UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Subsistema para la Estandarización de la Recuperación de Información de Mensajería Instantánea en Servicios de Redes Sociales

Autor

Víctor Manuel Herrera Pérez

Tutores

Ing. Yisel Valdés Rodríguez

Ing. Julio Amado López Palma

La Habana, Junio de 2015

"Año 57 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor del presente trabajo y autorizo exclusivamente a la Universidad de las Ciencias Informáticas para su uso a conveniencia.

Para que así conste, firmo la presente a los ___ días del mes de _____ del año _____.

Víctor M. Herrera Pérez	**Ing. Yisel Valdés Rodríguez**	**Ing. Julio A. López Palma**
Firma del Autor	**Firma del Tutor**	**Firma del Tutor**

"El futuro de nuestra Patria tiene que ser necesariamente un futuro de hombres de ciencia, tiene que ser un futuro de hombres de pensamiento […]"

DEDICATORIA

A mis padres, por apoyarme incondicionalmente, por aconsejarme en los momentos difíciles, y por guiarme en cada paso que he dado en la vida.

A mis abuelos, por todo el tiempo que dedicaron a mi educación, que hicieron que me convirtiera en lo que hoy soy.

A mis tíos, por considerarme como un hijo más.

A mis hermanos y primos, con los que he compartido los mejores momentos.

AGRADECIMIENTOS

A mis tutores Yisel y Julio, gracias por sus consejos.

A mis compañeros de aula, con los que he convivido estos cinco años y de los cuales he aprendido mucho, especialmente de Castillo y Leonardo.

A todas mis amistades de la UCI.

A todo el que de una forma u otra contribuyeron al desarrollo de este trabajo.

A mi familia.

RESUMEN

El Sistema de Análisis Inteligente de *Logs* (SAIL) es una herramienta desarrollada por el departamento de Ingeniería Social Universitaria perteneciente al Centro de Ideoinformática de la Universidad de las Ciencias Informáticas. Esta solución tiene como objetivos el seguimiento, recolección, medición y análisis de la información proveniente de las bases de datos de los servidores de mensajería instantánea integrados a los servicios de redes sociales, haciendo uso para ello de métodos estadísticos. Sin embargo, la heterogeneidad estructural y funcional de las bases de datos mencionadas obstaculiza el desarrollo y funcionamiento tanto de SAIL, como de otros productos con características similares elaborados en el departamento mencionado. El presente trabajo pretende aportar un medio de comunicación que permita la interoperabilidad de forma genérica entre SAIL y las bases de datos señaladas, a partir del desarrollo de una capa de abstracción que exponga sus funcionalidades como servicios *web*. Este documento recoge los resultados de la investigación realizada, presentando también el diseño y la arquitectura del sistema propuesto, así como las tecnologías y herramientas utilizadas para su construcción. Se muestran además los artefactos generados en el proceso de desarrollo de la solución, producto de la aplicación de la metodología *OpenUP*. También son exhibidos los resultados obtenidos en la fase de pruebas de la aplicación.

Palabras clave: bases de datos, mensajería instantánea, redes sociales, servicios web.

ÍNDICE

INTRODUCCIÓN 1

CAPÍTULO #1 FUNDAMENTACIÓN TEÓRICA DE SERVICIOS *WEB* Y MENSAJERÍA INSTANTÁNEA ..6

1.1 Estudio de sistemas homólogos 6

1.2 World Wide Web 7

1.3 Servicios *web* 8

1.4 Arquitectura Orientada a Servicios 15

1.5 Servicios de Redes Sociales 15

1.6 Mensajería instantánea 17

1.7 Openfire 18

1.8 Sistema de Análisis Inteligente de *Logs* 18

1.9 Tecnologías, herramientas y metodologías 19

1.10 Conclusiones parciales 31

CAPÍTULO #2 CARACTERÍSTICAS, ANÁLISIS Y DISEÑO DEL SISTEMA 32

2.1 Propuesta del sistema 32

2.2 Modelo de dominio del sistema 33

2.3 Captura de requerimientos del sistema 34

2.4 Modelo de casos de uso 37

2.5 Arquitectura del sistema 43

2.6 Modelo de diseño 44

2.7 Patrones de diseño 44

2.8 Conclusiones parciales 47

CAPÍTULO #3 IMPLEMENTACIÓN Y PRUEBAS 48

3.1 Implementación 48
3.2 Pruebas 51
3.3 Conclusiones parciales 57
CONCLUSIONES 58
RECOMENDACIONES 59
REFERENCIAS BIBLIOGRÁFICAS 60
BIBLIOGRAFÍA CONSULTADA 64
GLOSARIO DE TÉRMINOS 65
ANEXOS 67
Anexo #1: Descripción de casos de uso del sistema 67
Anexo #2: Diagramas de clase de diseño del sistema 72
Anexo #3: Diseños de casos de prueba del sistema 74
Anexo #4: Pruebas de seguridad 79

ÍNDICE DE FIGURAS

Fig. 1 Proceso de comunicación utilizando *SOAP* 10
Fig. 2 Proceso de comunicación utilizando *REST* 11
Fig. 3 Distribución del uso de protocolos y estilos de mensajería de servicios *web* 12
Fig. 4 Propuesta del sistema 32
Fig. 5 Modelo de dominio del sistema 34
Fig. 6 Diagrama de casos de uso del sistema 38
Fig. 7 Arquitectura de la aplicación 43
Fig. 8 Diagrama de clases de diseño utilizando estereotipos *web* del CU "Obtener registros dada una consulta" 44
Fig. 9 Patrón experto en la clase *Client* del sistema 45
Fig. 10 Ejemplo de uso del patrón decorador en el sistema 46
Fig. 11 Ejemplo de uso del patrón Instancia única en el sistema 46
Fig. 12 Modelo de datos del sistema 48
Fig. 13 Diagrama de componentes del sistema 50
Fig. 14 Diagrama de despliegue del sistema 51
Fig. 15 Resultados de las pruebas funcionales 54
Fig. 16 Resultados de las pruebas de seguridad 55
Fig. 17 Resultados de las pruebas de carga y estrés 56
Fig. 18 Diagrama de clases de diseño utilizando estereotipos *web* del CU "Autenticar usuario" 72
Fig. 19 Diagrama de clases de diseño utilizando estereotipos web del CU "Gestionar aplicaciones" 72
Fig. 20 Diagrama de clases de diseño utilizando estereotipos web del CU "Gestionar usuarios" 73

ÍNDICE DE TABLAS

Tabla 1 Comparativa entre *REST* y *SOAP* 13
Tabla 2 Actores del sistema 38
Tabla 3 Definición del CU “Listar plugins instalados en el sistema” 39
Tabla 4 Definición del CU “Mostrar documentación de un plugin” 39
Tabla 5 Definición del CU “Mostrar documentación del sistema” 39
Tabla 6 Definición del CU “Autenticar usuario” 39
Tabla 7 Definición del CU “Gestionar usuarios” 39
Tabla 8 Definición del CU “Registrar cuenta de usuario” 40
Tabla 9 Definición del CU “Recuperar cuenta de usuario” 40
Tabla 10 Definición del CU “Reenviar instrucciones de confirmación de cuenta de usuario” 40
Tabla 11 Definición del CU “Listar roles de usuario” 40
Tabla 12 Definición del CU “Contactar con equipo de soporte” 40
Tabla 13 Definición del CU “Obtener registros dada una consulta” 41
Tabla 14 Definición del CU “Gestionar aplicaciones” 41
Tabla 15 Descripción del CU “Obtener registros dada una consulta” 41
Tabla 16 Entidades del modelo de datos del sistema 49
Tabla 17 Diseño de casos de prueba para el CU “Obtener registros dada una consulta” 52
Tabla 18 Descripción de las variables del diseño de casos de prueba para el CU “Obtener registros dada una consulta” 53
Tabla 19 Descripción del CU “Listar *plugins* instalados en el sistema” 67
Tabla 20 Descripción del CU “Autenticar usuario” 67
Tabla 21 Descripción del CU “Gestionar aplicaciones” 68
Tabla 22 Diseño de casos de prueba para el CU “Gestionar aplicaciones cliente” 74
Tabla 23 Diseño de casos de prueba para el CU “Autenticar usuario” 78

INTRODUCCIÓN

Con el establecimiento de la primera conexión entre computadoras en los finales de la década de 1960, surge *Internet*, también conocida como la red de redes, por la dimensión que abarca globalmente [1]. A partir de su aparición, se convirtió en precursora de grandes cambios en el mundo de la informática y las comunicaciones, influenciando en gran medida la forma de interacción, comunicación e intercambio de datos entre instituciones, empresas y la sociedad en general [2].

En la actualidad, *Internet* provee numerosos servicios que incluyen el envío de correos electrónicos, la transmisión de archivos, las conversaciones en línea y la mensajería instantánea. La expansión de estos continúa de forma acelerada, con la inclusión reciente de la transmisión de contenido y comunicación multimedia, tales como telefonía y televisión.

Uno de los servicios mencionados, la mensajería instantánea, permite la comunicación basada en texto entre varios usuarios en tiempo real. Su surgimiento devino en un avance en la forma de comunicación a través de Internet, extendiéndose su uso de forma acelerada.

También la *WWW*[1], o la *web* como comúnmente se le conoce, ha ganado una notable popularidad como servicio de *Internet*. Ésta permitió poner a disposición de sus usuarios información de todo tipo, accesible desde cualquier lugar y de forma más rápida, fácil y eficiente que la información de cualquier medio físico [3] [4].

La *web* posibilitó a los usuarios compartir su propia información. Muestra de ello son los servicios de redes sociales, los cuales permiten el intercambio de ideas, experiencias, cultura, negocios, e información de cualquier tipo en general. Todo esto es posible sin importar que tan distanciados estén los individuos que participan, tanto en tiempo como en espacio.

Debido a su éxito, la *web* sufrió un vertiginoso crecimiento, haciendo que la localización y el análisis de su contenido fueran procesos complejos. Es por ello que surgen herramientas capaces de hacer estas tareas de forma automatizada, orientadas tanto a los usuarios comunes como a las organizaciones. En el ámbito

[1] *World Wide Web*

organizacional dichas herramientas cobran especial relevancia, específicamente en el análisis de información, con vista a la toma de decisiones acertadas por parte de los directivos en busca de hacer más eficaces y eficientes los procesos que tienen lugar en las instituciones, organismos y empresas [5].

Recientemente, con la llegada de los servicios *web*, se posibilitó la combinación de las funcionalidades de las herramientas mencionadas anteriormente, a partir del intercambio de información entre ellas a través de la red, independientemente de las propiedades de cada una y la plataforma sobre la que son ejecutadas. Esto conllevó a una integración de los procesos, lo cual permitió obtener soluciones que satisficieran aún más las necesidades de usuarios y organizaciones.

Cuba, a pesar de las numerosas limitaciones a las que se ve sometida en cuanto a recursos e infraestructura tecnológica por ser una nación subdesarrollada, no está exenta de los avances a escala global en la materia. La necesaria introducción de las Tecnologías de la Información y las Comunicaciones (TIC) en las organizaciones del país ha traído consigo la necesidad de desarrollar herramientas capaces de controlar y analizar la información concerniente a dichas organizaciones, teniendo en cuenta además factores como la eficiencia, la rapidez y el alto rendimiento.

Tal es el caso del Sistema de Análisis Inteligente de *Logs* (SAIL), herramienta cuyo desarrollo está a cargo del Departamento de Ingeniería Social Universitaria (ISU), perteneciente al Centro de Ideoinformática (CIDI) de la Universidad de las Ciencias Informáticas (UCI). Esta solución tiene como objetivos el seguimiento, recolección, medición y análisis de la información proveniente de las bases de datos de los servidores de mensajería instantánea integrados a los servicios de redes sociales, haciendo uso para ello de métodos estadísticos. De esta forma se provee a los directivos de la entidad un modo de generar reportes que brinden la posibilidad de analizar los registros obtenidos y facilitar así la toma de decisiones.

Sin embargo, la heterogeneidad estructural y funcional de las bases de datos mencionadas obstaculiza el desarrollo y funcionamiento tanto de SAIL, como de otros productos con características similares elaborados en el departamento de ISU. Entre las principales limitaciones vale destacar la imposibilidad de extraer información de aquellas bases de datos que no son soportadas por los productos citados, sin un previo rediseño de estos últimos para que sean capaces de procesarlas. Esto atenta contra el carácter genérico requerido por las soluciones desarrolladas en el centro, lo cual menoscaba su posible comercialización, al no cubrir todas las necesidades de los clientes potenciales. De igual forma, los desarrolladores del

departamento se ven obligados a analizar la estructura y funcionamiento de las nuevas bases de datos a procesar, lo cual, de conjunto con lo anteriormente planteado, conlleva a un gasto adicional de tiempo y recursos en el desarrollo de las soluciones.

Teniendo en cuenta los planteamientos anteriores se identifica el siguiente **problema de investigación**: ¿cómo recuperar la información almacenada en las diferentes bases de datos de los servidores de mensajería instantánea utilizados por los servicios de redes sociales, de forma genérica, abstrayéndose de su diseño estructural?

El **objeto de estudio** comprende entonces el proceso de interacción entre los servicios de redes sociales y las aplicaciones desarrolladas por terceros que hacen uso de la información almacenada en estos, siendo el **campo de acción** del presente trabajo el proceso de interacción entre las soluciones desarrolladas por el departamento ISU y las diferentes bases de datos de los servidores de mensajería instantánea utilizados por los servicios de redes sociales.

De esta forma se plantea entonces como **objetivo general** del presente trabajo: desarrollar una aplicación que, a partir de la implementación de servicios web, provea una capa de abstracción que permita la interacción, de forma genérica, entre las soluciones informáticas desarrolladas por el Departamento de Ingeniería Social Universitaria y las diferentes bases de datos de los servidores de mensajería instantánea utilizados por los servicios de redes sociales, siendo los **objetivos específicos**:

1. Analizar el marco teórico conceptual respecto a las tecnologías actuales para el desarrollo de servicios *web*.
2. Diseñar una aplicación que a partir de la implementación de servicios *web* provea una capa de abstracción para la estandarización de la recuperación de información desde los servicios de redes sociales.
3. Implementar una aplicación que a partir de la implementación de servicios *web* provea una capa de abstracción para la estandarización de la recuperación de información desde los servicios de redes sociales.
4. Validar el correcto funcionamiento de la aplicación implementada.

Para el cumplimiento de dichos objetivos se plantean las siguientes **tareas de la investigación**:

- Análisis de la bibliografía necesaria para conocer el estado actual de las tecnologías y herramientas que se utilizan para desarrollar servicios *web*.
- Definición de la metodología a utilizar en el desarrollo de la solución propuesta.
- Identificación de necesidades de información, requisitos funcionales y no funcionales.
- Documentación de los artefactos generados durante los procesos de análisis, diseño e implementación de la solución propuesta según la metodología seleccionada.
- Implementación de los servicios *web* que permitan la abstracción e interacción de las soluciones informáticas desarrolladas por el Departamento de Ingeniería Social Universitaria con las diferentes bases de datos de los servidores de mensajería instantánea utilizados por los servicios de redes sociales.
- Aplicación de pruebas de funcionalidad a los servicios *web* desarrollados.

En consecuencia a lo anteriormente planteado, se tiene como **idea a defender** la siguiente: el desarrollo de una aplicación que mediante la implementación de servicios *web* provea una capa de abstracción, permitirá la estandarización de la recuperación de información almacenada en las bases de datos de los servidores de mensajería instantánea utilizados por los servicios de redes sociales. De esta forma se garantiza la interacción de forma genérica entre las soluciones informáticas desarrolladas por el Departamento de Ingeniería Social Universitaria y las bases de datos señaladas.

Como base para la realización de las investigaciones relacionadas con el proceso de desarrollo de la solución propuesta, se utilizan los siguientes **métodos teóricos de investigación**:

- Analítico-sintético: permite la investigación detallada acerca de los aspectos relacionados con los servicios *web*, así como la implementación de los mismos, a través del análisis de la documentación existente.
- Histórico-lógico: permite constatar teóricamente cómo ha evolucionado la utilización de los servicios *web* en el desarrollo de aplicaciones hasta la actualidad.
- Modelación: permite la representación estructural, relaciones internas y características de la solución propuesta, haciendo uso de diagramas.

El contenido del presente trabajo cuenta con la estructura definida a continuación:

Capítulo #1. Fundamentación teórica de servicios *web* y mensajería instantánea: expone los conceptos relacionados con los servicios *web* y la mensajería instantánea, así como la fundamentación de las herramientas y tecnologías que se emplean en el presente trabajo, además de la metodología que guiará el proceso de desarrollo de la solución.

Capítulo #2. Características, análisis y diseño del sistema: abarca los temas relacionados con el dominio y la caracterización del sistema, así como sus requerimientos, el diseño de la arquitectura, y el tratamiento de errores. Plantea además la propuesta de solución a implementar.

Capítulo #3. Implementación y pruebas: describe el proceso de implementación del sistema, así como documenta la validación de la solución propuesta a partir de los casos de prueba creados para tal fin.

CAPÍTULO #1 FUNDAMENTACIÓN TEÓRICA DE SERVICIOS *WEB* Y MENSAJERÍA INSTANTÁNEA

En el presente capítulo se abordan los aspectos y conceptos que fundamentan teóricamente y describen el dominio de la presente investigación. En este se expone el estado actual del uso de los servicios *web* y la mensajería instantánea, así como la evolución y las tendencias en su implementación. Se analizan además las características, ventajas y desventajas de las principales metodologías, tecnologías y herramientas existentes en la actualidad, definiendo a partir de una comparativa entre ellas, cuáles se utilizarán para el desarrollo de la solución propuesta.

1.1 Estudio de sistemas homólogos

La extracción de información desde fuentes heterogéneas no es una tarea poco común. En el contexto organizacional, es a menudo necesario fusionar datos provenientes de diferentes sistemas de origen, para su posterior análisis. Es por ello que han surgido herramientas de *software* capaces de hacer esto posible.

Dichas herramientas, categorizadas como *Middlewares*[2] para Acceso a Información (*DAM*, del inglés *Data Access Middleware*), tienen como ventajas asociadas a su uso la posibilidad de comunicarse con múltiples fuentes de datos, la conversión del lenguaje de programación de la aplicación a un lenguaje aceptado por la fuente de datos de destino y la capacidad de respuesta en un formato y lenguaje aceptable para el solicitante [6].

Las aplicaciones de extracción, transformación y carga (*ETL*, del inglés *Extract, Transform and Load*) son ejemplos de herramientas DAM. Estas permiten la extracción de datos desde diferentes fuentes, para luego transformarlos, limpiarlos y cargarlos a otra base de datos, conceptos asociados a la minería de datos. Existen numerosas soluciones de este tipo, tanto destinadas al sector empresarial con carácter privativo, como de código abierto.

Aplicaciones como *Sybase Data Integration Suite* y *Oracle Fussion Middleware* ofrecen entornos gráficos para construir, administrar y mantener el proceso de integración de datos en sistemas de inteligencia de

[2] *Software* que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones.

negocio. La principal desventaja de estas aplicaciones es que son propietarias, por lo que es obligatorio el pago por su uso. También existen alternativas libres y de código abierto. Tal es el caso de *Scriptella ETL*, *Talend Open Studio* y *CloverETL*.

En la Universidad de las Ciencias Informáticas, no solo el departamento de ISU debe lidiar con el inconveniente de la recuperación de información desde fuentes de datos heterogéneas. Otros departamentos del Centro de Ideoinformática, así como los restantes centros de la universidad han visto esta problemática como una limitación para el desarrollo de sus aplicaciones. Para solucionarla han utilizado herramientas *ETL* con características similares a las señaladas anteriormente, siendo *Pentaho*[3] la más popular.

Pese a las bondades que brindan las herramientas destacadas, ninguna es capaz de solucionar la problemática planteada en el presente trabajo cumpliendo al mismo tiempo con los requerimientos del cliente. Uno de dichos requisitos es poder disponer en tiempo real de la información, algo imposible de alcanzar actualmente utilizando productos *ETL*, pues estos están diseñados para la extracción periódica de los datos, existiendo la posibilidad de excluir información relevante cuando el cliente realice una solicitud. Otro obstáculo derivado del uso de las herramientas aludidas es la necesidad de contar con grandes medios de almacenamiento para la recopilación de la información, lo cual implica una inversión adicional en caso de no disponer de estos medios.

A partir de lo anteriormente expuesto, urge la creación de una solución que incorpore las características positivas en cuanto a diseño y funcionalidades de los productos señalados, pero que no posea las dificultades antes descritas. Para paliar estas últimas se propone proveer una capa de abstracción utilizando servicios web como base.

