes estados bursátiles de la bolsa.

A continuación, se puede ver algunos de los estudios que se han realizado utilizando "Twitter" como origen de la información.

- **Rastrear la propagación de una enfermedad**: Gracias a la investigación de Adam Sadilek, como parte del equipo de Henry Kautz, en la universidad de Rochester y de su algoritmo, el cual forma parte del proyecto "GermTracker"; es posible seguir la evolución de una epidemia, en este caso de la gripe, a través del estudio de "Tweets".[16]
- **Futuro de la bolsa**: Una de las predicciones más importantes, enfocadas hacia el mundo de la economía, en 2010, Johan Bollen, Huina Mao y Xiao-Jun Zeng, fueron capaces de predecir, con una precisión del 86,7 % el *Down Jones Industrial Average* con tres días de antelación y basándose en el análisis de 9,8 millones de "Tweets". Este tipo de predicción, hizo que las empresas dedicaran muchos recursos, no solo por el hecho de saber cómo era la tendencia del mercado más allá de los patrones impuestos por Bloomberg, sino, especialmente para saber cuáles eran los valores de otros mercados. [6]

- **Estados emocionales**: Otro de los casos prácticos en los que las predicciones de Twitter, son válidas, es en el campo de la medicina. Mediante el análisis de los sentimientos de los "Tweets" de un paciente, se puede llegar a saber su estado anímico, psicológico e incluso poder predecir recaídas o actos que pongan en peligro la salud de dicho paciente o de los que la rodean. [9]
- **Seguimiento de intoxicaciones alimenticias en restaurantes**: Otro de los algoritmos de Adam Sailek, es el llamado "nEmesis", este algoritmo se encarga de analizar en qué restaurantes de ha intoxicado más gente. Esta variante del algoritmo de prevención de la gripe, y utilizando la geolocalización como parte imprescindible del algoritmo, es capaz de predecir los restaurantes con mayor probabilidad de servir alimentos en mal estado. Básicamente, el algoritmo analiza palabras del tipo dolor de tripa, comida mala, mal estado, ... y la localización del mismo para poder determinar el área donde se encuentra el restaurante, así como el nombre del mismo. [15]

Estos entre otros usos como: tendencias e intereses políticos de la población, ganador de las elecciones o predicción de atentados terroristas son unos de los muchos estudios que se pueden realizar utilizando la red social de Twitter como fuente de información.

Pero, ¿cómo conseguimos esos datos de Twitter? para cada uno de los estudios anteriormente enunciados, se tuvieron de desarrollar diversas herramientas para permitir la captura de datos de Twitter y poder acceder a los datos en bruto de un "Tweet". En este caso, nuestra herramienta proporciona un método rápido, fácil y cómodo de recolección de datos. PARSE4U Tracking permitirá acceder a los datos en bruto obtenidos de las redes sociales para poder realizar análisis propios o cualquier operación de transformación sobre los datos para utilizarlos en otras herramientas. De esta manera, nuestra herramienta se encargará de capturar los datos de las redes sociales, utilizando los filtros que el usuario crea conveniente. Así mismo, se proporcionará un servicio para posibilitar la consulta de estos datos a clientes externos.

De esta forma, si cualquier persona desea realizar algún tipo de estudio basándose en datos de Twitter, solo tendría que: acceder a nuestra herramienta, configurar un seguimiento con las restricciones que se deseen y esperar a obtener el número de datos deseados. Una vez obtenido el volumen de dato deseado, se permite el acceso a los datos a través de un servicio web y así poder alimentar sus algoritmos con los datos en bruto obtenidos.

1.2. Objetivos y Alcance del Sistema

Los objetivos principales que esta herramienta busca son:

1. Posibilitar la captura y almacenamiento de los datos de Twitter para su posterior consulta y análisis.
2. Posibilitar el acceso a los datos capturados.

La consecución de los objetivos anteriores se materializará en un producto software que permita capturar datos de Twitter basándose en determinados filtros, de una manera rápida y fácil; y permitirá la consulta directa sobre estos datos para ser consumidos por clientes externos. Para ello, el software será una aplicación alojada en un servidor de aplicaciones, accesible vía Web; que permitirá aislar la captura de datos del análisis de los datos, permitiendo que los clientes externos se centren en el análisis y no en la captura.

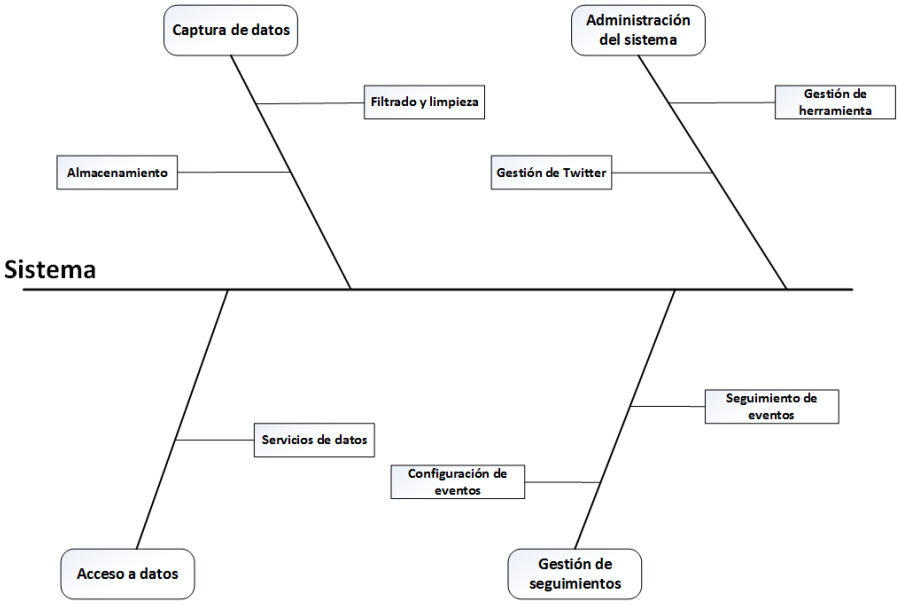


Figura 1.1: Árbol de características

Como se puede ver en la Figura: 1.1 , el sistema presentará cuatro características principales:

- La característica encargada de captura de datos se encargará de recibir los datos desde Twitter, filtrándolos y modelándolos para almacenarlos en la base de datos.
- la característica encargada del acceso a datos se encargará de permitir el acceso a los datos a través de servicios.
- La característica encargada de gestión de seguimientos será el encargado de permitir al usuario configurar los diferentes seguimientos que se realizaran, así como de los filtros y configuraciones.
- La característica de administración del sistema permitirá gestionar las cuentas de la API de Twitter y los usuarios y configuraciones de la herramienta.

1.3. Estado del Arte

Tras haber realizado un análisis del estado del arte, no hemos encontrado ninguna herramienta que permita almacenar y consultar los datos capturados de una red social. Si existen múltiples herramientas que permiten realizar análisis de las redes sociales; algunas de estas herramientas son:

- Followthehashtag.com: Es una aplicación web desarrollada por una empresa española con la que podemos realizar el seguimiento de eventos, marcas, usuarios en Twitter según se indique en el filtro de búsqueda. Uno de lo aspectos más destacados de esta herramienta es la generación de informes sobre una determinada búsqueda. Por otra parte, como aspecto negativo, podemos destacar que no permite la exportación de los datos que han sido capturados durante el análisis, sino que únicamente son presentados al usuario por pantalla, como se puede ver en la Figura: 1.2, en forma de gráficos, estadísticas, mapas como el mostrado en la Figura: 1.3 y clasificaciones de las diferentes publicaciones.
- Talkwalker.com: Talkwalker es otra aplicación web de monitorización y análisis en redes sociales como se muestra en la Figura: 1.4 que, a diferencia de Followthehashtag, realiza dicho

análisis en diversas redes además de Twitter como Google+, Instagram, YouTube, Facebook, Noticias o Blogs. Esta capacidad de análisis permite obtener una mayor información, aunque también más diversas y general que la que tuviese si se centrase en una única red. Al igual que en Followthehashtag, no se permite la opción de poder exportar directamente los datos capturados por la herramienta para poder ser analizados con otras aplicaciones.

- Keyhole.co : Es otra aplicación para seguimiento en Twitter e Instagram y su posterior análisis de los datos, como podemos ver en la Figura: 1.5. No podremos utilizar esta aplicación de una forma exhaustiva ya que la versión de prueba está muy limitada, además de que únicamente la proporcionan por tres días.

Existen otras aplicaciones como sumall.com en la Figura: 1.6 o followerwonk.com en la Figura: 1.7 que también permiten el análisis en Twitter, pero están más centradas hacia el seguimiento de cuentas de usuario.

Figura 1.2: Nube de palabras en FollowTheHashtag

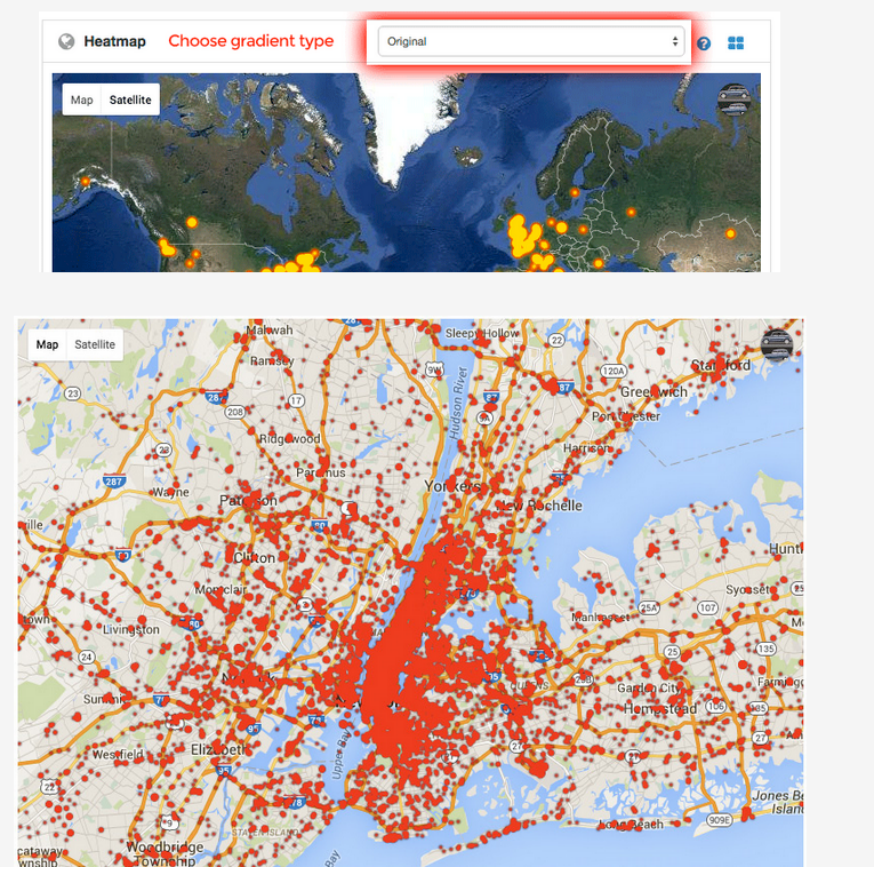


Figura 1.3: Análisis geográfico en FollowTheHashtag

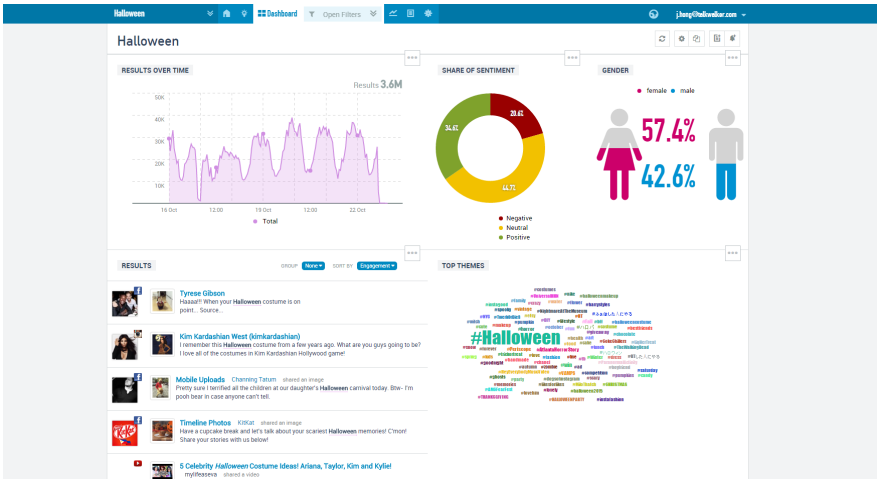


Figura 1.4: Resumen seguimiento en TalkWalker

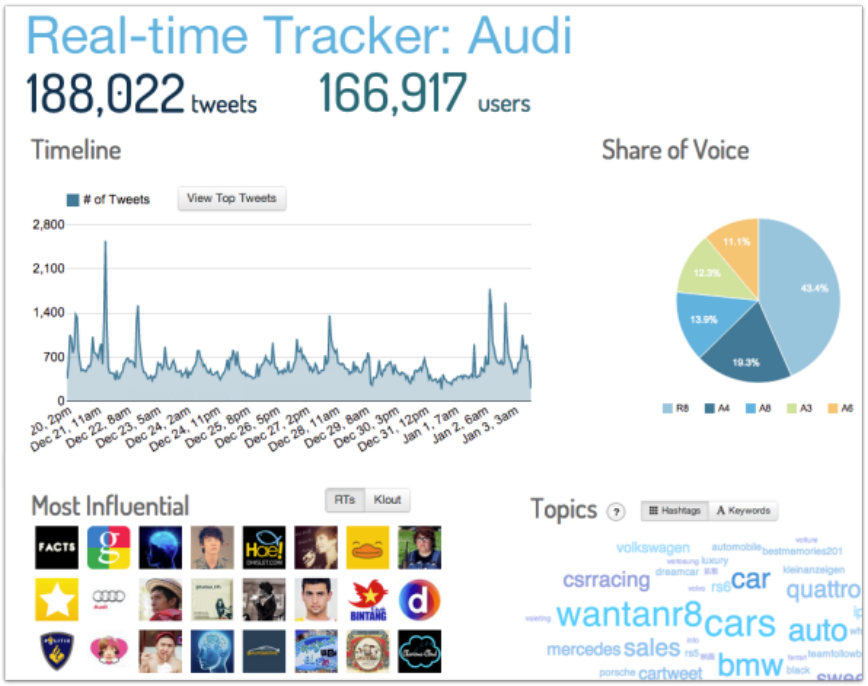


Figura 1.5: Resumen de evento en KeyHole.co

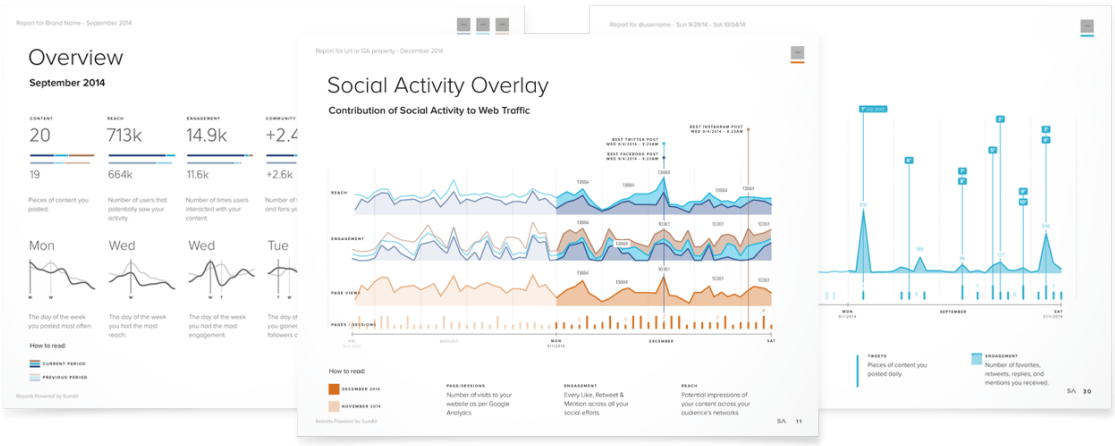


Figura 1.6: Resumen de un seguimiento en Sumall

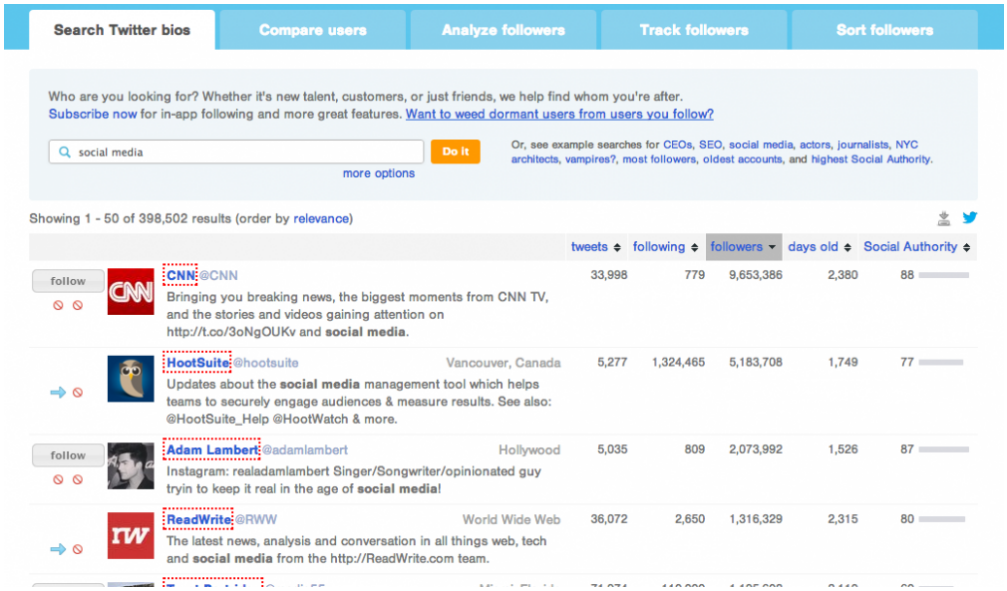


Figura 1.7: Resumen de un seguimiento en FollowerWonk

1.4. Contenido del CD-ROM

Se entregará un disco que contenga el código fuente del programa y una copia de esta memoria.

El contenido del CD-ROM se ha estructurado en los siguientes directorios:

- **Diagramas:** En este directorio se encuentran el diagrama de casos de uso, los diagramas de secuencia, los diagramas de clases y el árbol de características en formato PNG de alta resolución.

- **Proyecto:** En este directorio se encontraran los ficheros del proyecto desarrollado.

  Código fuente: En este directorio se encontrarán los ficheros que componen el proyecto con el código fuente.

  Despliegue: En este directorio se encontrará el fichero ".war" necesario para desplegar la aplicación en un servidor de aplicaciones.

  Dominio: En este directorio se encuentra una copia del dominio del servidor de aplicaciones para poder cargar esta copia de seguridad en otro servidor de aplicaciones. De esta manera, la configuracion de JAAS y las optimizaciones del servidor ya se encuentras en la copia de seguridad.

  Base de datos: en este directorio se encuentran los ficheros necesarios para cargar la base de datos con la estructura de colecciones y unos datos mínimos para su funcionamiento.

- **Capturas de pantalla:** En este directorio se encuentran las capturas de pantalla utilizadas en el manual de usuario.

- **Memoria TFG**: En este directorio se encuentra una copia en formato PDF de este documento.

1.5. Organización del Documento

En esta sección se va a describir la estructura del documento de modo que sirva de ayuda al lector. El presente documento se divide en los siguientes capítulos.

- **Capítulo 1. Introducción** En este capítulo encontramos una presentación al proyecto realizado, y se especifica la motivación para llevarlo a cabo. Además, se describirá el estado del arte donde se podrá obtener información de las herramientas similares que ya existen en el mercado y de ver que características diferencias el proyecto PARSE4U de las existentes.
- **Capítulo 2. Twitter Analytics** En este capítulo se explica el motivo de utilizar "Twitter" y no otra red social para capturar datos, las características que nos ofrece la "API de Twitter" y las posibilidades que existen para interactuar con estos servicios web.
- **Capítulo 3. Gestión del proyecto** Este capítulo aborda el presupuesto para este proyecto (estimación de costes económicos y temporales tanto reales como estimaciones previas).
- **Capítulo 4. Análisis del Sistema** Este capítulo detalla el dominio del proyecto PARSE4U Tracking y que se intenta solventar con la herramienta desarrollada. Se presentarán los requisitos y casos de uso obtenidos en el análisis del sistema software.
- **Capítulo 5. Diseño Software** En este capítulo se muestra la arquitectura lógica y física que presentará la herramienta PARSE4U Tracking. Además, este capítulo incluye detalles de diseño de los componentes más importantes del proyecto.
- **Capítulo 6. Implementación del Proyecto** En esta parte del documento se describe la implementación que se ha llevado a cabo para la elaboración del proyecto.
- **Capítulo 7. Pruebas** En este capítulo se detallan las pruebas que se han realizado para asegurarse que el proyecto funcione correctamente en base a los requisitos establecidos.
- **Capítulo 8. Manuales del Sistema** En este apartado, se incluyen instrucciones para poner en marcha el proyecto con las instalaciones necesarias; junto con un documento que especifica el funcionamiento y las tareas que se han de llevar a cabo para el uso de la herramienta.
- **Capítulo 9. Conclusiones y Trabajo Futuro** En este capítulo se hace una pequeña valoración del trabajo realizado, así como las posibles mejoras que se podrían llevar a cabo en el futuro.
- **Capítulo 10. Glosario** En esta seccion, se mostrará la terminología utilizada en este documento, así como, una breve definición de cada término.
- **Anexo A Casos de Uso** En este anexo, se mostrarán todos los casos de uso del sistema, así como una descripción de cada uno.

Capítulo 2

Twitter Analytics

En la actualidad las redes sociales son el centro de comunicación entre las personas, y tienen un crecimiento exponencial en los usuarios activos llegando en el caso de Facebook a contar con 1590 Millones de usuarios registrados. Si tenemos en cuenta que Europa tiene 742 Millones de habitantes, Facebook tiene el doble de usuarios registrados. Si observamos la tabla 2.1 de las 11 redes sociales más utilizadas en el mundo tenemos:

Red social	Nº usuarios (Millones)	Tipo
Facebook	1.590	General
YouTube	1.000	Vídeos
Whatsapp	1.000	Mensajería
QQ	829	Mensajería
Qzone	629	General
WeChat	550	Mensajería
Instagram	400	Foto / Vídeo
Weibo	400	Mensajería
Linkedin	400	Negocios
Twitter	350	Mensajería
Google+	343	General

Figura 2.1: Usuarios registrados en las 11 primeras redes sociales año 2015

¿Y si sumamos los usuarios registrados en estas 11 redes sociales? Nos encontramos con que las 11 primeras redes sociales en el año 2011 tienen un total de 7500 millones de usuarios registrados superando ligeramente la población mundial en el año 2015 que era de 7300 millones de personas.

Debido al auge de las redes sociales y al aumento exponencial de los usuarios activos de estas, nace el Social Media. Se conoce al Social Media como las plataformas online donde el contenido es creado por los propios usuarios usando las tecnologías de la Web 2.0, que facilitan la edición, la publicación y el intercambio de información. Si consideramos el análisis de las redes sociales

como un término que se usa para el proceso de recopilar y consolidar datos en bruto proveniente de redes sociales como Facebook o Twitter; y analizarla para apoyar la planificación y en la toma de decisiones.

En redes sociales como Facebook o Twitter tienen lugar cada día millones de conversaciones. Estas conversaciones contienen una gran cantidad de información proveniente de los clientes o de los potenciales clientes. Los usuarios comparten buenas y malas experiencias relacionadas con determinadas empresas, expresan opiniones sobre los productos, muestran sus necesidades y deseos respecto a futuros productos... Para las empresas, este flujo de conversaciones, representa una oportunidad para escuchar, determinar el impacto comercial, y aprovechar el nuevo conocimiento para influir en la imagen pública de la empresa, satisfacción de los clientes, etc. Con el análisis de las redes sociales, las organizaciones pueden conocer a sus clientes de una forma nunca antes conocida.

Al realiza un análisis de las redes sociales, no solo se debería tener en cuenta una red social, sino que se deberían realizar análisis de diversas redes sociales (Facebook, Twitter, Google+...), así mismo, se deberían realizar análisis de los foros, blogs o cualquier forma de comunicación que tengan los clientes con la empresa, utilizando la web. En este caso, nuestra herramienta está destinada a capturar datos que provengan de la red social Twitter.

2.1. ¿Por qué Analizar Twitter?

La idea de negocio que promociona esta herramienta podría ser válida para cualquier red social, como Facebook o Google+. Pero en este caso se ha decidido utilizar Twitter por diversos motivos: Twitter es una red social abierta, esto significa que no es necesario establecer una relación mutua entre dos usuarios para poder ver sus publicaciones; Tiene 350 millones de usuarios registrados de los que puedes obtener información; si utilizáramos Facebook, solo podemos obtener información de los usuarios que somos "amigos", en este caso el perfil con mayor número de "amigos" es Shakira con 89 millones, muy lejos de los 350 millones de usuarios a los que tendríamos alcance en Twitter. Al igual que Facebook, Twitter proporciona una API para poder consumir los datos de la red social; la API permite tanto leer las publicaciones como publicar contenido en la red social.

Al utilizar la API que Twitter ofrece para acceder a sus servicios, tenemos la posibilidad tanto de recibir y leer "Tweets", como la posibilidad de enviar "Tweets" y mensajes directos; en este caso nos centraremos solo en la posibilidad de recibir "Tweets". Para capturar estos "Tweets", la API ofrece la posibilidad de filtrar por usuario que lo envió, palabras utilizadas en el texto, "Hashtag" utilizado entre otras muchas opciones. Cuando un "Tweet" cumple los requisitos que se establecen en la API, Twitter retorna a través de su servicio el "Tweet" completo. Cuando hablamos de un "Tweet" completo, nos referimos, no solo al texto que el usuario ha enviado en la red social; sino que nos referimos a todos los metadatos relacionados con ese "Tweet". Como se puede ver en la Figura (2.2), existen múltiples atributos relacionados con cada uno de los "Tweets" que se capturan con la herramienta. Estos atributos o metadatos pueden ser de especial interés a la hora de hacer un análisis de las redes sociales. En función del tipo de estudio que se desee realizar, ciertos atributos tendrán una mayor importancia que otros, por eso nuestra herramienta leerá y almacenará toda la información para su futuro análisis y estudio.

Atributo	Descripción
Coordinates	Representa la localización geográfica del usuario cuando se envió el tweet
Favorite_count	Número de favoritos que ha recibido el tweet
Retweet_count	Número de retweets que ha recibido el tweet
Lang	Lenguaje del tweet
Source	Indica desde que plataforma se envió el tweet (IOS, Android, PC, Web...)
Created_at	Indica la fecha de creación de la cuenta de Twitter que envió el tweet
Followers_count	Indica el número de seguidores que tiene el usuario que creó el tweet
Listed_count	Nos muestra el número de veces que se ha publicado el tweet en el muro de un usuario
Profile_background_image_url	Dirección URL de la imagen de perfil del usuario
Hashtags	Hashtags utilizados en el tweet
Media	Propiedades y direcciones URL de los ficheros multimedia incrustados en el tweet
Bounding_box	Coordenadas GPS que forman un polígono que define la ubicación del usuario

Figura 2.2: Atributos más relevantes de un "Tweet"

2.2. API-Twitter

Si deseamos desarrollar una aplicación que envíe o reciba datos de Twitter, la red social nos proporciona de unas APIs para conectar nuestro software con sus servicios. En este caso Twitter nos posibilita el acceso a dos tipos de APIs, la API Rest que nos permitirá enviar "Tweets", buscar y leer, aunque con ciertas restricciones; también nos brinda una API de streaming, que nos posibilita solo el acceso a leer los "Tweets" que se publican en toda la red social, aunque con limitaciones mucho menos restrictivas que la API Rest. Anteriormente, Twitter proporcionaba una API llamada Search solo para hacer búsquedas de datos, pero estas características se han trasladado dentro de la API Rest y todas sus funcionalidades siguen disponibles, pero utilizando llamadas Rest. Si deseáramos crear una aplicación para dispositivos móviles que realice las funciones de un cliente de Twitter utilizaríamos la API Rest para poder enviar y recibir la información; en este caso nuestro enfoque se centra en poder capturar y leer mensajes de cualquier usuario de Twitter, por lo que debemos acceder utilizando la API de streaming.

API Rest La API Rest de Twitter nos permite leer y escribir datos en Twitter, publicar "Tweets", leer el perfil del autor, los datos de los seguidores , etc. La API Rest autentica a los usuarios de Twitter mediante el protocolo OAuth. Actualmente la API Rest se encuentra en la versión 1.1, por lo que con el paso del tiempo es probable que las características de esta se vean afectadas. Este servicio tiene varias limitaciones en cuanto al uso que se le puede dar; establece límites para el número de consultas por minuto que puede realizar una aplicación o usuario, número de búsquedas por minuto; en la Figura (2.3) se puede ver los límites tanto por aplicación como por usuario que establece Twitter para la API Rest.

Si queremos hacer uso de esta API nuestra aplicación deberá estar registrada en Twitter para poder autenticarse con los servicios. El envío y la recepción de información se realiza mediante llamadas Rest a los servicios que la red social proporciona. En el caso de hacer una llamada PUT, el servicio nos responderá con distintas cabeceras HTTP y diferentes respuestas en función de la

Title	Resource family	Requests / 15-min window (user auth)	Requests / 15-min window (app auth)
GET application/rate_limit_status	application	180	180
GET favorites/list	favorites	15	15
GET followers/ids	followers	15	15
GET followers/list	followers	15	30
GET friends/ids	friends	15	15
GET friends/list	friends	15	30
GET friendships/show	friendships	180	15
GET help/configuration	help	15	15
GET help/languages	help	15	15
GET help/privacy	help	15	15
GET help/tos	help	15	15
GET lists/list	lists	15	15
GET lists/members	lists	180	15
GET lists/members/show	lists	15	15
GET lists/memberships	lists	15	15
GET lists/ownerships	lists	15	15
GET lists/show	lists	15	15
GET lists/statuses	lists	180	180
GET lists/subscribers	lists	180	15
GET lists/subscribers/show	lists	15	15
GET lists/subscriptions	lists	15	15
GET search/tweets	search	180	450
GET statuses/lookup	statuses	180	60
GET statuses/retweeters/ids	statuses	15	60
GET statuses/retweets/:id	statuses	15	60
GET statuses/show/:id	statuses	180	180
GET statuses/user_timeline	statuses	180	300
GET trends/available	trends	15	15
GET trends/closest	trends	15	15
GET trends/place	trends	15	15
GET users/lookup	users	180	60
GET users/show	users	180	180
GET users/suggestions	users	15	15
GET users/suggestions/:slug	users	15	15
GET users/suggestions/:slug/members	users	15	15

Figura 2.3: Limite de API Rest Twitter

operación solicitada. En el caso de hacer una llamada GET, el servicio nos retornara un mensaje JSON con la información solicitada.

API Streaming Esta API nos permite acceder a los "Tweets" publicados por los usuarios de todo el mundo, sin la necesidad de que la cuenta de Twitter a la que está asociada la aplicación tenga un vínculo con los usuarios que publican. Como nuestra herramienta no necesita publicar mensajes en Twitter ni realizar búsquedas concretas en la red social, la mejor opción para el objetivo de la herramienta es la API de Streaming.

La API de Streaming necesita establecer una conexión HTTP permanente con el servicio de Twitter, por lo que en el momento en que un usuario publica un Tweet es recibido por la API de Streaming. En teoría mediante la API de Streaming se puede acceder a todo el flujo de datos de Twitter en tiempo real, pero Twitter en su documentación aclara que, si los filtros de captación de datos de la API de Streaming no son suficientemente restrictivos, en alguna ocasión puede limitar

el flujo de datos recibidos. A efectos prácticos, esto significa que realmente no podemos capturar todo el flujo de datos mundial de Twitter, pero si establecemos ciertos criterios para filtrar los datos, podremos obtener todos los "Tweets" que cumplan dichas condiciones.

2.3. Librerías Disponibles

Tanto si queremos utilizar la API Rest como la API Streaming, necesitamos de ciertas librerías para poder conectarnos con los servicios de Twitter. Existen dos librerías que han sido construidas y fabricadas por Twitter para desarrollos en Java, y Android estas librerías son:

- HBC (Hosebird Client): orientada a la conexión con la API de Streaming desde Java, es de código libre y gratuita. Su código fuente se puede encontrar en: GitHub-HBC. Esta librería implementa múltiples funcionalidades para recibir los datos; tales como, soporte a compresión de datos, autenticación OAuth, soporte a particionado de datos, reconexión automática, acceso a estadísticas relevantes.
- Twitter Kit for Android: orientada a la conexión con la API Rest para crear un cliente de Twitter en plataformas Android.Tambien es de código libre y de uso gratuito; su código fuente se puede encontrar también el GitHub-Twitter-Android-Kit

Ambas librerías están distribuidas bajo una licencia Apache 2.0, lo que implica que se requiere la conservación de los derechos de autor y el descargo de responsabilidad; pero puede modificarse y distribuirse bajo otro tipo de licencias. De igual manera se pueden encontrar múltiples librerías para diferentes lenguajes de programación, estas han sido desarrolladas por terceros y no tienen el soporte oficial de Twitter. Algunas de las más destacadas son:

- Multiplataforma

  Temboo: es un Framework funcional para IOS, Android, Java, PHP, Python, Ruby y NodeJS. Se puede encontrar en: Teemboo [1]

- ASP

  Asptwitter: Liberia para integrar Twitter en una página web ASP, incluye soporte para la última versión de las API de Twitter 1.1 Se puede encontrar en: Asptwitter [2]

- C++

  Twitcurl: es una librería para ser utilizada en C++. utiliza cURL para las peticiones y respuestas HTTP. Ha sido probada en Windows, Ubuntu, Debian. De hecho, debería funcionar en cualquier sistema operativo que soporte cURL. Se puede encontrar en: twitcurl [3]

- Clojure

  Twitter-API: se trata de un envoltorio para la API de Twitter, que funciona sobre Clojure http.async.client Implementa toda la taxonomia dela API de Twitter como Rest, Streaming y búsqueda. Se puede encontrar en: Twitter-API [4]

[1] Temboo: https://temboo.com/library/Library/Twitter/
[2] Asptwitter: http://www.timacheson.com/Blog/2013/jun/asptwitter
[3] Twitcurl: https://github.com/swatkat/twitcurl
[4] Twitter-API: https://github.com/adamwynne/twitter-api

- .NET

  LinqToTwitter: Librería de código abierto para .NET, utiliza la tecnología LINQ para consultas. Se puede encontrar en: LinqToTwitter [5]

  Spring.NET Social extensión for Twitter: es una extensión de Spring para .NET Se puede encontrar en: Spring .NET Social for Twitter [6]

  TweetSharp: librería en .NET para conexión con Twitter. Tiene soporte para Windows Phone. Se puede encontrar en: TweetSharp [7]

  Tweetinvi: librería basada en .Net su misión consiste en permitir una fácil conexión con twitter. Está enfocada principalmente para la API Streaming, pero también implementa las funcionalidades de Rest. Puede encontrarse en: Tweetinvi [8]

  Crafted Twitter: librería desarrollada en .NET con implementaciones para formularios y soporte para MVC. Destinada a incluir "Tweets" en páginas web. Esta librería puede encontrar en: Crafted [9]

- Java

  Twitter4j: es una librería desarrollada para Java, implementa todas las funcionalidades de la API de Twitter, tanto Rest como Streaming. Se distribuye bajo una licencia Apache 2.0 Esta integrado con Maven, se puede localizar en: Twitter4j. [10]

- Javascript / node.js

  TwitterJSClient: librería desarrollada en NodeJS, implementa las funcionalidades de la API rest. Puede localizarse en: TwitterJSClient [11]

  User-Stream: librería desarrollada en NodeJS, implementa las funciones de la API de Streaming. Puede encontrarse en: User-stream [12]

- Objective-C

  STTwitter: librería implementada en Objetive-C, usada para el desarrollo de clientes de "Twitter" para IOS. Implementa las funcionalidades de la API Rest, puede encontrarse en: STTwitter Se distribuye bajo una licencia BSD-3 [13]

  FHSTwitterEngine: librería desarrollada en Objetive-c para ser utilizada en el desarrollo de aplicaciones en Cocoa para IOS. Puede encontrarse en: FHSTwitterEngine [14]

- PHP

  TmhOAuth: Esta librería desarrollada para PHP se centra en permitir el uso del protocolo OAuth para conectarse con Twitter. Se distribuye bajo una licencia Apache 2.0, y su código puede encontrarse en: tmhOAuth [15]

  140dev Twitter Framework: es un framework completo para ser utilizando en PHP, este framework está compuesto por dos partes, un módulo que se encarga de capturar los datos

[5] LinqToTwitter: https://github.com/JoeMayo/LinqToTwitter
[6] Spring.NET Social extensión for Twitter: http://www.springframework.net/social-twitter/
[7] TweetSharp: https://github.com/danielcrenna/tweetsharp
[8] Tweetinvi: https://tweetinvi.codeplex.com/
[9] Crafted Twitter: https://github.com/craftedmedia/Crafted.Twitter
[10] Twitter4j: http://twitter4j.org/en/index.html
[11] TwitterJSClient: https://github.com/BoyCook/TwitterJSClient
[12] User-stream: https://github.com/aivis/user-stream
[13] STTwitter: https://github.com/nst/STTwitte
[14] FHSTwitterEngine: https://github.com/fhsjaagshs/FHSTwitterEngine
[15] TmhOAuth: https://github.com/themattharris/tmhOAuth

de la API Streaming de Twitter y se encarga de filtrarlos y almacenarlos en una base de datos MySql; y otro modulo que se encarga de visualizar en una web los "Tweets" capturados. Este framework se puede encontrar en: 140dev Twitter Framework [16]

Codebird: Es una librería en PHP para conectarse a los servicios de Twitter, permite el uso de la API tanto Rest como Streaming. Esta librería se distribuye bajo una versión GNU v3, y su código puede encontrase en: Codebird [17]

Codeigniter: Es una librería para PHP que permite la captura de datos de Twitter utilizando la API de búsqueda y de Streaming. Permite el caching de datos en memoria y utiliza un servidor de bases de datos Mysql. También implementa la autenticación utilizando el protocolo OAuth. Puede encontrarse su código en: Codeigniter [18]

Zend Framework: Es un framework desarrollado para PHP, puede encontrarse en: Zend framework [19]

Freebird-php: es una librería para conectar aplicaciones desarrolladas en PHP, esta librería solo utiliza la autenticación OAuth. Puede encontrarse el código en: freebird-php [20]

Twitter-API-php: es una librería basada en PHP, implementa todas las funciones de la API de Twitter. Se distribuye bajo una licencia MIT, su código puede encontrarse en: twitter-api-php [21]

Existen librerías para otros muchos lenguajes de programación como: Python, Ruby, Go, Lua, Perl. Asi mismo existen librerías exclusivas para la conexión mediante el protocolo OAuth en múltiples lenguajes de programación. Pueden localizarse todas las librerías en:
Twitter-libraries [22]

[16] 140dev Twitter Framework: http://140dev.com/free-twitter-api-source-code-library/
[17] Codebird: https://github.com/jublonet/codebird-php
[18] Codeigniter: https://github.com/ElliottLandsborough/Codeigniter-Twitter-Search-Library
[19] Zend Framework: http://framework.zend.com/
[20] Freebird-php: https://github.com/corbanb/freebird-php
[21] Twitter-API-php: https://github.com/J7mbo/twitter-api-php
[22] Twitter-libraries: https://dev.twitter.com/overview/api/twitter-libraries

Capítulo 3

Gestión del Proyecto

Este capítulo está destinado a la realización de la planificación del proyecto antes de entrar en el análisis del mismo. Aquí se describe la metodología que se va a seguir, la planificación temporal en la que se divide el proyecto y la estimación presupuestaria que costará su desarrollo.

