Facultad 10

Recolector de contenido web

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

Autores: Anlí Wilfredo Martínez Gallardo
Saimel Sáez Estrada

Tutores: Ing. Dovier Antonio Ripoll Méndez
Lic. Jose Albert Cruz Almaguer

Ciudad de La Habana, Cuba, Junio 2009

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 10 de la Universidad de las Ciencias Informáticas, así como a dicho centro, para que lo usen según estimen pertinente.

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

Anlí Wilfredo Martínez Gallardo

_____
Firma del Autor

Saimel Sáez Estrada

_____
Firma del Autor

Ing. Dovier Antonio Ripoll Méndez

_____
Firma del Tutor

Lic. Jose Albert Cruz Almaguer

_____
Firma del Tutor

DATOS DE CONTACTO

Dovier Antonio Ripoll Méndez, Ingeniero en Ciencias Informáticas. Graduado de la UCI, en el 2008, de Ingeniero en Ciencias Informáticas. Se desempeña en la UCI como: Jefe del Polo Productivo de Soluciones Informáticas para Internet y Profesor de la Asignatura Práctica Profesional en el Departamento de Ingeniería y Gestión de Software. Entre sus principales resultados destacan: miembro del Equipo UCI que participó en el X Maratón Regional Suramericano de Programación ACM-ICPC en Venezuela, donde obtuvieron el mejor lugar histórico de Cuba en tales concursos; autor de publicaciones en la Serie Científica Interna de la UCI y la Revista de Software Libre UXI; fundador/organizador de la Copa Void de Programación y la Competencia Estudiantil por Invitación de la UCI; fundador/desarrollador del Proyecto Filtrado de paquetes por contenido (Filpacon), participando en la implantación de dicho producto tanto en Cuba como en Venezuela; autor de trabajos presentados y/o defendidos en eventos científicos, tales como: Jornada Científica Estudiantil de la UCI, Concurso Nacional de Computación, Exposición de las BTJ, Seminario Iberoamericano de Seguridad en las Tecnologías de la Información, entre otros. Finalmente, ha realizado investigaciones en los siguientes temas: Filtrado de Contenidos de Internet, Minería de Datos, Categorización Automática de Textos, Agentes o Robots Web y Cibermetría. Actualmente puede ser contactado a través de la cuenta de correo electrónico daripoll@uci.cu.

DATOS DE CONTACTO

Jose Albert Cruz Almaguer, Licenciado en Ciencia de la Computación. Graduado del 2004 en la Universidad de Oriente. Asesor del Departamento Docente Central de Técnicas de Programación de la Universidad de las Ciencias Informáticas (UCI). Miembro del Grupo de Investigación de Programación Avanzada y del Grupo de Inteligencia Artificial Aplicada del Polo Productivo de Soluciones Informáticas para Internet. Miembro de la Sociedad Cubana de Matemática y Computación y representa a la UCI en los temas de Lenguajes de Programación. Ha impartido clases de Introducción a la Programación y Programación I desde 2004; e Inteligencia Artificial en el curso 2007-2008. Actualmente puede ser contactado a través de la cuenta de correo electrónico jalbert@uci.cu.

AGRADECIMIENTOS

Agradezco:

A mis padres Wilfredo Martínez Castro y Susana Gallardo Mok, por sus sabios consejos para guiarme por el camino correcto...
A mi hermano Liang y su esposa Liubis, por creer y confiar en mi...
A mi novia Miriela Pérez Peña, por estar incondicionalmente a mi lado brindándome su amor...
A mi sobrina Susanita, por el cariño que me ha demostrado siempre...
A mis hermanos y hermanas del Cerro, por su amor y verdadera amistad...
A mi amigo y compañero de tesis Saimel Sáez Estrada, por motivarme a buscar la excelencia en este trabajo...
A mi amigo Yenner Joaquín Díaz Núñez, por todo el tiempo dedicado y los aportes realizados a este trabajo...
A mis amigos Harold, Carpio, Lisván y Yuliette, por todos los momentos que pasamos juntos, nunca los olvidaré...
A mis compañeros de grupo 10504, por hacer de estos años los mejores de mi vida...
A mi tutor Dovier Antonio Ripoll Méndez, por sus consejos, su ayuda, su sabiduría y por velar en todo momento por la calidad de este trabajo...
A mi tutor Jose Albert Cruz Almaguer, por todo su apoyo y orientaciones oportunas...
A todos mis compañeros de proyecto, por brindarme su amistad...
A los profesores de la UCI Aleida Eva, Abel Meneses y Bertha Elena por la ayuda en la realización de este trabajo...
A los amigos del Centro Nacional de Capacitación del MINVEC Caridad Elena, Lídice y Alejandro por su apoyo en la materialización de este trabajo...

