UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1

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

Sistema para la búsqueda de información

Autores:
Yanisleidis Girón Vázquez
Oscar Arias Salas

Tutores:
Ing. Jorge Luis Betancourt González
MSc. Osiris Perez Moya

La Habana, Cuba, 23 de junio de 2015
"Año 57 de la Revolución"

Declaración de autoría

Declaramos ser autores del presente trabajo y autorizamos al centro CIDI de la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estime pertinente con este trabajo.
Para que así conste se firma la presente a los ___ días del mes de _____ del año _____.

Oscar Arias Salas

_____

Firma del Autor

Yanisleidis Girón Vázquez

_____

Firma del Autor

Ing. Jorge Luis Betancourt González

_____

Firma del Tutor

MSc. Osiris Perez Moya

_____

Firma del Tutor

Datos de contacto

Tutor: Jorge Luis Betancourt González

Formación académica: Ingeniero en Ciencias Informáticas, 2011.

Centro laboral: Centro de Ideoinformática.

Correo electrónico: jlbetancourt@uci.cu

Tutor: Osiris Perez Moya

Formación académica: Máster en Gestión de Proyectos Informáticos.

Centro laboral: Facultad 1. Departamento de Ingeniería de *Software*.

Correo electrónico: operez@uci.cu

FRASE

Sean siempre capaces de sentir en lo más hondo cualquier injusticia cometida contra cualquiera en cualquier lugar. Es la cualidad más linda de un revolucionario.

Ernesto "Che" Guevara

Agradecimientos

Yani

A mis padres Maira y Santiago*, porque ustedes lo son todo para mí, por su apoyo incondicional, por darme fuerzas para seguir luchando y por creer en mí cuando les prometí que volvería a la universidad para terminar lo que comencé y aquí está su regalo…"A ustedes gracias por existir".*

A mis hermanas Ari y Yanet*, en especial a* ***Ari (mi flaqui)****, por apoyarme y darme ánimos cuando lo necesitaba y tenerme como su ejemplo a seguir. Las quiero a las dos.*

A mi novio Yoe*, por estar siempre a mi lado, por compartir conmigo los buenos y malos momentos y por soportar mis malas crianzas. Gracias mi vida.*

A Edy, *a quien llamo cariñosamente como mi* ***"segundo papá"****, gracias por apoyarme y por darme ánimos de seguir siempre adelante.*

A mis abuelas Juana y Noemí, a Enrique y demás familiares*, por estar siempre al tanto de mí, por su ayuda y por su preocupación.*

A mis tutores Osiris y Jorgito*, gracias por apoyarme incondicionalmente y por soportarme a cualquier hora del día. Sin ustedes no hubiese sido posible desarrollar esta tesis. Agradecer además al profe* ***Eyeris****, por ser casi otro tutor en la tesis, por su ayuda, confianza y su apoyo para que todo saliera bien.*

Al mis mejores amigos Yisel, Sandra, Lía y Eddy*, gracias por su apoyo incondicional, por ser mis amigos del alma y por estar a mi lado cuando más lo necesité.*

A Nélida*, por acogerme en su casa, a sus hijos y familia. Gracias por el apoyo.*

A mi compañero de tesis Oscar*, gracias por confiar en mí, por considerarme una amiga más, por apoyarme en las decisiones tomadas y por ayudarme siempre en todo.*

A mis profesores y amistades de la universidad y fuera de ella*, gracias por el apoyo.*

AGRADECIMIENTOS

Oscar

A mis queridos padres Maricela (Naná) y Oscar*, así como mi hermano Marcial y mi novia* ***Arlety (Sin dudas mi familia más querida y cercana):*** *Gracias por su apoyo absoluto, que me dieron fuerzas cada segundo para seguir adelante cuando casi me rendía, por estar tan pendiente de mi tantos años y ayudarme a sonreír cada día".*

A mis padrinos*: Aunque no me Bautizaron así he querido llamarlos por convertirse en mis otros padres, brindándome su hogar y haciéndome olvidar lo lejos que estaba del mío, con su cariño* ***Gracias Idalberto y Walkis****.*

A mi familia en general sin querer olvidar a nadie, A mis abuelos, mis tíos, mis primos, mi hermano Yankiel, y en especial a mi tío Andrés y familia, mis suegritos queridos, mi cuñado Alejandro, y toda su familia en general *que me han acogido como otro hijo más, por apoyarme y darme ánimos cuando lo necesitaba.*

A mis tutores Osiris, Jorgito y Eyeris*, gracias por sus conocimientos, por su tiempo, por su apoyo. Sin ustedes no hubiese sido posible culminar esta tesis*

A mis amigos Maikel, Roannel, Walter, Luis, los chinos, Tatá, Cheché, Yaiselis*: La vida es una sola y sin amigos es imposible vivirla, yo me siento afortunado de contar con ustedes, solo en los momentos difíciles pude saber cuán importante fue su granito de arena, gracias mis panas.*

A mi compañera de tesis Yani: *gracias por ser la principal promotora para que esta tesis se hiciera, dándome ánimo e impulsándome a seguir en los momentos de mayor estrés.*

En fin, a todas las personas que contribuyeron a mi educación y a mi formación como profesional. Gracias a todos.

Dedicatoria

De Yani

A mis padres Maira y Santiago*, por estar siempre a mi lado, por la confianza depositada en mí y por hacerme creer que si se puede.* ***"Momi",*** *este regalo es para ti.*

A mis hermanas Ari y Yanet*, por creer en mí.*

A Edy*, por apoyarme en todo y estar al tanto de mi desempeño en la universidad.*

A mi novio Yoe*, gracias por ayudarme y apoyarme en todo.*

A mis abuelas Juana y Noemí*, por preocuparse tanto y por darme ánimos desde mis inicios en la universidad.*

A mis tutores*, por su esmero, dedicación y profesionalismo.*

A mis mejores amigos*, por el apoyo y la confianza depositada.*

A todas aquellas personas *que contribuyeron a mi educación y me brindaron su ayuda en los buenos y malos momentos.*

De Oscar

A mis padres Maricela (Naná) y Oscar, *por tener siempre el deseo de que su hijo se hiciera un profesional, y me apoyaron con todo desde el comienzo de mis estudios, a ustedes les va este regalo.*

Resumen

Actualmente en la Universidad de las Ciencias Informáticas (UCI) se han desarrollado sistemas que permiten la gestión de la información que está contenida en sitios web, pero no existe un sistema capaz de realizar búsquedas locales a texto completo en estos sitios. Para dar respuesta a esta necesidad, se desarrolla un sistema que permita facilitar la búsqueda de la información deseada en los sitios web y propicie la parametrización de sus funcionalidades. Este sistema se enmarca en la necesidad de facilitarle al usuario realizar búsquedas locales a texto completo de la información contenida en un sitio o portal web. Las consultas o búsquedas a texto completo suelen contener palabras y frases sencillas, o formas diversas de una palabra o frase; son aplicables en una amplia gama de escenarios empresariales, como el comercio electrónico, la búsqueda de elementos en un sitio web, etc. Para la realización del sistema, guiado por la metodología de desarrollo OpenUp, se seleccionaron como tecnologías: el marco de trabajo Symfony en su versión 2.3.6 para la programación en PHP 5.3.8, el servidor de indización Solr 4.10.3, como herramienta case *Visual Paradigm* 8.0 con UML 2.0 como lenguaje de modelado, el sistema de gestión de base de datos PostgreSQL, entre otras herramientas. Para la validación del sistema se definieron pruebas en los niveles de unidad, integración y sistema; lo que permitió corregir errores detectados logrando el correcto funcionamiento de la solución.

Palabras clave: búsquedas a texto completo, gestión de la información, información, sistema.

Índice de contenidos

Introducción……1
Capítulo 1. Fundamentación teórica……..4
1.1. Conceptos asociados al dominio del problema.....4
1.2. Análisis de otras soluciones existentes 7
1.3. Tecnologías de desarrollo 10
1.4. Conclusiones parciales 15
Capítulo 2. Concepción del sistema para la búsqueda de información……..17
2.1. Caracterización del entorno.....17
2.2. Entendimiento del negocio 18
2.3. Especificación de requisitos 19
2.4. Patrones de casos de uso utilizados 24
2.5. Especificación de casos de uso del sistema.....25
2.6. Modelo lógico de la base de datos 29
2.7. Arquitectura del sistema.....30
2.8. Patrones de diseño 34
2.9. Modelo de diseño.....36
2.10. Modelo de despliegue 37
2.11. Conclusiones parciales 39
Capítulo 3. Implementación y prueba del sistema……..40
3.1. Técnicas de programación 40
3.2. Programación Orientada a Objeto 40
3.3. Modelo de componentes que integran la solución informática.....40
3.4. Estándar de codificación 42
3.5. Estrategias de prueba 45
3.6. Conclusiones parciales 56
Conclusiones generales……..57
Recomendaciones……...58
Referencias bibliográficas……..59

Índice de figuras

Figura 1. Modelo de dominio 19
Figura 2. Diagrama de casos de uso del sistema 23
Figura 3. Ejemplo de uso del patrón de CU CRUD - Parcial 25
Figura 4. Ejemplo de uso del patrón de CU Múltiples Actores - Rol Común 25
Figura 5. Modelo lógico de la base de datos 29
Figura 6. Estilo llamada y retorno 30
Figura 7. Esquema simplificado de la arquitectura interna de Symfony2 32
Figura 8. Esquema que representa el CU Autenticar usuario con el patrón MVC 33
Figura 9. Diagrama de clases del diseño con estereotipos web del CU Gestionar feed de datos 37
Figura 10. Diagrama de despliegue del sistema 38
Figura 11. Diagrama de componentes del sistema 41
Figura 12. Ejemplo de indentación, llaves de apertura y cierre, y tamaño de las líneas 43
Figura 13. Ejemplo de convención de nomenclaturas en variables 43
Figura 14. Ejemplo de convención de nomenclaturas en clases 43
Figura 15. Ejemplo de convención de nomenclaturas en funciones 44
Figura 16. Ejemplo de estructuras de control 45
Figura 17. Resultados obtenidos de la herramienta JMeter para 50 usuarios 52
Figura 18. Resultados obtenidos de la herramienta JMeter para 250 usuarios 52
Figura 19. Resultado del proceso de pruebas 56

Índice de tablas

Tabla 1. Comparación entre sistemas 10
Tabla 2. Descripción de las clases del modelo del dominio 18
Tabla 3. Descripción del CU Gestionar feed de datos 26
Tabla 4: Prueba de integración Int_1 módulo CorreoBundle 47
Tabla 5. Prueba de integración Int_2 módulo FeedBundle 48
Tabla 6. Prueba de integración Int_3 módulo UsuarioBundle 48
Tabla 7. Prueba de integración Int_4 módulo knppaginatorBundle 49
Tabla 8. Descripción del caso de prueba para el RF Autenticar usuario 53
Tabla 9. Descripción de variables 54
Tabla 10. Listado de los requisitos funcionales del sistema 64

Introducción

Con el rápido crecimiento de las tecnologías en áreas como las redes sociales, la computación en la nube, las aplicaciones móviles, etc., uno de los retos fundamentales que enfrentan los arquitectos de información es el manejo del gran volumen de datos producidos y consumidos por los usuarios. Con estos volúmenes de información, la posibilidad de buscar el contenido deseado por el usuario cobra una importancia vital y es una característica que los usuarios esperan esté presente en los sitios web que visitan; convirtiéndose en un rasgo muy importante y a la vez muy complejo a la hora de implementar.

La búsqueda a texto completo son consultas o búsquedas lingüísticas que realizan los usuarios en una colección de datos sobre palabras y frases basándose en las reglas de un idioma determinado. Estas consultas pueden contener palabras y frases sencillas, o formas diversas de una palabra o frase [1].

En el centro de desarrollo de Ideoinformática (CIDI) de la Facultad 1 de la Universidad de las Ciencias Informáticas, actualmente se está haciendo uso de varios sistemas vinculados a la búsqueda y/o recuperación de información. Pero, estos sistemas no satisfacen las necesidades de una determinada entidad y del propio usuario. Los sistemas que existen hoy carecen de componentes necesarios que permitan una búsqueda de información adecuada. Lo que provoca que en el centro la información solicitada de un determinado sitio web no se encuentre disponible en el tiempo requerido, demorando entonces la actividad, y la calidad del trabajo que se quiere desarrollar por el personal que lo realiza.

La implementación de la búsqueda a texto completo en cualquier sitio web no es trivial, los sistemas gestores de bases de datos existentes poseen limitaciones relacionadas con esta funcionalidad. Existen sistemas como Solr que se enfocan en ofrecer características especializadas para este tipo de búsqueda. La instalación y utilización de este sistema no es sencillo y requiere el dominio de conceptos avanzados sobre las configuraciones ofrecidas por el propio sistema, lo que provoca que su integración con una aplicación existente requiera una cantidad significativa de tiempo y de personal altamente calificado capaz de implementar estas funcionalidades para mejorar la búsqueda en el sitio web. Objetivo que es muy difícil y poco recomendable cuando el sitio web no tiene gran complejidad y es desarrollado por poco personal. Además el despliegue de este sistema requiere de recursos de *hardware* de los cuales no siempre se dispone, al ser componentes de *software* que hacen uso significativo de la memoria RAM y otros recursos de la CPU, requiriendo servidores resistentes.

Teniendo en cuenta lo planteado, implementar funcionalidades de búsqueda que satisfagan las necesidades del usuario promedio puede ser complejo, costoso y lento. Las soluciones existentes en la

actualidad no son recomendables, debido a que la gran mayoría pertenecen a entidades privadas, además de estar fuera del ámbito nacional, para lo que todo sitio web que requiera de estas soluciones, debe poseer alta conectividad a Internet.

Se plantea como problema de la investigación:

¿Cómo mejorar la estrategia de búsqueda de la información deseada en los sitios web disminuyendo el tiempo empleado en su análisis así como la dependencia de Internet?

Luego de haber analizado el problema de investigación surge la necesidad de encontrar una solución teniendo como **objeto de estudio** de la investigación: el proceso de búsqueda de información en sitios web.

El **objetivo general** de este trabajo de diploma es desarrollar un sistema que permita facilitar la búsqueda de la información deseada en los sitios web y propicie la parametrización de sus funcionalidades.

Para dar solución al objetivo general del trabajo se plantearon los siguientes **objetivos específicos:**

- Elaborar el basamento teórico de la investigación.
- Diseñar el sistema para la búsqueda de información en sitios web.
- Demostrar que la solución es funcional.

Durante la investigación se **defiende la idea** de que con el desarrollo de un sistema que facilite la búsqueda de la información deseada en los sitios web propiciando la parametrización de sus funcionalidades, se mejorará la estrategia de búsqueda, disminuyendo el tiempo empleado en el análisis de la información, así como la dependencia de Internet.

Para cumplir con los objetivos se definieron las siguientes **tareas de la investigación:**

1. Elaboración del marco teórico-conceptual de la investigación.
2. Definición del entorno tecnológico para elaborar la propuesta de solución.
3. Diseño de la propuesta de solución.
4. Implementación del sistema para la búsqueda de información.
5. Diseño de los casos de prueba.
6. Validación del sistema para la búsqueda de información realizando pruebas funcionales.

Luego de cumplirse con todos los elementos antes mencionados, se obtendrán como **posibles resultados**:

- Un sistema orientado a servicios que ofrece la búsqueda de información a través de una interfaz web; brindando la posibilidad de registrar y autenticar clientes, configurar diversos

parámetros y visualizar estadísticas de interés. Además garantiza la seguridad de los datos almacenados en Solr y permite importar datos a través del protocolo HTTP.

- La documentación asociada al sistema para la búsqueda de información en sitios web.

Para cumplimentar las tareas de la investigación se utilizaron los siguientes métodos de investigación:

Analítico-Sintético: este método permitió la recopilación de información necesaria durante la realización del estudio del estado del arte para el desarrollo del trabajo mediante la revisión de documentos y artículos, de donde se extrajeron los elementos más significativos relacionados con el proceso de búsqueda de información. Además del análisis de las diferentes herramientas, metodologías y tecnologías a utilizar en el desarrollo del sistema.

Histórico-Lógico: es utilizado en el análisis de la evolución de sistemas similares que gestionen las búsquedas de información en los sitios web, de manera que permita buscar rasgos que los caractericen y aspectos para fundamentar la propuesta de solución a la problemática planteada.

Sistémico: este método es utilizado a la hora de dar cada una de las soluciones de los elementos que se van a desarrollar para lograr que actúe como un sistema.

Modelación: este método es utilizado en la representación, mediante el uso de diagramas, de las características del sistema a desarrollar, relaciones entre objetos; y las actividades que intervienen en el proceso de configuración del sistema para la búsqueda de información.

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

Capítulo 1. Fundamentación teórica. En este capítulo se abordarán los conceptos fundamentales que permitirán entender las características del sistema que se desea desarrollar, los conceptos relacionados con el tema de investigación, el análisis de los sistemas homólogos, la definición del entorno de trabajo en el cual se verá reflejado, la fundamentación de las tecnologías, herramientas, metodologías y lenguajes que se emplearán en la construcción del sistema.

Capítulo 2. Concepción del sistema para la búsqueda de información. En este capítulo se agruparán los temas relacionados con el dominio y la caracterización del sistema que se va a desarrollar, incluyendo la selección de los requisitos funcionales que se pretenden implementar y los requisitos no funcionales, con la finalidad de formular una propuesta de solución. Además se realiza el análisis de la propuesta de solución que se propone. Se realiza el modelado y se describen los diagramas que representan las funcionalidades del sistema, aplicando los patrones de arquitectura y diseño seleccionados.