3.1. Metodología

Para el desarrollo del software, se utilizará un modelo de desarrollo incremental. Con ello, se identifica un conjunto de fases que formaran una "iteración", la cual se va a repetir durante el proceso de desarrollo hasta llegar a la versión definitiva. Con esta metodología, conseguimos reducir el coste de realizar cambios y los tiempos de entrega.

Para aplicar este modelo, primero se realiza el análisis inicial del problema y un diseño preliminar. Posteriormente, se procede a iterar de una forma incremental un conjunto de procesos comunes a cada iteración. Como se puede ver en la Figura: 3.1: ,este conjunto de procesos serán:

- **Análisis:** Modificaciones en el análisis del software.
- **Diseño:** Cambios en el diseño respecto a la anterior iteración.
- **Codificación:** Implementación de los cambios realizados.
- **Pruebas:** Pruebas de la implementación realizada en la iteración.

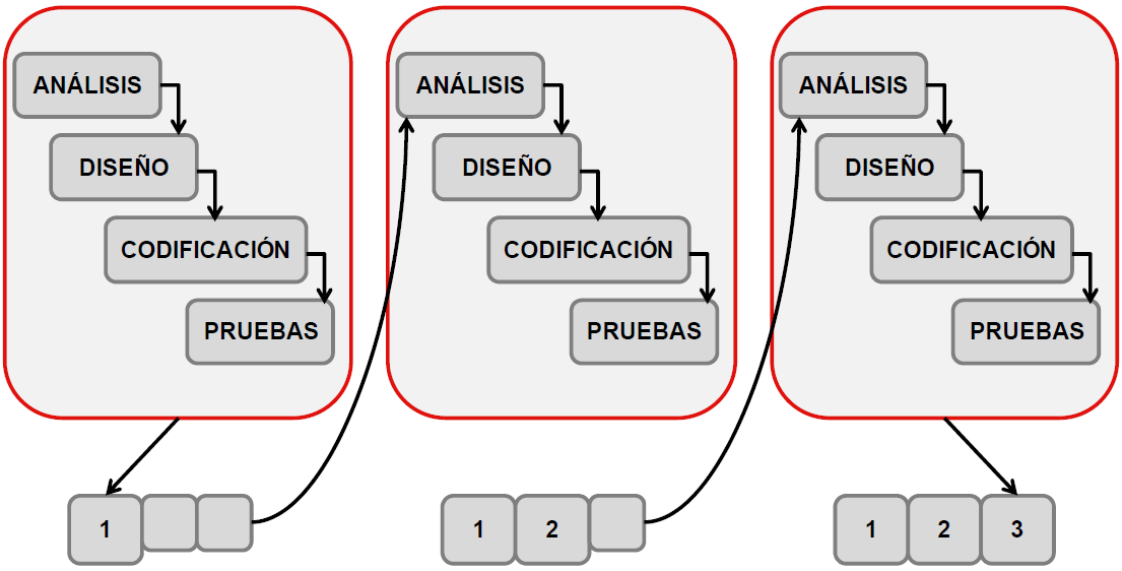


Figura 3.1: Modelo Incremental

3.1.1. Planificación Temporal

Se va a estructurar el trabajo de una forma incremental, dividiendo el tiempo total de 7 meses en 7 incrementos, cada incremento se corresponderá a un mes (diciembre, enero, febrero, marzo, abril, mayo y junio). Cada incremento tendrá una duración mensual. Cada uno de estos estará dividido en 6 etapas: análisis, diseño, implementación, prueba, despliegue y documentación.

Las etapas aparecen descritas a continuación:

- **Análisis de Requisitos:** Durante esta etapa, se intenta explicar lo que debería hacer el software para satisfacer las necesidades de los usuarios que lo utilizarán y se indica cual es la interfaz de usuario más adecuada para el programa. La captura, análisis y especificación de requisitos es una parte crucial: de esta etapa depende en gran medida que el software producido tenga la calidad esperada. Estos requisitos se determinan tomando en cuenta las necesidades a satisfacer del usuario final.
- **Diseño de componentes:** Se decide cómo funcionará, de forma general, el software sin entrar en detalles e incorporando consideraciones de la implementación tecnológica. Generalmente, se realiza en base a diagramas que permiten describir las interacciones entre las entidades y se diseñarán los componentes que darán respuesta a las funcionalidades del software.
- **Implementación:** En esta etapa, se comienza a programar el código para el producto, materializando las etapas anteriores.
- **Prueba de componentes:** Consiste en comprobar que los componentes diseñados funcionen correctamente en el momento en que se van implementando en el sistema. Cada vez que se completa el desarrollo de cada módulo del software, se probará de manera integral para comprobar que el programa funciona correctamente y cumple con los objetivos estipulados.
- **Prueba de aplicación y despliegue:** Consiste en comprobar que el software, con todos sus componentes ya implementados y probados, realice correctamente las tareas indicadas en la especificación de los requisitos. Si las pruebas resultan exitosas, entonces se puede comenzar a desplegar la aplicación a los usuarios finales.

- **Documentación:** En esta etapa se genera la documentación del propio desarrollo software, y la gestión del proyecto, pasando por modelo UML, diagramas de casos de uso, pruebas de usuario... con el objetivo de que dicha documentación pueda ser fácilmente alterada ante posibles correcciones y ampliaciones del sistema.

Nombre de tarea	Duración	Comienzo	Fin
Iteración 1	**30 días**	**mar 01/12/15**	**mié 30/12/15**
Análisis	16 días	mar 01/12/15	mié 16/12/15
Diseño	6 días	jue 17/12/15	mar 22/12/15
Implementación	4 días	mié 23/12/15	sáb 26/12/15
Prueba	2 días	dom 27/12/15	lun 28/12/15
Despliegue	1 día	mar 29/12/15	mar 29/12/15
Documentación	1 día	mié 30/12/15	mié 30/12/15
Iteración 2	**31 días**	**vie 01/01/16**	**dom 31/01/16**
Análisis	12 días	jue 31/12/15	lun 11/01/16
Diseño	8 días	mar 12/01/16	mar 19/01/16
Implementación	6 días	mié 20/01/16	lun 25/01/16
Prueba	2 días	mar 26/01/16	mié 27/01/16
Despliegue	1 día	jue 28/01/16	jue 28/01/16
Documentación	1 día	vie 29/01/16	vie 29/01/16
Iteración 3	**29 días**	**lun 01/02/16**	**lun 29/02/16**
Análisis	8 días	sáb 30/01/16	sáb 06/02/16
Diseño	6 días	dom 07/02/16	vie 12/02/16
Implementación	11 días	sáb 13/02/16	mar 23/02/16
Prueba	2 días	mié 24/02/16	jue 25/02/16
Despliegue	1 día	vie 26/02/16	vie 26/02/16
Documentación	2 días	sáb 27/02/16	dom 28/02/16
Iteración 4	**31 días**	**mar 01/03/16**	**jue 31/03/16**
Análisis	6 días	lun 29/02/16	sáb 05/03/16
Diseño	4 días	dom 06/03/16	mié 09/03/16
Implementación	14 días	jue 10/03/16	mié 23/03/16
Prueba	2 días	jue 24/03/16	vie 25/03/16
Despliegue	1 día	sáb 26/03/16	sáb 26/03/16
Documentación	3 días	dom 27/03/16	mar 29/03/16

Nombre de tarea	Duración	Comienzo	Fin
Iteración 5	**30 días**	**vie 01/04/16**	**sáb 30/04/16**
Análisis	4 días	mié 30/03/16	sáb 02/04/16
Diseño	3 días	dom 03/04/16	mar 05/04/16
Implementación	16 días	mié 06/04/16	jue 21/04/16
Prueba	2 días	vie 22/04/16	sáb 23/04/16
Despliegue	2 días	dom 24/04/16	lun 25/04/16
Documentación	3 días	mar 26/04/16	jue 28/04/16
Iteración 6	**31 días**	**dom 01/05/16**	**mar 31/05/16**
Análisis	3 días	vie 29/04/16	dom 01/05/16
Diseño	3 días	lun 02/05/16	mié 04/05/16
Implementación	15 días	jue 05/05/16	jue 19/05/16
Prueba	4 días	vie 20/05/16	lun 23/05/16
Despliegue	1 día	mar 24/05/16	mar 24/05/16
Documentación	4 días	mié 25/05/16	sáb 28/05/16
Iteración 7	**30 días**	**mié 01/06/16**	**jue 30/06/16**
Análisis	1 día	dom 29/05/16	dom 29/05/16
Diseño	2 días	lun 30/05/16	mar 31/05/16
Implementación	6 días	mié 01/06/16	lun 06/06/16
Prueba	5 días	mar 07/06/16	sáb 11/06/16
Despliegue	2 días	dom 12/06/16	lun 13/06/16
Documentación	13 días	mar 14/06/16	dom 26/06/16

Figura 3.2: Planificación temporal

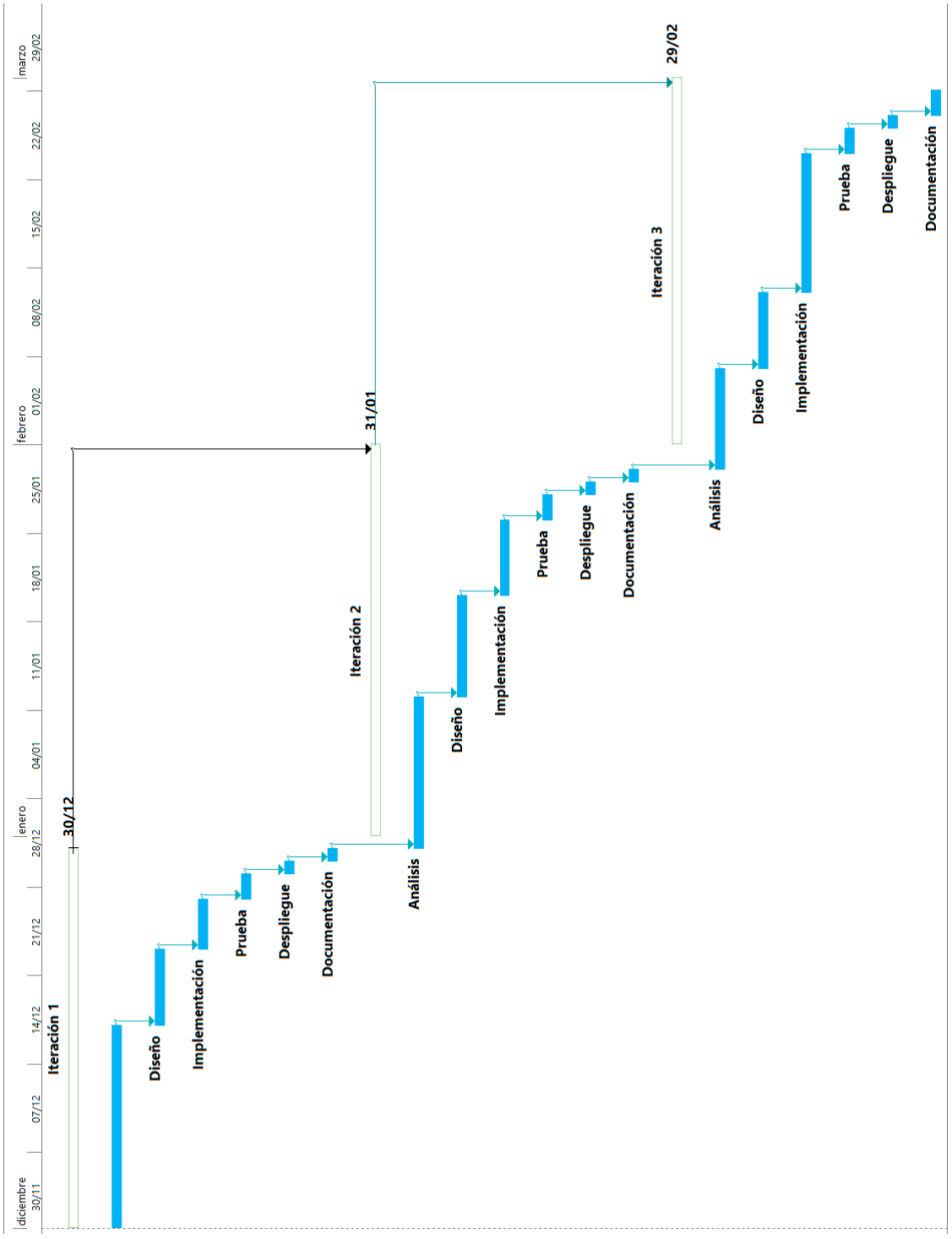


Figura 3.3: Parte 1 Estimación Temporal del Proyecto Utilizando un Diagrama de Gantt

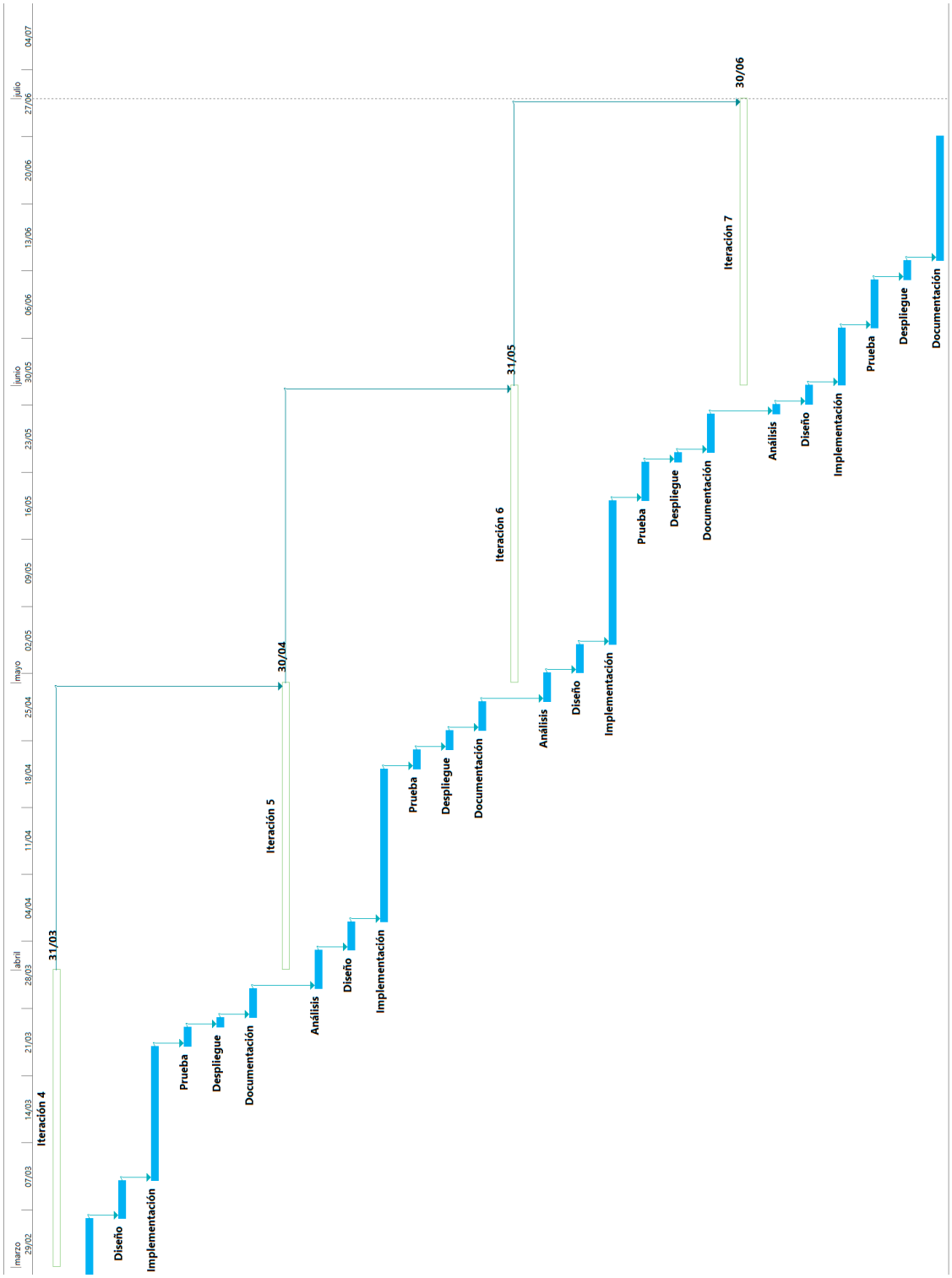


Figura 3.4: Parte 2 Estimación Temporal del Proyecto Utilizando un Diagrama de Gantt

3.2. Estimación de Costes Económicos

Para calcular el coste económico del desarrollo, primero se ha de realizar las estimaciones oportunas. Para realizar estas estimaciones se utilizan dos modelos: el modelo por Puntos de Función (**PF**) y el Modelo Constructivo de Costos (**COCOMO**). Una vez calculadas dichas estimaciones, se procederá a realizar el presupuesto del trabajo.

3.2.1. Estimación de Puntos de Función (PF)

Las métricas del software orientadas a la función son medidas indirectas del software y del proceso a desarrollar, centrándose en la funcionalidad o utilidad del programa. los puntos de función se obtienen utilizando una relación empírica basada en medidas cuantitativas del dominio de información de la aplicación y valoraciones subjetivas de la complejidad del software (Felipe Tijerina,2013).

La estimación de los costes por puntos de función, se trata de un tipo de estimación para medir el tamaño de un sistema en unidades independientes del lenguaje de programación, las metodologías o tecnologías utilizadas para su desarrollo. a continuación, se detalla el procedimiento a seguir:

1. Se definen los dominios de información y su complejidad:
   - Número de entradas (Datos que el usuario aporta al sistema: nombre de ficheros, menú de selección...)
   - Número de salidas (Datos que el sistema aporta al usuario: mensaje, informes, pantallas...)
   - Número de consultas externas (entradas que requieren de una respuesta por parte del sistema)
   - Número de ficheros lógicos internos (ficheros que sólo utiliza el sistema: ficheros, bases de datos ...)
   - Número de ficheros lógicos externos (Ficheros que pueden ser utilizados por otros sistemas: ficheros, bases de datos ...)
2. Se contabiliza el número de elementos de cada clase y según su complejidad (baja, media, alta).
3. Se obtienen los puntos de función no ajustados (**PFNA**) mediante la suma ponderada de las cantidades anteriores con los pesos de la Figura:3.5.
4. Una vez obtenidos los PFNA, éstos deben ser ajustados mediante un factor de ajuste (FA). Este factor se obtiene mediante la suma de 14 factores de complejidad (FC). A cada factor de complejidad se le atribuye un peso entre 0 y 5 que indica su grado de complejidad.

$$FA = 0,65 + (0,1 * \sum Complejidad) \tag{3.1}$$

5. Después de obtener el factor de ajuste (FA), éste se aplica a los puntos de función no ajustados (PFNA) para obtener los puntos de función ajustados (PFA)
6. Por ultimo, para obtener una estimación de las líneas de código, se utiliza una tabla de equivalencias entre los puntos de función y los lenguajes de programación. Puesto que, para el desarrollo del trabajo, solo se han utilizado ciertos lenguajes, en la tabla sólo se muestran dichos lenguajes. Los valores de la tabla:3.1 han sido obtenidos de la página web de QSM.

$$PFNA = \big(Nº\ Entradas\ x\ multiplicador\ (complejidad)\big) + \big(Nº\ Salidas\ x\ multiplicador\ (complejidad)\big) + \big(Nº\ Ficheros\ internos\ x\ multiplicador\ (complejidad)\big) + \big(Nº\ Ficheros\ externos\ x\ multiplicador\ (complejidad)\big) + (Nº\ Consultas\ externas\ x\ multiplicador\ (complejidad))$$

Dominio de información	Complejidad baja	Complejidad media	Complejidad alta
Entradas	×3	×4	×6
Salidas	×4	×5	×7
Consultas externas	×3	×4	×6
Ficheros lógicos internos	×7	×10	×15
Ficheros lógicos externos	×5	×7	×10

Figura 3.5: Pesos de los dominios de información según su complejidad

Lenguaje	LDC/PF
HTML	34
J2EE	46
JavaScript	47
Java	53

Tabla 3.1: Líneas de código por PF de cada lenguaje de programación

Estimación de PF del Proyecto

- **Número de entradas**:
  - Formulario de login: complejidad media.
  - Menús de navegación: complejidad media.
  - Formulario de creación de eventos: complejidad baja.
  - Formulario de creación de cuentas de Twitter: complejidad baja.
  - Formulario de creación de api key de Twitter: complejidad media.
  - Formulario de actualización de datos de usuario: complejidad media.
  - Formulario de registro de usuarios: complejidad baja.
  - Formulario de modificación de usuarios: complejidad baja.
- **Número de salidas**:
  - Listado de eventos: complejidad baja.
  - Listado de Cuentas de Twitter: complejidad baja.
  - Listado de API Keys: complejidad media.
  - Mensajes de error: complejidad baja.

- **Número de ficheros lógicos internos**:
  - Base de datos de la aplicación: complejidad alta.
  - base de datos de Tweets: complejidad baja.
- **Número de ficheros lógicos externos**:
  - Conexión Api Streaming de Twitter: complejidad alta.

Dominio de informacion	Complejidad	Total x Complejidad	Suma
Entradas	Baja	4x3	24
Entradas	Media	3x4	
Entradas	Alta	0x6	
Salidas	Baja	3x4	17
Salidas	Media	1x5	
Salidas	Alta	0x7	
Consultas externas	Baja	0x3	0
Consultas externas	Media	0x4	
Consultas externas	Alta	0x6	
Ficheros lógicos internos	Baja	1x7	22
Ficheros lógicos internos	Media	0x10	
Ficheros lógicos internos	Alta	1x15	
Ficheros lógicos externos	Baja	0x15	10
Ficheros lógicos externos	Media	0x7	
Ficheros lógicos externos	Alta	1x10	
Total punto de función (PFNA)			73

Figura 3.6: Puntos de Función No Ajustados (PFNA) PARSE4U

Utilizando los factores de complejidad obtenidos en la tabla: 3.7, aplicando la ecuación 3.1, obtenemos:

$$FA = 0,65 + (0,1 * 39) = 4,55 \tag{3.2}$$

Ahora se calculan los puntos de función ajustados aplicando a los puntos de función no ajustados de la tabla: 3.6, el factor de ajuste obtenido en la ecuación: 3.2

$$PFA = 4,55 * 73 = 332,15 \tag{3.3}$$

Para calcular las líneas de código de la herramienta se hace una media de las equivalencias de los lenguajes de programación: HTML, J2EE, JavaScript y Java; debido a que se van a utilizar los tres para su desarrollo y no se sabe la proporción que se utilizará de cada uno de ellos.

Factor de complejidad	Complejidad / Influencia
Comunicación de datos	3
Funciones distribuidas	3
Rendimiento	2
Gran carga de trabajo	4
Frecuencia de transacciones	4
Entradas on-line de datos	5
Requisitos de manejo del usuario final	1
Actualizaciones online	2
Procesos complejos	2
Utilizacion de otros sistemas	2
Facilidad de mantenimiento	2
Facilidad de operación	3
Instalacion en multiples lugares	3
Facilidad de cambio	3

Figura 3.7: Factores de complejidad PARSE4U

La equivalencia resultante es 1PF = 45 líneas de código. Por lo tanto, utilizando el número de Puntos de Función Ajustados obtenidos en la ecuación: 3.3, tenemos que:

$$LDC = 45 * 332,15 = 14947 \tag{3.4}$$

Por lo que, usando una estimación del desarrollo del software utilizando el cálculo de Puntos de Función, nuestra estimación indica que el proyecto tendrá unas 15000 líneas de código aproximadamente.

3.2.2. Estimación de Costes Utilizando COCOMO

Se trata de una estimación para medir el esfuerzo y tiempo que supondrá el desarrollo del proyecto. Para ello, el método del COCOMO se basa en una estimación previa del tamaño del software en líneas de código (LDC). Esta estimación se puede ver en la ecuación: 3.4. AL igual que en la sección anterior, primero se detallará el proceso que se ha de seguir y posteriormente se realizará la estimación.

El modelo de COCOMO, incluye tres modelos, en función de las características del sistema a desarrollar y en función del nivel de detalle y aproximación que se necesite.

- **Modelo orgánico:** Un pequeño grupo de desarrolladores experimentados diseñan y desarrollan software en un entorno familiar. El tamaño del software varía desde unos pocos miles de líneas de código (software de tamaño pequeño) a unas decenas de miles (software de tamaño medio).
- **Modelo semilibre:** Corresponde a un esquema intermedio entre el orgánico y el rígido; el grupo de desarrollo puede incluir una mezcla de desarrolladores con mucha y poca experiencia.

- **Modelo rígido:** El proyecto tiene fuertes restricciones, que pueden estar relacionadas con la funcionalidad y/o pueden ser técnicas,. El problema a resolver es único y es difícil basarse en la experiencia, puesto que puede no haberla.

Proyecto software	a	b	c	d
Organico	2,4	1,05	2,5	0,38
Empotrado	3	1,12	2,5	0,35
Semi-libre	3,6	1,2	2,5	0,32

Figura 3.8: Constantes de los modelos COCOMO

En caso de necesitar ajustar los factores de coste, debemos utilizar los datos que se muestran en la Figura: 3.9

Atributos	Valor					
	Muy bajo	Bajo	Nominal	Alto	Muy alto	Extra alto
Atributos de software						
Fiabilidad	0,75	0,88	1,00	1,15	1,40	
Tamaño de Base de datos		0,94	1,00	1,08	1,16	
Complejidad	0,70	0,85	1,00	1,15	1,30	1,65
Atributos de hardware						
Restricciones de tiempo de ejecución			1,00	1,11	1,30	1,66
Restricciones de memoria virtual			1,00	1,06	1,21	1,56
Volatilidad de la máquina virtual		0,87	1,00	1,15	1,30	
Tiempo de respuesta		0,87	1,00	1,07	1,15	
Atributos de personal						
Capacidad de análisis	1,46	1,19	1,00	0,86	0,71	
Experiencia en la aplicación	1,29	1,13	1,00	0,91	0,82	
Calidad de los programadores	1,42	1,17	1,00	0,86	0,70	
Experiencia en la máquina virtual	1,21	1,10	1,00	0,90		
Experiencia en el lenguaje	1,14	1,07	1,00	0,95		
Atributos del proyecto						
Técnicas actualizadas de programación	1,24	1,10	1,00	0,91	0,82	
Utilización de herramientas de software	1,24	1,10	1,00	0,91	0,83	
Restricciones de tiempo de desarrollo	1,23	1,08	1,00	1,04	1,10	

Figura 3.9: Factores de coste de COCOMO Intermedio

Las ecuaciones que se utilizan en COCOMO son las siguientes:

- Esfuerzo nominal (MM) [personas x mes]:

$$MM = a * (Kl^b) \tag{3.5}$$

- Esfuerzo (E) [personas x mes]:

$$E = MM * \Pi m(x_i) \tag{3.6}$$

- Tiempo de desarrollo (TDEV) [meses]:

$$TDEV = c * (E^d) \tag{3.7}$$

- Nº medio de personas (CosteH) [personas]:

$$CosteH = E/TDEV \tag{3.8}$$

Por lo tanto, clasificamos nuestro sistema a desarrollar como un sistema semilibre, ya que se encuentra en una posición intermedia entre el modelo orgánico y el modelo rígido.

Para pasar al modelo COCOMO en su vertiente intermedia, hay que aplicar un factor para el esfuerzo. Este factor se obtiene evaluando los 15 atributos descritos en la tabla 3.9. de esta forma, los valores recogidos para este proyecto son:

Atributos	Valor					
	Muy bajo	Bajo	Normal	Alto	Muy alto	Extra alto
Atributos de software						
Fiabilidad requerida				**1,15**		
Tamaño de Base de datos					**1,16**	
Complejidad del software			**1**			
Atributos de hardware						
Restricciones de tiempo de ejecución			**1**			
Restricciones de memoria virtual				**1,06**		
Volatilidad de la máquina virtual		**0,87**				
Restricciones de tiempo de respuesta				**1,07**		
Atributos de personal						
Capacidad de análisis				**0,86**		
Experiencia con el tipo de aplicación				**0,91**		
Calidad de los programadores				**0,86**		
Experiencia en la máquina virtual			**1**			
Experiencia en el lenguaje de programacion				**0,95**		
Atributos del proyecto						
Técnicas modernas de programación				**0,91**		
Empleo de herramientas de software					**0,83**	
Restricciones de tiempo de desarrollo				**1,04**		

Figura 3.10: Factores de coste para COCOMO intermedio

Si aplicamos las formulas del COCOMO a los Factores obtenidos, tenemos:

$$m(x) = \Pi m(x_i) = 1,15{*}1,16{*}1{*}1{*}1,06{*}0,87{*}1,07{*}0,86{*}0,91{*}0,86{*}1{*}0,95{*}0,91{*}0,83{*}1,04 \simeq 0,66 \tag{3.9}$$

Teniendo calculado el meso del factor de coste podemos calcular el esfuerzo de la aplicación en personas/mes

$$MM = a * (Kl^b) = 3 * (14,94^1, 12) = 76,97 \tag{3.10}$$

$$E = MM * \Pi m(x_i) = 76,97 * 0,66 = 50\ personas \tag{3.11}$$

$$TDEV = c * (E^d) = 2,5 * (50^{0,35}) = 9,8\ Meses \tag{3.12}$$

Aproximadamente, se necesitarían 296 días para terminar el proyecto.

$$CosteH = E/TDEV = 50/9,8 = 5,1\ Personas \tag{3.13}$$

3.2.3. Presupuesto

Para desarrollar el proyecto se necesitarán equipos hardware y herramientas software cuyo coste habrá que introducir en el presupuesto. También habrá que incluir el coste de los recursos humanos utilizados.

- **Presupuesto del Hardware:** Están incluidos: ordenador para el desarrollo, implantación y pruebas del proyecto tanto en local como en un servidor remoto; conexión a Internet para la obtención de información y conexión con Twitter, impresora para imprimir la documentación que sea necesaria entregar. Suponiendo que se trabajan 40 horas semanales, y cada año aproximadamente, cuenta con 49 semanas laborales; en un año se trabajan 1960 horas. Por lo que el hardware destinado al proyecto se utilizara 1960 horas al año.

Hardware	Uso (horas)	Coste del hardware (€)	Coste total (€)
Ordenador personal (5 años)	2000 h.	1400 €	285 €
Servidor (7años)	6000 h.	3000 €	410 €
Impresora (3 años)	70 h.	120 €	0.85 €
Conexión a internet (4 años)	6000 h.	540 €	92 €

Tabla 3.2: Presupuesto Hardware

TOTAL: 787,85 €

- **Presupuesto Software:** Se utilizarán las siguientes herramientas con sus costes asociados. Suponiendo que se trabajan 40 horas semanales de igual manera que para el cálculo del uso del Hardware.

Software	Uso (horas)	Coste del hardware (€)	Coste total (€)
Windows 10 (2 años)	2000 h.	220 €	35 €
GNU/Linux Ubuntu 16.04 (1años)	6000 h.	0 €	0 €
Netbeans IDE (2 años)	70 h.	0 €	0 €
GitHub (1 años)	70 h.	120 €	0 €
Glassfish Server (1 años)	70 h.	120 €	0 €
TexStudio (4 años)	6000 h.	540 €	0 €

Tabla 3.3: Presupuesto Software

TOTAL: 35 €

- **Presupuesto del desarrollo del proyecto:** En la siguiente tabla, se encuentra un desglose de las tareas necesarias para llevar a cabo el proyecto y la duración estimada de las mismas en horas de trabajo. Se ha estimado que los trabajos durarán meses, y que cada día se pretende trabajar 8 horas una media de 25 días por mes.

Tarea	Duración en horas
Estudio de la técnica	280
Requisitos del sistema	220
Análisis de componentes	170
Diseño de componentes	105
Implementación de componentes	135
Pruebas de componentes	60
Pruebas de aplicación	50
Documentación	290
TOTAL	1310

Tabla 3.4: Distribución de horas

- **Presupuesto de mano de obra:** A partir del esfuerzo obtenido en el modelo COCOMO en su variante intermedia, podemos realizar una estimación para hallar el presupuesto en recursos humanos. Suponiendo que el sueldo de cada uno de los programadores es de 1600€Brutos. Según la estimación de COCOMO, el coste sería 1600*9=14400 €

Si tenemos en cuenta las estimaciones realizadas tanto del Hardware, Software y la mano de obra tenemos que:

Presupuesto Total Estimado = Total Estimado Hardware + Total Estimado Software + Presupuesto de Mano de Obra

Presupuesto Estimado = 787.85 + 35 + 14400 = **15222.85 €**

3.3. Costes Económicos y Temporales Reales

Los resultados obtenidos con los métodos anteriores son estimaciones y nunca se debe considerar definitivos para calcular el coste económico y temporal definitivos, ya que no da un resultado exacto de los que va a costar desarrollar el sistema, sino que solamente deben usarse para dar una "idea" de cuánto va a costar y cuánto se va a tardar en el proceso de diseño y desarrollo del proyecto. Por eso, en esta sección se incluye un presupuesto final con los costes reales una vez terminado el trabajo de diseño y desarrollo del proyecto PARSE4U Tracking.

- **Presupuesto Hardware:** Ordenador personal para el desarrollo, implantación y pruebas del proyecto. Un servidor donde poder alojar la aplicación, una conexión a internet y una impresora para imprimir la documentación e información necesaria. Suponiendo que se trabajan 35 horas semanales, y cada año se aproxima que posee 49 semanas laborales. Entonces, al año, se trabajan 1715 horas, o lo que es lo mismo, se utilizaría cada componente Hardware 1715 horas al año.

Hardware	Uso (horas)	Coste del hardware (€)	Coste total (€)
Ordenador personal (5 años)	500 h.	1400 €	408 €
Servidor (7años)	2000 h.	3000 €	430 €
Impresora (3 años)	70 h.	120 €	30 €
Conexión a internet (4 años)	2000 h.	540 €	380 €

Tabla 3.5: Presupuesto Hardware Real

TOTAL: 1248€

- **Presupuesto Software:** Se utilizarán las siguientes herramientas con sus costes asociados. Suponiendo que se trabajan 40 horas semanales de igual manera que para el cálculo del uso del Hardware.

Software	Uso (horas)	Coste del hardware (€)	Coste total (€)
Windows 10 (2 años)	2000 h.	220 €	35 €
GNU/Linux Ubuntu 16.04 (1años)	6000 h.	0 €	0 €
Netbeans IDE (2 años)	70 h.	0 €	0 €
GitHub (1 años)	70 h.	120 €	0 €
Glassfish Server (1 años)	70 h.	120 €	0 €
TexStudio (4 años)	6000 h.	540 €	0 €

Tabla 3.6: Presupuesto Software Real

TOTAL: 35 €

- **Presupuesto del desarrollo del proyecto:** En la siguiente tabla, se encuentra un desglose de las tareas que se han llevado a cabo para el desarrollo del proyecto y la duración de las mismas, en horas de trabajo. Se ha considerado que se trabaja 8 horas al día una media de 25 días por mes.