Anlí Wilfredo Martínez Gallardo

AGRADECIMIENTOS

Agradezco:

A mis padres Jesús Napoleón Sáez Solano y Elidia Estrada Martínez, por traerme a la vida...
A mi hermano Sadiel Sáez Estrada, por darme todo su apoyo...
A mis abuelos Carlos, María y Rafaela, a mis tías Elisa y Enely, mis tíos Hermes y Jorge y al resto de mi familia, por lo mucho que me han ayudado...
A mis amigos de la infancia Robertico (El Chino) y Albertico (El Loco), por los buenos y malos momentos que hemos pasado juntos...
A mi novia Liset María Álvarez Barreras, por todo el amor que me ha dado...
A mi profesor de la secundaria Erik Lima, por incentivarme a aprender cada día más...
A mis amigos del IPVCE Yeisel, Ivan, Yohansi, Ansel, Tony, Ariel, Ramón, Cordero, Amauris, Suany, Dailyana, Dailín y Meilys, por compartir todo lo que teníamos y por tantas locuras que hicimos juntos...
A mis amigos de la Universidad Harold, Anlí, Roberto Carlos y Carpio, por sus consejos y por todo que me enseñaron en estos cinco años...
A todos mis compañeros de la brigada 10504, por hacerme reír tanto...
A mis compañeros de proyectos Yandry, Luis, Jose, Kiuver, Oscar, Yordan, Adrián, Siovel, Raúl Amambay, Daileny y Ana, por todo lo que hemos trabajado juntos...
A mi tutor Jose Albert, por su incondicional ayuda...
A mi tutor y amigo de la infancia Dovier, por todo lo que me ha enseñado, principalmente Matemáticas y Perl...
A Maidely Calderón Montero, Aleida Eva Sáez Aldana, Bertha Elena Romero Molina y Abel Meneses Abad, por toda su ayuda para realizar este trabajo...
A todas las personas que de una forma u otra, han hecho posible que escriba estas líneas de agradecimientos...

Saimel Sáez Estrada

DEDICATORIA

Dedico este trabajo a mis padres Wilfredo Martínez Castro y Susana Gallardo Mok, por ser lo más hermoso que tengo en la vida y a mi hermano Liang por servirme de ejemplo con su tenacidad y perseverancia. Todos ustedes viven para darme su cariño y afecto y esta es una pequeña manera de retribuirlos.

Anlí Wilfredo Martínez Gallardo

Dedico este trabajo a mis padres Jesús Napoleón Sáez Solano y Elidia Estrada Martínez y a mi hermano Sadiel Sáez Estrada, por lo mucho que me quieren y por toda la confianza que han depositado en mí, ustedes son mi fuente de inspiración.

Saimel Sáez Estrada

OBSERVACIONES INTRODUCTORIAS

- **Marcas comerciales y marcas de servicio.**
  Todas las marcas comerciales, marcas de servicio, logotipos y nombres de compañías mencionadas en este trabajo son propiedad de sus respectivos dueños.

- **Glosario de términos.**
  Se recomienda consultar la sección Glosario de Términos antes de comenzar a leer el texto principal de este trabajo. Los términos aparecen ordenados alfabéticamente.

- **Notas al pie de página.**
  Aparecen numeradas al pie de página y proporcionan información adicional sin interrumpir la secuencia lógica del texto principal.

- **Referencias bibliográficas.**
  Desde el texto principal, se indican entre paréntesis con el primer apellido del(de los) autor(es) y el año de publicación de la obra, por ejemplo (MARTÍNEZ y SÁEZ, 2009). Posteriormente, en la sección Referencias Bibliográficas aparecen ordenadas alfabéticamente y con los datos completos de cada obra:

  MARTÍNEZ A.W. y SÁEZ S. 2009. *Recolector de Contenido Web.* Tesis Diploma.
  Cuidad de La Habana, Cuba: Universidad de las Ciencias Informáticas,
  Facultad 10, 2009. [118] p.
  Disponible en:
  http://biblioteca.uci.cu

