UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1

Módulo de conexión a la red social Twitter para la Plataforma para la Red Social Twitter (PRST).

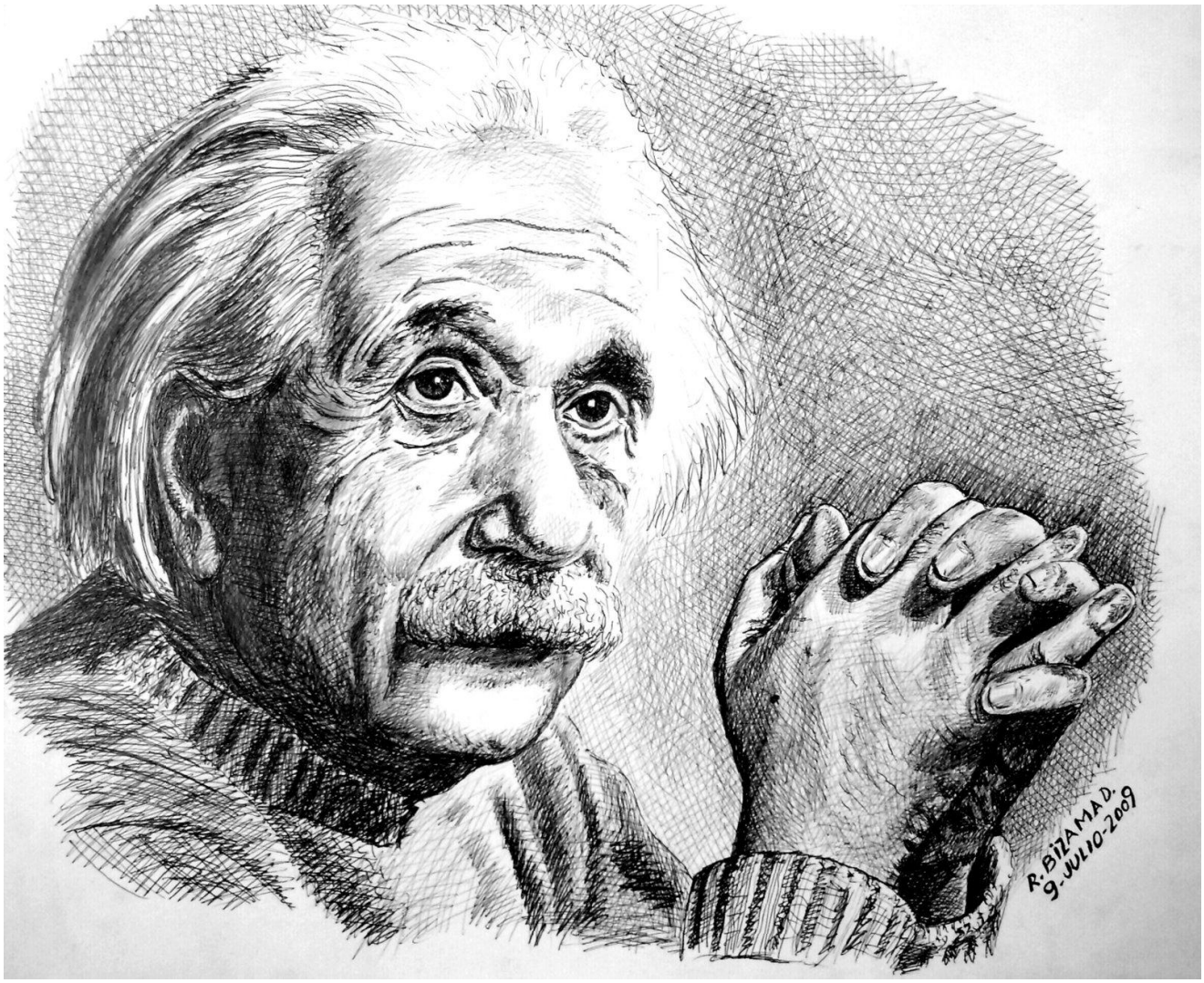
Trabajo de Diploma para Optar por el Título de Ingeniero en Ciencias Informáticas.

Autores: Surennys González Figueredo
Yuriesky Madrigal Vilches.

Tutores: Ing. José Gabriel Espinosa.
Ing. Odelkis Rodríguez Irsula.
Ing. Ezequiel Martínez Justiz.

La Habana, Junio de 2014

Pensamiento



HAY UNA FUERZA MOTRIZ MÁS PODEROSA QUE EL VAPOR,

LA ELECTRICIDAD Y LA ENERGÍA ATÓMICA:

¨LA VOLUNTAD¨.

Declaración de Autoría

Declaramos ser los únicos autores de este trabajo y autorizamos al Centro de Ideoinformática de la Universidad de las Ciencias Informáticas; para que haga el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año____.

Surennys González Figueredo

_____

Firma del autor

Yuriesky Madrigal Vilches

_____

Firma del autor

Ing. José Gabriel Espinosa

_____

Firma del tutor

Ing. Ezequiel Martínez Justiz

_____

Firma del tutor

Ing. Odelkis Rodríguez Irsula

_____

Firma del tutor

Agradecimientos

De Surennys

A mi mamita linda, por su amor invaluable e incondicional, sus consejos, su apoyo en cada minuto de mi vida. Por confiar en mí siempre y a mi padrastro Manuel.

A mi esposo Renier, por ser mi reparador de sueños y mi ángel de la guarda desde que lo conocí, por todo su amor y comprensión.

A mis tutores José Gabriel, Odelkis, Ezequiel, que se convirtieron casi sin querer en buenos amigos, por su colaboración, asesoramiento y por su tiempo y paciencia desde el principio.

A mi hermano Orlando, por ser único y quererme tanto y mi amigo Dianlys por su ayuda incondicional..

A toda mi familia, en especial a mis tías Elaine, Nancy, Milagro y Made y mis 3 abuelitos porque desde que comencé este gran viaje que es la universidad me han ayudado, de todas las formas posibles, a realizar mi sueño, a mi familia de Colón por llenarme de amor y cariño siempre.

A mis amigas Llilian, Yamilki, Ailén, Mimin, Dayamis, Dannelys por demostrarme que siempre estarán ahí, en las buenas y en las malas, y por estarlo durante tanto tiempo.

A mi compañero de tesis, por proveerme de los conocimientos necesarios para realizar este trabajo y por compartir su esfuerzo y conocimiento.

A mis compañeras de apartamento y compañeros del laboratorio 203.

A todo el que me preguntó alguna vez: “y la tesis… ¿cómo va?”

A las personas que de una forma u otra han hecho posible que hoy escriba, con la satisfacción del deber cumplido, estas líneas de agradecimientos...Muchas gracias.

De Yuriesky.

A mis padres porque siempre han estado cuando más los he necesitado y por apoyarme siempre.

A mi familia por su apoyo y su cariño.

A mis compañeros del grupo con los que he compartido momentos inolvidables en estos cinco años.

A Surennys mi compañera de tesis por compartir su esfuerzo y conocimiento.

A mis tutores José Gabriel, Ezequiel y Odelkis que son parte importante de este resultado, gracias por toda la dedicación brindada a lo largo del desarrollo de este trabajo de diploma.

A mis colegas de juegos y del laboratorio 203.

A mis tías por su preocupación y su apoyo incondicional……Muchas gracias a todos.

Dedicatoria

De Yuriesky

A toda mi familia, en especial a mis padres Flora Vilches González y Marcelino Madrigal Sánchez, a mi padrastro Enrique Hernández Álvarez y a mis tres hermanos Yuniesky, Elizabeth y Edisnay.

De Surennys

A los amores de mi vida, mis sobrinas Lorena, Loraine, Mariangel, Shanella y a mi tata Melisa para que sigan mi ejemplo y siempre se sienta orgullosa de su tía quien tanto las ama.

Resumen

El crecimiento acelerado de las Tecnologías de la Información y las Comunicaciones (TIC), específicamente en el área de las redes sociales, ha trascendido el mundo digital haciéndose más tangible. Entre los sistemas de redes sociales de mayor crecimiento se encuentra el servicio de Twitter. Este es un servicio gratuito de *microblogging*, que permite a sus usuarios enviar micro entradas basadas en textos cortos, llamados *tweets,* de una longitud máxima de 140 caracteres. Para garantizar una presencia activa, efectiva y eficiente en esta red social no es suficiente con las facilidades que brinda este sistema mediante su interfaz Web. Para esto se requiere utilizar otros canales de comunicación con Twitter que permiten de cierta forma automatizar la interacción con la red social.

En la Universidad de las Ciencias Informáticas se encuentra el Centro de Ideoinformática (CIDI), el cual tiene entre sus objetivos de trabajo desarrollar soluciones para la Web. Como paso inicial, de una plataforma para la interacción con la red social Twitter se ha iniciado la elaboración de un módulo que sirva de punto de intercambio de contenidos con Twitter. De esta manera, se reduciría la redundancia de código y la duplicidad de esfuerzos; además, sería posible establecer métricas sobre el uso de la red social así como optimizar el uso del ancho de banda de la Universidad.

Palabras clave: Aplicaciones federadas; Twitter; Interacción con redes sociales.

Índice

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

Introducción

El constante desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) constituye un factor fundamental en el avance de las diferentes ramas de la sociedad, prometiendo cambios principalmente en la informática y las telecomunicaciones. El impacto de la web social, o Web 2.0, dentro de estas tecnologías, ha influido considerablemente en el desarrollo de las mismas; poniendo al alcance de todos la posibilidad de integrarse como ciudadanos en el mundo de la Internet, manteniendo un intercambio libre y gratuito de información. "*La Web 2.0 se caracteriza por el desarrollo de tecnologías orientadas a la participación y colaboración entre comunidades virtuales, siendo las Redes Sociales, su máximo exponente"* (Dale Dougherty, 2004).

Las redes sociales "son servicios basados en la web que permiten a los individuos construir un perfil público o semipúblico dentro de un sistema delimitado, articular una lista de otros usuarios con los que comparten una conexión" (Social Network Sites, 2007). Una de las redes sociales más populares es la red social Twitter.

Twitter es un servicio gratuito de *microblogging* creado a finales del 2006, que permite a sus usuarios enviar microentradas basadas en textos cortos llamados *tweets*, de una longitud máxima de 140 caracteres. El envío de estos mensajes se puede realizar a través del sitio web de Twitter o por el servicio de mensajes cortos (Short Message Service dada sus siglas en Inglés SMS) desde un teléfono móvil (Historia de Twitter, 2013) .

El servicio rápidamente comenzó a ganar seguidores, se estima que cuenta con más de 560 millones de usuarios registrados en el 2013, generando un total de 15.000 *tweets* por segundo y unos 350 millones de *tweets* por día, convirtiéndose en uno de los diez sitios más visitados en Internet. Más allá de ser una simple red social, se ha transformado en una importante plataforma para los movimientos sociales, un medio en el que se comparte información constantemente, un catalizador de la globalización y de acontecimientos recientes que han sacudido el plano político del planeta. Debido a la actualización constante del contenido se considera una gran fuente de datos y uno de los sistemas de comunicación más utilizados; no sólo para información intrascendente, social, sino, como herramienta de comunicación entre profesionales (Los-records-de-twitter, 2013).

Twitter proporciona interfaces de programación de aplicaciones (Application *Programming Interface*, dada sus siglas en Inglés API) a los desarrolladores para integrar sus soluciones o interactuar con los servicios de la red social desde y fuera de ésta; ofreciendo las facilidades necesarias para que las aplicaciones de terceros interactúen con ella mediante sus *API's.* Dado el carácter modular de estas *API's*, es posible

desarrollar de forma independiente una interfaz para cada aplicación de terceros e integrarlos a través de una única interfaz. Por lo tanto, cualquier cambio en la red social Twitter sólo afectaría al módulo de esta red, sin modificar el resto necesariamente.

El Centro de Ideoinformática (CIDI) de la Universidad de las Ciencias Informáticas (UCI), tiene entre sus misiones proveer al país de soluciones, productos y servicios relacionados con las tecnologías de Internet. La red social Twitter forma parte de los campos de investigación dentro de CIDI; por considerarse una red social altamente exitosa con un heterogéneo y variado grupo de usuarios además del gran número de contenido público escrito por los mismos usuarios que a menudo muestran la opinión de estos respecto a determinados temas. Este contenido puede ser analizado para obtener una visión de la opinión general de todos los usuarios, lo cual puede ser útil para realizar estudios sociológicos. Sin embargo, esto tiene como desventaja que la cantidad de información a analizar es tanta que en ocasiones su posterior análisis se hace complejo.

Por tanto, se hace imprescindible la extensión de la Plataforma de la Red Social Twitter (PRST) con soluciones que permitan integrar y optimizar sus actividades. Estas soluciones no pueden desarrollarse contando únicamente con las facilidades que brinda la interfaz web de Twitter pues se necesitarían grandes cantidades de recursos humanos y tecnológicos que harían inviable o demasiado costoso el logro de los objetivos propuestos. Como paso inicial para el desarrollo de esta plataforma, se hace imprescindible el desarrollo de un módulo para la interacción con la red social Twitter que sirva de punto de intercambio de contenidos con la misma, y funcione como una capa de abstracción.

Este módulo deberá exponer sus funcionalidades como servicios para que estos sean consumidos por el resto de las aplicaciones federadas[1] de la PRST. Permitirá la interacción con los servicios de Twitter, de forma tal que se eviten redundancias, se aprovechen mejor los recursos tecnológicos y humanos involucrados en estas tareas. A su vez, logrará que todas las aplicaciones de la plataforma sean capaces de comunicarse y compartir datos mediante la utilización de un lenguaje común, consiguiendo una mayor rentabilidad de las inversiones a través de la optimización de los medios disponibles y optimizando el uso del ancho de banda de la Universidad.

Por lo antes planteado, se identifica el siguiente **problema de la investigación:** ¿Cómo abstraer a la Plataforma para la red social Twitter (PRST) de los mecanismos de acceso a la red social Twitter?

Lo antes expuesto lleva a analizar la comunicación entre sistemas informáticos planteando como **objeto de estudio** la interoperabilidad entre aplicaciones informáticas, enfocándose en el **campo de acción** los servicios web para la interacción con las *API´s* de la red social Twitter.

[1] Aplicaciones que en conjunto con el módulo de conexión a la red social Twitter conformarán la Plataforma Red Social Twitter.

Para darle solución al problema se plantea como **objetivo general** de este trabajo de diploma: Desarrollar una aplicación, que sirva como capa de abstracción para la comunicación de la PRST con la red social Twitter, y que exponga sus funcionalidades como servicios para que estos sean consumidos por el resto de las aplicaciones federadas en la PRST.

Desglosando de este los siguientes **objetivos específicos**:

1. Analizar el marco teórico conceptual y el estado del arte respecto a las tecnologías actuales para el desarrollo de herramientas para la abstracción de la interacción con la APIs de Twitter.
2. Diseñar e implementar una aplicación que permita la interacción indirecta de las aplicaciones federadas de la PRST con Twitter.
3. Validar el correcto funcionamiento de la aplicación para la abstracción del acceso a Twitter para la PRST.

Para guiar la investigación se plantea como **idea a defender:** El desarrollo de una aplicación, que sirva como capa de abstracción para la comunicación de la PRST con la red social Twitter, que exponga sus funcionalidades como servicios, para que estos sean consumidos por el resto de las aplicaciones federadas en la PRST.

En el transcurso de la investigación se utilizan **métodos de investigación científica**, específicamente los métodos teóricos:

Analítico-Sintético:

Este método permitirá analizar y comprender la teoría y documentación relacionada con el tema de investigación, contribuyendo así a extraer los elementos más importantes relacionados con el objeto de estudio. A través de su empleo se definen los conceptos y principios relacionados con la red social Twitter.

Histórico-Lógico: Este método ayuda a entender el surgimiento y la evolución del tema de la investigación, así como, las características actuales, conceptos, términos y vocabularios propios de las redes sociales en especial la red social Twitter.

El presente trabajo consta de **tres capítulos**, estructurados de la siguiente forma:

Capítulo 1. Fundamentación teórica.

En el presente capítulo se abordarán los conceptos fundamentales que permitirán entender las características de los servicios web y de las diferentes formas de interactuar con la red social Twitter. Así como la fundamentación de las herramientas y tecnologías que se emplearán en el desarrollo del trabajo de diploma.

Capítulo 2. Análisis y diseño.

En el presente capítulo se definen los principales requisitos de la propuesta de solución, los casos de uso y el refinamiento del análisis y el diseño. Se describen las clases que se utilizan en la implementación del sistema y las relaciones entre ellas. Se explica además la arquitectura del sistema, haciendo énfasis en los patrones arquitectónicos y de diseño utilizados. Y se realiza el diseño del sistema mediante los distintos artefactos propuestos por la metodología de desarrollo de *software* seleccionada.

Capítulo 3. Implementación y validación.

En el presente capítulo se describen las principales clases del sistema, así como los artefactos correspondientes a la fase de implementación de la metodología seleccionada. También se aplican los tipos de casos de prueba adecuados para validar la aplicación que permiten detectar y corregir los posibles errores a través de los resultados obtenidos en cada una de las iteraciones.

Capítulo 1: Fundamentación teórica

En el desarrollo de este capítulo se conceptualizará todo lo referente a la red social Twitter, se abordarán los conceptos fundamentales que permitirán entender las diferentes formas de interactuar con esta. Se realizará un estudio de las herramientas, metodologías y lenguajes de programación que se van a utilizar en la construcción de la propuesta de solución.

Referentes Teórico-Metodológicos.

Definición de Web 2.0

El término Web 2.0 fue acuñado por el americano Dale Dougherty de la editorial ¨O'Reilly Media¨ durante el desarrollo de una conferencia en el año 2004. El término surgió para referirse a nuevos sitios web que se diferenciaban de los sitios web más tradicionales englobados bajo la denominación Web 1.0. La característica diferencial es la participación colaborativa de los usuarios (Dale Dougherty, 2004).

La Web 2.0 propone una nueva visión de la *World Wide Web*, basada en la compartición de la información, el diseño centrado en el usuario y la colaboración entre los usuarios. Un sitio Web 2.0 es aquel que permite a los usuarios interactuar entre sí y crear contenido dentro de una comunidad virtual, a diferencia de la web tradicional en la que los usuarios se limitan a observar los contenidos (Dale Dougherty, 2004).

Servicios de Web 2.0

La Web 2.0 ofrece una serie de servicios o herramientas diseñadas para la compartición o creación de contenido o la interacción de los usuarios. En líneas generales, los servicios siguientes son ejemplos de los mismos.

- ***Blog*:** Un *blog* es un sitio web que recopila de forma cronológica textos, artículos o determinada información de uno o varios artículos. La información se muestra en el orden en el que se crea apareciendo primero el más reciente (Dale Dougherty, 2004).
- **Redes Sociales:** Las redes sociales son un medio de comunicación utilizado para relacionarse en línea por personas que comparten algún tipo de relación. En los últimos años han alcanzado mucha popularidad y cada día el número de usuarios de estas aumentan considerablemente (Dale Dougherty, 2004).
- **Microblogging**: También conocido como Nanoblogging, es un servicio que permite a sus usuarios publicar mensajes breves (generalmente, solo texto) con un tamaño alrededor de 140 caracteres. Las actualizaciones se muestran vía web, SMS, etc. Estas actualizaciones se muestran en la

página de perfil del usuario y también son enviadas a otros usuarios que tengan la opción de recibirlas (Dale Dougherty, 2004). El mejor ejemplo de este tipo de servicios es el que se usará a lo largo del proyecto por su gran popularidad: Twitter.

Twitter API

Twitter tiene publicada a disposición de sus usuarios su propia API, de forma tal que se puedan crear aplicaciones que se comuniquen con ésta, teniendo en cuenta ciertas restricciones. En el presente apartado, haremos un análisis exhaustivo acerca del API de Twitter y como ésta ayudará en el desarrollo del proyecto.

La API de Twitter tiene tres partes claramente diferenciadas, dos pertenecen al API Rest y la otra corresponde con la API de Streaming. Si bien, podemos dividir la API Res*t* en dos API´s, quedando la API estructurado de la siguiente forma:

La Rest API ofrece a los desarrolladores el acceso al *core* de los datos de Twitter. Todas las operaciones que se pueden hacer vía web son posibles realizarlas desde la API. Esta API permite obtener la colección de los *tweet* más recientes de un usuario determinado, así como un *tweet* determinado que se pasa por parámetro incluyendo el autor del mismo. Permite obtener una colección de *tweet* relevantes que coincidan con una consulta determinada entre otras funcionalidades. Dependiendo de la operación requiere o no autenticación, con el mismo criterio que en el acceso web. Soporta los formatos: *xml, json, rss, atom* (Twitter Development, 2012).