1.2 World Wide Web

La *World Wide Web* (*WWW*), comúnmente conocida como la *web*, es un sistema de distribución de documentos de hipertexto o hipermedios interconectados y accesibles vía *Internet*. Con un navegador *web*, un usuario visualiza sitios *web* compuestos por páginas que pueden contener texto, imágenes, vídeos u otros contenidos multimedia, y que son accesibles utilizando hiperenlaces [4].

[3] http://www.pentaho.com/

Aparejados a la *web* existen varios conceptos que hacen posible su funcionamiento. Algunos de estos son listados a continuación:

- ***Hypertext Transfer Protocol (HTTP)*:** es el protocolo utilizado en cada transacción de información a través de la *web*, siguiendo un esquema petición-respuesta entre un cliente y un servidor.
- ***Uniform Resource Identifier (URI)*:** un identificador de recursos uniforme es una cadena de caracteres que identifica un recurso en una red de forma unívoca. Se utiliza en la *web* para acceder a un recurso específico al hacer referencia a su identificador asociado.

Basándose en estos conceptos surgen los servicios *web*, a partir de los cuales se puede garantizar la interoperabilidad entre aplicaciones de *software*, por lo que el estudio de dichos servicios es de vital importancia para el presente trabajo de diploma.

1.3 Servicios *web*

Los últimos tiempos han estado marcados por el incremento de los servicios *web* disponibles en Internet. El aumento en la utilización de esta tecnología va aparejado con la creciente demanda de información a escala global. Numerosas aplicaciones en Internet ofrecen la oportunidad de acceder a su contenido a través de estos servicios, tales como *Google Search*, *Twitter* y *Facebook*, por solo mencionar algunas de ellas.

Existen múltiples definiciones sobre lo que son los servicios *web*, lo que muestra su complejidad a la hora de dar una adecuada definición que englobe todo lo que son e implican. Una posible sería hablar de ellos como un conjunto de aplicaciones o de tecnologías con capacidad para interactuar en la *web*. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios hacen llamadas a estos procedimientos a través de la *web* [7].

La interoperabilidad se consigue mediante la adopción de estándares abiertos patrocinados por las organizaciones *OASIS*[4] y *W3C*[5], comités responsables de la arquitectura y reglamentación de estos servicios.

[4] http://www.oasis-open.org/
[5] *World Wide Web Consortium* (http://w3.org/)

La principal razón para utilizarlos es que se pueden intercambiar mensajes entre aplicaciones heterogéneas a través del protocolo *HTTP* sobre *TCP*[6] por el puerto 80. Dado que las organizaciones protegen sus redes mediante *firewalls* que filtran y bloquean gran parte del tráfico de Internet, cierran la mayoría de los puertos *TCP* salvo el 80, utilizado por los navegadores *web*. Los servicios *web* utilizan este puerto para la conexión ya que no resultan bloqueados. Es importante señalar que estos se pueden utilizar sobre cualquier protocolo, sin embargo, *TCP* es el más utilizado por los motivos mencionados anteriormente.

Al conjunto de protocolos que permiten el funcionamiento de los servicios *web* se le denomina Pila de Protocolos para Servicios *Web* (*Web Service Protocol Stack*), y se distribuyen en cuatro capas fundamentales [8], caracterizadas a continuación:

Capa de Transporte

Esta capa es responsable del transporte de mensajes entre aplicaciones. Actualmente, esta capa incluye el Protocolo de Transferencia de Hipertexto (*HTTP*), el Protocolo Simple de Transferencia de Correo (*SMTP*), el Protocolo de Transferencia de Archivos (*FTP*) y otros protocolos más recientes, como *Blocks Extensible Exchange Protocol* (*BEEP*) [8].

Capa de Descripción del Servicio

Esta capa es responsable de describir la interfaz pública de un servicio *web* específico. Actualmente, la descripción del servicio se realiza a través del *Web Service Description Language* (*WSDL*) en *SOAP*, y del *Web Application Description Language* (*WADL*) en *REST* [8] [9].

Capa de Descubrimiento de Servicios

Centraliza servicios en un registro común tal que los servicios *web* de la red puedan publicar su localización y descripción, y hace que sea fácil descubrir que servicios están disponibles en la red. Actualmente, la *API*[7] *Universal Description Discovery and Integration* (*UDDI*) se utiliza normalmente para el descubrimiento de servicios [8].

[6] Protocolo de Control de Transmisión (*TCP*, del inglés *Transmission Control Protocol*)

[7] Interfaz de Programación de Aplicaciones (*API*, del inglés *Application Programming Interface*) es el conjunto de funciones, procedimientos o métodos que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción.

Capa de Mensajería

Área encargada de la codificación de los mensajes en un formato común, con la finalidad de que sean entendidos por los nodos involucrados en la comunicación. Dentro de la misma se encuentran los siguientes estándares:

- ***XML-RPC*[8]:** es un protocolo de llamada a procedimiento remoto que usa *XML* para codificar los datos y *HTTP* como protocolo de transmisión de mensajes. Es un protocolo muy simple ya que solo define unos cuantos tipos de datos y comandos útiles, además de una descripción completa de corta extensión [8].
- ***SOAP* (*Simple Object Access Protocol*):** es un protocolo basado en *XML* para el intercambio de información entre computadoras a través de una red. Funciona como una vía de transmisión entre un transmisor *SOAP* y un receptor *SOAP*, siendo necesario que los mensajes interactúen con un conjunto de aplicaciones para que se pueda generar un diálogo entre dos nodos. Los mensajes *SOAP* representan la unidad básica de comunicación entre dos nodos *SOAP* [10]. La siguiente figura muestra una representación del funcionamiento de este protocolo:

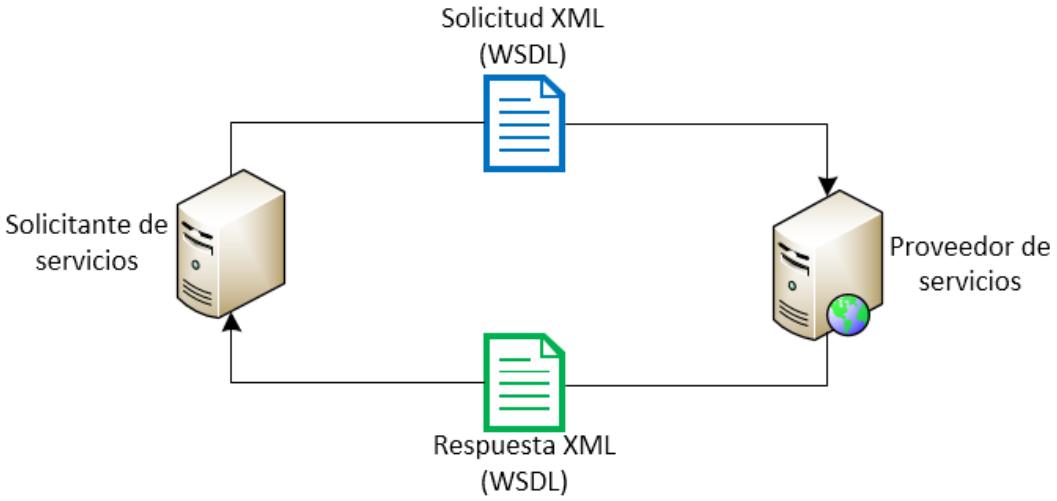

Fig. 1 Proceso de comunicación utilizando *SOAP*

[8] Llamada de Procedimiento Remoto (*RPC*, del inglés *Remote Procedure Call*)

Básicamente *SOAP* está conformado por los siguientes elementos:

- Un marco que describe el contenido de los mensajes, así como las instrucciones a ejecutar por el servicio.
- Un conjunto de reglas para representar los tipos de datos definidos.
- Convenciones para representar llamadas a procedimientos remotos y respuestas.

- ***REST* (*Representational State Transfer*):** se define como una técnica de arquitectura de *software* para sistemas hipermedia distribuidos como la *WWW* [11]. En la actualidad este término se utiliza para describir cualquier interfaz *web* simple que utilice *XML* y *HTTP* para el intercambio de información, sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, como es el caso de *SOAP*. El funcionamiento está representado por la siguiente imagen:

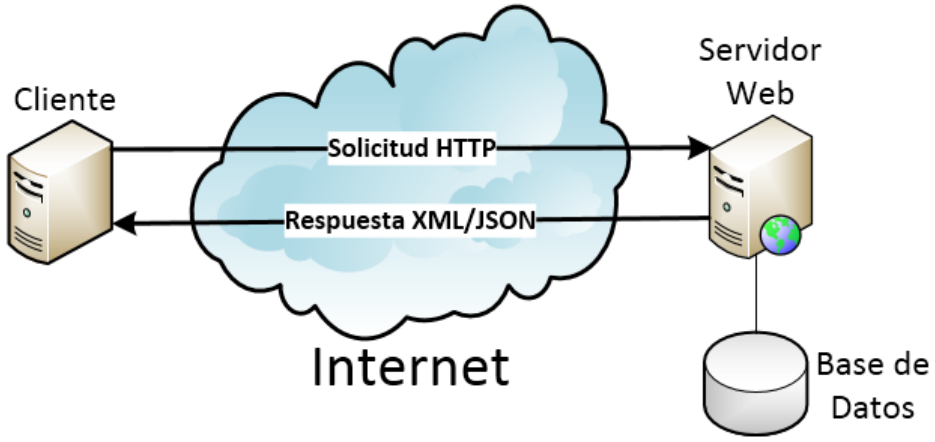

Fig. 2 Proceso de comunicación utilizando *REST*

El estándar de la capa de mensajería seleccionado para la construcción de servicios *web* determina la tipología de estos, por lo que es necesario establecer, a partir de una comparativa, el apropiado para el desarrollo de la solución propuesta en el presente trabajo.

1.3.1 Comparativa entre servicios *web* basados en *REST* y servicios *web* basados en *SOAP*

La selección del estándar de mensajería a utilizar deviene en un factor influyente a la hora de calificar el despliegue de los servicios *web*. La seguridad, la complejidad de la implementación, la extensibilidad, entre

otras, figuran como las características que comúnmente se tienen en cuenta a la hora de decantarse por uno de ellos, además de en qué medida satisface las necesidades del proyecto [12].

Según el último estudio realizado en mayo de 2011 por *ProgrammableWeb*[9], la simplicidad de la solución es uno de los principales parámetros que tienen en cuenta los diseñadores de servicios *web*. La siguiente figura muestra el comportamiento de la utilización de los estándares en el período comprendido entre el año 2006 y el 2011, tomando como muestra 3200 *APIs* almacenada en dicho directorio [13].

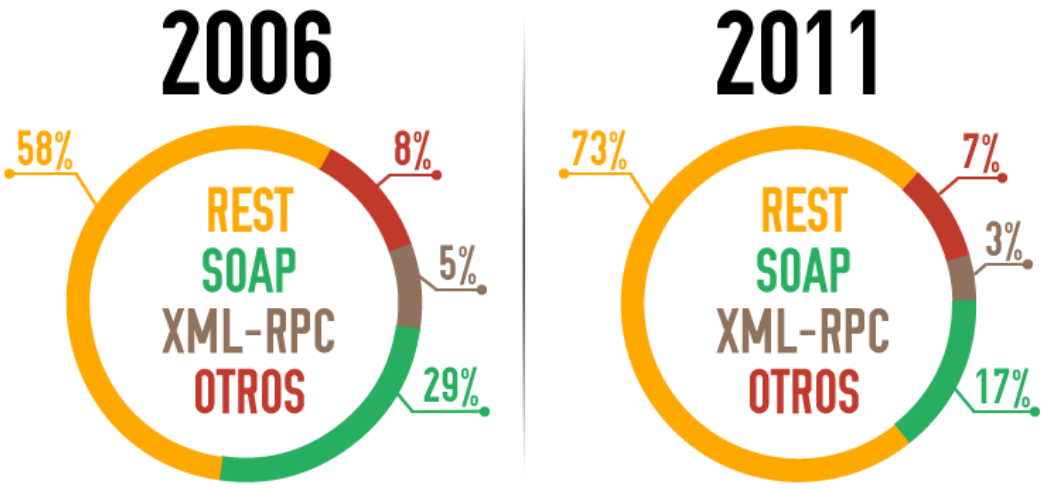

Fig. 3 Distribución del uso de protocolos y estilos de mensajería de servicios *web*

La tendencia sugiere que aumenta significativamente la utilización de *REST* en detrimento de las tecnologías restantes, debido a la facilidad de uso y escalabilidad de este. Su competidor más cercano, *SOAP*, muchas veces es criticado por los diseñadores de servicios *web* por ser un tanto complejo.

Para corroborar dicha tendencia, productos como *Twitter* y *Google Search* han hecho un extenso uso de *REST* para proporcionar acceso a sus datos, retirando sus *APIs SOAP* por ser soluciones menos versátiles para la mayoría de los casos de uso [14] [15].

La tabla mostrada a continuación refleja la comparación entre estas tecnologías [9].

[9] http://www.programmableweb.com/

Tabla 1 Comparativa entre *REST* y *SOAP*

	REST	***SOAP***
Tecnología	Mecanismo consistente en la asignación de *URIs* a los recursos.	Falta de un mecanismo de nombrado.
	Interacción dirigida por el usuario a través de formularios.	Flujo de eventos orquestados.
	Se centra en la escalabilidad y rendimiento a gran escala para sistemas distribuidos hipermedia.	Se centra en el diseño de aplicaciones distribuidas.
Protocolos que utiliza	*HTTP*	*HTTP, FTP, SMTP*
Formato de los mensajes	*XML, JSON*	*XML*
Descripción del servicio	*WADL*	*WSDL*
Seguridad	*HTTPS*	*WS-Security*
Implementación y uso	Fácil de construir y consumir	Fácil de consumir

REST se perfila como un estilo arquitectónico de comunicación entre aplicaciones mucho más sencillo y ligero que *SOAP*. Entre otras ventajas con respecto a su competidor, se puede destacar la generalidad de las interfaces, o sea, la capacidad de interacción entre cualquier cliente con cualquier servidor a partir de la utilización del protocolo *HTTP*, sin necesidad de realizar ninguna configuración especial.

Uno de los puntos débiles de *REST* radica en el tema de la seguridad. Los defensores de *SOAP* argumentan que *REST* no dispone de mecanismos tan completos de seguridad como es el caso de la especificación *WS-Security* [9]. En tal caso, una posible solución a este problema podría ser la utilización de un canal cifrado utilizando el protocolo *HTTPS*[10].

Para la selección de la tecnología a utilizar en el desarrollo de la solución propuesta por el presente trabajo, se deben tener en cuenta diversos factores, entre los que figuran la baja utilización de recursos, la escalabilidad y rendimiento para el trabajo con grandes volúmenes de datos, así como la optimización del tiempo de respuesta de la aplicación. Otro elemento a tener en cuenta es la posibilidad de acceder a los

[10] Protocolo seguro de transferencia de hipertexto (*HTTPS*, del inglés *Hypertext Transfer Protocol Secure*)

servicios expuestos por la aplicación desde múltiples sistemas, por lo que la concurrencia de acceso también juega un papel fundamental.

Teniendo en cuenta lo anteriormente expuesto, se justifica el uso de *REST* como estilo arquitectónico de comunicación para el desarrollo de la solución propuesta.

En correspondencia con la selección del estándar de mensajería a utilizar, es necesario identificar el formato de los mensajes más adecuado para el intercambio de información entre la aplicación y otros sistemas.

1.3.2 Lenguajes de intercambio de información

Para la comunicación entre los servicios *web* es necesario un lenguaje que permita el intercambio de la información. Entre estos lenguajes se pueden mencionar *XML* y *JSON*[11] como los utilizados preferentemente por los desarrolladores de servicios *web*.

El lenguaje de marcas extensible (*XML*, del inglés *Extensible Markup Language*), desarrollado por la *W3C*, es utilizado para el almacenamiento de datos de forma legible. Se deriva del lenguaje *SGML*[12] y permite definir la gramática de lenguajes específicos para estructurar documentos grandes. El uso de *XML* rebasa las fronteras de Internet, y se propone también como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede utilizar en la gran mayoría de las aplicaciones, como por ejemplo, bases de datos, editores de texto, hojas de cálculo e incluso juegos [16].

Por otro lado, *JSON* es un formato ligero para el intercambio de datos, utilizado como alternativa a *XML* para tal finalidad. Este ofrece los mismos beneficios en cuanto a interoperabilidad que *XML*, pero sin sus desventajas asociadas, lo cual le ha valido un aumento en cuanto a popularidad entre los desarrolladores [17].

Para el intercambio de información entre la solución propuesta y las aplicaciones con que debe interactuar, se decide utilizar *JSON* por su ligereza, lo cual permite optimizar el uso del canal de comunicación.

[11] *JavaScript Object Notation*

[12] Estándar de Lenguaje de Marcado Generalizado (SGML, del inglés *Standard Generalized Markup Language*)

1.4 Arquitectura Orientada a Servicios

A partir de la popularidad alcanzada por los servicios *web* como forma de intercambio de información entre sistemas heterogéneos, la utilización de una arquitectura única capaz de guiar el proceso de desarrollo, exposición y acceso a dichos servicios se ha convertido en un punto clave para lograr un correcto funcionamiento de los mismos.

Es así como surge *SOA*[13], un paradigma de arquitectura para diseñar y desarrollar sistemas distribuidos bajo distintos dominios de propiedad. *SOA* engloba un conjunto de patrones, principios y prácticas que permiten el desarrollo de soluciones que intercambien información entre sí, abstrayéndose de la tecnología utilizada para la creación de dichas soluciones [18].

Entre las principales bondades que ofrece *SOA*, valen destacar la facilidad y flexibilidad de integración con sistemas legados, la alineación directa a los procesos de negocio reduciendo costos de implementación, la innovación de servicios a clientes y una adaptación ágil ante cambios en el negocio [19]. Es por ello que su implementación en la realización de la solución propuesta servirá como garante del cumplimiento de los objetivos de la aplicación.

1.5 Servicios de Redes Sociales

Los servicios de redes sociales son medios de comunicación social que se centran en encontrar gente para relacionarse a través de *Internet*. Estos servicios están formados por personas que comparten alguna relación, mantienen intereses y actividades en común, o están interesados en explorar los intereses y las actividades de otros [20].

Hoy día, los servicios de redes sociales se han convertido en una de las aplicaciones más populares en *Internet*. Estos integran funcionalidades tales como la mensajería instantánea, los perfiles de usuario, los *blogs*, foros entre otras [21].

En correspondencia con el objetivo del presente trabajo, es necesario su estudio, tomando como muestra para ello dos de los más populares hoy día: *Facebook* y *Google+*.

[13] Arquitectura Orientada a Servicios (*SOA*, del inglés *Service Oriented Architecture*)

Facebook

Fue en su primer momento un sitio de acceso exclusivo para estudiantes de la Universidad de *Harvard*, en los Estados Unidos, pero debido a su aceptación cualquier persona que posea una cuenta de correo electrónico tiene la posibilidad de acceder. Este servicio de red social cuenta aproximadamente hasta la fecha con unos 1350 millones de usuarios activos, distribuidos por todo el mundo, así como traducciones a 70 idiomas distintos [22].

Entre sus principales servicios vale destacar los siguientes:

- Biografía: permite a los usuarios publicar información tal como imágenes, videos, enlaces, comentarios y actualizaciones de estado. Las publicaciones aparecen ordenadas de forma cronológica, lo cual mantiene organizadas las actividades de las personas de forma clara.
- Lista de amigos: permite localizar otras personas registradas y gestionar los contactos de los usuarios.
- *Chat*: este servicio permite comunicarse haciendo uso de la mensajería instantánea con los contactos del usuario, ya sea desde dispositivos móviles o computadoras.
- Grupos: permiten el intercambio de información entre personas con intereses comunes.
- Páginas: tienen como finalidad la divulgación de información relacionada con marcas, organizaciones en general o personalidades, y a diferencia de los grupos no contienen foros de discusión.
- Aplicaciones: herramientas desarrolladas por terceros que hacen uso de la *API* de Facebook para brindar otros servicios a los usuarios.

Google+

Es un servicio de red social operado por la empresa multinacional estadounidense *Google Inc.* Fue puesto en funcionamiento el 28 de Junio de 2011, alcanzando un rápido crecimiento desde entonces, con unos 20 millones de usuarios registrados a solo dos semanas de su lanzamiento [23].