Capítulo 3. Implementación y prueba del sistema. En este capítulo se describe la implementación de la solución informática, así como los componentes que la integran. Además, se presentan los diseños de casos de prueba a utilizar en la validación del sistema y se analizan los resultados de las pruebas realizadas que permiten evaluar la calidad de la propuesta de solución.

Capítulo 1. Fundamentación teórica

Introducción

El presente capítulo incluye los principales conceptos asociados al dominio del problema, un estudio de los sistemas homólogos existentes relacionados con la investigación, las metodologías de desarrollo de *software* utilizadas, los lenguajes de modelado, los lenguajes de programación para el desarrollo del sistema y las herramientas de ingeniería de *software* asistidas por computadora.

1.1. Conceptos asociados al dominio del problema

Para entender mejor las temáticas que serán abordadas en la investigación, se hace necesario relacionar a continuación un conjunto de conceptos asociados al dominio del problema.

Indexación

La indexación es la técnica para recuperar los datos contenidos en un fichero o en una zona de memoria por medio de una estructura organizada, en la cual se almacena la información asociada al término y además la ruta o dirección donde aparece en el documento. Para lograr un buen posicionamiento (primeras posiciones de resultados) la indexación es un requisito previo. Este término también es utilizado en el ámbito económico.

Constituye además el proceso mediante el cual se registran ordenadamente datos e informaciones para elaborar su índice, el cual facilita la búsqueda de información y ayuda a seleccionar la información pertinente de acuerdo con las necesidades de los usuarios.

Es el proceso de construir un índice invertido de la información enviada por los usuarios [2]. Un índice invertido es una estructura de datos de índice de almacenamiento de una asignación de contenido, como palabras o números, a sus ubicaciones en un archivo de base de datos o en un documento o conjunto de documentos. Con un Índice invertido, las consultas se pueden resolver mediante la búsqueda de las palabras en el índice y procesamiento de sus listas correspondientes [3].

La indexación es el proceso que registra ordenadamente la información para elaborar un índice. Este procedimiento es importante para las páginas web pues permite que estas aparezcan en alguno de los buscadores más importantes; aquella página que no se haya indexado como debería, no aparecerá entre los resultados de una búsqueda [4].

Teniendo en cuenta los conceptos anteriores, en la investigación se asume que la indexación es el proceso donde se registran datos de forma organizada con la finalidad de hacer más ágil la búsqueda de una información determinada y permitiendo además devolver los resultados de la información requerida por el usuario para satisfacer de este modo su consulta.

Información

La información es un conjunto organizado de datos procesados, que constituyen un mensaje que cambia el estado de conocimiento del sujeto, así como del sistema que lo recibe.

La información está constituida por un grupo de datos supervisados y ordenados, que sirven para construir un mensaje basado en un cierto fenómeno o ente. Permite resolver problemas y tomar decisiones, ya que su aprovechamiento racional es la base del conocimiento.

Chiavenato define la información como un conjunto de datos que poseen un significado, de modo tal que reducen la incertidumbre y aumentan el conocimiento de quien se acerca a contemplarlos. Estos datos se encuentran disponibles para su uso inmediato y sirven para clarificar incertidumbres sobre determinados temas [5].

Por su parte los autores Ferrell y Hirt, plantean que esos datos y conocimientos están orientados hacia la mejora de la toma de decisiones. Si un individuo se encuentra bien informado sobre un aspecto, seguramente su decisión al respecto podrá ser más acertada que la de uno que no lo esté [6].

Otros autores que han definido la información son Czinkota y Kotabe, quienes expresan que consiste en un conjunto de datos que han sido clasificados y ordenados con un propósito determinado [7]**.**

Luego de haber analizado los conceptos anteriores, se ha llegado a la conclusión que la información son datos sobre un suceso o fenómeno particular que al ser ordenados en un contexto sirven para disminuir la incertidumbre y aumentar el conocimiento sobre un tema específico. Además puede catalogarse como el complemento de varios datos que han sido guardados con la intención de ser utilizados en determinado momento por individuos que tengan permisos para acceder a ella con la finalidad de resolver un problema y de esta manera adquirir nuevas ideas para la realización de un determinado evento.

Capítulo 1. Fundamentación teórica

Recuperación de información

La recuperación de información es el conjunto de actividades orientadas a facilitar la localización de determinados datos u objetos, y las interrelaciones que estos tienen a su vez con otros.

Otros autores que han definido la información son [8], que indican que la recuperación de información es traer documentos relevantes desde un gran archivo en respuesta a una pregunta formulada.

La recuperación de la información (RI) es un ámbito que ha tomado una gran importancia en la última década. Esto está relacionado directamente con toda la información disponible en la *World Wide Web*[1] y la necesidad de herramientas que permitan gestionar, recuperar y filtrar esta información. La RI permite identificar los documentos relevantes de los que no lo son mediante su jerarquización en el momento de realizar la búsqueda. El elemento que se encarga de dicha identificación y jerarquización se llama modelo. Un modelo no es más que una representación abstracta de un proceso, probablemente del mundo real. Es posible utilizar modelos que se relacionan con diferentes áreas matemáticas y computacionales como son la probabilidad, el álgebra, el álgebra booleana, la inteligencia artificial, entre otras [9].

Partiendo de las ideas anteriores, se llega a la conclusión que el término recuperación de información está vinculado directamente con toda aquella información disponible en la web, así como la necesidad del uso de materiales o equipos que permitan su gestión, recuperación y el filtrado de esta. Además es el conjunto de acciones que facilita la búsqueda de la información requerida por un usuario para dar respuestas a un determinado problema que le sea planteado.

Servidor de indexación

Para referirse al término servidor de indexación se debe enfatizar ante todo en el concepto de servidor. En el campo de la informática, se le denomina servidor a un nodo que forma parte de una red, el cual provee servicios a otros nodos denominados clientes. Algunos de sus servicios habituales son los servicios de archivos, que permiten a los usuarios almacenar y acceder a los archivos de una computadora y los servicios de aplicaciones, que realizan tareas en beneficio directo del usuario final [10].

En otro sentido un servidor, como la misma palabra indica, es un programa que gestiona cualquier aplicación en el lado del servidor realizando conexiones bidireccionales o unidireccionales y síncronas o asíncronas con el cliente generando una respuesta en cualquier lenguaje o aplicación en el lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Es un

[1] Red informática mundial, según su traducción del inglés.

sistema usado para estandarizar la comunicación entre distintas plataformas y lenguajes de programación [11].

Partiendo de lo antes expuesto en relación con el término servidor, se llega a la conclusión que un servidor de indexación es una aplicación informática o programa que tiene la tarea de recuperar los datos contenidos en un fichero o en una zona de memoria por medio de un índice que guarda la posición de estos, beneficiando así a otras aplicaciones que son denominadas clientes. En otro sentido, puede ser un proceso que entrega información o sirve a otro proceso.

Sitio web

Conjunto organizado y coherente de páginas web (generalmente archivos en formato HTML, PHP, CSS) y objetos (gráficos, animaciones, sonidos), que tiene como función ofrecer, informar, publicitar o vender contenidos, productos y servicios al resto del mundo. Los sitios web pueden ser visualizados o accedidos desde un amplio abanico de dispositivos con conexión a Internet, como computadoras personales, portátiles, PDA y teléfonos móviles [12].

Partiendo del concepto anterior, se llega a la conclusión que un sitio web es un área gráfica organizada que está dedicada a algún tema particular o propósito específico. Los sitios web pueden contener hiperenlaces a cualquier otro sitio web de modo que la distinción entre ellos puede ser a veces compleja.

1.2. Análisis de otras soluciones existentes

En el mundo existen varias compañías que se han dedicado al desarrollo de aplicaciones para la recuperación de información, así como su indexación. En este epígrafe se analizan algunas de las soluciones existentes con gran auge en el mercado, con el propósito de identificar sus ventajas y desventajas y a su vez determinar las razones que hacen posible su uso.

En los últimos años Cuba ha tenido un gran avance en el desarrollo de la informática, pero aun así no se tiene conocimiento de que se haya realizado algún *software* o aplicación que brinde un servicio personalizado de búsqueda en sitios web. Es por ello que en el presente documento no se exponen resultados sobre la existencia a nivel nacional de soluciones, sistemas o plataformas que tengan como propósito la puesta en marcha del proceso antes mencionado o tengan relación con este tema o problemática.

En otro sentido, refiriéndose a los avances en el desarrollo de la informática pero esta vez a nivel internacional, según investigaciones realizadas se encontró documentación sobre varios buscadores o

sistemas que ponen en práctica el proceso de búsqueda de información. A continuación se relacionan algunas características propias que permite cada uno de ellos.

Es un buscador potente, inteligente, rápido e intuitivo que aporta valor al usuario, con un catálogo muy amplio. Se puede probar durante treinta días gratis, sin límites y sin ningún compromiso. Su tecnología de búsqueda está basada en una arquitectura de procesado semántico, que asegura a los clientes encontrar los productos que buscan, disminuyendo al mínimo los errores ortográficos y tipográficos en el proceso de búsqueda. En cuanto a su tecnología de búsqueda fonética permite corregir términos mal escritos ortográficamente, pero fonéticamente correctos.

Por otro lado en cuanto a su base de conocimiento, actualmente la tecnología de Doofinder resuelve más de 50 millones de búsquedas, lo que ha permitido crear una extensa base de conocimiento, asegurando un servicio totalmente optimizado desde el primer momento en que comenzó a desarrollarse. Doofinder es multilingüe y multiplataforma, por lo que se encuentra disponible en más de 30 idiomas. En otro sentido, permite conocer qué productos están buscando los clientes, cómo los están buscando y gracias al módulo oportunidades, cuáles buscan y no encuentran por estos no encontrarse dentro del catálogo.

Según la navegación por facetas permite a los clientes navegar dentro de los resultados de su búsqueda, pudiendo afinar los resultados en función de categorías como la marca, el precio, los colores y otros. Si de velocidad se trata en solo milisegundos Doofinder devuelve resultados, lo que disminuye la tasa de abandono, mejorando la tasa de conversión del comercio electrónico [13].

Es un servicio totalmente gestionado en la nube de AWS[2] que facilita la configuración, la gestión y el escalado rentable de una solución de búsqueda para su sitio web o aplicación. Admite 34 idiomas y características de búsqueda populares, como resaltar, autocompletar y la búsqueda geoespacial. Con este servicio se puede añadir rápidamente funciones de búsqueda a su sitio web o aplicación sin la necesidad de ser un experto en búsquedas. Se encarga de aprovisionar automáticamente los recursos necesarios e

[2] *Amazon Web Services* (AWS, por sus siglas en inglés). Colección de servicios de computación en la nube.

implementa un índice de búsqueda muy ajustado. Propone automáticamente un conjunto de campos y opciones de indización. Puede añadir o eliminar campos con facilidad y personalizar las opciones de búsqueda como el facetado y el resaltado. Ofrece un escalado automático, proactivo y potente de los dominios de búsqueda, además ofrece una recuperación y supervisión automáticas de las instancias de búsqueda. Se ajusta para garantizar una baja latencia y un alto rendimiento a gran escala. Indiza y busca datos estructurados y texto sin formato. Incluye sólidas características de búsqueda, como la búsqueda por facetas, la búsqueda de texto libre, la búsqueda booleana, etc. El sistema no es rentable. Brinda un costo total de propiedad bajo para sus aplicaciones de búsqueda en comparación con el funcionamiento de un entorno de búsqueda de su propiedad. Utiliza sólidos métodos criptográficos para autenticar a los usuarios e impedir el control no autorizado de los dominios. Amazon CloudSearch admite HTTP y se integra con IAM[3] para controlar el acceso al servicio de configuración de CloudSearch y a los servicios de documentos, búsquedas y sugerencias de cada dominio [14].

La facilidad de instalación, el libre uso, el rendimiento, el uso de Internet, la rentabilidad, la seguridad y la integración de búsqueda son criterios claves a la hora de realizar una comparación con el sistema que se propone y otros como Doofinder y Amazon CloudSearch que resuelven de forma similar lo que se desea desarrollar. Se escogieron estos criterios, pues los sistemas Doofinder y Amazon CloudSearch tienen un alto uso de Internet, lo que permite ser empleado por el sistema para la búsqueda de información que se desarrolla. El sistema debe poseer un alto rendimiento, es decir, debe ser un sistema rentable, que le posibilite al usuario beneficiarse de los servicios que presta y ante todo, que sea seguro, para que el usuario sienta la necesidad de no abandonar el sistema y le dé más uso explotando aún más sus servicios. Debe ser un sistema que se integre con otros sitios web, para que se intercambie más información y más servicios, para que el usuario se sienta satisfecho y considere la necesidad de permanecer en el sistema. Por último, el sistema debe ser fácil de instalar, facilitando su interacción con los usuarios y el acceso a los contenidos mostrados.

[3] *Identity and Access Management* (IAM, por sus siglas en inglés). Identidad y Acceso de Gestión.

Tabla 1. Comparación entre sistemas

Sistema o buscador	Facilidad de instalación	Libre uso	Rendimiento	Uso de Internet	Rentabilidad	Seguridad	Integración de búsqueda
Doofinder	Fácil	Sólo 30 días	Alto	Alto	No	Alta	Se integra
Amazon CloudSearch	Medianamente fácil	Privativo	Alto	Alto	No	Alta	Se integra

Resumen del análisis de los sistemas homólogos estudiados

El estudio de los diferentes sistemas informáticos existentes a nivel internacional permitió observar y analizar las funcionalidades para la búsqueda de información. Se pudo observar que estos sistemas tienen características y funcionalidades similares, ya que fueron creados con el mismo propósito de ayudar a los usuarios en la búsqueda de información en los sitios web. Las diferencias radican en que cada uno responde a ciertos requerimientos del sistema de información para el cual se implementó. Además se tiene en cuenta la forma en la que estos sistemas realizan las búsquedas de información en los sitios web. Conocer las funcionalidades que realizan los sistemas estudiados contribuyó al mejor entendimiento del objeto de estudio, pero se determinó que de estos sistemas, Doofinder es el que está más acorde con lo que se quiere diseñar según las necesidades de búsqueda de información en el ámbito nacional.

1.3. Tecnologías de desarrollo

Teniendo en cuenta las necesidades vistas y las características del entorno tecnológico definido en el centro CIDI, entidad donde se aplicará la solución propuesta, se realizó un estudio de las tendencias y tecnologías actuales posibles a emplear, descritas a continuación.

Metodología de desarrollo de *software*

La selección de la metodología adecuada, garantiza la creación de un *software* de calidad y desarrollo de un producto en el tiempo planificado y con los costos previamente establecidos.

Una metodología de desarrollo de *software* es un conjunto de pasos, procedimientos, técnicas y ayudas a la documentación que deben seguirse para desarrollar *software* [15]. Las metodologías de desarrollo de *software* constituyen una filosofía de trabajo que proporciona una base de procesos para llevar a cabo con

éxito cualquier proyecto informático. Existen dos grupos principales de metodologías: metodologías pesadas o tradicionales y metodologías ágiles. Cada una en particular propone un conjunto de procedimientos, técnicas, herramientas y soporte documental necesarios en todo proceso de ingeniería de *software* y que además, se corresponde con un determinado enfoque que define la forma en que se realizan las actividades. Las metodologías de desarrollo permiten conocer qué persona hace una determinada actividad, cuándo y cómo la debe hacer.

Proceso Unificado Abierto (*Open Unified Process*, *OpenUP*): Proceso de desarrollo unificado basado en *Rational Unified Process* (RUP, por sus siglas en inglés). OpenUp es una metodología de desarrollo diseñada para pequeños equipos de producción enfocados al desarrollo ágil. Define cuatro fases: Inicio, Elaboración, Construcción y Transición. Se caracteriza por permitir la colaboración para alinear los intereses y un entendimiento compartido, llevar un control de las necesidades y costos técnicos, y definir la evolución continua para reducir riesgos y obtener resultados.

Son diversos los beneficios de la utilización de esta metodología:

- Es apropiada para proyectos pequeños y de bajos recursos, permite disminuir las probabilidades de fracaso en los proyectos pequeños e incrementar las probabilidades de éxito.
- Permite detectar errores tempranos a través de un ciclo iterativo.
- Por ser una metodología ágil tiene un enfoque centrado en el cliente y con iteraciones cortas.
- Evita la elaboración de documentación, diagramas e iteraciones innecesarios requeridos en la metodología RUP [16].

Las características del equipo de desarrollo de este trabajo cumplen con los requisitos asociados con la metodología mencionada, dado que se está en presencia de un equipo pequeño que requiere la constante comunicación y que deberá realizar el proceso de implementación de forma iterativa según las necesidades de trabajo y requisitos previos. Al mismo tiempo, es necesario destacar que dicha metodología es una variante ligera de RUP y está contenida en la base tecnológica definida por el departamento SINI[4] del centro de desarrollo de Ideoinformática (CIDI) de la Facultad 1, centro al cual pertenecen los desarrolladores. Por otro lado, se selecciona el enfoque ágil, pues constituye un enfoque común, que posee competencia y una capacidad de resolución de problemas, apoya el proceso de toma

[4] Soluciones Informáticas para Internet (SINI, por sus siglas en español).

de decisiones, impone confianza y respeto mutuo y lo más importante posibilita una organización propia a la hora de desarrollar el sistema [17].

Lenguaje Unificado de Modelado (UML 2.0)

Es el Lenguaje Unificado de Modelado (UML, según sus siglas en inglés), orientado a objetos estándar de la industria para especificar, visualizar, construir y documentar los elementos de los sistemas de *software*. UML proporciona una forma estándar de escribir los planos de un sistema, cubriendo tanto aspectos conceptuales, tales como procesos del negocio y funciones del sistema, como las clases escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes de *software* reutilizables [18].