La Search API suministra los *tweets* con una profundidad en el tiempo de 7 días que se ajustan a la consulta solicitada. Es posible filtrar por cliente, lenguaje y localización. No requiere autenticación y los *tweets* se obtienen en formato *json* o *atom*. Ofrece una información más limitada del *tweet,* en concreto sobre los datos del autor en el que solo indica el Id, el nombre del usuario y la URL de su avatar. Las otras dos API´s si ofrecen el perfil completo del autor en el momento de la escritura del *tweet* (Twitter Development, 2012).

El conjunto de las API´s de Streaming que ofrece Twitter proporciona a los desarrolladores acceso de baja latencia de los datos *Tweet*. Se establece una conexión permanente por usuarios con los servidores de Twitter y mediante una petición HTTP se recibe un flujo continuo de *tweets* en formato *json*. Se puede obtener una muestra aleatoria, un filtrado por palabras claves o por usuarios. Sin embargo, los métodos más interesantes son cómo obtener todo el caudal de *tweets* o sólo los *tweets* que tienen enlaces o los *tweet* con *retweet* (Twitter Development, 2012).

Diferencias entre Rest y Streaming API

A diferencia del API Rest y la API Search, la API Streaming necesita mantener una conexión HTTP abierta. Mediante esta conexión, se reciben los tweets que satisfacen las condiciones o filtros, se llevan a cabo los procesos necesarios y se almacenan los resultados.

En los API´s Rest y Search, el proceso funciona de una manera más sencilla. Consideremos, por ejemplo, una web que permite a los usuarios realizar consultas interactuando con la API Rest, esta consulta (o consultas) se realiza a través del servidor HTTP, que manda la petición al API de Twitter. Una vez procesada y obtenido el resultado de la consulta, se envía al usuario.

Soluciones Internacionales

Librerías de Twitter

Desde el punto de vista del desarrollo, Twitter ofrece una serie de librerías para distintos lenguajes de programación con soporte al API de Twitter, lenguajes como Java, C++, *JavaScript,* PHP, *Python* entre otros (Twitter-libraries).

Dentro las librerías implementadas por el lenguaje de programación Java se encuentra la librería Twitter4J una librería que facilita la integración con Twitter, es decir, permite interactuar con la red social para gestionar *tweet,* usuarios, hilos, listas, mensajes directos, relaciones, favoritos, subscripciones, bloqueos, realizar y guardar búsquedas, reportar *spam*, etc. Es una librería completa y sencilla, no requiere ninguna dependencia ni configuración adicional y funciona en cualquier plataforma Java. Todo esto permite el acceso fácil y rápido desde cualquier lugar y el intercambio de opiniones entre usuarios de todo el mundo (Twitter4J, 2007).

Independientemente de las ventajas que pueda propiciar esta librería, sin embargo están muy lejos aún de las expectativas que se han trazado en este campo para CIDI. Por tal motivo se ha decidido implementar el módulo, que comparte ciertas semejanzas con dicha librería ya existente, pero sobretodo, ajustado a los intereses, estructura organizacional, estrategia de trabajo de CIDI. Se pretende lograr con este trabajo proporcionar un módulo que sirva de punto de intercambio de contenidos con Twitter y que funcione como capa de abstracción, exponiendo sus funcionalidades como servicios para que estos sean consumidos por el resto de las aplicaciones federadas de la PRST.

Soluciones Nacionales

En la investigación realizada no se encontró disponible en Cuba, ninguna plataforma nacional que interactúe de manera directa con la red social Twitter.

Metodologías de desarrollo

Las metodologías de desarrollo de *software* se pueden clasificar en: tradicionales y ágiles. Aquellas metodologías centradas en la definición detallada de los procesos, tareas a realizar, herramientas a utilizar y una extensa documentación, reciben el apelativo de metodologías tradicionales o robustas. Este tipo de tecnología es más eficaz en la medida en que el proyecto a realizar sea mayor, pues requiere de una gran organización (Karenny Brito Acuña, 2009).

Las metodologías ágiles hacen énfasis en la comunicación con el cliente y no exigen mucha documentación técnica. La comunicación en las metodologías ágiles se realiza cara a cara entre los miembros del proyecto y los clientes, evitando el trabajo de documentación fijado por las metodologías tradicionales. Están basadas en el trabajo organizado en equipos para la continua revisión y tratamiento de los productos de *software* alcanzados en cada iteración. Por otra parte, ofrecen una buena solución para proyectos de corta duración donde los requisitos no se conocen con exactitud, pues están pensadas para trabajar con incertidumbre (Karenny Brito Acuña, 2009).

De acuerdo con las características antes expuestas, se determina la utilización de una metodología de tipo ágil para el proceso de desarrollo de *software* del módulo de conexión para la red social Twitter para la PRST, destacando las características del equipo de trabajo, y que el tiempo de la construcción de la plataforma es corto. Se valora la selección entre XP *(eXtreme Programming*, por sus siglas en Inglés; Programación Extrema, en Español) y OpenUP (*Open Unified Process,* por sus siglas en Inglés; Proceso Unificado Abierto, en Español); para lo cual se abordan sus características fundamentales.

XP

La Metodología de desarrollo de *software* XP se considera una metodología exitosa utilizada para proyectos de corto plazo y equipos pequeños; ha generado gran interés por su creciente número de casos de éxito en la industria, requiriendo de gran disciplina. Un proceso ligero de bajo riesgo, flexible, predecible, científico y divertido de desarrollar. Su utilidad se mide en valores como la simplicidad, comunicación, realimentación y coraje. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto (Gregorio, 2002).

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo del *software*, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y proporcionando un buen clima de trabajo. Se basa en la retroalimentación continúa entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios (Gregorio, 2002).

OpenUP

OpenUP es una metodología de *software* ágil y unificado, que contiene el conjunto mínimo de prácticas que ayudan a los equipos a ser más eficaces en el desarrollo de *software*. OpenUP abraza una filosofía pragmática y ágil que se centra en la naturaleza colaborativa de desarrollo de *software*. Es un proceso iterativo que es Mínimo, Completo y Extensible que puede utilizarse tal cual o ampliarse para tratar una gran variedad de tipos de proyecto. Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Está organizada dentro de cuatro áreas principales de contenido: Comunicación y Colaboración, Intención, Solución y Administración (Ricardo Bulduino, 2007).

Está organizado en dos dimensiones diferentes pero interrelacionadas: el método y el proceso. El contenido del método es donde los elementos del método (roles, tareas, artefactos y lineamientos) son definidos, sin tener en cuenta como son utilizados en el ciclo de vida del proyecto. El proceso es donde los elementos del método son aplicados de forma ordenada en el tiempo. Muchos ciclos de vida para diferentes proyectos pueden ser creados a partir del mismo conjunto de elementos del método (Ricardo Bulduino, 2007).

Se caracteriza por cuatro principios básicos que se soportan mutuamente:

- Colaboración para alinear los intereses y un entendimiento compartido.
- Balance para confrontar las prioridades (necesidades y costos técnicos) para maximizar el valor para los *stakeholders* (persona o entidad que es afectada por las actividades de una organización).
- Enfoque en articular la arquitectura para facilitar la colaboración técnica, reducir los riesgos y minimizar excesos y trabajo extra.
- Evolución continúa para reducir riesgos y demostrar resultados.

Principales características con que cuenta la metodología OpenUP:

- Metodología de desarrollo de *software* de código abierto diseñado para pequeños equipos organizados, quienes quieren tomar una aproximación ágil del desarrollo.
- Proceso iterativo e incremental que es Mínimo, Completo y Extensible.
- Se valora la colaboración y el aporte de los *stakeholders* sobre los entregables y las formalidades innecesarias.
- Practicantes de desarrollo de *software* (desarrolladores, administradores de proyectos, analistas y probadores) trabajan juntos como un equipo de proyecto.
- No define un modelo de negocio ni de dominio necesario.
- Permite detectar errores tempranos a través de un ciclo iterativo.
- Evita la elaboración de documentación, diagramas e iteraciones innecesarias.
- Por ser una metodología ágil tiene un enfoque centrado al cliente y con iteraciones cortas.

Selección de la metodología:

Se selecciona como metodología a utilizar OpenUP, debido a que proporciona un marco ideal para el desarrollo de proyectos pequeños con escasos recursos y equipos de trabajos mínimos. OpenUP toma las mejores prácticas de RUP, a la vez que provee un conjunto simplificado de artefactos, roles, tareas y guías de trabajo. Perfecciona constantemente su documentación, con el objetivo de obtener retroalimentación y realizar mejoras respectivas mientras que XP deja las actualizaciones para el final del proceso. En lo que a diseño respecta OpenUP establece un desarrollo ágil, iterativo e incremental y XP solo hace correcciones puntuales. Entre los principios que promueve OpenUP se encuentra el mejoramiento continuo centrado en un desarrollo colaborativo del *software*, donde interactúan programador y cliente en cada una de las iteraciones. De esta forma se mitiga la mayor cantidad posible de riesgos en la construcción de la propuesta de solución.

Lenguaje de modelado

Lenguaje Unificado de Modelado (*Unified Modeling Language* dada sus siglas en Inglés UML) se centra en la representación gráfica de un sistema, capturando las partes esenciales del mismo. Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

- Visualizar: Permite expresar de una forma gráfica un sistema de manera que otro lo puede entender.
- Especificar: Permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado y pueden ser utilizados para su futura revisión.

El lenguaje UML tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático, desde el análisis con los casos de uso, el diseño con los diagramas de clases, hasta la implementación y configuración con los diagramas de despliegue. Aporta beneficios al desarrollo de una aplicación, ya que permite realizar una representación gráfica de la misma con un gran rigor en la especificación de sus componentes, lo que proporciona un mayor entendimiento de sus características y facilita su proceso de desarrollo (Rumbaugh, James, 2007).

El UML sirve para visualizar gráficamente sistemas complejos, tanto en el diseño del *software* como de la arquitectura de *hardware* donde se ejecutan, siendo esta la razón fundamental por la que se decide utilizar UML como lenguaje para la representación gráfica y documentación de los artefactos generados en el desarrollo de esta aplicación (Rumbaugh, James, 2007).

Se selecciona como lenguaje de modelado el UML, ya que es capaz de definir, detallar, documentar y construir un sistema de *software*. Es un lenguaje fácil de aprender, adaptable a cambios y ofrece gran variedad de diagramas para visualizar el *software* desde varias vías, facilitando así la relación entre clientes y desarrolladores.

Herramientas CASE

Las herramientas de Ingeniería de *Software* Asistida por Computadoras (*Computer Aided Software Engineering*, por sus siglas en Inglés CASE) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software*, reduciendo el costo de las mismas en términos de tiempo y dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del *software* en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras (Pressman, Roger, 2009).

Visual Paradigm 8.0

Es una herramienta CASE profesional que soporta todo el ciclo de vida del *software*: análisis y diseño orientado a objetos, construcción, prueba y despliegue. Dicha herramienta ayuda a una rápida construcción de la aplicación con alta calidad y a un menor costo. Permite modelar todos los tipos de diagrama de clases, código inverso, generar código desde diagramas y generar documentación (Visual Paradigm for UML, 2010).

La herramienta también proporciona abundantes tutoriales, demostraciones y proyectos UML. Como característica atractiva para los desarrolladores, presenta una interfaz de uso intuitiva y con muchas facilidades a la hora de modelar los diagramas que soportan la Ingeniería de Requerimientos (Visual Paradigm for UML, 2010).

La herramienta está diseñada para una amplia gama de usuarios, incluidos los ingenieros de *software*, analistas de sistemas y analistas de negocio, o para cualquiera que esté interesado en la construcción de forma fiable a gran escala de sistemas de *software*, utilizando un enfoque orientado a objetos. Presenta licencia gratuita y comercial (Visual Paradigm for UML, 2010).

Rational Rose Enterprise Edition.

Rational Rose brinda soporte al modelado con UML y a la vez ofrece distintas perspectivas del sistema. Propone un diseño dirigido por modelos que favorece en productividad a los desarrolladores, admitiendo el lenguaje de modelado UML y técnicas de modelado de objetos.

En la definición de sistemas esta herramienta permite que el equipo de desarrollo entienda mejor el problema, que identifique las necesidades del cliente en forma más efectiva y comunique la solución propuesta de forma más clara. Rational permite completar una gran parte de las disciplinas (flujos fundamentales) de RUP tales como, además de poseer capacidades de ingeniería inversa (Rational Rose, 2007):

- Captura de requisitos (parcialmente).
- Análisis y diseño (completamente).
- Implementación (como ayuda).
- Control de cambios y gestión de configuración (parcialmente).

Selección de la herramienta Case:

Se selecciona como herramienta de modelado el Visual Paradigm (versión 8.0) proporcionando calidad y rapidez a la hora de implementar y desarrollar una aplicación. No se escoge, a pesar de ser muy buena herramienta, Rational Rose pues no contiene el ciclo de vida completo del desarrollo del *software* y es admitida solo por el sistema operativo Windows. Visual Paradigm por otro lado, contiene todos los tipos de diagramas de clases y cuenta además con abundante documentación. Soporta el ciclo de vida completo del desarrollo del *software* ya sea análisis y diseño, construcción, pruebas y despliegue.

Lenguajes de Programación

Es aquella estructura que, con una cierta base sintáctica y semántica, imparte distintas instrucciones a un programa de computadora (Lenguaje de programación, 2008)**.**

Java

Es un lenguaje de programación orientado a objetos, ya que al estar agrupados en estructuras encapsuladas es más fácil su manipulación. Permite abrir, establecer y aceptar conexiones con los servidores o clientes remotos; facilita la creación de aplicaciones distribuidas ya que proporciona una colección de clases para aplicaciones en red. Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los *bytecodes*, semejantes a las instrucciones de ensamblador. Además es altamente fiable en comparación con C, ya que se han eliminado muchas características con la aritmética de punteros, proporciona numerosas comprobaciones en compilación y en el tiempo de ejecución. Por otra parte, es interpretado, ya que los *bytecodes* se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (Lenguaje de programacion, 2007).

Python

Python es un lenguaje de programación interpretado, orientado a objetos, multiplataforma y de sintaxis sencilla. Permite dividir un programa en módulos reutilizables desde otros programas en Python. El lenguaje incorpora una gran colección de módulos estándar que se pueden utilizar como base de los programas (o como ejemplos para empezar a aprender Python). También hay módulos incluidos que proporcionan E/S de ficheros, llamadas al sistema, "*sockets*" y hasta interfaces gráficas con el usuario. Python permite escribir programas muy compactos y legibles. Los programas escritos en *Python* son normalmente mucho más cortos que sus equivalentes en C o C++, es ampliable o conocido como lenguaje Integrador (Python, 2007).

PHP

Es un lenguaje de programación utilizado para la creación de sitios web, es interpretado del lado servidor, utilizado para la generación de páginas web dinámicas, embebidas en páginas HTML y ejecutadas en el servidor. No necesita ser compilado para ejecutarse. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas. PHP está diseñado específicamente para ser un lenguaje más seguro, y con la selección correcta de opciones de configuración en tiempos de compilación y ejecución y siguiendo algunas prácticas correctas de programación. Es multiplataforma y puede ser utilizado sobre los sistemas operativos: *GNU/Linux, Windows*, entre otros (Granado, 2004).

Selección del lenguaje de programación.

Se selecciona como lenguaje de programación Python ya que proporciona un buen equilibrio entre lo práctico y lo conceptual, puesto que Python es un lenguaje de programación interpretado, orientado a objetos, multiplataforma y de sintaxis sencilla. Permite además escribir programas muy compactos y legibles, normalmente mucho más cortos. No se escoge PHP, a pesar de su calidad, ya que es un lenguaje muy lento en sitios web que tendrán muchas peticiones por segundo o cargas muy pesadas de acceso a la base de datos. No se utiliza el lenguaje de programación java pues su curva de aprendizaje es muy pesada, es muy complejo respecto a Python, lo cual no lo hace justificable para un desarrollo medianamente simple y haciendo que la ejecución de las aplicaciones sea muy lenta.

Tecnologías a utilizar:

Django es un *framework*[2] web de código abierto escrito en Python, que permite construir aplicaciones web más rápido y con menos código. Fue inicialmente desarrollado para gestionar aplicaciones web de páginas orientadas a noticias de *word online*, más tarde se liberó bajo licencia BSD[3]. Django se centra en

[2] Es una estructura de soporte definida, mediante la cual otro proyecto de *software* puede ser organizado y desarrollado.

[3] Distribución de *Software* Berkeley, es una licencia de *software* libre.

automatizar todo lo posible y se adhiere al principio DRY (*Don't Repeat Yourself*) (djangoproject.com, 2013).

Django está fuertemente inspirado en la filosofía de desarrollo Modelo Vista Controlador, sus desarrolladores declaran públicamente que no se sienten especialmente atados a observar estrictamente ningún paradigma particular y en cambio prefieren hacer "lo que les parece correcto". Gracias al poder de las capas *mediator* y *foundation,* Django permite que los desarrolladores se dediquen a construir los objetos *Entity* y la lógica de presentación y control para ellos (djangoproject.com, 2013).

Se trabajará además con Celery también escrito en Python, este es una cola de tareas o trabajos asincrónicos basados en paso de mensajes distribuidos. Se centra en la operación en tiempo real y es muy fácil de integrar con los *frameworks* web. Las unidades de ejecución, llamadas tareas, se ejecutan simultáneamente en un único o varios servidores de trabajo utilizando el multiprocesamiento, se utiliza en los sistemas de producción para procesar millones de tareas de un día (Celery: Distributed Task Queue, 2011).

RabbitMQ.

En la implementación del sistema que se expone en la presente investigación se utilizará RabbitMQ, un servidor de paso de mensajes, el cual permitirá distribuir las distintas funcionalidades del módulo entre diversos nodos, de modo que el sistema obtenido pueda escalar a una mayor cantidad de usuarios. RabbitMQ Implementa el estándar *Advanced Message Queuing Protocol* (AMQP dadas sus siglas en Inglés). El servidor RabbitMQ está desarrollado utilizando el lenguaje de programación Erlang y el *framework* Open Telecom *Platform* (dadas sus siglas en Inglés OTP) para construir sus capacidades de ejecución distribuida y conmutación ante errores. *Rabbit Technologies Ltd*., la compañía que lo desarrolla, fue adquirida en abril de 2010 por la división *SpringSource* de VMWare. A partir de este momento, es esta última compañía la que desarrolla y da soporte para RabbitMQ. El código fuente está liberado bajo la licencia *Mozilla Public License* (RabbitMQ, 2010).

Conexión con la red social Twitter

Para la interacción con las API´s de Twitter el Módulo de conexión a la red social Twitter para la PRST utilizará la librería Twython, una librería escrita en Python que servirá de nexo de unión entre la aplicación y la Rest API de Twitter, facilitando el proceso de consulta y de gestión de la información del servicio de la red social, de forma transparente para la aplicación.

Utilización de la interfaz del Django Rest Framework

En la presente investigación se hará uso de la interfaz del Django Rest Framework como ayuda para validar los requisitos funcionales de la aplicación, pues la misma no cuenta con una interfaz web, y para demostrar que cada requisito funcional cumple con el objetivo propuesto; se hizo necesario utilizar esta interfaz. Se utilizará además como interfaz para realizar los diagramas del diseño, es decir, demostrar como se realizará la interacción del actor con las funcionalidades y para realizar las pruebas funcionales al sistema.

Entorno de Desarrollo Integrado (IDE)

Entorno de Desarrollo Integrado, son un conjunto de herramientas para el programador, que suelen incluir en una misma suite, un buen editor de código, administrador de proyectos y archivos, enlace transparente a compiladores y *debuggers*[4] integración con sistemas controladores de versiones o repositorios (Luciano, 2009).

Netbeans

Es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Es un producto libre y gratuito sin restricciones de uso. La plataforma permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de *software* llamados módulos. Estos a su vez pueden ser desarrollados independientemente, razón por la que las aplicaciones basadas en Netbeans pueden ser extendidas fácilmente por otros desarrolladores de *software*. Con Netbeans se puede desarrollar aplicaciones de escritorio, web*, mobile, enterprise,* con lenguajes de programación como Java, C/C++ (Netbeans.org, 2012).

Eclipse

El Eclipse es un entorno de desarrollo integrado (*Integrated Development Environment*, dada sus siglas en Inglés IDE) que facilita enormemente las tareas de edición, compilación y ejecución de programas durante su fase de desarrollo. Aunque Eclipse pretende ser un entorno versátil soportando varios lenguajes de programación, es con el lenguaje Java con el que mejor se integra y con el que ha ganado su popularidad. Es una aplicación gratuita y de código abierto, disponible en la red para su descarga e incluida ya en muchas distribuciones de Linux (Eclipse, 2010).