Las principales funcionalidad que ofrece son las siguientes:

- Personas: permite a los usuarios organizar sus contactos en grupos, llamados círculos, para así compartir información a través de diversos productos y servicios de *Google*.
- *Hangouts*: permite la comunicación mediante mensajería instantánea y videoconferencias entre las personas añadidas a los círculos de los usuarios.
- Comunidades: funciona de forma similar a los grupos de *Facebook*.
- Al igual que otras aplicaciones de *Google*, *Google+* permite la integración con otros productos desarrollados por la empresa, tales como *Gmail*, *Google Calendar* y *Google Docs*.

De los servicios ofrecidos por ambas aplicaciones caracterizadas anteriormente, es de interés el análisis de la mensajería instantánea, por ser la fuente de información que utilizará la solución propuesta por el presente trabajo.

1.6 Mensajería instantánea

La mensajería instantánea es una forma de comunicación en tiempo real entre dos o más personas, basada en el envío de texto a través de dispositivos conectados a una red como *Internet*. Para permitir el intercambio de información, es necesario un cliente de mensajería instantánea que se conecta a un servidor a través de un protocolo de comunicación [24].

Son numerosos los clientes de mensajería instantánea que existen en la actualidad, tales como *Pidgin*, *Pandion*, *ICQ*, *Yahoo! Messenger* y *AOL Instant Messenger*, por solo mencionar algunos.

De igual forma, son varios los protocolos utilizados, predominando los propietarios como *Hangouts* de *Google*, *Bonjour* de *Apple* e *ICQ* de *AOL Inc.* Entre los protocolos abiertos, resalta *XMPP*[14], el cual está basado en *XML*, característica que le confiere adaptabilidad y sencillez [25].

Para brindar el servicio de mensajería instantánea se necesita un servidor. Entre los sistemas abiertos existentes en la actualidad para acometer esta tarea se pueden mencionar *ejabberd*, *Prosody*, *Tigase* y *Openfire*. El presente trabajo se centrará en el estudio de este último, teniendo en cuenta que la solución propuesta interactuará en su versión inicial con esta aplicación.

[14] Protocolo Extensible de Mensajería y Comunicación de Presencia (*XMPP*, del inglés *Extensible Messaging and Presence Protocol*)

1.7 Openfire

Openfire es un sistema *open source* de mensajería instantánea desarrollado con el lenguaje *Java* y con licencia *GPL* (*General Public License*), el cual utiliza el protocolo *XMPP* para la comunicación. Cuenta además con una extensión comercial denominada *Openfire Enterprise Edition* [26].

Es válido destacar que entre sus logros se encuentra que en el año 2006 ganó el premio que entrega el *ServerWatch Product* al mejor servidor de comunicaciones en tiempo real.

Además de las funcionalidades antes expuestas, también brinda características como: interfaz que posibilita agregar *plugins*, panel de administración *web*, es adaptable según las necesidades, permite la interacción con clientes como: *Yahoo! Messenger*, *AOL Instant Messenger*, *ICQ*; permite el registro de todas conversaciones entre usuarios, control total de clientes, reportes en tiempo real, integración con *VoIP*, brinda estadísticas del servidor, mensajes, paquetes, entre otros; posibilita el empleo del *P2P*, transferencia de archivos, permite la compresión de datos, tarjetas personales con *Avatar*, gestión de mensajes *offline*, autenticación vía certificados, *Kerbeos*, *LDAP*, *PAM* y *Radius*; y el almacenamiento en *Active Directory*, *LDAP*, *MS SQL*, *MySQL*, *Oracle* y *PostgreSQL* [27].

1.8 Sistema de Análisis Inteligente de *Logs*

El desarrollo del Sistema de Análisis Inteligente de *Logs* (SAIL) viene aparejado con la necesidad de seguir, recolectar, medir y analizar la información proveniente de las bases de datos de las redes sociales en Internet, a partir del uso de métodos estadísticos, para facilitar la toma de decisiones en base a la información recolectada.

SAIL automatiza una serie de procesos relacionados con la gestión del registro de información empleado por los servidores de mensajería instantánea de las redes sociales en *Internet*. Entre los procesos automatizados más importantes se encuentran:

- Definición de necesidades de información.
- Definición de las fuentes de información.
- Definición de los roles y permisos de usuarios.
- Definición de información a recolectar.
- Definición de formatos de información a recolectar.

- Adicionar, editar, listar y mostrar información.
- Búsqueda simple y avanzada.
- Seguimiento de información.
- Generación de reportes (*HTML* y *PDF*)

1.9 Tecnologías, herramientas y metodologías

En el desarrollo de *software* influyen un conjunto de elementos que permiten una correcta implementación de dicho proceso: las herramientas, tecnologías y la metodología para el desarrollo. La acertada selección de estos elementos garantiza la calidad del producto final.

1.9.1 Lenguajes de programación

Los lenguajes de programación constituyen el núcleo de la construcción de un *software*, pues son los que aportan las funcionalidades del sistema. La implementación de la solución propuesta demanda entonces un estudio de los principales lenguajes de programación orientados a la *web*, los cuales son relacionados a continuación:

***PHP* (*Hypertext Preprocessor*)**

Lenguaje de código abierto, muy popular, especialmente adecuado para el desarrollo *web* de contenido dinámico y que puede ser incrustado en documentos *HTML* [28]. Se distingue por ser un lenguaje del lado del servidor, o sea, la programación de los *scripts PHP* se ejecuta en el servidor, generando un documento *HTML*, el cual es enviado al cliente.

Lo mejor de utilizar *PHP* es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales [28].

El principal inconveniente que posee este lenguaje radica en que al ser un lenguaje interpretado, suele funcionar considerablemente más lento que otros lenguajes de bajo nivel.

Java

Lenguaje de programación de propósito general, diseñado específicamente para ser ejecutado en varias plataformas sin necesidad de recompilar el código.

Desarrollado por *Sun Microsystem* y posteriormente adquirido por *Oracle*, su sintaxis se deriva en gran medida de los lenguajes de bajo nivel *C* y *C++*, aunque no posee algunas herramientas de bajo nivel como estos, lo cual induce a varios errores [29].

La versatilidad de la tecnología *Java*, así como la portabilidad de su plataforma, y la seguridad que aporta le ha valido el aumento de su popularidad para la implementación de aplicaciones orientadas a la *web*.

Python

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, de uso general, utiliza tipado dinámico y es multiplataforma. Su filosofía hace hincapié en una sintaxis que favorezca un código legible.

Es administrado por la *Python Software Foundation*. Posee una licencia de código abierto, denominada *Python Software Foundation License*, la cual es compatible con la licencia *GPL*.

En general, *Python* cuenta con una larga lista de usuarios, así como una activa comunidad de desarrollo. Debido a su amplio uso así como más de 15 años de experiencia en su implementación, es estable y robusto. Además de ser empleado por usuarios individuales, también cuenta con el apoyo de varias organizaciones que lo utilizan en el desarrollo de productos informáticos [30]. Ejemplo de ello son:

- El sistema de búsqueda en la *web* de *Google* hace uso extensivo de *Python* en su funcionamiento.
- El servicio de compartir videos de *YouTube* está escrito en *Python*.
- Compañías como *Intel*, *Cisco*, *Hewlett-Packard*, *Seagate*, *Qualcomm*, e *IBM* utilizan *Python* para pruebas de *hardware*.
- La Agencia de Seguridad Nacional de los Estados Unidos (*NSA*) utiliza *Python* en el campo de la criptografía y el análisis de inteligencia.

Las principales ventajas de este lenguaje radican en la simpleza, claridad y sencillez de su sintaxis, así como en su portabilidad, flexibilidad, la gran cantidad de bibliotecas disponibles, su potencia y la comunidad que se mueve tras él. No es recomendado, sin embargo, para la programación de bajo nivel.

De las ventajas ofrecidas, son de interés la portabilidad y flexibilidad que *Python* le brinde a la solución a implementar. Su uso garantiza además la eficiencia, rapidez y el alto rendimiento de la aplicación, teniendo como basamento su potencia. Esto justifica el uso de este lenguaje de programación en el presente trabajo de diploma.

1.9.2 Sistemas de Gestión de Bases de Datos

Un Sistema de Gestión de Bases de Datos (SGBD) es un conjunto de programas que permite a los usuarios crear y mantener una base de datos. Por lo tanto, un SGBD es un *software* de propósito general que facilita el proceso de definir, construir y manipular bases de datos para diversas aplicaciones [31].

MySQL

Es un SGBD relacional, distribuido bajo licencia *GNU GPL*, y licencia comercial para aquellas empresas interesadas en utilizarlo con carácter privativo. *MySQL* es el SGBD de código abierto más popular, con más de 100 millones de copias del *software* descargadas y distribuidas [32].

Es ampliamente conocido y utilizado por su facilidad de uso y notable rendimiento y es además, muy confiable en términos de estabilidad, lo cual lo convierte la elección de muchos desarrolladores. También empresas de renombre en el sector de la informática, como *Google*, *Yahoo!*, *Nokia* y *Wikipedia* apuestan por el uso de *MySQL* en sus productos [32].

Entre las principales características de este SGBD destacan:

- Es multiplataforma.
- Sigue un diseño multihilo que le permite soportar una carga considerable de manera eficiente.
- Es multiusuario.
- Uso de procedimientos y vistas almacenados.

PostgreSQL

Es un SGBD objeto-relacional, desarrollado por el grupo *PostgreSQL Global Development Group.* Es *software* libre, de código abierto y multiplataforma. Cuenta con el respaldo de numerosas empresas, en cuyo listado resaltan *Cisco*, *Skype*, *Sun Microsystems*, *Red Hat* y *Apple* [33].

Entre sus características figuran:

- Alta concurrencia, siendo posible el acceso y escritura de información de forma simultánea.
- Amplia variedad de tipos de datos nativos, siendo posible añadir tipos de datos personalizados.
- Uso de procedimientos almacenados.
- Alta escalabilidad.

A raíz de la necesidad de desarrollar una aplicación donde la concurrencia y la escalabilidad son factores clave, se decide utilizar este SGBD para la implementación de la solución propuesta.

1.9.3 Lenguaje de modelado

El lenguaje de modelado es un conjunto de notaciones que incluye símbolos y sus distintas formas de estructurarlos y organizarlos. Se utiliza para la construcción de los modelos del sistema, que son la abstracción de las entidades del mundo real, y que ayudan a entenderlo y comprenderlo con todas sus características y funcionalidades [34].

Lenguaje Unificado de Modelado (*UML*)

Este lenguaje es utilizado para especificar, visualizar, construir y documentar los artefactos de los sistemas de software, así como para el modelado del negocio y otros sistemas [35].

Es un lenguaje simbólico y de notaciones, que cubre todas las vistas necesarias para desarrollar un *software*, así como las fases de análisis, diseño e implementación de este. Asegura además la interrelación entre diferentes esquemas o modelos que se comunican entre sí para brindar una mejor información del sistema a desarrollar.

1.9.4 Servidores *web*

Un servidor *web* o servidor *HTTP* es un programa informático que procesa una aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente. Estos generan una respuesta en cualquier lenguaje entendible por el cliente. Para la transmisión de todos estos datos suele utilizarse generalmente el protocolo *HTTP*, perteneciente a la capa de aplicación del modelo *OSI* [36].

Apache2

El servidor *HTTP Apache* es un servidor *web HTTP* de código abierto, para plataformas *Unix* (*BSD*, *GNU/Linux*, etc.), *Microsoft Windows*, *Macintosh* y otras, que implementa el protocolo *HTTP*/1.12 y la noción de sitio virtual [36].

El servidor *Apache* se desarrolla dentro del proyecto *HTTP Server* (*httpd*) de la *Apache Software Foundation*. Presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración [36].

Apache tiene amplia aceptación en la red: desde 1996, es el servidor *HTTP* más usado. La principal deficiencia de este servidor radica en su rendimiento, ya que por sí solo no es ligero ni rápido, factor influyente en la *web* actual.

Nginx

Es un servidor *web*/*proxy* inverso ligero de alto rendimiento y un *proxy* para protocolos de correo electrónico (*IMAP*, *POP3*). Es *software* libre y de código abierto, licenciado bajo la Licencia *BSD* simplificada. Es multiplataforma, por lo que corre en sistemas tipo *Unix* (*GNU/Linux*, *BSD*, *Solaris*, *Mac OS X*) y *Windows* [37].

El sistema es usado por una larga lista de sitios *web* conocidos, tales como: *WordPress*, *Netflix*, *Hulu*, *GitHub*, *Ohloh*, *SourceForge*, *TorrentReactor* y partes de *Facebook* (como el servidor de descarga de ficheros *zip* pesados) [37].

Entre las principales características presentes en este servidor se pueden listar [38]:

- Servidor de archivos estáticos, índices y autoindexado.
- *Proxy* inverso con opciones de caché.
- Balanceo de carga.
- Tolerancia a fallos.
- Soporte de *HTTP* sobre *SSL*.
- Soporte para *FastCGI* con opciones de caché.
- Servidores virtuales basados en nombre y/o en dirección *IP*.

- *Streaming* de archivos *FLV* y *MP4.8.*
- Soporte para autenticación.
- Compatible con *IPv6.*
- Soporte para protocolo *SPDY.*
- Compresión *gzip*.
- Habilitado para soportar más de 10.000 conexiones simultáneas.

Debido a que se centra en la utilización eficiente de los recursos, así como al alto rendimiento, es la opción ideal para garantizar el acceso al contenido estático (imágenes, hojas de estilo, *scripts*) y como *proxy* inverso de la solución propuesta.

1.9.5 Entornos Integrados de Desarrollo (*IDE*)

Un Entorno Integrado de Desarrollo (*IDE*, del inglés *Integrated Development Environment*) es una aplicación informática, compuesta por un conjunto de herramientas que facilitan el trabajo con uno o varios lenguajes de programación. Están compuestos por un editor de texto, un compilador, un intérprete y un depurador. A menudo ofrecen un sistema de control de versiones así como la factibilidad para ayudar en la construcción de interfaces gráficas de usuarios.

Existen diversos *IDEs*, los cuales difieren en sus características en dependencia del lenguaje de programación al cual dan soporte. Es por ello que consecuentemente con el lenguaje seleccionado previamente para el desarrollo de la solución propuesta se debe seleccionar un *IDE* acorde al mismo.

Eclipse

Eclipse es un *IDE* multiplataforma, de código abierto, gratuito y basado en *Java*. Fue desarrollado por la compañía *IBM* y actualmente lo desarrolla la fundación *Eclipse*. Da soporte a otros lenguajes de programación, como *C*, *C++*, *PHP* y *Python*. A su plataforma base se le pueden añadir extensiones para ampliar sus funcionalidades. Dispone de un editor de texto con resaltado de sintaxis y su compilación es en tiempo real. Posibilita el control de versiones. Presenta asistentes para la creación de proyectos y la refactorización de código [39].

PyCharm

PyCharm es un *IDE* para *Python* que presenta una asistencia y un análisis de código único para el desarrollo productivo en todos los niveles. Presenta un completo conjunto de herramientas de avanzada para el desarrollo *web* de aplicaciones en dicho lenguaje, ofreciendo compatibilidad con *frameworks* como *Django* y *Flask*, además de soportar otros lenguajes como *JavaScript*, *HTML* y *CSS*. Presenta auto-completamiento de código, señalamiento de errores o sintaxis, chequeo de errores con soluciones fáciles y un conjunto de bibliotecas de gran utilidad. Posibilita una fácil navegación para los proyectos y el código de estos debido a las vistas estructuradas, que facilitan un rápido movimiento entre archivos, clases y métodos. Mantiene el código bajo control de chequeos, asistencia de pruebas, refactorizaciones y un conjunto de inspecciones que ayudan a codificar de forma limpia y sostenible [40].

Las ventajas que ofrece este *IDE* hacen que sea el seleccionado para el desarrollo de la solución propuesta en el presente trabajo, haciendo énfasis en la compatibilidad con el *framework* de desarrollo a utilizar para la implementación de la aplicación, además de las herramientas que ofrece para el trabajo con esta, como es el caso de la herramienta de pruebas de servicios *web REST*.

1.9.6 Herramientas *CASE*

Las herramientas *CASE*[15] se utilizan para ayudar en las actividades del proceso de *software* como el análisis y el diseño. Incluyen editores de modelado, diccionarios de datos, compiladores, depuradores y herramientas de construcción de sistemas. Proporcionan ayuda al proceso de *software* automatizando algunas de sus actividades como el desarrollo de modelos gráficos. Disminuyen el tiempo de desarrollo y aumentan la productividad, minimizando el esfuerzo de codificación durante la implementación de sistemas [41].

Rational Rose

Rational Rose es una herramienta desarrollada y mantenida por *Rational Software Corporation*. Utiliza el *UML* como medio para facilitar el proceso de modelado con un conjunto de estereotipos predefinidos, teniendo la capacidad de crear, visualiza, modificar y manipular los componentes de un modelo. No es una herramienta gratuita. Habilita asistentes para crear clases y provee plantillas de código que pueden

[15] Ingeniería de *Software* Asistida por Computadora (*CASE*, del inglés *Computer Aided Software Engineering*)

aumentar significativamente la cantidad de código fuente generada. Adicionalmente, permite aplicar algunos patrones de diseño [42].

Visual Paradigm

Visual Paradigm es una herramienta multiplataforma que utiliza el *UML* y soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite realizar casi todos los tipos de diagramas, posibilitando generar código desde estos, realizar ingeniería inversa y generar documentación. Presenta licencia gratuita y comercial. Además, es apoyada por un conjunto de lenguajes como *Java*, *C++*, *PHP* y *Python* [43].

Su gratuidad y amplia cobertura de las distintas fases del proceso de desarrollo de *software*, así como las herramientas que ofrece, hacen que *Visual Paradigm* sea la opción a escoger para el desarrollo del presente trabajo.

1.9.7 *Frameworks* de desarrollo de servicios *web*

Debido a la necesidad de estandarizar las prácticas comunes, así como los criterios y conceptos a la hora de desarrollar aplicaciones de *software*, surgen los *frameworks* de desarrollo o marcos de trabajo, los cuales definen una estructura conceptual y tecnológica de soporte definido, a partir de artefactos o módulos previamente desarrollados, que sirven de base para la organización y construcción de *software* [44].

La creación de servicios *web* también está marcada por el uso de *frameworks*, pues estos permiten enfocarse en el diseño de los servicios, dedicando menos tiempo a la implementación de estos.

En la actualidad existe una gran diversidad de *frameworks* para la creación de servicios *web*, diferenciándose por el lenguaje de programación que utilizan para la implementación de dichos servicios. A continuación se relacionan algunos de estos *frameworks*:

***Zend Framework* 2 (*ZF*)**

Es un *framework* de código abierto para desarrollar aplicaciones y servicios *web* con *PHP* 5. *ZF* es una implementación que usa código 100% orientado a objetos. En la estructura de los componentes de *ZF* cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado [45].

Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de *Zend Framework* conforman un *framework* de aplicaciones web al combinarse. *ZF* ofrece una implementación *MVC*[16], una abstracción de base de datos, y un componente de formularios que implementa la prestación de formularios *HTML*, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos [45].

Provee soporte para la creación de servicios *web* basados en *SOAP*, *REST*, *XML-RPC* entre otros.

Apache Axis2

Es un motor nuclear para servicios *web*, implementado en los lenguajes *Java* y *C*. No solo provee la capacidad de agregar servicios a las aplicaciones, sino que además puede funcionar de forma autónoma como un servidor.

Es compatible con *SOAP* 1.1 y *SOAP* 1.2, además de ofrecer soporte para *REST*. Una de sus ventajas reside en la posibilidad de ofrecer interfaces tanto en *SOAP* como en *REST* de una misma implementación de la lógica del negocio.

Django REST Framework

Es una herramienta desarrollada en *Python*, para el desarrollo de servicios *web* basados en la arquitectura *REST*, de forma sencilla y rápida. Cuenta con las siguientes características [46]:

- Flexible y potente.
- Proporciona acceso a las *APIs* creadas desde el navegador *web*, mejorando la usabilidad a los desarrolladores.
- Cuenta con políticas de autenticación para el acceso a los servicios *web* implementados.
- Extensa documentación y una gran comunidad de desarrollo.

[16] Patrón Modelo-Vista-Controlador

Flask

Es un *microframework* escrito en *Python* bajo licencia *BSD*[17]. *Flask* provee, a partir de la flexibilidad ofrecida por *Python*, una arquitectura simple para el desarrollo *web*.

Al ser un *microframework*, su núcleo no provee una capa de abstracción de bases de datos, validación de formularios u otros componentes que a menudo forman parte del núcleo de los *frameworks* tradicionales. En contraste con esto, *Flask* ofrece soporte para extensiones de todo tipo que permiten la inclusión de estas características [47].