Herramienta Case

Las herramientas CASE (*Computer Aided Software Engineering*, según el significado de sus siglas en inglés) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software*, reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del *software* en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, compilación automática, documentación o detección de errores, entre otras [19].

Visual Paradigm (versión 8.0)

Para el modelado del *software* se utiliza como herramienta CASE*, Visual Paradigm* que utiliza UML como lenguaje de modelado. Es fácil de usar y soporta todo el ciclo de vida del desarrollo de *software*: análisis, diseño orientado a objetos, construcción, prueba y despliegue. Permite realizar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación para facilitar una mayor comprensión de la implementación. Adicionalmente es sencilla de instalar, fácil de utilizar y actualizar. Además es multiplataforma y cuenta con una abundante documentación, como son: tutoriales, demostraciones interactivas, entre otros recursos [20].

Lenguaje de programación

Un lenguaje de programación es una técnica estándar de comunicación que permite expresar las instrucciones que han de ser ejecutadas en una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen un programa informático. Un lenguaje de programación permite a un

programador especificar de manera precisa: sobre qué datos una computadora debe operar, cómo deben ser almacenados y transmitidos estos datos y, qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural [21].

PHP (versión 5.3.8)

PHP (*Hypertext Pre-processor,* en español Pre-procesador de Hipertextos), es un lenguaje de programación de código abierto e interpretado, diseñado originalmente para la creación de páginas web dinámicas. Se usa principalmente para la interpretación del lado del servidor (*server-side scripting*) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de diferentes tipos de programas incluyendo aplicaciones con interfaz gráfica. Como característica muy importante, se puede decir que puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. Además, permite la conexión a diferentes tipos de servidores de bases de datos tales como: MySQL, PostgreSQL, Oracle, ODBC, DB2, Microsoft SQL Server y SQLite. Este lenguaje también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como Unix (Linux o Mac OS X) y Microsoft Windows [22].

Marco de trabajo Symfony (versión 2.3.6)

Symfony en su versión 2.3 es un *framework* de aplicaciones web escrito en PHP que sigue el modelo-vista-controlador (MVC). Tiene como objetivo acelerar la creación y mantenimiento de aplicaciones web y sustituir repetitivas tareas de codificación.

Symfony 2.3.6 es una de las versiones populares de Symfony para desarrollar aplicaciones PHP. Symfony 2.3.6 ha sido ideado para potenciar al límite todas las características nuevas de PHP 5.3 y por eso es uno de los *frameworks* PHP con mejor rendimiento [23].

Características:

- Fácil de instalar y configurar en la mayoría de plataformas (y garantizado para trabajar en el estándar de *Windows*).
- Base de datos del motor independiente.
- Permite configuraciones de seguridad.
- Fácil de usar, en la mayoría de los casos, pero todavía lo suficientemente flexible como para adaptarse a los casos complejos.

- Partiendo de la premisa de convención sobre configuración el desarrollador tiene que configurar solo lo no convencional.
- Compatible con patrones de diseño [23].

Entorno de desarrollo integrado Netbeans IDE

El entorno de desarrollo integrado (IDE) NetBeans en su versión 8.0.1 está disponible para Windows, MacOS, Linux y Solaris. Es un IDE de código abierto y una plataforma de aplicación, que puede ser usada como una estructura de soporte general para compilar cualquier tipo de aplicación. Es una herramienta para que los programadores puedan escribir, depurar y ejecutar programas. Está escrito en Java, es un producto libre y gratuito sin restricciones de uso [24].

Algunas de sus características:

- Propone un esqueleto para organizar el código fuente, el editor conjuntamente integra los lenguajes como HTML, JavaScript y CSS. Además posee un sistema para examinar todos los directorios de cada proyecto, haciendo reconocimiento y carga de clases, métodos y objetos, para acelerar la programación.
- El editor de PHP, es ágil y robusto, contiene ayuda en línea, reconocimiento de sintaxis y todo lo que provee la última versión de PHP [24].

PgAdmin lll (versión 1.16.1)

PgAdmin III es el sistema para gestionar las bases de datos que se utiliza para el diseño y sistema de gestión en los sistemas *Windows*. Es libremente disponible bajo los términos de la Licencia Artística y puede ser redistribuido siempre que los términos de la licencia se cumplen. PgAdmin 1.16.1 es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia *Open Source*. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma [25].

PostgreSQL 9.2

PostgreSQL 9.2 es el sistema de gestión de base de datos relacional orientada a objeto, publicado bajo la licencia BSD. Es el sistema gestor de bases de datos (SGBD) de código abierto más potente del mercado.

Principales ventajas:

- Estabilidad, flexibilidad y alto rendimiento.

- Administración efectiva de seguridad.
- Ágil navegación y administración de base de datos.
- Excelentes herramientas visuales y de texto para elaboración de consultas.
- Se puede extender su funcionalidad.
- Permite crear o migrar aplicaciones desde Access, Visual Basic, Visual Fox Pro, y Visual C/C.
- Varias Interfaces de Programación: ODBC, JDBC, C/C++, SQL embebido, Perl, Python, PHP.
- Conexión vía reenvío de puerto local a través del túnel SSH.
- Impresionantes opciones de exportación e importación de datos [26].

Solr (versión 4.10.3)

Apache Solr es un servidor de indización basado en Apache Lucene. Posee una API REST[5] que permite a los desarrolladores de *software* abstraerse de la complejidad de Lucene. Permite manipular de manera sencilla índices distribuidos [27].

Solr es la popular e increíblemente rápida plataforma de búsqueda empresarial, y de código abierto basada en Apache Lucene. Es altamente fiable, escalable y tolerante a fallos, proporciona un distribuido de indexación, la replicación y la consulta con equilibrio de carga, conmutación por error automatizada y recuperación, así como la configuración centralizada y más [28].

Solr proporciona un número de características de consultas potentes, incluyendo:

- Lógica condicional utilizando *AND, OR* y *NOT.*
- La coincidencia con comodín.
- Consultas rango de fechas y números.
- Frase consulta con decantación para permitir una cierta distancia entre los términos.
- Expresiones regulares.
- Consultas de función [29].

1.4. Conclusiones parciales

El análisis de los términos asociados al objeto de estudio permitió definir el marco conceptual de la investigación.

[5] *Application Programming Interface* (API, por sus siglas en inglés). Interfaz de Programación de Aplicaciones, o librería de funciones a la que se accede por el protocolo HTTP.

Después de haber analizado las tendencias en el desarrollo de sistemas para la búsqueda de información, se identificó que estos son sistemas complejos, con inmensidad de opciones de configuración, privativos y exigen un acceso pleno a Internet.

El estudio de las tecnologías definidas para el desarrollo del sistema para la búsqueda de información en el centro CIDI permitió asumir que para el transcurso de la investigación se tengan en cuenta que la tendencia en este tipo de sistemas en relación a lo estudiado es a plataformas ofrecidas como servicio sin la necesidad de la instalación de *software* o *hardware* en los servidores de los clientes.

Las soluciones existentes hoy en día que brindan un servicio personalizado de búsqueda en los sitios web no son favorables para Cuba por ser privativas y por no tener esta un acceso pleno a Internet.

Capítulo 2. Concepción del sistema para la búsqueda de información

Introducción

En el presente capítulo se abordarán los aspectos fundamentales relacionados con el diseño del sistema a desarrollar. Entre los elementos a destacar se encuentran la descripción de la propuesta de solución y el diagrama del modelo del dominio, mediante el cual se representan los objetos reales que intervienen en el proceso de búsqueda de información. Como vía para definir las futuras funcionalidades de la aplicación y qué usuarios podrán tener acceso a las mismas, se generaron los artefactos relacionados con las especificaciones de los requerimientos funcionales y no funcionales que deberá poseer el sistema, así como la especificación de los casos de uso. Como parte del diseño de la aplicación se definieron el estilo y los patrones de arquitectura y diseño que se emplearán en el desarrollo de la propuesta de solución.

2.1. Caracterización del entorno

Actualmente en la Universidad de las Ciencias Informáticas (UCI) existen soluciones que permiten realizar búsquedas locales a texto completo, pero no existe un sistema capaz de realizar búsquedas de información mediante un enlace con otros sitios web y que muestre o archive en su contenido toda la información que ha sido solicitada por un usuario. Para eliminar este problema se propone desarrollar un sistema que permita mejorar la estrategia de la búsqueda de información a texto completo para lograr la obtención de gran cantidad de información en el sitio con el fin de beneficiar al usuario disminuyendo el tiempo empleado en el análisis de este proceso.

Propuesta de solución

Debido a que las soluciones existentes no logran satisfacer las necesidades del usuario, pues las búsquedas de información realizadas localmente en los sitios web son una responsabilidad de los administradores, además de que cada sistema para la gestión de contenido implementa su propio mecanismo de búsqueda, se hace necesario construir un sistema que facilite la búsqueda de información a texto completo en los sitios web.

Algunos de los beneficios esperados de esta investigación, una vez que sea implantada, evaluada y probada son:

- El sistema mostrará al usuario aquellas opciones a las que está autorizado a acceder y realizar búsquedas de información haciendo uso de las funcionalidades permitidas teniendo en cuenta el rol que desempeñe en el sistema.
- El sistema ofrecerá la búsqueda de información a través de una interfaz web y brindará la posibilidad de registrar y autenticar clientes, configurar diversos parámetros y visualizar estadísticas de interés.
- El sistema garantizará la seguridad de los datos almacenados en Solr y permitirá importar datos a través del protocolo HTTP.
- El sistema permitirá visualizar las respuestas de las búsquedas realizadas por el usuario.

2.2. Entendimiento del negocio

El entendimiento del negocio posibilita obtener una visión más clara del proceso en cuestión. Para ello se utiliza el modelo de dominio, también conocido como modelo conceptual, es un artefacto construido mediante el lenguaje UML, que se representa como un diagrama de clases donde los contenidos no son conceptos propios de un sistema de *software* sino de la propia realidad física. El modelo de dominio permite ser utilizado por el analista como un medio para comprender el sector industrial o de negocios al cual el sistema va a servir [30].

Descripción de clases del modelo de dominio

La modelación del dominio constituye la herramienta esencial para garantizar la comprensión y descripción de las clases o conceptos y sus relaciones más importantes dentro del contexto del problema. A continuación se muestra la descripción de los conceptos identificados.

Tabla 2. Descripción de las clases del modelo del dominio

Concepto	Descripción
Administrador (*webmaster*)	Usuario con permisos que gestiona, administra y configura las funcionalidades del sistema.
Usuario	Usuario que gestiona y muestra estadísticas de los *feed* y genera los *script*.
Feed de datos	Son todos los datos organizados que se obtienen luego de la búsqueda de información

	realizada.
Sitio web	Área gráfica organizada que está típicamente dedicada a algún tema particular o propósito específico, la cual será navegable por el usuario.

Diagrama o modelo de dominio

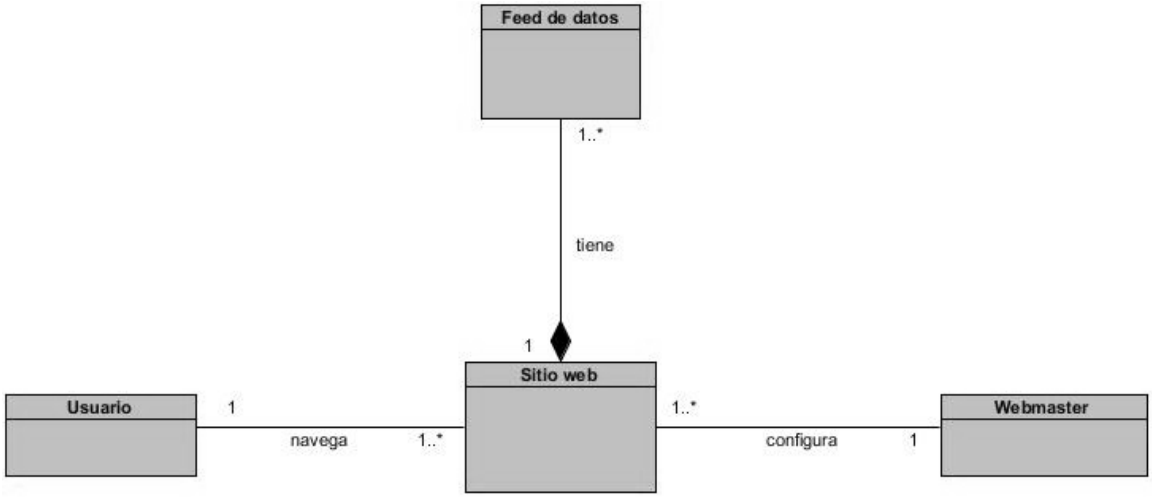

Figura 1. Modelo de dominio

El modelo de dominio representado en la figura anterior, explica gráficamente los conceptos importantes y muestra la relación existente entre los objetos fundamentales que intervienen en el funcionamiento del sistema para la búsqueda de información.

2.3. Especificación de requisitos

Una de las actividades más importantes en el desarrollo de *software* es la captura de requisitos, debido a que la identificación correcta de los mismos influye directamente en la calidad del proceso y producto final. Los requisitos del *software* constituyen las descripciones de los servicios y funcionalidades que brinda una aplicación informática, teniendo en cuenta además las restricciones operativas a las cuales están sujetas. De manera general, estos requisitos expresan las necesidades objetivas que presentan los usuarios, ante un sistema que resuelve un problema en particular de un determinado dominio [31].

Una vez definidos los conceptos principales relacionados con el dominio, se muestran a continuación los requerimientos funcionales y no funcionales, teniendo en cuenta el objetivo planteado al inicio del trabajo.

Capítulo 2. Concepción del sistema para la búsqueda de información

Requisitos funcionales

Un requisito funcional es una característica requerida por un sistema que expresa una capacidad de acción del mismo o una funcionalidad [32]. A continuación se listan los requisitos funcionales identificados para el sistema a desarrollar. En el **Anexo # 1** se muestra una tabla que contiene estos requisitos funcionales, cada uno de ellos con la prioridad que poseen de acuerdo a su importancia en el sistema y el caso de uso del sistema con el que están relacionados.

RF: Constituye la nomenclatura utilizada para hacer referencia a cada requisito funcional.

RF 1. Autenticar usuario.

RF 2. Registrar usuario.

RF 3. Ver perfil de usuario.

RF 4. Cambiar perfil de usuario.

RF 5. Confirmar usuario.

RF 6. Gestionar los usuarios con acceso al sistema.

RF 6.1. Mostrar listado de los usuarios registrados.

RF 6.2. Mostrar detalles de un usuario.

RF 6.3. Registrar los datos de un nuevo usuario.

RF 6.4. Modificar los datos de un usuario.

RF 6.5. Eliminar los datos de un usuario.

RF 7. Gestionar *feed* de datos.

RF 7.1. Mostrar listado de los *feed* obtenidos.

RF 7.2. Mostrar detalles de un *feed*.

RF 7.3. Registrar los datos de un nuevo *feed*.

RF 7.4. Modificar los datos de un *feed*.

RF 7.5. Eliminar los datos de un *feed*.

RF 8. Generar *script* para el uso del servicio que brinda el sistema.

RF 9. Importar *feed* de datos.

RF 10. Generar gráficos de estadísticas de los *feed* de datos.

RF 10.1. Mostrar gráfico estadístico de los *feed* de datos filtrado por cantidad de *feed* o rango de tiempo.

RF 11. Realizar búsqueda sobre un *feed* de datos.

RF 12. Gestionar propiedades del sistema.

RF 12.1. Mostrar listado de las propiedades del sistema registradas.

RF 12.2. Mostrar detalles de una propiedad del sistema.

RF 12.3. Modificar los datos de una propiedad del sistema.

Requisitos no funcionales

La calidad y eficiencia de la solución propuesta no solo depende de las capacidades o condiciones que debe cumplir el sistema sino que también procura ser más atractivo al cliente, más seguro y rápido.

Los requerimientos no funcionales se presentan, en la mayoría de los casos, como las propiedades o cualidades que el sistema debe poseer. No obstante, desde otras aristas, pueden concebirse como las restricciones de las funcionalidades del sistema [33].

RNF: Constituye la nomenclatura utilizada para hacer referencia a cada requisito no funcional.

A continuación se describen las cualidades con las que debe cumplir el sistema:

Requerimientos de *software*

RNF 1. Se requiere una distribución GNU/Linux como sistema operativo.

Requerimientos de *hardware*

RNF 2. El servidor web debe poseer un procesador Intel o similar, a 3.30 GHz o superior.

RNF 3. El sistema necesita disponer de un servidor con al menos 2 GB de memoria RAM y 160 GB HDD o superior.

RNF 4. El sistema debe ejecutarse en Mozilla Firefox en su versión 5.0 e Internet Explorer en su versión 8, según la documentación de Bootstrap.

Requerimientos de diseño e implementación

RNF 5. El sistema deberá ser compatible con PHP en su versión 5.3.8 o superior.

RNF 6. El *framework* de desarrollo que se debe utilizar es Symfony en su versión 2.3.6.

RNF 7. El IDE que se debe utilizar es el NetBeans en su versión 8.0.1 con el componente de PHP integrado.

RNF 8: La herramienta de modelado que se debe utilizar es *Visual Paradigm* en su versión 8.0.

RNF 9: El sistema debe utilizar para su maquetado Bootstrap en su versión 3.1.1.

Requerimientos de interfaz de usuario

RNF 10. La interfaz del sistema debe ser sencilla, intuitiva y amigable para sus usuarios.