[4]Depuradores.

Es un desarrollo de IBM[5] cuyo código fuente fue puesto a disposición de los usuarios. En sí mismo Eclipse es un marco y un conjunto de servicios para construir un entorno de desarrollo a partir de componentes conectados (*plug-ins*) (Eclipse, 2010).

Spyder

Spyder es un entorno de desarrollo de código libre para el lenguaje de programación Python que provee características similares a las de MATLAB en un programa simple y ligero. Se encuentra disponible para todas las plataformas *(Windows*, *Linux)* y cuenta con características de edición avanzada, pruebas interactivas, depuración e introspección (Spyder, 2012).

Spyder tiene características bastante interesantes, lo que permite que a pesar de su sencillez sea bastante potente, su editor cuenta con autocompletado de código, resaltado de ocurrencias, muestra de errores, navegador de clases y funciones (Spyder, 2012).

También cuenta con una consola Python interna donde se puede interactuar y probar funciones antes de implementarlas en el código, todas ejecutándose en procesos separados para evitar conflictos. Otras características interesantes con las que cuenta son sin duda el explorador de variables, el inspector de objetos, el historial de la consola interactiva, entre muchas otras.

Selección del IDE

Para el desarrollo de la aplicación objetivo de esta tesis, se ha escogido como IDE de desarrollo Spyder ya que está sustentado por ser gratuito, libre y potente. Proporciona todas las herramientas y funciones necesarias para realizar el presente trabajo, recogidas además en una atractiva interfaz que lo hace fácil y agradable de usar. La integración con el lenguaje y *framework* seleccionados lo convierten en la selección más certera para este tipo de herramienta.

Sistema Gestor de Base de Datos

Los sistemas de gestión de bases de datos (SGBD) son un tipo de *software* dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. El propósito general de estos sistemas es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización (PostgreSQL-es).

Por su parte, los almacenes de datos pueden ser relacionales y no relacionales, a estos últimos también se les conoce comúnmente como bases de datos No SQL (PostgreSQL-es).

[5] (*International Business Machines*). Empresa que fabrica y comercializa hardware, *software* y servicios relacionados con la informática.

Las bases de datos relacionales cumplen con el modelo relacional, el cual es el modelo más utilizado en la actualidad para implementar bases de datos ya planificadas. Permiten establecer conexiones entre los datos que están guardados en tablas y a través de dichas conexiones relacionar los datos de ambas tablas; de ahí proviene su nombre: "Modelo Relacional" (PostgreSQL-es).

Por otro lado, los almacenes de datos no relacionales no proporcionan garantías ACID[6]. Normalmente no tienen esquemas fijos de tablas ni sentencias *join*[7]. Estos sistemas responden a las necesidades de escalabilidad horizontal que tienen cada vez más empresas (PostgreSQL-es).

Dentro de los sistemas de gestión de bases de datos relacionales usados a nivel internacional se encuentran: PostgreSQL, MySQL, ORALE *y* SQL-Server.

Actualmente los almacenes de datos No SQL que más sobresalen son: MongoDB, Cassandra *y* BigTable, todos usados en aplicaciones que manejan grandes volúmenes de datos como son: Twitter, Facebook y Google respectivamente.

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (postgresql.org.es, 2012).

MongoDB

Un estudio enfocado en las base de datos no relacionales arroja que MongoDB constituye una solución escalable y de alto rendimiento de almacenes de datos No SQL. Es de código abierto y escrito en C++, este sistema tiene orientado el almacenamiento de datos en documentos al estilo *json* con esquemas dinámicos, que ofrecen potencia y simplicidad. Se destaca por conservar los índices de todos los atributos y hacer mucho más flexible la agregación y procesamiento de datos (PostgreSQL-es).

Esto tiene ventajas y desventajas, sin embargo, existe un sesgo en el esquema, que puede hacer ciertas consultas un poco menos elegante que en el esquema relacional. Las colecciones en MongoDB son análogas a las tablas en una base de datos relacional. Cada colección contiene varios documentos. Como se mencionó anteriormente estos documentos pueden ser grandes. No requiere que en una colección

[6] Conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción.
[7] Sentencia para relacionar tablas.

todos los documentos tengan la misma estructura, sin embargo, en la práctica, la mayoría de las colecciones son altamente homogéneas (MongoDB, 2011).

Selección del sistema gestor de bases de datos

Dada las características planteadas se selecciona MongoDB para el almacenamiento de los datos relacionados con el módulo de conexión a la red social Twitter para la PRST. La selección de este sistema de almacenamiento de datos se justifica porque logra un mejor manejo de la información frente a un volúmen considerable de datos, ya que es necesario manejar una gran cantidad de documentos y metadatos asociados a los mismos.

Herramientas de validación

JMeter

Jmeter 2.9 es una herramienta de código abierto implementada en Java, que permite realizar pruebas de comportamiento funcional y medir el rendimiento del *software*. También se puede utilizar para realizar pruebas de estrés, por ejemplo, en un servidor, y poner a prueba su rendimiento. Permite además generar un plan de pruebas en forma de fichero con extensión .jmx que lleva implícito cada uno de los datos utilizados durante la prueba (Jmeter, 2011)

Conclusiones del Capítulo

Durante la realización de este capítulo se plantearon conceptos fundamentales relacionados con la investigación, así como el estudio de las diferentes temáticas, contribuyendo al enriquecimiento de la misma. Se define la metodología que se utilizará para la realización del *software* aportando los conocimientos necesarios para lograr el desarrollo de la aplicación. Además, se describen las tecnologías y herramientas necesarias para la elaboración del *software*, exponiendo su importancia para la realización del sistema.

Capítulo 2: Características y diseño del Sistema

Este capítulo tiene como propósito presentar la propuesta de solución del módulo de conexión a la red social Twitter para la plataforma red social Twitter. En el mismo se definen las clases y el modelo de dominio, así como los requerimientos funcionales y no funcionales que debe complementar el sistema para su aplicación. Además, se identifican y describen los actores, los casos de uso del Sistema, los patrones de arquitectura y diseño, diagramas de diseño y diagrama de despliegue; utilizando la metodología de desarrollo OpenUP junto al lenguaje de modelado UML.

Especificación de los requisitos del sistema

El proceso de captura de requisitos, da inicio a la interacción con el cliente. Durante este proceso se hace una relación de las necesidades que tiene el usuario y se define lo que debe hacer el sistema. Se describen las condiciones que se necesitan para que los requisitos funcionales puedan ser cumplidos y se detallan los usuarios que van a interactuar con la aplicación.

Requisitos funcionales

Un requisito funcional define el comportamiento interno del *software*: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica. El núcleo del requisito es la descripción del comportamiento requerido, que debe ser clara y concisa (Sommerville, Ian, 2006)

Prioridad de cliente	Cantidad de requisitos funcionales
Alta	21
Media	5
Baja	0
Total	26

Tabla 1: Requisitos funcionales del sistema.

Listado de requisitos funcionales:

RF 1: Devolver los *tweets* más recientes de un usuario.

RF 2: Devolver los 100 *retweets* más recientes de Twitter especificado por un identificador determinado.

RF 3: Devolver un *tweet* con la información de sus *retweets.*

RF 4: Crear un *tweet*.

RF 5: Devolver una lista de usuarios que han retweetado un *tweet* especificado por un identificador.

RF 6: Buscar un *tweet.*

RF 7: Devolver una colección de Id de usuarios de los amigos de un usuario determinado.

RF 8: Devolver las relaciones entre los usuarios autenticados.

RF 9: Permitir seguir a un usuario especificado por el identificador.

RF 10: Permitir dejar de seguir a un usuario especificado por un parámetro identificador.

RF 11: Devolver la información acerca de la relación entre dos usuarios.

RF 12: Devolver una colección de objetos de usuarios para cada usuario que tiene como amigo.

RF 13: Devolver una colección de objetos de usuario para los usuarios que siguen a un usuario en específico.

RF 14: Retornar las características de la cuenta de un usuario.

RF 15: Verificar las credenciales de un usuario.

RF 16: Devolver información de un usuario especificado por el parámetro id.

RF 17: Búsqueda simple de usuarios públicos.

RF 18: Devolver los últimos 20 *tweet* favoritos del usuario.

RF 19: Devolver una línea de tiempo de *tweet.*

RF 20: Comprobar si un usuario específico es miembro de una lista específica.

RF 21: Devolver los miembros de una lista específica.

RF 22: Devolver información de un lugar conocido.

RF 23: Devolver los 10 mejores temas de tendencias para un WOEID específica.

RF 24: Devolver tendencias de información de un lugar disponible.

RF 25: Dado un lugar determinado devolver información del lugar más cercano a él.

RF 26: Reportar a un usuario como cuenta *Spam* a Twitter.

Requisitos no Funcionales

Los requerimientos no funcionales especifican cualidades, propiedades del sistema; como restricciones del entorno o de la implementación, rendimiento, dependencias de la plataforma, facilidad de mantenimiento, extensibilidad y fiabilidad (Rumbaugh, James, 2007).

Listado de requisitos no funcionales:

Eficiencia

RnF 1. Tiempo de respuesta por transacción. El sistema debe ser capaz de responder en un tiempo máximo de 2 segundos a las peticiones de los usuarios.

RnF 2. Cantidad de peticiones concurrentes: El sistema debe ser capaz de soportar un mínimo de 50 peticiones concurrentes.

Seguridad

RnF 3. El sistema debe tener protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

Hardware

RnF 4. Para que el servidor realice todas las funcionalidades especificadas se requiere:

- 2GB de memoria RAM como mínimo.
- Procesador familia INTEL u otro con una velocidad mínima de 2.10GHz.
- Espacio en disco duro de 80GB.

RnF 5. Para el acceso a la aplicación la estación de trabajo debe contar como mínimo con:

- 256MB de memoria RAM.
- Procesador familia INTEL u otro con una velocidad mínima de 1.10GHz.
- Espacio en disco duro de 1GB.

Modelo de Dominio

El modelo del dominio es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Facilita en gran medida la comprensión del problema mediante un vocabulario común a los usuarios, clientes, desarrolladores e interesados en el desarrollo del sistema. Muestra la relación entre los principales conceptos y contribuye a una correcta y eficiente captura de requisitos (Jacobson, 2000).

Diagrama de Clases del Modelo del Dominio

Para entender la estructura organizacional que se modela se realiza el siguiente modelo del dominio (ver Ilustración 1). El mismo relaciona clases usando el lenguaje UML para observar las relaciones entre entidades, actores y comprender el flujo de trabajo que se evidencia en el negocio que se modela.

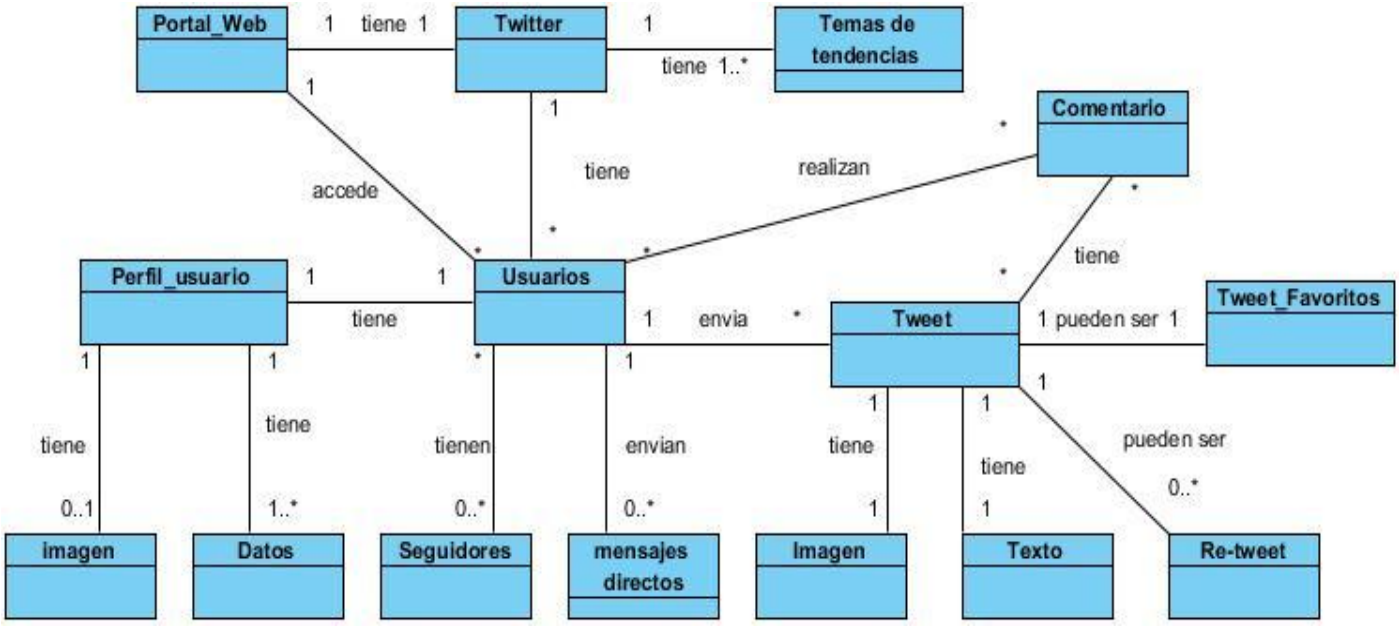


Ilustración 1: Modelo del Dominio

Descripción de Clases del Modelo del Dominio.

Conceptos del modelo de dominio

- Twitter**:** Módulo de la red social Twitter.
- Portal _Web**:** Portal donde el usuario accede para ver los contenidos.
- Usuarios: Entidad que accede a los servicios de la plataforma.
- Perfil de usuarios: Recopilación de datos base de los usuarios.
- *Tweets*: Mensajes enviados entre los usuarios de la plataforma Twitter.
- Tweet_Favoritos: *Tweets* marcados como favoritos de un usuario.
- Texto: Es el texto que contiene un *tweet.*
- Mensajes directos: Enviar un mensaje privado a otro usuario, de forma que solo pueda leerlo el usuario destinatario.
- Imagen: Carga de imagen o contenidos realizadas por los usuarios.
- Datos: Información personal de los usuarios.
- Comentario: Comentarios realizados por los usuarios a los contenidos.

- Seguidores: Es un listado de personas que siguen a un usuario en específico.
- Temas de tendencias: Son temas de interés de Twitter.
- *Retweet*: Reenviar (retwittear) un *tweet.*

Descripción del sistema a desarrollar

El sistema a desarrollar tiene como objetivo fundamental implementar una aplicación, que sirva como capa de abstracción para la comunicación de la PRST con la red social Twitter, que exponga sus funcionalidades como servicios para que estos sean consumidos por el resto de las aplicaciones federadas de la PRST, debe ser capaz de interactuar con los servicios de la red social Twitter de tal forma que se eviten redundancias, y se aprovechen mejor los recursos humanos y tecnológicos.

Modelo de casos de uso del Sistema

El Modelo de Casos de Uso del Sistema especifica los actores con quien interactúa la aplicación, los casos de uso del sistema y las relaciones que existen entre ellos. Representa un esquema donde se recopilan las funcionalidades a automatizar y se determina cómo será utilizado desde el punto de vista del usuario (actor), pues se construye sobre la base de sus necesidades (Jacobson, 2000).

Actores del Sistema

Los actores representan a los usuarios que intercambian información con el sistema (Jacobson, 2000).

Actor del sistema	Descripción
Sistema Externo	Sistemas o aplicaciones federadas de la PRST que gestionan información adquirida de Twitter y hacen solicitudes de servicios.

Tabla 2: Descripción de los actores del sistema

Diagrama de Casos de Uso del Sistema

Para una mejor comprensión de la solución, se muestra a continuación un diagrama con los casos de uso del sistema (ver Figura 2 Diagrama de casos de uso del módulo de conexión a la red social Twitter para la PRST).

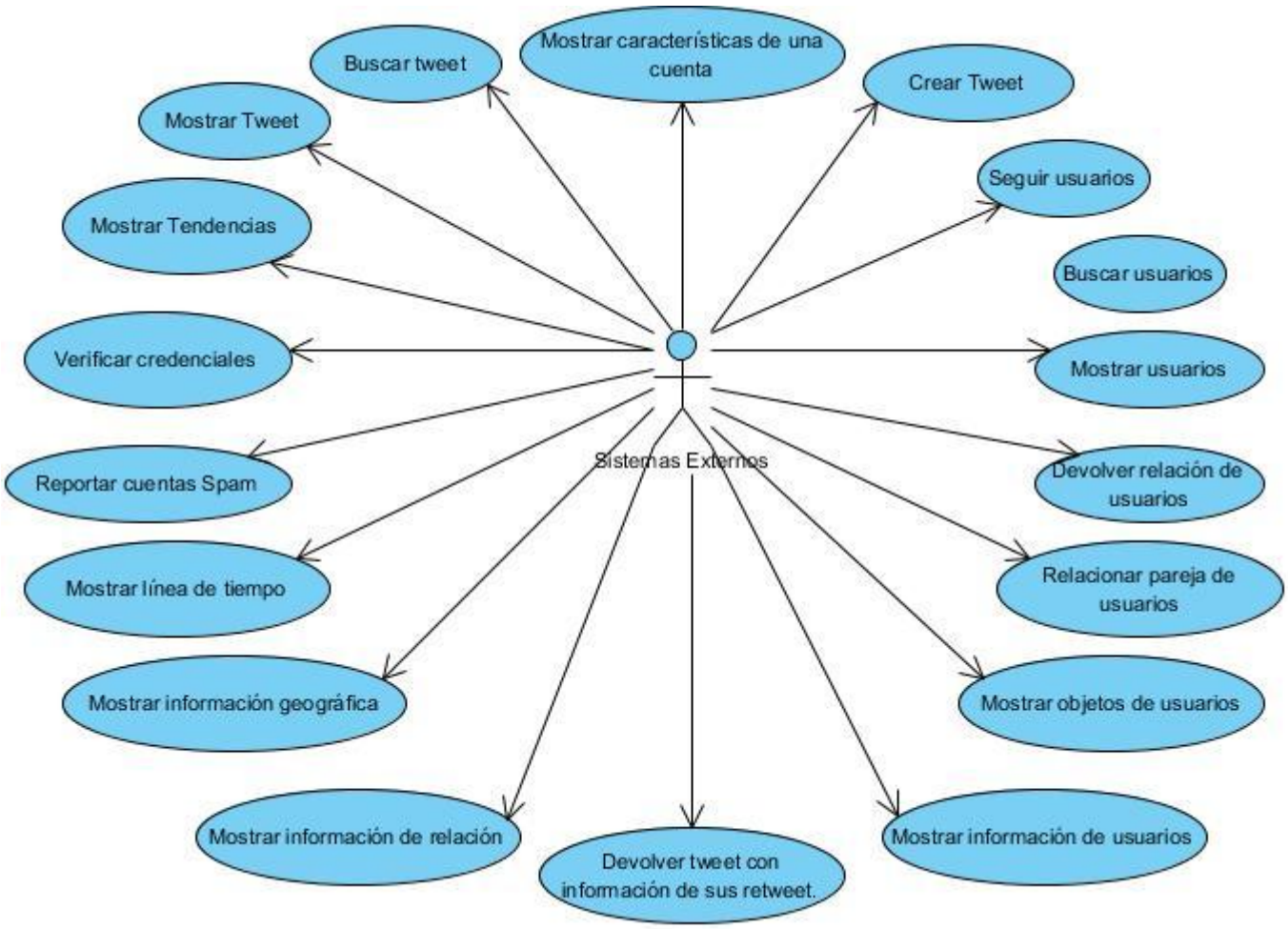


Ilustración 2: Diagrama de casos de usos del sistema.

Descripción de Casos de Uso del Sistema

Al utilizar la metodología de desarrollo de *software* OpenUP, los casos de uso del sistema son descritos en el artefacto Especificación de Casos de Uso. En el siguiente epígrafe se describen algunos que son de prioridad alta para el cliente, en el desarrollo del módulo de conexión de redes sociales Twitter para la PRST (Tabla 3).

CU_Crear*Tweet*.