Tarea	Duración en horas
Estudio de la técnica	250
Requisitos del sistema	180
Análisis de componentes	140
Diseño de componentes	95
Implementación de componentes	175
Pruebas de componentes	70
Pruebas de aplicación	55
Documentación	320
TOTAL	1285

Tabla 3.7: Distribución de Horas Real

- **Presupuesto de mano de obra:** A partir del esfuerzo obtenido en el modelo COCOMO en su variante intermedia, podemos realizar una estimación para hallar el presupuesto en recursos humanos. Suponiendo que el sueldo de cada uno de los programadores es de 1800€Brutos. Según la estimación de COCOMO, el coste sería 1800*9=16200 €

Si tenemos en cuenta los cálculos realizados tanto del Hardware, Software y la mano de obra tenemos que:

Presupuesto Total = Total Hardware + Total Software +
Presupuesto de Mano de Obra

Presupuesto Definitivo = 1248 + 35 + 16200 = **17483 €**

3.3.1. Coste Temporal

Durante el desarrollo del proyecto se ha seguido un modelo de trabajo incremental, dividiendo el tiempo total en 8 incrementos. Cada incremento esta dividido en seis etapas: análisis, diseño, implementación, prueba, despliegue y documentación.

Nombre de tarea	Duración	Comienzo	Fin	Nombre de tarea2	Duración2	Comienzo2	Fin2
Iteración 1	**30 días**	**mar 01/12/15**	**mié 30/12/15**	**Iteración 5**	**30 días**	**vie 01/04/16**	**sáb 30/04/16**
Análisis	16 días	mar 01/12/15	mié 16/12/15	Análisis	4 días	mié 30/03/16	sáb 02/04/16
Diseño	6 días	jue 17/12/15	mar 22/12/15	Diseño	3 días	dom 03/04/16	mar 05/04/16
Implementación	4 días	mié 23/12/15	sáb 26/12/15	Implementación	16 días	mié 06/04/16	jue 21/04/16
Prueba	2 días	dom 27/12/15	lun 28/12/15	Prueba	2 días	vie 22/04/16	sáb 23/04/16
Despliegue	1 día	mar 29/12/15	mar 29/12/15	Despliegue	2 días	dom 24/04/16	lun 25/04/16
Documentación	1 día	mié 30/12/15	mié 30/12/15	Documentación	3 días	mar 26/04/16	jue 28/04/16
Iteración 2	**31 días**	**vie 01/01/16**	**dom 31/01/16**	**Iteración 6**	**31 días**	**dom 01/05/16**	**mar 31/05/16**
Análisis	12 días	jue 31/12/15	lun 11/01/16	Análisis	3 días	vie 29/04/16	dom 01/05/16
Diseño	8 días	mar 12/01/16	mar 19/01/16	Diseño	3 días	lun 02/05/16	mié 04/05/16
Implementación	6 días	mié 20/01/16	lun 25/01/16	Implementación	15 días	jue 05/05/16	jue 19/05/16
Prueba	2 días	mar 26/01/16	mié 27/01/16	Prueba	4 días	vie 20/05/16	lun 23/05/16
Despliegue	1 día	jue 28/01/16	jue 28/01/16	Despliegue	1 día	mar 24/05/16	mar 24/05/16
Documentación	1 día	vie 29/01/16	vie 29/01/16	Documentación	4 días	mié 25/05/16	sáb 28/05/16
Iteración 3	**29 días**	**lun 01/02/16**	**lun 29/02/16**	**Iteración 7**	**30 días**	**mié 01/06/16**	**jue 30/06/16**
Análisis	8 días	sáb 30/01/16	sáb 06/02/16	Análisis	1 día	dom 29/05/16	dom 29/05/16
Diseño	6 días	dom 07/02/16	vie 12/02/16	Diseño	2 días	lun 30/05/16	mar 31/05/16
Implementación	11 días	sáb 13/02/16	mar 23/02/16	Implementación	6 días	mié 01/06/16	lun 06/06/16
Prueba	2 días	mié 24/02/16	jue 25/02/16	Prueba	5 días	mar 07/06/16	sáb 11/06/16
Despliegue	1 día	vie 26/02/16	vie 26/02/16	Despliegue	2 días	dom 12/06/16	lun 13/06/16
Documentación	2 días	sáb 27/02/16	dom 28/02/16	Documentación	13 días	mar 14/06/16	dom 26/06/16
Iteración 4	**31 días**	**mar 01/03/16**	**jue 31/03/16**	**Iteracion 8**	**20 días**	**mar 14/06/16**	**dom 03/07/16**
Análisis	6 días	lun 29/02/16	sáb 05/03/16	Análisis	2 días	mar 14/06/16	mié 15/06/16
Diseño	4 días	dom 06/03/16	mié 09/03/16	Diseño	2 días	lun 27/06/16	mar 28/06/16
Implementación	14 días	jue 10/03/16	mié 23/03/16	Implementación	3 días	lun 04/07/16	mié 06/07/16
Prueba	2 días	jue 24/03/16	vie 25/03/16	Prueba	4 días	jue 07/07/16	dom 10/07/16
Despliegue	1 día	sáb 26/03/16	sáb 26/03/16	Despliegue	1 día	lun 11/07/16	lun 11/07/16
Documentación	3 días	dom 27/03/16	mar 29/03/16	Documentación	8 días	mar 12/07/16	mar 19/07/16

Figura 3.11: Planificación Temporal Real

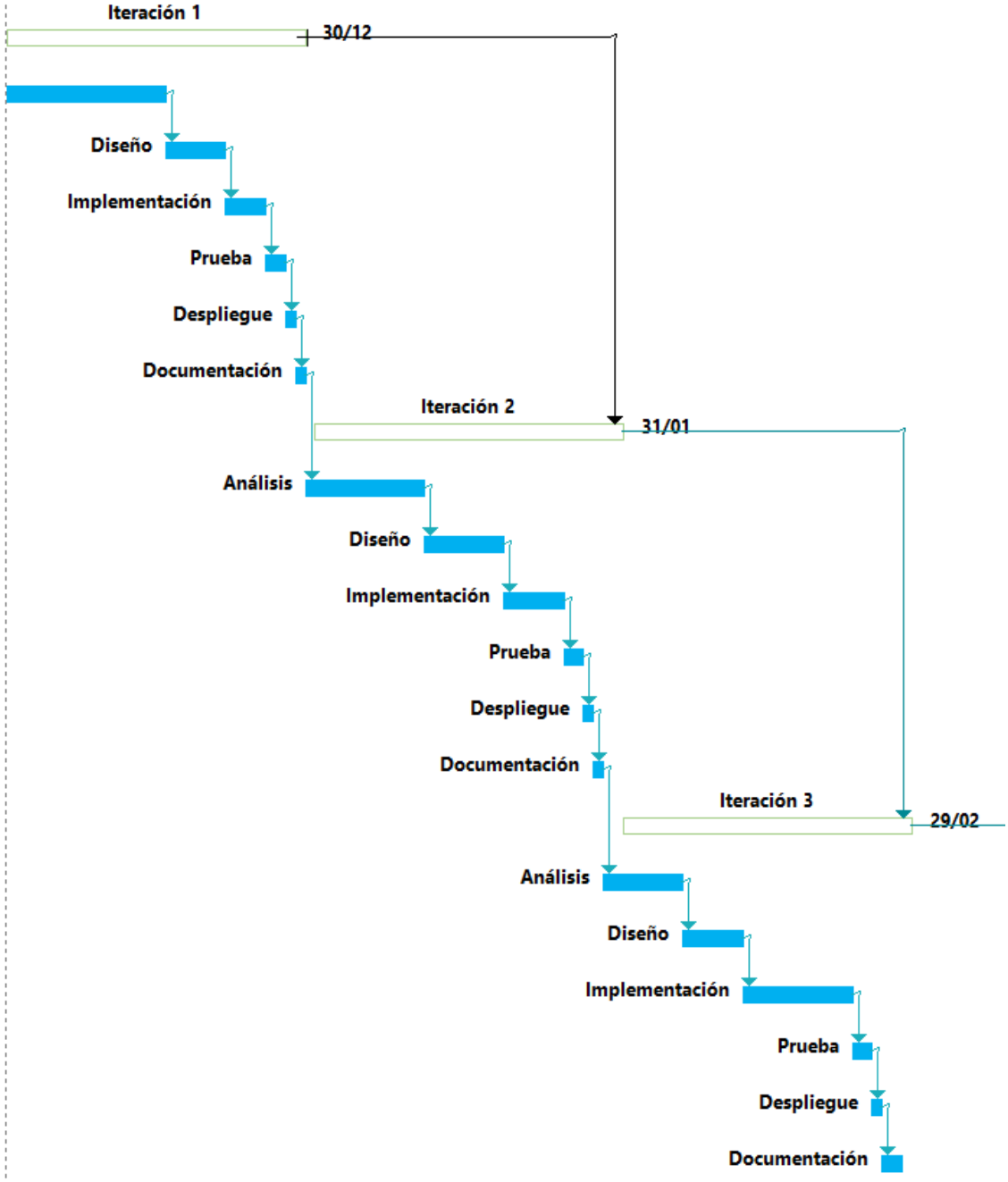


Figura 3.12: Parte 1: Coste Temporal del Proyecto Representado en un Diagrama de Gantt

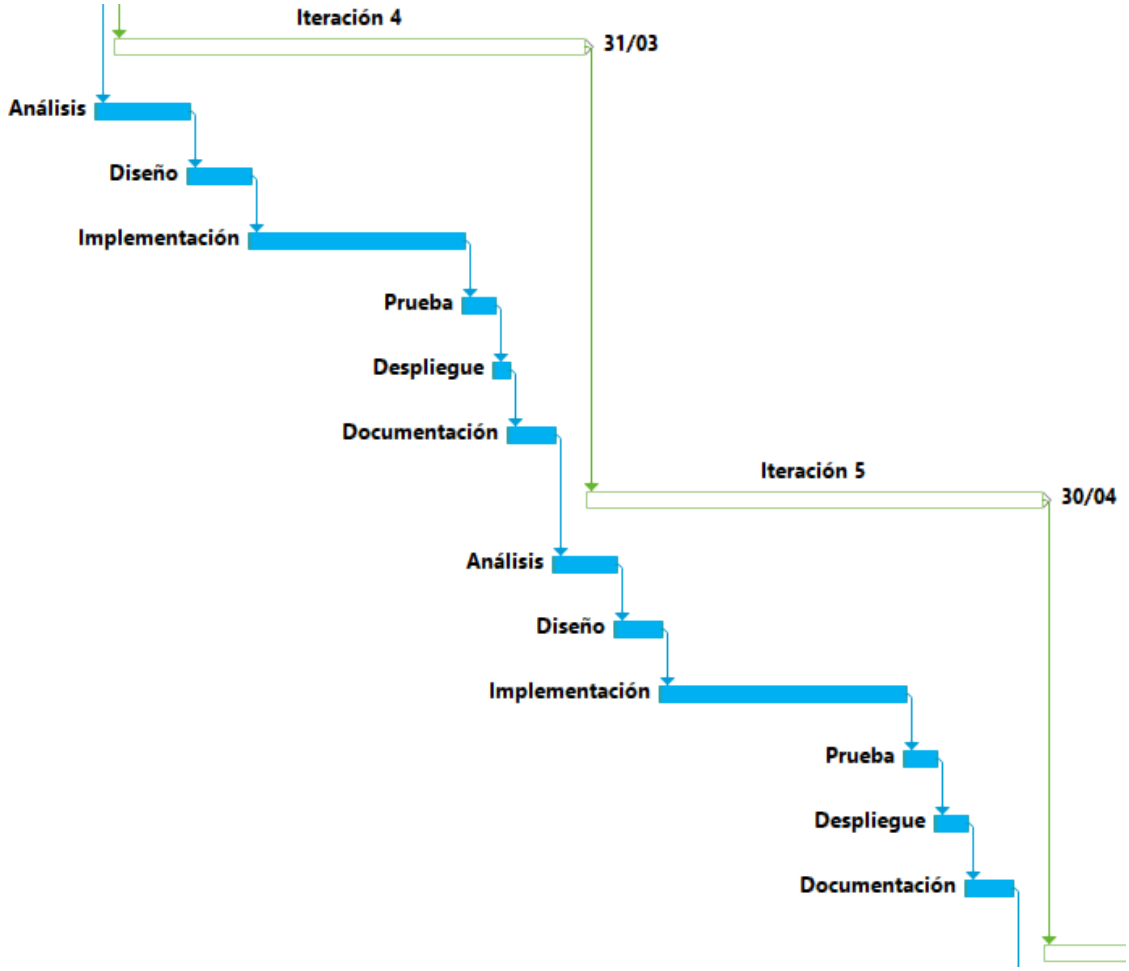


Figura 3.13: Parte 2: Coste Temporal del Proyecto Representado en un Diagrama de Gantt

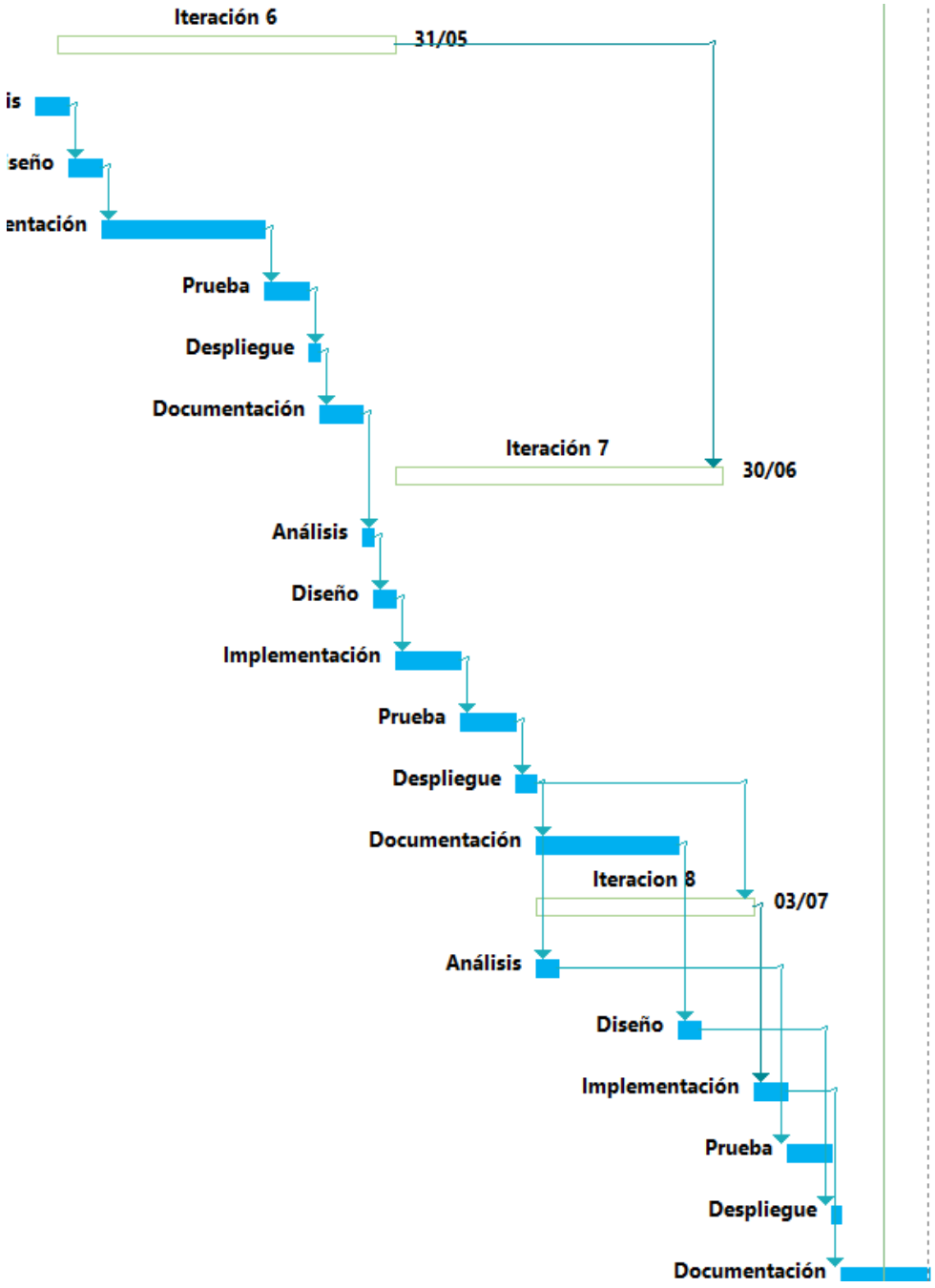


Figura 3.14: Parte 3: Coste Temporal del Proyecto Representado en un Diagrama de Gantt

Capítulo 4

Análisis del Sistema

En este capítulo del documento, vamos a especificar los requisitos que debe cumplir el proyecto y el diagrama de casos de uso de la aplicación. También se incluye la especificación de cada caso de uso y su relación con los requisitos. De esta manera los casos de uso describen todas las interacciones que tendrán los usuarios con el software; de igual manera, un requisito software es una necesidad documentada, sobre el contenido, forma o funcionalidad del software a desarrollar.

Uno de los principales problemas que tienen las herramientas existentes es que no ofrecen la posibilidad de realizar búsquedas en el pasado salvo unos pocos días debido a las limitaciones de la API de Twitter para búsquedas; en cambio, nuestra herramienta al utilizar la API de Streaming, todos los "Tweets" capturados son almacenados en nuestro sistema por lo que una vez se han capturado se pueden consultar en un futuro sin ninguna restricción.

Otra de las características que diferencian a nuestra herramienta de otras existentes, es la posibilidad de acceder a los "Tweets" almacenados utilizando una API rest propia, permitiendo que herramientas externas realicen sus propios análisis.

A continuación se van a resumir las diferentes características (tanto funcionales como no funcionales) del sistema, antes de entrar en mayor detalle a lo largo del capítulo.

- Tener una interfaz de usuario sencilla, donde para realizar una acción, el usuario tenga que realizar el menor número posible de pasos.
- La interfaz debe ser similar a la utilizada en herramientas similares o aplicaciones que los usuarios conozcan.
- La herramienta deberá ser escalable, permitiendo que aumenten en número tanto de usuarios como de datos sin que afecte al rendimiento de la aplicación.
- El programa deberá ser capaz tanto de autenticar usuarios existentes como de crear nuevos usuarios.
- El programa deberá permitir a los usuarios añadir sus propias cuentas de Twitter para la captura de datos.
- El programa deberá permitir que los usuarios configuren e inicien las capturas de los datos.
- El programa deberá permitir el acceso a los datos capturados a través de servicios web utilizando diferentes parámetros.

- El programa deberá bloquear al acceso a los datos capturados de una forma privada a usuarios no autorizados.

4.1. Actores del Sistema

Se llaman actores del sistema a todas las entidades externas al sistema que guarda una relación con este y que le demanda una cierta funcionalidad. Esto incluye a personas humanas que interaccionan con el software, pero también incluye a todos los sistemas externos que estén relacionados.

Act-01	**Usuario no registrado**
Versión	1.0 (10/04/2016)
Descripción	Tipo específico de Usuario cuando aun no se ha autenticado en el sistema.
Comentarios	Ninguno

Tabla 4.1: Act-01: Usuario registrado

Act-02	**Usuario**
Versión	1.0 (10/04/2016)
Descripción	Tipo específico de Usuario después de autenticarse en el sistema con sus credenciales.
Comentarios	Ninguno

Tabla 4.2: Act-02: usuario

Act-03	**Usuario administrador**
Versión	1.0 (10/04/2016)
Descripción	Tipo específico de Usuario con permisos de administración del sistema.
Comentarios	Ninguno

Tabla 4.3: Act-03: Usuario administrador

Act-04	**Twitter API**
Versión	1.0 (10/04/2016)
Descripción	Actor secundario que representa a la API publica de Twitter.
Comentarios	Ninguno

Tabla 4.4: Act-04: Twitter API

4.2. Requisitos de usuario

En esta parte del documento, se procederá a especificar los requisitos que debe cumplir el proyecto, así como el diagrama de casos de uso de la herramienta. Además, se incluye la especificación de cada uno de los requisitos de usuario.

Los diagramas de casos de uso representan el comportamiento que el sistema debe presentar evitando usar un lenguaje técnico, prefiriendo la lengua del usuario final o del experto del campo del saber al que se va a aplicar. Los casos de uso son a menudo, elaborados por el analista de

requisitos y los clientes. cada caso de uso se centra en describir como alcanzar una meta o tarea. A raíz de esto, un caso de uso, se puede entender como una interacción entre el actor y el sistema en respuesta a un evento. El diagrama de casos de uso de la herramienta se puede ver en la Figura:4.2

Los casos de uso se relacionan entre sí a través de varios tipos de relaciones:

- **Asociación.** Relación entre un actor y un caso de uso que sirve para describir la participación del actor en dicha función. Se suele representar como una línea continua.
- **Usa («include»).** Relación de dependencia entre dos casos de uso que denota la inclusión de un comportamiento en el escenario del otro. Se representa con una línea discontinua y una etiqueta de «include».
- **Extiende («extends»).** Relación de dependencia entre dos casos de uso que denota que un caso de uso es una especialización de otro. Se representa con una línea discontinua y con la etiqueta «extends»

Además, un actor puede heredar los casos de uso con los que se relaciona otro actor mediante el uso de la notación de generalización, que se representa con una flecha continua que sube desde el actor "hijo" al actor "padre". para representar los límites del sistema con el exterior, se suele encuadrar todos los casos de uso dentro de una caja o recuadro.

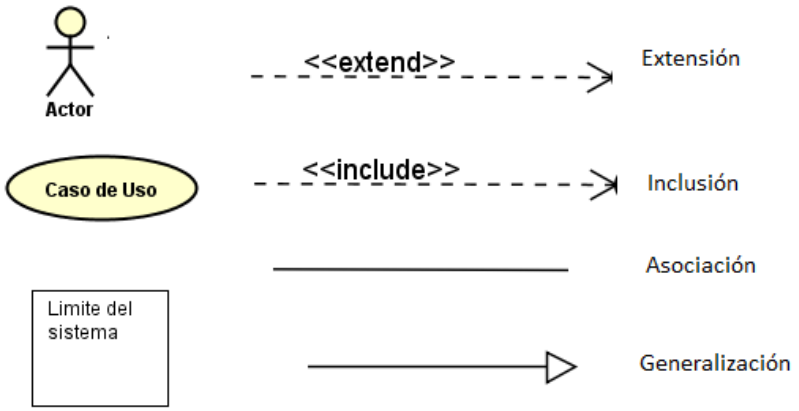


Figura 4.1: Notación de casos de uso

4.2.1. Modelo de Casos de Uso

En la Figura 4.2 se puede ver el Modelo de Casos de Uso del sistema. Como se puede apreciar, existe una relación de generalización entre los usuarios del sistema; donde el usuario "administrador", además de poseer sus propios casos de uso, hereda los casos de uso del usuario "registrado". Así, mismo el usuario registrado hereda los casos de uso del usuario "no registrado". Por lo tanto, cualquier usuario antes de realizar una acción debe previamente autenticarse en el sistema utilizando sus credenciales.

También, se puede apreciar, que existe un usuario llamado Cliente de "Twitter" este actor no es una persona física, sino que es una parte de nuestro sistema que interactúa con la API de Twitter para capturar datos y almacenarlos en la base de datos de nuestro sistema.

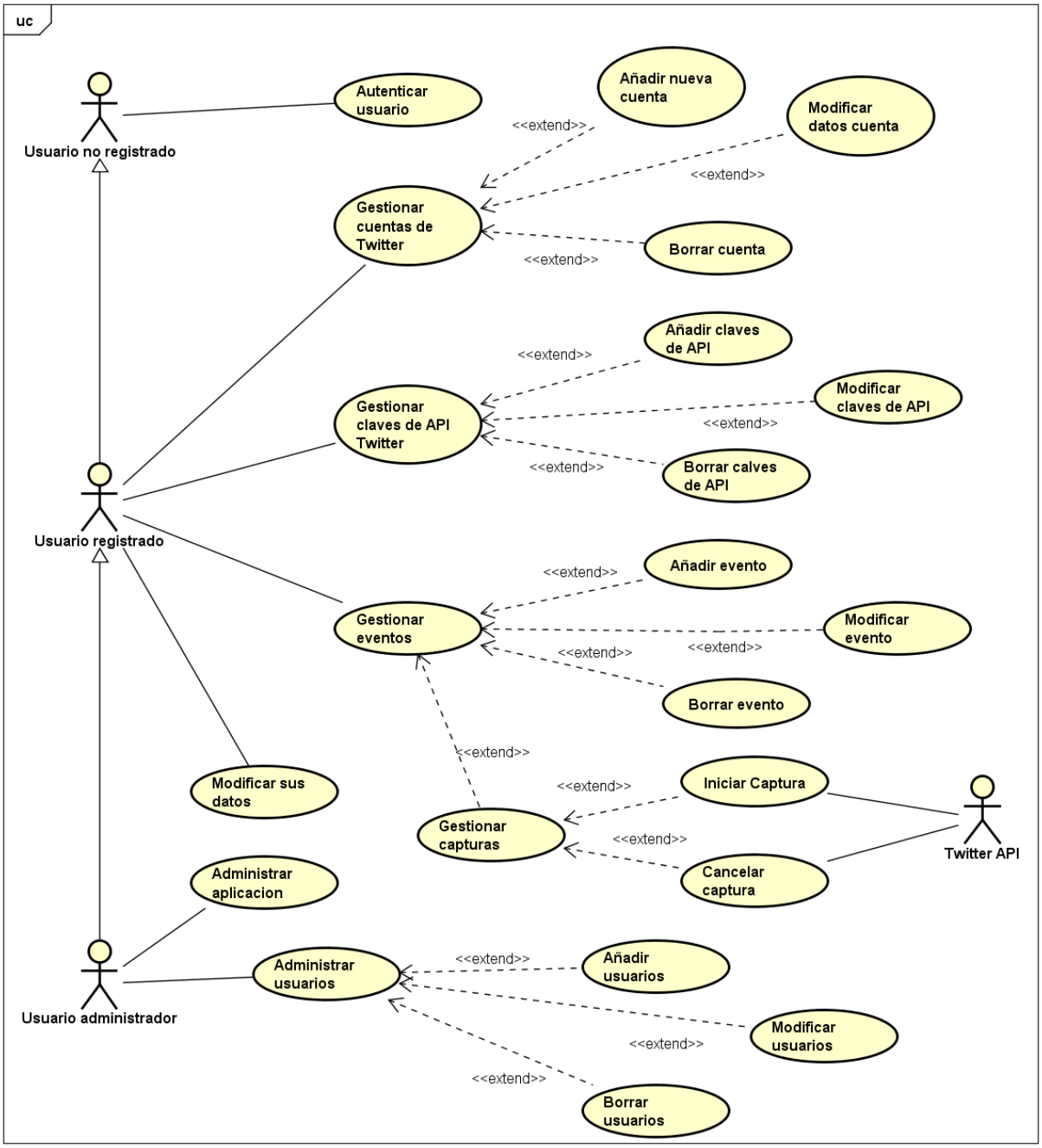


Figura 4.2: Modelo de Casos de Uso

4.2.2. Especificación

CDU-01	Autenticar usuario	**CDU-12**	Modificar Evento
CDU-02	Gestionar cuenta de Twitter	**CDU-13**	Borrar Evento
CDU-03	Añadir nueva cuenta	**CDU-14**	Gestionar capturas
CDU-04	Modificar datos cuenta	**CDU-15**	Iniciar Captura
CDU-05	Borrar cuenta	**CDU-16**	Cancelar Captura
CDU-06	Gestionar claves de API	**CDU-17**	Modificar sus datos
CDU-07	Añadir claves API	**CDU-18**	Administrar usuarios
CDU-08	Modificar claves API	**CDU-19**	Añadir usuarios
CDU-09	Borrar claves API	**CDU-20**	Modificar usuarios
CDU-10	Gestionar Eventos	**CDU-21**	Borrar usuarios
CDU-11	Añadir Evento	**CDU-22**	Administrar aplicación

Tabla 4.5: Lista Casos de Uso

En las siguientes figuras se pueden encontrar la especificación de los Casos de Uso más relevantes del sistema. El resto de Casos de Uso existentes en el cuadro:4.5, pueden encontrarse en el AnexoA

CDU-01	**Autenticar Usuario**
Versión	2.0 (26/06/2016)
Dependencias	Ninguna
Precondición	El usuario deberá estar previamente registrado en el sistema.
Descripción	El sistema deberá permitir a los usuarios autenticarse utilizando los credenciales almacenados en la base de datos
Secuencia normal	1. El usuario introduce sus credenciales. 2. El sistema comprueba sus credenciales.
Postcondición	Si son correctos el sistema crea una nueva sesión de usuario.
Excepciones	1. Si los credenciales introducidos no son correctos, el sistema mostrará un mensaje de error al usuario.
Frecuencia	10 veces/día.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno.

Figura 4.3: Especificación CDU-01 Autenticar Usuario

CDU-02	Gestionar Cuenta de Twitter
Versión	2.0 (26/06/2016)
Dependencias	CDU-01
Precondición	El usuario deberá estar previamente autenticado en el sistema.
Descripción	El sistema deberá permitir a los usuarios realizar acciones sobre las cuentas de Twitter que posea el usuario
Secuencia normal	1. El usuario solicita gestionar las cuentas de Twitter 2. El sistema muestra un listado con las acciones que el usuario puede realizar sobre sus cuentas de Twitter 3. El usuario elige una de las opciones: a. Si el usuario solicita "Añadir una nueva cuenta", se realiza el caso de uso "CDU-03: Añadir nueva cuenta" b. SI el usuario solicita "Modificar una cuenta existente", se realiza el caso de uso "CDU-04: Modificar una cuenta existente" c. SI el usuario solicita "Borrar una cuenta", se realiza el "CDU-05 Borrar cuenta"
Postcondición	Ninguna.
Excepciones	1. Si el usuario no ha añadido ninguna cuenta de Twitter, algunas acciones no estarán disponibles.
Frecuencia	5 veces/día.
Importancia	Alta
Prioridad	Media
Comentarios	Ninguno.

Figura 4.4: Especificación CDU-02 Gestionar cuenta de Twitter

CDU-06	Gestionar Claves de API
Versión	2.0 (26/06/2016)
Dependencias	CDU-01
Precondición	El usuario deberá estar previamente autenticado en el sistema.
Descripción	El sistema deberá permitir a los usuarios realizar acciones sobre las claves de API de Twitter.
Secuencia normal	1. El usuario solicita gestionar las Claves de API. 2. El sistema muestra las acciones disponibles sobre las Claves de API del usuario. 3. El usuario selecciona una de las opciones: a. SI el usuario solicita "Añadir una clave API", se realiza el caso de uso "CDU-07: Añadir Claves de API" b. Si el usuario solicita "Modificar una Clave de API existente", se realiza el "CDU-08: Modificar Claves de API" c. Si el usuario solicita "Eliminar una Clave de API", se realiza el "CDU-09: Eliminar Clave API"
Postcondición	Ninguna.
Excepciones	1. Si el usuario no ha añadido ninguna clave de API en el sistema, algunas acciones no estarán disponibles
Frecuencia	5 veces/día.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno.

Figura 4.5: Especificación CDU-06 Gestionar claves de API

CDU-10	Gestionar Eventos
Versión	2.0 (26/06/2016)
Dependencias	CDU-01
Precondición	El usuario deberá estar previamente autenticado en el sistema.
Descripción	El sistema deberá permitir a los usuarios realizar acciones sobre los eventos.
Secuencia normal	1. El usuario solicita gestionar los eventos. 2. El sistema muestra las opciones disponibles. 3. El usuario selecciona una de las opciones: a. Si el usuario solicita "Añadir un Nuevo Evento", se realiza el "CDU-11: Añadir Nuevo Evento" b. Si el usuario solicita "Modificar un Evento Existente", se realiza el "CDU-12: Modificar un Evento" c. Si el usuario solicita "Borrar un Evento", se realiza el "CDU-13: Borrar un Evento"
Postcondición	Ninguna.
Excepciones	1. Si el usuario no ha añadido previamente ningún evento, algunas opciones no estarán disponibles.
Frecuencia	10 veces/día.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno.

Figura 4.6: Especificación CDU-10 Gestionar eventos

CDU-14	Gestionar Capturas
Versión	2.0 (26/06/2016)
Dependencias	CDU-01
Precondición	El usuario deberá estar previamente autenticado en el sistema.
Descripción	El sistema deberá permitir a los usuarios realizar acciones sobre las capturas de datos de sus eventos
Secuencia normal	1. El usuario accede al menú de "Ver eventos" y pulsa la opción de "Capturar" sobre el evento. 2. El sistema muestra las opciones disponibles. 3. El usuario elige una de las opciones: a. Si el usuario selecciona la opción "Iniciar Captura", se realiza el "CDU:15 Iniciar Captura" b. Si el usuario selecciona la opción "Cancelar Captura", se realiza el "CDU-16: Cancelar Captura"
Postcondición	Ninguna.
Excepciones	Ninguna.
Frecuencia	10 veces/día.
Importancia	Alta
Prioridad	Media
Comentarios	Ninguno.

Figura 4.7: Especificación CDU-14 Gestionar capturas

CDU-17	Modificar sus datos
Versión	2.0 (26/06/2016)
Dependencias	CDU-01
Precondición	El usuario deberá estar previamente autenticado en el sistema.
Descripción	El sistema deberá permitir a los usuarios modificar sus datos de usuario.
Secuencia normal	1. El usuario selecciona el menú de "Mis datos" 2. El sistema mostrará los datos actuales del usuario. 3. El usuario modificará los datos deseados y pulsa en "Guardar". 4. El sistema actualiza los datos del usuario.
Postcondición	Ninguna.
Excepciones	1. Si los datos introducidos no son correctos, el sistema mostrará un mensaje de error al usuario.
Frecuencia	2 veces/día.
Importancia	Baja
Prioridad	Baja
Comentarios	Ninguno.

Figura 4.8: Especificación CDU-17 Modificar sus datos

CDU-18	Administrar usuarios
Versión	2.0 (26/06/2016)
Dependencias	CDU-01
Precondición	El usuario deberá estar previamente autenticado en el sistema como un usuario administrador
Descripción	El sistema deberá permitir a los usuarios administradores realizar gestiones sobre los usuarios de la aplicaciones
Secuencia normal	1. El usuario selecciona el menú de "Usuarios" 2. El sistema mostrará los datos actuales de la configuración del sistema.
Postcondición	Ninguna.
Excepciones	Ninguna.
Frecuencia	2 veces/día.
Importancia	Media
Prioridad	Baja
Comentarios	Ninguno.

Figura 4.9: Especificación CDU-18 Administrar usuarios

CDU-22	Administrar aplicación
Versión	2.0 (26/06/2016)
Dependencias	CDU-01
Precondición	El usuario deberá estar previamente autenticado en el sistema como un usuario administrador
Descripción	El sistema deberá permitir a los usuarios administradores modificar los nombres de las bases de datos y colecciones que se usan por la aplicación
Secuencia normal	1. El usuario selecciona el menú de "Administrar APP" 2. El sistema mostrará los datos actuales del sistema. 3. El usuario modifica los datos deseados. 4. El sistema se modifica los datos en el fichero de configuración.
Postcondición	Ninguna.
Excepciones	Ninguna.
Frecuencia	1 vez / mes
Importancia	Baja
Prioridad	Baja
Comentarios	Ninguno.

Figura 4.10: Especificación CDU-22 Administrar aplicación

4.3. Requisitos Funcionales

Un requisito funcional es aquel que describe una funcionalidad del software. PARSE4U va a implementar los siguientes requisitos funcionales:

- **RF-01 Autenticar Usuarios:** El sistema deberá comprobar si las credenciales introducidas por el usuario corresponden con las credenciales almacenadas en la base de datos.
- **RF-02 Error en la Autenticación:** El sistema deberá mostrar un mensaje de error en caso de que las credenciales introducidas no sean correctas.
- **RF-03 Crear una sesión de usuario:** El sistema deberá iniciar una sesión cuando la autenticación del usuario sea correcta.
- **RF-04 Comprobar el Grupo de un Usuario:** El sistema deberá comprobar si el usuario que accede al sistema pertenece al grupo de administradores de la aplicación o no.
- **RF-05 Listar Cuentas de Twitter:** El sistema deberá mostrar un listado de las cuentas de Twitter del usuario autenticado.
- **RF-06 Añadir Nuevas Cuentas de Twitter:** El sistema deberá añadir a la base de datos nuevas cuentas de Twitter, almacenando que usuario de la aplicación es el propietario de la cuenta.
- **RF-07 Validar Datos Cuenta Twitter:** El sistema deberá validar los datos de una cuenta de Twitter antes de añadirla a base de datos.
- **RF-08 Error de Validación en Cuenta de Twitter:** El sistema deberá mostrar un mensaje de error al usuario en caso de que los datos introducidos al añadir una cuenta de Twitter no sean correcto.