RNF 11. El sistema permitirá visualizar el contenido de la información de forma organizada y legible.

Requerimientos de seguridad

RNF 12. El acceso al sistema se realizará mediante una conexión segura por HTTPS.

RNF 13. La seguridad se establecerá por roles que se le asignarán a los usuarios que interactúen con el sistema (administrador y usuario).

RNF 14. Toda la información debe estar protegida del acceso no autorizado.

Requerimientos de usabilidad

RNF 15. El sistema deberá poseer una interfaz y navegación asequible y funcional tanto para usuarios expertos como para los que no tienen conocimientos profundos del sistema.

Requerimientos de disponibilidad

RNF 16. Los usuarios del sistema deben tener acceso (según sus permisos) en todo momento a la información solicitada.

Casos de uso del sistema

El modelo de casos de uso proporciona la entrada fundamental para el análisis, el diseño y las pruebas; este tiene como objetivo definir y describir las funcionalidades del *software* a desarrollar. Los casos de uso son fragmentos de funcionalidades que el sistema ofrece para aportar un resultado de valor para sus actores [33]. De manera más precisa, un caso de uso especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores. En el presente documento se especifican los casos de uso del sistema a desarrollar y algunas de sus especificaciones.

Diagrama de casos de uso del sistema

A continuación se presenta el diagrama de casos de uso del sistema donde se muestra la relación entre los actores que deberán interactuar con el sistema y los casos de uso que le corresponden a cada uno.

CU: Es la nomenclatura utilizada para referirse a cada caso de uso.

Capítulo 2. Concepción del sistema para la búsqueda de información

Figura 2. Diagrama de casos de uso del sistema

En el diagrama anterior se encuentran representados en total, 16 casos de uso, y la jerarquía de actores que interactúan con cada uno de ellos, teniendo en cuenta los permisos y roles correspondientes. Se identificaron varios CU que se consideran críticos debido a la relevancia que tienen para el correcto funcionamiento del sistema en general. Los CU críticos se consideran importantes para los usuarios porque cubren las principales tareas o funciones que el sistema ha de realizar. Definen la arquitectura básica del sistema que se propone. Son los CU arquitectónicamente significativos [34].

CU Gestionar usuarios con acceso al sistema: se encarga de todos los procesos asociados al usuario que interactúa con el sistema: listar los usuarios registrados, registrar los datos de un nuevo usuario, modificar los datos de un usuario, entre otros.

CU Gestionar *feed* de datos: engloba todos los procesos asociados a los *feed* de datos, que no son más que un conjunto o una estructura de datos organizados que importa el propio sistema; entre los procesos se encuentran: listar los *feed* obtenidos, mostrar detalles de un *feed*, registrar los datos de un nuevo *feed*, entre otros.

CU Generar *script*: se refiere al proceso de generar un *script* determinado que tendrá el sistema para que el usuario pueda hacer uso de los servicios que brinda el propio sistema.

CU Autenticar usuario: se refiere al proceso de autenticación del usuario en el sistema. Es uno de los casos de uso más importantes para los usuarios porque cubre una de las principales tareas o funciones que el sistema ha de realizar.

Con el objetivo de lograr una mayor reutilización y una mejor representación en los diagramas, existen varios casos de uso incluidos, ejemplo de ello son:

CU Filtrar por cantidad: representa la opción de filtrar los resultados de los reportes y gráficos estadísticos por cantidad de datos en los *feed*.

CU Filtrar por rango de tiempo: representa la opción de filtrar los resultados de los reportes y gráficos estadísticos de la cantidad de datos en los *feed* por rangos de tiempos.

También se muestra un caso de uso extendido que posibilita representar funcionalidades que no se ejecutarán siempre, sino bajo determinadas circunstancias dentro de otros casos de uso:

CU Mostrar gráfico estadístico: se refiere a la opción de mostrar todos los resultados de los *feed* de datos en un rango de tiempo definido por el actor.

2.4. Patrones de casos de uso utilizados

Los patrones de casos de uso posibilitan una alta comprensión del comportamiento del sistema y generalmente son utilizados como plantillas que describen cómo deberían ser estructurados y organizados los casos de uso. A continuación se describen aquellos empleados en la representación del diagrama de casos de uso descrito en la sección anterior.

CRUD Parcial: el empleo de este patrón permite simplificar el tamaño y facilita el análisis del modelo mediante el agrupamiento de varias operaciones dentro de un caso de uso, cuando estas contribuyen al

mismo valor del negocio y existe alguna que por su complejidad, extensión o significación, debe modelarse como un caso de uso independiente.

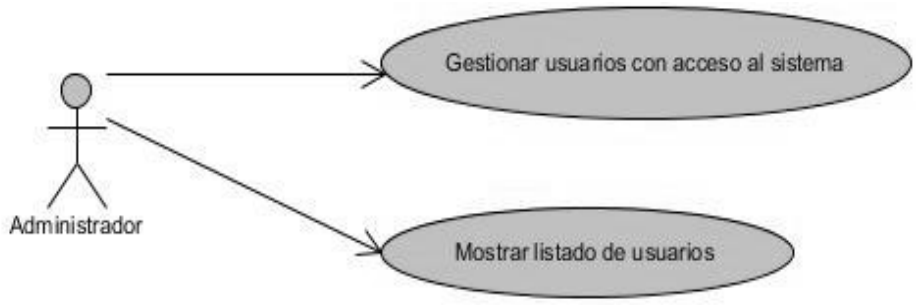

Figura 3. Ejemplo de uso del patrón de CU CRUD - Parcial

Múltiples Actores - Rol Común: este patrón es utilizado cuando existe una sola entidad externa que interactúa con cada instancia de un caso de uso; lo usual es que esta situación ocurra cuando dos o más personas con roles diferentes dentro del negocio interactúen de la misma forma con un caso de uso.

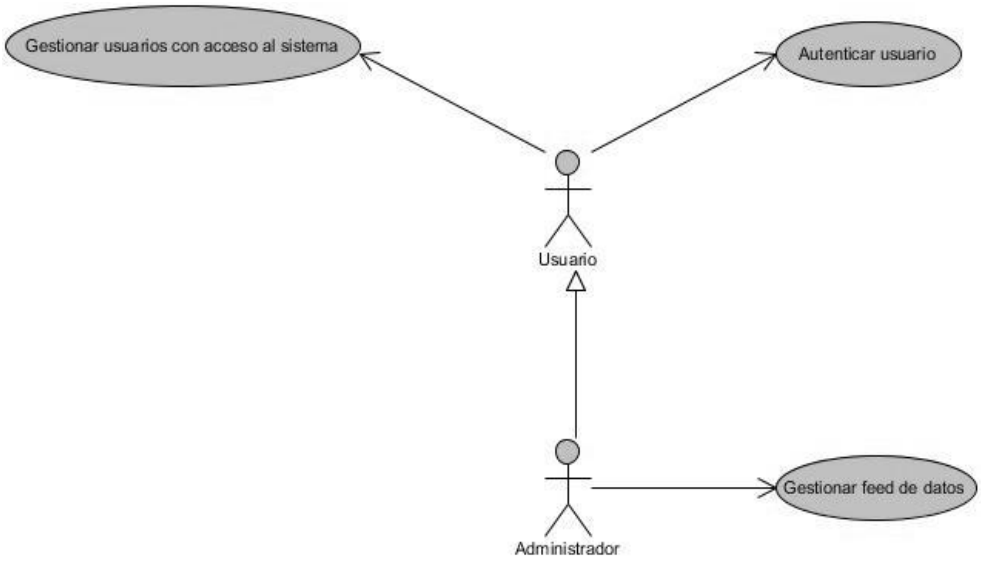

Figura 4. Ejemplo de uso del patrón de CU Múltiples Actores - Rol Común

2.5. Especificación de casos de uso del sistema

La especificación de casos de uso del sistema tiene como objetivo mostrar una descripción detallada de la secuencia de las acciones que se realizan en cada caso de uso.

A continuación se realiza la descripción de uno de los CU críticos del sistema, el resto de las descripciones se encuentran en el **Anexo # 2.**

Capítulo 2. Concepción del sistema para la búsqueda de información

Descripción del CU "Gestionar *feed* de datos"

Tabla 3. Descripción del CU Gestionar *feed* de datos

Objetivo	Mostrar listado, mostrar detalles, registrar, modificar y eliminar los datos de un *feed*.
Actores	Usuario
Resumen	El caso de uso se inicia cuando el usuario decide mostrar listado, mostrar detalles, registrar, modificar y eliminar datos de un *feed*.
Complejidad	Media
Prioridad	Crítico
Requisitos funcionales que se relacionan	RF 7.1, RF 7.2, RF 7.3, RF 7.4, RF 7.5
Precondiciones	El usuario debe estar autenticado en el sistema. El usuario debe tener permisos en el sistema para acceder a sus *feed* de datos.
Postcondiciones	Queda mostrado el listado. Quedan mostrados los detalles, registrados, modificados o eliminados los datos de un *feed*.
Flujo de eventos	
Flujo básico Gestionar *feed* de datos	

	Actor	**Sistema**
1.	1. Accede al sistema para gestionar sus *feed* de datos.	Permite realizar varias acciones con un *feed* de datos: -Mostrar listado de los *feed* obtenidos. -Mostrar detalles de un *feed*. -Registrar los datos de un nuevo *feed*. Ver sección 1: "Registrar los datos de un nuevo *feed*." -Modificar los datos de un *feed*. Ver sección 2: "Modificar los datos de un *feed*." -Eliminar los datos de un *feed*. Ver sección 3: "Eliminar los datos de un *feed*."
2.		Termina el CU.

Sección 1: "Registrar los datos de un nuevo *feed*."		
Flujo básico Gestionar *feed* de datos.		
	Actor	**Sistema**
1.	1. Selecciona la opción Registrar los datos de un nuevo *feed*.	2. El sistema muestra un formulario con los campos necesarios para registrar los datos de un nuevo *feed*.
2.	3. Llena los campos *(*nombre, url*)*	4. Verifica que todos los campos han sido llenados correctamente. Registra correctamente los datos del nuevo *feed* y muestra un mensaje: "El *feed* de datos se ha registrado correctamente, seleccione los campos."
Flujos alternos		
Nº Evento "Falta un campo por llenar y el formato en el campo fecha de actualización es incorrecto"		
	Actor	**Sistema**
1.		4.1 Si se dejó algún campo obligatorio en blanco el sistema muestra un mensaje de error indicando que debe llenar el campo obligatorio y pasa a la acción 3 del flujo normal de los eventos.
2.		4.2 Si entra una url con formato incorrecto, el sistema muestra el mensaje de error indicando que el campo url no es válido.
3.		4.3 El sistema regresa al paso 2 del flujo normal de eventos.
4.	5. Presiona el botón "Cancelar"	
Sección 2: "Modificar los datos de un *feed*."		
Flujo básico Gestionar *feed* de datos.		
	Actor	**Sistema**

1.	1. Selecciona la opción Lista de *feed* de datos.	2. Muestra un listado con todos los *feed* que están almacenados en el sistema junto con la opción de modificar.
2.	3. Selecciona la opción modificar.	4. Muestra los campos para modificarlos.
3.	5. Modifica los datos deseados.	6. Verifica que todos los campos han sido llenados correctamente. Modifica correctamente los datos de un *feed* y muestra un mensaje: “Los cambios han sido guardados.”
Flujos alternos		
Nº Evento “Falta un campo por llenar y el formato en el campo fecha de actualización es incorrecto”		
	Actor	**Sistema**
1.		6.1 Si se dejó algún campo obligatorio en blanco el sistema muestra un mensaje de error indicando que se debe llenar el campo obligatorio y pasa a la acción 3 del flujo normal de eventos.
2.		6.2 Si entra una url con formato incorrecto, el sistema muestra el mensaje de error indicando que el campo url no es válido.
3.		6.3 El sistema regresa al paso 2 del flujo normal de eventos.
4.	7. Presiona el botón “Cancelar”	
Sección 3: “Eliminar los datos de un *feed*.”		
Flujo básico Gestionar *feed* de datos.		
	Actor	**Sistema**
1.	1. Selecciona la opción Lista de *feed* de datos.	2. Muestra un listado con todos los *feed* de datos que están almacenados en el sistema junto con la opción de eliminar.
2.	3. Selecciona la opción Eliminar.	4. Muestra un mensaje: “¿Desea eliminar su *feed* de datos?” junto con la opción de confirmar o cancelar.

		Se elimina el *feed* de datos y el sistema muestra el mensaje: "*Feed* eliminado correctamente."
Flujos alternos		
Nº Evento "Presiona el botón cancelar".		
1.	**Actor**	**Sistema**
2.		4.1 El sistema regresa al paso 2 del flujo normal de eventos.
3.	5. Presiona el botón "Cancelar".	

2.6. Modelo lógico de la base de datos

En el proceso de abstracción que conduce a la creación de la base de datos, desempeña una función prioritaria el modelo de datos. El modelo de datos, como abstracción del universo de discurso, es el enfoque utilizado para la representación de las entidades y sus características dentro de la base de datos. Es una colección de herramientas conceptuales para describir los datos, las relaciones que existen entre ellos, semántica asociada a los datos y restricciones de consistencia [35].

A continuación se describen las colecciones de la base de datos, que se ilustran en la Figura 5:

Usuario: almacena los datos de todos los usuarios registrados en el sistema.

***Feed*:** contiene los datos asociados a un *feed* de datos.

***Collection* 1:** almacena la información del sitio que está asociada al *feed* de datos.

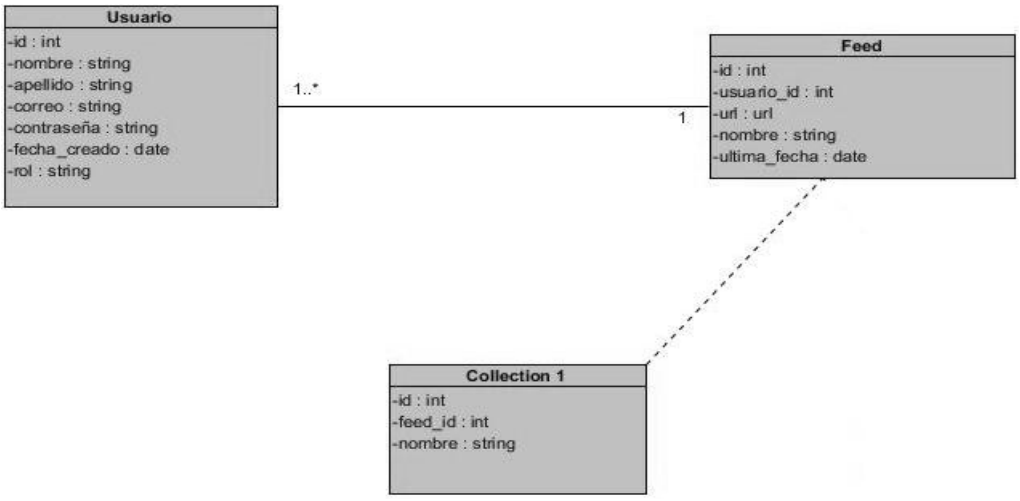

Figura 5. Modelo lógico de la base de datos

Capítulo 2. Concepción del sistema para la búsqueda de información

2.7. Arquitectura del sistema

La arquitectura de *software* es la estructura de un sistema, que contiene elementos del *software*, las propiedades externas visibles de esos elementos y las relaciones entre ellos [36]. Es el resultado de ensamblar un cierto número de elementos arquitectónicos, para satisfacer la funcionalidad y ejecución de los requisitos del sistema, así como los requisitos no funcionales del mismo, como la fiabilidad, escalabilidad, portabilidad y disponibilidad. Por consiguiente, se puede concluir que una correcta definición de la arquitectura hace factible el diseño y la construcción de los componentes y sus relaciones.
A continuación se describe la arquitectura cliente-servidor, empleada en el desarrollo de aplicaciones web.

Estilo arquitectónico

Se utiliza como estilo arquitectónico llamada y retorno (ver Figura 6).
El estilo arquitectónico llamada y retorno es una estructura de programa clásica que descompone la función en una jerarquía de control, donde el programa principal invoca un número de programas que a su vez pueden invocar a otros componentes. Permite al diseñador del *software* (arquitecto del sistema) construir una estructura de programa relativamente fácil de modificar y ajustar a escala.

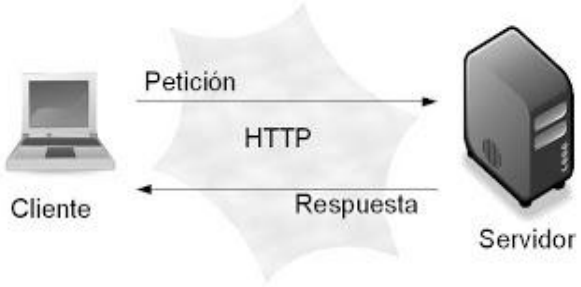

Figura 6. Estilo llamada y retorno

Algunas de las características del estilo arquitectónico llamada y retorno son:

- Hilo de control simple soportado por los lenguajes de programación.
- Usa una estructura implícita de subsistemas.
- Pretende incrementar el desempeño distribuyendo el trabajo en múltiples procesadores.

Tiene como ventajas:

- Es utilizado en grandes sistemas de *software*.
- La descomposición en módulos disminuye la complejidad [37].

En esta arquitectura los clientes acceden a los servicios proporcionados por un servidor a través de llamadas a procedimientos remotos, usando un protocolo de petición y respuesta tal como el protocolo HTTP usado en la *World Wide Web*. Usualmente el trabajo pesado lo hace el servidor, mientras que los

procesos del cliente solo se ocupan de la interacción con el usuario a través de interfaces gráficas. Esto permite distribuir físicamente los procesos y los datos, reduciendo el tráfico de la red.