Entre sus principales bondades valen destacar:

- Cuenta con un servidor de desarrollo y un depurador integrado.
- Soporte integrado para pruebas unitarias.
- Soporte a peticiones utilizando la arquitectura *REST*.
- Extensiva documentación.
- Gran número de extensiones que permiten añadir nuevas funcionalidades a la aplicación.

Elementos como la flexibilidad, ligereza y sencillez hacen que *Flask* sea el elegido para la implementación de la solución propuesta en el presente trabajo de diploma. El uso de este *framework* hace que el código de la aplicación sea más compacto, al instalar solamente las extensiones necesarias para el correcto funcionamiento de la solución, lo cual se traduce en un mayor rendimiento.

1.9.8 Metodologías de desarrollo de *software*

Las metodologías de desarrollo de *software* son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de *software*. Estas indican los pasos a seguir y las actividades a realizar para la construcción de una solución que cumpla determinadas características, indicando además qué personas deben participar en el desarrollo de dichas actividades y qué papel deben desempeñar. Además detallan la información que se debe producir como resultado de una actividad y la necesaria para comenzarla [48].

[17] *Berkeley Software Distribution*

Estas metodologías se clasifican fundamentalmente en dos grupos: las metodologías ágiles, enfocadas en el trabajo en equipo, la interacción constante entre el equipo de desarrollo y el cliente, así como en la habilidad de responder a los cambios que puedan ocurrir; y las metodologías tradicionales, centradas en generar una documentación exhaustiva de cada proceso relacionado con el proyecto.

***Rational Unified Process* (*RUP*)**

El Proceso Unificado de *Rational* es un proceso de la ingeniería de *software* que define cómo se debe desarrollar un producto de *software*. Este trata el desarrollo de las soluciones de forma iterativa e incremental, dividiendo el proyecto en partes más pequeñas. Se caracteriza también por ser centrado en la arquitectura y guiado por casos de usos [49]. Se categoriza como una metodología tradicional.

RUP abarca cuatro fases [49]:

- Fase de inicio: Comprende la comunicación con el cliente y las actividades de planeación, llevándose a cabo la extracción y el refinamiento de los requisitos funcionales del sistema.
- Fase de elaboración: Incluye la comunicación con el cliente y las actividades de modelado del análisis y el diseño, con énfasis en las definiciones de clases y representaciones arquitectónicas.
- Fase de construcción: En esta fase se refinan y traducen los modelos del análisis y el diseño en componentes de *software* ya implementados.
- Fase de transición: En este momento se transfiere el *software* del desarrollador al cliente para realizar las pruebas y obtener la aceptación de este.

RUP también define los roles que desempeña una persona en un determinado momento. Esta metodología es apropiada para proyectos grandes, dado que requiere un equipo de trabajo capaz de administrar un proceso complejo en varias etapas [50].

***eXtreme Programming* (*XP*)**

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de *software*, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. *XP* se basa en la retroalimentación continua 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. *XP* se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico [51].

En *XP* los requerimientos, parte fundamental de todo programa a desarrollar, son tomados como escenarios y se les conoce como historias del usuario, que posteriormente son divididas en una serie de tareas. En *XP* los programadores trabajan en parejas y desarrollan pruebas para cada tarea, para posteriormente generar código nuevo que debe ejecutarse satisfactoriamente. Una vez que las pruebas son superadas, el código se integra al sistema [52].

OpenUP

OpenUP es un proceso de desarrollo de *software* que busca la unión de las principales ventajas de las metodologías tanto tradicionales como las ágiles. Por lo tanto no provee lineamientos para todos los elementos que se manejan en un proyecto, al igual que las metodologías tradicionales, pero tiene los componentes básicos que pueden servir de base a procesos específicos, limitación muy común en las metodologías ágiles [53].

La mayoría de los elementos de *OpenUP* están declarados para fomentar el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto, sus objetivos, alcance y avances [53].

Como metodología híbrida, el objetivo de *OpenUP* es un desarrollo ágil del producto, pero a la misma vez, documentando las principales actividades que ocurren en dicho proceso. De esta forma, no es necesario generar todos los artefactos requeridos por las metodologías tradicionales, y a diferencia de las metodologías ágiles, se hace uso de elementos considerados innecesarios en estas últimas.

Para el desarrollo de la solución propuesta en el presente trabajo de diploma se selecciona esta metodología, pues posee las siguientes ventajas que permiten guiar satisfactoriamente el desarrollo de la aplicación:

- Sigue un proceso iterativo e incremental, lo cual permite la detección temprana de posibles errores.
- No define un modelo de negocio ni de dominio necesario, por lo que es adaptable a posibles cambios que pudieran surgir.

- Evita la elaboración de documentación innecesaria, por lo que solamente se contará con los artefactos necesarios para guiar el proceso de desarrollo.
- Tiene un enfoque centrado en el cliente.

1.10 Conclusiones parciales

En este capítulo se estudiaron temas como los antecedentes, definiciones, características y tecnologías relacionadas con los servicios *web* y la mensajería instantánea. Se obtuvieron además los elementos que conforman la fundamentación teórica del presente trabajo de diploma, a partir del estudio de soluciones existentes. Dicho estudio permitió también identificar los elementos que conforman la base tecnológica de la solución propuesta, siendo seleccionados *Python* 2.7.6 como lenguaje de programación, *PostgreSQL* 9.3 como SGBD, *UML* 2.1 como lenguaje de modelado, *Nginx* 1.4.6 como servidor *web* y *proxy* inverso de la aplicación, *PyCharm* 4.3 como *IDE*, *Visual Paradigm* 5 como herramienta *CASE*, el *framework* de desarrollo a utilizar será *Flask* 0.10.1 y la metodología de desarrollo de *software* a seguir será *OpenUP*.

CAPÍTULO #2 CARACTERÍSTICAS, ANÁLISIS Y DISEÑO DEL SISTEMA

El presente capítulo tiene como propósito abordar las características del sistema con la finalidad de concretar su diseño, de forma tal que se garantice el cumplimiento de los objetivos trazados. Para ello se modelan los procesos y elementos involucrados en el desarrollo y el funcionamiento del *software*. Se hace además una relatoría de los requerimientos con que debe cumplir la aplicación.

2.1 Propuesta del sistema

La capa de abstracción a implementar recibirá peticiones enviadas por terceros sistemas. Estas peticiones incluyen consultas parametrizadas que permiten ajustar la configuración de la extracción de datos provenientes de los servidores de mensajería instantánea vinculados a los servicios de redes sociales. A partir de la solicitud, el sistema genera un mensaje de respuesta con la información requerida.

Los desarrolladores de sistemas externos deberán implementar una *API* acorde a la plataforma en que estos estén definidos. Esta *API* será la encargada de la comunicación con la solución propuesta, haciendo uso de *JSON* como lenguaje de comunicación para la transacción de información.

La siguiente figura expresa de forma gráfica el funcionamiento básico de la aplicación:

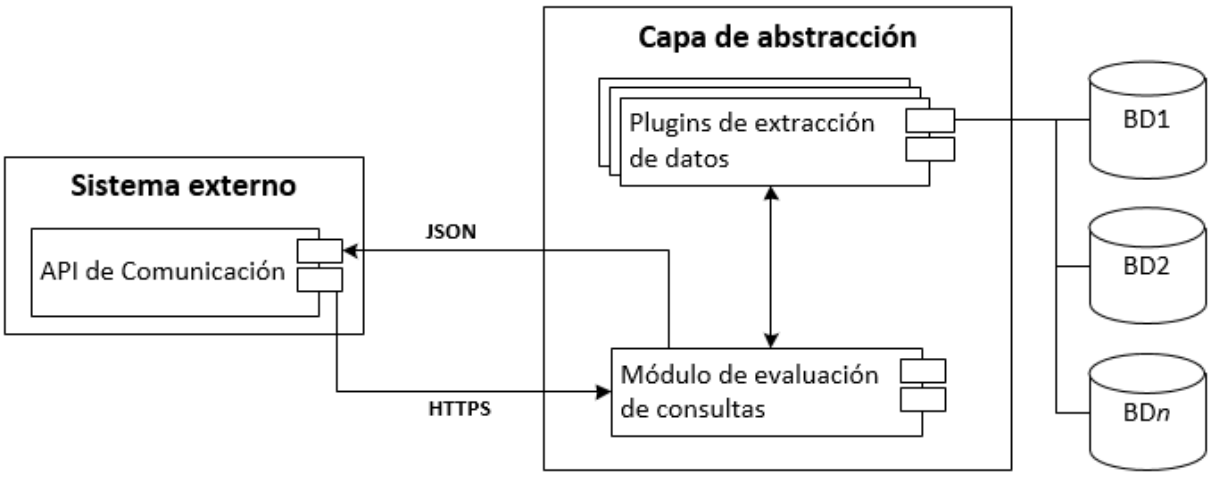

Fig. 4 Propuesta del sistema

A continuación se explican los elementos involucrados en el funcionamiento del sistema:

- **Sistema externo:** representa aquella aplicación que necesita acceder a la información almacenada en las bases de datos de mensajería instantánea de los servicios de redes sociales. Incluye la *API* de comunicación.
- **API de Comunicación:** permite el intercambio de información entre el sistema externo que la implementa y la capa de abstracción. A partir de su uso, se envía una petición a la aplicación a través del protocolo *HTTPS*, recibiendo como respuesta la información solicitada en formato *JSON*.
- **Módulo de evaluación de consultas:** se encarga de procesar las consultas hechas al sistema y devolver una respuesta a las aplicaciones externas en formato *JSON*. Evalúa los parámetros recibidos, ejecutando los *plugins* de la aplicación que permitan obtener los datos solicitados.
- ***Plugin* de extracción de datos:** contiene las funcionalidades para la extracción de información de las bases de datos de los servicios de redes sociales.

La configuración y mantenimiento del sistema requiere además de una interfaz que permita realizar estas tareas de forma gráfica. A esta tendrán acceso los administradores, que se encargarán de la gestión de los componentes de la aplicación, mientras que los desarrolladores la deberán utilizar para registrar los sistemas externos en los que se hará uso de los servicios provistos por la capa de abstracción.

2.2 Modelo de dominio del sistema

El modelo de dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés [54]. Su uso posibilita ilustrar los principales conceptos con los que trabaja el sistema a desarrollar, permitiendo que se obtenga una mejor comprensión del entorno y una representación visual de los elementos más significativos dentro del contexto de interés.

La siguiente figura muestra el diagrama de clases del modelo de dominio definido para el sistema a desarrollar:

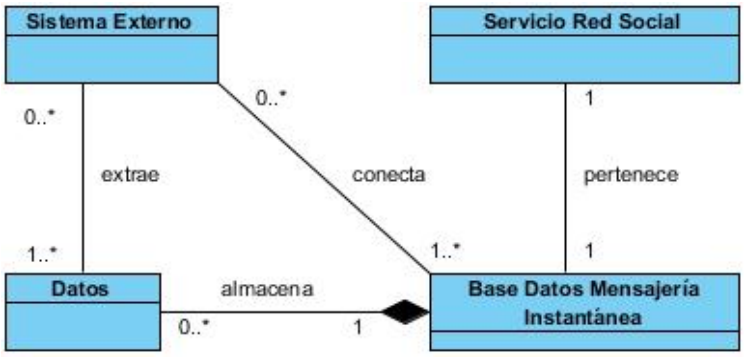

Fig. 5 Modelo de dominio del sistema

La solución propuesta en el presente trabajo surge en el contexto de la necesidad de extraer, de forma genérica, la información almacenada en las bases de datos de los servidores de mensajería instantánea vinculados a los servicios de redes sociales.

A continuación se explican los conceptos asociados a las clases fundamentales del diagrama:

- **Sistema externo:** aplicación informática que hará uso de los datos almacenados en los servidores de mensajería instantánea vinculados a los servicios de redes sociales.
- **Base de datos de mensajería instantánea:** representa las bases de datos utilizadas por los servidores mencionados.
- **Servicio de red social:** medio de comunicación social, que engloba un conjunto de tecnologías que permiten la interrelación de personas a través de *Internet*.

2.3 Captura de requerimientos del sistema

Los requisitos son capacidades y condiciones con las cuales debe ser conforme el sistema, y más ampliamente, el proyecto. Estos se dividen en requisitos funcionales, los cuales expresan el comportamiento de la aplicación; y requisitos no funcionales, que representan restricciones o condicionantes del proyecto [54].

A continuación se relacionan los requisitos definidos para la solución propuesta:

2.3.1 Requisitos funcionales

RF1. Listar *plugins* instalados en el sistema.

RF2. Mostrar documentación de un *plugin*.

RF3. Mostrar documentación del sistema.

RF4. Autenticar usuario.

RF5. Crear usuario.

RF6. Eliminar usuario.

RF7. Listar usuarios existentes.

RF8. Modificar usuario existente.

RF9. Registrar cuenta de usuario.

RF10. Recuperar cuenta de usuario.

RF11. Reenviar instrucciones de confirmación de cuenta de usuario.

RF12. Listar roles de usuario.

RF13. Contactar con equipo de soporte.

RF14. Obtener registros dada una consulta.

RF15. Registrar aplicación cliente.

RF16. Eliminar aplicación cliente registrada.

RF17. Listar aplicaciones cliente registradas.

RF18. Modificar aplicación cliente registrada.

2.3.2 Requisitos no funcionales

Usabilidad

RnF 1. La solución debe proveer sus funcionalidades como servicios *web*.

RnF 2. La solución debe permitir recuperar los registros almacenados en las bases de datos de los servidores de mensajería instantánea utilizados por los servicios de redes sociales.

RnF 3. Los requisitos mínimos de *hardware* para la ejecución de la aplicación serán: *CPU Core2Duo* 2.0 GHz o superior, 2 Gb *RAM* o superior, 5 Gb de espacio libre en disco.

RnF 4. Para la ejecución de la aplicación deberán estar instaladas las siguientes aplicaciones: *Ubuntu Server* 14.04 *LTS*, *Python* 2.7.6.

Confiabilidad

RnF 5. Solamente los usuarios registrados pueden modificar la configuración de la aplicación.

RnF 6. El acceso a los servicios provistos por la solución será a través de canales cifrados utilizando los protocolos *HTTPS* y *SMTPS*.

RnF 7. La aplicación debe utilizar mecanismos de seguridad ante ataques *DDoS*, *CSRF*, *XSS* e Inyecciones *SQL*.

Eficiencia

RnF 8. El sistema debe soportar hasta 100 transacciones por segundo.

RnF 9. El sistema debe soportar hasta 100 conexiones concurrentes.

Soporte

RnF 10. El sistema estará bien documentado de forma tal que el tiempo de mantenimiento sea mínimo en caso de necesitarse.

Restricciones de diseño

RnF 11. Las restricciones de diseño de la aplicación serán:

- Lenguaje de programación: El lenguaje definido para el desarrollo de la solución es *Python*.
- Sistema operativo: La aplicación se desarrolla sobre el sistema operativo *Ubuntu* 14.04 *LTS*.
- Control de versiones: Como sistema de control de versiones se utiliza el *Subversion* (*SVN*).
- El sistema debe ser multiplataforma, extensible, modular y utilizar el patrón arquitectónico de N capas.
- El sistema se desarrolla utilizando el *framework* de desarrollo *Flask* 0.10.1.

Requisitos para la documentación de usuarios en línea y ayuda del sistema

RnF 12. Los manuales de usuario y de instalación del sistema deben hacerse teniendo en cuenta el estándar *IEEE Std*. 1063-2001 *(IEEE Standard for Software User Documentation*).

Interfaz

RnF 13. Debe utilizarse un patrón que permita estructurar y hacer intuitiva la navegación en la interfaz de usuario.

RnF 14. La interfaz debe tener un diseño uniforme para todos los usuarios.

Requisitos legales, de derechos de autor y otros

RnF 15. Las herramientas seleccionadas para el desarrollo del producto deben estar respaldadas por licencias libres, bajo las condiciones de *software* libre.

2.4 Modelo de casos de uso

El presente epígrafe evidencia los casos de uso del sistema propuesto, la descripción de estos, así como los actores involucrados.

2.4.1 Definición de los casos de uso del sistema

Los casos de uso (CU) engloban los requisitos funcionales de la aplicación, y definen la forma en que se comportará un sistema [54]. El diagrama de casos de uso permite ilustrarlos, así como a los actores y las relaciones entre ellos.

A continuación se muestra dicho diagrama correspondiente al sistema a implementar:

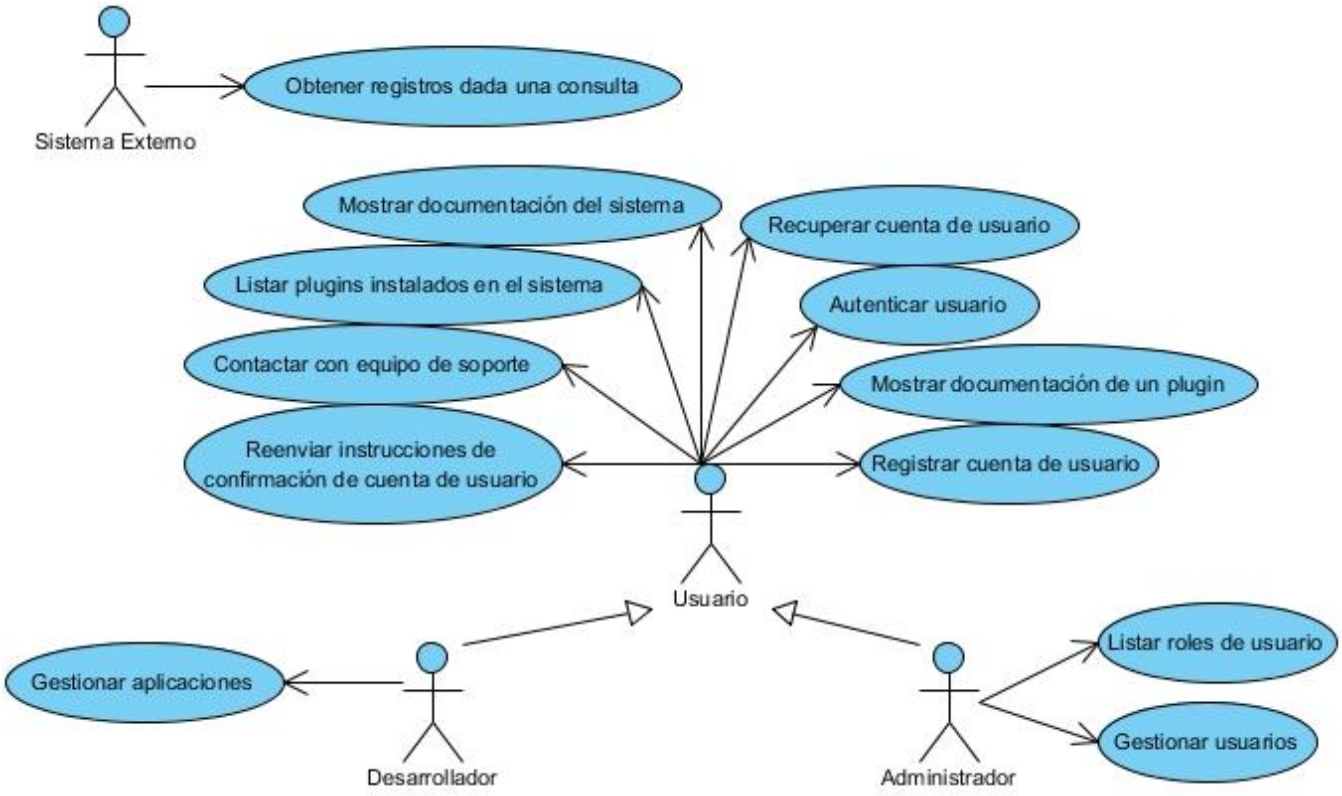

Fig. 6 Diagrama de casos de uso del sistema

Los actores que interactúan con el sistema representan los roles que asumen una o varias personas, u otros sistemas informáticos. En este caso, los actores que participan en los procesos de la solución propuesta son descritos a continuación:

Tabla 2 Actores del sistema

Actor	Descripción
Sistema Externo	Es el sistema desarrollado por terceros que consume los servicios brindados por la aplicación para obtener la información de las bases de datos de los servidores de mensajería instantánea utilizados por los servicios de redes sociales.
Desarrollador	Representa el desarrollador de sistemas externos interesado en hacer uso de los servicios provistos por la aplicación.
Administrador	Es el usuario encargado de la configuración y mantenimiento de la aplicación.