- **RF-09 Comprobar si Existe una Cuenta de Twitter:** El sistema deberá comprobar si la cuenta de Twitter que se quiere añadir ya existe en el sistema.
- **RF-10 Error Cuenta de Twitter Existente:** El sistema deberá mostrar un mensaje de error al usuario en caso de que la cuenta de Twitter que intenta añadir ya exista en el sistema.
- **RF-11 Modificar Datos de una Cuenta de Twitter:** El sistema deberá permitir modificar los datos almacenados de una cuenta de Twitter.
- **RF-12 Validar la Modificación de una Cuenta de Twitter:** El sistema deberá validar los datos modificados de una cuenta de Twitter antes de que se modifiquen en la base de datos.
- **RF-13 Error en la Validación de las Modificaciones de una Cuenta de Twitter:** El sistema deberá mostrar un mensaje de error si los datos modificados de una cuenta de Twitter no son correctos.
- **RF-14 Borrar una Cuenta de Twitter:** El sistema deberá permitir al usuario propietario de una cuenta de Twitter eliminarla del sistema.
- **RF-15 Listar Claves de API:** El sistema deberá mostrar un listado de las claves de API de Twitter que el usuario autenticado ha añadido al sistema.
- **RF-16 Añadir Nuevas Claves de API:** El sistema deberá permitir añadir nuevas claves de API de Twitter en el sistema.
- **RF-17 Validar Datos de Clave de API:** El sistema deberá comprobar que los datos de una clave de API que se desea añadir sean correctos.
- **RF-18 Error de Validación en los Datos de una Clave de API:** El sistema deberá mostrar un error al usuario si los datos de una calve de api de Twitter no son correctas.
- **RF-19 Comprobar si existe una Clave de API:** El sistema deberá comprobar si las claves de API de Twitter que se desean añadir ya existen en el sistema
- **RF-20 Claves de API Existentes:** El sistema deberá mostrar al usuario un mensaje de error en caso que ya existan una claves de API con el mismo nombre.
- **RF-21 Modificar Claves de API:** El sistema deberá permitir al usuario autenticado modificar sus claves de API.
- **RF-22 Validar Modificaciones en Claves de API:** El sistema deberá validar las modificaciones en los datos de las Claves de API.
- **RF-23 Error en Validación de Modificación Claves de API:** El sistema deberá mostrar un error al usuario si los datos modificados en una calve de API no son correctos.
- **RF-24 Borrar una Clave de API:** El sistema deberá permitir al usuario eliminar una clave de API de Twitter que previamente haya añadido al sistema.
- **RF-25 Listar Eventos:** El sistema deberá mostrar un listado de los eventos que el usuario ha añadido al sistema.
- **RF-26 Añadir Nuevo Evento:** El sistema deberá añadir a la base de datos un nuevo evento.

- **RF-27 Validar los Datos de un Evento Nuevo:** El sistema deberá comprobar si los datos de un evento son correctos antes de añadirlo en la base de datos.
- **RF-28 Mensaje de Error en Validación en Creación de Evento:** El sistema deberá mostrar un mensaje de error al usuario si los datos introducidos al crear un evento nuevo no son correctos.
- **RF-29 Comprobar si un Evento Existe:** El sistema deberá comprobar si existe un evento con el mismo nombre que el evento que se quiere añadir.
- **RF-30 Mensaje Evento Existente:** El sistema deberá mostrar un mensaje de error al usuario en caso de que el nombre del evento ya este en uso.
- **RF-31 Modificar Datos de un Evento:** El sistema deberá permitir modificar los datos existentes de un evento que el usuario haya creado anteriormente.
- **RF-32 Validar Datos Modificados de un Evento:** El sistema deberá validar si los datos modificados de un evento son correctos antes de modificar la información en la base de datos.
- **RF-33 Mensaje de Error en Validación al Modificar un Evento:** El sistema deberá mostrar un mensaje de error al usuario en caso de que los datos modificados no sean correctos.
- **RF-34 Comprobar si un Evento esta Capturando Datos:** El sistema deberá comprobar si un evento esta capturando datos de Twitter.
- **RF-35 Mensaje de Error si el Evento está Capturando:** El sistema deberá permitir modificar los datos de un evento si este está capturando datos de Twitter.
- **RF-36 Borrar un Evento:** El sistema deberá eliminar un evento del sistema si este no está capturando datos de Twitter.
- **RF-37 Recuperar Datos para un Seguimiento:** El sistema deberá recuperar todos los datos necesarios para iniciar una captura.
- **RF-38 Crear un Hilo para la Captura:** El sistema deberá crear un hilo de ejecución independiente para cada captura de datos de Twitter.
- **RF-39 Establecer Conexión en Streaming con la API de Twitter:** El sistema deberá establecer una conexión HTTP permanente con la API en Streaming de Twitter utilizando las claves de API y los filtros seleccionados por el usuario.
- **RF-40 Almacenar Datos recibidos:** El sistema deberá almacenar en la base de datos los "Tweets" capturados de cada evento.
- **RF-41 Cancelar Hilo de Ejecución de una Captura-:** El sistema deberá cancelar el hilo de ejecución de una captura.
- **RF-42 Cerrar Conexión HTTP con la API de Streaming de Twitter:** El sistema deberá cerrar la conexión HTTP en Streaming con la API de Twitter.
- **RF-43 Modificar datos del Usuario:** El sistema deberá permitir al usuario autenticado modificar sus datos de usuario.
- **RF-44 Validar Modificación de Datos de Usuario:** El sistema deberá validar los datos modificados del usuario antes de actualizar la información en la base de datos.

- **RF-45 Mensaje de Error al Modificar Datos de Usuario:** El sistema deberá mostrar un mensaje de error si los datos modificados del usuario no son válidos.
- **RF-46 Listar usuarios de la aplicación:** El sistema deberá mostrar un listado de los usuarios que existen en la aplicación.
- **RF-47 Añadir usuarios:** El sistema deberá añadir nuevos usuarios a la aplicación.
- **RF-48 Modificar usuarios:** El sistema deberá permitir a los usuarios administradores modificar los datos de los usuarios.
- **RF-49 Eliminar usuarios:** El sistema deberá eliminar un usuario del sistema.
- **RF-50 Administrar la Aplicación:** El sistema deberá permitir a los usuarios administradores modificar el fichero de configuración de la aplicación.

4.4. Requisitos no Funcionales

- **RNF-01:** El sistema deberá estar disponible 24h/7días.
- **RNF-02:** Todos los módulos del sistema deberán estar desarrollados con base a la tecnología J2EE y el uso del Framework Primefaces.
- **RNF-03:** Las peticiones rest deberá responderse en un tiempo inferior a 1s, sin tener en cuenta el tiempo de envío de los datos.
- **RNF-04:** La aplicación deberá ser accesible desde un navegador web.
- **RNF-05:** La aplicación deberá estar optimizada para el acceso desde un navegador web de PC.
- **RNF-06:** El sistema deberá soportar como mínimo 500 usuarios concurrentes.
- **RNF-07:** Las contraseñas deberán estar cifradas utilizando un algoritmo seguro SHA-256.

4.4.1. Requisitos de Interfaz Externa

- **RIE-01:** El sistema deberá mostrar los eventos que existen en la herramienta.
- **RIE-02:** EL sistema deberá mostrar los eventos que pertenezcan a un usuario concreto de la herramienta. Recibiendo como parámetro el nombre del usuario.
- **RIE-03:** El sistema deberá mostrar los datos de un evento concreto. Recibiendo como parámetro el nombre del evento.
- **RIE-04:** El sistema deberá mostrar todos los "Tweets" capturados de un evento. Recibiendo el nombre del evento como un parámetro.
- **RIE-05:** El sistema deberá mostrar la geolocalización de "Tweets" de un determinado evento. Recibiendo el nombre del evento como un parámetro.
- **RIE-06:** El sistema deberá mostrar el texto de todos los "Tweets" para un evento concreto. Recibiendo el nombre del evento como un parámetro.
- **RIE-07:** El sistema deberá mostrar la localización de los usuarios que han publicado cada "Tweets" de un determinado evento. Recibiendo el nombre del evento como un parámetro.

- **RIE-08:** El sistema deberá mostrar un resumen del evento. Recibiendo el nombre del evento como un parámetro.
- **RIE-09:** El sistema deberá mostrar datos de los usuarios que han publicado un "Tweet" en un evento concreto. Recibiendo el nombre del evento como un parámetro.
- **RIE-10:** El sistema deberá mostrar el número de "Seguidores" de cada usuario que ha participado en el evento. Recibiendo el nombre del evento como un parámetro.
- **RIE-11:** El sistema deberá mostrar la fecha de publicación de cada uno de los "Tweets" capturados en un evento concreto. Recibiendo el nombre del evento como un parámetro.
- **RIE-12:** El sistema deberá mostrar los datos de los usuarios que han participado en un evento concreto. Recibiendo como parámetro el nombre del evento; y permitiendo establecer un límite del número de resultados que este devuelve.
- **RIE-13:** El sistema deberá leer las respuestas del servicio web proporcionado por "Twitter".

4.5. Requisitos de Información

- **RI-01:** El sistema deberá almacenar información sobre los usuarios de la aplicación. En concreto: nombre de usuario, contraseña, nombre completo de la persona y dirección de correo electrónico.
- **RI-02:** El sistema deberá almacenar información sobre las cuentas de "Twitter" que se utilizan en la herramienta. En concreto: nombre de la cuenta, descripción, nombre completo del propietario y dirección de correo electrónico.
- **RI-03:** El sistema deberá almacenar información sobre las aplicaciones de desarrollo de "Twitter" utilizadas para la conexión con la API. En concreto: cuenta de "Twitter" a la que pertenece, nombre de la aplicación de desarrollo, descripción de la aplicación, Consumer Key(API Key), Consumer Secret (API Secret), Access Token y Access token Secret.
- **RI-04:** El sistema deberá almacenar información sobre los eventos que existen en la herramienta. En concreto: nombre del evento, fecha de inicio del evento, fecha de fin del evento y palabras clave utilizadas para la captura de datos.
- **RI-05:** El sistema deberá almacenar información sobre los seguimientos que existen en la aplicación. En concreto: nombre del evento, lugar donde se almacenan los datos, el usuario propietario del evento, estado en el que se encuentra la captura.
- **RI-06:** El sistema deberá constar de dos bases de datos independientes, una para el funcionamiento de la aplicación y otra base de datos exclusivamente para almacenar los "Tweets" de los seguimientos.
- **RI-07:** El sistema deberá separar los datos de cada uno de los eventos.

4.6. Diseño conceptual de la Base de Datos

En la Figura: 4.11, se puede observar el modelo conceptual que describe la base de datos. Debido a que se trabaja con un "SGBD NoSQL" las relaciones que existen entre las entidades no se verán reflejadas en la base de datos, debido a que MongoDB no implementa claves foráneas. Estas relaciones se llevan a cabo a nivel de la aplicación, siendo esta la encargada de controlar la integridad referencial entre las entidades.

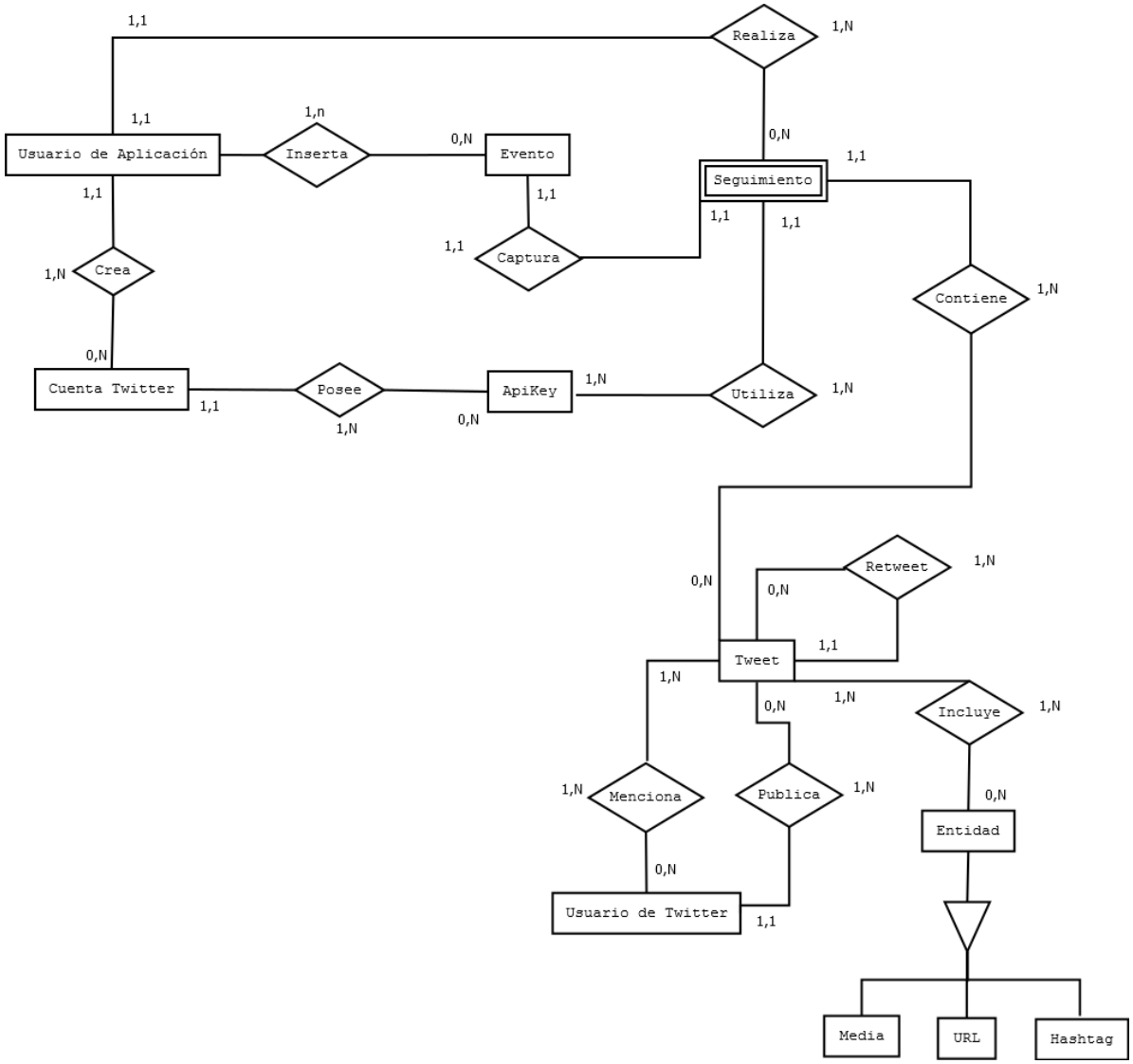


Figura 4.11: Modelo Conceptual de la Base de Datos de la aplicación

Una vez desarrollado el Modelo Conceptual de la Base de Datos el paso a un nivel de detalle inferior no es tan "inmediato" como puede resultar en un modelo relacional. Para poder observar el diseño utilizado en la base de datos, existen ciertas "analogías" que existen entre un modelo relacional y el "SGBD MongoDB", como se puede observar en la tabla: 4.6 .

SGBD Relacional	SGBD MongoDB
Base de Datos	Base de Datos
Tabla	Colección
Tupla	Documento
Columna	Campo
Clave primaria	_id
Group by	Agregación

Tabla 4.6: Comparación términos SGBD Relacional y MongoDB

En la Figura: 4.12 se muestra el modelo lógico que se ha seguido para desarrollar la base de datos "tfg_Twitter", encargada de almacenar la información de la aplicación. Esta base de datos contiene las colecciones necesarias para el funcionamiento de la herramienta. También existe una segunda base de datos llamada "parse4uData", esta base de datos tendrá una colección por cada uno de los eventos que existan en el sistema; en cada colección se almacenarán los "Tweets" capturados.

Figura 4.12: Modelo lógico de la Base de Datos "tfg_Twitter"

El diseño de la base de datos "parse4uData" es excesivamente grande como para poder visualizarse en un gráfico. Dado que en cada uno de los "Tweets" existen más de 150 campos diferentes. Para mostrar los datos que existen en las colecciones que almacenan los "Tweets" se va a describir el modelo de datos de cada "Tweet" con los principales campos de cada uno.

Datos del "Tweet": Estos campos aportan información relativa al "Tweet"

- **In_reply_to_status_id:** Identificación del "Tweet" original, siendo este una respuesta. Se proporciona el dato en formato long.
- **Created_at:** Hora UTC (Coordinated Universal Time o Tiempo universal coordinado) de la publicación del "Tweet". Se proporciona el dato en formato String
- **In_reply_to_user_id_str:** Identificación del usuario del "Tweet" original, siendo este una respuesta. Se proporciona el dato en formato long.
- **Source:** Texto que indica desde que plataforma o aplicación se publicó el "Tweet". Se proporciona el dato en formato String.
- **Retweet_count:** Número de veces que el "Tweet" ha sido retwitteado. Se proporciona el dato en formato int.
- **In_reply_to_screen_name:** Nombre del usuario del "Tweet" original, siendo este una respuesta. Se proporciona el dato en formato String.
- **Facorite_count:** Indica cuantas veces le han dado a "Fav" al "Tweet". Se proporciona el dato en formato int.
- **Id:** Identificador del "Tweet". Se proporciona el dato en formato long.
- **Text:** Texto del "Tweet". Se proporciona el dato en formato String.
- **Place:** Si el "Tweet" está asociado a un lugar, este valor indicara el nombre del lugar. Se proporciona el dato en formato String.
- **Lang:** identificador del idioma en el que fue escrito el "Tweet". Se proporciona el dato en formato String.
- **Coordinates:** Representa la ubicación geográfica del "Tweet" (Longitud, Latitud). Se proporciona el dato en formato float.

Datos del usuario: Estos campos aportan información relativa al usuario que publicó el "Tweet"

- **User-friends_count:** Indica el número de seguidores que tenía el usuario cuando publicó el "Tweet". Se proporciona el dato en formato int.
- **User-profile_image_url_https:** Aporta la URL de la imagen de perfil del usuario. Se proporciona el dato en formato String.
- **User-profile_background_image_url:** Aporta la URL de la imagen de fondo del perfil de usuario. Se proporciona el dato en formato String.
- **User-default_profile_image:** Campo booleano que indica si el usuario ha modificado su imagen de usuario o fondo (especialmente útil para la identificación de bots o perfiles falsos).
- **User-favourites_count:** Número de "Tweets" que el usuario ha marcado como favorito. Se proporciona el dato en formato int.
- **User-description:** Texto que el usuario ha incluido como su descripción en el perfil de usuario. Se proporciona el dato en formato String.
- **User-created_at:** Fecha y hora UTC en la que se creó la cuenta del usuario. Se proporciona el dato en formato String.
- **User-protected**: Indica si el usuario tiene configurado su perfil como protegido, solo podrán ver sus "Tweets" los usuarios que sigan y sean seguidos por el usuario. Se proporciona el dato en formato String.

- **User-screen_name:** Nombre de usuario o alias que identifica a este usuario. Se proporciona el dato en formato String.
- **User-id:** identificador único del usuario. Se proporciona el dato en formato long.
- **User-geo_enabled:** Indica si el usuario tiene habilitada la geoposición en sus "Tweets".Se proporciona el dato en formato String.
- **User-lang:** Idioma de la interfaz de usuario. Se proporciona el dato en formato String.
- **User-verified:** Indica si la cuenta de "Twitter" que publicó el mensaje es una cuenta verificada. Se proporciona el dato en formato String.
- **User-time_zone:** Zona horaria del usuario. Se proporciona el dato en formato String.
- **User-url:** URL proporcionada por el usuario que se asocia a su perfil. Se proporciona el dato en formato String.
- **User-statuses_count:** Indica el número de "Tweets" que ha publicado el usuario. Se proporciona el dato en formato int.
- **User-followers_count:** Indica el número de seguidores que tiene el usuario en el momento de publicación del "Tweet". Se proporciona el dato en formato int.
- **User-location:** Indica la localización predefinida del usuario. Se proporciona el dato en formato String.

Datos sobre los usuarios mencionados: Estos datos aportan información sobre los usuarios a los que se menciona e un "Tweet"

- **Entities-user_mentions-indices:** Un array de enteros que representa, dentro del texto del "Tweet" en qué posición comienza y termina la referencia al usuario.
- **Entities-user_mentions-screen_name:** Nombre del usuario al que se hace referencia. Se proporciona el dato en formato String.
- **Entities-user_mentions-id:** Identificador del usuario a que se hace referencia. Se proporciona el dato en formato int.
- **Entities-user_mentions-name:** Nombre que se muestra del usuario al que se hace referencia. Se proporciona el dato en formato String.

Datos sobre las entidades: Las entidades son contenido añadido a un "Tweet", estas pueden ser de tipo Media(Imágenes o vídeos), de tipo URL (enlaces a otras páginas) y de tipo Hashtag.

- **Entities-hashtag-indices:** Un array de enteros que indican en qué posición comienza y termina un hashtag dentro del texto del "Tweet".
- **Entities-hashtag-text:** Nombre del hashtag utilizado, excluyendo el símbolo "#". Se proporciona el dato en formato String.
- **Entities-urls-indices**: Representa las posiciones donde comienza y termina el enlaces dentro del texto del "Tweet". Se proporciona el dato en formato array de enteros.
- **Entities-urls-url:** Texto que representa el valor de la URL que se ha incluido en el texto. Se proporciona el dato en formato String.
- **Entities-media-dysplay_url:** Dirección URL de los multimedias que se muestra a los usuarios. Se proporciona el dato en formato String.

- **Entities-media-indices:** Array de enteros, que representa las posiciones dentro del texto del "Tweet" donde comienza y termina la referencia el fichero multimedia.
- **Entities-media-sizes:** Muestra los tamaños disponibles del archivo multimedia. Se proporciona el dato en formato String.
- **Entities-media-id:** Identificador de los ficheros multimedia. Se proporciona el dato en formato id.
- **Entities-media-type:** Indica el tipo de fichero que se ha incluido. Se proporciona el dato en formato String.
- **Entities-media-media_url:** Dirección URL que apunta directamente al fichero multimedia. Se proporciona el dato en formato String.

Estos son solo los atributos más relevantes de cada "Tweet", existen otros campos, pero carecen de utilidad, debido a que están en desuso o son valores duplicados utilizando otros formatos. Todos estos atributos, están presentes en cada uno de los "Tweets" que se capturan por el sistema, por lo que por una parte existe mucha información duplicada, por otra parte, nos permitirá ver una evolución de un perfil de usuario a lo largo del tiempo.

Capítulo 5

Diseño Software

En este capítulo del documento, se procederá a describir los diferentes componentes y la arquitectura lógica y física que presenta el sistema. Para ello, se proporcionará una breve introducción de en qué consisten las arquitecturas lógica y física; y posteriormente se detallará en profundidad la arquitectura de PARSE4U.

5.1. Arquitectura Lógica

La arquitectura lógica expresa cuales son los componentes lógicos que participan en el sistema y la relación entre ellos. La arquitectura lógica se puede ver gráficamente en la Figura:5.1

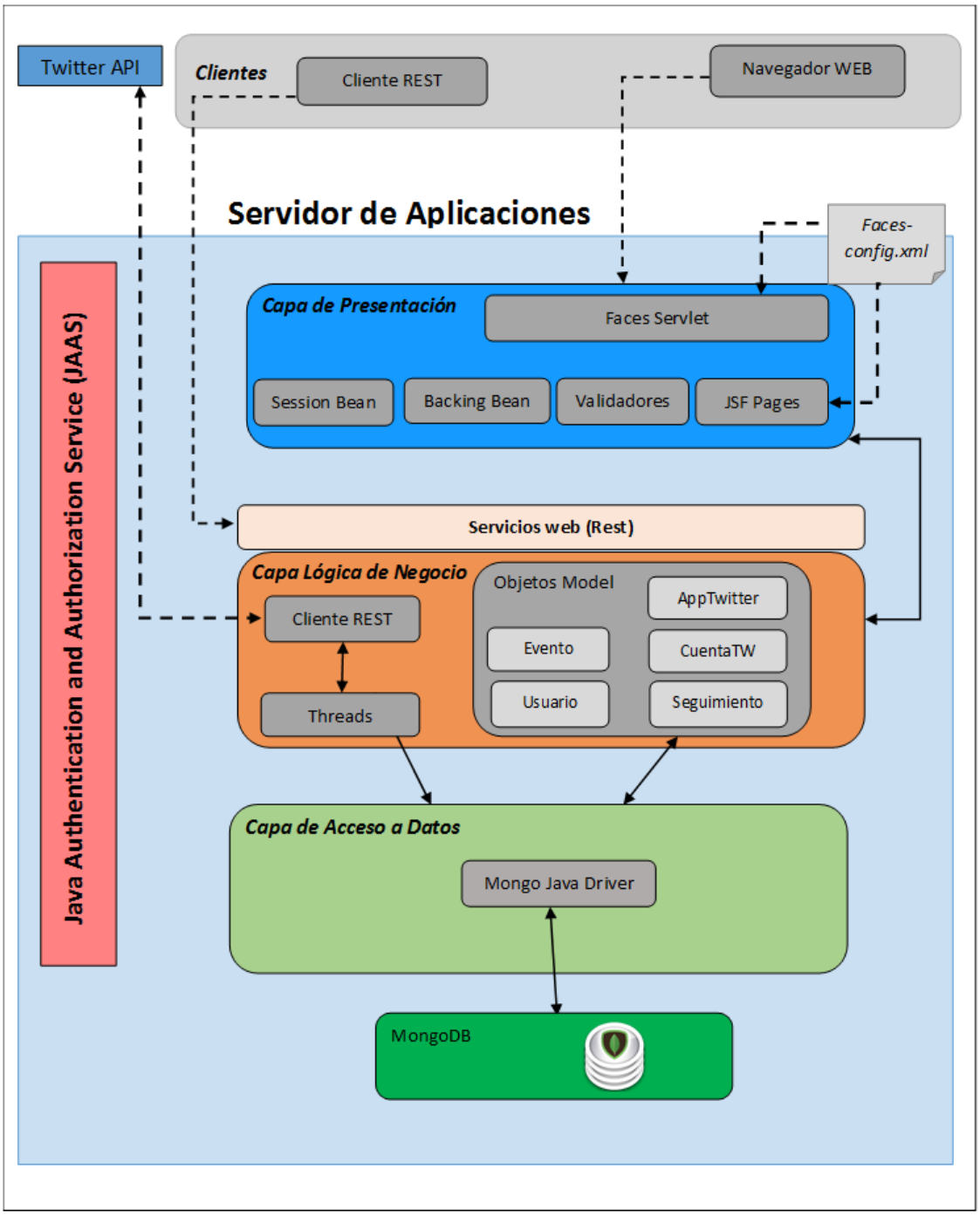


Figura 5.1: Arquitectura lógica del sistema

A la hora de diseñar la arquitectura lógica se ha elegido un modelo sencillo de tres capas, siendo la primera de ellas la relacionada con la capa de presentación y la parte cliente (que en este caso corresponde con las tecnologías relacionadas con JSF), una segunda capa que incluye toda la lógica de negocio de la parte servidor, y una tercera capa para el acceso a los datos. Transversal a las capas hay una capa para dotar de seguridad a la aplicación, implementada en este caso utilizando la API JAAS y utilizando un Realm específico para la base de datos.

En la primera capa se encuentran los componentes de JSF (Servlet y pages), los componentes de lógica de negocio del cliente, que son los session beans y los backing beans, así como los conversores y validadores usados (validadores para formularios y conversores como SHA-256).

En la capa intermedia encontramos la capa de lógica de negocio. Aquí encontramos todos los objetos "Modelo" que representan las entidades de la base de datos. Tambien encontramos en esta capa, las clases encargadas de la gestión de los múltiples hilos de ejecución. Tambien encontramos aquí, el cliente que se conectará a la "API de Twitter" para obtener los datos.

Por encima de la capa intermedia (Lógica de Negocio), encontramos los servicios Rest que nuestra aplicación expone al exterior, para que los clientes Rest puedan utilizar los servicios web para recuperar los datos capturados por la herramienta.

En la última capa se encuentra los relacionado con el acceso a datos; en este caso al no utilizar una base de datos relacional no se hace uso de la API JPA, sino que se utiliza el driver oficial de MongoDB para Java. Al utilizar una base de datos no relacional, esta no permite la integridad referencial por lo que esta característica recae sobre el programa.

Así mismo, para poder hacer uso de la API JAAS, se necesitó de un Real específico para poder trabajar con mongoDB, para ello se modificó y compilo un Realm propio utilizando como base el código de "Fabienvauchelles", este código está disponible en GitHub, bajo una licencia GNU, se puede encontrar en: GitHub. [1]

5.1.1. ¿Por qué elegir MongoDB?

Una de las principales características de trabajar con la API de una red social tan extensa como es "Twitter", implica ser consciente del gran volumen de datos a los que la herramienta se enfrenta. Por hacernos una idea, durante el Mobile World Congress, se realizó un seguimiento del "hashtag" oficial y durante la semana que duró el evento se capturaron más de 4Gb de datos. Teniendo en cuenta que esto es solo un evento, si lo extrapolamos a la cantidad de usuarios y eventos que nuestra herramienta puede utilizar, nos encontramos con el problema de que una base de datos Relacional como puede ser MySql puede no ser la mejor opción para el almacenamiento de los "Tweets".

Debido al problema en el volumen de los datos y como aprendizaje para el alumno, se decidió utilizar una base de datos no relacional. Existen múltiples bases de datos de este tipo, en función del tipo de almacenamiento que busque:

- **Orientadas a documentos:** Son aquellas que gestionan datos semi-estructurados. Estos datos son almacenados en algún formato estándar como XML, JSON o BSON. Son las bases de datos más versátiles y populares. Entre ellas nos encontramos: MongoDB y CouchDB.
- **Orientadas a columnas:** Este tipo de bases de datos están orientadas consultas y agregaciones sobre grandes volúmenes de datos. Funcionan de una forma similar a las bases de datos relacionales, pero almacenando la información en columnas en lugar de en registros. Algunas de las más conocidas son: LucidDB y HBase. Esta última, se encuentra dentro del proyecto Apache Hadoop.

[1] https://github.com/fabienvauchelles/glassfish-mongo-realm

- **De clave valor:** Estas son las más sencillas. Simplemente almacenan tuplas que contienen una clave y su valor. Cuando se quiere recuperar un dato, se busca su clave y se recupera el valor. En esta categoría encontramos: Cassandra ,DynamoDB y Redi.
- **En grafo:** Basadas en la teoría de grafos, utilizan nodos y aristas para representar los datos almacenados. Son muy útiles para guardar información en modelos con múltiples relaciones, como redes y conexiones sociales. Podemos encontrar: Infinite Graph y Neo4j.

El primer motivo de elegir como "SGBD"(Sistema Gestor de Bases de Datos) MongoDB fue su popularidad. Al ser la base de datos NoSql más popular se entendía que se encontraría más información para su integración con una plataforma "JavaEE". El segundo motivo de esta elección reside en el cómo se reciben los datos de la API de "Twitter" y como se ofrecerán a los clientes a través de Servicios Web. Tanto la API de la red social como nuestra API ofrecen los datos en un formato JSON. Al utilizar MongoDB el formato JSON/BSON, nos facilita la tarea de "ingestión" dado que el procesamiento de los datos de entrada es prácticamente nulo.

Otra ventaja de utilizar MongoDB es que este "SGBD" no necesita seguir un esquema en los datos, por lo que no todos los "Tweets" deben contener toda la información, sino que sigue una estructura dinámica; y está especialmente diseñado para garantizar la escalabilidad debido a sus opciones de replicación y "sharding" permiten tener un sistema escalable horizontalmente sin complicaciones.

5.2. Arquitectura Física

La arquitectura física representa cuales son los componentes físicos (clientes web, clientes rest, servidores, bases de datos...) que forman el sistema, así como las relaciones entre ellos.

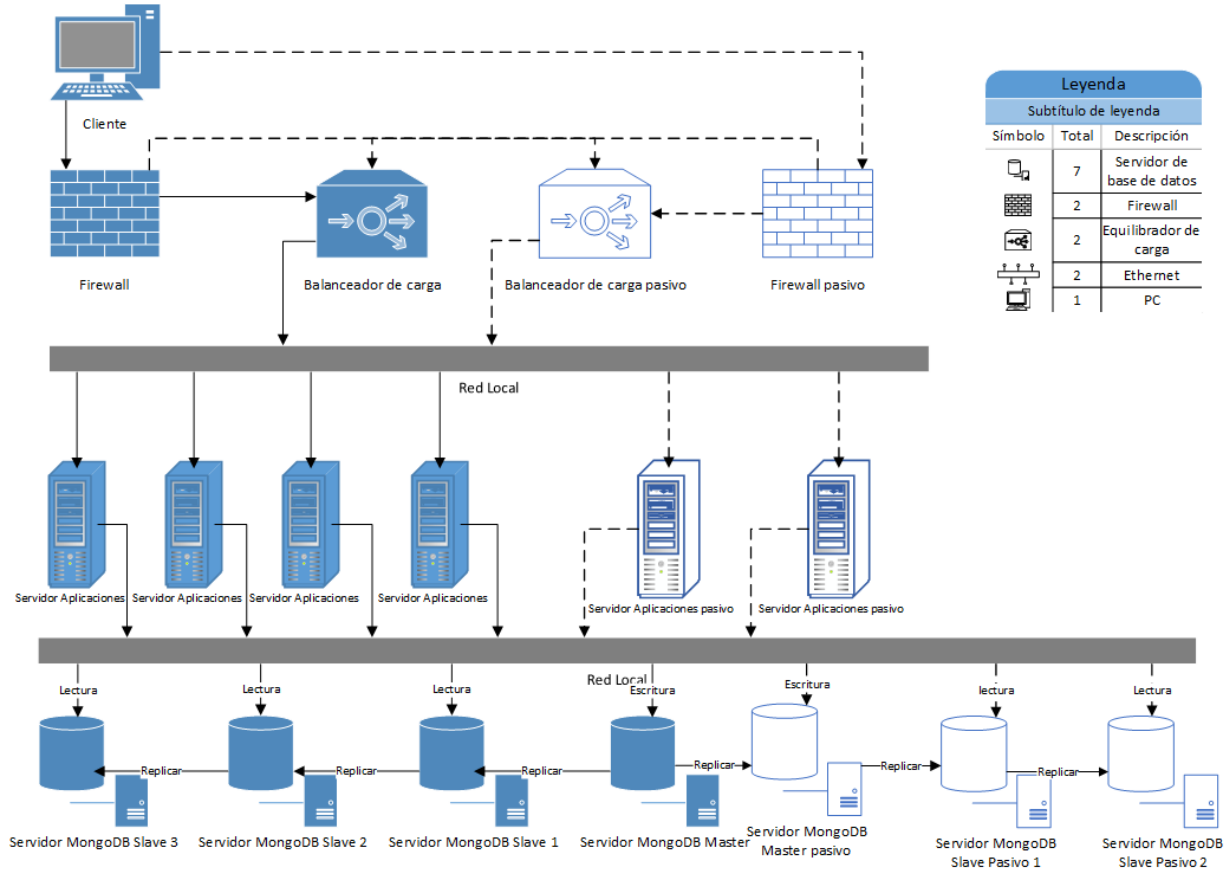


Figura 5.2: Arquitectura física PARSE4U

A la hora de diseñar la arquitectura física se ha buscado cumplir en la mayor medida posible los requisitos no funcionales de seguridad, escalabilidad, disponibilidad y rendimiento; y se ha tenido menos en cuenta el posible gasto de la compra e implantación de dicha arquitectura.

Como se observa en la Figura 5.2, de todos los servidores existen al menos una réplica pasiva, permitiendo de esta manera activar el servidor pasivo siempre que alguno de los servidores activos no esté disponible. Si en algún momento muy puntual se excede el número máximo de usuarios esperados, se pueda activar de forma temporal para evitar así la congestión en los servidores.

En la entrada, se ha situado un firewall o cortafuegos, para poder filtrar las peticiones que recibe el sistema. De igual manera, se ha situado un balanceador de carga activo (y uno pasivo en caso de fallo), que permite dividir la carga entre los servidores de aplicaciones.

Los servidores de aplicaciones son los encargados de recibir las peticiones y responder con la información solicitada por los clientes. De igual manera, todos los servidores de aplicaciones se conectan con los servidores de bases de datos. En este caso, en función de si las peticiones a la base de datos son de escritura o lectura tenemos: en caso de solicitar una escritura, se envía la información al "Servidor MongoDB Master" y este una vez realizada la escritura, realiza una replicación de la información en el resto de servidores; en caso de que la petición sea una lectura, existen 3 servidores de aplicaciones "Mongo Slave" que serán los encargados de procesar las peticiones de lectura.

En el caso de los servidores de aplicaciones, se han situado dos servidores de aplicaciones suplementarios para permitir el correcto funcionamiento en caso de fallo de alguno, o en casos de excesivas peticiones. Para los servidores de bases de datos, tenemos una réplica pasiva del "servidor MongoDB Master" y dos réplicas para los "servidores Mongodb Slave". De esta manera tendremos casi garantizado un funcionamiento ininterrumpido del sistema.

A la hora de seleccionar la cantidad de servidores, se estima que el sistema puede llegar a tener 5000 usuarios concurrentes, estimando que cada servidor de aplicaciones es capaz de manejar 1000 usuarios. En el caso de los servidores de bases de datos, si suponemos que cada uno de los servidores tiene 6Gb de ram, cada servidor puede manejar entorno a 5500 peticiones simultáneas. Por lo que no debería existir ningún problema en cuánto al rendimiento bajo gran carga de trabajo.

5.3. Diagrama de Clases

Los diagramas de clases de diseño permiten conocer las principales clases que forman parte del sistema. En ellos se muestran las clases del modelo de dominio y las relaciones entre estas clases.

En este caso se utiliza el patrón de diseño Modelo-Vista-Controlador. Este patrón separa los datos y la lógica de negocio de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Se divide la arquitectura en tres componentes:

- **Modelo:** El modelo es la representación de la informacion con la cual opera el sistema opera, gestionando las consultas a los datos. Envía a la "Vista" aquella parte de la informacion cuando sea necesario ser mostrada. Las peticiones de acceso o manipulación de la informacion llegan al "Modelo" a través del "Controlador".
- **Controlador:** Responde a eventos e invoca peticiones al "Modelo" cuando se realiza alguna solicitud sobre la informacion.
- **Vista:** Presenta el "Modelo" en un formato adecuado para interactuar (habitualmente mediante una interfaz de usuario) por tanto, requiere que el "Modelo" proporcione la informacion que debe representar.