Patrón arquitectónico

Un patrón es la forma o método con el que se han podido solucionar de forma simple y elegante problemas en el desarrollo de aplicaciones. Son soluciones que están apoyadas en la experiencia de las compañías y en general, de los desarrolladores [38].

Un patrón arquitectónico representa un esquema de organización estructural de gran importancia para el desarrollo de sistemas de *software*. Además, proporciona un grupo de subsistemas predefinidos, especificando sus responsabilidades e incorporando algunas reglas y guías para organizar las relaciones entre ellos [38].

En el desarrollo web existe un constante cambio, debido a esto los patrones deben ser bien aplicados con el objetivo de mantener un lenguaje común en una comunidad de desarrolladores. Uno de los patrones arquitectónicos utilizados frecuentemente en el desarrollo de aplicaciones web es el Modelo Vista Controlador (MVC). Esto se debe a que permite separar en capas las partes de una aplicación. Por tanto, para lograr una correcta compatibilidad con el mismo, el sistema a desarrollar se basa en la implementación de este patrón. A continuación se describe su funcionamiento en el marco de trabajo Symfony.

Implementación del Modelo Vista Controlador por Symfony:

Symfony2 basa su funcionamiento interno en la arquitectura Modelo – Vista – Controlador (MVC). No obstante, según su creador Fabien Potencier: "Symfony2 no es un *framework* MVC. Symfony2 proporciona herramientas para la parte del Controlador y de la Vista. La parte del Modelo es responsabilidad del programador" [39].

El patrón MVC permite definir claramente las capas que se aplican en la arquitectura cliente-servidor; separa la parte gráfica de una aplicación de los procesos lógicos y de los datos de la misma. Este patrón se divide en tres componentes fundamentales [40].

En cualquier caso, resulta esencial conocer cómo se aplican los principios fundamentales de la arquitectura MVC a las aplicaciones Symfony2. La siguiente figura muestra el esquema simplificado del funcionamiento interno de Symfony2.

Capítulo 2. Concepción del sistema para la búsqueda de información

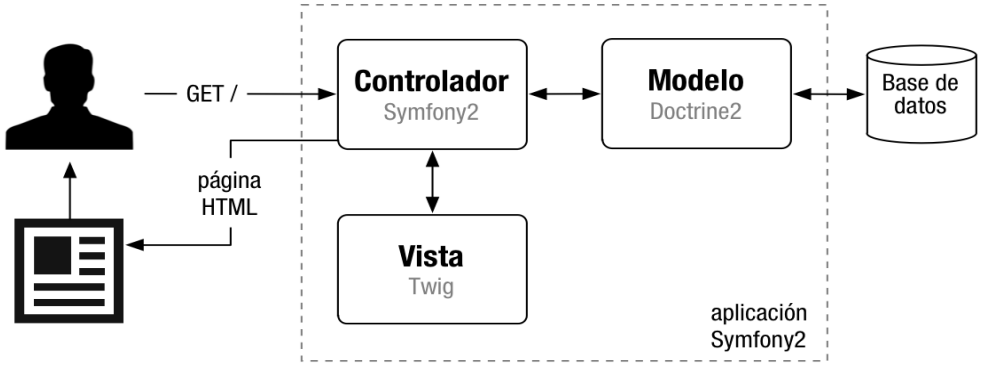

Figura 7. Esquema simplificado de la arquitectura interna de Symfony2

Modelo: representa la información con la que trabaja la aplicación, o sea, su lógica de negocio. Gestiona los datos solicitados por el controlador. Esto se evidencia cuando se necesita mostrar la fuente de datos.

Vista: convierte el modelo en una página web que facilita al usuario interactuar con ella. Contiene los elementos asociados a la presentación de los datos y la información visual de los elementos que conformarán el reporte, encontrándose el código asociado a las interfaces, que se encargan de visualizar la información que el controlador devuelve.

Controlador: es el encargado de procesar las interacciones del usuario y ejecuta los cambios adecuados en el modelo o en la vista. La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista), lo que permite un mantenimiento más sencillo de las aplicaciones. El controlador es el encargado de aislar al modelo y a la vista de los detalles del protocolo usado para las peticiones (HTTP, consola de comandos, correo electrónico). El modelo se encarga de la abstracción de la lógica referida a los datos, lo que permite que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos que la aplicación utiliza. El controlador se encarga de la transferencia de información entre la vista y el modelo. Esto se aprecia cuando se envía desde la vista la configuración del reporte al controlador, este lo captura y se lo manda al modelo para que lo almacene [40].

En el sistema para la búsqueda de información el patrón Modelo Vista Controlador (MVC) se aplica de la siguiente manera:

1. El cliente se conecta con el sistema y este consulta la tabla de enrutamiento, que es donde se encuentran todas las rutas de los controladores que tiene el sistema.
2. A través de esta ruta se conecta con el controlador.

3. El controlador va al modelo y obtiene los datos, este a su vez se conecta con la base de datos para obtener la información.
4. Esta información es mostrada al usuario mediante la vista.

A pesar de que se pueden llegar a desarrollar aplicaciones web muy diversas con Symfony2, el funcionamiento interno es muy similar: 1) el **Controlador** gestiona las peticiones realizadas, 2) el **Modelo** busca la información que se le pide, 3) la **Vista** crea páginas con plantillas y datos [39].

A continuación se muestra un esquema de ejemplo que representa la explicación de cómo funciona el patrón Modelo Vista Controlador (MVC) para el CU Autenticar usuario.

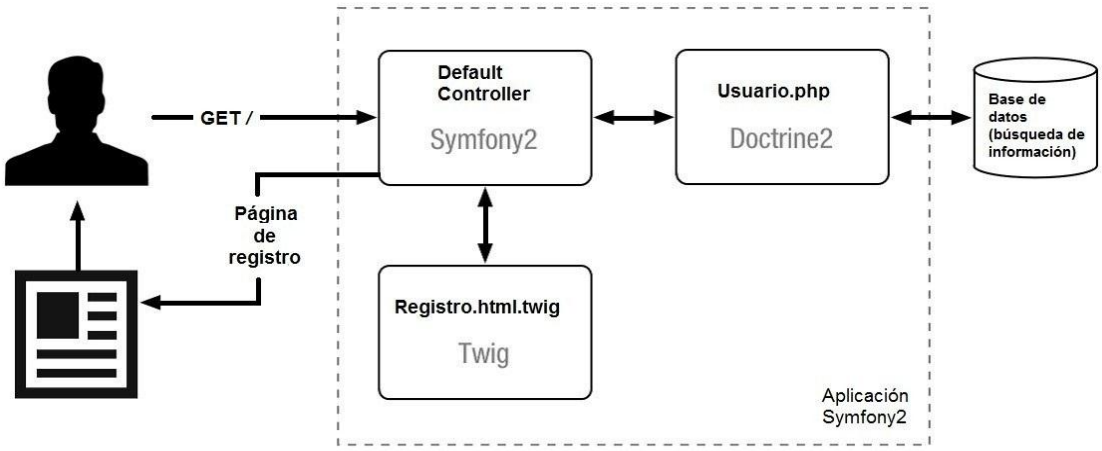

Figura 8. Esquema que representa el CU Autenticar usuario con el patrón MVC

***DefaultController.php*:** representa la información con la que trabaja la aplicación, o sea, su lógica de negocio. Se encarga de hacer las peticiones a la base de datos mediante el ORM[6] de Doctrine para enviárselos a la vista twig.

Usuario.php: convierte el modelo en una página web que facilita al usuario interactuar con ella. Contiene los elementos asociados a la presentación de los datos y la información visual de los elementos que conformarán el reporte, encontrándose el código asociado a las interfaces, que se encargan de visualizar la información que el controlador devuelve. Mapea cada uno de los registros de la base de datos.

Registro.html.twig: es el encargado de procesar las interacciones del usuario y ejecuta los cambios adecuados en el modelo o en la vista. El controlador es el encargado de aislar al modelo y a la vista de los

[6] *Object Relational Mapping* (ORM, por sus siglas en inglés). Mapeador de objetos relacional. Doctrine es un ORM escrito en PHP.

detalles del protocolo usado para las peticiones (HTTP, consola de comandos, correo electrónico). Con los datos que le son enviados por el *default controller* construye la página de registro.

2.8. Patrones de diseño

Los patrones de diseño representan la descripción de un problema particular y recurrente que aparece en contextos específicos, y presenta un esquema genérico demostrado con éxito para su solución; este último se especifica mediante la descripción de los componentes que la constituyen, sus responsabilidades y desarrollos, así como también la forma como estos colaboran entre sí [41].

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de *software* y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

Patrones de diseño GoF

Los patrones GoF (*Gang of Four*, en español banda de los cuatros), describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad. Permiten crear grupos de objetos para ayudar a realizar tareas complejas. Algunos de los patrones GoF empleados en el desarrollo del sistema se describen a continuación.

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. En el sistema todas las clases controladoras son instancias únicas de la clase *Controller*.

Mediador (*Mediator*): define un objeto que coordina la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto. Se aplica en las bibliotecas que funcionan como mediadoras entre las clases controladoras y las modelos o de acceso a datos.

Observador (*Observer*): define una dependencia de uno a muchos, entre objetos, de forma que cuando un objeto cambie de estado se notifiquen y actualicen automáticamente todos los objetos que dependen de él. Se evidencia en la clase *loader* que es el objeto *load* de las clases controladoras. Se encarga de

cargar los elementos del marco de trabajo, como por ejemplo bibliotecas y modelos. Es el encargado de actualizar la controladora instanciada.

Patrones de diseño GRASP

Los patrones GRASP (*General Responsibility Assignment Software Patterns*, en español patrones generales de *software* de asignación de responsabilidades), los cuales dan la medida de un refinamiento del diseño, describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones [42].

Para el diseño de la solución propuesta se tuvieron en cuenta los siguientes patrones: experto, creador, bajo acoplamiento, alta cohesión y controlador. Es importante y vale la pena dominar estos cinco patrones, porque se refieren a cuestiones muy básicas, comunes y a aspectos fundamentales del diseño [42].

Experto: este patrón plantea que se debe asignar una responsabilidad al experto en información, en otras palabras, a la clase que cuenta con la información necesaria para cumplir su responsabilidad. De esta manera se conserva el encapsulamiento de la información, pues los objetos ejecutan las tareas que le corresponden de acuerdo a la información que conservan, lo que proporciona que los sistemas sean más resistentes y fáciles de mantener.

Creador: guía la asignación de responsabilidades de la creación de objetos. Su propósito fundamental es encontrar un creador que deba conectar con el objeto producido en cualquier evento. En la clase *loader* que es el objeto *load* de las clases controladoras, se encarga de cargar los elementos del marco de trabajo como bibliotecas, modelos, etc.

Bajo acoplamiento: es el encargado de asignar una responsabilidad, de manera tal que no se creen gran cantidad de dependencias de una clase, con respecto al resto de las clases; de forma tal que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases.

Relacionado con el bajo acoplamiento se hace uso también del patrón alta cohesión. Este plantea que la información que almacena una clase debe ser coherente y estar relacionada con la clase.

Alta cohesión: cada elemento debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Se utilizó para asignar responsabilidades de manera tal que una clase no tenga sobrecarga de funcionalidades y estas estén relacionadas entre sí.

Controlador: se evidencia en el uso de las clases controladoras, que se implementan con la aplicación del patrón de arquitectura que establece el marco de trabajo. Es el intermediario en la capa vista y biblioteca, obteniendo los datos y enviándoselos a estas, de forma tal que se garantice la comunicación entre los eventos externos del sistema en la capa de presentación y los componentes de la capa de negocio. Asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.

2.9. Modelo de diseño

El modelo de diseño es el modelo encargado de describir la ejecución de los casos de uso del sistema, y se utiliza como medio de abstracción del modelo de implementación y el código fuente del *software*. Su principal objetivo es transmitir, a través de la representación mediante diagramas, el conocimiento en profundidad de los aspectos que tienen relación con los requerimientos no funcionales y restricciones referentes a los lenguajes de programación.

Diagrama de clases del diseño con estereotipos web

Los diagramas de clases del diseño constituyen diagramas de estructura estática que presentan las clases del sistema y las relaciones entre ellas; expresan lo que el sistema puede hacer y cómo puede ser construido. Es muy significativo al constituir estos artefactos los que se necesitan modelar para que el desarrollador los implemente y obtener así el producto final con la calidad requerida [36].

A continuación se muestra el diagrama de clases del diseño correspondiente al CU descrito anteriormente. En el diagrama se muestran las relaciones entre las clases que participan, teniendo como característica exclusiva sobre los diagramas de clases tradicionales, el uso de los estereotipos UML especiales que son específicos para los diagramas de clases web, los cuales permiten modelar de forma segura la arquitectura y funcionamiento de aplicaciones de este tipo. El resto de los diagramas de clases del diseño se encuentran en el **Anexo # 3.**

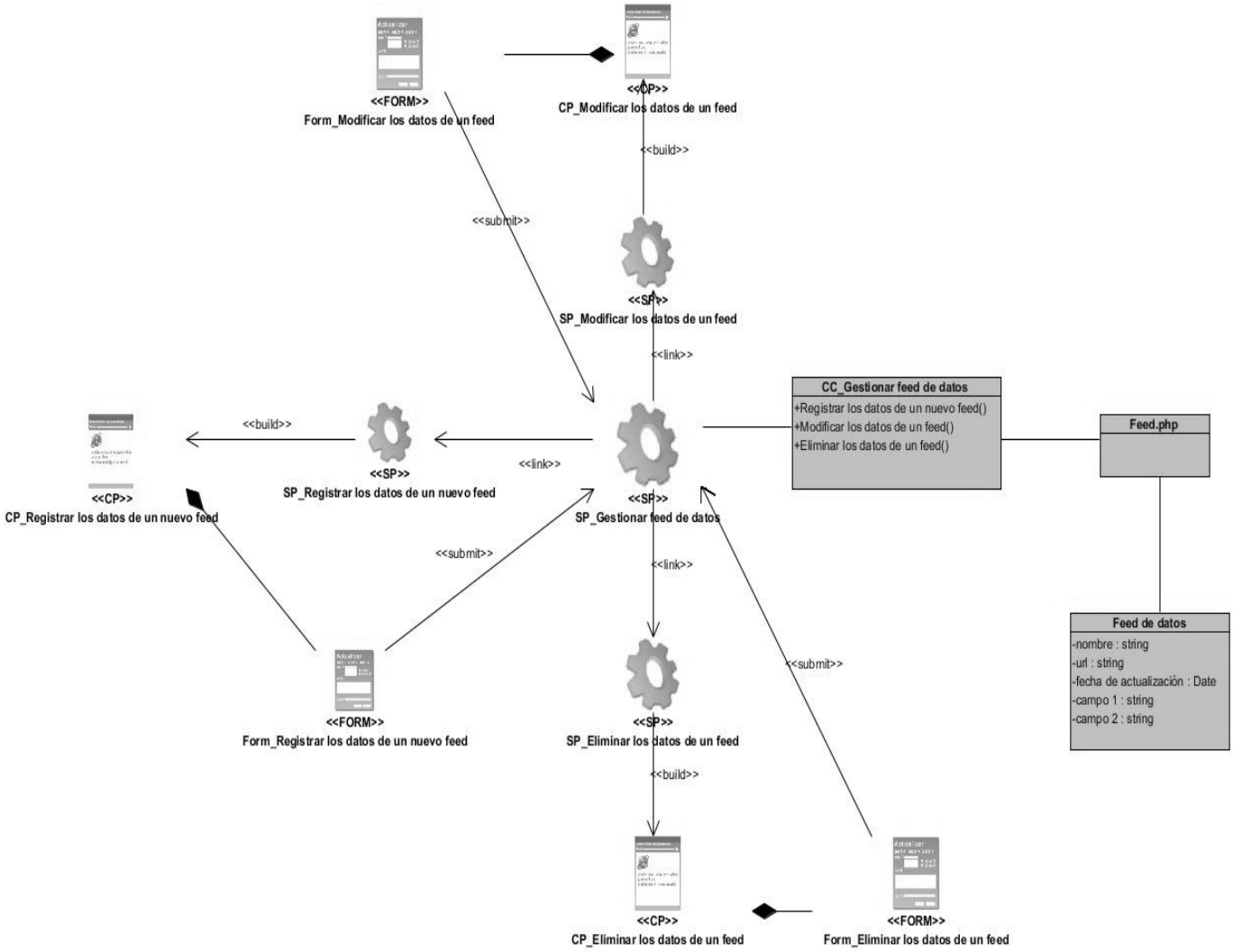

Figura 9. Diagrama de clases del diseño con estereotipos web del CU Gestionar *feed* de datos

2.10. Modelo de despliegue

El modelo de despliegue (ver Figura 10) proporciona un modelo detallado de la forma en la que los componentes se extenderán a lo largo de la infraestructura del sistema. Define las capacidades de red, las especificaciones del servidor, los requisitos de *hardware* y otra información relacionada al despliegue del sistema propuesto.

Un diagrama de despliegue muestra cómo se configuran las instancias de los componentes y los procesos para la ejecución *run-time* en las instancias de los nodos de proceso [42].

Sobre el modelo de despliegue deben hacerse las siguientes observaciones:

- Los nodos poseen relaciones de comunicación entre ellos, tales como TCP/IP, HTTP/HTTPS.
- Cada nodo representa un recurso de cómputo, puede ser un procesador o una pc cliente.