Se identifican entonces como casos de uso del sistema los siguientes:

Tabla 3 Definición del CU "Listar plugins instalados en el sistema"

CU1	Listar *plugins* instalados en el sistema
Actor	Administrador, Desarrollador
Descripción	Muestra la información relacionada con los *plugins* instalados en el sistema.
Referencia	RF1

Tabla 4 Definición del CU "Mostrar documentación de un plugin"

CU2	Mostrar documentación de un *plugin*
Actor	Administrador, Desarrollador
Descripción	Muestra la documentación asociada a las funcionalidades de un *plugin* instalado en el sistema.
Referencia	RF2

Tabla 5 Definición del CU "Mostrar documentación del sistema"

CU3	Mostrar documentación del sistema
Actor	Administrador, Desarrollador
Descripción	Muestra la documentación del sistema.
Referencia	RF3

Tabla 6 Definición del CU "Autenticar usuario"

CU4	Autenticar usuario
Actor	Administrador, Desarrollador
Descripción	Permite el acceso a las funcionalidades restringidas de la aplicación.
Referencia	RF4

Tabla 7 Definición del CU "Gestionar usuarios"

CU5	Gestionar usuarios
Actor	Administrador
Descripción	Permite la creación, eliminación, modificación y visualización de los usuarios que accederán al sistema.
Referencia	RF5, RF6, RF7, RF8

Tabla 8 Definición del CU "Registrar cuenta de usuario"

CU6	Registrar cuenta de usuario
Actor	Administrador, Desarrollador
Descripción	Permite el registro de usuarios anónimos en el sistema.
Referencia	RF9

Tabla 9 Definición del CU "Recuperar cuenta de usuario"

CU7	Recuperar cuenta de usuario
Actor	Administrador, Desarrollador
Descripción	Permite al usuario obtener acceso a su cuenta en caso de olvidar la contraseña.
Referencia	RF10

Tabla 10 Definición del CU "Reenviar instrucciones de confirmación de cuenta de usuario"

CU8	Reenviar instrucciones de confirmación de cuenta de usuario
Actor	Administrador, Desarrollador
Descripción	Permite al usuario recibir nuevamente las instrucciones de confirmación de creación de su cuenta.
Referencia	RF11

Tabla 11 Definición del CU "Listar roles de usuario"

CU9	Listar roles de usuario
Actor	Administrador
Descripción	Permite a los administradores del sistema visualizar los roles disponibles para ser asignados a los usuarios.
Referencia	RF12

Tabla 12 Definición del CU "Contactar con equipo de soporte"

CU10	Contactar con equipo de soporte
Actor	Administrador, Desarrollador
Descripción	Permite el envío de mensajes al equipo de soporte del sistema.
Referencia	RF13

Tabla 13 Definición del CU "Obtener registros dada una consulta"

CU11	Obtener registros dada una consulta
Actor	Sistema externo
Descripción	Permitir la recuperación de información desde los servidores de mensajería instantánea utilizados por los servicios de redes sociales, a partir de la recepción de una petición hecha por un sistema externo.
Referencia	RF14

Tabla 14 Definición del CU "Gestionar aplicaciones"

CU12	Gestionar aplicaciones
Actor	Desarrollador
Descripción	Permite el registro, eliminación, modificación y visualización de los terceros sistemas que interactuarán con la aplicación.
Referencia	RF15, RF16, RF17, RF18

2.4.2 Descripción de los casos de uso del sistema

A continuación se muestra la descripción de uno de los casos de uso del sistema a implementar:

Tabla 15 Descripción del CU "Obtener registros dada una consulta"

Caso de uso	Obtener registros dada una consulta.	
Objetivo	Permitir la recuperación de información desde los servidores de mensajería instantánea utilizados por los servicios de redes sociales, a partir de la recepción de una petición hecha por un sistema externo.	
Actores	Sistema externo (inicia).	
Resumen	Se inicia cuando un sistema externo envía una petición a la aplicación solicitando los datos almacenados en las bases de datos de los servidores de mensajería instantánea utilizados por los servicios de redes sociales. La aplicación evalúa la consulta, extrae los datos y devuelve una respuesta, terminando así el caso de uso.	
Complejidad	Alta.	
Prioridad	Alta.	
Precondiciones	El sistema externo debe estar autorizado para acceder a los datos que solicita.	
Poscondiciones	Se devuelven los registros solicitados por la aplicación externa.	
Flujo de eventos		
Flujo básico "Obtener registros dada una consulta"		
	Actor	**Sistema**

1.	Envía una petición utilizando la dirección *URI* "/api/search" provista por el sistema, utilizando el método *HTTP GET* y uno o varios de los siguientes parámetros: ***fromid* (cadena de texto):** nombre de usuario del remitente ***toid* (cadena de texto):** nombre de usuario del destinatario ***sentdate* (entero):** fecha de envío del mensaje en milisegundos (*timestamp*) ***body* (cadena de texto):** lista de palabras clave contenidas en los mensajes ***conversationid* (entero):** identificador de la conversación ***count* (entero):** cantidad de mensajes a obtener ***plugins* (cadena de texto):** lista de *plugins* a ejecutar ***friends* (cadena de texto):** lista de usuarios de los que se desea obtener la lista de amigos	Verifica la petición, y en caso de ser correcta, realiza una consulta a las bases de datos de los servidores de mensajería instantánea, teniendo en cuenta los parámetros recibidos para filtrar la búsqueda. Ejecuta los *plugins* instalados para la extracción de datos y recopila la información devuelta por cada uno, devolviendo en una respuesta al sistema externo dicha información en formato *JSON*.
2.		Finaliza el CU.
Flujos alternos		
No. 1 "Petición utilizando un método distinto a *GET*"		
1.	Envía la petición utilizando un método distinto a *GET*.	Devuelve un mensaje de error en formato *JSON* con estado 405 "*Method Not Allowed*", indicando que el método no está permitido.
No. 2 "Parámetros inválidos"		
1.	Envía la petición violando la restricción del tipo de dato de al menos un parámetro especificado.	Devuelve un mensaje de error en formato *JSON* con estado 400 "*Bad Request*", indicando que el parámetro debe cumplir con la restricción.
No. 3 "Cliente no autorizado"		
1.	Envía la petición sin tener autorización previa del sistema.	Devuelve un mensaje de error en formato *JSON* con estado 401 "*Unauthorized*", indicando que el sistema externo que intenta acceder no está autorizado para interactuar con la aplicación.
No. 4 "Parámetros no especificados"		
1.	Envía la petición sin especificar parámetros.	Devuelve todos los registros sin filtrar obtenidos por los *plugins* de extracción de datos.

Relaciones	**CU Incluidos**	Ninguno
	CU Extendidos	Ninguno

El Anexo #1 del presente documento muestra otras descripciones de casos de uso asociadas al sistema.

2.5 Arquitectura del sistema

La arquitectura de un *software* alude a la estructura global de este y a las formas en que dicha estructura proporciona la integridad conceptual de un sistema. En su forma más simple, la arquitectura es la estructura jerárquica de los componentes de un programa, la manera en que estos interactúan y la estructura de datos que van a utilizar dichos componentes [55].

Para el desarrollo de la solución propuesta se decide utilizar el estilo arquitectónico N-Capas. Este se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva del problema a resolver. Los roles indican el tipo y la forma de interacción entre las capas, mientras que las responsabilidades la funcionalidad que implementan. Este estilo descompone los servicios de forma que la mayoría de las interacciones ocurren solo entre capas vecinas y cada capa solo contiene la funcionalidad relacionada con las tareas que debe llevar a cabo. Además, las capas inferiores son independientes de las capas superiores [56].

El uso de este estilo permite aprovechar las siguientes ventajas:

- ✓ Abstracción: no es necesario conocer el funcionamiento interno de otras capas, solo la forma de interacción con estas.
- ✓ Aislamiento: los cambios internos de una capa no afectan el sistema completo.

La aplicación a implementar cuenta con tres capas, descritas a continuación:

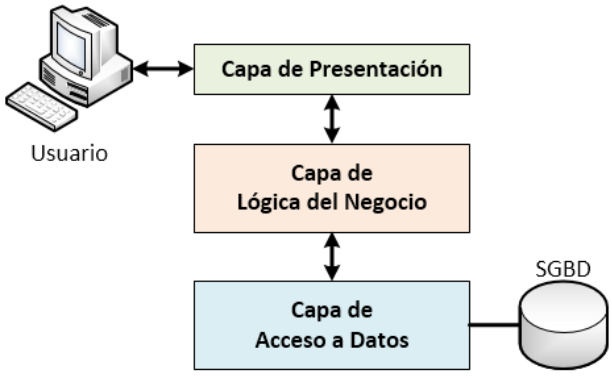

Fig. 7 Arquitectura de la aplicación

Capa de Presentación: encargada de interactuar con el usuario, con el objetivo de mostrar o capturar información. Se comunica únicamente con la capa de Lógica del Negocio.

Capa de Lógica del Negocio: se encarga de recibir las peticiones hechas por los usuarios, así como de emitir una respuesta a estas. Esta capa se comunica con la capa de Presentación, para recibir las solicitudes y presentar los resultados, y con la capa de Acceso a Datos, para solicitar al gestor de bases de datos almacenar o recuperar datos de él.

Capa de Acceso a Datos: es la encargada de manipular los datos de la aplicación.

2.6 Modelo de diseño

El modelo de diseño comprende las representaciones de los datos, arquitectura, interfaces y componentes involucrados en el funcionamiento de un *software* [55]. Para el modelado se utiliza el diagrama de clases de diseño, el cual ilustra las especificaciones de las clases de la aplicación

El siguiente diagrama esboza el diseño del CU “Obtener registros dada una consulta”:

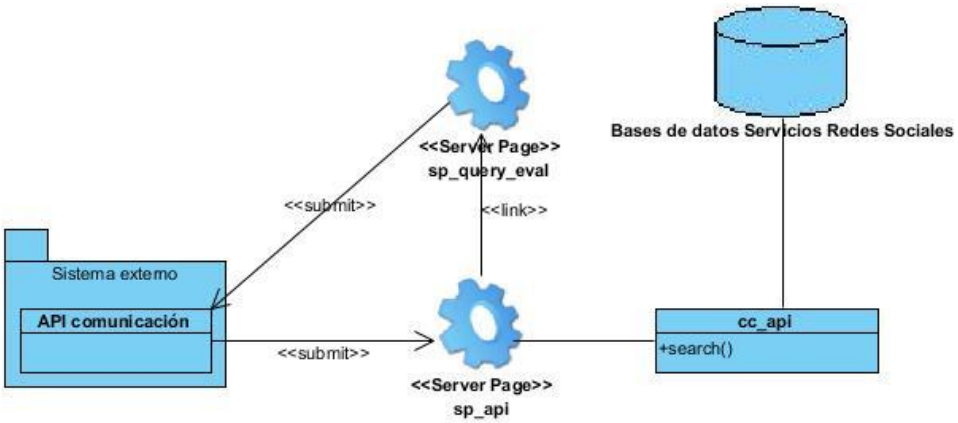

Fig. 8 Diagrama de clases de diseño utilizando estereotipos *web* del CU “Obtener registros dada una consulta”

El Anexo #2 del presente documento muestra otros diagramas de este tipo asociados al sistema.

2.7 Patrones de diseño

Los patrones de diseño permiten la reutilización de diseños exitosos, basando nuevos diseños en experiencias previas. Estos patrones resuelven problemas específicos de diseño, y vuelven el diseño

orientado a objetos más flexible, elegante y extremadamente reutilizable [55]. Estos patrones pueden ser clasificados en dos categorías: *GRASP* y *GoF*.

Patrones *GRASP*

Los Patrones Generales de *Software* para Asignar Responsabilidades (*GRASP*, del inglés *General Responsibility Assignment Software Patterns*) permiten, como su nombre lo indica, la asignación de responsabilidades a objetos [54]. Los siguientes patrones *GRASP* son utilizados en la solución propuesta:

- **Alta cohesión:** define que la información almacenada en una clase debe ser coherente y estar relacionada con esta. Propone además, que no se debe saturar una clase de métodos, sino asignar las responsabilidades a cada clase correspondiendo a la información que almacena. Este patrón se utiliza en todas las clases del sistema.
- **Bajo acoplamiento:** el uso de este patrón garantiza que las clases estén lo menos ligadas posible entre sí, de tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre las clases. Este patrón se utiliza en todas las clases del sistema.
- **Experto:** indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. Un ejemplo del uso de este patrón se muestra en la clase *Client* del sistema, la cual contiene la información relacionada con los sistemas externos que son registrados en la aplicación, así como los métodos encargados de gestionar dichos datos.

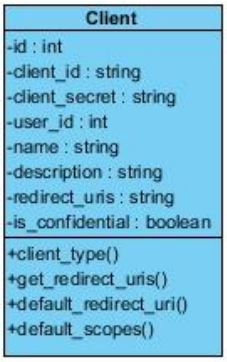

Fig. 9 Patrón experto en la clase *Client* del sistema

Patrones *GoF*

Los patrones *GoF* (La Pandilla de los Cuatro, del inglés *Gang-Of-Four*) son en el campo del diseño orientado a objetos los más conocidos y usados en la actualidad. Estos describen las formas en las que pueden ser organizados los objetos para trabajar unos con otros [54]. Para la implementación de la solución propuesta fueron utilizados varios de estos patrones, entre los que destacan:

- **Decorador (*Decorator*):** permite añadir dinámicamente funcionalidades a un objeto. Ejemplo del uso de este patrón se evidencia en el método *owner_required* del sistema, el cual se encarga de decorar otros métodos evitando que se ejecuten sin los permisos requeridos, con el objetivo de proteger los recursos de los usuarios.

```python
def owner_required(func):
    @wraps(func)
    def decorated_view(*args, **kwargs):
        id = request.args.get('id', None) or request.form.get('id', None)
        if id:
            client = Client.query.get(id)
            if client.user_id != current_user.id:
                flash(u'No tienes permisos para acceder a este recurso.', category='error')
                return redirect(url_for('admin.index'))

        return func(*args, **kwargs)
    return decorated_view
```

Fig. 10 Ejemplo de uso del patrón decorador en el sistema

- **Instancia única (*Singleton*):** garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. *Python* provee dicho mecanismo de forma nativa, siendo posible acceder a la instancia de la siguiente forma:

```python
from extensions.db import db
```

Fig. 11 Ejemplo de uso del patrón Instancia única en el sistema

2.8 Conclusiones parciales

El presente capítulo posibilitó presentar la propuesta del sistema a desarrollar. Para ello se definió el modelo de dominio, así como los detalles de los elementos que intervienen en este. También se identificaron los requerimientos con que debe cumplir la aplicación para la consecución de los objetivos del presente trabajo, describiéndolos a través del modelo de casos de uso. Se determinó además que el sistema a desarrollar sigue la arquitectura N-Capas, dividiéndose específicamente en tres capas (Presentación-Lógica del Negocio-Acceso a Datos). Para la adecuada organización de las clases de la aplicación se decidió hacer uso de los patrones *GRASP* y *GoF*, lo cual permite hacer más fluida la implementación de la solución.

CAPÍTULO #3 IMPLEMENTACIÓN Y PRUEBAS

Luego de definir el diseño y arquitectura de la solución propuesta, se procede a implementar un *software* que cumpla con los requerimientos anteriormente planteados. El presente capítulo detalla los artefactos generados en esta etapa, mostrando los estándares de codificación a seguir, así como el diagrama de componentes de la solución, el diagrama de despliegue y el modelo de datos utilizado por el sistema. Expone además evidencias de las pruebas realizadas al sistema para validar su correcto funcionamiento, mostrando los resultados obtenidos.

3.1 Implementación

A continuación se describen los elementos involucrados en la implementación, fase en la cual se materializa la solución propuesta.

3.1.1 Modelo de datos del sistema

Un modelo de datos es una colección de conceptos y reglas que se emplean para describir la estructura de una base de datos que incluye entidades, atributos y relaciones entre estos [57]. La siguiente figura representa el modelo de datos de la solución propuesta:

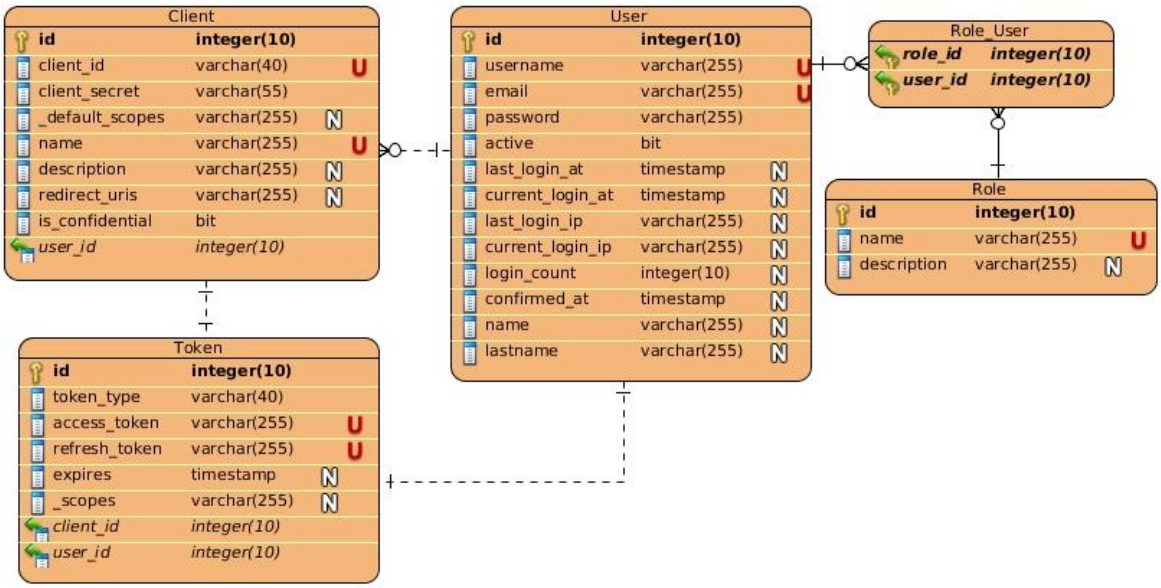

Fig. 12 Modelo de datos del sistema

A continuación son descritas las entidades que intervienen en el modelo de datos del sistema:

Tabla 16 Entidades del modelo de datos del sistema

Entidad	Descripción
User	Almacena la información relacionada con los usuarios registrados en el sistema.
Role	Contiene la información de los roles de usuario disponibles en la aplicación.
Client	Almacena la información de las aplicaciones externas que harán uso del sistema y que son registradas por los usuarios.
Token	Contiene los códigos de seguridad (*tokens*) asignados a las aplicaciones externas para garantizar la seguridad en la transferencia de información.

3.1.2 Estándares de codificación

Con el objetivo de aportar claridad al entendimiento del código fuente de la aplicación para su futuro mantenimiento, es necesario el uso de estándares o convenciones de codificación. Dichos estándares varían en dependencia del lenguaje utilizado para la implementación del *software*.

Los estándares utilizados para el desarrollo de la solución propuesta se corresponden con los definidos en el documento "*PEP 0008 - Style Guide for Python Code*" disponible en el sitio *web* oficial de *Python*. Entre los principales lineamientos establecidos en dicho documento se encuentran los siguientes:

- Toda la codificación debe realizarse en idioma inglés.
- Las sentencias de código deben limitarse a 80 caracteres por línea.
- Las funciones no anidadas y las definiciones de clases se deben separar con dos líneas en blanco y las definiciones de métodos dentro de una misma clase solo con una línea. Además se pueden utilizar líneas en blanco extras para indicar secciones lógicas dentro de las funciones.
- Para un mejor entendimiento de las sentencias *import*, estas deben colocarse en distintas líneas, aunque es correcto utilizar la coma en situaciones en las que se están importando varias clases de un mismo directorio. Los *import* deben colocarse siempre en la parte superior del archivo, antes de las variables globales y las constantes del módulo.
- Los nombres de las diferentes estructuras de código deben ser lo más descriptivos posible:
    - Los nombres de las clases adoptan la notación *UpperCamelCase*, la cual define que los nombres de los diferentes elementos del código están formados por palabras que comienzan

por mayúscula seguida de minúsculas, sin hacer uso del guion bajo como delimitador entre palabras.

- Los nombres de los atributos de las clases deben ser definidos en minúsculas, pudiéndose utilizar guiones bajos para separar las palabras, con el objetivo de mejorar la legibilidad del código.
- Para los nombres de las propiedades, las funciones, los parámetros de las funciones y las variables declaradas dentro de estas se debe utilizar la misma notación de los atributos de las clases.

3.1.3 Diagrama de componentes del sistema

El diagrama de componentes representa la división del *software* en partes más pequeñas denominadas componentes. Estas describen los elementos del sistema, ya sean código fuente, binario o ejecutable, la organización que presentan y las relaciones entre ellos, que se utilizan para indicar que un componente usa los servicios ofrecidos por otro componente [58].