- **Formatos para lectura.**
  El presente trabajo se ha optimizado para uso digital e impreso en papel Letter. No obstante, se recomienda usar la versión digital cuando así proceda; puesto que es un archivo PDF, generado a partir de un documento OTD creado en OpenOffice Writer, con enlaces que facilitan la navegación.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS.....XI
ÍNDICE DE TABLAS.....XII
LISTA DE ACRÓNIMOS.....XIV
RESUMEN.....XV
ABSTRACT.....XVI
INTRODUCCIÓN.....1
1. CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.....6
1.1. Introducción.....6
1.2. La Web.....6
1.2.1. Conceptos básicos.....8
1.3. Recuperación de Información.....10
1.3.1. Métodos para recuperar información.....11
1.3.1.1. Directorios Web Temáticos.....12
1.3.1.2. Buscadores Web.....13
1.4. Recolección Masiva de Contenido Web.....17
1.4.1. Enfoque manual.....18
1.4.2. Enfoque automático.....19
1.4.2.1. Recolectores de contenido web.....20
1.4.2.1.1. Políticas de comportamiento.....23
1.4.2.1.2. Recolectores en paralelo.....24
1.4.2.1.3. Protocolo Estándar para la Exclusión de Robots.....27
1.4.2.1.4. Robots web privativos.....27
1.4.2.1.5. Robots web libres.....29
1.4.2.1.6. Características ideales.....32
1.5. Necesidades de información de PPSINI.....32
1.6. Herramientas a utilizar.....35
1.7. Metodología de desarrollo.....39
1.7.1. Metodología ágil SXP.....40
1.8. Investigaciones similares.....41
1.9. Conclusiones.....43

2. CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.....45
2.1. Introducción.....45
2.2. Modelo del Dominio.....45
2.3. Descripción de la Propuesta del Sistema.....47
2.4. Modelo de Datos.....49
2.5. Lista de Reserva del Producto.....51
2.6. Historias de Usuario.....54
2.7. Diagrama de Componentes.....62
2.8. Conclusiones.....64
3. CAPÍTULO 3: DESARROLLO Y VALIDACIÓN DE SINIBOT.....65
3.1. Introducción.....65
3.2. Plan de Releases.....65
3.3. Tareas de Ingeniería.....66
3.4. Casos de Prueba de Aceptación.....78
3.5. Resultados obtenidos.....87
3.5.1. Acerca del tiempo de desarrollo.....87
3.5.2. Acerca de las funcionalidades obtenidas.....88
3.6. Conclusiones.....88
CONCLUSIONES GENERALES.....90
RECOMENDACIONES.....92
REFERENCIAS BIBLIOGRÁFICAS.....93
GLOSARIO DE TÉRMINOS.....101

ÍNDICE DE FIGURAS

Figura 1: Principales características de la Web.....9
Figura 2: Ejemplo de índice invertido para tres páginas web.....14
Figura 3: Diseño en cascada de un buscador web.....16
Figura 4: Crecimiento de la Web.....19
Figura 5: Funcionamiento de un recolector de contenido web.....22
Figura 6: Recolección de contenido web mediante un único proceso.....25
Figura 7: Recolección paralela en un mismo ordenador.....26
Figura 8: Recolección distribuida en diferentes ordenadores.....26
Figura 9: Modelo del Dominio.....46
Figura 10: Modelo de Datos del Módulo de Gestión de Información.....50
Figura 11: Modelo de Datos del Módulo de Gestión de Usuarios.....51
Figura 12: Diagrama de Componentes.....63
Anexo 1: Prototipo de Interfaz del Módulo de Gestión de Recorridos.....98
Anexo 2: Prototipo de Interfaz del Módulo de Gestión de Información.....99
Anexo 3: Prototipo de Interfaz del Módulo de Gestión de Usuarios.....100