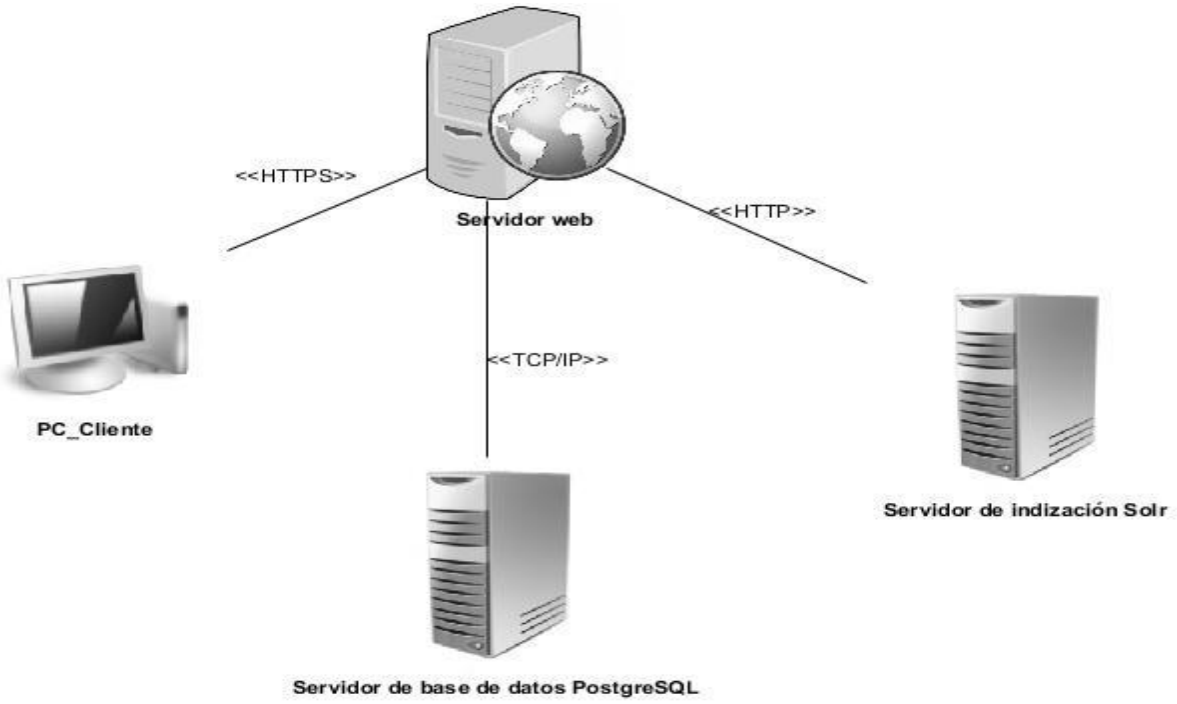

Figura 10. Diagrama de despliegue del sistema

A continuación se describen cada uno de los nodos presentes en el diagrama de despliegue de la figura anterior y la comunicación entre ellos.

PC Cliente: computadora que cuenta con un navegador actualizado y que sigue los estándares web (Mozilla Firefox versión 30.0 o superior). Se recomiendan estaciones de trabajo con sistema operativo GNU/Linux.

Servidor web: representa una estación donde estará montado el servidor web sobre el cual correrá el sistema para la búsqueda de información.

Servidor de base de datos PostgreSQL: representa el servidor donde estará el SGBD PostgresSQL que dará respuesta a las peticiones hechas por la aplicación.

Servidor de indización Solr: representa el servidor de indización que permitirá manipular de manera sencilla los índices distribuidos.

Capítulo 2. Concepción del sistema para la búsqueda de información

2.11. Conclusiones parciales

Con la culminación del presente capítulo se dio cumplimiento a cada uno de los objetivos trazados, destacándose de manera general los siguientes aspectos:

- A través de la obtención de los requisitos funcionales se pudo identificar lo que debe hacer el sistema, lográndose un mayor entendimiento de su funcionamiento.
- Los requisitos no funcionales permitieron definir los atributos que debe exhibir el sistema.
- La concepción de la propuesta de solución y los artefactos generados ayudan al programador a entender las funcionalidades que desea el cliente y por tanto se implementa un sistema conforme a las necesidades del mismo.

Todos estos aspectos relacionados con el análisis y diseño del producto, han creado las condiciones para efectuar la implementación del sistema para la búsqueda de información.

Capítulo 3. Implementación y prueba del sistema

Introducción

Una vez concluido el proceso de análisis y diseño del sistema para la búsqueda de información están creadas todas las condiciones para generar el código fuente, obtener una solución funcional y validar su funcionamiento. En el presente capítulo se desarrolla la solución propuesta siguiendo un estándar de codificación guiado por las especificaciones de requisitos descritas en el capítulo anterior, se identifican las técnicas utilizadas para la validación de los requisitos y por último se define la estrategia de pruebas de *software* que se realizaron para garantizar el correcto funcionamiento del sistema.

3.1. Técnicas de programación

Las técnicas de programación constituyen una parte esencial en el proceso de desarrollo e ingeniería del *software* dentro del ámbito informático. Cada técnica tiene sus propias peculiaridades y distintos métodos de resolución de problemas, así como la implementación de estándares de ciertas compañías o instituciones.

Durante el desarrollo del sistema se utilizó como técnica de programación, la Programación Orientada a Objeto (POO).

3.2. Programación Orientada a Objeto

La POO es una técnica que se convierte en un paradigma de programación cuyo elemento fundamental son los objetos y sus interacciones para diseñar y desarrollar aplicaciones informáticas. Existen varios lenguajes que soportan la orientación a objetos, estos fomentan la reutilización y extensión del código. Además, permiten crear sistemas más complejos y agiliza el desarrollo del *software*, facilitando el trabajo en equipo y el mantenimiento del mismo. La POO incluye los conceptos de herencia, modularidad, polimorfismo y encapsulamiento. Esta técnica de programación es una forma especial de programar, muy cercana a como se expresarían las cosas en la vida real, puesto que está basada en el modo de pensar del hombre y en el modo de operar de la computadora [43].

3.3. Modelo de componentes que integran la solución informática

El modelo de componentes representa la forma en que es estructurado un sistema informático atendiendo a las diferentes partes que lo componen. Partiendo de este punto, Sommerville puntualiza que cada componente debe ser tratado como una unidad de composición independiente e indispensable dentro de

un sistema, y que puede contraer relaciones de dependencia con otros componentes. Algunos ejemplos de componentes físicos lo constituyen los archivos, módulos, bibliotecas, ejecutables, binarios, entre otros [31].

Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos *software* que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes o bibliotecas cargadas dinámicamente [36]**.** A continuación se presenta el diagrama de componentes para el sistema que se propone (ver Figura 11).

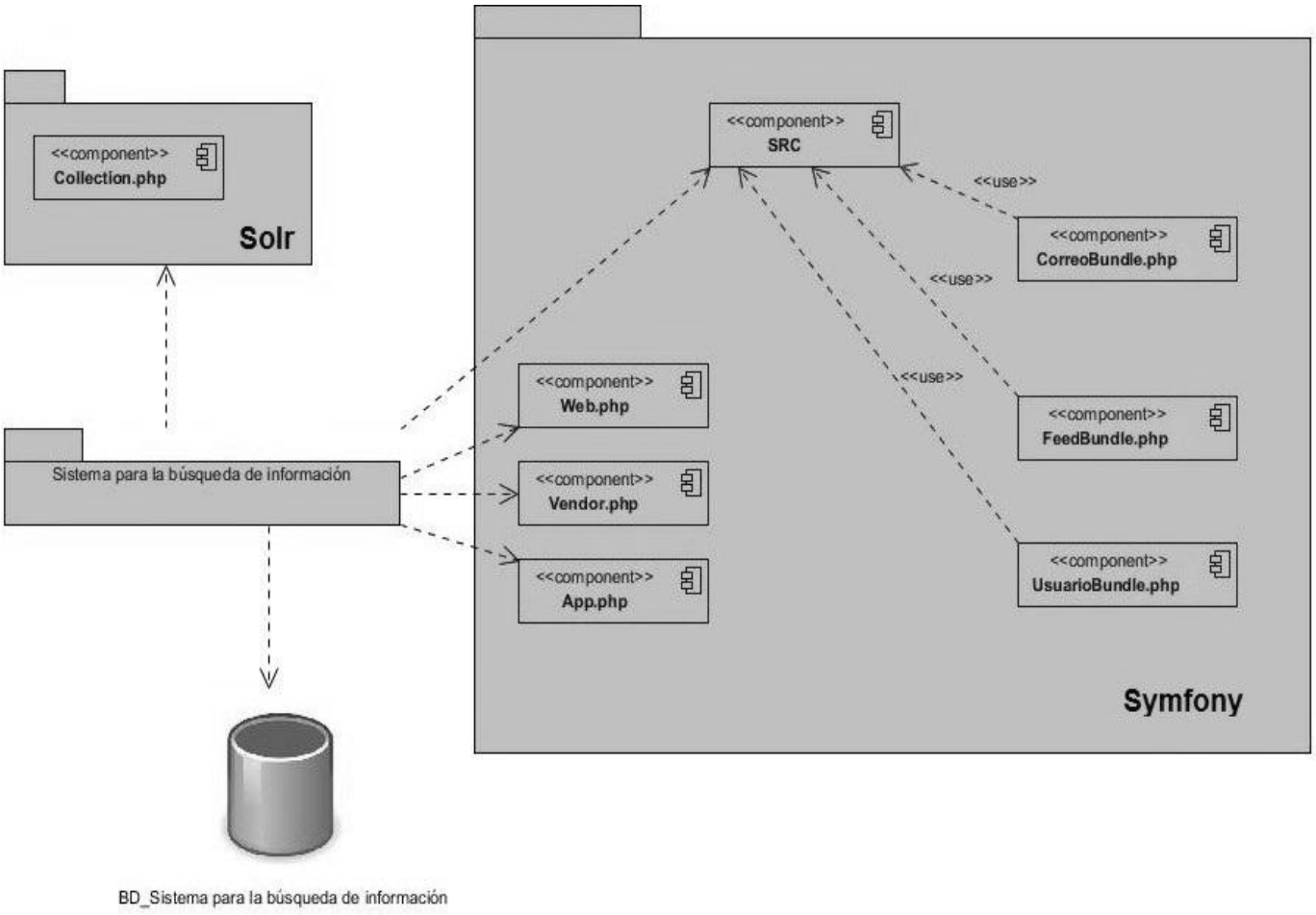

Figura 11. Diagrama de componentes del sistema

CorreoBundle.php: se encarga de enviar correos a los usuarios para validar su cuenta.
FeedBundle.php: se encarga de la gestión de los *feed* de datos.
UsuarioBundle.php: es el paquete encargado de la gestión de los usuarios.
Vendor.php: contiene las bibliotecas y componentes de Symfony, y las bibliotecas de terceros.
App.php: contiene las configuraciones del *framework* de aplicaciones web Symfony.
Collection.php: colección en Solr donde se almacenan los datos de búsqueda.
BD_Sistema de búsqueda de información: contiene todos los datos del sistema para la búsqueda de información.

3.4. Estándar de codificación

Los estándares de codificación son especificaciones o estilos que establecen la forma de generar el código funcional de las aplicaciones informáticas. El uso de técnicas de codificación sólidas y la realización de buenas prácticas de programación, con vistas a generar un código de alta calidad, es de gran importancia para la calidad del *software*, y para obtener un buen rendimiento. Además, si se aplica un estándar de codificación bien definido existen muchas posibilidades de que un proyecto de *software* se convierta en un sistema fácil de comprender y de mantener.
Para facilitar el entendimiento del código y precisar un modelo a seguir, se adoptaron determinados estándares de codificación que a continuación se describen, agrupados por los aspectos en los que fueron utilizados.

Identificadores

Para la definición del nombre de las clases, funciones y variables se tuvieron en cuenta los estilos de escritura *lowerCamelCase* y *UpperCamelCase*. El estilo *lowerCamelCase* establece que la separación entre palabras internas de los identificadores deberá realizarse escribiendo la letra inicial en mayúscula, a excepción de la primera palabra. Además no deberá colocarse ningún carácter especial entre palabras de los identificadores. A excepción del estilo anterior, en el estilo *UpperCamelCase,* la primera letra de cada una de las palabras es mayúscula.
Ejemplos de uso:

- Clase: **class** Feed
- Función: **public function** portadaAction ()
- Variable: $id;

Indentación, llaves de apertura y cierre y tamaño de líneas: usar una indentación sin tabulaciones, con un equivalente a cuatro espacios, para mantener integridad en las revisiones. El uso de las llaves "{}" será en la misma línea y seguido del nombre del método. La longitud de las líneas de código es aproximadamente de 75-80 caracteres, para mantener así la legibilidad del código.

Ejemplo:

```
public function portadaAction() {
    return $this->render('FeedBundle:Default:portada.html.twig');
}
```

Figura 12. Ejemplo de indentación, llaves de apertura y cierre, y tamaño de las líneas

Convención de nomenclatura

Variables: se rigen por la nomenclatura *lowerCamelCase.* Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula.

Ejemplo:

```
class Feed {

    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;
```

Figura 13. Ejemplo de convención de nomenclaturas en variables

Clases: siempre comienzan con mayúscula. Se rigen por el estilo *UpperCamelCase*.

Ejemplo:

```
class Usuario implements UserInterface {

    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     *
     */
```

Figura 14. Ejemplo de convención de nomenclaturas en clases

Funciones: se rigen por la nomenclatura *lowerCamelCase*. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. Los parámetros son separados por espacio luego de la coma que los separa.

Ejemplo:

```
public function eliminarUsuariosAction($usuario) {
    $usuariologueado = $this->get('security.context')->getToken()->getUser();
    $em = $this->getDoctrine()->getManager();
    $usuarios = $em->getRepository('UsuarioBundle:Usuario')->find($usuario);
    if ($usuariologueado == $usuarios) {
        $this->get('session')->getFlashBag()->add('danger', 'Usted está autenticado, no puede eliminarse');
        return $this->render('FeedBundle:Default:portada.html.twig');
    }
    return $this->render('UsuarioBundle:Default:eliminar.html.twig', array('usuario' => $usuario));
}
```

Figura 15. Ejemplo de convención de nomenclaturas en funciones

Ficheros: todo siempre en mayúscula y en caso de nombres compuestos se usa la mayúscula para el segundo nombre.

- **Para las vistas:** nombres intuitivos y relacionados con el formulario o vista que representa.
- **Para los modelos:** con el mismo nombre de la clase que representa.
- **Para las controladoras:** con el nombre que tiene por defecto el *framework* de desarrollo (*DefaultController*).

Estructuras de control: se incluye *if, for, foreach, while, switch.* Entre las estructuras de control y los paréntesis existe un espacio. Se recomienda utilizar siempre llaves de apertura y cierre, incluso en situaciones en las que técnicamente son opcionales. Esto aumenta la legibilidad y disminuye la probabilidad de errores lógicos.

Si las condiciones son tan largas que sobrepasan el tamaño de las líneas, estas se dividen en varias líneas. En el mejor de los casos cuando la condición es muy extensa, esta se puede dividir en variables y compararlas dentro de la estructuras de control.

Documentación: todos los archivos tienen la documentación asociada al mismo.

Buenas prácticas: para facilitar la legibilidad del código se usa un salto de línea antes de las estructuras de control y definición de las funciones.

Ejemplos:

```
public function modificarAction() {
    $usuario = $this->get('security.context')->getToken()->getUser();
    $formulario = $this->createForm(new UsuarioType(), $usuario);
    $peticion = $this->getRequest();
    $passwordOriginal = $formulario->getData()->getPassword();
    $formulario->handleRequest($peticion);
    if ($formulario->isValid()) {
        if (null == $usuario->getPassword()) {
            $usuario->setPassword($passwordOriginal);
        } else {
            $encoder = $this->get('security.encoder_factory')
                    ->getEncoder($usuario);
            $usuario->setSalt(md5(time()));
            $passwordCodificado = $encoder->encodePassword(
                    $usuario->getPassword(), $usuario->getSalt()
            );
            $usuario->setPassword($passwordCodificado);
        }
```

Figura 16. Ejemplo de estructuras de control

3.5. Estrategias de prueba

Las pruebas de *software* son un elemento crítico para la garantía de calidad del producto y representan una revisión final de las especificaciones, del diseño y de la codificación [30].

Las pruebas de *software* son un conjunto de actividades que se pueden planificar por adelantado y llevar a cabo sistemáticamente. Por esta razón, se debe definir en el proceso de la ingeniería del *software* una plantilla para las pruebas del *software*: un conjunto de pasos en los que se puede situar los métodos

específicos de diseño de casos de prueba. Las pruebas constituyen el último bastión desde el que se puede evaluar la calidad y de forma más pragmática descubrir los errores. La prueba del *software* es un elemento de un tema más amplio que, a menudo, es conocido como verificación y validación [30].

Para probar el funcionamiento del sistema, se distinguen como niveles de pruebas: de unidad, de integración y de sistema. En el nivel de sistema se utiliza como tipo de pruebas las de rendimiento, estrés y las de funcionalidad, aplicando el método de caja negra con la técnica de particiones equivalentes.

Nivel de unidad

La prueba de unidad centra el proceso de verificación en la menor unidad del diseño de *software*: el componente, *software* o módulo. La prueba se centra en cada módulo individualmente asegurando que funcionen adecuadamente como una unidad [30].

Nivel de integración

La prueba de integración se centra en probar los componentes del sistema combinado. Se centra en descubrir errores de las especificaciones de las clases [30].

Para ambos niveles se aplicarán pruebas de caja negra guiadas por la técnica de prueba partición de equivalencia.