El diagrama de componentes de la solución propuesta queda representado de la siguiente forma:

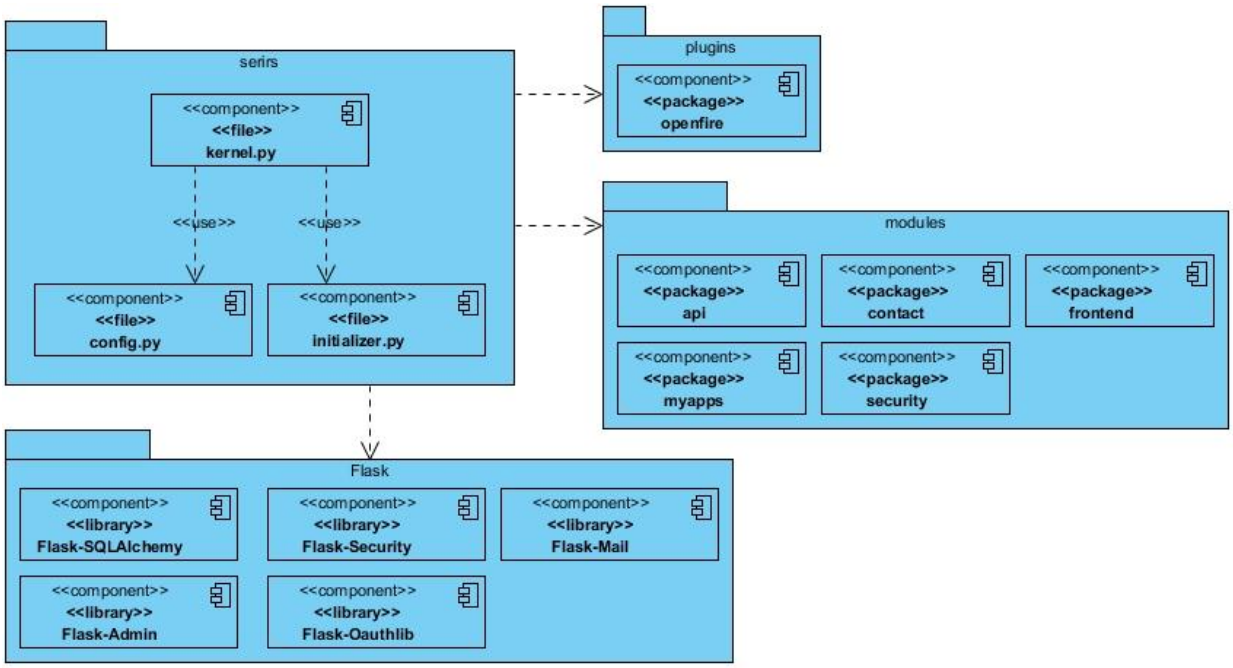

Fig. 13 Diagrama de componentes del sistema

3.1.4 Diagrama de despliegue del sistema

El diagrama de despliegue modela el *hardware* utilizado por un sistema en ambientes de puesta en práctica o de prueba. Describe la topología del sistema y muestra las relaciones físicas entre los componentes de *hardware* y *software* en el producto final, es decir, la configuración de los elementos físicos de procesamiento (servidores, ordenadores o dispositivos) en tiempo de ejecución y los componentes de *software* que se ejecutan en ellos [58].

La siguiente figura muestra el diagrama de despliegue definido para la presente solución:

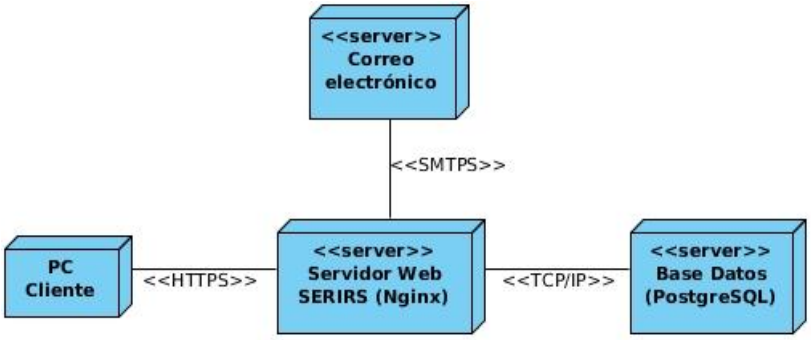

Fig. 14 Diagrama de despliegue del sistema

Los clientes, para el acceso ya sea a la capa de abstracción como a la interfaz web que ofrece el sistema para su gestión, deberán utilizar el protocolo *HTTPS*. Por otro lado, para garantizar el pleno funcionamiento de la aplicación, es necesaria la interacción de esta con los servicios de correo electrónico y bases de datos, utilizando para ello el protocolo *SMTPS* y la familia de protocolos *TCP/IP* respectivamente.

3.2 Pruebas

Las pruebas son de gran importancia en el proceso de desarrollo del *software* ya que permiten perfeccionar el producto final. Para ello se debe partir de la verificación del correcto funcionamiento del sistema, así como el cumplimiento de los requerimientos establecidos por el cliente, corrigiendo los errores encontrados en esta fase. A continuación se describen las pruebas realizadas a la solución propuesta y los correspondientes resultados obtenidos:

3.2.1 Pruebas funcionales

Las pruebas funcionales son realizadas desde el punto de vista de la interacción del usuario con el sistema a través de su interfaz, lo cual permite valorar el funcionamiento de la aplicación [59]. La metodología utilizada para la realización del presente trabajo de diploma propone el diseño de casos de prueba orientados a los casos de uso para validar el correcto desempeño de la solución y el cumplimiento de los objetivos de esta. Se adoptó como estrategia la realización de estas pruebas en dos iteraciones, lo cual permitió chequear la corrección de las no conformidades encontradas.

Como muestra se expone a continuación el diseño de casos de prueba para el CU “Obtener registros dada una consulta”. La siguiente tabla ilustra un fragmento de este diseño, mientras que los restantes figuran en el Anexo #3 del presente trabajo.

Tabla 17 Diseño de casos de prueba para el CU “Obtener registros dada una consulta”

Escenario	Descripción	Remitente	Destinatario	Respuesta del sistema	Flujo central
EC 1.1 Obtener resultado global.	La aplicación externa hace una petición sin especificar parámetros.	V vacío	V vacío	Devuelve todos los registros extraídos en formato *JSON*.	Enviar una petición utilizando el método *GET* a la *URL* “/api/search” del sistema.
EC 1.2 Obtener resultado filtrado.	La aplicación externa hace una petición especificando al menos un parámetro.	V vmherrera	V vacío	Devuelve los registros filtrados en formato *JSON*, teniendo en cuenta los valores de los parámetros especificados.	Enviar una petición con al menos un parámetro, utilizando el método *GET* a la *URL* “/api/search” del sistema.
		V vacío	V vacío		
EC 1.3 Parámetros inválidos.	La aplicación externa hace una petición especificando al menos un parámetro con valor inválido.	N/A	N/A	Devuelve un mensaje de error en formato *JSON* con estado 400 “*Bad Request*”, indicando los parámetros inválidos.	Enviar una petición con al menos un parámetro inválido, utilizando el método *GET* a la *URL* “/api/search” del sistema.
		N/A	N/A		

EC 1.4 Método no permitido.	La aplicación externa hace una petición utilizando un método distinto a *GET*.	N/A	N/A	Devuelve un mensaje de error en formato *JSON* con estado 405 "*Method Not Allowed*", indicando que el método no está permitido.	Enviar una petición utilizando un método distinto de *GET* a la *URL* "/api/search" del sistema.
EC 1.5 Cliente no autorizado.	La aplicación externa hace una petición sin obtener autorización previa.	N/A	N/A	Devuelve un mensaje de error en formato *JSON* con estado 401 "*Unauthorized*", indicando que el sistema externo que intenta acceder no está autorizado para interactuar con la aplicación.	Enviar una petición utilizando el método *GET* a la *URL* "/api/search" del sistema sin otorgar autorización a la aplicación que envía la petición.

Tabla 18 Descripción de las variables del diseño de casos de prueba para el CU "Obtener registros dada una consulta"

No.	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Remitente	Cadena de texto	Si	
2	Destinatario	Cadena de texto	Si	
3	Fecha de envío	Número entero	Si	Tiempo en milisegundos (*timestamp*).
4	Cuerpo del mensaje	Cadena de texto	Si	
5	Identificador de conversación	Número entero	Si	
6	*Plugins* a ejecutar	Arreglo de valores	Si	Múltiples cadenas de texto.

Resultados de las pruebas funcionales

Luego de llevar a cabo las pruebas de funcionalidad del sistema, fueron detectadas 7 no conformidades en total, siendo los errores más significativos la presentación de la información en distintos lenguajes y problemas en los vínculos de la aplicación. La primera iteración permitió identificar 5 no conformidades, mientras que en la segunda fueron encontradas 2 no conformidades, dándosele solución a todas.

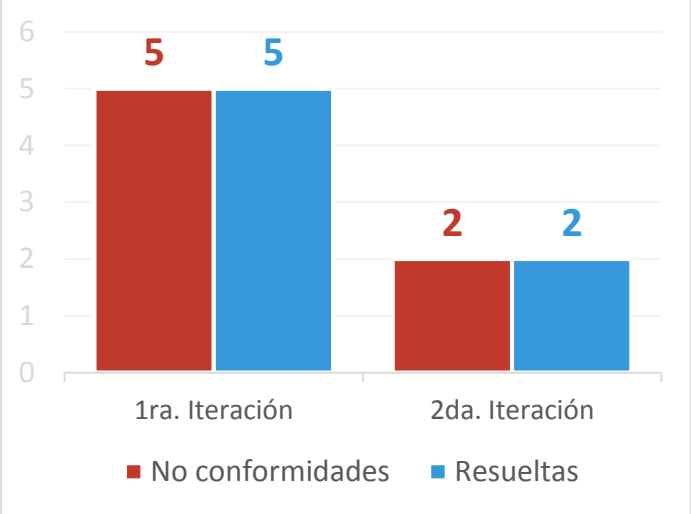

Fig. 15 Resultados de las pruebas funcionales

3.2.2 Pruebas de seguridad

Las pruebas de seguridad permiten llevar a cabo una evaluación de los sistemas desde el punto de vista externo a este. Tienen como objetivo hacer un análisis con el fin de encontrar fallos de seguridad tanto en el diseño como en la implementación de la aplicación. Además buscan medir la confidencialidad, integridad y disponibilidad de los datos, partiendo de la identificación de amenazas y riesgos en el uso de interfaces de usuario final.

La realización de estas pruebas fue dividida en dos niveles, aplicándose en el primero de ellos una lista de chequeo de pruebas de seguridad, encargada de medir cuatro indicadores (ver Anexo #4). En el segundo nivel se utilizó la herramienta *Acunetix Web Vulnerability Scanner 8.0*[18], la cual permite la detección de vulnerabilidades comunes en aplicaciones *web*, tales como inyecciones *SQL*, *Cross-Site Scripting* y fallos en los formularios [60]. Para cada uno de los niveles definidos se ejecutaron dos iteraciones, con el objetivo de minimizar los fallos de la aplicación.

Resultados de las pruebas de seguridad

Luego de una primera iteración de las pruebas de seguridad para la aplicación, fueron detectadas tres no conformidades críticas que incumplen con la lista de chequeo en varios aspectos, siendo evaluada la prueba

[18] https://www.acunetix.com/vulnerability-scanner/

de insatisfactoria. Dichas no conformidades fueron solucionadas en su totalidad. La segunda iteración no arrojó no conformidades.

La herramienta utilizada en el segundo nivel de las pruebas de seguridad detectó un total de 10 no conformidades medias, 2 no conformidades bajas, y 45 no conformidades informacionales, para un total de 57 no conformidades en la primera iteración. En la segunda iteración no fueron encontradas no conformidades que pudieran afectar la seguridad del sistema.

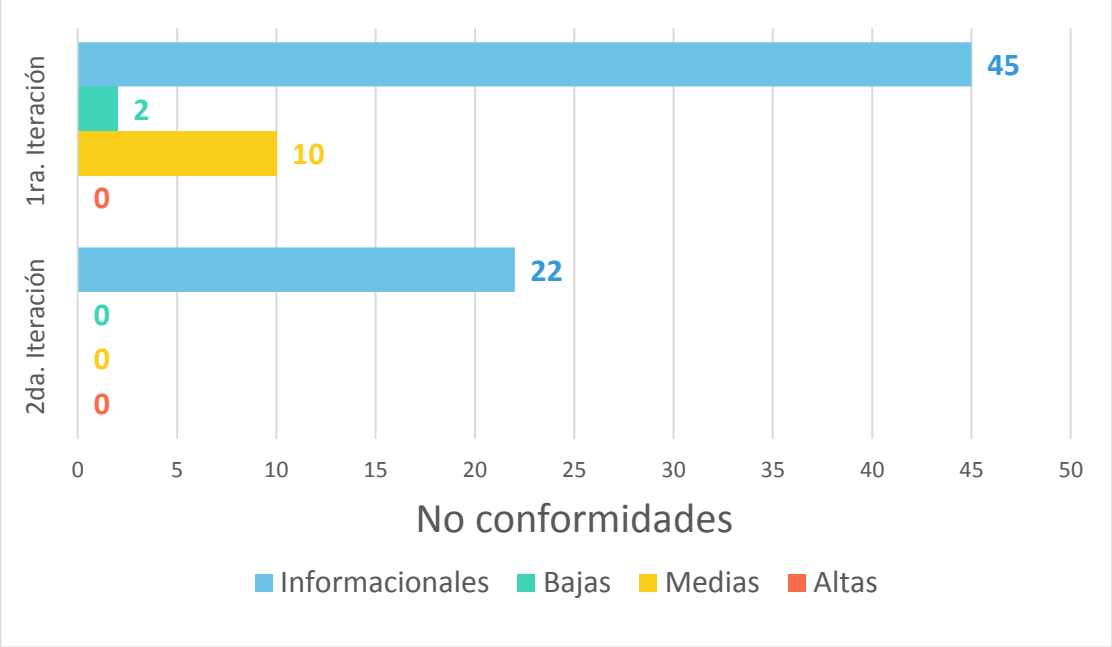

Fig. 16 Resultados de las pruebas de seguridad

3.2.3 Pruebas de carga y estrés

Las pruebas de carga se definen como las pruebas utilizadas para validar y valorar la aceptabilidad de los límites operacionales de un sistema, para atender a un conjunto de usuarios de manera simultánea. En estas se chequea de manera anticipada, el funcionamiento que tendrá el servidor de la aplicación *web* cuando esté en plena operación [61].

Por otro lado, las pruebas de estrés buscan obtener datos sobre la carga del sistema, que ayuden a realizar el dimensionamiento de este. Estas pruebas generan carga en el sistema hasta hacerlo inutilizable con el objetivo de establecer alertas para poder anticipar un fallo total del sistema [62].

Para la realización de dichas pruebas fue utilizada la herramienta *Apache JMeter 2.3.1*[19], la cual permite medir el rendimiento de recursos como servicios *web*, lenguajes dinámicos y bases de datos, entre otros. El ambiente para llevar a cabo las pruebas fue el siguiente:

- Sistema Operativo: *Ubuntu* 14.04 *LTS*.
- Microprocesador *Intel Core2Duo* a 2.20 MHz.
- Capacidad de Memoria *RAM* 2 Gb

Resultados de las pruebas de carga y estrés

Al simular con la herramienta antes mencionada la conexión concurrente de entre 50 a 100 usuarios, lo cual se considera la cifra media que pudieran acceder a la aplicación, se obtuvieron tiempos de respuesta por parte del servidor que variaban desde 1 a 4 segundos, cifras consideradas satisfactorias para estos tipos de sistemas. Más allá de los 100 usuarios y hasta 300, los tiempos de respuesta se mantuvieron aceptables, permaneciendo la aplicación sin fallos. La siguiente gráfica muestra los tiempos de respuesta obtenidos en correspondencia con los parámetros dados, a partir del uso de la herramienta señalada:

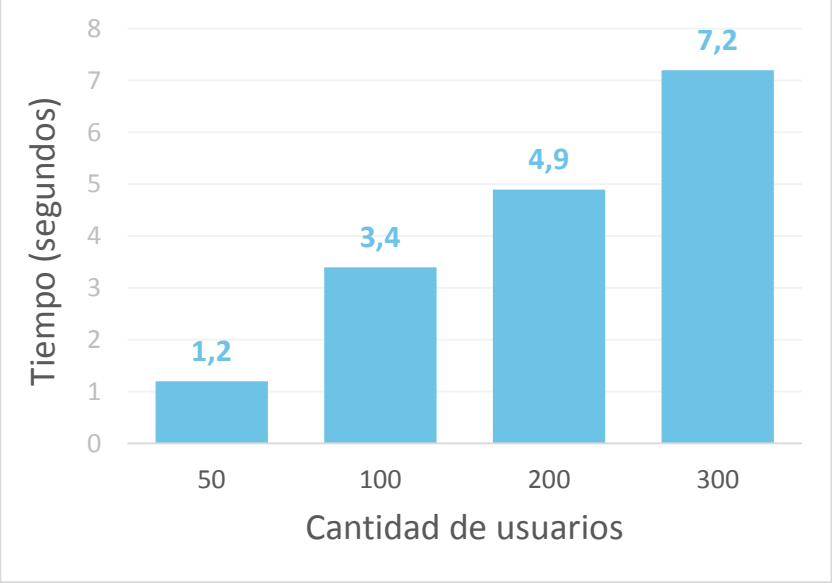

Fig. 17 Resultados de las pruebas de carga y estrés

[19] http://jmeter.apache.org/

3.3 Conclusiones parciales

En el presente capítulo se definieron los elementos relacionados con la implementación del sistema, así como su etapa de pruebas. Se puntualizaron temas como el estándar de codificación utilizado para el desarrollo del producto, que posibilitó generar código fácil de entender y mantener por otros desarrolladores. Se tuvo en cuenta también el diagrama de despliegue de la solución, el cual permitió esclarecer la distribución física de los componentes del sistema. Además, se pudo constatar el cumplimiento de los requerimientos establecidos para el sistema, a partir de las pruebas funcionales, de seguridad y de carga y estrés aplicadas a la solución implementada.

CONCLUSIONES

- El análisis de los elementos teóricos relacionados con la implementación de servicios *web*, así como la selección de las herramientas y tecnologías más idóneas para llevar a cabo esta tarea, propiciaron el desarrollo de una solución de acuerdo a las necesidades existentes.
- Fue diseñada una solución que permite la interoperabilidad entre aplicaciones al funcionar como medio de comunicación entre las herramientas de *software* desarrolladas en el Departamento de Ingeniería Social Universitaria y las bases de datos de los servidores de mensajería instantánea utilizados por los Servicios de Redes Sociales.
- La realización de las pruebas a la aplicación permitió comprobar su correcto funcionamiento, demostrándose la validez de esta con resultados satisfactorios.

RECOMENDACIONES

Con el objetivo de continuar perfeccionando la solución, se propone a los desarrolladores encargados de su mejora y mantenimiento las siguientes recomendaciones:

- Desarrollar una interfaz de usuario adaptable a dispositivos móviles (*smartphones*, *tablets*).
- Permitir el acceso de los usuarios a la aplicación a través de la autenticación basada en el Protocolo Ligero de Acceso a Directorios (*LDAP*, del inglés *Lightweight Directory Access Protocol*).
- Implementar nuevos *plugins* para la aplicación que permitan ampliar el soporte a otras bases de datos de mensajería instantánea.
- Permitir a los sistemas externos obtener la información solicitada a través de las consultas hechas a la aplicación en formato *XML*.

REFERENCIAS BIBLIOGRÁFICAS

1. **LEINER, Barry, y otros.** *Brief Histoy of the Internet.* [En línea] 2009. http://www.internetsociety.org/internet/what-internet/history-internet/brief-history-internet.

2. **TEMMEL, Markus, y otros.** *The impact of the Internet on our daily life.* [En línea] 2014. https://www.tru.ca/cpj/essay.html.

3. **MITRA, Shyamal.** *Power and Impact of the Web.* [En línea] 2011. https://www.cs.utexas.edu/~mitra/honors/web2.html.

4. **BERNERS-LEE, Tim, y otros.** *World-Wide Web: the information universe.* s.l. : Internet Research, 1992. págs. 52-58.

5. **DIAZ, Daymara.** *Toma de decisiones: el imperativo diario de la vida en la organización moderna.* [En línea] 2005. http://bvs.sld.cu/revistas/aci/vol13_3_05/aci10305.htm.