Objetivo	Crear *tweet*
Actores	Sistema Externo.
Resumen	El CU se inicia cuando una de las aplicaciones federadas de la PRST solicita el servicio mediante la URL correspondiente a Crear *tweet*. El caso de uso termina cuando el sistema crea y publica dicho *tweet.*
Complejidad	Alta
Prioridad	Alta

Precondiciones	Que se hayan verificado las credenciales del usuario.	
Postcondiciones	Queda publicado el *tweet* en la interfaz de la plataforma de Twitter.	
Requisitos funcionales	RF 4	
Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	Sistema
1	El usuario mediante una URL realiza una petición al servicio Crear *tweet* mediante su URL correspondiente y le pasa a esa petición los parámetros en formato *json.*	El sistema valida los datos para verificar que sean correctos.
2		El sistema valida las credenciales del usuario.
3		El sistema luego de haber validado las credenciales del usuario sin son válidas crea el *tweet* en la interfaz web de Twitter. Termina el caso de uso.
Sección 1: Credenciales no validas		
Flujo básico <Nombre del flujo básico>		
	Actor	Sistema
1.		Si las credenciales del usuario son incorrectas el sistema muestra un mensaje indicando que las “credenciales no válidas”.
Flujos alternos		
Nº Evento “Los datos proporcionados por el usuario no son correctos.”		
	Actor	Sistema
		Muestra un mensaje de error “Datos incorrectos.”
Relaciones		
Requisitos no funcionales		
Asuntos pendientes		

Tabla 3: Descripción del caso de uso Crear *Tweet*

CU_Mostrar información de un usuario.

Objetivo	Mostrar información de un usuario.
Actores	Sistema Externo
Resumen	El CU se inicia cuando una de las aplicaciones de la PRST solicita el servicio mediante la URL Mostrar información de un usuario.
Complejidad	Alta
Prioridad	Alta
Precondiciones	
Postcondiciones	Se mostró la información solicitada por el usuario.
Requisitos	RF16

funcionales		
Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	Sistema
	El usuario realiza la petición de servicios Mostrar información de usuarios mediante una URL.	
	El usuario envía en formato *json* los parámetros que lleva esa petición	El sistema captura los datos enviados por el usuario lo valida y devuelve una respuesta de acuerdo a los parámetros enviados en formato *json*.
		Termina el CU.
Flujos alternos		
Flujo básico < Los datos proporcionados por el usuario no son correctos>		
	Actor	Sistema
		Muestra un mensaje de error “Datos incorrectos.”
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	Sistema
1.		
Relaciones	CU Incluidos	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes		

Tabla 4: Descripción del caso de uso Mostrar información de un usuario

Las restantes descripciones de casos de uso se encuentran en los anexos.

Modelo de diseño

El modelo de diseño es una abstracción de la implementación del sistema. Se utiliza para concebir y para documentar el diseño del sistema de *software*. Es un producto de trabajo integral y compuesto que abarca todas las clases de diseño, subsistema, paquetes, colaboraciones y las relaciones entre ellos (Larman, Craig, 2000).

Diagrama de clases del diseño

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de *software* y de las interfaces de una aplicación. Normalmente contienen información acerca de las clases, asociaciones, atributos, métodos y dependencias (Larman, Craig, 2000).

Descripción del diagrama de clases del diseño Crear_*Tweet.*

Para crear un *tweet* el usuario hace una petición mediante la URL *api/update_user_status,* donde se le pasan los parámetros necesarios para ejecutar el servicio solicitado, en este caso el usuario de Twitter y el texto del *tweet*, la clase *View* captura los datos y manda a ejecutar en la clase *Task* que es donde están programados todos los servicios del módulo de conexión que ejecute la tarea *update_user_status_task*. Esta clase usa la librería Twython para conectarse con la API de Twitter para realizar la petición. Pero para que el servicio sea completado con éxito en este caso es necesario verificar las credenciales del usuario por lo que la clase "*Services*" establece una conexión con la base de datos de *PostgreeSQL* para verificar las credenciales del usuario y se conecta a la base de datos creada en MongoDB para guardar la respuesta dada por la API de Twitter y ser enviada al usuario que hizo la petición.

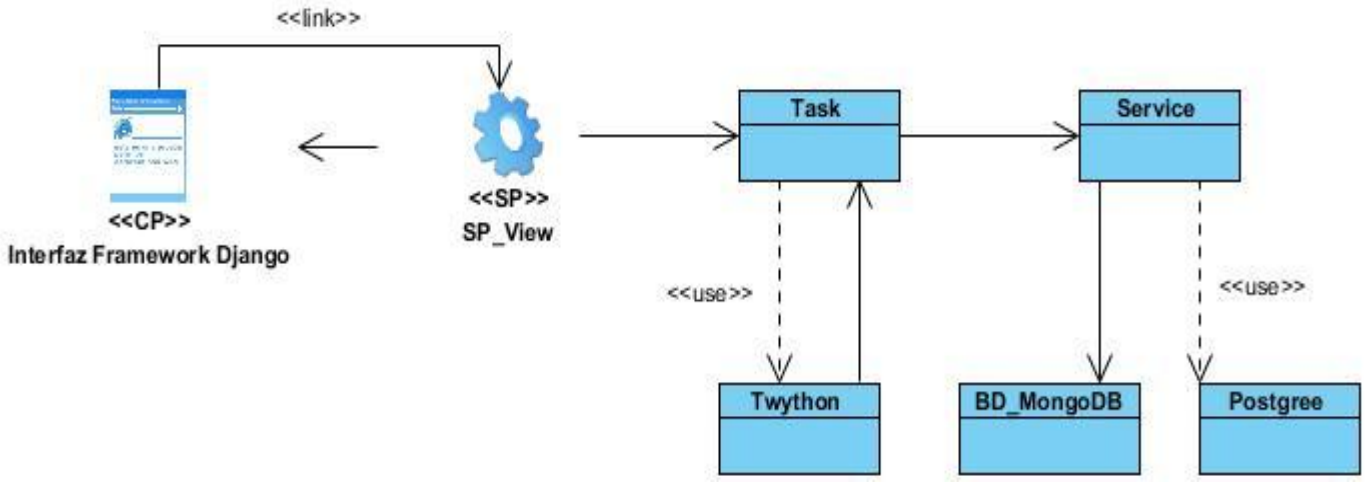


Ilustración 3: Diagrama de clases del diseño para el CU_Crear_*Tweet*

Descripción del diagrama de clases del diseño Mostrar Información de un usuario.

Para mostrar información de un usuario, el actor que interactúa con el sistema hace una petición mediante la URL *api/account_verify_credentials*, donde se le pasa los parámetros necesarios para ejecutar el servicio solicitado, (en este caso el id de usuario o el nombre), la clase View captura los datos y manda a ejecutar en la clase *Task* que es donde están programados todos los servicios del módulo de conexión que ejecuta la tarea. Esta clase usa la librería Twython para conectarse con la API de Twitter para realizar la petición. En este caso de uso se le da respuesta al usuario sin tener que verificar las credenciales por lo que la clase "*Service*" se conecta a la base de dato creada en MongoDB para guardar la respuesta dada por la API de Twitter y ser enviada al usuario que hizo la petición.

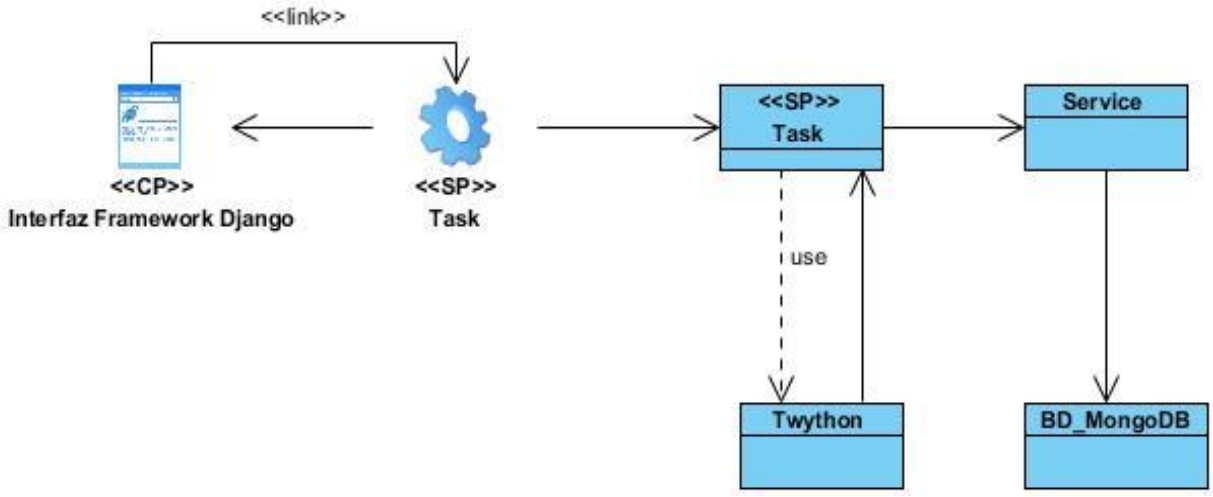


Ilustración 4: Diagrama de clases del diseño para el CU_Mostrar información de un usuario.

Diagrama de secuencia

Los diagramas de secuencia revelan la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela por cada sección de la descripción de los casos de uso. El diagrama de secuencia contiene los detalles de implementación del escenario, incluyendo los objetos y clases que se aplican para implementar el escenario, además de los mensajes entre los objetos (Larman, Craig, 2000).

El diagrama de secuencia está constituido por la línea de vida que representa la existencia de un objeto a lo largo de un período de tiempo. El foco de control que es el rectángulo delgado que representa el período de tiempo durante el cual el objeto ejecuta una acción. Para la realización de los casos de uso en el diseño se utilizan los diagramas de secuencia debido a que representan el flujo de acciones con mayor claridad (Larman, Craig, 2000).

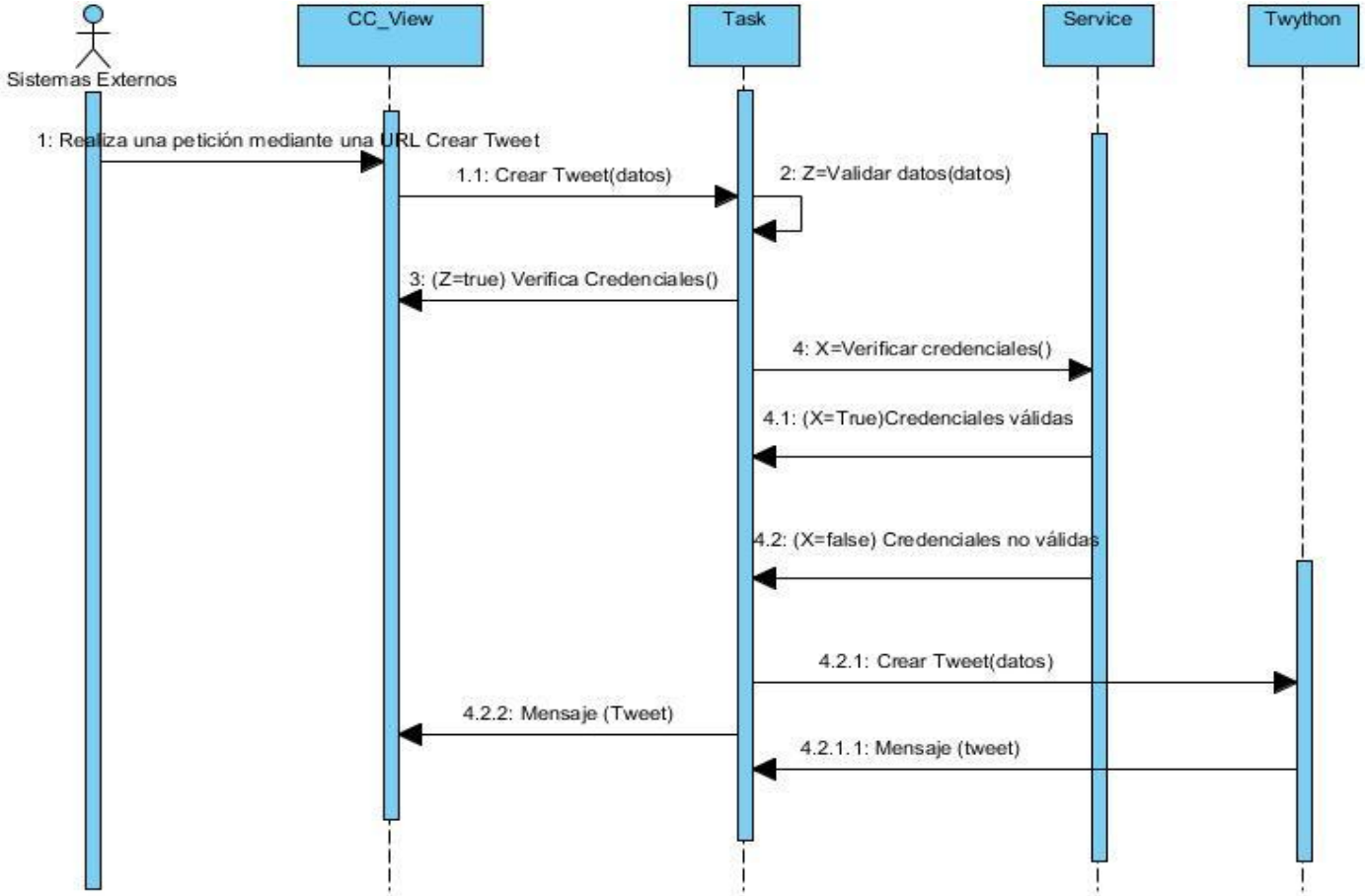


Ilustración 5: Diagrama de secuencia del caso de uso Crear Tweet.

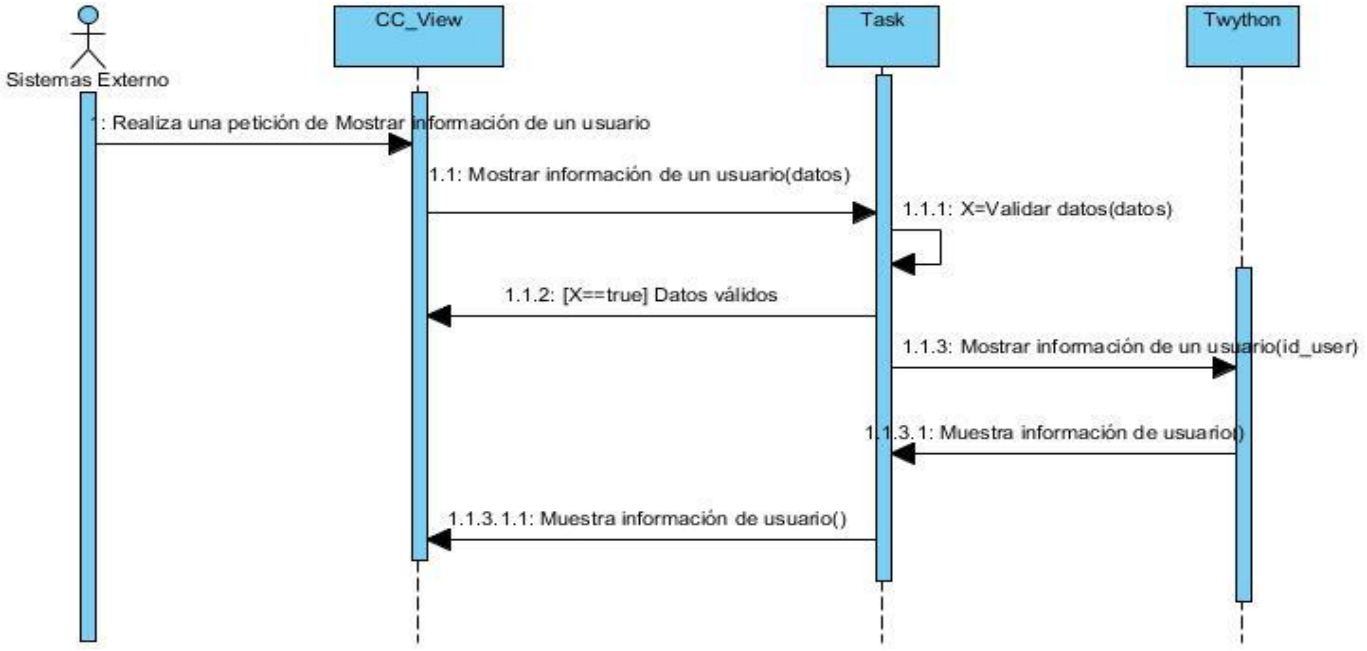


Ilustración 6: Diagrama de secuencia del caso de uso Mostrar información de un usuario.

Patrones de diseño

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia. En el desarrollo de multitud de aplicaciones hay problemas de diseños que se repiten o que son análogos, es decir, que responden a un cierto patrón. Con el uso de patrones los diseños serán mucho más flexibles, modulares y reutilizables. Estos han evolucionado el diseño orientado a objetos, todo buen arquitecto de *software* debe conocerlos (Larman, Craig, 2000).

Patrones de diseño utilizados:

Patrones GRASP

Bajo acoplamiento: El acoplamiento es la medida de fuerza en que un elemento tiene conocimiento de otros elementos. El bajo acoplamiento es un principio que asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Significa que debe haber pocas dependencias entre las clases para ser más entendidas cuando estén aisladas porque no tendrían tantas dependencias de otras clases (Larman, Craig, 2000).En la aplicación este patrón se evidencia en las clases del negocio, ya que cada una de ellas tiene la menor dependencia posible.

Alta cohesión: La cohesión es una medida de cuán relacionadas están las responsabilidades de una clase. Una alta cohesión permite a las clases que están muy relacionadas no realizar un enorme trabajo. Las clases que presentan baja cohesión son difíciles de comprender, reutilizar y conservar (Larman, Craig, 2000).En la aplicación cada clase tiene bien definidas sus responsabilidades, este patrón se hace evidente en la clase *Task* ya que esta hace uso de los métodos de la clase *Service* para evitar la sobrecarga de funcionalidades.

Experto: La responsabilidad de realizar una labor es de la clase que tiene o puede tener la información necesaria (atributos). Se le asigna la responsabilidad al experto en información. Esto garantiza que el sistema sea más fácil de entender (Larman, Craig, 2000). En la aplicación este patrón se ve reflejado en las clases del negocio que cada una tiene una función específica.

Creador: Este patrón es el que crea, el que guía la asignación de responsabilidades relacionadas con la creación de objetos. Una clase B tiene la responsabilidad de crear un objeto de una clase A cuando la clase está contenida en la clase B, la clase B en una agregación o composición de la clase A, la clase B almacena a la clase A, la clase B inicializa los datos de la clase A y la clase B usa la clase A (Larman, Craig, 2000). En la presente investigación la clase View hace uso de las implementaciones de la clase *Task*, a esta accede mediante su respectiva instancia.

Patrones GOF

***Facade*/Fachada**:

Proporciona una interfaz unificada de alto nivel que agrupa las funcionalidades de un subsistema facilitando su uso. La fachada sabe qué clase es responsable de cada petición, pero es completamente transparente para ellas. En el sistema cada una de las funcionalidades acceden a la vista que proporciona el *framework* Django que se usó para brindar los servicios en este caso *Django Rest Framework* que proporciona una interfaz en la cual se pueden obtener las respuestas de cada uno de los servicios del sistema.

Descripción de la arquitectura

Para el desarrollo del módulo de conexión a la red social Twitter para la PRST se decidió emplear una arquitectura de *software* de tres capas, separando la lógica de negocios de la lógica de diseño. La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y en caso que sobrevenga algún cambio solo se corrige al nivel requerido sin tener que revisar todo el código. La clave de una aplicación en N capas está en la gestión de dependencias: los componentes de una capa pueden interactuar solo con componentes de la misma capa o bien con otros componentes de capas inferiores, lo cual ayuda a reducir las dependencias entre componentes de diferentes niveles (De La Torre, C y otros, 2010).

En una arquitectura en N capas, cada capa usa los servicios brindados por la capa inferior y expone servicios a la capa superior, además dicho estilo favorece la abstracción entre las capas, minimiza las dependencias y permite que se pueden reemplazar fácilmente. Para la utilización del estilo arquitectura en capas es importante decidir qué capas tener y la responsabilidad de cada una (De La Torre, C y otros, 2010).

La capa de presentación o también conocida como capa de usuario es la responsable de comunicar y capturar información al usuario e interpretar sus acciones. En esta capa se atenderán las peticiones de los usuarios que hagan uso de los servicios que estarán expuestos por el módulo de conexión. El uso de Django como *framework* y del módulo *Django rest framework* para brindar las funcionalidades del módulo como servicios *Rest* permite tener una vista genérica proporcionada por el *framework* que garantiza visualizar la respuesta de cada uno de los servicios implementados (De La Torre, C y otros, 2010).

La capa de lógica del negocio o también conocida como lógica del dominio es la responsable de representar conceptos de negocio, información sobre la situación de los procesos de negocio e implementación de las reglas del dominio. La principal razón de implementar capas de lógica del negocio radica en diferenciar y separar muy claramente entre el comportamiento de las reglas del negocio de los

detalles de implementación de infraestructura (acceso a datos y repositorios concretos ligados a una tecnología específica) (De La Torre, C y otros, 2010).