En la Figura 5.3, se pueden ver las clases principales del proyecto, como el diagrama completo puede resultar demasiado complejo, se ha decidido incluir en la memoria un diagrama reducido. En el CD-ROM se puede encontrar una versión extendida del diagrama de clases completo.[2]

Las clases Usuario, Evento, CuantaTW, AppTwitter y Seguimiento representan la capa del modelo. Las clases "Bean" son las encargadas de la interacción con los objetos "Model" y las vistas. Estas clases componen la capa "Controlador". En esta capa encontramos las clases UsuarioSessionBean, EventoSessionBean, CuentaTWSessionBean, AppTWSessionBean y SeguimientoSessionBean. En el diagrama solo se muestran las variables y métodos de la clase SeguimientoSessionBean y se han omitido en las otras clases debido a su similitud y para simplificar el diagrama.

Existe un tipo especial de "bean", el llamado AppBean, este "bean" es común a toda la aplicación y se encuentra siempre en ejecución. En este objeto se almacenan las referencias al objeto Tarea que estén en uso en la aplicación. La clase llamada "WebController" implementa todos los servicios web que esta herramienta debe implementar.

[2] \Diagramas\Diagrama_de_Clases_completo

El diagrama de clases mostrado en la Figura: 5.3 es una version reducida de el diagrama de clases.En diagrama completo adjunto en el CD, se pueden apreciar todos los métodos y variables de estas clases. Así mismo, se incluyen las clases "BackingBean" estas clases se utilizan para almacenar la información de los formularios de manera temporal. También se pueden ver en el diagrama completo, las clases encargadas de la conexión con la base de datos; de igual manera se incluyen las clases encargadas de la validación de datos desde el lado servidor, estas clases se utilizan para realizar una segunda validación de los datos introducidos en los formularios.

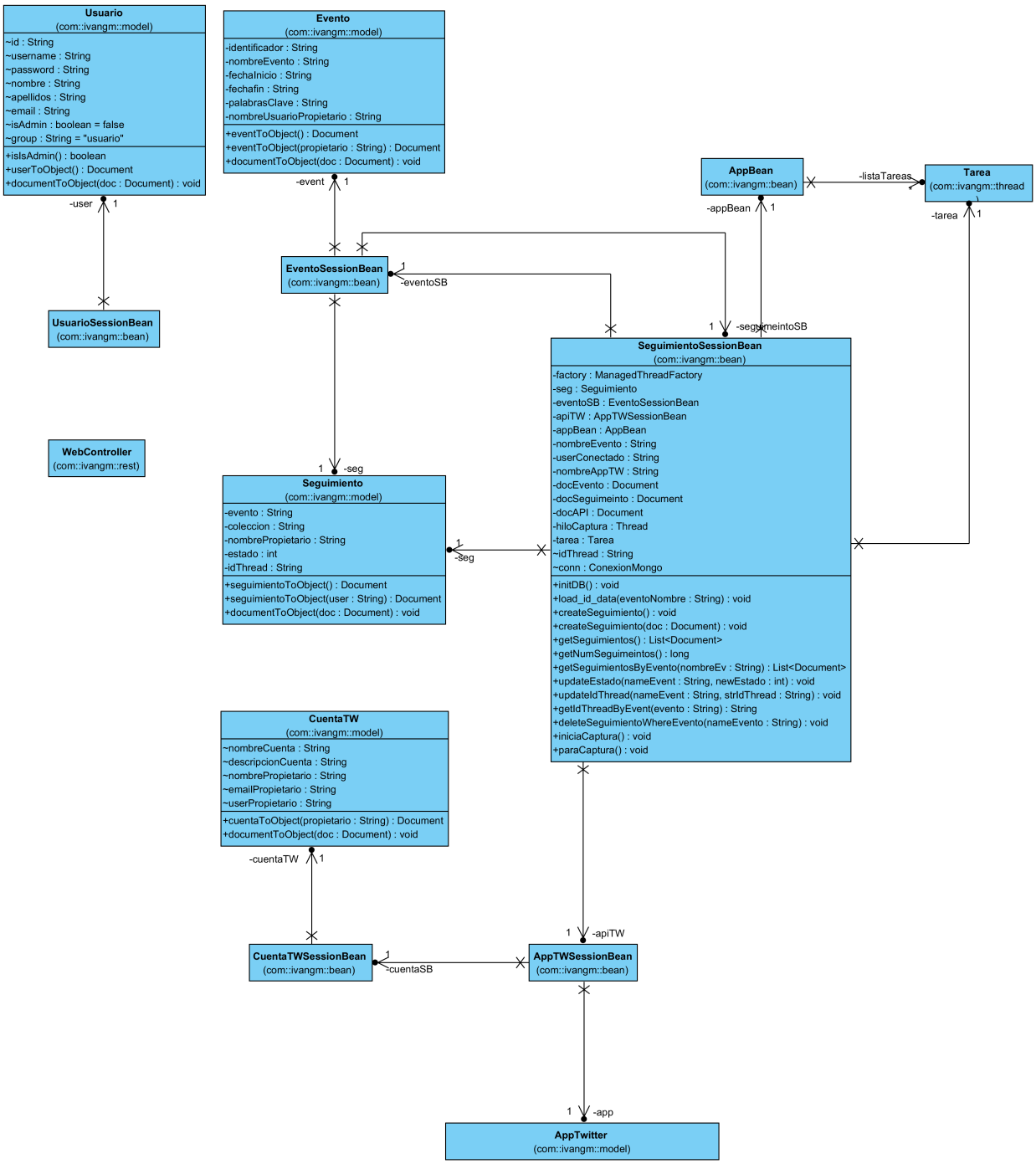


Figura 5.3: Diagrama de Clases PARSE4U

5.4. Diagramas de Secuencia

En las siguientes figuras, se pueden ver los diagramas de secuencia que muestran la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso. Debido a la similitud en las operaciones de "Creación", "Modificación" y "Eliminación", se han incluido los Diagramas de Secuencia para las operaciones "CRUD" de una Evento; se podría extender a las otras entidades dada su similitud.

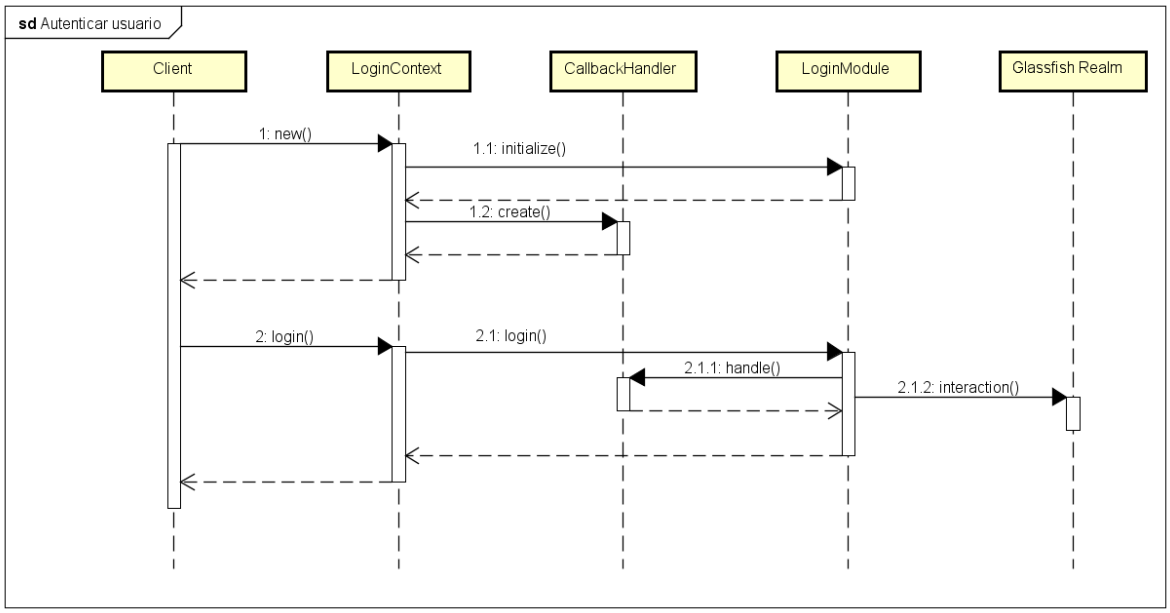


Figura 5.4: Diagrama de Secuencia del Caso de Uso "Autenticar Usuario"

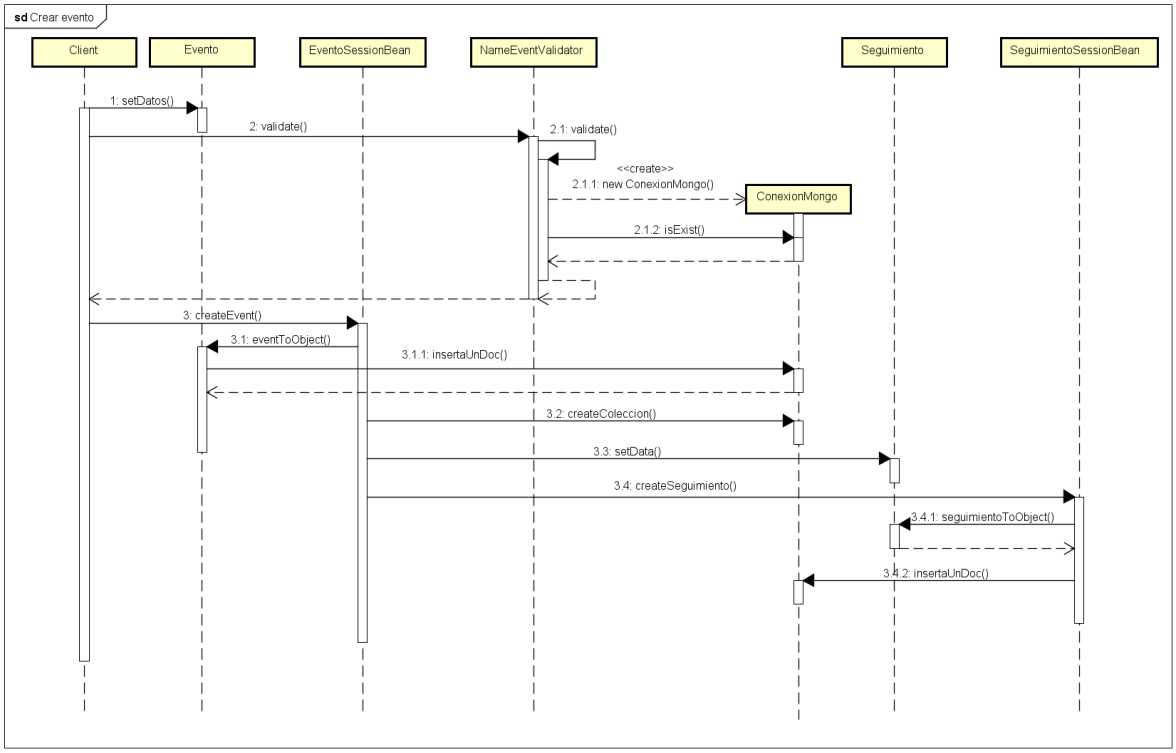


Figura 5.5: Diagrama de Secuencia del Caso de Uso “Crear Nuevo Evento”

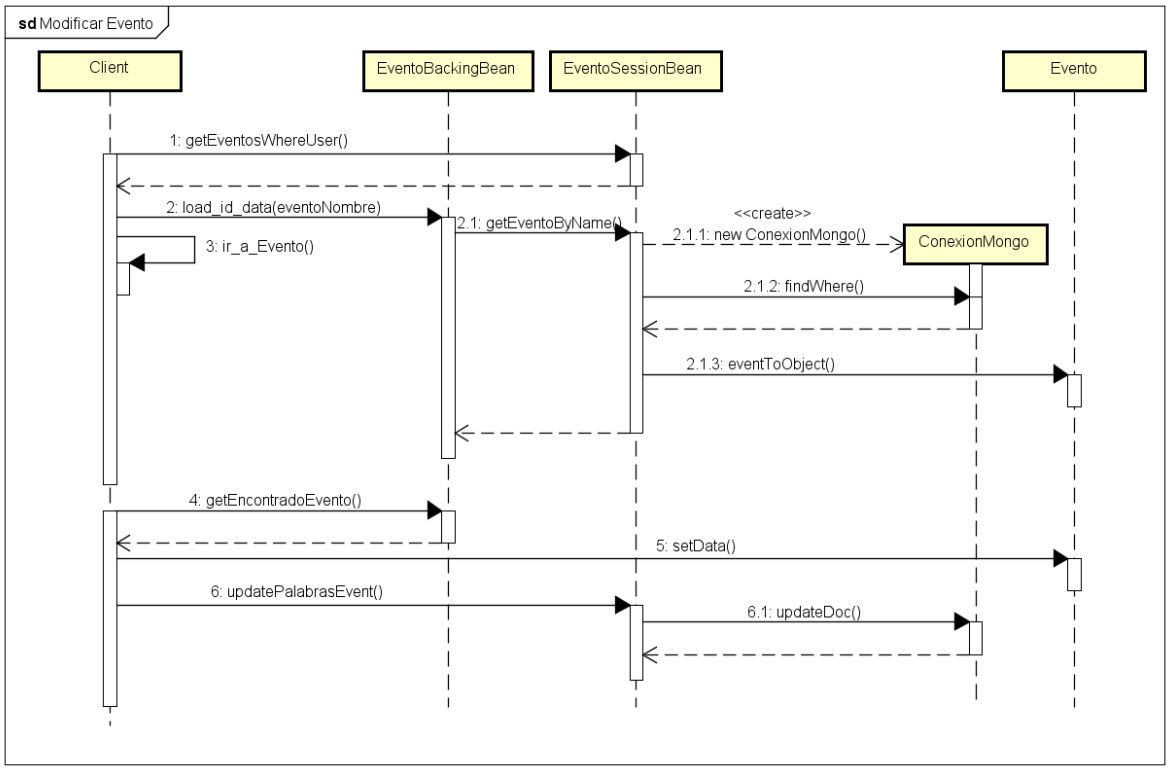


Figura 5.6: Diagrama de Secuencia del Caso de Uso “Modificar Evento”

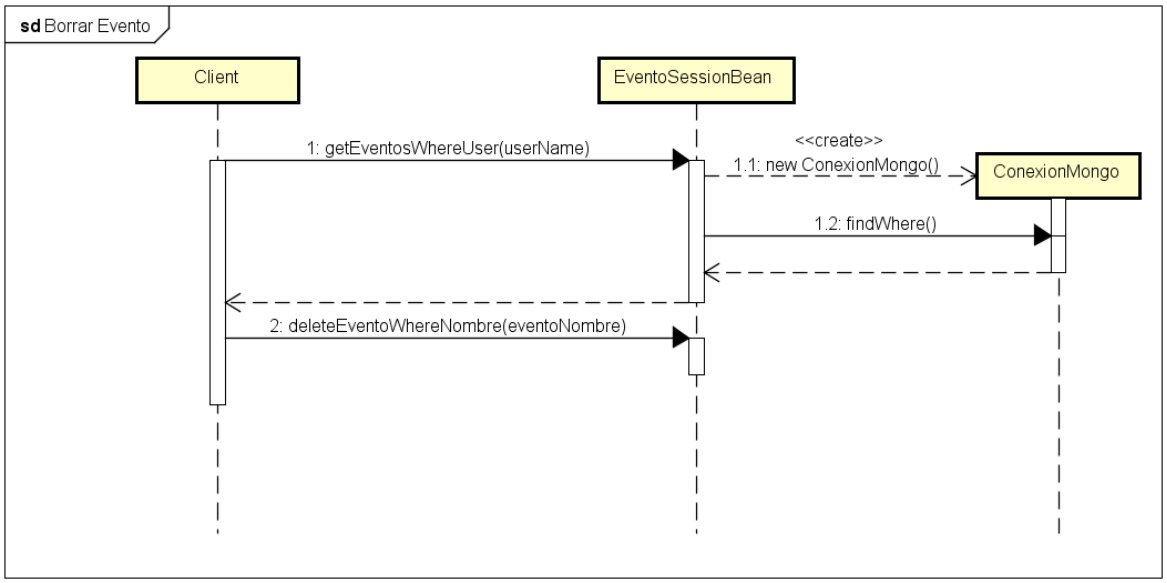


Figura 5.7: Diagrama de Secuencia del Caso de Uso "Borrar Evento"

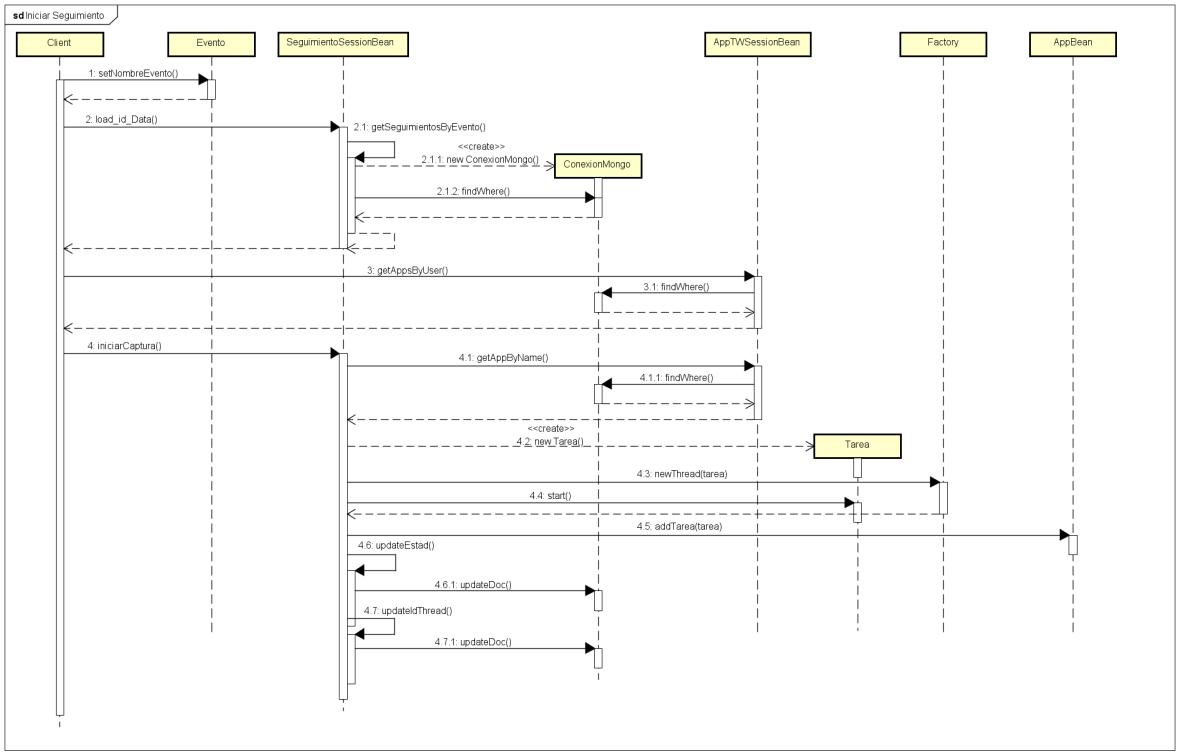


Figura 5.8: Diagrama de Secuencia del Caso de Uso "Iniciar Captura"

5.5. Diseño de la Interfaz

Este apartado sirve para mostrar una aproximación de cómo va a ser la interfaz de usuario del sistema.

En la Figura: 5.9, se puede ver un prototipo de la interfaz de la aplicación, esta se usará para autenticar un usuario en el sistema.

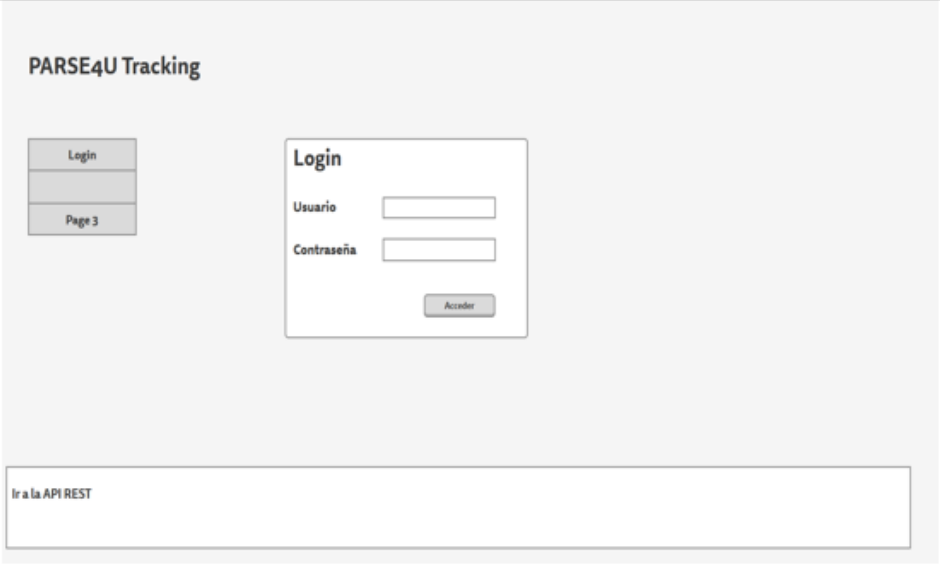
Nombre	Autenticación
Descripción	Página de acceso a la herramienta mediante un formulario donde se introducen los credenciales del usuario
Activación	El usuario pulsa el enlace de login.
Boceto	
Eventos	Identificarse

Figura 5.9: Prototipo Interfaz de la Pagina de Login

En la Figura: 5.10, se puede ver un prototipo de la interfaz donde el usuario podrá ver un listado de los eventos creados por él.

Nombre	Ver Eventos
Descripción	Página que proporciona un listado de los eventos que el usuario a creado.
Activación	El usuario pulsa el enlace de Ver Eventos.
Boceto	
Eventos	Editar, borrar, captura

Figura 5.10: Prototipo Interfaz de la Pagina de Ver Eventos

En la figura: 5.11, se puede ver un prototipo de la interfaz de usuario que se utilizará para añadir nuevos eventos al sistema.

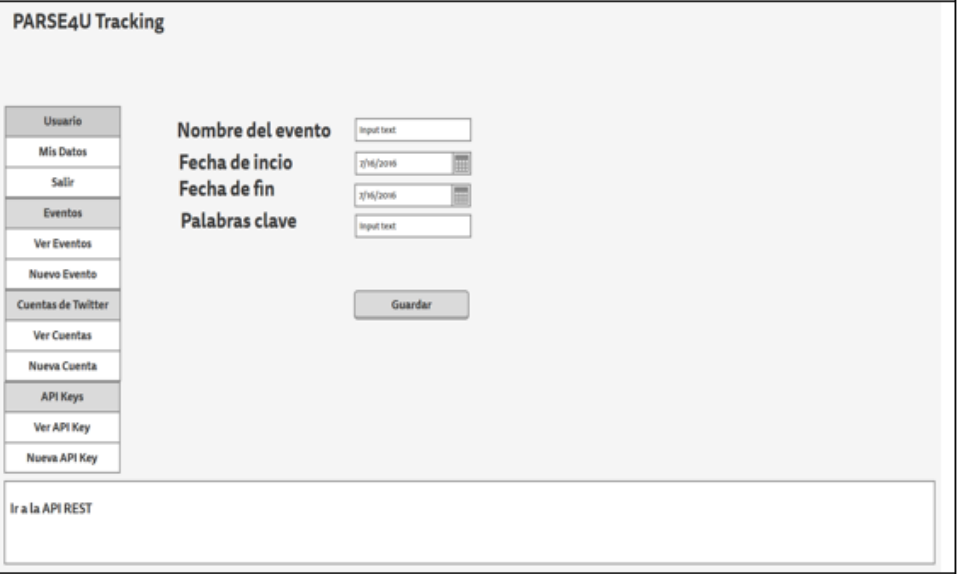
Nombre	Nuevo Evento
Descripción	Página que permite a los usuarios añadir nuevos eventos en el sistema
Activación	El usuario pulsa el enlace de Nuevo Evento
Boceto	
Eventos	Guardar

Figura 5.11: Prototipo Interfaz de la Pagina de Nuevo Evento

En la Figura: 5.12, se puede ver un prototipo de la interfaz de usuario que se utilizará para iniciar la captura de un evento.

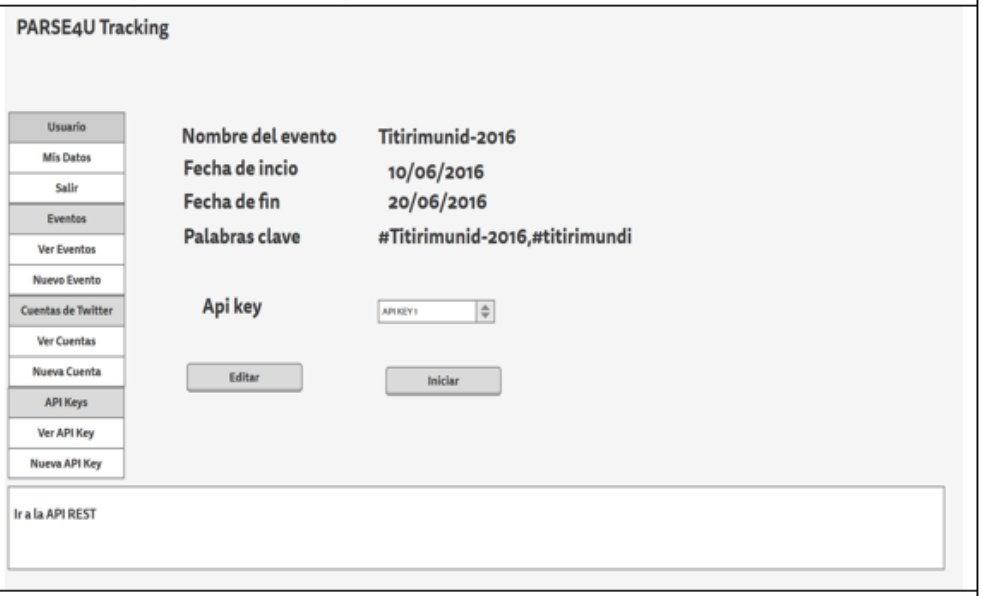
Nombre	Iniciar Captura
Descripción	Página que permite a los usuarios iniciar la captura de un evento
Activación	El usuario pulsa el botón Captura de un evento
Boceto	
Eventos	Editar, iniciar

Figura 5.12: Prototipo de Interfaz de la Pagina de Iniciar Captura

Capítulo 6

Implementación del Proyecto

En este capítulo se explicará cómo se ha llevado a cabo la implementación del proyecto. Para ello se hará una breve descripción del servidor de aplicaciones utilizado y su configuración; así mismo, se explicará la implementación de la base de datos y la conexión con la herramienta.

6.1. Descripción Técnica

La herramienta implementada se desarrolla en torno a la captura de datos provenientes de "Twitter". Esto implica que nuestra herramienta debe establecer múltiples conexiones con la API de "Twitter", una conexión por cada evento que se desee capturar. Para garantizar el correcto funcionamiento y la escalabilidad de la herramienta, cada una de las conexiones con la API de "Twitter" se ejecutan en un hilo de ejecución propio. De esta manera existirán tantos hilos en ejecución como capturas activas existan. De esta manera, cuando el usuario cierre sesión y abandone la aplicación, el hilo de ejecución permanecerá con una conexión en "streaming" con la API.

Una vez que un evento recibe un "Tweet" el sistema lo convierte en un JSON utilizando un "parser" propio del driver de MongoDB; se inicia una conexión con la base de datos y se libera el recurso, permitiendo así que existan múltiples inserciones de manera simultánea sin aumentar el consumo de recursos. Tanto en la escritura de "Tweets" como en el funcionamiento de la herramienta, se mantiene la conexión con la base de datos el menor tiempo posible para que no existan problemas en el consumo de memoria cuando el sistema tenga múltiples usuarios y capturas.

Esta herramienta posee tres aspectos innovadores frente a otras herramientas existentes. Primero, el utilizar un sistema gestor de bases de datos no-relacional, lo que implica que las herramientas habituales para el desarrollo como IDEs o la API de persistencia de Java, no están preparadas todavía para utilizar estas bases de datos, por lo que presentan un reto en el desarrollo. Segundo, para permitir que existan múltiples hilos de ejecución y estos estén dentro del control del servidor de aplicaciones Glassfish, se hace uso de la API de Concurrencia de Java EE 7; aunque la especificación de Java EE 7 es del año 2013, la API de Concurrencia es relativamente nueva. Tercero, el permitir a los clientes disponer de un servicio Rest que les proporcione todos los datos capturados en formato JSON; a diferencia de otras herramientas similares, que solo permiten realizar análisis en base a los datos de "Twitter" o permiten exportar solo algunos atributos de cada "Tweet" y con muchas limitaciones temporales.

6.2. Paso de la Arquitectura Lógica a la Implementación

La implementación del proyecto se ha realizado basándose en la arquitectura lógica mostrada en la Figura 5.1, de modo que todo lo que se representa en la arquitectura lógica se vea implementado en el desarrollo final.

A la hora de desarrollar el proyecto, una de las primeras decisiones que se tomaron fue el utilizar MongoDB como sistema gestor de bases de datos. Esto es debido a que permitía almacenar un gran volumen de datos sin problemas de rendimiento; es de código abierto y gratuito. Otra de las cuestiones que nos hacen decantarnos por este SGBD es el hecho de que trabaje con documento JSON, lo que nos permite una perfecta integración con los servicios Rest de "Twitter" y nuestros propios servicios Rest.

Una vez elegida la base de datos que se utilizaría, uno de los primeros problemas con los que tuvimos que lidiar, es el hecho de que la API de Autenticación y Autorización de Java "JAAS" no permite autenticar usuarios de una base de datos MongoDB de una manera nativa. Para ello se recurrió a un proyecto publicado en GitHub de código libre: glassfish-mongo-realm [1], este proyecto del usuario "fabienvauchelles" sumado a unas modificaciones, nos permitieron poder configurar la autenticación de usuarios utilizando la propia API de Java.

Una vez comenzado el desarrollo de la herramienta, estudiamos las posibilidades existentes para conectar una aplicación en Java EE con una base de datos MongoDB. Aunque existen varios "framework" como Hibernate OGM [2] o Eclipselink [3], estos todavía se encuentran en desarrollo y no existen versiones estables, por lo que no son totalmente recomendados. Existen otros "framework", se puede ver en: MongoDB EcoSystem Java [4]. Por lo que se decidió utilizar directamente el driver de MongoDB para Java e implementar las operaciones necesarias. Estas operaciones han sido implementadas de manera que sean lo más re utilizables posibles, tanto dentro del proyecto como para otros proyectos externos.

El realizar operaciones con el driver de MongoDB implica un cambio en la manera tradicional de realizar consultas a una base de datos. En MongoDB las consultas se realizan utilizando filtros, estos filtros se componen de "Documentos JSON". Durante el desarrollo del proyecto, se observó un problema en el consumo de memoria del sistema; analizando el funcionamiento, se comprobó que el driver de MongoDB para Java no liberaba correctamente los recursos, lo que implicaba que, tras realizar múltiples conexiones con la base de datos, el servidor de aplicaciones se quedaba sin memoria ram. Para solventar ese problema, se modificaron las operaciones a la base de datos, liberando manualmente los recursos una vez que finaliza cada una de las consultas.

Para la implementación de los servicios Rest se utilizó Jersey [5], que es una implementación de JAX-RS (Java API for Rest Ful Web Services). Para el desarrollo de los servicios web, se han implementado los servicios Rest necesarios para la construcción de un prototipo de PARSE4U, que ofrece un conjunto sencillo de analíticas sobre datos de "Twitter".

[1] Glassfish-mongo-realm https://github.com/fabienvauchelles/glassfish-mongo-realm
[2] Hibernate OGM: http://hibernate.org/ogm/
[3] Eclipselink: http://www.eclipse.org/eclipselink/
[4] MongoDB Ecosystem Java: https://docs.mongodb.com/ecosystem/drivers/java/#third-party-frameworks-and-libraries
[5] Jersey: https://jersey.java.net/

Para que los posibles clientes puedan ver los servicios Rest implementados, comprobar su funcionamiento y ver el formato de cada una de las peticiones rest, se ha utilizado el "framework" Swagger [6]. Está desarrollado para ser utilizado en Scala, Java, JavaScript, Ruby, PHP o ActionScript. Además, existen módulos para enganchar con proyectos en Node.js, Grails o Spring. De esta forma con unas pequeñas anotaciones en el código de los servicios Rest, permitimos que se genere automáticamente la documentación de los servicios Rest en el mismo momento en que el servicio ha sido implementado.

Para permitir que cada uno de los usuarios de la aplicación pudiera establecer una conexión con la API de "Twitter" y que permaneciera abierta incluso cuando el usuario abandonara la herramienta, era necesario que cada captura se ejecutara en un hilo exclusivo para esa captura. Pero si estos hilos no son controlados por el servidor de aplicaciones, podrían dejar sin memoria disponible al servidor de aplicaciones y causar la caída del servicio. Para solventar estos problemas, se utilizó la API de concurrencia de Java. Cuando el usuario inicia una nueva captura, se crea un nuevo hilo de ejecución utilizando la "factoría" de hilos de Java EE 7. Cada hilo implementa la interfaz "Runnable". Una vez que el hilo ha sido creado, se inicia su ejecución, llamando al método "Run" del hilo. Este método establece la conexión con la API de "Twitter". Una vez que se ha establecido la conexión, se almacena la referencia al hilo de ejecución en un "Application Bean" común a todos los usuarios, lo que nos permite, conocer cada hilo de ejecución que usuario inicio la captura y que datos está capturando. Debido a que solo se almacenan las referencias a los objetos, no existen problemas de memoria si el número de capturas activas es elevado.

6.3. Herramientas Empleadas en la Implementación del Proyecto

En este apartado se exponen y explican las tecnologías y herramientas que se han utilizado para el desarrollo del sistema.

- **Java EE 7**: Anteriormente conocido como Java Enterprise Edition o J2EE, es una plataforma de programación que tiene como base el lenguaje de programación Java, para desarrollar y ejecutar aplicaciones empresariales. Permite utilizar arquitecturas de N capas distribuidas y se apoya en componentes modulares, ejecutándose en un servidor de aplicaciones. Java EE se compone de varias especificaciones de API como: JDBC, RMI, JMS, Servicios Web, XML ... También posee algunas especificaciones únicas propias de esta plataforma como: Enterprise Javabean, servlets o Java Server Faces. Todo ello permite a los desarrolladores crear aplicaciones empresariales portables entre plataformas, escalable, seguras e integrables con tecnologías anteriores. La especificación original de J2EE fue desarrollada por "Sun Microsystems" y liberada públicamente en Diciembre del año 2002. La versión estable actual es java EE 7, cuya especificación es la "JSR 342", publicada en mayo de 2013.
- **Glassfish 4.1.1**: Glassfish es un servidor de aplicaciones de software libre desarrollado por "Sun Microsystems", posteriormente adquirida por "Oracle". Este servidor de aplicaciones implementa todas las funcionalidades de la especificación Java EE 7. Es gratuito y de código abierto, distribuyéndose bajo una licencia "GNU GPL". Existe una versión comercial denominada "Oracle Glassfish Enterprise Server". La primera versión se publicó en Junio de 2005. Desde la versión 4.0 publicada en Junio de 20013 es compatible con java EE 7.
- **MongoDB**: MongoDB es un sistema gestor de bases de datos "NoSQL" orientado a documento. Esta desarrollado bajo el concepto de código abierto. Esta base de datos guarda estructuras de documentos similares a JSON con un esquema dinámico. Internamente utiliza

[6]Swagger: http://swagger.io/

BSON, el formato BSON es una extensión del formato JSON, permitiendo almacenar valores en formato "Float", "Date", "Expresiones regulares" y código Java Script", así mismo, se introducen modificaciones en el formato JSON para mejorar el rendimiento. El desarrollo de MongoDB comenzó en el año 2007 por la compañía "10gen". Fue en 2009 cuando MongoDB fue lanzado como un producto independiente y bajo una licencia de código abierto. Algunas de sus características principales son: Consultas Ad-hoc (búsqueda por campos, rangos y expresiones regulares), indexación (Cualquier documento puede ser indexado), replicación (Soporta el tipo de replicación primario-secundario), balanceo de carga (se puede escalar de forma horizontal usando el concepto "shard" los datos se dividen en múltiples servidores), almacenamiento de archivos (puede ser utilizado como un sistema de archivos, teniendo la ventaja del balanceo de carga y replicación), agregación (proporciona un framework de agregación que permite realizar operaciones similares al comando SQL "Group By") y ejecución de JavaScript en el lado del servidor. Algunos de los problemas que implica trabajar con MongoDB son: no implementa las propiedades ACID (el no implementar las propiedades "ACID" genera que la base de datos no asegure la durabilidad, integridad, consistencia y aislamiento requerido en las transacciones), problemas de consistencia (no se garantiza la consistencia de los documentos, lo que puede implicar leer versiones obsoletas de los datos), bloqueo a nivel de documento (la base de datos bloquea a nivel de documento ante cada operación de escritura. Sólo se podrán hacer operaciones de escritura concurrentes entre distintos documentos), problemas de escalabilidad (puede tener problemas de rendimiento cuando el tamaño de una colección supera los 100GB)

- **Mongo Java Driver 3.2:** Para la conexión con la base de datos, se utiliza el driver oficial de MongoDB para Java, en su versión 3.2.
- **Visualización**: Para el desarrollo de la capa de presentación se han utilizado las tecnologías "Java Server Faces", hojas de estilo "CSS" y scripts "Java Script".
- **JSON:** JSON (JavaScript Object Notation) es un formato de texto ligero para el intercambio de datos, debido a su amplia adopción se considera como alternativa a "XML". Se utiliza el formato "JSON" para la recepción y envío de información a través de los servicios Rest.
- **Maven:** Maven es una herramienta para la gestión de proyectos Java creada por "Jason van Zyl" en 2002. Su funcionamiento es similar a "Apache Ant". Utiliza un modelo de configuración más simple, basado en "XML". Maven utiliza un POM (project Object Model) para describir el proyecto software a construir, sus dependencias de otros módulos y componentes externos y el orden de construcción de los elementos.
- **PrimeFaces:** Es una librería de componentes para "JSF" de código abierto. Tiene soporte de Ajax, permite desarrollar aplicaciones web para dispositivos móviles
- **Netbeans:** NetBeans es un entorno de desarrollo integrado libre, desarrollado principalmente para Java. Es un proyecto de código abierto. El proyecto Netbeans fue fundado por "Sun Microsystems" en el año 2000. Es multiplataforma. Se ha utilizado la versión 8.1.
- **GitHub:** GitHub es una plataforma de desarrollo colaborativo, que permite alojar proyectos utilizando el sistema de control de versiones "Git". Esta plataforma permite: crear una wiki para cada proyecto, páginas web del proyecto, generar gráficos para ver como los desarrolladores trabajan en sus repositorios, seguir y observar otros desarrolladores y proyecto. Fue lanzado en febrero de 2008.