6. **UBM Tech.** *A Typology Of Middleware .* [En línea] 2015. http://www.networkcomputing.com/netdesign/cdmwtypo.htm.

7. **W3C.** *Guía Breve de Servicios Web.* [En línea] 2014. http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.

8. **CERAMI, Ethan.** *Web Services Essentials.* Primera Edición. s.l. : O'Reilly Media Inc., 2002. págs. 10-19, 26-41. ISBN 0-596-00224-6.

9. **NAVARRO, Rafael.** *REST vs. Web Services.* [En línea] 2007. http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf.

10. **W3C.** *SOAP Version 1.2 Part 1: Messaging Framework.* [En línea] 2007. http://www.w3.org/TR/2007/REC-soap12-part1-20070427.

11. **FIELDING, Roy T.** *Architectural Styles and the Design of Network-based Software Architectures.* [En línea] 2000. http://roy.gbiv.com/pubs/dissertation/top.htm.

12. **RICHARDSON, Leonard y RUBY, Sam.** *RESTful Web Services.* Primera Edición. s.l. : O'Reilly Media Inc., 2007. págs. 299-314. ISBN 0-596-52926-0.

13. **ProgrammableWeb.** [En línea] http://www.programmableweb.com.

14. **THOLOMÉ, Eric.** A well earned retirement for the SOAP Search API. *The official Google Code Blog.* [En línea] http://googlecode.blogspot.com/2009/08/well-earned-retirement-for-soap-search.html.

15. **Twitter Inc.** Twitter Developers REST APIs. [En línea] https://dev.twitter.com/rest/public.

16. **MONTERO, Ramón.** *XML, Iniciación y referencia.* s.l. : McGraw-Hill, 2001. ISBN: 84-481-2894-X.

17. **JSON.org.** *JSON: The Fat-Free Alternative to XML.* [En línea] http://www.json.org/xml.html.

18. **OASIS.** *Reference Model for Service Oriented Architecture 1.0.* [En línea] 2006. http://docs.oasis-open.org/soa-rm/v1.0/.

19. **ERL, Thomas.** *SOA: principles of service design.* s.l. : Prentice Hall, 2008. ISBN-10 0132344823.

20. **BOYD, Danah.** *Social Network Sites: Definition, History, and Scholarship.* s.l. : Ellison, 2007.

21. **DECEMBER, John.** *Social Nets Information.* [En línea] December Communications, Inc., 2014. http://www.december.com/net/social.html.

22. **Facebook Inc.** *Statistics.* [En línea] 2014. [Citado el: 14 de Enero de 2015.] https://newsroom.fb.com/company-info/.

23. **CNET.** *Google+ hits 20 million mark in three weeks.* [En línea] 21 de Julio de 2011. http://news.cnet.com/8301-1023_3-20081650-93/google-hits-20-million-mark-in-three-weeks/?part=rss&subj=news&tag=2547-1_3-0-20.

24. **NARDI, Bonnie, WHITTAKER, Steve y BRADNER, Erin.** *Interaction and outeraction: instant messaging in action.* s.l. : ACM, 2000.

25. **SAINT-ANDRE, Peter.** *Extensible messaging and presence protocol (XMPP): Core.* [En línea] 2011. http://tools.ietf.org/html/rfc6120. ISSN: 2070-1721.

26. **Ignite Realtime.** *Openfire Server.* [En línea] http://www.igniterealtime.org/projects/openfire/index.jsp.

27. —. Openfire Documentation. [En línea] 2015. http://www.igniterealtime.org/projects/openfire/documentation.jsp.

28. **ACHOUR, Mehdi, y otros.** *Manual de PHP.* [En línea] 2014. http://php.net/manual/es/intro-whatis.php.

29. **Oracle.** *About Java.* [En línea] 2014. http://www.java.com/about.

30. **LUTZ, Mark.** *Learning Python.* Tercera Edición. s.l. : O'Reilly Media Inc., 2008. pág. 8. ISBN-10 0-596-51398-4.

31. **Universidad de Belgrano.** *Conceptos Fundamentales de las Bases de Datos.* [En línea] 2014. http://www.ub.edu.ar/catedras/ingenieria/Datos/capitulo1/cap11.htm.

32. **ORACLE.** *About MySQL.* [En línea] 2015. http://www.mysql.com/about/.

33. **The PostgreSQL Global Development Group .** *About ProstgreSQL.* [En línea] 2015. http://www.postgresql.org/about.

34. **ASTEASUAIN, Fernando.** *UML.* s.l. : GRADI, 2009. ISBN 978-987-1347-95-7.

35. **Object Management Group.** *OMG Unified Modeling Language Specification.* [En línea] 2001. http://www.omg.org.

36. **KABIR, Mohammed.** *La biblia del servidor Apache2.* 2004. págs. 39-50.

37. **Nginx Community.** *Nginx Wiki.* [En línea] 2014. http://wiki.nginx.org/Main.

38. **NGINX, Inc.** *NGINX Plus Technicals Specifications.* [En línea] 2015. http://nginx.com/products/technical-specs/.

39. **IBM DeveloperWorks.** *Iniciándose en la plataforma Eclipse.* [En línea] 2014. http://www.ibm.com/developerworks/ssa/library/os-ecov/.

40. **JetBrains.** *PyCharm.* [En línea] 2014. http://www.jetbrains.com/pycharm/.

41. **SOMMERVILLE, Ian.** *Ingeniería de Software.* s.l. : Pearson, 2005.

42. **DAVIS, Jim.** *Rational Rose.* [En línea] 2005. http://www.cse.sc.edu/~jimdavis/Tools/rational_rose.htm.

43. **Visual Paradigm.** *Visual Paradigm Features.* [En línea] 2014. http://www.visual-paradigm.com/.

44. **RIEHLE, Dirk.** *Framework Design: A Role Modeling Approach.* [En línea] 2000. http://www.riehle.org/computer-science/research/dissertation/diss-a4.pdf.

45. **Zend Technologies Ltd.** *About Zend Framework 2.* [En línea] 2014. http://www.framework.zend.com/about/.

46. **CHRISTIE, Tom.** *Django REST Framework.* [En línea] 2014. http://www.django-rest-framework.org/.

47. **RONACHER, Armin.** *Flask Documentation.* 2014.

48. **Universidad de Murcia.** *Metodologías de desarrollo de software.* [En línea] 2006. http://www.um.es/docencia/barzana/IAGP/Iagp2.html.

49. **Rational Software.** *Rational Unified Process: Best Practices for Software Development Teams.* 1999.

50. **SHAHID, Najam, y otros.** Rational Unified Process. [En línea] 2009. http://ovais.khan.tripod.com/papers/Rational_Unified_Process.pdf.

51. **CANÓS, José, LETELIER, Patricio y PENADÉS, María del Carmen.** *Metodologías Ágiles en el Desarrollo de Software.* s.l. : Universidad Politécnica de Valencia, 2003.

52. **URQUIZA, José, MARTÍNEZ, Alfonso y IBARGÜENGOITIA, Guadalupe.** *Las Metodologías Ágiles y las Arquitecturas de Software.* s.l. : UAM-Iztapalapa, 2010.

53. **Eclipse Foundation.** *OpenUP.* [En línea] 2012. http://epf.eclipse.org/wikis/openup/.

54. **LARMAN, Craig.** *UML y Patrones.* Segunda Edición. s.l. : Prentice Hall, 2002.

55. **PRESSMAN, Roger S.** *Ingeniería del Softare. Un enfoque práctico.* Quinta Edición. s.l. : McGraw-Hill, 2001.

56. **DE LA TORRE, C., y otros.** *Guía de arquitectura N-Capas orientada al dominio con .NET 4.0.* s.l. : Krasis Press, 2010.

57. **ALVARADO, José M.** *Bases de datos. Unidad 2. Modelo de datos.* s.l. : Universidad Nacional de Ingeniería Augusto César Sandino.

58. **CAMPDERRICH, Benet.** *Ingeniería de Software.* s.l. : Editorial UOC, 2002.

59. **ORÉ, Alexander.** *Pruebas funcionales.* [En línea] 2009. http://www.calidadysoftware.com/testing/pruebas_funcionales.php.

60. **Acunetix.** *Web Application Security with Acunetix Web Vulnerability Scanner.* [En línea] 2015. https://www.acunetix.com/vulnerability-scanner/.

61. **Guía Digital.** *Pruebas de Carga.* [En línea] 2015. http://www.guiadigital.gob.cl/articulo/pruebas-de-carga.

62. **Globe Testing.** *Pruebas de rendimiento.* [En línea] 2015. http://www.globetesting.com/pruebas-de-rendimiento/.

63. **FIGUEROA, Robert, SOLÍS, Camilo y CABRERA, Armando.** *Metodologías tradicionales vs. Metodologías ágiles.* s.l. : Universidad Técnica Particular de Loja, 2004.

64. **Sybase Inc.** *Sybase Data Integration Suite 1.1 Overview Guide.* [En línea] 2007. http://infocenter.sybase.com/archive/index.jsp?topic=/com.sybase.help.disuite_etl_1.1/title.htm.

65. **Oracle.** *Oracle Data Integration.* [En línea] 2015. http://www.oracle.com/us/products/middleware/data-integration/overview/index.html.

66. **The Scriptella Project Team.** *Scriptella ETL Reference Documentation.* [En línea] 2012. http://scriptella.javaforge.com/reference/index.html.

67. **Talend.** *Talend Open Studio.* [En línea] 2015. http://www.talend.com/products/talend-open-studio.

68. **CloverETL.** *CloverETL Data Integration.* [En línea] 2015. http://www.cloveretl.com/products/community-edition.

BIBLIOGRAFÍA CONSULTADA

BERNERS-LEE, Tim, y otros. World-Wide Web: the information universe. s.l. : Internet Research, 1992.

CloverETL. CloverETL Data Integration. [En línea] 2015. http://www.cloveretl.com/products/community-edition.

HAFNER, Katie. Where wizards stay up late: The origins of the Internet. s.l. : Simon and Schuster, 1998.

Oracle. Oracle Data Integration. [En línea] 2015. http://www.oracle.com/us/products/middleware/data-integration/overview/index.html.

Sybase Inc. Sybase Data Integration Suite 1.1 Overview Guide. [En línea] 2007. http://infocenter.sybase.com/archive/index.jsp?topic=/com.sybase.help.disuite_etl_1.1/title.htm.

Talend. Talend Open Studio. [En línea] 2015. http://www.talend.com/products/talend-open-studio.

The Scriptella Project Team. Scriptella ETL Reference Documentation. [En línea] 2012. http://scriptella.javaforge.com/reference/index.html.

VAN ROSSUM, Guido, y otros. PEP 0008 -- Style Guide for Python Code. [En línea] 2015. https://www.python.org/dev/peps/pep-0008/

GLOSARIO DE TÉRMINOS

API*: *Application Programming Interface o Interfaz de programación de aplicaciones, 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.

ETL*: *Extract*, *Transform and Load o Extraer, transformar y cargar, es el proceso que permite a las organizaciones mover datos desde múltiples fuentes, reformatearlos y limpiarlos, y cargarlos en otra base de datos, para analizar, o en otro sistema operacional para apoyar un proceso de negocio.

***Framework*:** es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de *software* concretos, que puede servir de base para la organización y desarrollo de *software*.

FTP*: *File Transfer Protocol o protocolo de transferencia de archivos, es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red *TCP*, basado en la arquitectura cliente-servidor.

GPL*: *GNU General Public License o Licencia Pública General de *GNU*.

HTTPS*: *Hypertext Transfer Protocol Secure o protocolo seguro de transferencia de hipertexto, es un protocolo de aplicación basado en el protocolo *HTTP* que posibilita la transferencia segura de datos de hipertexto.

Internet: conjunto descentralizado de redes de comunicación interconectadas, de alcance global.

LDAP*: *Lightweight Directory Access Protocol o protocolo ligero de acceso a directorios, es un protocolo que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

***Middleware*:** es un *software* que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, redes, *hardware* y/o sistemas operativos.

***Plugin*:** es una aplicación que se relaciona con otra para aportarle una función nueva a la primera.

***RPC*: *Remote Procedure Call* o llamada a procedimiento remoto, es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos.

***SMTPS*: *Simple Mail Transfer Protocol* o protocolo seguro para la transferencia simple de correo electrónico, es un protocolo de aplicación basado en el protocolo *SMTP*, utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos.

***SSL*: *Secure Sockets Layer* o capa de conexión segura, es un protocolo criptográfico que proporciona comunicaciones seguras por una red, comúnmente *Internet*.

***TCP/IP*:** Protocolo de Control de Transmisión/Protocolo de *Internet*, es un conjunto de protocolos de red en los cuales se basa *Internet*, ya que permiten la transmisión de datos entre computadoras.

TIC: Tecnologías de la Información y las Comunicaciones.

***VoIP*: *Voice over IP* o Voz sobre Protocolo de *Internet* es un grupo de recursos que hacen posible que la señal de voz viaje a través de *Internet* empleando el protocolo *IP*.

***WS-Security*:** es un protocolo de comunicaciones que suministra un medio para aplicar seguridad a los servicios web.

***WWW:** World Wide Web* o Red informática mundial, es un sistema de distribución de documentos de hipertexto o hipermedios interconectados y accesibles vía *Internet*.

***XMPP*: *Extensible Messaging and Presence Protocol* o Protocolo extensible de mensajería y comunicación de presencia, es un protocolo que permite establecer una plataforma para el intercambio de datos en formato *XML*, utilizado por aplicaciones de mensajería instantánea.

ANEXOS

Anexo #1: Descripción de casos de uso del sistema

Tabla 19 Descripción del CU "Listar *plugins* instalados en el sistema"

CU	Listar *plugins* instalados en el sistema		
Objetivo	Mostrar la información relacionada con los *plugins* instalados en el sistema.		
Actores	Administrador (inicia), Desarrollador (inicia).		
Resumen	Se inicia cuando un usuario desea visualizar la información relacionada con los *plugins* instalados en el sistema. La aplicación muestra un listado de los *plugins*, finalizando así el caso de uso.		
Complejidad	Media.		
Prioridad	Media.		
Precondiciones	Debe existir al menos un *plugin* instalado en el sistema.		
Postcondiciones	Se muestra una lista con los *plugins* instalados en el sistema.		
Flujo de eventos			
Flujo básico "Listar *plugins* instalados en el sistema"			
	Actor	**Sistema**	
1.	Hace clic en el vínculo "*Plugins*" del menú superior.	Muestra un listado de los *plugins* instalados en el sistema con los campos Nombre, Descripción, Autor, Versión y un vínculo hacia la página de la documentación del *plugin*.	
2.		Finaliza el CU.	
Relaciones	**CU incluidos**	Ninguno	
	CU extendidos	Ninguno	

Tabla 20 Descripción del CU "Autenticar usuario"

CU	Autenticar usuario.
Objetivo	Permitir el acceso a las funcionalidades restringidas de la aplicación.
Actores	Administrador (inicia), Desarrollador (inicia).
Resumen	Se inicia cuando un usuario desea obtener privilegios para interactuar con el sistema. La aplicación valida los datos proporcionados por el usuario y, en caso de que sean correctos, le concede privilegios, terminando así el caso de uso.
Complejidad	Baja.

Prioridad	Media.	
Precondiciones	La cuenta del usuario que intenta acceder al sistema debe existir y estar activa.	
Postcondiciones	Se autentica el usuario y se crea una sesión para este.	
Flujo de eventos		
Flujo básico "Autenticar usuario"		
	Actor	**Sistema**
1.	Hace clic en el botón "Acceder" del menú superior.	Muestra un formulario con los siguientes campos: Usuario (obligatorio) Contraseña (obligatorio)
2.	Entra los datos solicitados y hace clic en el botón "Entrar" del formulario.	Valida los datos entrados por el usuario y, en caso de ser correctos, le otorga permisos en correspondencia con el rol que tenga asignado.
3.		Finaliza el CU.
Flujos alternos		
No. 1 "Campos obligatorios sin valor"		
1.	Hace clic en el botón "Entrar" del formulario sin proporcionar valores para los campos obligatorios del formulario.	Muestra un mensaje de error notificando que se deben proveer valores para los campos obligatorios, y vuelve a solicitar los datos.
No. 2 "Usuario o contraseña incorrectos"		
1.	Provee un nombre de usuario inexistente o inactivo, o una contraseña incorrecta, y hace clic en el botón "Entrar" del formulario.	Muestra el mensaje de error: "Usuario o contraseña incorrecta", y vuelve a solicitar los datos.
Relaciones	**CU incluidos**	Ninguno
	CU extendidos	Ninguno

Tabla 21 Descripción del CU "Gestionar aplicaciones"

CU	Gestionar aplicaciones.
Objetivo	Registro, eliminación, modificación y visualización de los terceros sistemas que interactuarán con la aplicación.
Actores	Desarrollador (inicia).
Resumen	Se inicia cuando un desarrollador desea registrar, eliminar, editar o visualizar la información de los terceros sistemas que harán uso de los servicios web. La aplicación ejecuta la funcionalidad solicitada, finalizando así el Caso de Uso.
Complejidad	Baja.
Prioridad	Alta.

Precondiciones	El usuario que realiza la acción debe estar autenticado y tener asignado el rol de Desarrollador.	
Postcondiciones	Se ejecuta la acción solicitada.	
Flujo de eventos		
Flujo básico "Gestionar aplicaciones cliente"		
	Actor	**Sistema**
1.		Muestra las acciones disponibles para la gestión de las aplicaciones cliente: Registrar aplicación (ver Sección 1) Eliminar aplicación (ver Sección 2) Modificar aplicación (ver Sección 3) Listar aplicaciones (ver Sección 4)
Sección 1: "Registrar aplicación"		
Flujo básico "Registrar aplicación"		
1.	Hace clic en la opción "Nuevo" del panel lateral derecho de la interfaz de gestión de aplicaciones.	Muestra un formulario con los siguientes campos: Nombre (obligatorio, único) Descripción URI de la aplicación (obligatorio)
2.	Introduce los datos solicitados y hace clic en el botón "Guardar" del formulario.	Valida los datos entrados y, en caso de ser correctos, los almacena, notificando que la operación se realizó con éxito.
3.		Finaliza el CU.
Flujos alternos "Registrar aplicación"		
No. 1 "Campos obligatorios sin valor"		
1.	Hace clic en el botón "Guardar" sin proporcionar valores para los campos obligatorios del formulario.	Muestra un mensaje de error notificando que se deben proveer valores para los campos obligatorios, y vuelve a solicitar los datos.
No. 2 "Valores ya registrados en los campos únicos"		
1.	Introduce valores ya registrados en el sistema para los campos únicos, y hace clic en el botón "Guardar" del formulario.	Muestra un mensaje de error notificando que debe proveer otro valor para los campos únicos, y vuelve a solicitar los datos.
No. 3 "Nombre no cumple con el requisito de cantidad de caracteres permitidos"		
1.	No introduce entre 2 y 255 caracteres en el campo de Nombre y hace clic en el botón "Guardar" del formulario.	Muestra un mensaje de error notificando las restricciones del campo en cuanto a cantidad de caracteres, y vuelve a solicitar los datos.
No. 4 "URI de la aplicación inválida"		

1.	Proporciona un valor erróneo para el campo de la URI de la aplicación y hace clic en el botón “Guardar” del formulario.	Muestra un mensaje de error notificando que debe introducir una dirección URI válida y vuelve a solicitar los datos.
No. 5 “Nombre de aplicación inválido”		
1.	Proporciona un valor erróneo para el campo de Nombre de aplicación y hace clic en el botón “Guardar” del formulario.	Muestra un mensaje de error notificando que el campo solamente debe contener letras, números y el caracter especial _, además de comenzar con una letra. Vuelve a solicitar los datos.
Sección 2: “Eliminar aplicación”		
Flujo básico “Eliminar aplicación”		
1.	Hace clic en el botón “Eliminar”  asociado a una aplicación cliente mostrada en la lista de las aplicaciones registradas en el sistema, o selecciona varios elementos de dicha lista y hace clic en la opción “Herramientas > Eliminar”.	Muestra un mensaje de confirmación notificando la acción que se va a realizar.
2.	Confirma que desea eliminar la o las aplicaciones haciendo clic en el botón “Aceptar” del mensaje de confirmación.	Elimina la aplicación o las aplicaciones seleccionadas y notifica que la operación se realizó con éxito.
3.		Finaliza el CU.
Flujos alternos “Eliminar aplicación”		
No. 1 “Ninguna aplicación seleccionada para eliminar”		
1.	No selecciona ningún elemento de la lista de aplicaciones registradas en el sistema, y hace clic en la opción “Herramientas > Eliminar”.	Muestra un mensaje notificando que se debe seleccionar al menos un elemento para eliminar.
No. 2 “Cancelada la operación de eliminar”		
1.	No confirma que desea eliminar la aplicación o las aplicaciones seleccionadas haciendo clic en el botón “Cancelar” del mensaje de confirmación de la eliminación.	Vuelve al listado de las aplicaciones sin eliminar la aplicación o aplicaciones seleccionadas.
Sección 3: “Modificar aplicación”		
Flujo básico “Modificar aplicación”		
1.	Hace clic en el botón “Editar”  asociado a una aplicación mostrada en la lista de las aplicaciones registradas en el sistema, o en la opción “Editar” del menú lateral derecho de la interfaz de gestión de aplicaciones.	Muestra un formulario con los siguientes campos: Nombre (obligatorio, único) Descripción URI de la aplicación (obligatorio)

2.	Introduce los datos solicitados y hace clic en el botón “Guardar” del formulario.	Valida los datos entrados y, en caso de ser correctos, actualiza la aplicación, notificando que la operación se realizó con éxito.
3.		Finaliza el CU.
Flujos alternos “Modificar aplicación”		
No. 1 “Campos obligatorios sin valor”		
1.	Hace clic en el botón “Guardar” sin proporcionar valores para los campos obligatorios del formulario.	Muestra un mensaje de error notificando que se deben proveer valores para los campos obligatorios, y vuelve a solicitar los datos.
No. 2 “Valores ya registrados en los campos únicos”		
1.	Introduce valores ya registrados en el sistema para los campos únicos, y hace clic en el botón “Guardar” del formulario.	Muestra un mensaje de error notificando que debe proveer otro valor para los campos únicos, y vuelve a solicitar los datos.
No. 3 “Nombre no cumple con el requisito de cantidad de caracteres permitidos”		
1.	No introduce entre 2 y 255 caracteres en el campo de Nombre y hace clic en el botón “Guardar” del formulario.	Muestra un mensaje de error notificando las restricciones del campo en cuanto a cantidad de caracteres, y vuelve a solicitar los datos.
No. 4 “URI de la aplicación inválida”		
1.	Proporciona un valor erróneo para el campo de la URI de la aplicación y hace clic en el botón “Guardar” del formulario.	Muestra un mensaje de error notificando que debe introducir una dirección URI válida y vuelve a solicitar los datos.
Sección 4: “Listar aplicaciones”		
Flujo básico “Listar aplicaciones”		
1.	Hace clic en la opción “Mis Aplicaciones” del panel superior de navegación de la interfaz de gestión de aplicaciones, o en la opción “Lista” del menú lateral derecho de dicha interfaz.	Muestra un listado de las aplicaciones registradas por el usuario en el sistema, con los campos Nombre y Descripción.
2.		Fin del CU.
Flujos alternos “Listar aplicaciones”		
No. 1 “Ninguna aplicación registrada”		
1.		Muestra un mensaje notificando que no existen aplicaciones registradas por el usuario en el sistema.