En la capa lógica del negocio además estará el acceso a dato encargado de persistir las entidades que se manejan en el negocio, el acceso a los datos almacenados y reúne todos los aspectos del *software* que tienen que ver con el manejo de los datos persistentes. Es donde se gestiona el almacenamiento físico y recuperación de datos. Esta capa expone el acceso a datos a la capa superior. Además debe cumplir los requerimientos de la aplicación a nivel de rendimiento, seguridad, mantenibilidad y soportar cambios de requerimientos de negocio.

La capa de conexión con Twitter es la encargada de manejar todo el proceso de conexión con la API de Twitter mediante la librería Twython para acceder a la API de la red social y acceder a la información de Twitter.

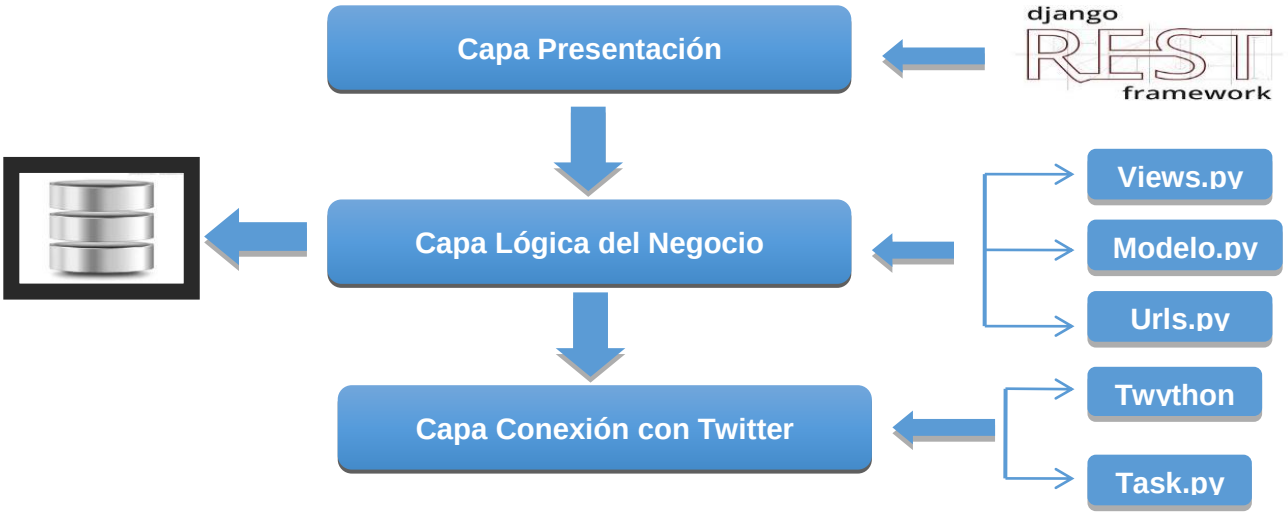


Ilustración 7: Arquitectura del sistema.

Diseño de la base de datos

La persistencia de los datos, para su posterior procesamiento, es un elemento crítico en la implementación del sistema que compete a esta investigación. Como se apuntó anteriormente el sistema estará sustentado por el almacenamiento de los datos en MongoDB. Aunque las entidades en este tipo de base de datos no guardan relación, es necesario realizar una modelación del modo en que se conservarán los datos. MongoDB tiene su propio "lenguaje" a la hora de nombrar ciertos términos que constituyen paradigmas de las bases de datos que funcionan bajo la influencia del SQL. En MongoDB, un documento es la unidad básica de datos, comparable a una *tupla* en bases de datos SQL, pero mucho más expresiva. Una colección de documentos puede ser vista como el equivalente a una tabla de los sistemas relacionales (MongoDB, 2011).

Teniendo en cuenta que las colecciones en MongoDB no guardan relación entre sí, al menos no de forma intencional, se optó para el almacenamiento de los datos de la aplicación tres colecciones que serán detalladas a continuación, almacenadas a su vez en la base de datos.

Colección de MongoDB

En estas colecciones se almacenarán los datos de los *tweets*, usuarios, listas de los cuáles se servirá el sistema para su funcionamiento.

La colección de datos ***UsersCollections*** almacenará los datos correspondientes a los usuarios registrados en la red social Twitter es decir se almacenará todo lo referente a un usuario como por ejemplo sus seguidores y de quien él es seguidor.

La colección ***TweetsCollections*** almacenará los datos asociados a cada *tweet* almacenado en la aplicación, como por ejemplo los *tweet* más recientes de un usuario.

La colección de datos ***ListCollections*** almacenará los datos correspondientes a las listas en Twitter ejemplo las listas de seguidores de un usuario determinado.

Ilustración 8: Colecciones de la base de datos MongoDB.

Modelo de despliegue

El diagrama de despliegue permite indicar la situación física de los componentes lógicos desarrollados. Este modelo es utilizado para capturar los elementos de configuración del procesamiento y las conexiones entre dichos elementos. Cada *Hardware* es identificado como un nodo; un nodo es un elemento donde se ejecutan los componentes, entre ellos existen relaciones que son representados como medios de comunicación tales como HTTP, TCP/IP, etc (Diagrama de Despliegue, 2010).

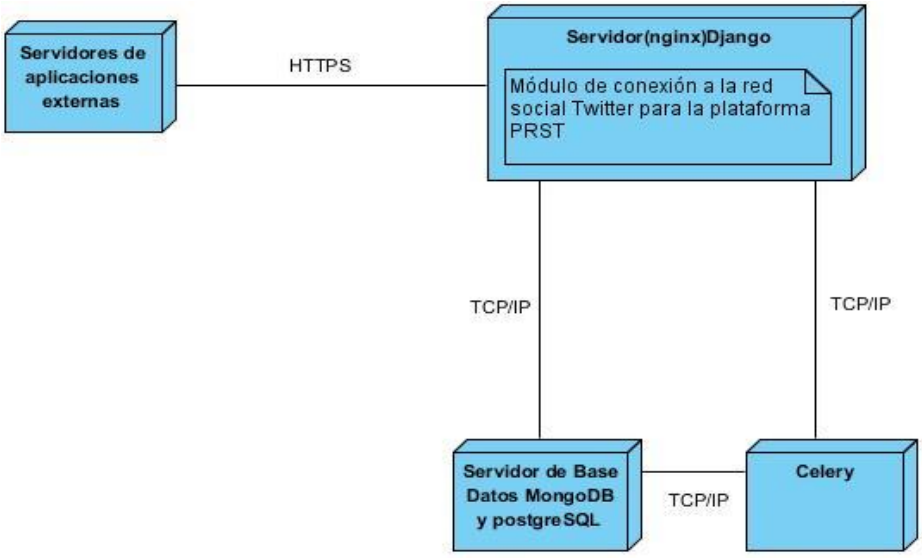


Ilustración 9: Diagrama de Despliegue.

Descripción del diagrama de despliegue

Servidores de aplicaciones externas: Representa el conjunto de computadoras a través de las cuales los usuarios pueden consultar y actualizar la información que se encuentra en el Servidor Web. La comunicación entre las PC Clientes y el Servidor Web se establece utilizando el protocolo de comunicación HTTPS.

Servidor (*nginx*) *Django*: Es el nodo que funciona de intermediario entre las PC Clientes que hacen peticiones y el servidor donde se encuentra la información. Este servidor web toma los datos, realiza sus funciones y presenta la información a las PC Clientes. La comunicación entre el Servidor Web y el servidor de Bases de datos se establece utilizando el protocolo de comunicación TCP/IP.

Servidor de BD: Es el nodo que contiene toda la información del sistema.

Celery: Es el nodo donde se ejecutan las tareas encargadas del intercambio de datos con Twitter de forma distribuida.

Conclusiones del Capítulo

En el presente capítulo se describió la propuesta del Módulo de conexión a la red social Twitter para la PRST. Para mejor entendimiento del problema se realizó el modelo de dominio mediante la representación de los principales conceptos identificados y las relaciones entre ellos. Con el fin de determinar las funcionalidades del sistema se identificaron los requerimientos funcionales y no funcionales los cuales

responden a las necesidades del cliente. Utilizando como metodología de desarrollo de *software* OpenUP y UML como lenguaje de modelado se establecieron los actores, casos de uso del sistema y sus respectivas descripciones. Se modeló la arquitectura general del sistema a desarrollar con el objetivo de dominar el comportamiento de sus módulos. Se realizaron los diagramas del diseño con estereotipos web que muestran de forma abstracta cómo se implementará el sistema y los de interacción específicamente los de secuencia que muestran los pasos lógicos de los objetos que envían y reciben mensajes. Se analizaron las características del sistema a desarrollar para modelar el diagrama de despliegue de la aplicación el cual permitirá un mejor entendimiento del funcionamiento de la aplicación.

Capítulo 3: Implementación y prueba del sistema

El presente capítulo comprende la implementación de los componentes del módulo de conexión a la red social Twitter para la PRST, para lo cual se describe su modelo de componentes. Además, se realiza la validación del sistema a través del plan de pruebas.

Modelo de implementación

El modelo de implementación describe como los elementos del diseño y las clases se implementan en términos de componentes. Contiene los diagramas de componentes y el diagrama de despliegue que comienza a desarrollarse en el flujo de trabajo de diseño y que se perfecciona en la implementación (Jacobson, 2000).

Diagrama de paquetes.

Un diagrama de paquetes muestra como un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema.

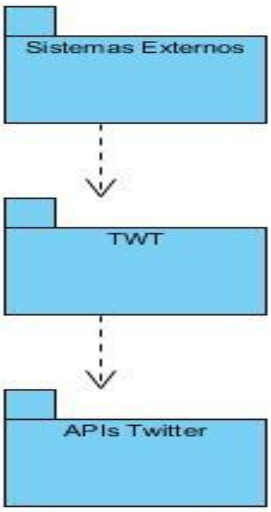


Ilustración 10: Diagrama de paquetes del sistema.

A continuación se hace una breve descripción de cada componente del diagrama.

Componentes	Descripción
API´s Twitter	Colección de servicios que serán consumidos.

TWT	Módulo de conexión a la red social Twitter para la PRST. Paquete que contiene los módulos desarrollados para el sistema.
Sistemas Externos	Aplicaciones o sistemas externos que accederán al sistema para realizar solicitudes de servicios y gestionar información adquirida de Twitter.

Tabla 5: Descripción de los componentes del diagrama de despliegue.

Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos de *software* que entran en el desarrollo de aplicaciones informáticas. Pueden ser simples archivos, paquetes o bibliotecas cargadas dinámicamente (Jacobson, 2000).

A continuación se presenta el diagrama de componentes para el sistema que se propone (ver Figura 11).

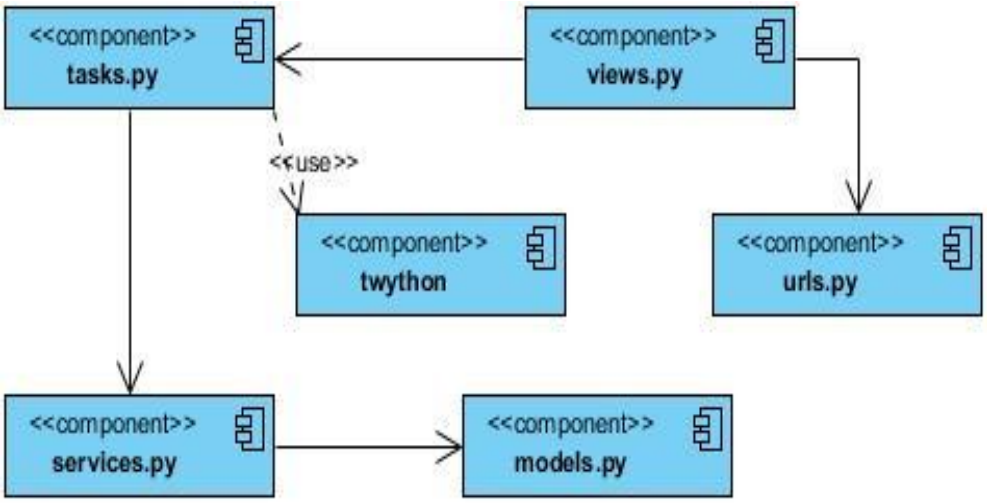


Ilustración 11: Diagrama de componentes del sistema.

En la siguiente tabla se describen los componentes modelados en el diagrama anterior:

Componentes	Descripción
Views.py	Paquete que contiene la clase *Views* que hace función de controladora, es la que controla la lógica del negocio de la aplicación, en ella se capturan los datos enviados por los sistemas externos en formato json, seguido manda la petición a la clase *Tasks* para que ejecute una tarea determinada, según el servicio que sea solicitado y a su vez es la encargada de enviarle la respuesta en formato json a la aplicación que hizo la petición.
Models.py	Su función es la de gestionar los datos de las credenciales de los usuarios.

Services.py	Paquete que contiene la clase *Services* que es usada por la clase *Tasks* para el manejo de las credenciales y la usa también la clase *Views* para la conexión con la base de datos en Mongodb.
Tasks.py	En este paquete se encuentra la clase *Task,* una clase que contiene todas las tareas que serán ejecutadas por *Celery*, se encarga de manejar el flujo de datos desde y hacia Twitter haciendo uso de la librería Twython así como del manejo de las credenciales de los usuarios.
Twython	En este paquete se encuentra la librería Twython que servirá de nexo entre la aplicación y la REST API de Twitter, para gestionar el intercambio de información entre ellas.
URLs.py	Paquete que contiene la clase donde se encuentran todas las direcciones de los servicios disponibles en el módulo de conexión con la red social Twitter.

Tabla 6: Descripción de componentes del sistema.

Estándares de codificación

Los estándares de código resultan importantes en cualquier proyecto de desarrollo, pero son especialmente importantes cuando muchos desarrolladores trabajan en el mismo proyecto. Los estándares de código ayudan a asegurar que el código tenga una alta calidad, menos errores, y pueda ser mantenido fácilmente.

A continuación se exponen varios estándares que presenta Python, utilizados durante el desarrollo del sistema (ver Tabla 7):

Estándares de codificación	Descripción
Indentación	Consiste en insertar espacios en blanco o tabuladores en determinadas líneas de código para facilitar su comprensión. En Python la indentación usa 4 espacios por cada nivel de indentación.
Tabuladores o espacios	Se utiliza la forma más popular de indentar en Python sólo espacios. El código indentado con una mezcla de tabuladores y espacios no es aconsejable.
Tamaño máximo de línea.	Todas las líneas están limitadas a un máximo de 79 caracteres.
Líneas en blanco	Separa las funciones no anidadas y las definiciones de clases con dos líneas en blanco. Las definiciones de métodos dentro de una misma clase se separan con una línea en blanco. Se usan líneas en blanco extras (de forma reservada) para separar grupos de funciones relacionadas.

Tabla 7: Estándares de codificación

Codificación de caracteres

En todos los módulos descritos se ha usado la codificación UTF-8 lo que es especificado al intérprete en la segunda línea de todos los módulos.

```
#!/usr/bin/env python
# -*-coding: utf-8 -*-
```

Imports

Los módulos importados se han colocados en distintas líneas, por ejemplo:

Sí:

```
import os
import sys
```

No:

```
Import sys, os
```

Los imports se han agrupado siguiendo el siguiente orden:

1. Imports de la librería estándar
2. Imports internos del paquete
3. Imports externos de paquetes relacionados se añade una línea en blanco después de cada grupo de imports.

Espacios en blanco en expresiones y sentencias.

Se usan espacios en blanco extra de la siguiente forma:

Inmediatamente después de entrar en un paréntesis o antes de salir de un paréntesis, corchete o llave.

Sí: spam(ham[1], {eggs: 2})

No: spam(ham[1], { eggs: 2 })

Inmediatamente antes de una coma, punto y coma, o dos puntos:

Sí: if x == 4: print x, y; x, y = y, x

No: if x == 4: print x , y ; x , y = y , x

Inmediatamente antes de abrir un paréntesis para una lista de argumentos de una llamada a una función:

Sí: spam(1)

No: spam (1)

Inmediatamente antes de abrir un paréntesis usado como índice o para particionar:

Sí: dict['key'] = list[index]

No: dict ['key'] = list [index]

Más de un espacio alrededor de un operador de asignación (u otro operador) para alinearlo con otro.

Sí:

```
x = 1
y = 2
long_variable = 3
```

No:

```
X         =1
Y          =2
long_variable = 3
```

Se usan espacios alrededor de los operadores aritméticos:

Sí:

```
i = i + 1
submitted += 1
x = x * 2 - 1
hypot2 = x * x + y * y
c = (a + b) * (a -b)
```

No:

```
i=i+1
submitted +=1
x = x*2 -1
hypot2 = x*x + y*y
```

```
c = (a+b) * (a-b)
```

No se usan espacios alrededor del signo '=' cuando se use para indicar el nombre de un argumento o el valor de un parámetro por defecto.

Sí:

```
def complex(real, imag=0.0):
return magic(r=real, i=imag)
```

No:

```
def complex(real, imag = 0.0):
return magic(r = real, i = imag)
```

Aunque a veces es adecuado colocar un if/for/while con un cuerpo pequeño en la misma línea, nunca se hace para sentencias multi-cláusula.

Preferiblemente no:

```
if foo == 'blah': do_blah_thing()
for x in lst: total += x
while t < 10: t = delay()
```

Definitivamente no:

```
if foo == 'blah': do_blah_thing()
else: do_non_blah_thing()
try: something()
finally: cleanup()
do_one(); do_two(); do_three(long, argument, list, like, this)
if foo == 'blah': one(); two(); three()
```

Convenciones de nombres.

Estilos de nombrado:

- minúsculas_con_guiones_bajos
- PalabrasEnMayúsculas (*CapWords o CamelCase*, llamado así por el parecido de las mayúsculas con las jorobas de los camellos). Algunas veces también se llaman *StudlyCaps*.
- capitalizaciónMezclada(se diferencia de PalabrasEnMayúsculas porque la primera letra está en minúsculas)

Nota: Cuando se usan abreviaturas en *CapWords*, se mantiene en mayúsculas todas las letras de la abreviatura. Por lo tanto *HTTPServer*Error es mejor que HttpServerError.

Nombres de paquetes y módulos:

Los módulos tienen nombres cortos formados en su totalidad por letras minúsculas. Se utilizan guiones bajos en el nombre del módulo si mejora la legibilidad. Los paquetes Python también tienen nombres cortos formados por letras minúsculas.

Nombres de clases:

Los nombres de clases usan la convención *CapWords*.

Nombres de funciones:

Los nombres de funciones están en letras minúsculas, con palabras separadas mediante guiones bajos según sea necesario para mejorar la legibilidad.

Argumentos de funciones y métodos:

Se usa siempre *'self'* como primer argumento de los métodos de instancia.

Nombres de métodos y variables de instancia:

Se utilizan las reglas de los nombres de funciones: minúsculas con palabras separadas por guiones bajos cuando es necesario para mejorar la legibilidad. Se usan guiones bajos al inicio sólo para métodos no públicos y variables de instancia. Para evitar colisiones de nombres con subclases, se utilizan dos guiones bajos al principio del nombre para invocar las reglas de planchado de nombres de Python.

Validación del sistema

El flujo de trabajo de prueba de un sistema tiene como objetivo verificar el *software* para comprobar si cumple con sus requisitos. Además, se desarrollan distintos tipos de pruebas en función de los objetivos del sistema. Los tipos de pruebas pueden ser funcionales, de aceptación, de seguridad, de rendimiento, entre otras. Las pruebas verifican que el sistema ofrece a los actores las funcionalidades especificadas.

Pruebas de Funcionalidad

Este tipo de prueba se aplica con el objetivo de localizar errores en el funcionamiento del sistema. Permite identificar repuestas del módulo de conexión a la red social Twitter para la PRST. Al ejecutarse las pruebas de funcionalidad para todos los casos de uso con entradas diferentes, se comprueba el cumplimiento o no de cada uno de los requisitos de *software* definidos en el capítulo anterior (Pressman, Roger, 2009). El

objetivo final de estas es garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable.

A continuación se describe uno de los escenarios y los casos de prueba de prioridad alta para el cliente.

CP_Crear *Tweet*.