- **JMeter:** JMeter es un proyecto de "Apache" que puede ser utilizado como una herramienta de pruebas de carga para analizar y medir el desempeño de una variedad de servicios, especialmente aplicaciones web. Puede ser usado como una herramienta de pruebas unitarias para conexiones con bases de datos. Es considerado como una herramienta de "generación de carga". Su primera versión fue lanzada en el año 2001 y actualmente se encuentra en su versión 3.0 lanzada en Mayo del año 2016.
- **Log4j:** Log4j es una biblioteca open source desarrollada en java por "Apache Software foundation" que permite a los desarrolladores de software escribir mensajes de registro, cuyo propósito es dejar constancia de una determinada transacción en tiempo de ejecución. Log4j permite filtrar mensajes en función de su importancia.
- **Twitter4j:** Es una librería Java, que permite la integración con la red social "Twitter'. Para su funcionamiento no necesita de otras librerías, soporta todas las funcionalidades disponibles en la API 1.1 de "Twitter".
- **TeXstudio y MiKTex:** Editor gráfico y motor para la elaboración de documentos "Latex".

Capítulo 7

Pruebas

En esta etapa del documento se reflejará las pruebas realizadas durante el desarrollo del proyecto. para ello se mostrarán los resultados en forma de tablas donde se describen de forma cuantitativa y cualitativa el funcionamiento de la aplicación. Existen dos tipos de pruebas:

- **Pruebas de caja blanca:** Se realizan sobre las funciones internas de un módulo o clase.
- **Pruebas de caja negra:** Se comprueban que los requisitos funcionales se han respetado y cumplido. Es decir, permite obtener conjuntos de condiciones de entrada se ejecuten correctamente en todos los requisitos funcionales de un programa.

7.1. Resultados

En este capítulo se incluyen las pruebas realizadas durante y después del desarrollo de la aplicacion. A continuación se pueden ver las pruebas que se han llevado a cabo.

El diseño de las pruebas de caja negra se presentan a continuación. Estas pruebas se diseñan en la fase de análisis, pero se muestran los resultados en este punto. Las pruebas de caja negra se basan en validar las respuestas que el sistema produce al introducir una determinadas entrada.

PCN-01	**Identificar un usuario**
Propósito	Autenticar un usuario, comprobando los credenciales introducidos por el usuario con los datos almacenados en la base de datos.
Prerrequisito	El usuario debe estar registrado en el sistema
Datos de entrada	Datos del formulario de "Login" correctos
Pasos	1. El usuario introduce sus datos en el formulario
Resultado esperado	El usuario se identifica en el sistema y se crea una sesión de usuario.
Resultado obtenido	El usuario accede a la aplicación y se crea la sesión de usuario.
Resultado de la prueba	Correcto

Figura 7.1: Prueba de Caja Negra 01 : Identificar un Usuario

PCN-02	Cerrar sesión
Propósito	Cerrar la sesión de un usuario autenticado en el sistema.
Prerrequisito	El usuario debe estar autenticado en el sistema
Datos de entrada	Ninguno
Pasos	1. El usuario pulsa el botón de salir y confirma la acción.
Resultado esperado	El usuario sale del sistema y se elimina la sesión.
Resultado obtenido	El usuario abandona la aplicación y se elimina la sesión de usuario.
Resultado de la prueba	Correcto

Figura 7.2: Prueba de Caja Negra 02 : Cerrar Sesión

PCN-03	Crear nuevo evento
Propósito	Añadir un nuevo evento al sistema
Prerrequisito	El usuario debe estar autenticado en el sistema
Datos de entrada	Datos del formulario "Nuevo Evento" validos.
Pasos	1. El usuario rellena el formulario y pulsa guardar.
Resultado esperado	El nuevo evento se añade a la base de datos, se añade un nuevo documento en la colección de seguimientos y se inicializa una colección en la base de datos de parse4uData.
Resultado obtenido	Se crea un nuevo evento con los datos de entrada.
Resultado de la prueba	correcto

Figura 7.3: Prueba de Caja Negra 03 : Crear Nuevo Evento

PCN-04	Ver Eventos
Propósito	Visualizar un listado de los eventos del usuario
Prerrequisito	El usuario debe estar autenticado en el sistema
Datos de entrada	Ninguno
Pasos	1. El usuario pulsa en el menú la opción "Ver Eventos"
Resultado esperado	El sistema muestra una tabla con los datos de los eventos que el usuario ha creado.
Resultado obtenido	Se muestra un listado con los eventos que el usuario creo anteriormente.
Resultado de la prueba	Correcto

Figura 7.4: Prueba de Caja negra 04 : Ver Eventos

PCN-05	Modificar Evento
Propósito	Modificar los datos de un evento existente.
Prerrequisito	El usuario debe estar autenticado en el sistema y haber añadido al menos un evento.
Datos de entrada	Datos modificados del evento correctos
Pasos	1. El usuario pulsa en el menú la opción “Editar” de un evento. 2. El sistema muestra un formulario con los datos actuales del evento. 3. El usuario modifica los datos deseados y pulsa “Guardar”
Resultado esperado	El sistema actualiza los datos modificados del evento.
Resultado obtenido	Los datos modificado se han actualizado en la base de datos
Resultado de la prueba	Correcto

Figura 7.5: Prueba de Caja Negra 05 : Modificar Evento

PCN-06	Eliminar Evento
Propósito	Eliminar un evento existente
Prerrequisito	El usuario debe estar autenticado en el sistema y haber añadido al menos un evento.
Datos de entrada	Ninguno
Pasos	1. El usuario pulsa en el menú la opción “Borrar” de un evento.
Resultado esperado	El sistema marca el evento como borrado.
Resultado obtenido	El evento se ha modificado, estableciendo el estado como borrado
Resultado de la prueba	Correcto

Figura 7.6: Prueba de Caja Negra 06 : Eliminar Evento

PCN-07	Iniciar Captura
Propósito	Iniciar la captura de datos de un evento
Prerrequisito	El usuario debe estar autenticado en el sistema y haber añadido al menos un evento, una cuenta de Twitter y una Api Key.
Datos de entrada	Nombre de Api Key
Pasos	1. El usuario pulsa en el menú la opción "Capturar" de un evento. 2. El sistema muestra los datos del evento y un listado de las Api Key del usuario. 3. El usuario pulsa iniciar
Resultado esperado	El sistema crea un nuevo hilo de ejecución. El hilo establece una conexión con la Api de Twitter. Se almacena la referencia del hilo en el Bean Application
Resultado obtenido	Se inicia un nuevo hilo, se establece una conexión con la Api de Twitter, la referencia del hilo se almacena en el appBean
Resultado de la prueba	Correcto

Figura 7.7: Prueba de Caja Negra 07 : Iniciar Captura

PCN-08	Cancelar captura
Propósito	Cancelar una captura.
Prerrequisito	El usuario debe estar autenticado en el sistema y haber iniciado la captura de un evento.
Datos de entrada	Ninguno
Pasos	1. El usuario pulsa "Cancelar captura" de un evento. 2. El sistema localiza el hilo de ejecución en el Bean Application 3. Accede al hilo de ejecución y cierra la conexión con la Api de Twitter 4. El sistema inicia una parada segura del hilo 5. El sistema elimina la referencia al hilo de ejecución del Bean Application.
Resultado esperado	El sistema cierra la conexión con Twitter. Se libera los recursos del hilo. Se elimina el hilo de ejecución.
Resultado obtenido	El hilo de ejecución se elimina, pero la conexión con Twitter sigue activa.
Resultado de la prueba	Incorrecto

Figura 7.8: Prueba de Caja Negra 08 : Cancelar Captura

PCN-09	Almacenar un Tweet
Propósito	Escribir un Tweets recibido en la base de datos
Prerrequisito	El usuario debe estar autenticado en el sistema y haber iniciado la captura de un evento.
Datos de entrada	Tweet recibido.
Pasos	1. El sistema recibe un tweet a través de la API de Streaming. 2. El sistema lee los datos del hilo y almacena el tweet en la colección correspondiente.
Resultado esperado	El sistema convierte el tweet a un objeto Document. Inserta el documento en la colección correspondiente al evento por el cual se ha recibido.
Resultado obtenido	El tweet se añade a la colección correspondiente.
Resultado de la prueba	Correcto

Figura 7.9: Prueba de Caja Negra 09 : Almacenar un "Tweet"

PCN-10	Identificar un usuario
Propósito	Autenticar un usuario, comprobando los credenciales introducidos por el usuario con los datos almacenados en la base de datos.
Prerrequisito	El usuario debe estar registrado en el sistema
Datos de entrada	Datos del formulario de "Login" incorrectos
Pasos	1. El usuario introduce sus datos en el formulario
Resultado esperado	El sistema muestra un mensaje de error al usuario y no permite el acceso a la aplicación.
Resultado obtenido	Se recibe un mensaje de error y no se puede acceder las páginas de la herramienta.
Resultado de la prueba	Correcto

Figura 7.10: Prueba de Caja Negra 10: Identificar un Usuario con Credenciales Incorrectos

PCN-11	Crear nuevo evento
Propósito	Añadir un nuevo evento al sistema
Prerrequisito	El usuario debe estar autenticado en el sistema
Datos de entrada	Datos del formulario "Nuevo Evento", utilizando un nombre de evento que ya exista.
Pasos	1. El usuario pulsa el botón de salir y confirma la acción.
Resultado esperado	El sistema muestra un mensaje de error indicando al usuario que ese evento ya existe; y no se añade a la base de datos
Resultado obtenido	El evento se ha añadido a la base de datos, obteniendo varios eventos con el mismo nombre.
Resultado de la prueba	Incorrecto

Figura 7.11: Prueba de Caja Negra 11: Crear un Nuevo Evento Usando un Nombre de Evento Existente

PCN-12	Modificar Evento
Propósito	Modificar los datos de un evento existente.
Prerrequisito	El usuario debe estar autenticado en el sistema y haber añadido al menos un evento.
Datos de entrada	Datos modificados del evento incorrectos
Pasos	1. El usuario pulsa en el menú la opción “Editar” de un evento. 2. El sistema muestra un formulario con los datos actuales del evento. 3. El usuario modifica los datos deseados y pulsa “Guardar”
Resultado esperado	El sistema muestra un error al usuario y no modifica los datos.
Resultado obtenido	Se actualizan los datos del evento sin avisar al usuario del error
Resultado de la prueba	Incorrecto

Figura 7.12: Prueba de Caja Negra 12: Modificar datos de un Evento con Datos Incorrectos

7.2. Análisis de los Resultados

Debido a la similitud en las pruebas de caja negra entre las acciones de los eventos, cuentas de Twitter y Api Keys se han mostrado solo las pruebas de caja negra de la operaciones de los eventos; se aplicarían de igual manera estas pruebas a las cuentas de “Twitter” y Api Key.

Todos los resultados se han ido obteniendo a través de las diferentes pruebas que se han hecho en el sistema, que garantizan un correcto funcionamiento de la herramienta. Si bien, puede haber algunas funcionalidades o servicios Rest que se crean convenientes, se pueden incluir en futuras versiones del proyecto si fuera necesario.

Cabe a destacar que esta herramienta se ha probado utilizando cargas de trabajo sintéticas, por lo que, aunque se estima correcto, no se conoce a ciencia cierta el desempeño real que tendría en un entorno real.

Capítulo 8

Manuales del Sistema

En este capítulo, se desarrollan los manuales necesarios tanto para instalar el sistema como para utilizarlo.

8.1. Manual de Instalación

Antes de poder desplegar la aplicación y poder utilizarla, es necesario instalar múltiples servicios y programas para el funcionamiento de la plataforma al completo.

En este caso se va a mostrar la instalación en un Sistema Operativo Windows 10, pero la instalación en otros sistemas operativos tipo Gnu/Linux se lleva a cabo de una manera similar.

8.1.1. Servidor de Bases de Datos

Primero se debe realizar la instalación del servidor de Bases de Datos MongoDB. Su descarga está disponible en: MongoDB Server [1], en este caso se utiliza la versión para "Windows Server 2008 R2 64-bits and later, with SSL support", en su versión "Comumunity Server (Gratuita)". Una vez descargado el fichero con extensión ".msi", ejecutamos el fichero y configuramos la ruta donde se desea realizar la instalación. Una vez completada la instalación, debemos crear un directorio llamado "data" y dentro de este otro directorio llamado "db", donde se almacenan las bases de datos y otro directorio llamado "log" donde se almacenarán los logs del sistema gestor de bases de datos.

[1] Mongo DB Server: https://www.mongodb.com/download-center#community

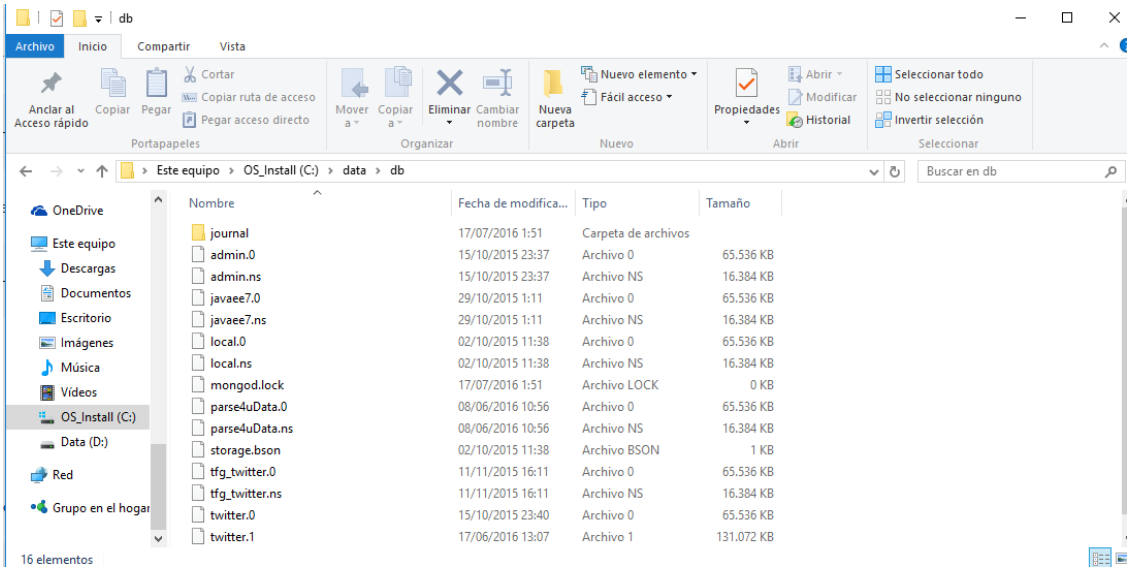


Figura 8.1: Proceso de Instalación MongoDB, creación de directorios

Una vez creados los directorios, podemos crear un fichero de configuración llamado "'mongod.cfg' donde se define la ruta de almacenamiento de los datos y de los logs, de la siguiente forma:

```
systemLog:
destination: file
path: c:\data\log\mongod.log
storage:
dbPath: c:\data\db
```

Y creamos un nuevo servicio de Windows, para garantizar que siempre esté en funcionamiento el servidor de bases de datos. Para ello, ejecutamos la instrucción:

```
sc.exe create MongoDB binPath= "C:\mongodb\bin\mongod.exe
--service --config=\"C:\mongodb\mongod.cfg\"" DisplayName= "MongoDB" start= "auto"
```

Donde definimos:"binPath": directorio del ejecutable de MongoDb, "–service:" para indicar que es un servicio, "–config:" ruta del fichero de configuración mongod.cfg, "DysplayName:" Nombre del servicio y "start=auto" para indicar que el servicio se inicie automáticamente.

Una vez creado el servicio comprobamos el correcto funcionamiento accediendo al directorio de instalación y ejecutando el fichero "mongod.exe", este nos mostrará la consola de administración del sistema gestor.

Para llevar a cabo esta instalación, se han seguido los pasos del manual de instalación oficial, puede encontrarse en: Install Mongo on Windows [2]. Si deseamos configurar la seguridad de la base de datos añadiendo usuarios o roles de usuarios, podemos encontrar toda la información necesaria en la documentación oficial: MongoDB Security [3]

[2]Install Mongo on Windows: https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/
[3]MongoDB Security: https://docs.mongodb.com/manual/administration/security-checklist/

8.1.2. Servidor de Aplicaciones

En esta sección mostraremos los pasos necesarios para instalar y configurar el Servidor de Aplicaciones Glassfish. Como requisito, para poder ejecutar el servidor de aplicaciones, es necesario tener instalado el "JDK" de Java. Este puede descargarse desde: Java JDK. En este caso seleccionaremos la versión Windows x64, dado que para poder configurar correctamente el servidor de aplicaciones y poder asignar la cantidad de memoria ram necesaria, es necesaria una plataforma de "64bits".

Una vez descargado el instalador del "JDK" de Java, ejecutamos el fichero ".exe" y lo instalamos en la ruta deseada. Con esto, ya tendríamos los pre requisitos para instalar el Servidor de Aplicaciones. Procedemos a descargar Glassfish 4.1.1 Full Platform, esta versión implementa toda la especificación de Java EE 7. Para su descarga accedemos a: GlassFish Server Download [4] y descargamos la versión Full Platform.

Una vez completada la descarga, extraemos el fichero con extensión ".zip" en la ruta donde deseemos almacenar el servidor de aplicaciones. Y accedemos al directorio "D:\GlassFish\glassfish\bin", en nuestro caso; ejecutamos el comando "asadmin.bat start-domain", esto nos iniciara el dominio por defecto del servidor de aplicaciones.

Para acceder a la administrador de servidor de aplicaciones, accedemos a la URL http://localhost:4848/.

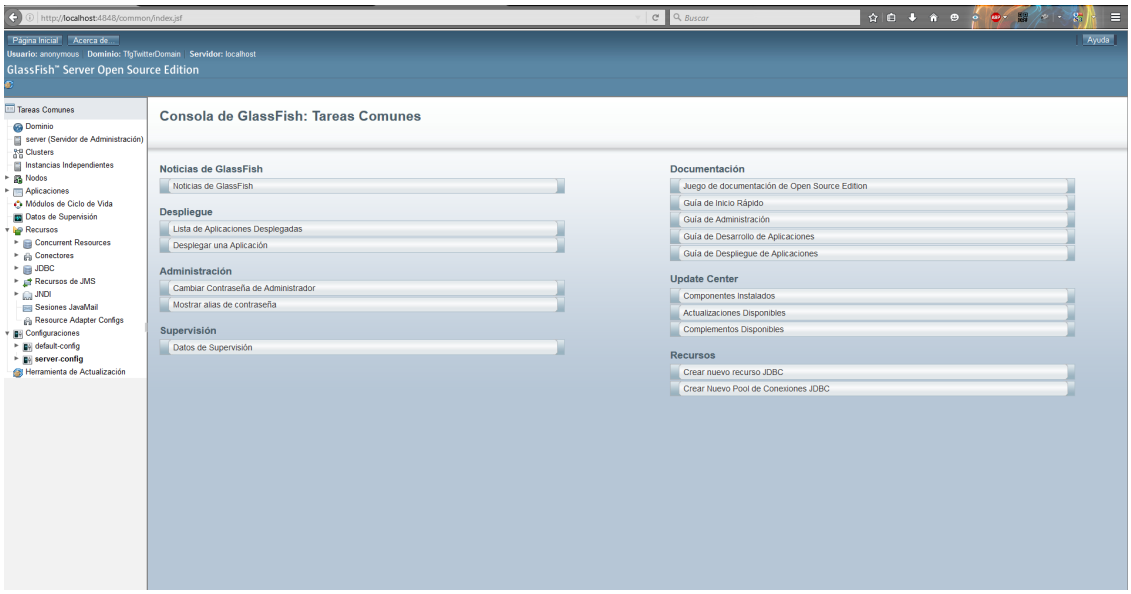


Figura 8.2: Instalación de Servidor de Aplicaciones Glassfish, consola de administración

Al final de este capítulo se mostrarán las modificaciones realizadas en el servidor de aplicaciones para optimizar su rendimiento y mejorar sus prestaciones.

[4]Glassfish Server Download: https://glassfish.java.net/download.html

8.1.3. Configuración JAAS

Una vez instalado tanto el servidor de bases de datos como el servidor de aplicaciones, debemos configurar e dominio de Glassfish para permitir la autenticación y autorización de usuarios utilizando la API JAAS de Java. Para el correcto funcionamiento del Realm, es necesario añadir unas librerías al servidor de aplicaciones, para que este pueda conectarse con la base de datos.

- Descargamos la librería "Apache Commons Codec" desde: Apache Commons Codec [5].
- Descargamos el Realm para MongoDB desde: Realm MongoDB [6]
- Descargamos la última versión del driver de MongoDB para Java. Puede encontrarse en: MongoDB Java Driver [7]

Una vez descargados los tres ficheros, debemos copiarlos a la ruta del dominio de Glassfish: "D:\GlassFish \glassfish \domains\TfgTwitterDomain\lib\" en nuestro caso. Una vez copiadas las librerías debemos reiniciar el servidor de aplicaciones para garantizar que estas se carguen en el sistema, para ello ejecutamos la instrucción "asadmin.bat restart-domain".

Una vez añadidas y cargadas las librerías podemos crear un nuevo Realm. Para ello accedemos a la administración del servidor de aplicaciones mediante la URL: http://localhost:4848/. En el menú de la izquierda accedemos a la sección "Configurations / server-config / security / Realms".

---

[5] Apache Commons Codec: http://commons.apache.org/proper/commons-codec/download_codec.cgi

[6] Realm MongoDB: https://oss.sonatype.org/service/local/repositories/releases/content/com/vaushell/glassfish-mongo-realm/1.0.0/glassfish-mongo-realm-1.0.0.jar

[7] MongoDB Java Driver: http://mongodb.github.io/mongo-java-driver/

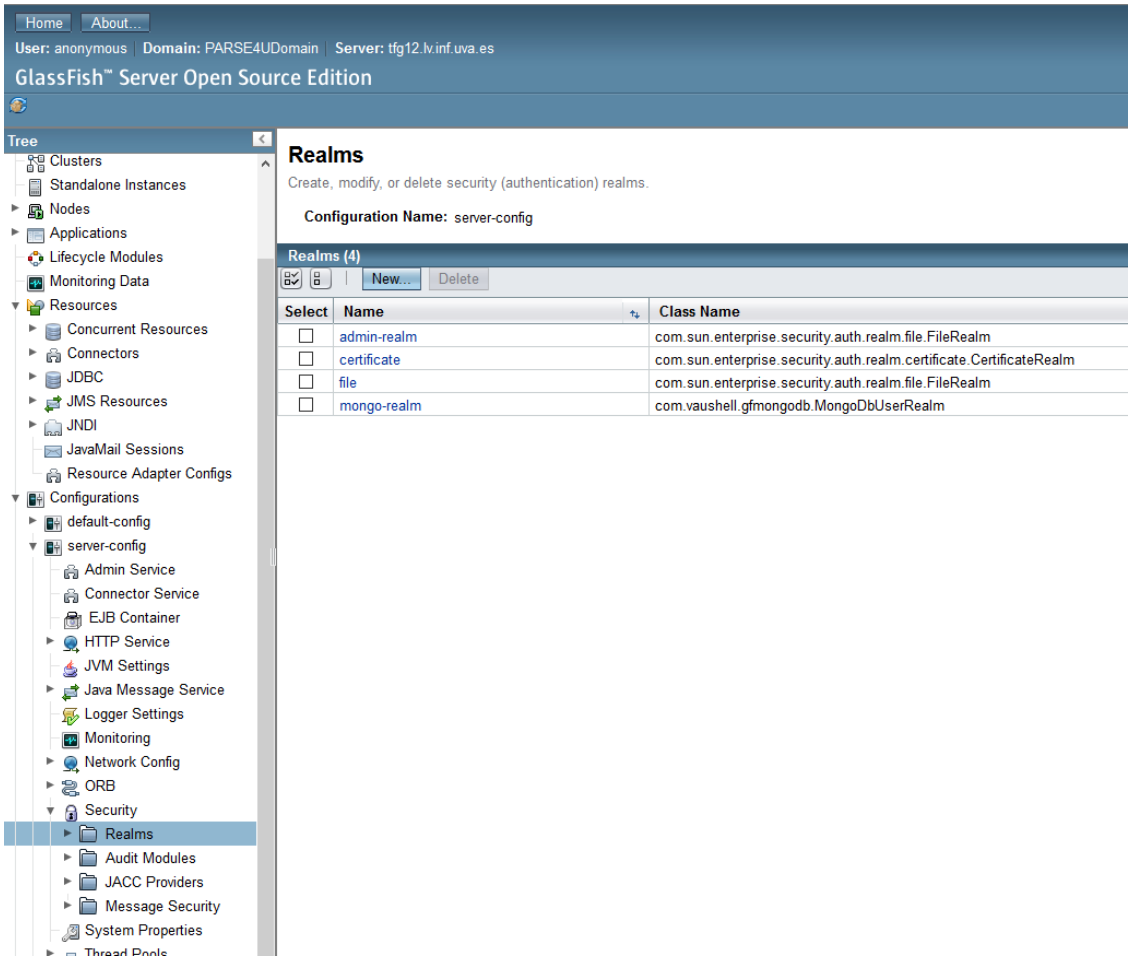


Figura 8.3: Configuración Real JAAS sobre MongoDB

Y procedemos a añadir un nuevo "Realm" con estos datos:

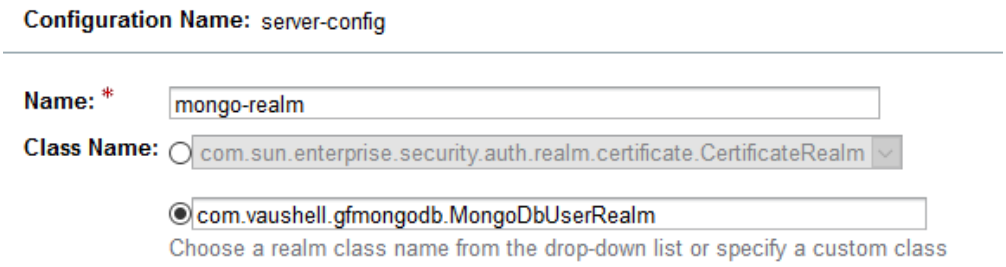


Figura 8.4: Creación Realm sobre MongoDB

Y añadimos las siguientes propiedades, que indicaran al Realm los nombres de la base de datos, colecciones y campos utilizados para autenticar usuarios.

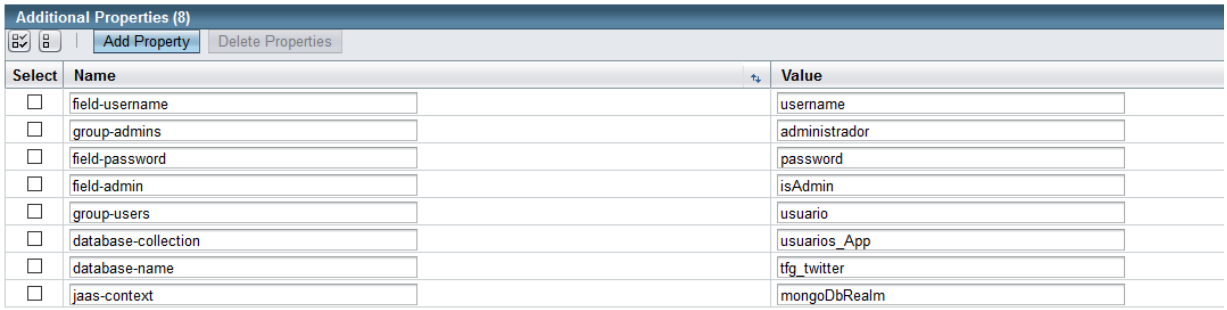


Figura 8.5: Propiedades del Realm sobre MongoDB

Al pulsar el botón de "Save" se añadirá un nuevo Realm listo para ser utilizado por la herramienta. Se recomienda no modificar los valores proporcionados para garantizar su correcto funcionamiento.

Toda esta información está basada en el trabajo de "fabienvauchelles", su proyecto puede encontrarse: GitHub [8].

8.1.4. Despliegue de la Aplicación

Por último, debemos desplegar la aplicación en el servidor de aplicaciones, para que esta esté disponible. Para ello, accedemos a la URL http://localhost:4848/ y seleccionamos en el menú de la izquierda la sección "Aplicaciones". Aquí podremos desplegar nuevas aplicaciones o ver las aplicaciones actualmente desplegadas.

Para desplegar una nueva aplicación, pulsamos el botón "Desplegar" en el formulario elegimos la opción "archivo empaquetado que Cargar al Servidor" en caso de que este sea un servidor remoto, o "Archivo empaquetado local o directorio accesible desde GlassFish Server" en caso de estar trabajando en un servidor local. En cualquiera de los casos, necesitamos proporcionar al servidor un fichero empaquetado con extensión ".war". Este se encuentra en el CD adjunto con la documentación, dentro de la carpeta "target" del proyecto.

[8]Glassfish-Mongo-Realm : https://github.com/fabienvauchelles/glassfish-mongo-realm

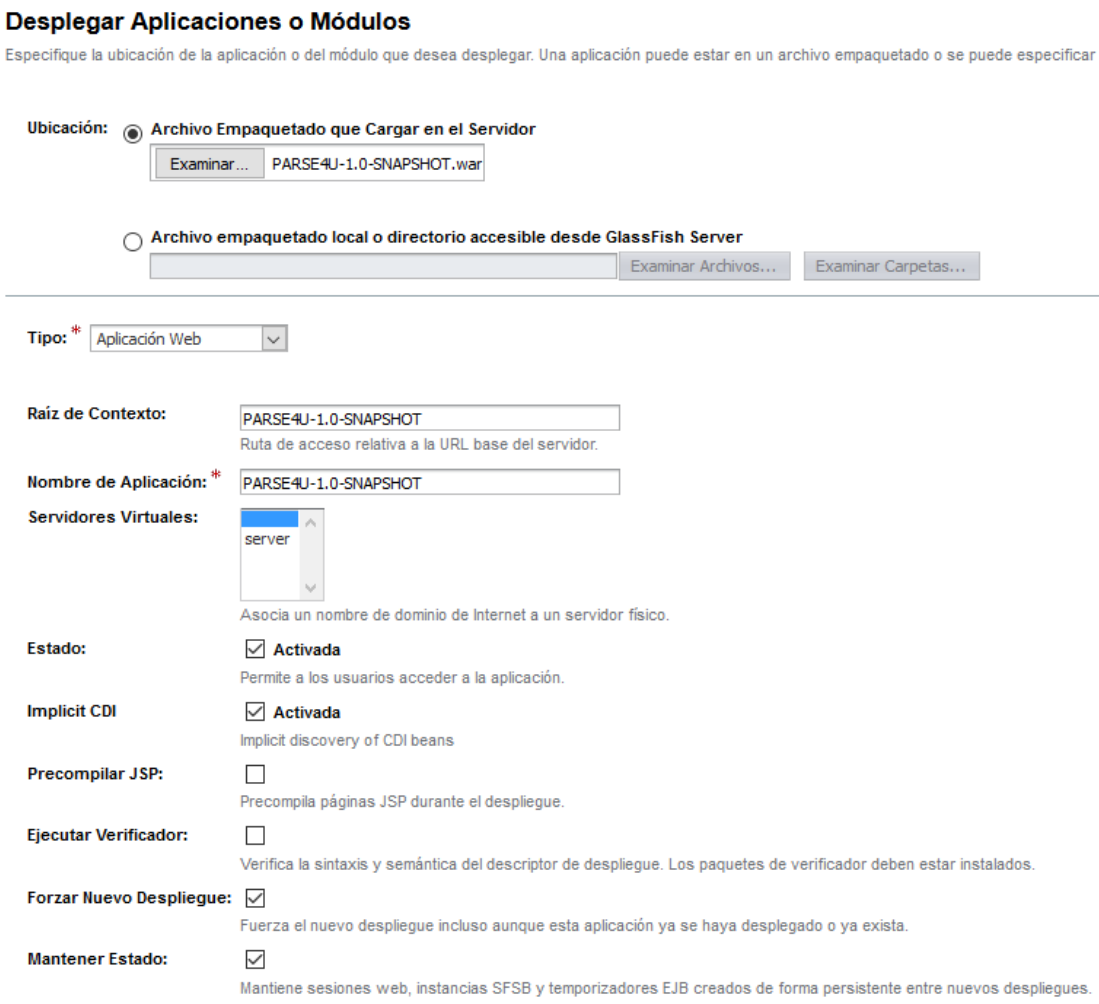


Figura 8.6: Despliegue de la Aplicación

Una vez pulsemos en "Aceptar" el fichero empaquetado se subirá al servidor de aplicaciones y se realizará el despliegue de la aplicación. Cuando este proceso termine, podremos acceder a la aplicación mediante la URL: http://localhost:8080/PARSE4U/

8.1.5. Optimización del Servidor GlassFish

Como se indicaba en la sección de instalación del servidor Glassfish, se han realizado unas modificaciones para optimizar el funcionamiento del servidor de aplicaciones. Para llevar a cabo esta Optimización se ha tomado como base la documentación oficial, puede encontrarse en: GlassFish Server Open Source Edition Performance Tuning Guide [9] A continuación, se describen cada una de las modificaciones que se han realizado:

- **-server**: al modificar la opción "-client" por "-server" indicamos al servidor de aplicaciones que actúe como un servidor y no como cliente. Esta opción de necesaria para un servidor de aplicaciones en producción.
- **-XX:ParallelGCThreads=2 :** esta opción permite al servidor de aplicaciones tener dos recolectores de basura de manera concurrente, una por cada procesador del servidor.

[9]GlassFish Server Open Source Edition Performance Tuning Guide: https://glassfish.java.net/docs/4.0/performance-tuning-guide.pdf

- **-Xmx3584m y -Xms1024m**: indican el máximo y mínimo de memoria ram que utilizara el servidor de aplicaciones, en este caso, el servidor donde se ejecuta tiene 6 Gb de memoria ram, por lo que se le asigna un mínimo de 1Gb de ram y un máximo de 3,5Gb de ram, dejando suficiente memoria disponible para otros servicios y el propio sistema operativo.
- **-XX:MaxPermSize=192m :** esta opción indica al servidor de aplicaciones que desde los primero 1024MB de ram hasta los 3,5 GB de ram, ira ampliando la memoria en uso en bloques de 192 MB.
- **com.sun.grizzly.useKeepAliveAlgorithm=true:** esta opción permite reducir el tiempo de bloqueo de las entradas/salidas.

Estas opciones se modifican en el apartado: "Configuración - serverConfig - configuración de JVM - opciones de JVM"

Select	Value
☐	-Dorg.glassfish.additionalOSGiBundlesToStart=org.apache.felix.shell,org.apache.felix.gogo.run
☐	-Djavax.management.builder.initial=com.sun.enterprise.v3.admin.AppServerMBeanServerBuild
☐	-Djavax.net.ssl.keyStore=${com.sun.aas.instanceRoot}/config/keystore.jks
☐	-server
☐	-DANTLR_USE_DIRECT_CLASS_LOADING=true
☐	-Dcom.ctc.wstx.returnNullForDefaultNamespace=true
☐	-XX:ParallelGCThreads=2
☐	-Dfelix.fileinstall.bundles.startTransient=true
☐	-Djavax.net.ssl.trustStore=${com.sun.aas.instanceRoot}/config/cacerts.jks
☐	-Dosgi.shell.telnet.ip=127.0.0.1
☐	-Dfelix.fileinstall.log.level=2
☐	-XX:+UnlockDiagnosticVMOptions
☐	-Djava.security.auth.login.config=${com.sun.aas.instanceRoot}/config/login.conf
☐	-Xss128k
☐	-Dfelix.fileinstall.disableConfigSave=false
☐	-Djava.awt.headless=true
☐	-Djdbc.drivers=org.apache.derby.jdbc.ClientDriver
☐	-Djdk.corba.allowOutputStreamSubclass=true
☐	-Dosgi.shell.telnet.port=6666
☐	-Dosgi.shell.telnet.maxconn=1
☐	-Djavax.xml.accessExternalSchema=all
☐	-Xms1024m
☐	-Djava.ext.dirs=${com.sun.aas.javaRoot}/lib/ext${path.separator}${com.sun.aas.javaRoot}/jre/lib
☐	-Djava.security.policy=${com.sun.aas.instanceRoot}/config/server.policy
☐	-Dgosh.args=--nointeractive
☐	-Dcom.sun.enterprise.config.config_environment_factory_class=com.sun.enterprise.config.serv
☐	-XX:MaxPermSize=192m
☐	-Djava.endorsed.dirs=${com.sun.aas.installRoot}/modules/endorsed${path.separator}${com.s
☐	-Dcom.sun.enterprise.security.httpsOutboundKeyAlias=s1as
☐	-Dfelix.fileinstall.poll=5000
☐	-XX:+DisableExplicitGC
☐	-Dfelix.fileinstall.bundles.new.start=true
☐	-Dcom.sun.grizzly.useKeepAliveAlgorithm=true
☐	-XX:NewRatio=2
☐	-Xmx3584m
☐	-Dfelix.fileinstall.dir=${com.sun.aas.installRoot}/modules/autostart/

Figura 8.7: Configuración para la Optimización del Servidor Glassfish

Estas son solo algunas de las recomendaciones para configurar un servidor de aplicaciones Glassfish para producción, también se deberían modificar los parámetros de los logs para generar únicamente los registros de los módulos que sean necesarios y fijar un nivel de log superior.