Relaciones	**CU incluidos**	Ninguno
	CU extendidos	Ninguno

Anexo #2: Diagramas de clase de diseño del sistema

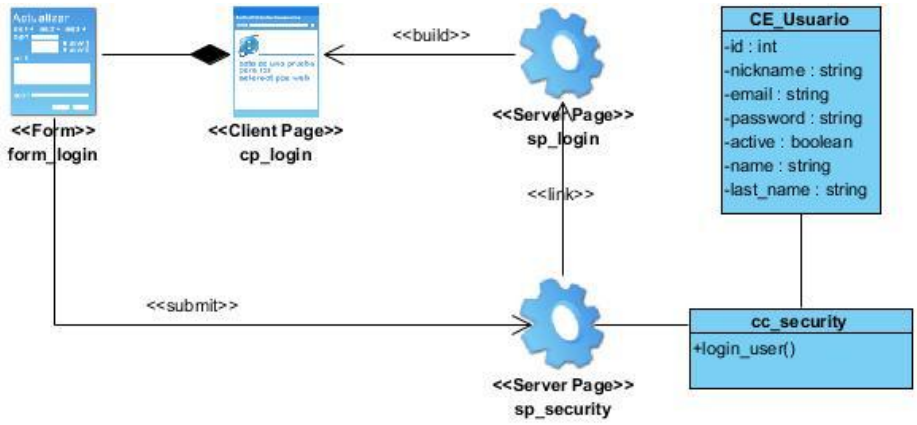

Fig. 18 Diagrama de clases de diseño utilizando estereotipos *web* del CU "Autenticar usuario"

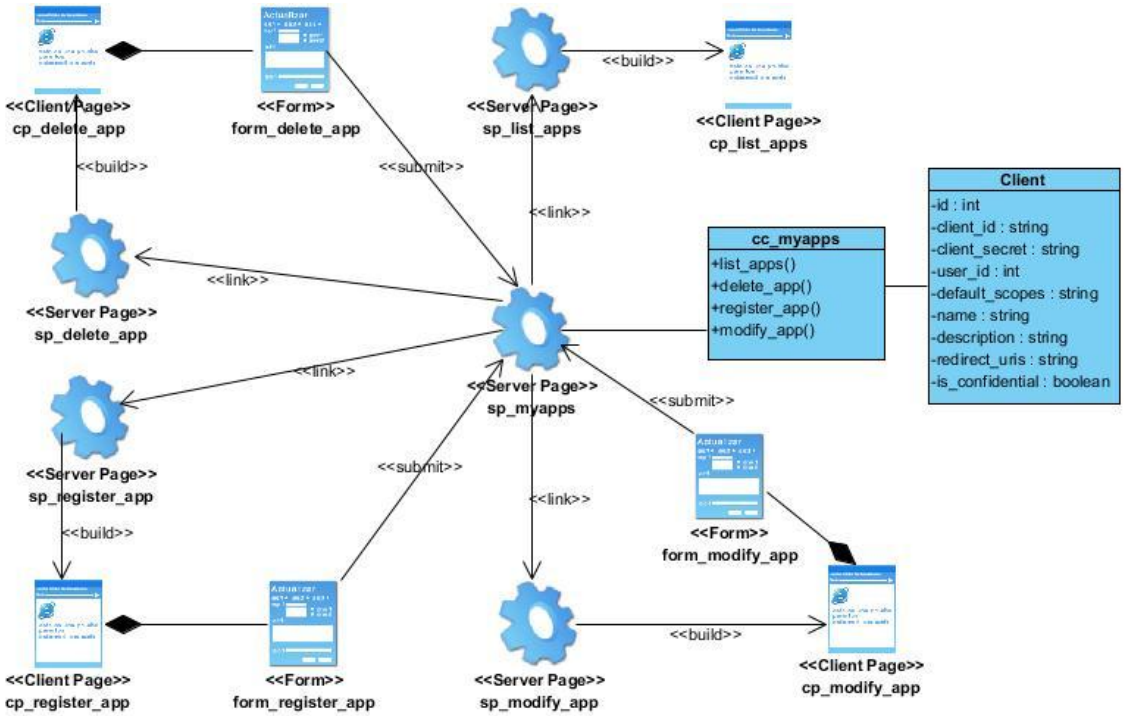

Fig. 19 Diagrama de clases de diseño utilizando estereotipos web del CU "Gestionar aplicaciones"

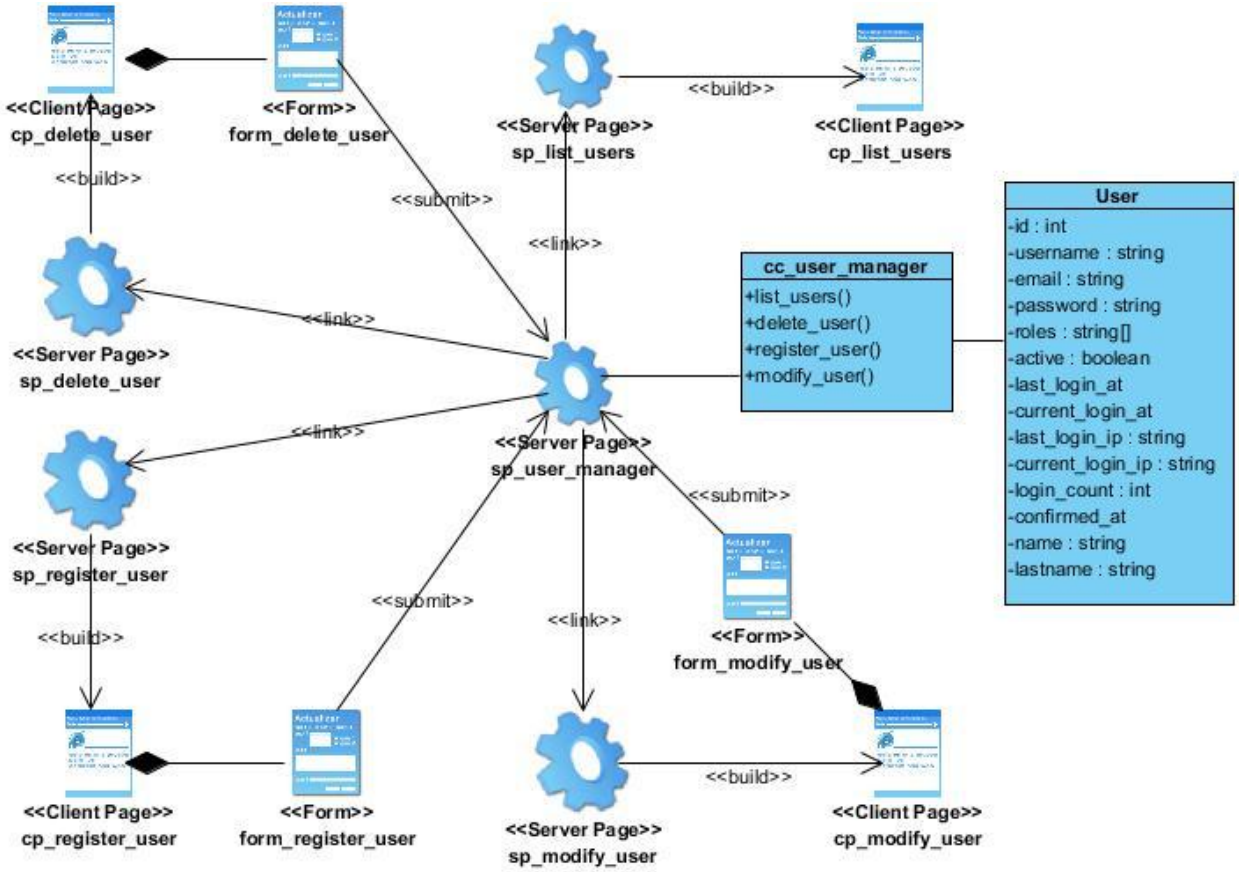

Fig. 20 Diagrama de clases de diseño utilizando estereotipos web del CU “Gestionar usuarios”

Anexo #3: Diseños de casos de prueba del sistema

Tabla 22 Diseño de casos de prueba para el CU "Gestionar aplicaciones cliente"

Escenario	Descripción	Nombre de aplicación	Descripción	URI de aplicación	Respuesta del sistema	Flujo central
EC 1.1 Aplicación registrada correctamente.	El desarrollador proporciona los datos solicitados por el sistema correctamente.	V SERIRS	N/A	V http://serirs.dev/authorized	Registra la aplicación y muestra la notificación "Se ha creado una nueva aplicación satisfactoriamente".	Hacer clic en el vínculo "Nuevo" del menú lateral derecho de la interfaz de administración de aplicaciones. Introducir los datos solicitados por el formulario y hacer clic en el botón "Guardar".
EC 1.2 Campos obligatorios sin valor.	El desarrollador envía el formulario dejando algún campo obligatorio sin valor.	I vacío	N/A	I vacío	Muestra el mensaje de error correspondiente al campo vacío indicando que es necesario llenarlo.	Hacer clic en el botón "Guardar" del formulario dejando al menos un campo vacío.
		I vacío	N/A	V http://serirs.dev/authorized		
EC 1.3 Nombre de aplicación duplicado.	El desarrollador introduce un valor ya registrado en el sistema para el nombre de la aplicación.	I valor ya existente	N/A	V http://serirs.dev/authorized	Muestra el mensaje de error "Debes proveer otro nombre para la aplicación. Este nombre ya está en uso".	Hacer clic en el botón "Guardar" del formulario luego de introducir un valor ya existente en el campo Nombre de aplicación.
EC 1.4 Violada restricción de cantidad de caracteres para el campo de nombre de la aplicación.	El desarrollador introduce un valor que no cumple la restricción de cantidad de caracteres del	I a	N/A	V http://serirs.dev/authorized	Muestra el mensaje de error "El campo debe contener entre 2 y 255 caracteres".	Hacer clic en el botón "Guardar" del formulario luego de introducir un valor que no cumpla la restricción de cantidad de caracteres permitidos

	campo de nombre de aplicación.					del campo Nombre de aplicación.
EC 1.5 Formato incorrecto para los valores de los campos.	El desarrollador introduce un valor incorrecto en al menos un campo del formulario.	I MyApp! V MyApp	N/A N/A	I cualquier_cosa I un@correo.cu	Muestra el mensaje de error correspondiente al campo con valor incorrecto, indicando la restricción del campo en cuanto a su formato.	Hacer clic en el botón “Guardar” del formulario luego de introducir un valor que no cumpla las restricciones de formato de los campos Nombre de aplicación y URI de aplicación.
EC 2.1 Aplicación eliminada correctamente.	El desarrollador elimina correctamente una o varias aplicaciones registradas en el sistema.	N/A	N/A	N/A	Elimina la o las aplicaciones seleccionadas y muestra el mensaje “Se ha eliminado la aplicación satisfactoriamente”.	Hacer clic en el botón “Eliminar” asociado a una de las aplicaciones de la lista de la interfaz de gestión de aplicaciones, o seleccionar varios elementos de dicha lista y hacer clic en “Herramientas > Eliminar”. Confirmar el mensaje de eliminación.
EC 2.2 Ninguna aplicación seleccionada para eliminar.	El desarrollador no selecciona ninguna aplicación para eliminar.	N/A	N/A	N/A	Muestra el mensaje de error “Debes seleccionar al menos un elemento”.	Hacer clic en “Herramientas > Eliminar” sin seleccionar al menos un elemento de la lista de aplicaciones.
		V	N/A	V		

EC 3.1 Aplicación modificada correctamente.	El desarrollador proporciona los datos solicitados correctamente.	SERIRS		https://dev.serirs.uci.cu/auth	Actualiza los datos de la aplicación y muestra el mensaje “La aplicación fue modificada satisfactoriamente”.	Hacer clic en el botón “Editar” asociado a una de las aplicaciones de la lista de la interfaz de gestión de aplicaciones. Introducir los datos solicitados por el formulario y hacer clic en el botón “Guardar”.
EC 3.2 Campos obligatorios sin valor.	El desarrollador envía el formulario dejando algún campo obligatorio sin valor.	I vacío I vacío	N/A	I vacío V http://serirs.dev/authorized	Muestra el mensaje de error correspondiente al campo vacío indicando que es necesario llenarlo.	Hacer clic en el botón “Guardar” del formulario dejando al menos un campo vacío.
EC 3.3 Nombre de aplicación duplicado.	El desarrollador introduce un valor ya registrado (distinto al propio valor de la aplicación que se está modificando) en el sistema para el nombre de la aplicación.	I valor ya existente	N/A	V http://serirs.dev/authorized	Muestra el mensaje de error “Debes proveer otro nombre para la aplicación. Este nombre ya está en uso”.	Hacer clic en el botón “Guardar” del formulario luego de introducir un valor ya existente en el campo Nombre de aplicación.
EC 3.4 Violada restricción de cantidad de caracteres para el campo	El desarrollador introduce un valor que no cumple la restricción de cantidad de	I a	N/A	V http://serirs.dev/authorized	Muestra el mensaje de error “El campo debe contener entre 2 y 255 caracteres”.	Hacer clic en el botón “Guardar” del formulario luego de introducir un valor que no cumpla la restricción de cantidad de

de nombre de la aplicación.	caracteres del campo de nombre de aplicación.					caracteres permitidos del campo Nombre de aplicación.
EC 3.5 Formato incorrecto para los valores de los campos.	El desarrollador introduce un valor incorrecto en al menos un campo del formulario.	I MyApp! V MyApp	N/A N/A 	I cualquier_cosa I un@correo.cu	Muestra el mensaje de error correspondiente al campo con valor incorrecto, indicando la restricción del campo en cuanto a su formato.	Hacer clic en el botón “Guardar” del formulario luego de introducir un valor que no cumpla las restricciones de formato de los campos Nombre de aplicación y URI de aplicación.
EC 4.1 Listar aplicaciones.	El desarrollador desea conocer sus aplicaciones registradas en el sistema.	N/A	N/A	N/A	Muestra un listado con las aplicaciones registradas en el sistema, reflejando de cada una el Nombre de aplicación y Descripción, además de las opciones para Visualizar, Editar y Eliminar.	Hacer clic en el vínculo “Mis Aplicaciones” de la barra de navegación de la interfaz de administración.
EC 4.2 Ninguna aplicación registrada.	No existe ninguna aplicación registrada en el sistema.	N/A	N/A	N/A	Muestra el mensaje “No existen aplicaciones registradas”.	

Tabla 23 Diseño de casos de prueba para el CU “Autenticar usuario”

Escenario	Descripción	Nombre de usuario	Contraseña	Respuesta del sistema	Flujo central
EC 1.1 Autenticación correcta.	El usuario proporciona correctamente los datos que le son solicitados para acceder a la aplicación.	V	V	Crea una sesión para el usuario y muestra su página de perfil.	Hacer clic en el botón “Acceder” del menú superior de la aplicación. Introducir los datos solicitados correctamente en el formulario.
		admin	admin		
EC 1.2 Campos obligatorios sin valor.	El usuario envía el formulario dejando algún campo obligatorio sin valor.	I	I	Muestra el mensaje de error correspondiente al campo vacío indicando que es necesario llenarlo.	Hacer clic en el botón “Entrar” del formulario dejando al menos un campo vacío.
		vacío	vacío		
		V	I		
		admin	vacío		
EC 1.3 Nombre de usuario o Contraseña incorrectos.	El usuario introduce información incorrecta y envía el formulario.	V	I	Muestra el error “Usuario o contraseña incorrecta”.	Hacer clic en el botón “Entrar” del formulario luego de introducir datos erróneos.
		admin	passwordincorrecto		
		I	V		
		adminnoexiste	admin		

Anexo #4: Pruebas de seguridad

Forma de Uso de la Lista de Chequeo:

Peso: Define si el indicador a evaluar es crítico o no.

Evaluación (Eval): Es la forma de evaluar el indicador en cuestión. Este se evalúa de 1 en caso de mal y 0 en caso que elemento revisado no presente errores.

Cantidad de elementos afectados: Especifica la cantidad de errores encontrados sobre el mismo indicador.

Comentario: Especifica los señalamientos o sugerencias que quiera incluir la persona que aplica la lista de chequeo.

Estructura del Documento: Abarca todos los aspectos definidos por el expediente de proyecto o el formato establecido por el proyecto.

N.P. (No Procede): Se usa para especificar que el indicador a evaluar no se puede aplicar en ese caso.

Estructura de la lista de chequeo:

Pruebas de Autorización					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Critico	Puede un usuario estándar (no administrador) modificar sus privilegios en la aplicación.	0		0	
Critico	Puede un usuario estándar (no administrador) modificar los privilegios de otro usuario.	0		0	
Pruebas de Gestión de Sesiones					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	Al copiar la URL de la aplicación después de estar autenticado, cerrar el navegador y volver a abrirlo para pegar la dirección copiada anteriormente, la aplicación permite que el usuario entre a la aplicación.	0		0	

Crítico	Al cerrar la sesión de un usuario y dar clic en el botón del navegador “Atrás” la aplicación vuelve entrar a la sesión autenticada.	1		1	Al cerrar la sesión y el usuario dar clic en el botón de “Atrás” el usuario vuelve a autenticarse
Comprobación del Sistema de Autenticación					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	Se bloquea la sesión del usuario después de un tiempo X (establecido por la aplicación) estando sin usar.	1		1	No se bloquea la sección del usuario después de un tiempo X
Crítico	Se bloquea la cuenta del usuario después de un número X (establecidos por la aplicación) de intento de *login* fallidos por el usuario. De ser así definir la cantidad de intentos en la columna Comentarios.	1		1	No se bloquea la cuenta.
Validación de Datos					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	Se enmascaran datos confiables cuando se visualicen en la aplicación (Por ejemplo: Contraseñas).	0		0	