Escenario	Descripción	*scren_name*	*status*	Respuesta del sistema	Flujo central
EC 1.1 Crear *Tweet*.	El sistema debe permitir al usuario crear un *tweet*.	V yuriesky2	V Hola mundo	El sistema verifica que el nombre de usuario insertado sea correcto, si lo es, muestra la respuesta al servicio solicitado y lo almacena en la BD_MongoDB.	1-El usuario solicita el servicio mediante la URL correspondiente al mismo. 2- El sistema brinda la posibilidad de introducir los datos en la interfaz del Django Rest Framework 3- El usuario introduce el nombre de usuario y el texto y selecciona el botón "Post".
EC1.2 Crear *tweet* con uno de los campos vacíos	El sistema no crea el tweet debido a que se dejaron campos vacíos.	I yuriesky2	sin texto	El sistema verifica que los campos no estén vacíos, si lo están muestra un mensaje de "Error" si no lo están, muestra la respuesta al servicio solicitado y lo almacena en la BD_MongoDB.	
EC1.3 Crear *tweet* con uno de los campos incorrectos.	El sistema no crea el *tweet* debido a que uno de los datos introducidos es incorrecto.	I yuriesky22	I Hola mundo	El sistema verifica que el nombre de usuario insertado sea correcto y que no deje texto en blanco, si no lo es muestra un mensaje de "Error" si lo es, muestra la respuesta al servicio solicitado y lo almacena en la BD_MongoDB.	

Ilustración 12: Casos de prueba del caso de uso Crear *Tweet*

Como resultado de la aplicación de este tipo de prueba, se detectaron (10) no conformidades, distribuidas en 3 iteraciones como se muestra en la Ilustración 13. Las no conformidades se agrupan en tres tipos: errores y campos no validados del sistema (6), funcionalidades no implementadas (3) y funcionalidades ineficientes (1) (ver Ilustración 13).

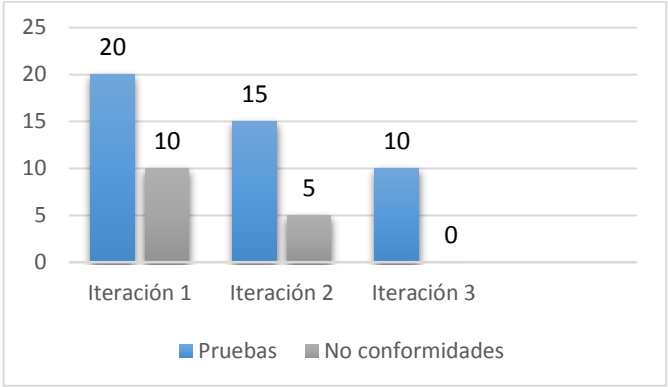


Ilustración 13: Iteraciones de las pruebas funcionales realizadas

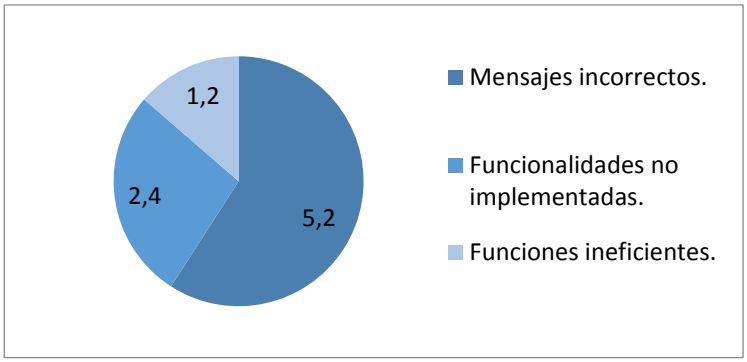


Ilustración 14: Distribución de las no conformidades identificadas.

Pruebas de rendimiento

Mediante las pruebas de rendimiento es posible hallar tendencias y comportamientos para los elementos de una aplicación, los cuales generan bajo rendimiento. Este tipo de pruebas permiten identificar cuellos de botella, capacidad de concurrencia de usuarios, tiempos de respuesta de operaciones de negocio a nivel de sistema, establecer un marco de referencia para pruebas futuras, determinar el cumplimiento de los objetivos de rendimiento y requerimientos no funcionales, entre otros (V&V Quality, 2013).

Pruebas de carga: Mediante la ejecución de las pruebas de Carga es posible identificar la capacidad de recuperación de un sistema cuando es sometido a cargas variables, tanto de usuarios como de procesos. Al realizar las pruebas de carga se puede determinar el tiempo de respuesta de todas las transacciones críticas del sistema y encontrar cuellos de botella de la aplicación (V&V Quality, 2013).

Prueba de estrés: Mediante las pruebas de Estrés es posible identificar la capacidad de respuesta de un sistema bajo condiciones de carga extrema, representadas por una alta concurrencia de usuarios y/o

procesos. Una vez realizadas las pruebas de estrés se podrá conocer el punto de quiebre del aplicativo en términos de capacidad de respuesta, con lo cual será posible establecer acciones de optimización en diferentes niveles para asegurar una mejor capacidad de concurrencia de usuarios y procesos, que se verá reflejada en una óptima operación de negocio (V&V Quality, 2013).

Para la realización de las pruebas de rendimiento se utiliza la herramienta JMeter, estas pruebas fueron divididas en test de 50, 100 y 150 hilos cada uno, los cuales simulan la cantidad de usuarios que acceden a las funcionalidades concurrentemente. Se tiene en cuenta el escenario donde se encuentra la aplicación y las condiciones del *hardware y software* de la computadora donde es ejecutado el muestreo; atendiendo a las siguientes características:

Hardware: Posee una memoria RAM de 2GB, procesador Intel Core 2 duo a 2.10 GHz y una capacidad de disco duro de 160GB.

Software: Tiene instalado como plataforma de desarrollo Debian en su versión 6.0.7.

Se utilizó el componente “Informe Agregado", el cual permite visualizar los resultados del muestreo a algunas funcionalidades del módulo de conexión con la red social Twitter, obteniendo resultados más precisos.

Para una concurrencia de 50 usuarios se realizaron 200 peticiones al servidor obteniéndose un tiempo mínimo de respuesta de 35 segundos, con un margen de error del 0.0 % y un rendimiento promedio de 24.5 segundos (ver Figura 5 en los anexos). Para 100 usuarios concurrentes se ejecutaron 400 peticiones al servidor con un tiempo mínimo de respuesta de 32 segundos, obteniendo un margen de error de 0.0 % y un rendimiento promedio de 23.7 segundos (ver Figura 6 en los anexos). Y para un total de 150 usuarios concurrentes se efectuaron 600 peticiones al servidor con un tiempo mínimo de respuesta de 25 segundos, un margen de error de 0.0 % y un rendimiento promedio de 17.8 segundos para el número de peticiones realizadas.

Conclusiones del Capítulo

En el capítulo que concluye, el desarrollo de componentes, constituyen la base para la implementación del módulo de conexión a la red social Twitter para la PRST. A través del modelado del diagrama de componentes, se representan los módulos utilizados con sus respectivos ficheros, ejecutables, entre otros; permitiendo un mejor entendimiento de cómo está estructurado el sistema. La validación de la aplicación se realizó utilizando pruebas funcionales, y de rendimiento; arrojó resultados positivos que demostraron el cumplimiento de los requisitos de *software* definidos.

Conclusiones Generales

Concluida la investigación es posible destacar una serie de resultados que se enumeran a continuación:

- El estudio de las *API´s* que brinda Twitter permitió comprender cómo funciona el proceso de interacción con la red social.
- Se logró diseñar y desarrollar el Módulo de conexión para la PRST, encaminado a brindar una solución viable al problema identificado.
- La arquitectura propuesta para el módulo permitirá que las aplicaciones de PRST se cohesionen y que las tecnologías utilizadas para la implementación de estas sean las más apropiadas para sus requerimientos.
- La aplicación de pruebas permitió validar el correcto funcionamiento de la aplicación desarrollada simulando ambientes reales en diferentes entornos de ejecución.
- La integración de las aplicaciones federadas a PRST con el módulo de conexión permitirá interactuar con los servicios de la red social Twitter.

Recomendaciones

Con el objetivo de lograr mejoras en el funcionamiento de la aplicación desarrollada como parte de esta investigación, se recomienda:

- Continuar con el estudio a la red social Twitter, pues cada día surgen nuevas actualizaciones de servicios que optimizarían el funcionamiento del módulo.
- Continuar con el proceso de desarrollo del sistema con el fin de incorporarle nuevas funcionalidades para facilitar el trabajo de las aplicaciones federadas a la PRST.

Glosario de términos

ACID: Conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción.

***API*:** Interfaz de programación de aplicaciones (IPA) o API (del Inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción.

BSD son las siglas de "*Berkeley Software Distribution*". Así se llamó a las distribuciones de código fuente que se hicieron en la Universidad de Berkeley en California y que en origen eran extensiones del sistema operativo UNIX® de AT&T *Research*. Varios proyectos de sistemas operativos de código abierto tienen su origen en una distribución de éste código conocida como 4.4BSD-Lite. Añaden además un buen número de paquetes de otros proyectos de Código Abierto, incluyendo de forma destacada al proyecto GNU.

***CASE*:** *Computer Aided Software Engineering*, por sus siglas en Inglés, en Español Sistemas de Ingeniería de *Software* Asistidos por Computador.

Framework: Es una estructura de soporte definida, mediante la cual otro proyecto de *software* puede ser organizado y desarrollado.

HTTP: *HypertextTransferProtocol* por sus siglas en Inglés, en Español Protocolo de Transferencia de Hipertexto. Es el protocolo usado en cada transacción de la *World Wide Web*.

HTTPS: Hypertext Transfer *Protocol Secure* por sus siglas en Inglés, en Español Protocolo Seguro de Transferencia de Hipertexto. Es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de Hipertexto, es decir, es la versión segura de HTTP.

Informática: Disciplina que estudia el tratamiento automático de la información utilizando dispositivos electrónicos y sistemas computacionales.

Interoperabilidad: Capacidad que tiene un producto o un sistema, cuyas interfaces son totalmente conocidas, para funcionar con otros productos o sistemas existentes o futuros y eso sin restricción de acceso o de implementación.

Paquete: Mecanismo de propósito general para organizar elementos en grupos.

PHP: *Hypertext* Pre-*Processor*, por sus siglas en Inglés, en Español Preprocesador de Hipertexto.

RUP: *Rational Unified Process*, por sus siglas en inglés, en español Proceso Unificado del *Rational*. Es una metodología de desarrollo de *software* tradicional, actualmente propiedad de IBM. Junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Software: Se refiere al equipamiento lógico o soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas.

SSL: *Secure Sockets Layer* por sus siglas en Inglés SSL, en Español Capa de Conexión Segura. Es un protocolo criptográfico que proporciona comunicación segura por una red, comúnmente Internet.

Stakeholder: Es un término inglés utilizado por primera vez por R. E. *Freeman* en su obra: "*Strategic* Management: A *Stakeholder Approach*" para referirse a «quienes pueden afectar o son afectados por las actividades de una empresa».

TCP/IP: Se le denomina conjunto de protocolos TCP/IP, en referencia a los dos protocolos más importantes que la componen: Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP), que fueron dos de los primeros en definirse.

***UML*:** *Unified Modeling Language* dadas sus siglas en Inglés, en Español Lenguaje Unificado de Modelado. Es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de *software*.

MATLAB: *Software* matemático con entorno de desarrollo integrado (IDE) que tiene un lenguaje de programación propio (Lenguaje M) y es multiplataforma. *Software* de un gran uso en Centros de Investigación y Desarrollo así como en universidades.

Referencias Bibliográficas

Lenguaje de programacion. 2007. Java. [En línea] 2007. [Citado el: 20 de 01 de 2010.] http://www.iec.csic.es/criptonomicon/java/quesjava.html.

Rational Rose. 2007. Rational Rose. Pagina Oficial Rational Rose. [En línea] 2007. [Citado el: 20 de 02 de 2014.] http://www.rational.com.ar/herramientas/roseenterprise.html.

Aliaga, Antonio y Miani Agustin, Marcos. 2011. *PostgreSQL.* 2011.

Biblioteca UCI . 2014. Catálogo Biblioteca UCI en línea. [En línea] 2014. [Citado el: 15 de 03 de 2014.] http://catalogoenlinea.uci.cu/.

Celery: Distributed Task Queue. 2011. Homepage | Celery: Distributed Task Queue. [En línea] 2011. [Citado el: 04 de 05 de 2014.] http://www.celeryproject.org/.

Cynthia Lorena Corso. 2007. Lenguaje_de_programacion_python. [En línea] 2007. http://labsys.frc.utn.edu.ar/pdf/latinoamerica_educa_III/lenguaje_de_programacion_python.pdf.

Dale Dougherty. 2004. Concepto de Web 2.0. [En línea] O'Reilly Media, 2004. http://www.ite.educacion.es/formacion/materiales/155/cd/modulo_1_Iniciacionblog/concepto_de_web_20.html.

De la Torre, C y otros. 2010. *Guía de Arquitectura N-Capas orientada al dominio con .NET 4.0.* España: Microsoft Ibérica SRL. : s.n., 2010.

De La Torre, C y otros. 2010. *Guía de Arquitectura N-Capas orientada al dominio con .NET 4.0.* España: : Microsoft Ibérica SRL., 2010.

Diagrama de Despliegue. 2010. Guía de Usuario de Enterprise Architect 7.0. Diagrama de Despliegue. [En línea] 6 de 05 de 2010. http://www.sparxsystems.com.ar/download/ayuda/index.html?deploymentdiagram.htm.

djangoproject.com. 2013. www.djangoproject.com/. [En línea] 2013. [Citado el: 20 de 02 de 2014.] https://www.djangoproject.com/.

Eclipse. 2010. Eclipse, entorno de desarrollo integrado . [En línea] 2010. [Citado el: 06 de 04 de 2014.] http://www.ecured.cu/index.php/Eclipse,_entorno_de_desarrollo_integrado.

Eva.uci.cu. 2014. Eva.uci.cu. [En línea] 2014. [Citado el: 26 de 02 de 2014.] http://eva.uci.cu/mod/resource/view.php?id=8500&subdir=/El_Proceso_Unificado_de_Desarrollo.

Fernandez Luna. 2006. Entorno desarrollo Integrado . [En línea] septiembre de 2006. [Citado el: 20 de 01 de 2014.] http://leo.ugr.es/J2ME/INTRO/intro_11.htm.

Granado. 2004. www. es.scribd.com. . [En línea] 2004. [Citado el: 25 de 02 de 2014.] http://es.scribd.com/doc/36261999/Libro-de-programacion-en-PHP-5.

Gregorio. 2002. *Programacion extrema y Software Libre.* 2002.

Historia de Twitter. 2013. Historia de Twitter. [En línea] 2013. [Citado el: 06 de 04 de 2014.] http://www.cad.com.mx/historia_de_twitter.htm.

IDE de desarrollo Open Source. 2011. IDE de desarrollo Open Source. [En línea] 05 de 10 de 2011. http://aplicaciones.org/eclipse-ide-de-desarrollo-open-source.

Ingenieria del Software. 05_Parte_III_Desarrollo_iterativo_e_incremental.pdf. [En línea] [Citado el: 15 de 03 de 2014.] https://eva.uci.cu.

Interoperability-definition. interoperability-definition. [En línea] http://interoperability-definition.info/es/.

Iteroperabilidad. 2008. Inabartiateam. [En línea] 2008. [Citado el: 06 de 02 de 2014.] http://www.abartiateam.com/interoperabilidad.

Jacobson. 2000. *El Proceso Unificado de Desarrollo de Software.* 2000.

James Clyde Mitchell. 1969. *Social Networks in Urban Settings.* 1969.

Jmeter. 2011. The Apache Software Fundation. [En línea] 2011. [Citado el: 28 de 02 de 2014.] http://jmeter.apache.org/.

Karenny Brito Acuña. 2009. *Seleccion de metodologias de Desarrollo para aplicaciones Web en la Facultad de Informatica de la universidad de Cienfuegos.* Cienfuegos : Cuba : s.n., 2009.

Larman, Craig. 2000. *UML y Patrones.* México : Prentice Hall,, 2000. 1ra Edición.

Lenguaje de programación. 2008. Definición de lenguaje de programación - Qué es, Significado y Concepto. [En línea] 2008. [Citado el: 20 de 04 de 2014.] http://definicion.de/lenguaje-de-programacion/.

Los-records-de-twitter. 2013. los-records-de-twitter. [En línea] 2013. [Citado el: 20 de 02 de 2014.] http://asktutorial.com/los-records-de-twitter-350-millones-de-tweets-por-dia/.

Luciano. 2009. *Entornos de Desarrollo Integrado para Java. .* 2009.

MongoDB. 2011. MongoDB.com. [En línea] 2011. [Citado el: 05 de 02 de 2014.] http://www.mongodb.com..

Netbeans.org. 2012. Netbeans.org. [En línea] 2012. [Citado el: 26 de 01 de 2014.] http://netbeans.otg/community/releases/72.

Perdue, Tim. 2000. [En línea] 30 de 07 de 2000. [Citado el: 06 de 02 de 2014.] http://www.phpbuilder.com/columns/tim20000705.php3?page=4.

postgresql.org.es. 2012. Sobre PostgreSQL | www.postgresql.org.es. [En línea] 2012. [Citado el: 25 de 01 de 2014.] http://www.postgresql.org.es/sobre_postgresql.

PostgreSQL-es. www.postgresql-es.org. [En línea] [Citado el: 04 de 02 de 2014.] http://www.postgresql-es.org/principal.

Pressman, Roger. 2009. Ingenieria de Software: Un enfoque práctico. . [aut. libro] Roger Pressman. *Ingenieria de Software.* s.l. : McGraw-Hill, 2009.

Python. 2007. *Lenguaje_de_programacion_python.* 2007.

RabbitMQ. 2010. RabbitMQ - Messaging that just works. [En línea] 2010. [Citado el: 06 de 06 de 2014.] http://www.rabbitmq.com/.

Rational Rose. 2007. Pagina Oficial Rational Rose. [En línea] 2007. http://www.rational.com.ar/herramientas/roseenterprise.html.

—. 2007. Software, IBM Rational. [En línea] IBM Corporation, 2007. http://www.rational.com.

Ricardo Bulduino. 2007. Introduccion to Open UP. [En línea] 2007. http://www.eclipse.org/epf/general/OpenUP.pdf..

Rumbaugh, James. 2007. El Lenguaje Unificado de Modelado. Manual de Referencia. [aut. libro] Addison Wesley. *UML.* Madrid : s.n., 2007.

—. 2007. El Lenguaje Unificado de Modelado. Manual de Referencia. [En línea] 2007. [Citado el: 25 de 01 de 2014.] Rumbaugh, James. 2007. El Lenguaje Unificado de Modelado. Manual de Referencia. [aut. libro] Addison Wesley. UML. Madrid : s.n., 2007..

Sistema Gestor de Bases de datos. 2012. Ecured.cu. [En línea] 2012. [Citado el: 05 de 02 de 2014.] http://www.ecured.cu/index.php/Sistema_Gestor_de_Base_de_Datos..

Sistema_Gestor_de_Base_de_Datos. 2012. Ecured.cu. [En línea] 2012. [Citado el: 25 de 01 de 2014.] http://www.ecured.cu/index.php/Sistema_Gestor_de_Base_de_Datos.

Social Network Sites. **Boyd, Danah M; Ellison, Nicole B. 2007.** 1, 17 de Diciembre de 2007, Journal of Computer-Mediated Communication, Vol. 13.

Sommerville, Ian. 2006. *Software Engineering.* 2006.

Spyder. 2012. Pinguino Libre.org. [En línea] 28 de 06 de 2012. [Citado el: 23 de 01 de 2014.] http://www.pinguinolibre.org/2012/06/28/spyder-un-ide-cientifico-para-python/..

Tim O'Reilly. [En línea] http://es.wikipedia.org/wiki/Tim_O'Reilly.

Twitter Development. 2012. dev.twitter.com/docs. [En línea] 2012. [Citado el: 20 de 01 de 2014.] https://dev.twitter.com/docs..

Twitter. 2014. Twiiter for Business. [En línea] 1 de Febrero de 2014. https://business.twitter.com/twitter-101.

Twitter4J. 2007. Twitter4J - A Java library for the Twitter API. [En línea] 2007. [Citado el: 06 de 02 de 2014-- 22:11:49.] http://twitter4j.org/en/index.html.

Twitter-libraries. dev.twitter.com/docs/twitter-libraries. [En línea] [Citado el: 15 de 03 de 2014.] https://dev.twitter.com/docs/twitter-libraries.

V&V Quality. 2013. V&V Quality. [En línea] 2013.

Visual Paradigm for UML. 2010. Visual Paradigm for UML. [En línea] 2010. [Citado el: 26 de 01 de 2014.] http://www.visual-paradigm.com/.

Bibliografía

Lenguaje de programacion. 2007. Java. [En línea] 2007. [Citado el: 20 de 01 de 2010.] http://www.iec.csic.es/criptonomicon/java/quesjava.html.