- Técnica de la partición de equivalencia.

La partición equivalente es una técnica de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores, que de otro modo requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número de casos de prueba que hay que desarrollar. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Las condiciones de entrada son un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica.

Las clases de equivalencia se pueden definir a través de las siguientes condiciones [30].

- Si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos inválidas.

- Si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos inválidas.
- Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una inválida.
- Si una condición de entrada es lógica, se define una clase válida y una inválida.

La ejecución de las pruebas de integración posibilitó verificar el trabajo conjunto de los componentes del sistema en cuestión, junto al servidor de indización Solr. Se hizo énfasis en la interacción entre estos componentes, lo que posibilitó la detección de incoherencias en el funcionamiento de la aplicación. Estas incoherencias fueron corregidas, logrando así una correcta integración de los componentes internos del sistema entre sí, con el servidor de indización Solr y la base de datos en PostgreSQL.

La prueba de integración aplicada, es la definida por el Centro de Calidad para Soluciones Informáticas (CALISOFT). Este propone un diseño de casos de pruebas específicos para este nivel de prueba. En estos casos de pruebas, se especifican los datos del sistema al cual se debe integrar. Cuando el sistema desarrollado funciona correctamente con estos datos, se asume que las pruebas de integración fueron satisfactorias. A continuación se muestran los diseños de casos de pruebas realizados.

Tabla 4: Prueba de integración Int_1 módulo CorreoBundle

Caso de prueba: Prueba de integración al módulo CorreoBundle.
Módulo a integrar: módulo CorreoBundle.
Número de caso de prueba: Int_1 **Condiciones de ejecución:** se realiza la petición de los datos del usuario.
Descripción de la prueba: comprobar que el módulo CorreoBundle envía correos cuando un usuario se registra en la aplicación.
Entradas/Pasos de ejecución: el controlador del módulo CorreoBundle obtiene los datos de la persona que se ha registrado (correo electrónico).
Resultado esperado: el módulo CorreoBundle envía correos de registros al usuario registrado en el sistema.

Evaluación: prueba satisfactoria.

Tabla 5. Prueba de integración Int_2 módulo FeedBundle

Caso de prueba: Prueba de integración al módulo FeedBundle.
Módulo a integrar: módulo FeedBundle.
Número de caso de prueba: Int_2 **Condiciones de ejecución:** el usuario debe estar autenticado en el sistema.
Descripción de la prueba: comprobar que el módulo FeedBundle es capaz de registrar un nuevo *feed* en el sistema.
Entradas/Pasos de ejecución: el controlador del módulo FeedBundle realiza una petición al usuario mediante un formulario pidiendo los datos del *feed* a registrar.
Resultado esperado: el módulo FeedBundle registra el nuevo *feed* en la base de datos.
Evaluación: prueba satisfactoria.

Tabla 6. Prueba de integración Int_3 módulo UsuarioBundle

Caso de prueba: Prueba de integración al módulo UsuarioBundle.
Módulo a integrar: módulo UsuarioBundle.
Número de caso de prueba: Int_3 **Condiciones de ejecución:** se realiza la petición de los datos del usuario.
Descripción de la prueba: comprobar que el módulo UsuarioBundle es capaz de registrar un nuevo usuario en el sistema.
Entradas/Pasos de ejecución: el controlador del módulo UsuarioBundle realiza una petición al usuario mediante un formulario pidiendo sus datos para ser registrado en el sistema.
Resultado esperado: el módulo UsuarioBundle registra el nuevo usuario en la base de datos.

Evaluación: prueba satisfactoria.

Tabla 7. Prueba de integración Int_4 módulo knppaginatorBundle

Caso de prueba: Prueba de integración al módulo knppaginatorBundle.
Módulo a integrar: módulo knppaginatorBundle.
Número de caso de prueba: Int_4 **Condiciones de ejecución:** se realiza la petición de consultar a Solr los datos de un *feed*.
Descripción de la prueba: comprobar que el módulo knppaginatorBundle realiza la paginación debajo de las tablas de los datos de un *feed*.
Entradas/Pasos de ejecución: el controlador del módulo knppaginatorBundle se encarga de realizar la consulta interna a Solr de lo que se desea paginar, con solo de pasarle los elementos a mostrar y la consulta a ejecutar.
Resultado esperado: el módulo knppaginatorBundle pagina los datos de un *feed*.
Evaluación: prueba satisfactoria.

Nivel de sistema

La prueba de sistema se centra en probar el *software* funcionando como un todo. Es aceptable para cuando el *software* se encuentra en la fase de construcción. Trata de probar que los objetivos para los que fue construida la aplicación no se cumplen en su totalidad y que por tanto hay que cambiar cosas en la aplicación. Se usan como base los objetivos originales. No existe un método específico, sino se dan lineamientos a la hora de preparar los casos de prueba. Se finaliza cuando se cumplieron los meses o las semanas programadas y se han hallado N errores. En las pruebas de sistema una de las estrategias muy usada es las pruebas de caja negra [30].

Tipos de pruebas de sistemas:

- Pruebas de rendimiento.
- Pruebas de funcionalidad.

- Pruebas de estrés.

Pruebas de caja negra

Plantea Pressman que la prueba de caja negra intenta encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación [30].

Las pruebas de caja negra se desarrollan sobre la interfaz del *software*. Estas pruebas se encargan de verificar que las funciones que debe desempeñar el sistema son operativas. Se centran en los requisitos funcionales del *software*, sin interesarse por el funcionamiento interno del mismo. Para el desarrollo de las pruebas del sistema se utiliza el método de pruebas de caja negra y se eligió la técnica de partición equivalente que se explicará más adelante.

Particiones de equivalencias

- Identificar las clases de equivalencia válidas e inválidas para cada condición de entrada.
- Escribir un nuevo caso de prueba para cubrir tantas clases de equivalencia válidas como sea posible.
- Escribir un nuevo caso de prueba que cubra una y solo una clase de equivalencia inválida.

Especificaciones de clases válidas e inválidas

Para un rango de tres clases de equivalencia: por debajo, en el rango y por encima del rango.

Para un valor concreto de tres clases de equivalencia: por debajo del valor, el valor en concreto y por encima del valor.

Para un valor dentro de un conjunto de dos clases de equivalencia: en el conjunto o fuera de él.

Para un booleano de dos clases equivalentes: verdadero o falso.

Pruebas de rendimiento

Para las pruebas de rendimiento, específicamente pruebas de carga y estrés, que se le realizaron al sistema para la búsqueda de información, se utilizó la herramienta JMeter en su versión 2.13.

JMeter es un *software* de código abierto, diseñado para pruebas de carga de comportamiento funcionales y la medición del rendimiento. Originalmente fue diseñado para probar las aplicaciones web, pero desde entonces se ha expandido a otras funciones de prueba. Es utilizado para probar el rendimiento tanto en los recursos estáticos como dinámicos. Puede ser utilizado para simular una carga pesada en un servidor, grupos de servidores, la red o un objeto para poner a prueba su resistencia o para analizar el rendimiento general en diferentes tipos de carga. Puede usarse además para hacer un análisis gráfico de rendimiento o para probar el comportamiento del servidor, *script* u objeto con cargas pesadas concurrentes [44].

Para la realización de las pruebas se hizo necesario tener en cuenta las condiciones del escenario, tanto del *hardware* como del *software*, donde se encuentra el sistema para obtener una correcta información de comportamiento y resultados en general. Por tanto se hizo necesario simular las pruebas en un escenario con las características siguientes:

***Hardware*:**

- Tipo de procesador: Intel o similar, con una velocidad a 3.30 GHz o superior.
- Memoria: 2 GB de RAM y 160 GB HDD o superior.
- Tipo de red: Ethernet 10/100Mbps.

***Software*:**

- Tipo de servidor web: Apache
- Tipo de servidor de indización: Solr 4.10.3.
- Máximo de hilos concurrentes (simulación de usuarios): 50, 250.
- Plataforma: una distribución GNU/Linux como sistema operativo.
- Servidor de BD: PostgreSQL 9.2.

Resultados de las pruebas de rendimiento

Después de la simulación realizada de 50 y 250 usuarios concurrentes, se obtiene 0 % de error de las peticiones realizadas al sistema con un rendimiento de 105,6/sec y 17,0/sec, por lo tanto la prueba resulta satisfactoria, (ver Figuras 17 y 18).

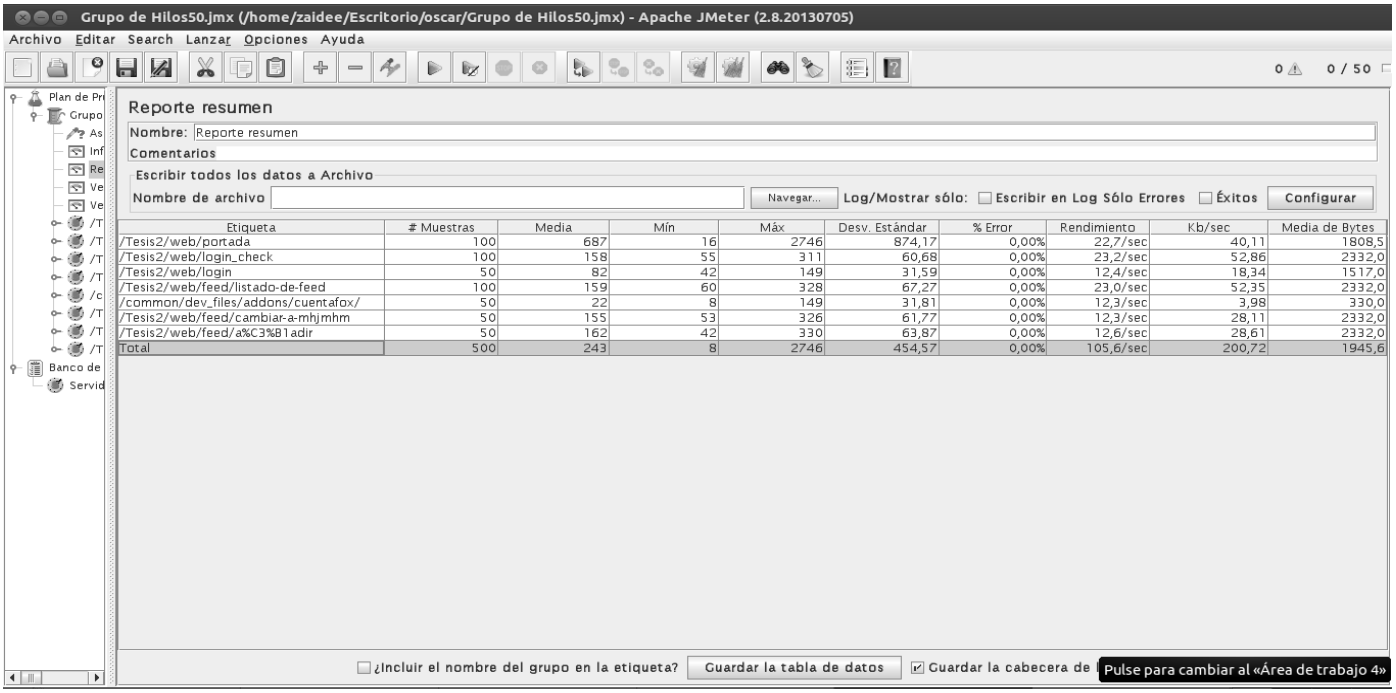

Figura 17. Resultados obtenidos de la herramienta JMeter para 50 usuarios

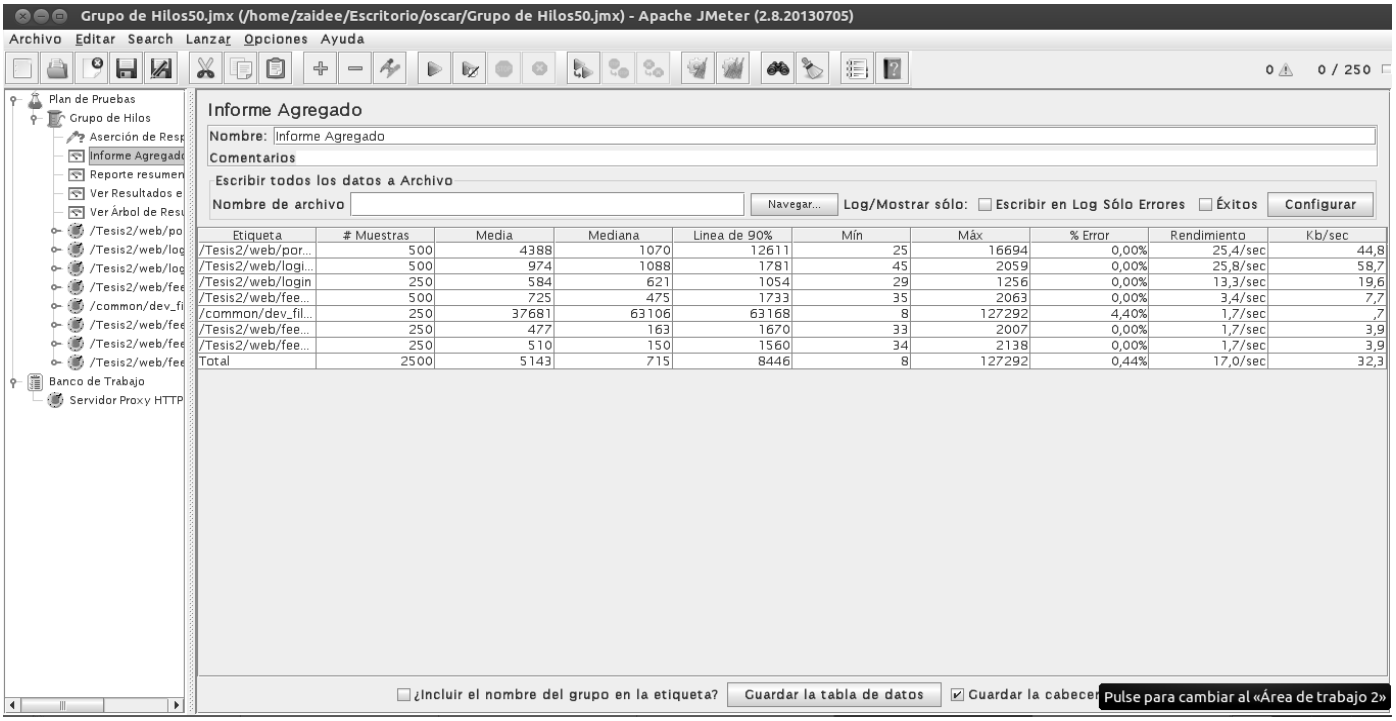

Figura 18. Resultados obtenidos de la herramienta JMeter para 250 usuarios

Diseños de casos de prueba

Un caso de prueba es una especificación, usualmente formal, de un conjunto de entradas de prueba, condiciones de ejecución y resultados esperados, identificados con el propósito de hacer una evaluación de aspectos particulares de un elemento objeto de prueba [37].

A continuación se describe el caso de prueba realizado al requisito funcional: autenticar usuario. Para consultar otros casos de pruebas diseñados para validar el funcionamiento del sistema para la búsqueda de información, ver **Anexo # 4.**

Tabla 8. Descripción del caso de prueba para el RF Autenticar usuario

Autenticar usuario					
Escenario	**Descripción**	**Dirección de correo**	**Contraseña**	**Respuesta del sistema**	**Flujo central**
SC 1.1 El sistema autentica al usuario de forma correcta.	El usuario se autentica por vez primera al sistema.	V	V	El sistema autentica al usuario y le permite el acceso a las funcionalidades del sistema, según su rol.	1- El usuario teclea en el navegador la url del sitio web. 2- El sistema le muestra un formulario para su autenticación. 3- El usuario introduce la información y selecciona la opción Acceder.
		ygiron@estudiantes.uci.cu	yanita		
SC 1.2 El sistema no autentica al usuario.		I	V	El sistema no autentica al usuario y emite los siguientes mensajes de errores: Introduzca una dirección de correo.	1- El usuario teclea en el navegador la url del sitio web. 2- El sistema le
		456kj	osalas		
		V	I		

		osalas@estudiantes.uci.cu	vacío		muestra un formulario para su autenticación. 3- El usuario introduce la información y selecciona la opción Acceder.
		I	I		
		7	adf		
SC 1.3 El sistema no autentica al usuario dejando campos obligatorios vacíos.		I	I	"Lo sentimos. Los campos dirección de correo y contraseña son obligatorios.	1- El usuario teclea en el navegador la url del sitio web. 2- El sistema le muestra un formulario para su autenticación. 3- El usuario introduce la información y selecciona la opción Acceder.
		vacío	vacío		

Tabla 9. Descripción de variables

No	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Dirección de correo	Campo de texto	NO	No se permiten caracteres extraños, excepto el @, y el punto. Ejemplo: osalas@estudiantes.uci.cu.
2	Contraseña	Campo de texto	NO	Longitud mínima 6 caracteres. Permite todos los caracteres.

Pruebas de integración

Las pruebas de integración son definidas para verificar el correcto ensamblaje entre los distintos módulos que conforman un sistema informático. Las mismas validan que estos componentes realmente funcionan juntos, son llamados correctamente y además, transfieren los datos correctos en el tiempo preciso y por las vías de comunicación establecidas [31].

Una vez realizadas las pruebas funcionales a cada componente interno de manera independiente, y verificado que las funcionalidades implementadas se corresponden de acuerdo a los requisitos funcionales y no funcionales establecidos, se pudo comprobar el correcto funcionamiento de los componentes mediante el estudio del flujo de datos entre ellos.