8.2. Manual de Usuario

Al igual que todo software, se necesita de un manual de instrucciones que permita a los usuarios recurrir a este en caso de dudas sobre el uso y funcionamiento de la aplicación.

La herramienta es accesible desde un navegador web, por lo que, los usuarios finales de la herramienta, no necesitas instalar ningún software. la complejidad de la instalación recae en el lado del servidor, como se ha podido ver en la sección 8.1.

8.2.1. Inicio de Aplicación

La primera pantalla que ve el usuario cuando accede a la aplicación web es la pantalla de bienvenida, donde se muestran las características de la herramienta y sus funcionalidades.

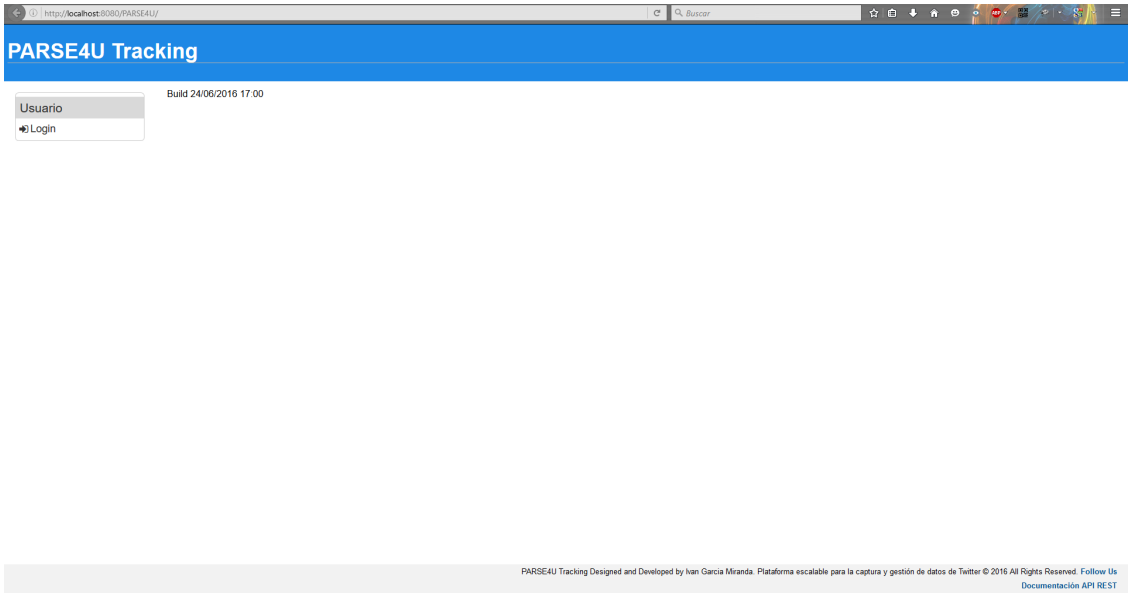


Figura 8.8: Página de "Bienvenida" PARSE4U

La herramienta ha sido diseñada para que solo pueda ser utilizada por usuarios registrados, para ello, un administrador deberá haber añadido el usuario al sistema. El usuario verá en el menú de la izquierda, el acceso a la página de "login". En la que se mostrará el formulario para que el usuario introduzca su usuario y contraseña.

8.2.2. Acceso a la Aplicación

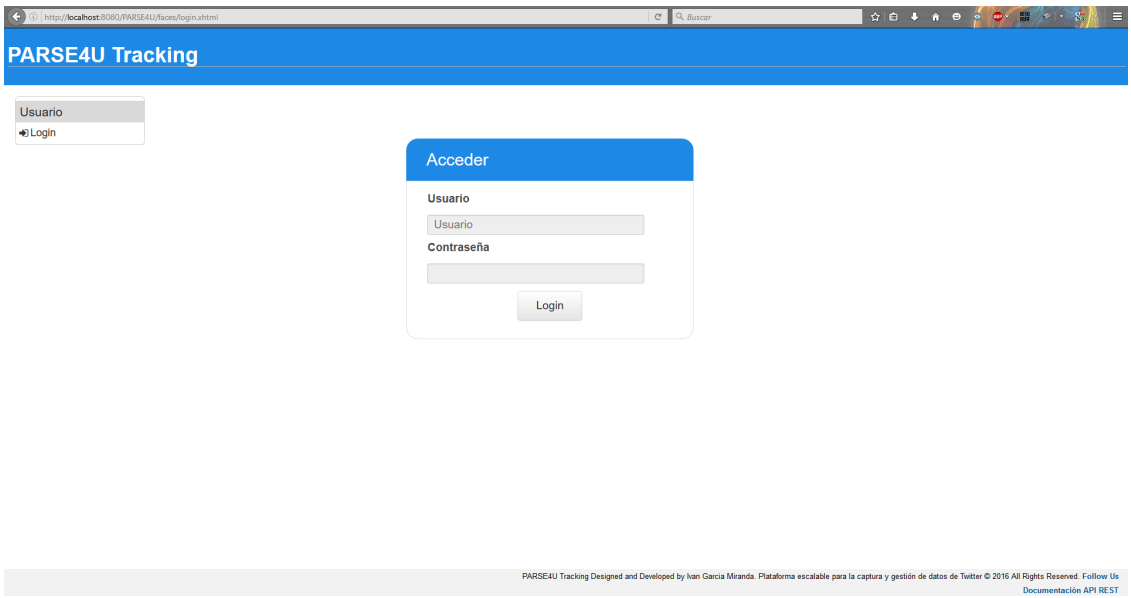


Figura 8.9: Página de "login" PARSE4U

Una vez que el usuario se ha autenticado en el sistema utilizando sus credenciales, se mostrará la página principal de los usuarios registrados. aquí podrá ver un resumen de los datos que su usuario ha capturado.

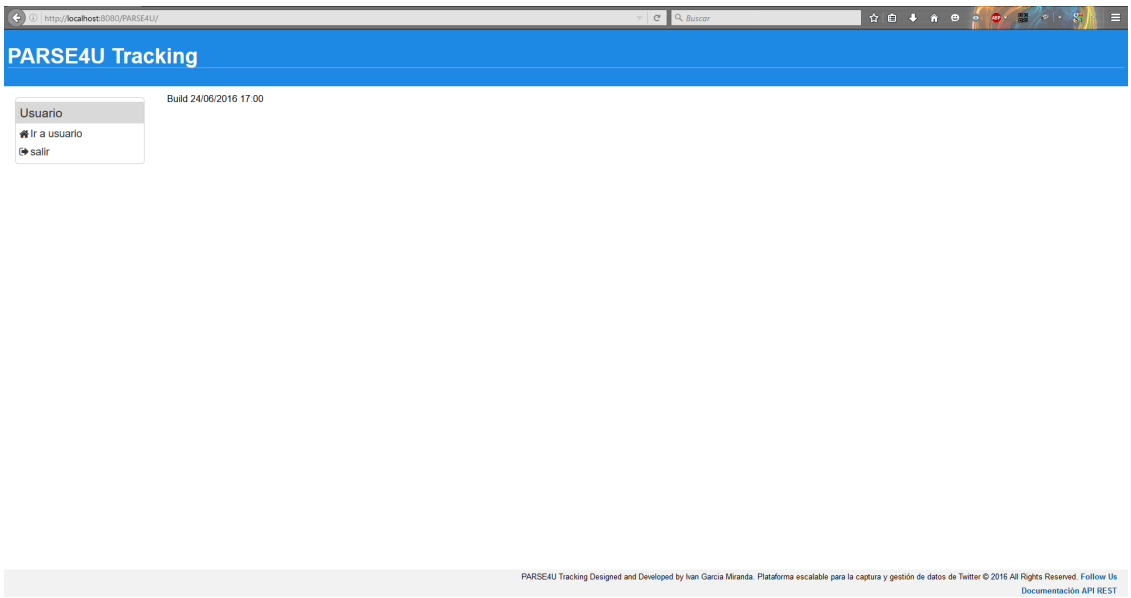


Figura 8.10: Página de Inicio para Usuarios Registrados PARSE4U

Una vez que el usuario ya se ha identificado, podrá acceder a los contenidos de la herramienta. Para ello, a la izquierda se mostrará el menú para navegar entre las diferentes opciones. También dispone de una opción para cerrar la sesión y salir del sistema.

Figura 8.11: Detalle del menú de navegación PARSE4U

El principal objetivo de la herramienta es permitir a los usuarios realizar capturas de eventos en base a ciertas palabras clave. Para ello, los usuarios deben poder añadir nuevos eventos al sistema, para capturar tweets relacionados con el evento.

8.2.3. Añadir Nuevo Evento

Para añadir un nuevo evento, el usuario, deberá pulsar la opción "Nuevo Evento" en le menú izquierdo. A continuación, se mostrará un formulario con los datos necesarios que se deben cumplimentar para añadir un nuevo evento para su posterior captura de datos.

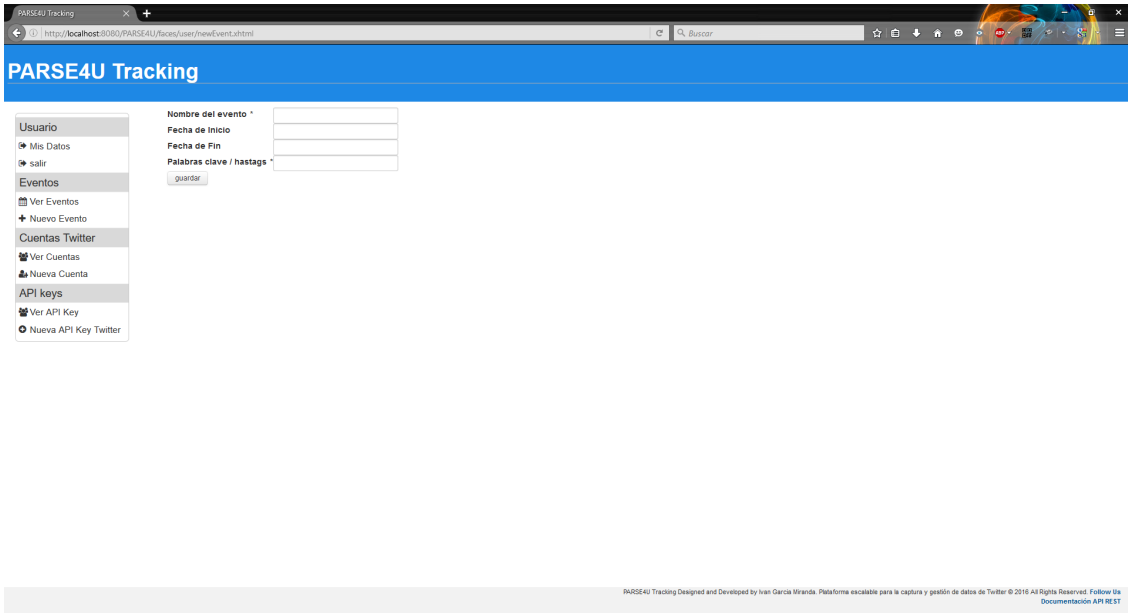


Figura 8.12: Página de "Nuevo Evento" PARSE4U

Cuando se rellenen los datos del formulario y pulse el botón de, el sistema añadirá un nuevo evento a la lista de eventos del usuario.

8.2.4. Ver Eventos

Si el usuario desea ver los eventos que ha añadido en el sistema, sólo debe pulsar en "Ver Eventos" del menú de navegación anterior. Se le mostrará una tabla con los datos de los eventos que el usuario ha añadido al sistema.

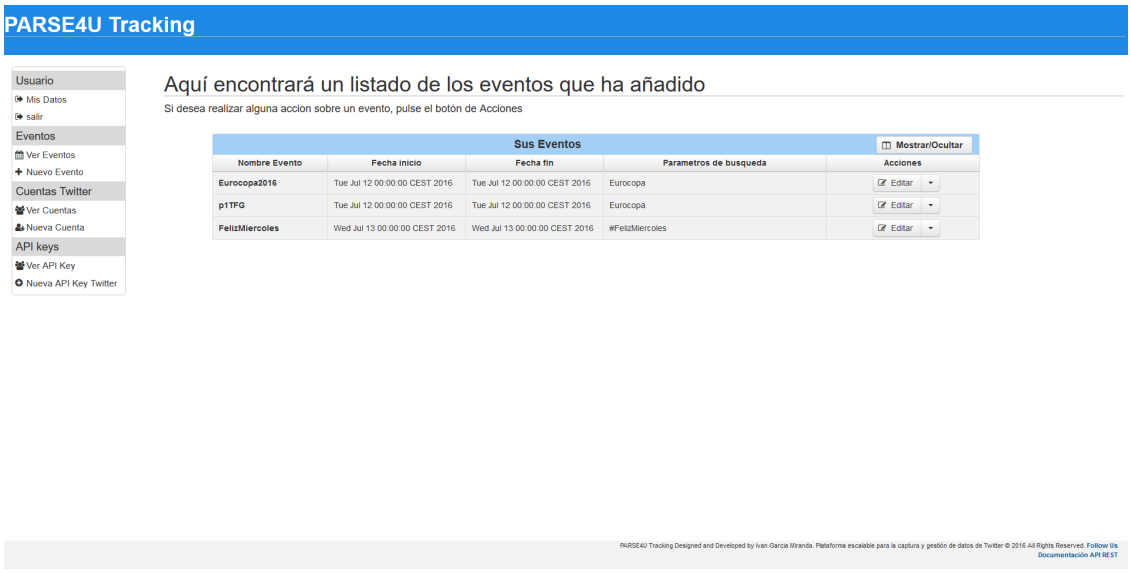


Figura 8.13: Página de "Ver Eventos" PARSE4U

8.2.5. Editar un Evento

Si el usuario desea realizar una acción como editar o borrar el evento, en cada uno de los eventos se puede ver un botón con las acciones. Cuando el usuario quiera editar uno de los eventos, al pulsar sobre el botón editar, se le mostrará un formulario con los datos actuales del evento. Si el usuario modifica alguno de los datos y pulsa en editar, se actualizarán los datos del evento y regresara a la lista de los eventos.

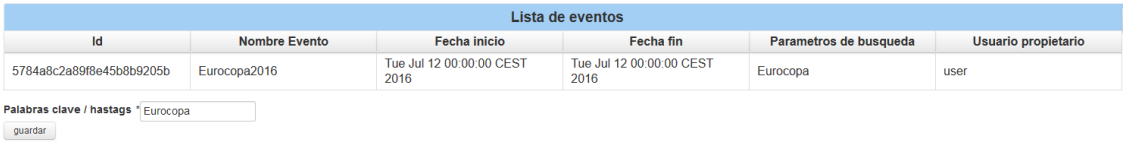


Figura 8.14: Página de "Editar Evento" PARSE4U

8.2.6. Borrar un Evento

En caso de que el usuario quiera eliminar un evento existente, al pulsar en el desplegable del botón de acciones se desplegaran las opciones. Si el usuario pulsa la opción de borrar, el sistema eliminara el evento.

8.2.7. Ver Cuentas de Twitter

Si el usuario desea ver un listado de las cuentas de "Twitter" que ha añadido al sistema, al pulsar en el menú lateral la opción, "Ver Cuentas" le mostrará un listado con sus cuentas.

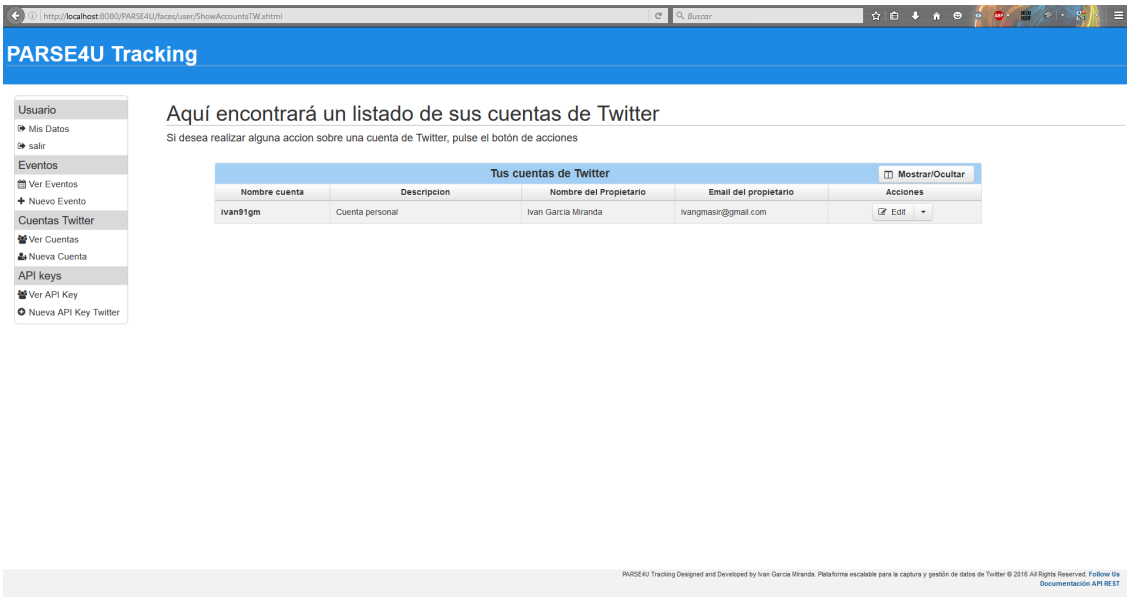


Figura 8.15: Página de "Ver Cuentas de Twitter" PARSE4U

8.2.8. Añadir Cuenta de Twitter

Para poder capturar datos de la red social “Twitter” se necesita añadir en el sistema la cuenta que se utilizara para conectarse a la API pública de “Twitter. Para añadir una nueva cuenta de “Twitter” el usuario deberá pulsar en la opción “Nueva Cuenta” en el menú lateral de la herramienta. A continuación, se abrirá un formulario, con los valores necesarios para añadir una nueva cuenta de “Twitter” al sistema.

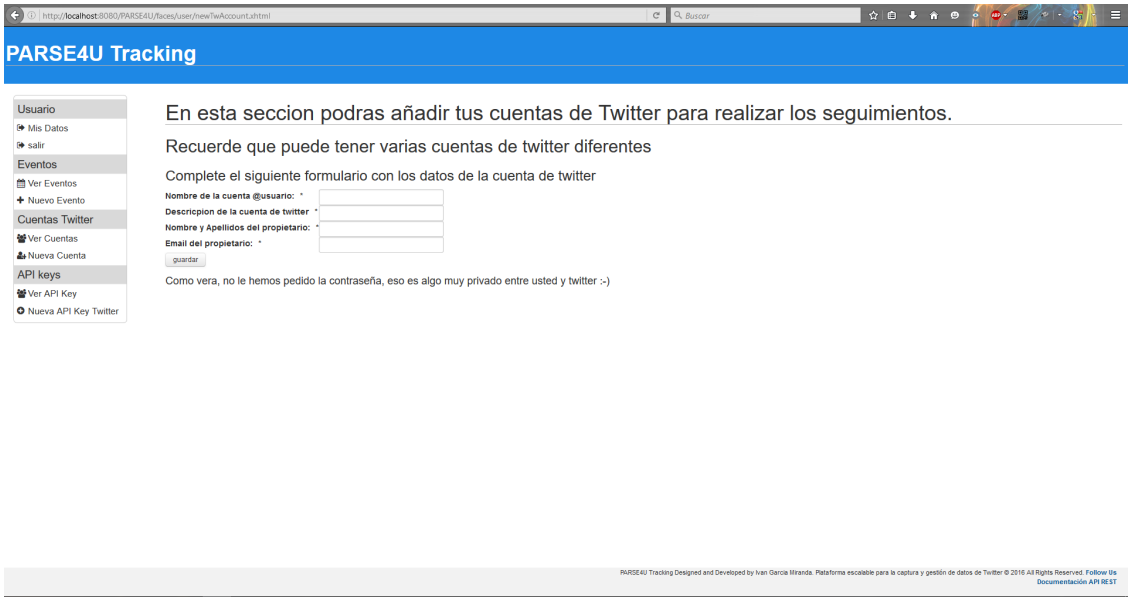


Figura 8.16: Página de “Nueva Cuenta” PARSE4U

8.2.9. Modificar Cuenta de Twitter

Al igual que en el caso de los eventos, si el usuario necesita modificar los datos de una cuenta de “Twitter” existente. En la tabla, a la derecha de cada uno de los eventos, aparece el botón “Editar”. al pulsarlo, se mostrará un formulario con los datos actuales de la cuenta de “Twitter”. Si el usuario modifica los datos y pulsa en “Guardar” se actualizarán los datos de la cuenta y se regresará al listado de cuentas.

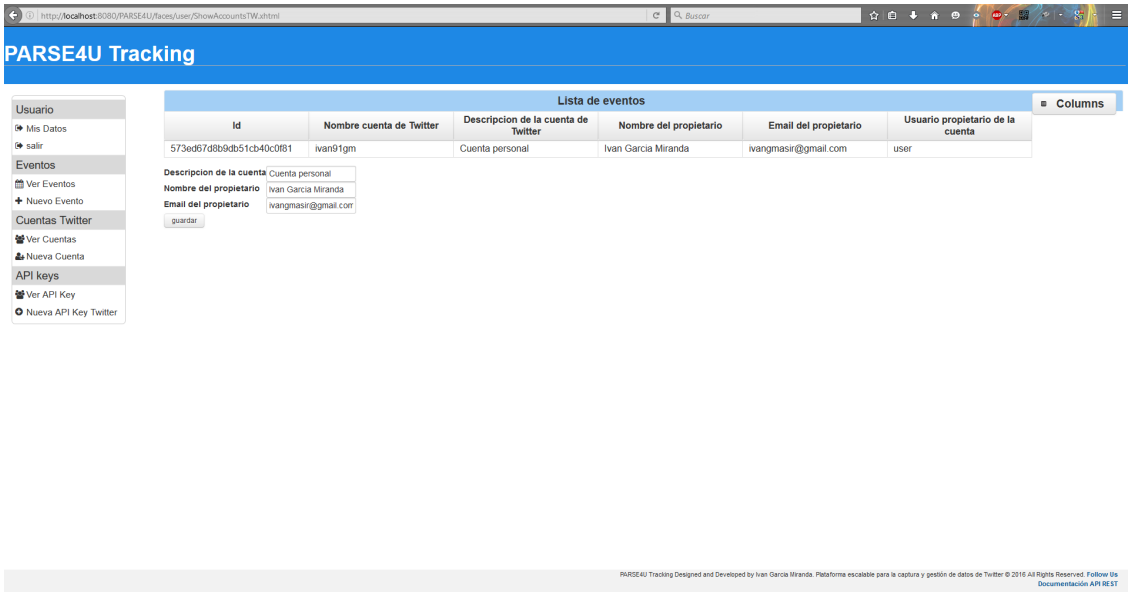


Figura 8.17: Página de "Editar Cuenta de Twitter" PARSE4U

Figura 8.18

8.2.10. Eliminar Cuenta de Twitter

Cuando un usuario desea eliminar una de sus cuentas de "Twitter", solo necesita pulsar el botón de acciones en el listado de las cuentas. Si pulsa la opción "Eliminar", la cuenta seleccionada será eliminada del sistema.

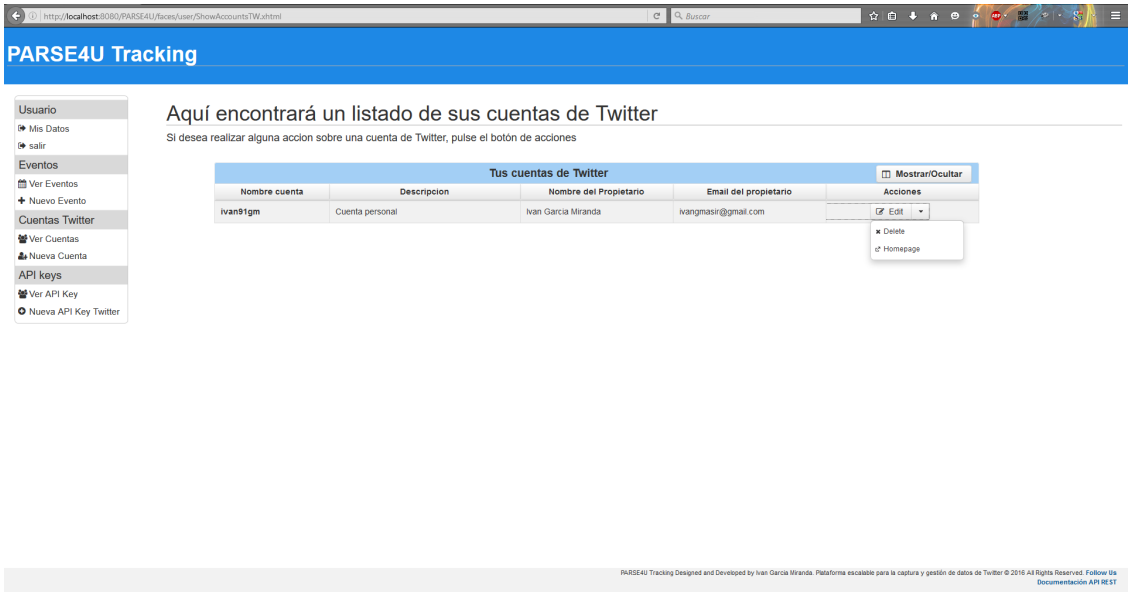


Figura 8.19: Página "Eliminar una Cuenta de Twitter" PARSE4U

8.2.11. Añadir Clave API Twitter

Una vez se han añadido al sistema, una cuenta de "Twitter" y un evento, solo es necesario añadir una aplicación de desarrollo registrada en "Twitter", al crear una aplicación de desarrollo

la red social, nos proporciona cuatro claves necesarias para conectar con la API de 'Streaming" y poder leer los datos que estén relacionados con el evento que se desea.

Para añadir una nueva clave de API, el usuario deberá pulsar la opción "Nueva API key Twitter" en el menú lateral. A continuación, se le mostrará un formulario donde podrá añadir las claves de la "API de Twitter". Para obtener las claves, el usuario deberá registrar una aplicación de desarrollo a través de esta dirección:https://apps.twitter.com/[10].

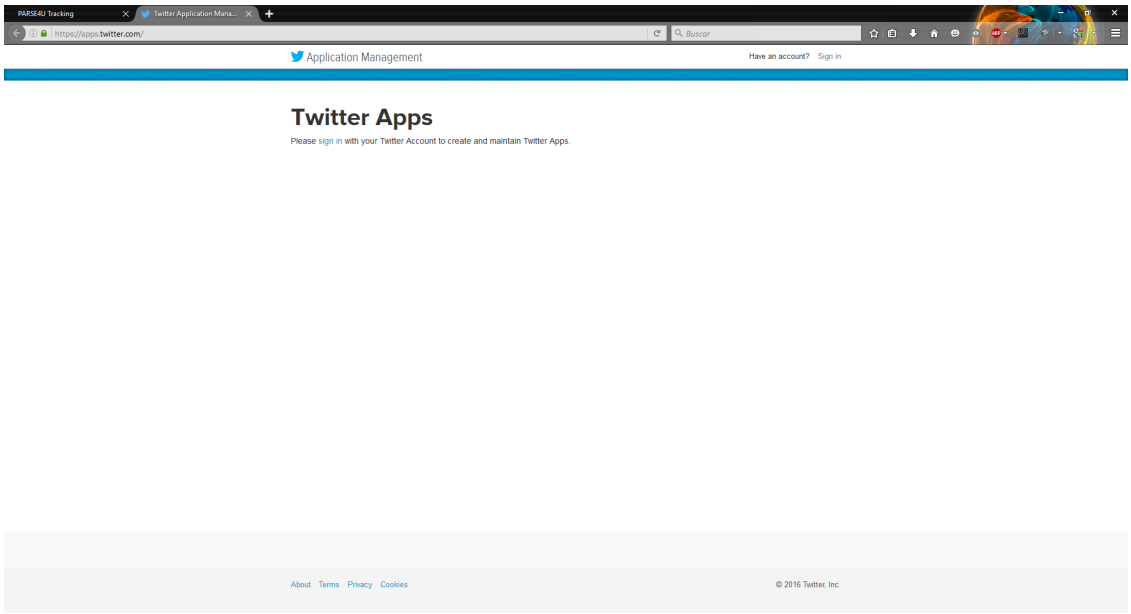


Figura 8.20: Inicio Registro de Nueva Aplicación en Twitter

Una vez autenticado con su cuenta de "Twitter" el usuario deberá pulsar en la opción "Create New App". La página nos pedirá varios datos relacionados con la aplicación.

1. **Name:** Nombre que describe a la aplicación.
2. **Description:** Descripción de la aplicación.
3. **Website:** Dirección web de la aplicación, por ejemplo: "http://parse4u.com"
4. **Yes, I agree:** Confirmar los términos y condiciones de uso de.
5. Una vez completados los datos anteriores, es necesario pulsar el botón de "Create your Twitter application"
6. Nos mostrará la información de la aplicación creada.
7. En el caso de que esta aplicación de desarrollo solo se utilice para recibir datos desde la "API de Streaming" y no para enviar mensajes a Twitter, se recomienda acceder al menú de "Permissions" y establecer la aplicación con permisos de "Read Only".
8. Para obtener las claves, se accede al menú de "Keys and Access Tokens"
9. En esta opción podemos regenerar las claves obtenidas y obtener unas nuevas

[10]Twitter Developer Application Management: https://apps.twitter.com/

10. Para obtener unas claves nuevas pulsamos en la opción “Create my Access token”
11. El sistema nos genera las claves necesarias para establecer la conexión.
12. Las claves generadas que debemos almacenar y añadir a la herramienta PARSE4U son: Consumer Key(API Key), Consumer Secret (API Secret), Access Token y Access Token Secret.