Rational Rose. 2007. Rational Rose. Pagina Oficial Rational Rose. [En línea] 2007. [Citado el: 20 de 02 de 2014.] http://www.rational.com.ar/herramientas/roseenterprise.html.

Aliaga, Antonio y Miani Agustin, Marcos. 2011. *PostgreSQL.* 2011.

Biblioteca UCI . 2014. Catálogo Biblioteca UCI en línea. [En línea] 2014. [Citado el: 15 de 03 de 2014.] http://catalogoenlinea.uci.cu/.

Celery: Distributed Task Queue. 2011. Homepage | Celery: Distributed Task Queue. [En línea] 2011. [Citado el: 04 de 05 de 2014.] http://www.celeryproject.org/.

Cynthia Lorena Corso. 2007. Lenguaje_de_programacion_python. [En línea] 2007. http://labsys.frc.utn.edu.ar/pdf/latinoamerica_educa_III/lenguaje_de_programacion_python.pdf.

Dale Dougherty. 2004. Concepto de Web 2.0. [En línea] O'Reilly Media, 2004. http://www.ite.educacion.es/formacion/materiales/155/cd/modulo_1_Iniciacionblog/concepto_de_web_20.html.

De la Torre, C y otros. 2010. *Guía de Arquitectura N-Capas orientada al dominio con .NET 4.0.* España: Microsoft Ibérica SRL. : s.n., 2010.

De La Torre, C y otros. 2010. *Guía de Arquitectura N-Capas orientada al dominio con .NET 4.0.* España: : Microsoft Ibérica SRL., 2010.

Diagrama de Despliegue. 2010. Guía de Usuario de Enterprise Architect 7.0. Diagrama de Despliegue. [En línea] 6 de 05 de 2010. http://www.sparxsystems.com.ar/download/ayuda/index.html?deploymentdiagram.htm.

djangoproject.com. 2013. www.djangoproject.com/. [En línea] 2013. [Citado el: 20 de 02 de 2014.] https://www.djangoproject.com/.

Eclipse. 2010. Eclipse, entorno de desarrollo integrado . [En línea] 2010. [Citado el: 06 de 04 de 2014.] http://www.ecured.cu/index.php/Eclipse,_entorno_de_desarrollo_integrado.

Eva.uci.cu. 2014. Eva.uci.cu. [En línea] 2014. [Citado el: 26 de 02 de 2014.] http://eva.uci.cu/mod/resource/view.php?id=8500&subdir=/El_Proceso_Unificado_de_Desarrollo.

Fernandez Luna. 2006. Entorno desarrollo Integrado . [En línea] septiembre de 2006. [Citado el: 20 de 01 de 2014.] http://leo.ugr.es/J2ME/INTRO/intro_11.htm.

Granado. 2004. www. es.scribd.com. . [En línea] 2004. [Citado el: 25 de 02 de 2014.] http://es.scribd.com/doc/36261999/Libro-de-programacion-en-PHP-5.

Gregorio. 2002. *Programacion extrema y Software Libre.* 2002.

Historia de Twitter. 2013. Historia de Twitter. [En línea] 2013. [Citado el: 06 de 04 de 2014.] http://www.cad.com.mx/historia_de_twitter.htm.

IDE de desarrollo Open Source. 2011. IDE de desarrollo Open Source. [En línea] 05 de 10 de 2011. http://aplicaciones.org/eclipse-ide-de-desarrollo-open-source.

Ingenieria del Software. 05_Parte_III_Desarrollo_iterativo_e_incremental.pdf. [En línea] [Citado el: 15 de 03 de 2014.] https://eva.uci.cu.

Interoperability-definition. interoperability-definition. [En línea] http://interoperability-definition.info/es/.

Iteroperabilidad. 2008. Inabartiateam. [En línea] 2008. [Citado el: 06 de 02 de 2014.] http://www.abartiateam.com/interoperabilidad.

Jacobson. 2000. *El Proceso Unificado de Desarrollo de Software.* 2000.

James Clyde Mitchell. 1969. *Social Networks in Urban Settings.* 1969.

Jmeter. 2011. The Apache Software Fundation. [En línea] 2011. [Citado el: 28 de 02 de 2014.] http://jmeter.apache.org/.

Karenny Brito Acuña. 2009. *Seleccion de metodologias de Desarrollo para aplicaciones Web en la Facultad de Informatica de la universidad de Cienfuegos.* Cienfuegos : Cuba : s.n., 2009.

Larman, Craig. 2000. *UML y Patrones.* México : Prentice Hall,, 2000. 1ra Edición.

Lenguaje de programación. 2008. Definición de lenguaje de programación - Qué es, Significado y Concepto. [En línea] 2008. [Citado el: 20 de 04 de 2014.] http://definicion.de/lenguaje-de-programacion/.

Los-records-de-twitter. 2013. los-records-de-twitter. [En línea] 2013. [Citado el: 20 de 02 de 2014.] http://asktutorial.com/los-records-de-twitter-350-millones-de-tweets-por-dia/.

Luciano. 2009. *Entornos de Desarrollo Integrado para Java. .* 2009.

MongoDB. 2011. MongoDB.com. [En línea] 2011. [Citado el: 05 de 02 de 2014.] http://www.mongodb.com..

Netbeans.org. 2012. Netbeans.org. [En línea] 2012. [Citado el: 26 de 01 de 2014.] http://netbeans.otg/community/releases/72.

Perdue, Tim. 2000. [En línea] 30 de 07 de 2000. [Citado el: 06 de 02 de 2014.] http://www.phpbuilder.com/columns/tim20000705.php3?page=4.

postgresql.org.es. 2012. Sobre PostgreSQL | www.postgresql.org.es. [En línea] 2012. [Citado el: 25 de 01 de 2014.] http://www.postgresql.org.es/sobre_postgresql.

PostgreSQL-es. www.postgresql-es.org. [En línea] [Citado el: 04 de 02 de 2014.] http://www.postgresql-es.org/principal.

Pressman, Roger. 2009. Ingenieria de Software: Un enfoque práctico. . [aut. libro] Roger Pressman. *Ingenieria de Software.* s.l. : McGraw-Hill, 2009.

Python. 2007. *Lenguaje_de_programacion_python.* 2007.

RabbitMQ. 2010. RabbitMQ - Messaging that just works. [En línea] 2010. [Citado el: 06 de 06 de 2014.] http://www.rabbitmq.com/.

Rational Rose. 2007. Pagina Oficial Rational Rose. [En línea] 2007. http://www.rational.com.ar/herramientas/roseenterprise.html.

—. 2007. Software, IBM Rational. [En línea] IBM Corporation, 2007. http://www.rational.com.

Ricardo Bulduino. 2007. Introduccion to Open UP. [En línea] 2007. http://www.eclipse.org/epf/general/OpenUP.pdf..

Rumbaugh, James. 2007. El Lenguaje Unificado de Modelado. Manual de Referencia. [aut. libro] Addison Wesley. *UML.* Madrid : s.n., 2007.

—. 2007. El Lenguaje Unificado de Modelado. Manual de Referencia. [En línea] 2007. [Citado el: 25 de 01 de 2014.] Rumbaugh, James. 2007. El Lenguaje Unificado de Modelado. Manual de Referencia. [aut. libro] Addison Wesley. UML. Madrid : s.n., 2007..

Sistema Gestor de Bases de datos. 2012. Ecured.cu. [En línea] 2012. [Citado el: 05 de 02 de 2014.] http://www.ecured.cu/index.php/Sistema_Gestor_de_Base_de_Datos..

Sistema_Gestor_de_Base_de_Datos. 2012. Ecured.cu. [En línea] 2012. [Citado el: 25 de 01 de 2014.] http://www.ecured.cu/index.php/Sistema_Gestor_de_Base_de_Datos.

Social Network Sites. **Boyd, Danah M; Ellison, Nicole B. 2007.** 1, 17 de Diciembre de 2007, Journal of Computer-Mediated Communication, Vol. 13.

Sommerville, Ian. 2006. *Software Engineering.* 2006.

Spyder. 2012. Pinguino Libre.org. [En línea] 28 de 06 de 2012. [Citado el: 23 de 01 de 2014.] http://www.pinguinolibre.org/2012/06/28/spyder-un-ide-cientifico-para-python/..

Tim O'Reilly. [En línea] http://es.wikipedia.org/wiki/Tim_O'Reilly.

Twitter Development. 2012. dev.twitter.com/docs. [En línea] 2012. [Citado el: 20 de 01 de 2014.] https://dev.twitter.com/docs..

Twitter. 2014. Twiiter for Business. [En línea] 1 de Febrero de 2014. https://business.twitter.com/twitter-101.

Twitter4J. 2007. Twitter4J - A Java library for the Twitter API. [En línea] 2007. [Citado el: 06 de 02 de 2014-- 22:11:49.] http://twitter4j.org/en/index.html.

Twitter-libraries. dev.twitter.com/docs/twitter-libraries. [En línea] [Citado el: 15 de 03 de 2014.] https://dev.twitter.com/docs/twitter-libraries.

V&V Quality. 2013. V&V Quality. [En línea] 2013.

Visual Paradigm for UML. 2010. Visual Paradigm for UML. [En línea] 2010. [Citado el: 26 de 01 de 2014.] http://www.visual-paradigm.com/.

Anexos

Tablas de descripcion de casos de uso.

Tabla 1: CU Buscar Tweet.

Objetivo	Buscar tweet.	
Actores	Sistema Externo	
Resumen	El caso de uso se inicia cuando una de las aplicaciones de la PRST mediante la URL correspondiente solicita el servicio de realizar una búsqueda de tweet, el sistema, busca los tweet de acuerdo a un criterio y muestra la respuesta al servicio. Termina el caso de uso.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones		
Postcondiciones	Se obtuvo el tweet de acuerdo al criterio de búsqueda introducido.	
Requisitos funcionales	RF 6	
Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	Sistema
	El usuario realiza la petición de servicios Buscar Tweet mediante una URL.	
	El usuario envía en formato json el criterio de búsqueda que lleva esa petición (consulta).	El sistema captura los datos enviados por el usuario y devuelve una respuesta que cumpla con el criterio de búsqueda especificado.
		Termina el CU.
Flujos alternos		
Flujo básico <Criterio de búsqueda erróneo >		
	Actor	Sistema
		Muestra un mensaje avisando que no se encontraron resultados que cumplan con el criterio de búsqueda especificado. Termina el CU.
Flujos alternos		
Nº Evento		
	Actor	Sistema
1.		
2.		
Relaciones	CU Incluidos	
	CU Extendidos	

Requisitos no funcionales	
Asuntos pendientes	

Tabla 2: CU Buscar Usuarios.

Objetivo	Buscar usuarios públicos.	
Actores	Sistema Externo	
Resumen	El caso de uso se inicia cuando una de las aplicaciones de la PRST mediante la URL correspondiente solicita el servicio de realizar una búsqueda de usuarios, el sistema, busca el usuario de acuerdo a un criterio y muestra la respuesta al servicio. Termina el caso de uso.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones		
Postcondiciones	Se obtuvo el usuario de acuerdo al criterio de búsqueda introducido.	
Requisitos funcionales	RF 6	
Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	Sistema
1	El usuario realiza la petición de servicios Buscar usuarios públicos mediante su URL correspondiente.	
2	El usuario envía en formato json el criterio de búsqueda que lleva esa petición (consulta).	El sistema captura los datos enviados por el usuario y verifica que sean correctos.
3		El sistema válida las credenciales del usuario.
4		El sistema luego de haber validado las credenciales del usuario devuelve una respuesta de acuerdo al criterio de búsqueda especificado. Termina el caso de uso.
Flujos alternos Sesión 1: Credenciales no válidas		
Flujo básico <Nombre del flujo básico>		
	Actor	Sistema
		Si las credenciales del usuario son incorrectas el sistema muestra un mensaje indicando que

		“Credenciales no válidas”.
Flujos alternos		
Nº Evento <Criterio de búsquedas erróneo >		
	Actor	Sistema
1.		Muestra un mensaje avisando que no se encontraron resultados que cumplan con el criterio de búsqueda especificado. Termina el CU.
2.		
Flujos alternos		
Relaciones	CU Incluidos	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes		
Asuntos pendientes		

Tabla3: CU Verificar Credenciales.

Objetivo	Verificar credenciales	
Actores	Sistema Externo	
Resumen	El caso de uso se inicia cuando una de las aplicaciones de la PRST mediante la URL correspondiente solicita el servicio Verificar credenciales de un usuario, el sistema, verifica las credenciales y muestra la respuesta al servicio solicitado. Termina el caso de uso.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones		
Postcondiciones	Quedan validadas las credenciales del usuario.	
Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	Sistema
1.	El usuario realiza la petición de servicios Verificar Credenciales mediante su URL correspondiente.	El sistema envía los datos al servidor para verificar que sean correctos.
2		El sistema valida las credenciales del usuario.

3		El sistema luego de a ver validado las credenciales del usuario sin son válidas devuelve una respuesta. Termina el caso de uso.
Flujos alternos		
Nº Evento “Credenciales no válidas ”		
	Actor	Sistema
		Si las credenciales del usuario son incorrectas el sistema muestra un mensaje indicando “Credenciales no válidas”.

Relaciones	CU Incluidos	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes		

Tabla 4: CU_Mostrar tweet con información de retweet.

Objetivo	Mostrar tweet con información de retweet.
Actores	Sistema Externo
Resumen	El caso de uso se inicia cuando una de las aplicaciones de la PRST mediante la URL correspondiente solicita el servicio de Mostrar tweet con información de retweet, el sistema, busca los tweet de acuerdo a un criterio y muestra la respuesta al servicio solicitado. Termina el caso de uso.
Complejidad	Alta
Prioridad	Alta
Precondiciones	
Postcondiciones	Se mostraron los tweet con información de sus retweet de acuerdo al servicio solicitado por el usuario.
Requisitos funcionales	RF 3
Flujo de eventos	
Flujo básico <Nombre del flujo básico>	

	Actor	Sistema
1	El usuario realiza la petición de servicio Mostrar Tweet con información de retweets mediante su URL correspondiente.	
2	El usuario envía en formato json los parámetros que lleva esa petición.	El sistema envía los datos al servidor para verificar que sean correctos.
3		El sistema valida las credenciales del usuario.
		El sistema luego de a ver validado las credenciales del usuario si son correctas devuelve una respuesta. Termina el caso de uso.
Flujos alternos		
Flujo básico <Credenciales no válidas>		
	Actor	**Sistema**
		Si las credenciales del usuario son incorrectas el sistema muestra un mensaje indicando que "Credenciales no válidas".
Flujos alternos		
Nº Evento < >		
	Actor	**Sistema**

Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes		

Diagrama de clases del diseño:

Figura 1: CU_Buscar Tweet.

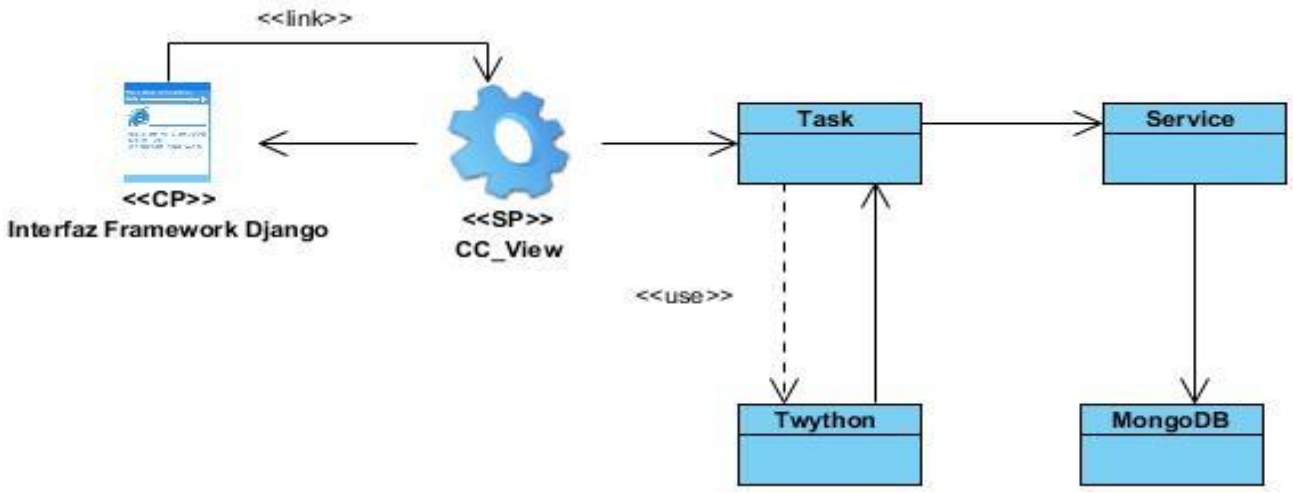
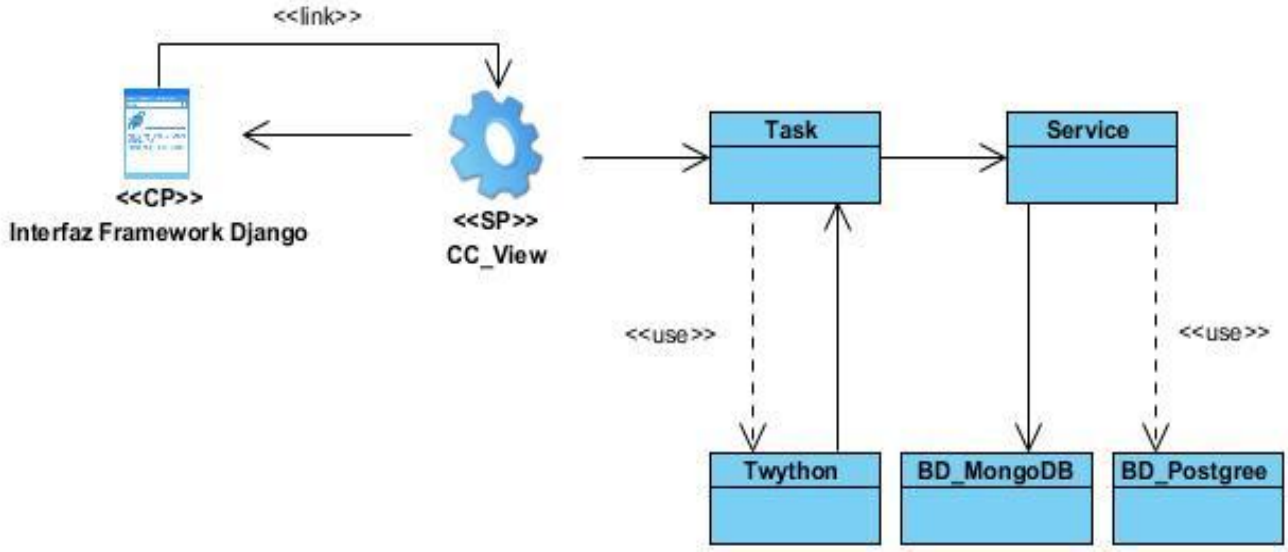