Luego de las pruebas realizadas, se hace necesaria la realización de pruebas de integración, con la finalidad de validar la compatibilidad y el funcionamiento de todas las interfaces que comunican las diferentes partes que conforman el sistema.

Para la realización de las pruebas de integración se llevaron a cabo varias acciones, a continuación se mencionan algunas:

- Comprobación del funcionamiento del enlace entre el servidor de indización Solr y el sistema para la búsqueda de información implementado.
- Verificación de la conexión entre el sistema y la base de datos en PostgreSQL.
- Verificación de la correcta lectura y escritura de los archivos de configuración en formatos JSON y XML, comprobando que los símbolos y caracteres extraños sean codificados y decodificados adecuadamente.

Resultado de las pruebas realizadas

Luego de concluido el proceso de pruebas realizadas al *software*, y analizando los resultados obtenidos, se puede ver cómo la calidad del producto aumenta con respecto a las especificaciones del mismo, demostrando que el sistema se encuentra listo para ser entregado al cliente, o pasar a otra etapa de prueba superior. En este proceso se realizaron cuatro revisiones. La primera revisión arrojó como resultado, que de un total de doce requisitos funcionales, se identificaron cuatro no conformidades, específicamente en errores ortográficos y de implementación en el sistema. En la segunda revisión, las no conformidades encontradas anteriormente habían sido analizadas y corregidas. De 20 requisitos funcionales, se identificaron ocho no conformidades a las cuales se les dio solución. En la tercera revisión

de pruebas, de 26 requisitos funcionales, se obtuvo un total de doce no conformidades las cuales fueron resueltas. Por último se realizó una cuarta revisión, donde no se encontraron no conformidades. Las no conformidades encontradas estuvieron asociadas a errores ortográficos y de implementación en el sistema, errores de validación de campos, funcionalidades no implementadas, errores en el registro de información en la base de datos, y otros. Para ver el gráfico representativo de lo anteriormente explicado, ver Figura 19.

Las pruebas de integración de los módulos correoBundle, feedBundle, usuarioBundle y knppaginatorBundle que integran el sistema para la búsqueda de información, demostraron que la integración es exitosa. Esto permite que la obtención de datos de estos módulos, se realice de forma satisfactoria.

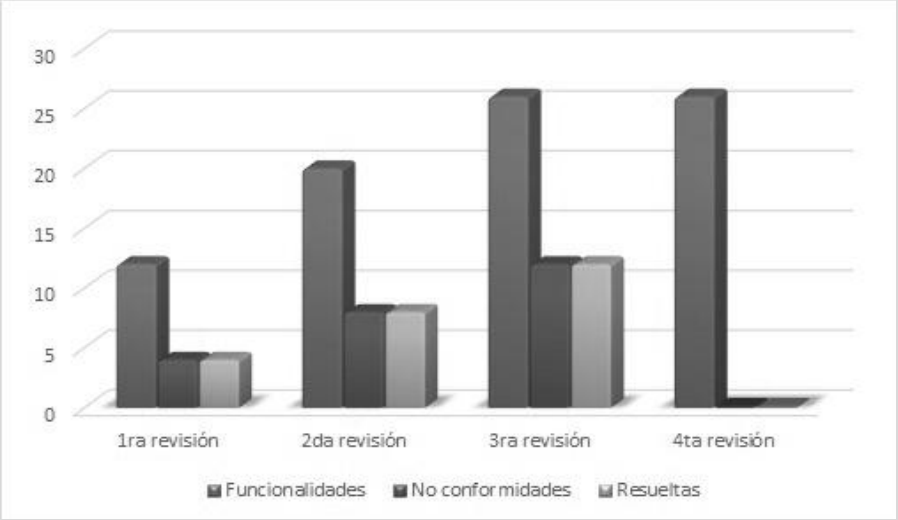

Figura 19. Resultado del proceso de pruebas

3.6. Conclusiones parciales

Con el uso del estándar de codificación, las definiciones para el tratamiento de errores y las técnicas de programación definidas, se logró implementar el sistema que brinda solución a la problemática planteada. La realización de pruebas unitarias, de integración y los casos de prueba de caja negra permitieron corregir errores detectados logrando la correcta implementación de los requisitos funcionales.

Conclusiones generales

El presente trabajo dio cumplimiento a cada uno de los objetivos trazados, pudiéndose destacar de manera general las conclusiones siguientes:

- El análisis de las soluciones homólogas, herramientas y tecnologías utilizadas en el desarrollo de sistemas para realizar búsquedas en sitios web permitió definir el entorno tecnológico a utilizar, así como las posibles funcionalidades a incluir.
- La utilización de la metodología OpenUp para el desarrollo de la solución, permitió generar los artefactos necesarios que facilitaron el entendimiento de la solución para luego implementarla.
- La definición de una estrategia de verificación y validación del sistema permitió elaborar los casos de pruebas asociados, los cuales arrojaron resultados satisfactorios demostrando que la solución es funcional y que puede ser integrada a otros sitios web.
- El desarrollo del sistema, permitió facilitar la búsqueda de la información deseada en los sitios web así como la parametrización de sus funcionalidades, logrando a su vez, el cumplimiento del objetivo general propuesto.

Recomendaciones

Una vez concluido el desarrollo del sistema para la búsqueda de información, cumplidos los objetivos trazados y teniendo en cuenta que es una primera versión del sistema, se recomienda:

- Desarrollar complementos o módulos para los gestores de contenido más populares como WordPress, Joomla, Drupal, etc., permitiendo una mayor facilidad de integración entre los portales web de los usuarios y el sistema desarrollado.
- Desarrollar bibliotecas para distintos lenguajes de programación que permitan realizar búsquedas en el sistema de aquellos sitios web de los clientes desarrollados con diferentes tecnologías.
- Realizar un estudio de las diferentes estrategias de monetización que puedan ser aplicadas al sistema desarrollado.

Referencias bibliográficas

[1] ***Msdn, Microsoft.*** Cómo: Definir catálogos de texto completo, (noviembre 2007). [En línea]. [Citado el: 10 de septiembre de 2014]. Disponible en web: https://msdn.microsoft.com/es-es/library/bb386267%28v=vs.90%29.aspx

[2] **Arenas, Marcelo. Y otros.** *Cómo funciona La Web.* Centro de investigación de la Web. Departamento de Ciencias de la Computación Universidad de Chile. Primera Edición, Pág. 55 (junio 2008) [En línea]. [Citado el: 12 de septiembre de 2014]. Disponible en web: http://sunshine.prod.uci.cu/gridfs/sunshine/books/Como_funciona_la_web.pdf

[3] **Docsetools.** *Índice invertido*, 2015. [En línea]. [Citado el: 13 de septiembre de 2014]. Disponible en web: http://docsetools.com/articulos-noticias-consejos/article_127166.html

[4] **Definición ABC.** *Definición de Indexación*, (2007-2015). [En línea]. [Citado el: 13 de septiembre de 2014]. Disponible en web: http://www.definicionabc.com/general/indexacion.php#ixzz3KdCjOZet

[5] **Chiavenato, Idalberto.** *Introducción a la Teoría General de la Administración.* McGraw-Hill Interamericana Séptima Edición, Pág. 110 (2006). [En línea]. [Citado el: 19 de septiembre de 2014]. Disponible en web: http://www.intercambiosvirtuales.org/libros-manuales/introduccion-a-la-teoria-general-de-la-administracion-7ma-edicion-mcgraw-hill

[6] **Ferrell O. C. y Hirt Geoffrey.** *Introducción a los Negocios en un Mundo Cambiante.* McGraw-Hill Interamericana, Cuarta Edición, Pág. 121 (2004).

[7] **Czinkota, Michael y Kotabe, Masaaki.** *Administración de Mercadotecnia.* International Thomson Editores, Segunda Edición, Pág. 115. (2001).

[8] **Pérez-Carballo, J. and Strzalkowski, T.** *Natural language information retrieval: progress report.* Information Processing and Management 36, 2000. p. 155-178.

[9] **Fonseca Reyna, Yunior César.** *Recuperación de la información: taxonomía de sus modelos*. Revista Cubana de las Ciencias Informáticas, Vol. 6, No 2 (2012). [En línea]. [Citado el: 25 de septiembre de 2014]. Disponible en web: http://rcci.uci.cu/index.php?journal=rcci&page=issue&op=view&path[]=18

REFERENCIAS BIBLIOGRÁFICAS

[10] **Definición de servidor** - Qué es, Significado y Concepto. [En línea]. [Citado el: 11 de octubre de 2014]. Disponible en web: http://definicion.de/servidor/#ixzz3KgaOzlK5

[11] **Servidor web.** [En línea]. [Citado el: 17 de octubre de 2014]. Disponible en web: http://www.ecured.cu/index.php/Servidor_Web

[12] **Definición sitio web.** [En línea]. [Citado el: 21 de octubre de 2014]. Disponible en web: http://www.ecured.cu/index.php/Sitio_Web

[13] **¿Qué es doofinder?** [En línea]. [Citado el: 21 de octubre de 2014]. Disponible en web: http://www.doofinder.com

[14] **Amazon CloudSearch.** [En línea]. [Citado el: 25 de octubre de 2014]. Disponible en web: http://aws.amazon.com/es/cloudsearch/

[15] **Metodologías de desarrollo de *software*.** [En línea]. [Citado el 3 de noviembre de 2014]. Disponible en web: http://www2.rhernando.net/modules/tutorials/doc/ing/met_soft.html

[16] **OpenUP-Ecured.** [En línea]. [Citado el: 7 de noviembre de 2014]. Disponible en web: http://www.ecured.cu/index.php/OpenUp.

[17] **Boehm, B. y Turner, R.;** 2003. “Balancing Agility and Discipline: A Guide for the Perplexed”; Chapter1 (The Two Approaches), Chapter2 (Summary) y Chapter4.

[18] **OMG, Object Management Group.** 2012. UML. [En línea]. [Citado el: 11 de noviembre de 2014]. Disponible en web: http://www.uml.org/

[19] **Herramientas Case - Monografias.com.** [En línea]. [Citado el: 17 de noviembre de 2014]. Disponible en web: http://www.monografias.com/trabajos14/herramicase/herramicase.shtml

[20] **Elejalde Chacón, Reinier.** *Módulo para la creación de modelos de contenidos para Alfresco*. Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas. La Habana, Cuba: Universidad de las Ciencias Informáticas, 2009.

REFERENCIAS BIBLIOGRÁFICAS

[21] **Enciclopedia Universal.** Enciclopedia Universal Académica. *Lenguaje de programación.* [En línea]. [Citado el: 21 de noviembre de 2014]. Disponible en web: http://enciclopedia_universal.esacademic.com/74817/Lenguaje_de_programaci%C3%B3n

[22] **Enciclopedia Universal.** Enciclopedia Universal Académica. *PHP.* [En línea]. [Citado el: 25 de noviembre de 2014]. Disponible en web: http://www.esacademic.com/dic.nsf/eswiki/3167

[23] **Eguiluz, Javier.** *Desarrollo web ágil con Symfony2.* La Habana: s.n., 2012.

[24] **Oracle Corporation and/or its affiliates.** NetBeans. [En línea] 2012. [Citado el: 27 de noviembre de 2014] Disponible en web: http://netbeans.org/community/releases/68/index_es.html

[25] **Guia_ubunto.** ***Guia_ubunto.*** [En línea] 2012. [Citado el: 29 de noviembre de 2014] Disponible en web: http://www.guia-ubuntu.com/?title=PostgreSQL

[26] **PostgreSQL.** [En línea] 2008. [Citado el: 30 de noviembre de 2014] Disponible: http://www.postgresql.org.es/sobre_postgresql

[27] **Solr.** [En línea]. [Citado el: 7 de diciembre de 2014]. Disponible en web: http://lucene.apache.org/solr/

[28] **Apache Solr** [En línea]. [Citado el: 11 de diciembre de 2014]. Disponible en web: http://sunshine.prod.uci.cu/gridfs/sunshine/books/motores_busqueda_empresariales.pdf

[29] **Grainger, Trey y Potter Timothy.** *Solr_in_Action.pdf*. Manning Publications Co. Pág. 22 (2014). [En línea]. [Citado el: 15 de diciembre de 2014]. Disponible en: http://sunshine.prod.uci.cu/search/solr

[30] **Pressman, Roger S.** *Ingeniería del software: un enfoque práctico.* La Habana, Cuba: Editorial Félix Varela, 2005.

[31] **Sommerville, I.** *Ingeniería del software*. 7 ed. Pearson Educación, Madrid: 2005. 21 p. ---. *Software Engineering.* Editado por: Addison-Wesley. 2007.

REFERENCIAS BIBLIOGRÁFICAS

[32] **Rodríguez Figueroa, Inalbis.** *Módulo de Textos para el Sistema de Gestión Documental para la Prensa*. Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas. La Habana, Cuba: Universidad de las Ciencias Informáticas, 2012.

[33] **Pressman, R. S.** *Ingeniería del software.* 7 ed. McGraw-Hill Interamericana de España S.L., 2010. 959 p. ISBN 9786071503145.

[34] **Pressman, R. S.** *Ingeniería del software: un enfoque práctico.* 6 ed. La Habana, Cuba: Editorial Félix Varela, 2005. (Capítulo 7 Epígrafes. 7.5 y 7.6).

[35] **Espinosa, I. Y.** C. M. *Arquitectura de Software Vista de Datos.* Editado por: Universitaria, S. D. G. 2010, 31 p.

[36] **Jacobson, Ivar.** 2000. El Proceso Unificado de Desarrollo de *Software*. [En línea] 2000. [Citado el: 18 de enero de 2015.]

[37] **Pressman, R. S.** *Ingeniería del software: un enfoque práctico.* 5 ed. Parte III. Métodos convencionales para la ingeniería de *software.* Editorial McGraw-Hill Interamericana de España S.L., 2012. 640 p. ISBN 8448132149.

[38] **López, S.** *Cómo mantener el patrón modelo-vista-controlador en una aplicación orientada a la WEB.* 2009. [En línea]. [Citado el: 18 de enero de 2015]. Disponible en: http://biblioteca.uniminuto.edu/ojs/index.php/Inventum/article/view/132/125

[39] **what-is-symfony2.** [En línea]. [Citado el: 28 de enero de 2015]. Disponible en web: http://fabien.potencier.org/article/49/what-is-symfony2

[40] **Patron_Modelo_Vista_Controlador.** [En línea] 2003. [Citado el: 10 de febrero de 2015]. Disponible en web: http://exequielc.wordpress.com/2007/08/20/arquitectura-modelovistacontrolador/

[41] **Larman, Craig.** *UML y Patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado*. 2da Edición. Pearson, 2003. 520 p.

REFERENCIAS BIBLIOGRÁFICAS

[42] **Larman, Craig.** *"UML y Patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado."* 3ra Edición. Pearson, 2005. Capítulo 16, 38. Epígrafes 16.3 – 16.10, 38.2 – 38.4. Pág. 162-177, 503-505.

[43] **Camacho, J.** *Programación orientada a objetos.* (2009). [En línea]. [Citado el: 19 de mayo de 2015.] Disponible en web: http://www.slideshare.net/javiersegura/programacion-orientada-a-objetos-1431389.

[44] **Foundation, the Apache *Software*. 2015.** The Apache *Software* Foundation. [En línea] 2015. [Citado el: 8 de mayo de 2015.] Disponible en web: http://jmeter.apache.org/download_jmeter.cgi

Anexo # 1. Listado de requisitos

Tabla 10. Listado de los requisitos funcionales del sistema

No.	Requisito funcional	Prioridad	Caso de uso del sistema
RF 1	Autenticar usuario.	Alta	Autenticar usuario.
RF 2	Registrar usuario.	Alta	Registrar usuario.
RF 3	Ver perfil de usuario.	Alta	Ver perfil de usuario.
RF 4	Cambiar perfil de usuario.	Alta	Cambiar perfil de usuario.
RF 5	Confirmar usuario.	Alta	Confirmar usuario.
RF 6.1	Mostrar listado de los usuarios registrados.	Alta	Mostar datos.
RF 6.2	Mostrar detalles de un usuario.	Alta	Mostrar datos.
RF 6.3	Registrar los datos de un nuevo usuario.	Alta	Gestionar usuarios con acceso al sistema.
RF 6.4	Modificar los datos de un usuario.	Alta	Gestionar usuarios con acceso al sistema.
RF 6.5	Eliminar los datos de un usuario.	Alta	Gestionar usuarios con acceso al sistema.
RF 7.1	Mostrar listado de los *feed* obtenidos.	Alta	Mostrar datos
RF 7.2	Mostrar detalles de un *feed*.	Alta	Mostrar datos.
RF 7.3	Registrar los datos de un nuevo *feed*.	Alta	Gestionar *feed* de datos.
RF 7.4	Modificar los datos de un *feed*.	Alta	Gestionar *feed* de datos.
RF 7.5	Eliminar los datos de un *feed*.	Alta	Gestionar *feed* de datos.
RF 8	Generar *script* para el uso del servicio que brinda el sistema.	Alta	Generar *script*.
RF 9	Importar *feed* de datos.	Alta	Importar *feed* de datos.
RF 10.1	Mostrar gráfico estadístico de los *feed* de datos filtrado	Alta	Gestionar gráfico estadístico.

	por cantidad de *feed* o rango de tiempo.		
RF 11	Realizar búsqueda sobre un *feed* de datos.	Alta	Mostrar datos.
RF 12.1	Mostrar listado de las propiedades del sistema registradas.	Baja	Mostrar datos.
RF 12.2	Mostrar detalles de una propiedad del sistema.	Baja	Mostrar datos.
RF 12.3	Modificar los datos de una propiedad del sistema.	Baja	Mostrar datos de propiedad del sistema.