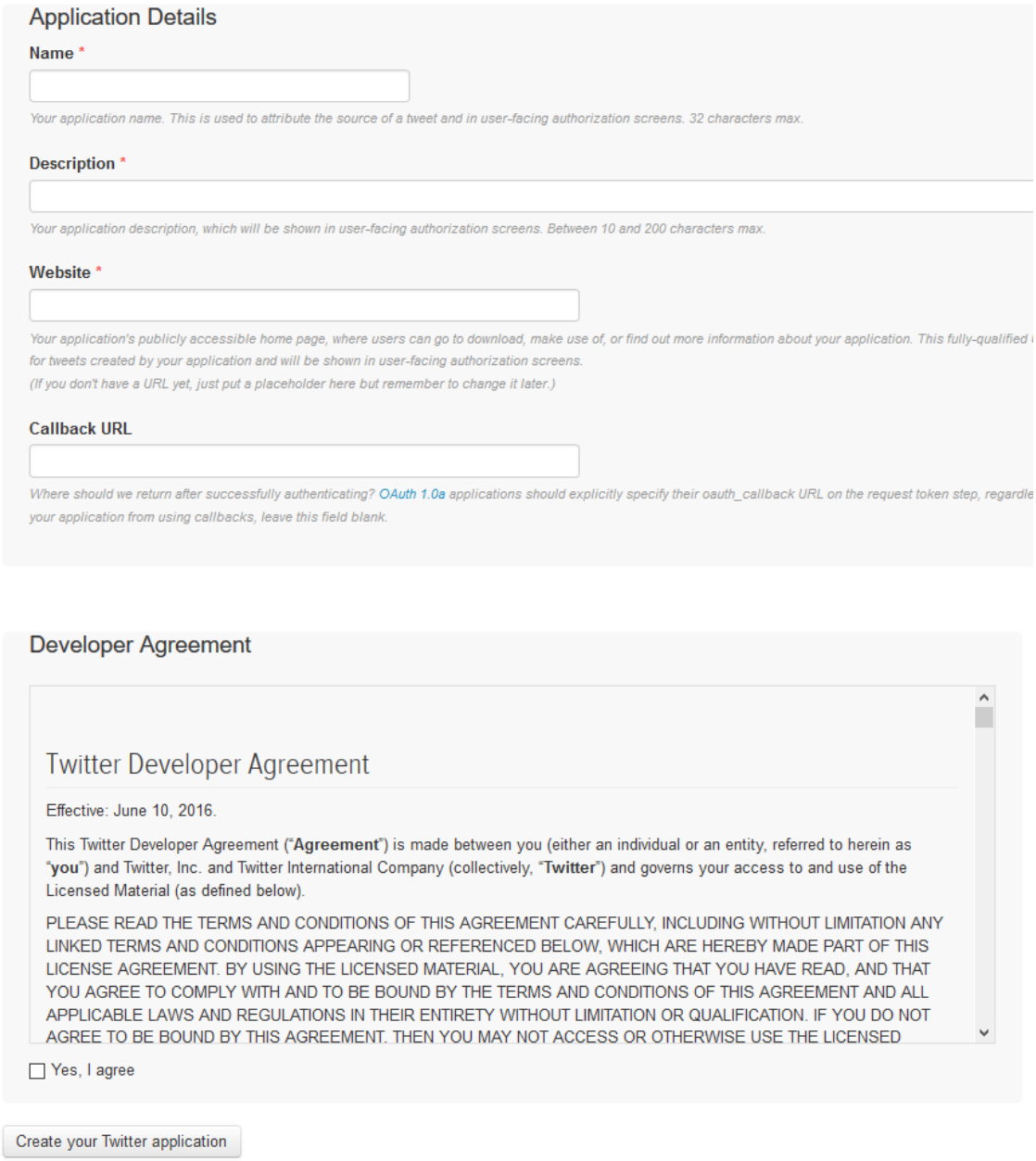


Figura 8.21: Formulario de Registro Nueva App Twitter

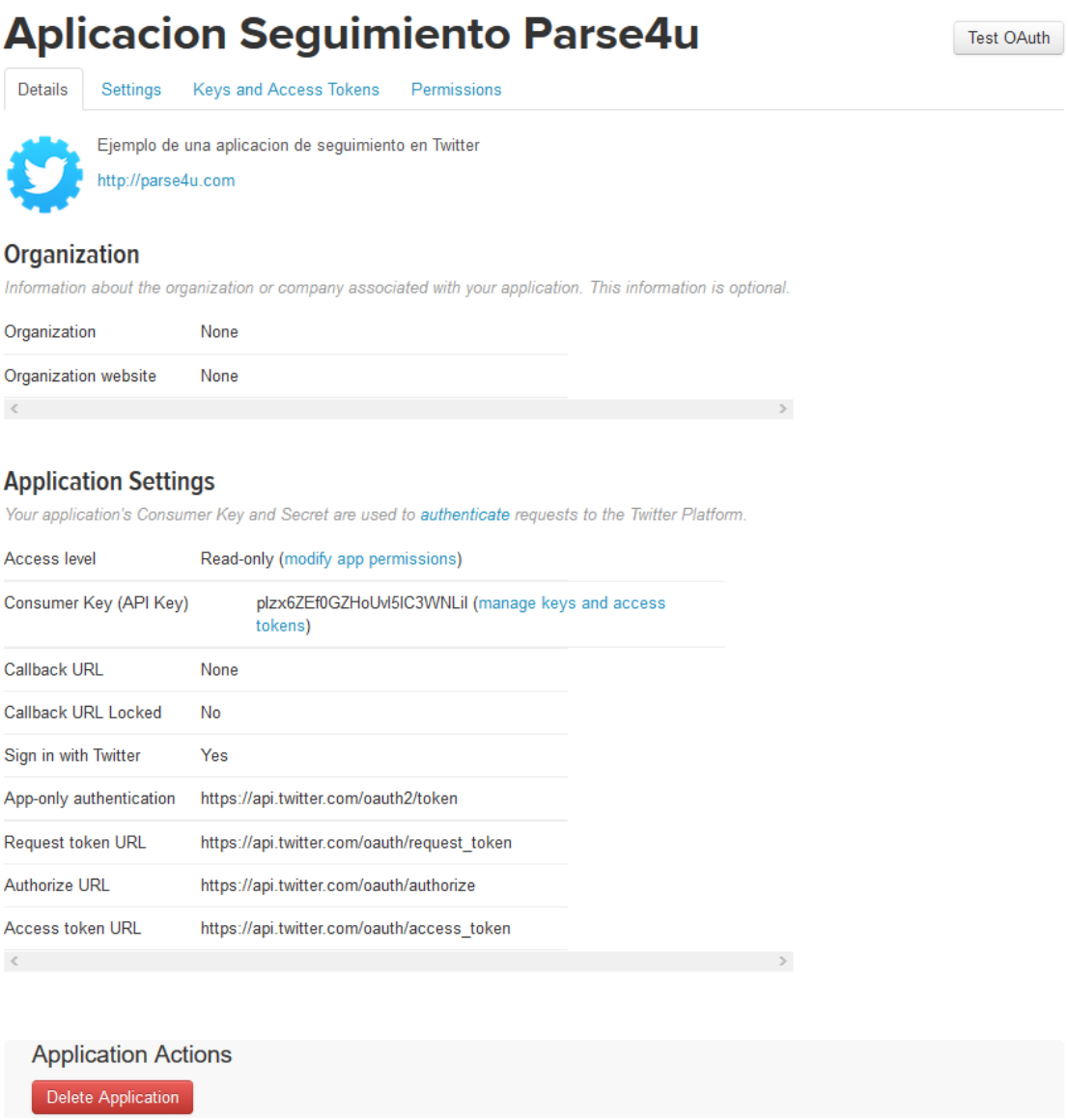


Figura 8.22: Detalles de una Aplicación de Ejemplo

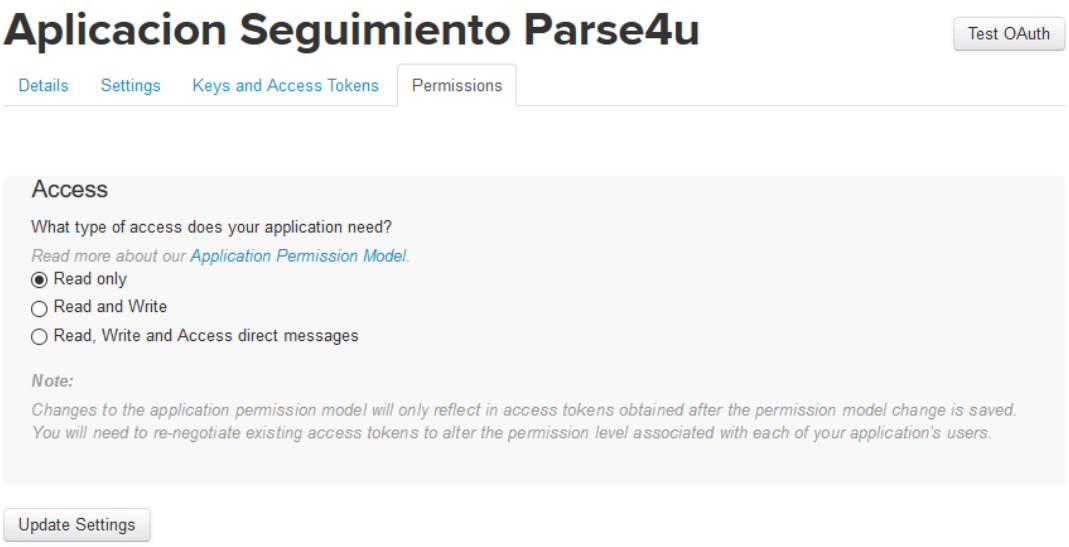


Figura 8.23: Establecer Permiso de “Read only” de una aplicación

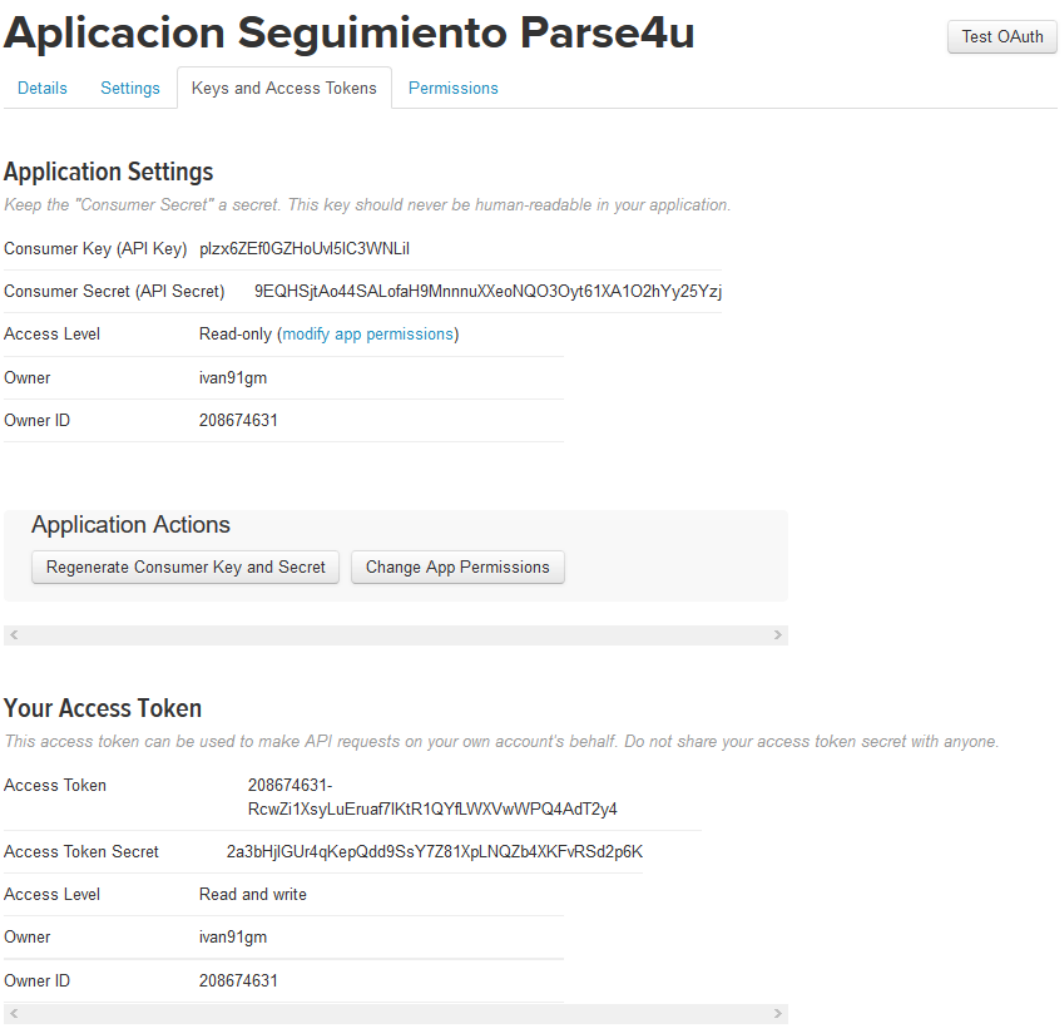


Figura 8.24: Ejemplo de Obtención de las Claves de API Twitter

Una vez obtenidas las claves de la “API de Twitter” debemos almacenarlas en la herramientas PARSE4U para poder capturar datos utilizando estas claves. Para ello, añadimos los datos obtenidos en el formulario de “Nueva Clave API Key Twitter”.

Figura 8.25: Página de “Nueva API Key Twitter” con los Datos de la Aplicación

Al copiar los datos de la página de “Twitter” hay que tener especial cuidado de no almacenar espacios en blanco, ya que existirán problemas en la autenticación al iniciar las capturas.

8.2.12. Iniciar Captura

El objetivo principal de la herramienta PARSE4U Tracking, es la captura de datos de “Twitter” para ello, el usuario deberá haber completado los pasos anteriormente descritos. Una vez completados, si el usuario desea iniciar una captura deberá acceder a la lista de eventos, pulsando en el menú lateral, la opción “Ver Eventos”. Se mostrará un listado de los eventos que el usuario tiene en el sistema.

Para iniciar la captura de un evento, deberá pulsa en el botón de acciones la opción “Captura”. A continuación, se mostrará los datos con los que se van a iniciar la captura, y se pedirá al usuario que seleccione una de la ”API Key“ que previamente ha añadido.

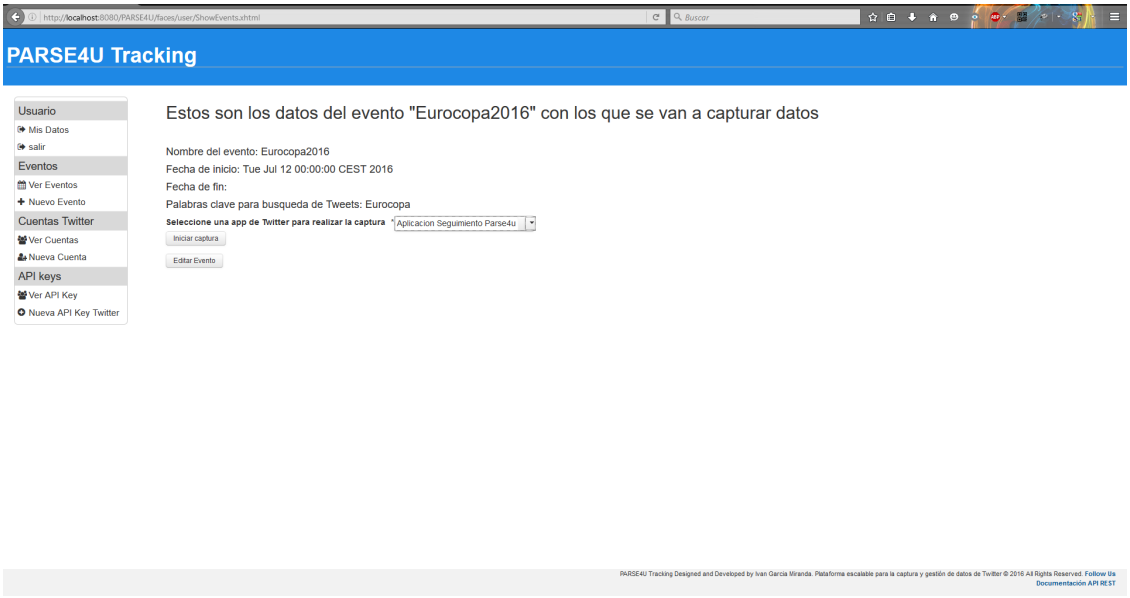


Figura 8.26: Página de "Iniciar Captura" PARSE4U

Al pulsar el botón de iniciar captura, el sistema comenzará a capturar datos con los parámetros indicados. A partir de ese momento, podrá acceder a los datos capturados a través de la API Rest que proporciona PARSE4U. Encontrará un enlace a la documentación de la Api Rest en el pie de página.

8.2.13. Cancelar Captura

Una vez que el usuario ha iniciado una o varias capturas de datos, si desea cancelar una captura de datos, deberá acceder a la lista de los eventos y pulsar botón de "Acciones" eligiendo la opción de "Captura". En caso de que el evento seleccionado, este en una captura activa, aparecerá el botón de "Parar Captura". Al pulsarlo se cerrará la conexión con "Twitter" y se dejaran de recibir datos.

Figura 8.27: Página de "Cancelar Captura" PARSE4U

8.2.14. Ver Clave API Twitter

Una vez que el usuario ha añadido "Claves de API Key Twitter" si desea ver los datos de cada una de ellas, modificar los datos o borrarlas, podrá realizar estas acciones a través de la página de "Ver Api Keys". En esta página se mostrará un listado de las aplicaciones de desarrollo que se han añadido.

Figura 8.28: Página de "Ver API Keys Twitter" PARSE4U

8.2.15. Modificar Clave API Twitter

En el caso de que el usuario desee modificar las claves de la aplicación de desarrollo, deberá pulsar la opción "Editar" del menú de Acciones. Al pulsarlo, se mostrará un formulario con los datos actuales, si estos se modifican; al pulsar en "Guardar" el sistema actualizará los valores.

Figura 8.29: Página de "Modificar API Key Twitter" PARSE4U

8.2.16. Borrar Clave API Twitter

En cambio, si lo que desea el usuario, es eliminar una aplicación de desarrollo, deberá pulsar la opción 'Borrar' del botón de Acciones.

Figura 8.30: Página de "'Borrar API Key Twitter" PARSE4U

Capítulo 9

Conclusiones y Trabajo Futuro

Por último, se muestran las conclusiones finales que podemos extraer tras la realización del proyecto y de los principales logros conseguidos. Además, se plantearán posibles ampliaciones y mejoras de la herramienta.

Una de las primeras dificultades con las que me topé en el inicio de este proyecto, es el volumen de datos con el que teníamos que trabajar. Para hacer una estimación del volumen de datos, se realizaron varias capturas de diferentes eventos utilizando sus "hashtag". Esto nos permitió hacernos una idea de que un evento relativamente pequeño nos proporcionaba aproximadamente 1 GB de datos. Esto implica que, si nuestro proyecto iba a ser usado por múltiples usuarios de forma concurrente, nuestra plataforma tendría que soportar grandes volúmenes de datos. Debido a esto, se optó por utilizar una base de datos "NoSQL" ya que estas están optimizadas e ideadas para trabajan con volúmenes de datos grandes de una manera más rápida que una base de datos relacional.

La siguiente dificultad con la que me encontré fue a la hora de trabajar con el sistema gestor de bases de datos "MongoDB". Es una base de datos relativamente nueva y aún está en constante evolución, lo que hace que mucha de la información que se encuentra en la red o libros este anticuada para las últimas versiones. Esto también implica que todavía no se integra adecuadamente con los entornos de desarrollo como "Netbeans" ni la API de persistencia de Java soporta este tipo de bases de datos. Por lo que todas las conexiones y operaciones con las bases de datos se han tenido que desarrollar manualmente.

La falta de integración con la API de persistencia de Java, sumado a que no se tenía una experiencia previa con bases de datos "NoSQL" provocó que la curva de aprendizaje en el comienzo del desarrollo fuera más lenta de lo previsto. Pero esto nos permitió aprender a utilizar una base de datos fuera de lo que estamos acostumbrados y nos obligó a pensar métodos y algoritmos óptimos para trabajar con la base de datos, que redujeran al mínimo posible la carga de trabajo.

Al necesitar hilos de ejecución diferentes para cada captura, el sistema debía ser capaz de gestionar y controlar cada uno de los hilos. Aunque se tenía un cierto conocimiento a la hora de trabajar con multi-hilos, al plantear esta metodología en un servidor de aplicaciones esta idea cambia. Debido a que, para el correcto funcionamiento, estos hilos de ejecución deben estar administrados y controlados por el servidor de aplicaciones, para garantizar que el servidor no deje de funcionar por un excesivo consumo de recursos. Se barajaron varias posibilidades, pero al final se decidió utilizar la solución óptima. Esta solución consiste en el uso de la API de Concurrencia que implementa la especificación Java EE 7. Esta API ha sido especialmente diseñada para trabajar de modo

"multitarea" en la aplicación empresarial. Esta API no se conocía anteriormente y la documentación encontrada en la red es prácticamente nula, únicamente la documentación oficial de "Oracle" nos permitió llevar a cabo la implementación de una manera adecuada.

9.1. Conclusiones

La conclusión principal del TFG es que al estar acostumbrado a utilizar frameworks, la integración de los IDEs con las bases de datos habituales y el uso de exclusivo de bases de datos relacionales, nos ayudan en el desarrollo de aplicaciones, pero a su vez, actúan de una manera opaca para el desarrollado. Lo que implica que muchas veces no se tenga conocimiento de las tareas que estas herramientas realizan. Un claro ejemplo son los frameworks como Hibernate o Eclipselink, nos permiten realizar conexiones y consultas a bases de datos habituales con un par de "clicks" de ratón al esta integrados en los IDEs. Pero cuando no utilizamos una base de datos habitual, es responsabilidad del programador diseñar e implementar todas las tareas que antes realizaban estas herramientas de una forma opaca.

La otra conclusión que se extrae de la realización del TFG es que a la hora, tanto de diseñar como de implementar un software, la solución que propones no suele ser la adecuada. Sino que, conforme el software evoluciona tienes que realizar múltiples modificaciones tanto en el análisis del software como en la implementación y documentación.

9.2. Trabajos Futuros

Aunque la herramienta desarrollada para este trabajo final es plenamente operativa existen varios aspectos que a lo largo del tiempo se podrían mejorar.

- **Re-diseño de la interfaz de usuario:** Aunque esta herramienta no está destinada al "gran público" la interfaz de usuario es algo que no se debería descuidar; por ello es recomendable realizar mejoras y optimización en la interfaz de usuario, haciendo así la herramienta más atractiva visualmente y no solo funcionalmente.
- **Inclusión de múltiples servicios Rest:** La versión del proyecto presentada ofrece varios servicios Rest a los posibles clientes. En esta primera versión se han implementado los servicios Rest necesarios para el funcionamiento de la otra herramienta que compone el proyecto PARSE4U, esta ha sido desarrollada en el Trabajo Final de Grado por Álvaro Monedero. Por lo que, si nuestra herramienta debe permitir a otros clientes utilizar los servicios Rest para realizar sus propios análisis o estudios, se deberán implementar muchos otros que puedan ser de utilidad, reduciendo el coste comunicacional de los clientes.
- **Implementación de búsqueda de "Tweets":** Aunque actualmente nuestra herramienta realiza capturas en tiempo real de "Tweets", los datos que se hayan generado antes de iniciar la captura y después de pararla no son almacenados, por lo que puede ser de especial interés, no solo capturar datos en tiempo real, sino, además de estos, hacer una búsqueda en el pasado para conseguir un mayor número de datos.

Existen múltiples caminos por los que esta herramienta podría seguir su desarrollo en el futuro. Nuestra idea es desarrollar la herramienta con las necesidades de nuestros potenciales clientes; escuchar a los usuarios de la aplicación para saber cuáles son sus necesidades e implementarlas dentro de la medida de lo posible.

El desarrollo de esta herramienta se ha llevado a cabo de la mejor manera posible, pero estamos trabajando en un producto software; este producto debe estar en continua evolución y mejora, por lo que algunas de las mejoras que se deberían llevar a cabo en el futuro son:

- **Reducción del coste computacional de los clientes:** El servidor debería proporcionar a los clientes algunas estadísticas y datos pre-calculados, para reducir el trabajo que estos deben realizar.
- **Optimizar y reducir el envío de información:** Se deberían optimizar los servicios Rest para enviar la menor cantidad de información posible, y enviar esta información comprimida, reduciendo así el tiempo de recepción de los datos por parte de los clientes.
- **Permitir la autenticación con "Twitter" mediante tokens:** La red social "Twitter" al igual que muchas APIs públicas, permiten autenticarse en sus servicios web utilizando tokens, al implementar esta mejora, reducimos el trabajo a realizar por los usuarios, ya que estos no tendrían que dar de alta nuevas aplicaciones de desarrollo en "Twitter" sino que mediante el token se crearían y autenticarían automáticamente.
- **Posibilitar el uso de la herramienta mediante servicios Rest:** De esta manera, permitiríamos a los usuarios la posibilidad de crear eventos e iniciar capturas sin acceder a la interfaz web de la aplicación. De esta manera se permitiría que las aplicaciones clientes creen eventos y realicen las tareas necesarias automáticamente. De igual manera, se podría desarrollar fácilmente una aplicación móvil para interactuar con la aplicación haciendo uso de estos servicios Rest.

9.3. Conocimientos Adquiridos

Tras haber completado el desarrollo de este proyecto, he adquirido una serie de conocimientos que serán de gran ayuda en mi futuro laboral.

En primer lugar, me ha permitido conocer el mundo de MongoDB y aprender el uso de un sistema gestor de bases de datos no conocido previamente. Aunque anteriormente ya la conocía, gracias a este trabajo he podido extender mi conocimiento en el ámbito de las bases de datos.

En segundo lugar, me ha permitido conocer en mayor profundidad algunas APIs que forman parte de la especificación Java EE, algunas de ellas ya se conocían anteriormente y otras de ellas no se sabia de su existencia. También, me ha permitido conocer más sobre el funcionamiento del servidor de aplicaciones Glassfish; se ha aprendido a solventar muchos de los problemas habituales con los despliegues de herramientas, y he podido aprender ligeramente algunas cuestiones sobre optimización de Glassfish.

Con la elaboración de este Trabajo Final de Grado he aprendido una metodología de trabajo independiente. Como alumno, estoy acostumbrado a seguir unas pautas y unas fechas para el desarrollo de trabajos. En cambio, en la elaboración de este trabajo debe ser el propio alumno es que se establezca sus propios hitos y busque su mejor forma de trabajo.

Ademas, en muchas ocasiones he tenido que aplicar los conocimientos adquiridos en otras asignaturas de la carrera para elaborar este proyecto:

- Para la programación de este proyecto ha sido importante tener los conocimientos adquiridos en las asignaturas **Programación Orientada a Objetos y Plataformas de Software Empresariales**. Permitiéndome tener unos conocimientos básico, tanto de Java como de Java EE.
- Para el diseño y optimización de las consultas sobre la base de datos, ha sido necesario los conocimientos adquiridos en la asignatura de **Administración de Bases de Datos**, lo que me ha permitido diseñar de una manera correcta y optimizada las operaciones sobre la base de datos.
- Para la elaboración de la documentación y el diseño de los diagramas, han sido necesarios los conocimientos adquiridos en las asignaturas **Proceso de Desarrollo del Software, Gestión de Proyectos Basados en la TI y Análisis de Requisitos**.

Referencias

[1] *Implementar un servicio Rest con JAX-RS (Jersey)*

[2] *Twitter API Documentation*

[3] *Twitter4J Documentation*

[4] *Integrating Swagger Into JAX-RS With Java EE 6 Specification.* Dezember 2013

[5] *MongoDB Realm for Glassfish.* November 2014

[6] Bollen, Johan ; Mao, Huina ; Zeng, Xiaojun: Twitter mood predicts the stock market. En: *Journal of Computational Science* 2 (2011), Nr. 1, p. 1–8

[7] Chodorow, Kristina: *MongoDB: the definitive guide.* .º'Reilly Media, Inc.", 2013

[8] Jendrock, Eric ; Cervera-Navarro, Ricardo ; Evans, Ian ; Haase, Kim ; Markito, William: *The Java EE 7 Tutorial.* Vol. 1. Addison-Wesley Professional, 2014

[9] Kumar, Saurav ; Maskara, Siddartha ; Chandak, Nitin ; Goswami, Saptarsi: Empirical Study of Relationship between Twitter Mood and Stock Market from an Indian Context. En: *International Journal of Applied Information Systems (IJJAISS)* 8 (2015), Mai, Nr. 7, p. 1–5

[10] Kumar, Shamanth ; Morstatter, Fred ; Liu, Huan: *Twitter Data Analytics.* New York, NY, USA : Springer, 2013

[11] Marchioni, Francesco: *MongoDB for Java developers.* Packt Publ., 2015

[12] Mkyong. *Java mongoDB Tutorial.* Mai 2011

[13] MongoDB. *The MongoDB 3.2 Manual*

[14] Primefaces. *PrimeFaces Documentation*

[15] Sadilek, Adam ; Brennan, Sean ; Kautz, Henry ; Silenzio, Vincent: nEmesis: which restaurants should you avoid today? En: *First AAAI Conference on Human Computation and Crowdsourcing*, 2013

[16] Sadilek, Adam ; Kautz, Henry A. ; Silenzio, Vincent: Modeling Spread of Disease from Social Interactions. En: *ICWSM*, 2012

[17] Sarhan, Asharaf. *java.util.concurrent.ThreadFactory Example.* Dezember 2014

[18] Seguin, Karl. *The Little MongoDB Book.* 2011

[19] Varaksin, Oleg: *PrimeFaces Cookbook.* Packt Publishing Ltd, 2013

Glosario

Término	Definición
Java EE	Es una plataforma de programación, parte de la Plataforma Java, para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process. Java EE es también considerada informalmente como un estándar debido a que los proveedores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE; estandarizado por The Java Community Process / JCP.
Twitter	Es una aplicación web gratuita de microblogging que reúne las ventajas de los blogs las redes sociales y la mensajería instantánea. Esta nueva forma de comunicación permite a sus usuarios estar en contacto en tiempo real con personas de su interés a través de mensajes breves de texto a los que se denominan Updates (actualizaciones) o Tweets.
Tweet	Un tweet o tuit es una publicación o actualización de estado realizada en la plataforma de microblogging conocida como Twitter. Como tal, un tuit es un mensaje cuyo límite de extensión son 140 caracteres. Puede contener letras, números, signos y enlaces.Los tuits, además, pueden contener hashtags o etiquetas que permiten establecer el tema o enfoque que se le pretende dar a la publicación o relacionarlo con un tema de conversación que se encuentra en el trending topic o tendencias del momento.
MongoDB	MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL. En lugar de guardar los datos en tablas como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos en documentos similares a JSON con un esquema dinámico (MongoDB utiliza una especificación llamada BSON) haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

Término	Definición
IDE	Un ambiente de desarrollo interactivo o entorno de desarrollo integrado (en ingles Integrated development environment)es una aplicación de software que proporciona los servicios esenciales para facilitar al desarrollador la tarea de desarrollo de software. Normalmente, un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDEs tienen auto-completado inteligente de código y editores de interfaces de usuario.
API	Son un conjunto de comandos, funciones y protocolos informáticos que permiten a los desarrolladores crear programas específicos para ciertos sistemas operativos. Las API simplifican en gran medida el trabajo de un creador de programas ya que no tiene que «escribir» códigos desde cero. Estas permiten al informático usar funciones predefinidas para interactuar con el sistema operativo o con otro programa.
Streaming	El streaming (también denominado transmisión, transmisión por secuencias, lectura en continuo, difusión en continuo o descarga continua) es la distribución digital de multimedia a través de una red de computadoras de manera que el usuario consume el producto(generalmente archivo de vídeo o audio) en paralelo mientras que se lo descarga. La palabra streaming se refiere a una corriente continua que fluye sin interrupción.Este tipo de tecnología funciona mediante un búfer de datos que va almacenando el flujo de descarga en la estación del usuario para luego mostrarle el material descargado. Esto se contrapone al mecanismo de descarga de archivos que requiere que el usuario descargue los archivos por completo para poder acceder al contenido.
Hashtag	Es una cadena de caracteres formada por una o varias palabras concatenadas y precedidas por una almohadilla o numeral (#). Es ,por lo tanto, una etiqueta de metadatos precedida de un carácter especial con el fin de que tanto el sistema como el usuario la identifiquen de forma rápida.Se usa en servicios web tales como Twitter Telegram,FriendFeed, Facebook, Google+, Instagram, Weibo o en mensajería basada,en protocolos IRC para señalar un tema sobre el que gira una conversación.
REST (Representational State transfer)	Es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.
JSON	Acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript aunque hoy, debido a su amplia adopción,como alternativa a XML se considera un formato de lenguaje independiente.

Anexos

Anexo A

Especificación de Casos de Uso

CDU-03	**Añadir Nueva Cuenta**
Versión	2.0 (26/06/2016)
Dependencias	CDU-02
Precondición	El usuario deberá estar previamente autenticado en el sistema.
Descripción	El sistema deberá permitir a los usuarios añadir nuevas cuentas de Twitter.
Secuencia normal	1. El usuario selecciona la opción de "Añadir nueva cuenta" 2. El sistema solicita los datos de una nueva cuenta. 3. El usuario introduce los datos y pulsa "Guardar" 4. El sistema comprueba que los datos sean correctos y almacena los datos introducidos en la base de datos.
Postcondición	Ninguna.
Excepciones	1. Si los datos introducidos no son correctos, el sistema mostrara un mensaje de error al usuario. 2. Si la cuenta de Twitter introducida ya existe , el sistema mostrará un mensaje de error al usuario.
Frecuencia	5 veces/día.
Importancia	Alta
Prioridad	Media
Comentarios	Ninguno.

Figura A.1: Especificación CDU-03 Añadir nueva cuenta

CDU-04	Modificar Datos de Cuenta
Versión	2.0 (26/06/2016)
Dependencias	CDU-02
Precondición	El usuario deberá estar previamente autenticado en el sistema y haber añadido al menos una cuenta de Twitter
Descripción	El sistema deberá permitir a los usuarios modificar los datos de una cuenta de Twitter existente.
Secuencia normal	1. El usuario selecciona la opción de editar. 2. El sistema muestra los datos actuales de la cuenta. 3. El usuario modifica los datos deseados y pulsa "Guardar". 4. El sistema comprueba los datos introducidos y actualiza la información.
Postcondición	Ninguna.
Excepciones	1. Si el usuario no ha introducido ninguna cuenta de Twitter, el sistema mostrará un mensaje de error. 2. Si los datos introducidos no son correctos, el sistema mostrará un mensaje de error al usuario
Frecuencia	1 veces/día.
Importancia	Baja
Prioridad	Baja
Comentarios	Ninguno.

Figura A.2: Especificación CDU-04 Modificar cuenta

CDU-05	Eliminar Cuenta
Versión	2.0 (26/06/2016)
Dependencias	CDU-02
Precondición	El usuario deberá estar previamente autenticado en el sistema y haber añadido al menos una cuenta de Twitter
Descripción	El sistema deberá permitir a los usuarios eliminar una cuenta de Twitter previamente introducida en el sistema
Secuencia normal	1. El usuario pulsa la opción de "Eliminar" sobre una de sus cuentas de Twitter 2. El sistema elimina la cuenta de Twitter.
Postcondición	Ninguna.
Excepciones	1. Si el usuario no ha introducido previamente una cuenta de Twitter, el sistema mostrará un mensaje de error al usuario.
Frecuencia	1 veces/día.
Importancia	Baja
Prioridad	Media
Comentarios	Ninguno.

Figura A.3: Especificación CDU-05 Borrar cuenta

CDU-07	Añadir Claves de API
Versión	2.0 (26/06/2016)
Dependencias	CDU-06
Precondición	El usuario deberá estar previamente autenticado en el sistema.
Descripción	El sistema deberá permitir a los usuarios añadir nuevas claves de API de Twitter al sistema.
Secuencia normal	1. El usuario selecciona la opción de "Añadir nueva clave API" 2. El sistema muestra los atributos de una clave de API. 3. El usuario introduce los datos de las claves de API proporcionados por Twitter. 4. El sistema comprueba los datos y los añade a la herramienta.
Postcondición	Ninguna.
Excepciones	1. Si los datos introducidos no son correctos, el sistema mostrará un mensaje de error al usuario. 2. Si la clave de API ya existe, el sistema mostrara un mensaje de error al usuario.
Frecuencia	5 veces/día.
Importancia	Alta
Prioridad	Media
Comentarios	Ninguno.

Figura A.4: Especificación CDU-07 Añadir claves API

CDU-08	Modificar Claves de API
Versión	2.0 (26/06/2016)
Dependencias	CDU-06
Precondición	El usuario deberá estar previamente autenticado en el sistema y haber añadido al menos una clave de API en el sistema
Descripción	El sistema deberá permitir a los usuarios modificar los datos de una Clave de API existente.
Secuencia normal	1. El usuario selecciona la opción de editar una de sus claves de API. 2. El sistema mostrara los datos actuales de la clave de API. 3. El usuario modifica los datos deseados y pulsa "Guardar". 4. El sistema comprueba los datos y actualiza en la herramienta.
Postcondición	Ninguna.
Excepciones	1. Si los datos introducidos no son correctos, el sistema mostrara un mensaje de error al usuario. 2. Si el usuario no ha añadido previamente ninguna clave de API el sistema mostrará un mensaje de error al usuario
Frecuencia	2 veces/día.
Importancia	Baja
Prioridad	Baja
Comentarios	Ninguno.

Figura A.5: Especificación CDU-08 Modificar claves API

CDU-09	Borrar Claves de API
Versión	2.0 (26/06/2016)
Dependencias	CDU-06
Precondición	El usuario deberá estar previamente autenticado en el sistema y haber añadido al menos una clave de API en el sistema
Descripción	El sistema deberá permitir a los usuarios borrar una Clave de API existente.
Secuencia normal	1. El usuario selecciona la opción" Borrar" sobre una de sus claves de API. 2. El sistema elimina la clave de API.
Postcondición	Ninguna.
Excepciones	Ninguna.
Frecuencia	2 veces/día.
Importancia	Baja
Prioridad	Baja
Comentarios	Ninguno.

Figura A.6: Especificación CDU-09 Eliminar claves API

CDU-11	Añadir Evento
Versión	2.0 (26/06/2016)
Dependencias	CDU-10
Precondición	El usuario deberá estar previamente autenticado en el sistema.
Descripción	El sistema deberá permitir a los usuarios añadir nuevos eventos.
Secuencia normal	1. El usuario selecciona la opción de "Nuevo evento" 2. El sistema solicita los atributos de un evento. 3. El usuario introduce los datos del evento y pulsa "Guardar" 4. El sistema comprueba que lo datos son correctos y los almacena.
Postcondición	Ninguna.
Excepciones	1. Si los datos introducidos no son correctos, el sistema mostrará un mensaje de error al usuario. 2. Si el evento introducido ya existe en el sistema, se mostrará un mensaje de error al usuario
Frecuencia	10 veces/día.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno.

Figura A.7: Especificación CDU-11 Añadir evento

CDU-12	Modificar Evento
Versión	2.0 (26/06/2016)
Dependencias	CDU-10
Precondición	El usuario deberá estar previamente autenticado en el sistema y deberá haber añadido al menos un evento en el sistema.
Descripción	El sistema deberá permitir a los usuarios añadir nuevos eventos.
Secuencia normal	1. El usuario selecciona la opción de "Modificar" sobre uno de sus eventos. 2. El sistema muestra los datos actuales del evento. 3. El usuario modifica los datos deseados y pulsa "Guardar" 4. El sistema comprueba los datos y actualiza la información.
Postcondición	Ninguna.
Excepciones	1. Si los datos introducidos no son correctos, el sistema mostrará un mensaje de error al usuario. 2. Si el evento seleccionado está actualmente capturando datos, el sistema no permitirá modificar los datos del evento.
Frecuencia	5 veces/día.
Importancia	Media
Prioridad	Baja
Comentarios	Ninguno.

Figura A.8: Especificación CDU-12 Modificar evento

CDU-13	Borrar Evento
Versión	2.0 (26/06/2016)
Dependencias	CDU-10
Precondición	El usuario deberá estar previamente autenticado en el sistema y deberá haber añadido al menos un evento en el sistema.
Descripción	El sistema deberá permitir a los usuarios borrar un evento
Secuencia normal	1. El usuario selecciona la opción de "Borrar" sobre uno de sus eventos. 2. El sistema marca el evento como eliminado en el sistema.
Postcondición	El evento queda marcado como eliminado en el sistema, no se podrá modificar, visualizar ni iniciar una captura, pero sus datos seguirán siendo accesibles a través de los servicios web.
Excepciones	1. Si el evento seleccionado se encuentra capturando datos, el sistema mostrará un mensaje de error al usuario
Frecuencia	1 veces/día.
Importancia	Baja
Prioridad	Baja
Comentarios	Ninguno.

Figura A.9: Especificación CDU-13 Eliminar evento

CDU-15	Iniciar Captura
Versión	2.0 (26/06/2016)
Dependencias	CDU-13
Precondición	El usuario deberá estar previamente autenticado en el sistema.
Descripción	El sistema deberá permitir a los usuarios realizar acciones sobre las capturas de datos de sus eventos
Secuencia normal	1. El usuario selecciona la clave de API con la que desea capturar datos del evento. 2. El sistema inicia una captura de datos y marca la captura esta activa.
Postcondición	El sistema deberá crear un nuevo hilo para cada captura.
Excepciones	Ninguna.
Frecuencia	10 veces/día.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno.

Figura A.10: Especificación CDU-15 Iniciar captura

CDU-16	Cancelar Captura
Versión	2.0 (26/06/2016)
Dependencias	CDU-13
Precondición	El usuario deberá estar previamente autenticado en el sistema.
Descripción	El sistema deberá permitir a los usuarios realizar acciones sobre las capturas de datos de sus eventos
Secuencia normal	1. El usuario selecciona el evento que quiere cancelar su captura. 2. El sistema cancela la captura de datos y marca la captura como cancelada.
Postcondición	El sistema deberá eliminar el hilo de la captura.
Excepciones	Ninguna.
Frecuencia	10 veces/día.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno.

Figura A.11: Especificación CDU-16 Cancelar captura

CDU-19	Añadir usuarios
Versión	2.0 (26/06/2016)
Dependencias	CDU-18
Precondición	El usuario deberá estar previamente autenticado en el sistema como un usuario administrador
Descripción	El sistema deberá permitir a los usuarios administradores añadir nuevos usuarios a la aplicaciones
Secuencia normal	1. El usuario selecciona la opción de "Añadir usuario" 2. El sistema solicita los datos de un usuario. 3. El usuario introduce los datos y pulsa en "Guardar" 4. El sistema comprueba los datos y añade un nuevo usuario al sistema.
Postcondición	Ninguna.
Excepciones	1. Si los datos introducidos no son correctos, el sistema mostrará un mensaje de error al usuario. 2. Si el usuario introducido ya existe se mostrará un mensaje de error al usuario.
Frecuencia	2 veces/día.
Importancia	Media
Prioridad	Baja
Comentarios	Ninguno.

Figura A.12: Especificación CDU-19 Añadir usuarios

CDU-20	Modificar usuarios
Versión	2.0 (26/06/2016)
Dependencias	CDU-18
Precondición	El usuario deberá estar previamente autenticado en el sistema como un usuario administrador
Descripción	El sistema deberá permitir a los usuarios administradores modificar los usuarios a la aplicaciones
Secuencia normal	1. El usuario selecciona la opción de "Modificar" un usuario. 2. El sistema muestra los datos actuales del usuario. 3. El usuario modifica los datos deseados y pulsa en guardar. 4. El sistema comprueba si los datos son correctos y actualiza la información.
Postcondición	Ninguna.
Excepciones	1. Si los datos introducidos no son correctos, el sistema mostrará un mensaje de error al usuario.
Frecuencia	2 veces/día.
Importancia	Baja
Prioridad	Baja
Comentarios	Ninguno.

Figura A.13: Especificación CDU-20 Modificar usuarios

CDU-21	Borrar usuarios
Versión	2.0 (26/06/2016)
Dependencias	CDU-18
Precondición	El usuario deberá estar previamente autenticado en el sistema como un usuario administrador
Descripción	El sistema deberá permitir a los usuarios administradores eliminar los usuarios a la aplicaciones
Secuencia normal	1. El usuario selecciona la opción de "Eliminar" un usuario. 2. El sistema eliminará los datos del usuario.
Postcondición	Ninguna.
Excepciones	Ninguna.
Frecuencia	2 veces/día.
Importancia	Baja
Prioridad	Baja
Comentarios	Ninguno.

Figura A.14: Especificación CDU-21 Borrar usuarios