Figura 2: CU_Verificar Credenciales



Diagramas de Secuencia

Figura 3: Diagrama de Secuencia CU_Verificar Credenciales

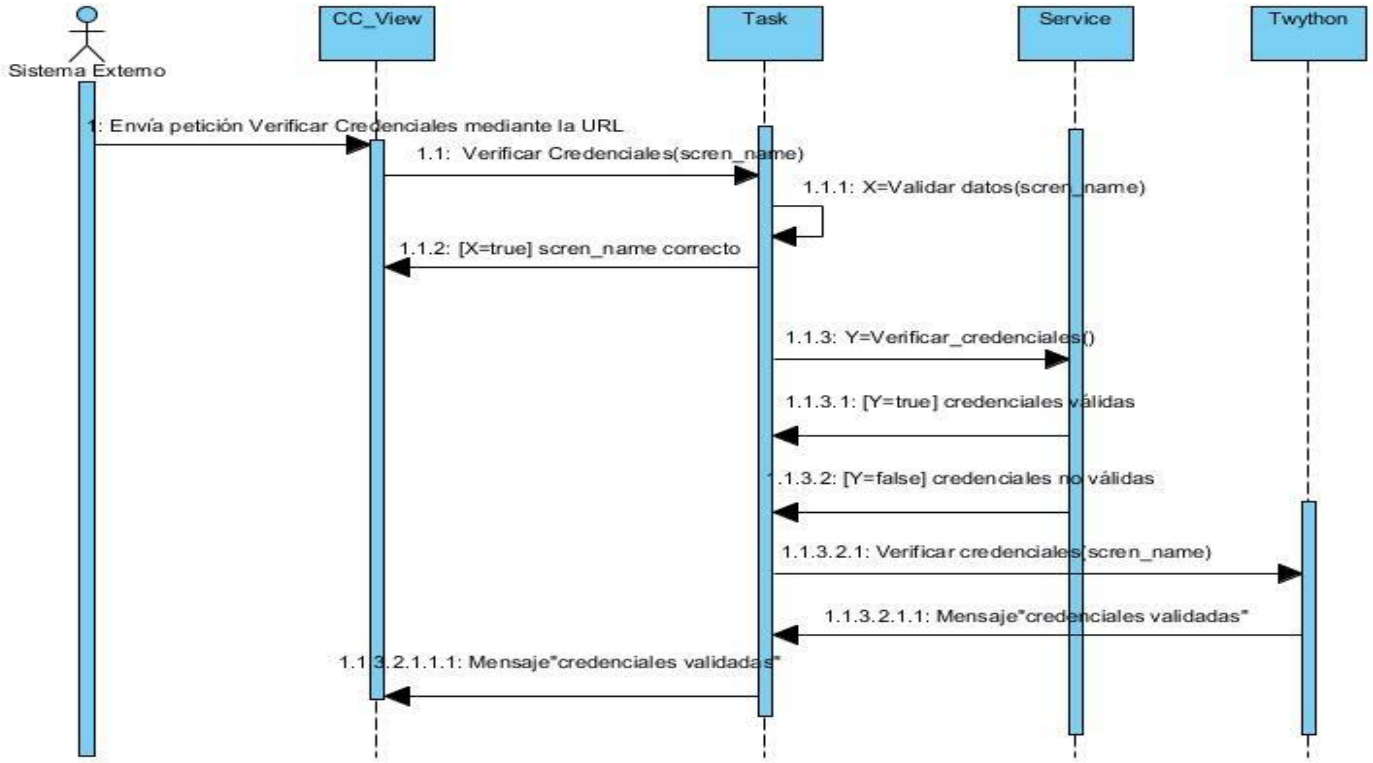
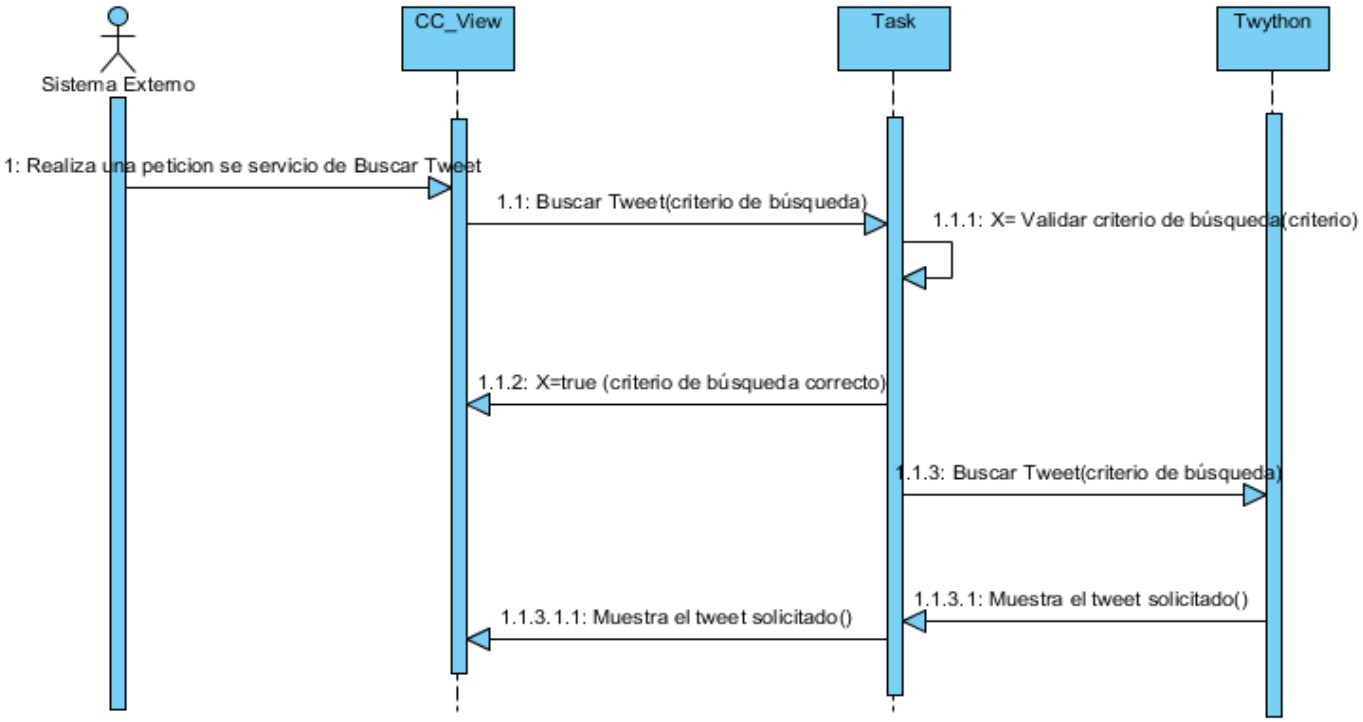


Figura 4: Diagrama de Secuencia CU_Buscar Tweet



A continuación se describe algunos de los escenarios y los casos de prueba de prioridad alta para el cliente.

Tabla 6: CP_Buscar Tweet.

Escenario	Descripción	consulta	Respuesta del sistema	Flujo central
EC 1.1 Buscar Tweet.	El sistema debe permitir al usuario realizar búsquedas de los tweet de acuerdo a un criterio y luego mostrar el resultado.	V python	El sistema verifica que el criterio de búsqueda insertada exista, si no existe, muestra la respuesta al servicio solicitado y lo almacena en la BD_MongoDB	1- El usuario solicita el servicio mediante la URL correspondiente al mismo. 2- El sistema brinda la posibilidad de introducir los datos en la interfaz del Django Rest Framework. 3- El usuario introduce como parámetro un criterio de búsqueda y selecciona el botón “Post”.
EC 1.2 Buscar Tweet sin criterio de búsqueda.	El sistema no es capaz de buscar tweet sin un criterio de búsqueda.	I sin criterio de búsqueda.	El sistema verifica que el criterio de búsqueda insertado exista, si no existe muestra un mensaje de "Error" si lo es, muestra la respuesta al servicio solicitado y lo almacena en la BD_MongoDB	

Tabla 7: CP_Buscar usuarios.

Escenario	Descripción	consulta	screen_name	Respuesta del sistema	Flujo central
EC 1.1 Buscar usuarios	El sistema debe permitir realizar búsquedas de usuarios de acuerdo a un criterio de búsquedas.	 pepetropical	V yuriesky2	El sistema verifica que el nombre de usuario insertado sea correcto, si lo es, muestra la respuesta al servicio solicitado y lo almacena en la BD_MongoDB.	1-El usuario solicita el servicio mediante la URL correspondiente al mismo. 2- El sistema brinda la posibilidad de introducir los datos en la interfaz del Django Rest Framework. 3- El usuario introduce el nombre del usuario determinado y selecciona el botón "Post".
EC 1.2 Buscar usuarios con nombres de usuario incorrectos.	El sistema no busca al usuario debido a que el nombre usuario es introducido es incorrecto.	I pepetropical3	I yurieski2	El sistema verifica que el nombre de usuario insertado sea correcto, si no lo es muestra un mensaje de "Error"	

				si lo es, muestra la respuesta al servicio solicitado y lo almacena en la BD_MongoDB.	

Tabla 8: CP_Verificar Credenciales.

Escenario	Descripción	screen_name	Respuesta del sistema	Flujo central
EC 1.1 Verificar las credenciales de un usuario autenticado.	El sistema debe permitir verificar las credenciales de un usuario autenticado.	V yuriesky2	El sistema verifica que el nombre de usuario insertado sea correcto, si lo es, muestra la respuesta al servicio solicitado y lo almacena en la BD_MongoDB.	1-El usuario solicita el servicio mediante la URL correspondiente al mismo. 2- El "sistema brinda la posibilidad de introducir los datos en la interfaz del Django Rest Framework. 3- El usuario introduce el nombre de usuario y selecciona el botón “Post”.
EC1.2 Verificar credenciales con nombre de usuario incorrecto.	El sistema no verifica las credenciales debido a que el nombre de usuario es incorrecto.	I yurieski2	El sistema verifica que el nombre del usuario insertado sea correcto, si no lo es, muestra un mensaje de "Error" si los es, muestra la respuesta al servicio solicitado y lo almacena en la BD_MongoDB.	

Tabla 9: CP_Devolver un tweet con información de sus retweet.

Escenario	Descripción	id_tweet	Respuesta del sistema	Flujo central
EC 1.1 Devolver un tweet con información de sus retweet.	El sistema debe permitir introducir el id del tweet determinado del que se quiere recibir la respuesta del servicio solicitado.	V 456123	El sistema verifica que el id del tweet insertado sea correcto, si lo es, muestra la respuesta al servicio solicitado y lo almacena en la BD_MongoDB	1-El usuario solicita el servicio mediante la URL correspondiente al mismo. 2- El sistema brinda la posibilidad de introducir los datos en la interfaz del DjangoRestFramework. 3- El usuario introduce el id del tweet determinado y selecciona el botón "Post".
EC 1.2 Devolver tweet con información de sus retweet con Id del tweet incorrecto.	El sistema devuelve los tweet con información de sus retweet debido a que el id del tweet introducido es incorrecto.	I 1234aaa	El sistema verifica que el id del tweet insertado sea correcto, si no lo es muestra un mensaje de "Error" si lo es, muestra la respuesta al servicio solicitado y lo	

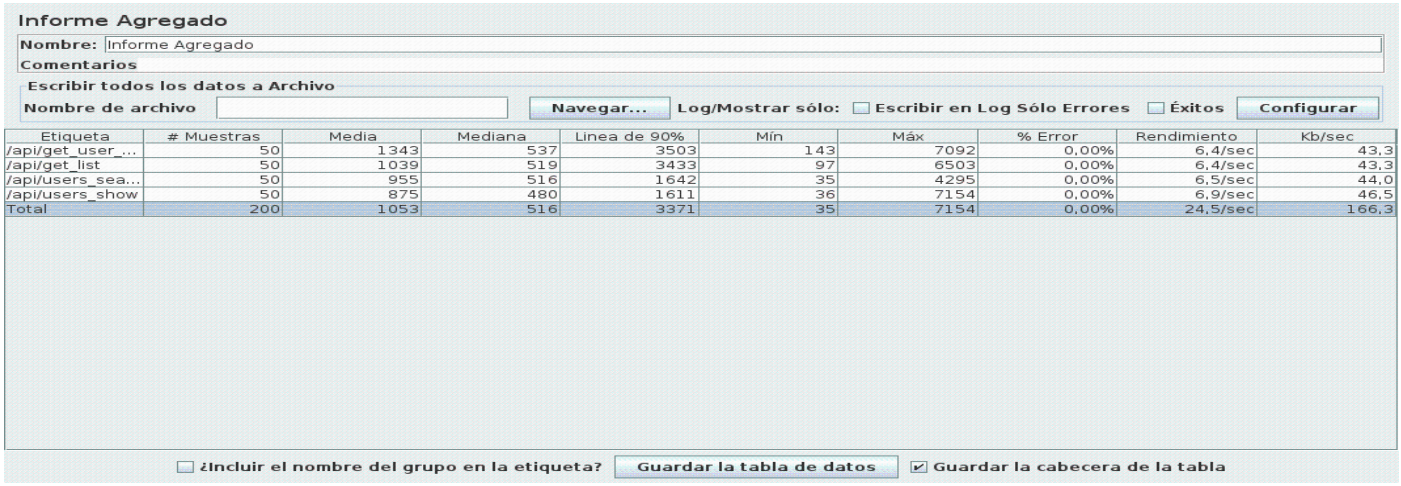
			almacena en la BD_MongoDB	

Tabla 10: Permitir seguir a un usuario especificado por el id.

Escenario	Descripción	Follow	target_ screen _name	screen_name	Respuesta del sistema	Flujo central
EC 1.1 Permitir seguir a un usuario especificado por el id	El sistema debe permitir introducir el nombre de usuario al se quiere seguir y el nombre de usuario seguidor y si el follow es true es seguidor	V	V	V	El sistema verifica que el nombre de usuario insertado sea correcto, si lo es, muestra la respuesta al servicio solicitado y lo almacena en la BD_MongoDB	1-El usuario solicita el servicio mediante la URL correspondiente al mismo. 2- El "sistema brinda la posibilidad de introducir los datos en la interfaz del Django Rest Framework. 3- El usuario introduce los datos y selecciona el botón “Post”.
		true	surysury2	yuriesky2		
EC 1.2 Permitir seguir a un usuario especificado por el id con datos de usuarios incorrectos.	El sistema no permite seguir un usuario especifico debido a que el id o el nombre de usuario es incorrecto	I	I	I	El sistema verifica que el nombre de usuario insertado sea correcto, si no lo es muestra un mensaje de "Error" si lo es, muestra la respuesta al servicio solicitado y lo almacena en la BD_MongoDB	
		false	surisuri	yurieskis2		

Informe agregado de la herramienta JMeter

Figura 5: Informe resultante de la herramienta JMeter para 50 usuarios.



Informe Agregado

Nombre: Informe Agregado

Comentarios

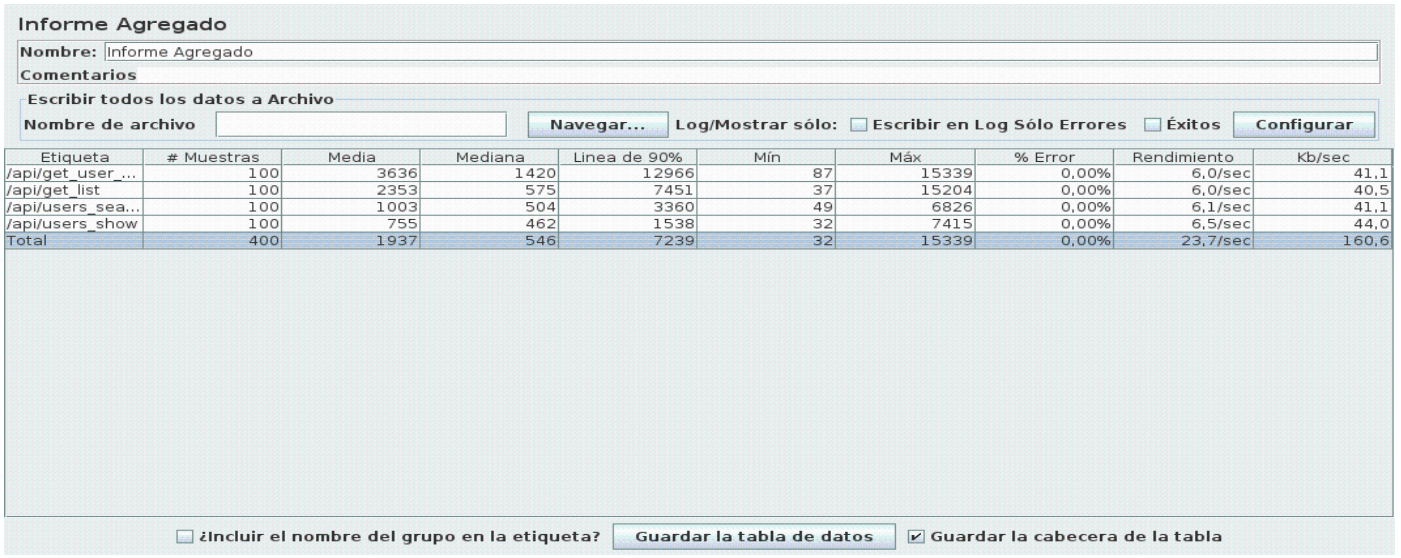
Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Mostrar sólo: Escribir en Log Sólo Errores | Éxitos | Configurar

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/api/get_user_...	50	1343	537	3503	143	7092	0,00%	6,4/sec	43,3
/api/get_list	50	1039	519	3433	97	6503	0,00%	6,4/sec	43,3
/api/users_sea...	50	955	516	1642	35	4295	0,00%	6,5/sec	44,0
/api/users_show	50	875	480	1611	36	7154	0,00%	6,9/sec	46,5
Total	200	1053	516	3371	35	7154	0,00%	24,5/sec	166,3

¿Incluir el nombre del grupo en la etiqueta? | Guardar la tabla de datos | Guardar la cabecera de la tabla

Figura 6: Informe resultante de la herramienta JMeter para 100 usuarios



Informe Agregado

Nombre: Informe Agregado

Comentarios

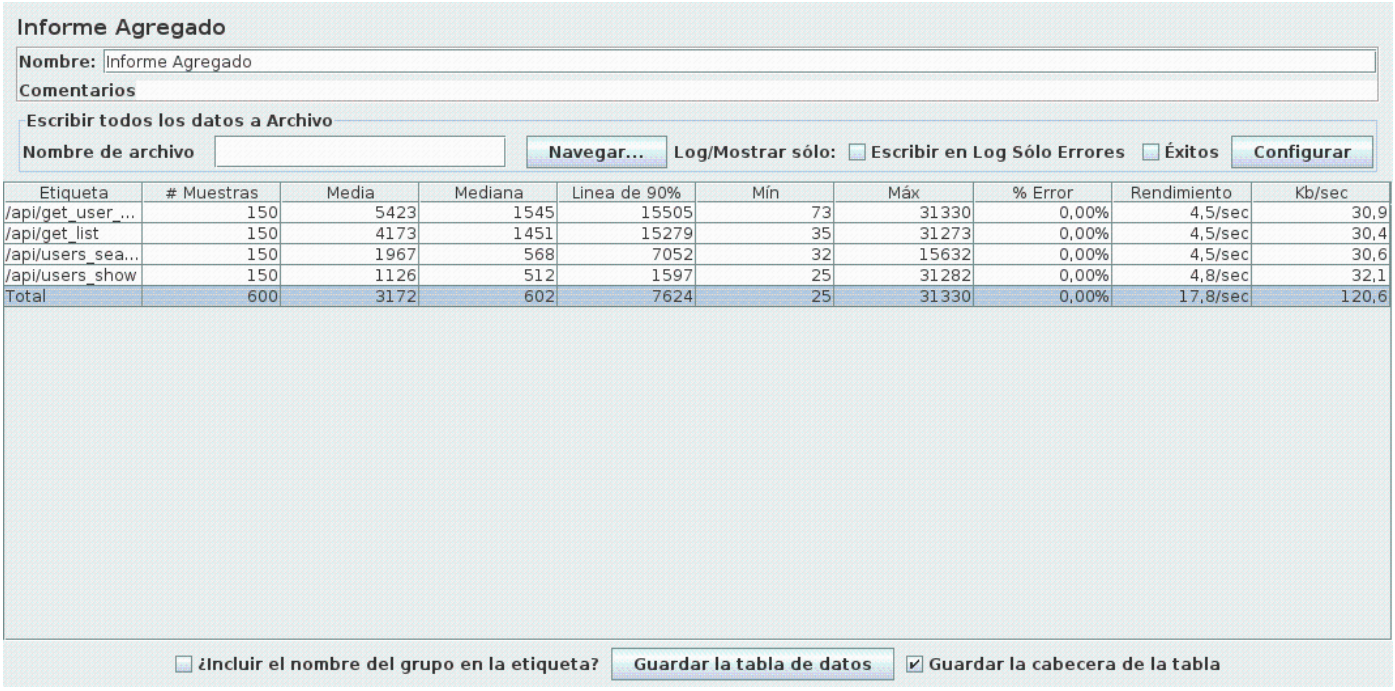
Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Mostrar sólo: Escribir en Log Sólo Errores | Éxitos | Configurar

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/api/get_user_...	100	3636	1420	12966	87	15339	0,00%	6,0/sec	41,1
/api/get_list	100	2353	575	7451	37	15204	0,00%	6,0/sec	40,5
/api/users_sea...	100	1003	504	3360	49	6826	0,00%	6,1/sec	41,1
/api/users_show	100	755	462	1538	32	7415	0,00%	6,5/sec	44,0
Total	400	1937	546	7239	32	15339	0,00%	23,7/sec	160,6

¿Incluir el nombre del grupo en la etiqueta? | Guardar la tabla de datos | Guardar la cabecera de la tabla

Figura 7: Informe resultante de la herramienta JMeter para 150 usuarios



Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
/api/get_user_...	150	5423	1545	15505	73	31330	0,00%	4,5/sec	30,9
/api/get_list	150	4173	1451	15279	35	31273	0,00%	4,5/sec	30,4
/api/users_sea...	150	1967	568	7052	32	15632	0,00%	4,5/sec	30,6
/api/users_show	150	1126	512	1597	25	31282	0,00%	4,8/sec	32,1
Total	600	3172	602	7624	25	31330	0,00%	17,8/sec	120,6