ÍNDICE DE TABLAS

Tabla 1: Lista de Reserva del Producto.....53
Tabla 2: Historia de Usuario Comenzar Recorrido.....54
Tabla 3: Historia de Usuario Extraer URL a Visitar.....55
Tabla 4: Historia de Usuario Recuperar Características y Contenido.....55
Tabla 5: Historia de Usuario Detectar Error.....56
Tabla 6: Historia de Usuario Extraer Enlaces.....56
Tabla 7: Historia de Usuario Realizar Recolección en Paralelo.....57
Tabla 8: Historia de Usuario Generar Estadísticas.....57
Tabla 9: Historia de Usuario Almacenar Información.....58
Tabla 10: Historia de Usuario Establecer Lista de URLs Iniciales.....58
Tabla 11: Historia de Usuario Configurar Recorrido.....59
Tabla 12: Historia de Usuario Gestionar Estado de los Recorridos.....59
Tabla 13: Historia de Usuario Gestionar Información Recolectada.....60
Tabla 14: Historia de Usuario Gestionar Usuarios.....60
Tabla 15: Historia de Usuario Almacenar Características de Hardware.....61
Tabla 16: Historia de Usuario Crear Instalador.....61
Tabla 17: Historia de Usuario Realizar Tratamiento de Errores.....62
Tabla 18: Plan de Releases de la versión v0.1 de SINIBOT.....66
Tabla 19: Tarea de Ingeniería Estudiar Almacenamiento de Contenido en Larbin.....67
Tabla 20: Tarea de Ingeniería Estudiar Librería pqxx.....67
Tabla 21: Tarea de Ingeniería Crear Base de Datos.....68
Tabla 22: Tarea de Ingeniería Almacenar Contenido Complementario.....68
Tabla 23: Tarea de Ingeniería Estudiar FreeNAS.....69
Tabla 24: Tarea de Ingeniería Instalar FreeNAS.....69
Tabla 25: Tarea de Ingeniería Almacenar Contenido de los Documentos.....70
Tabla 26: Tarea de Ingeniería Estudiar Obtención de URLs de Partida.....70
Tabla 27: Tarea de Ingeniería Interfaz para Establecer las URLs de Partida.....71
Tabla 28: Tarea de Ingeniería Almacenar URLs de Partida.....71
Tabla 29: Tarea de Ingeniería Estudiar Opciones de Configuración.....72
Tabla 30: Tarea de Ingeniería Interfaz para Configurar los Recorridos.....72

Tabla 31: Tarea de Ingeniería Almacenar Opciones de Configuración.....73
Tabla 32: Tarea de Ingeniería Gestionar Estado de los Recorridos.....73
Tabla 33: Tarea de Ingeniería Gestionar Información Recolectada.....74
Tabla 34: Tarea de Ingeniería Estudiar Librería FreakAuth_light.....74
Tabla 35: Tarea de Ingeniería Configurar la Librería FreakAuth_light.....75
Tabla 36: Tarea de Ingeniería Obtener Características de Hardware.....75
Tabla 37: Tarea de Ingeniería Almacenar Características de Hardware.....76
Tabla 38: Tarea de Ingeniería Estudiar Proceso de Instalación de Larbin.....76
Tabla 39: Tarea de Ingeniería Integración de los Componentes de SINIBOT.....77
Tabla 40: Tarea de Ingeniería Crear Instalador.....77
Tabla 41: Tarea de Ingeniería Realizar Tratamiento de Errores.....78
Tabla 42: Caso de Prueba de Aceptación U-SINIBOT-1-01.....79
Tabla 43: Caso de Prueba de Aceptación U-SINIBOT-2-01.....79
Tabla 44: Caso de Prueba de Aceptación U-SINIBOT-3-01.....80
Tabla 45: Caso de Prueba de Aceptación U-SINIBOT-4-01.....80
Tabla 46: Caso de Prueba de Aceptación U-SINIBOT-5-01.....81
Tabla 47: Caso de Prueba de Aceptación U-SINIBOT-6-01.....81
Tabla 48: Caso de Prueba de Aceptación U-SINIBOT-7-01.....82
Tabla 49: Caso de Prueba de Aceptación U-SINIBOT-8-01.....82
Tabla 50: Caso de Prueba de Aceptación U-SINIBOT-9-01.....83
Tabla 51: Caso de Prueba de Aceptación U-SINIBOT-10-01.....83
Tabla 52: Caso de Prueba de Aceptación U-SINIBOT-11-01.....84
Tabla 53: Caso de Prueba de Aceptación U-SINIBOT-11-02.....84
Tabla 54: Caso de Prueba de Aceptación U-SINIBOT-12-01.....85
Tabla 55: Caso de Prueba de Aceptación U-SINIBOT-13-01.....85
Tabla 56: Caso de Prueba de Aceptación U-SINIBOT-14-01.....86
Tabla 57: Caso de Prueba de Aceptación U-SINIBOT-15-01.....86

LISTA DE ACRÓNIMOS

- **BUSCAWEB** – Buscador Web.
- **CPAN** – *Comprehensive Perl Active Network.*
- **FTP** – Protocolo para Transferencia de Archivos, *(File Transfer Protocol).*
- **GEWEB** – Generador de Estudios Webmétricos.
- **GPL** – Licencia Pública General, *(General Public License).*
- **GPsRI** – Grupo de Proyectos de Recuperación de Información.
- **HTML** – Lenguaje para el Formato de Documentos de Hipertexto, *(HyperText Markup Language).*
- **HTTP** – Protocolo para la Transferencia de Hipertexto, *(HyperText Transfer Protocol).*
- **MOCIC** – Motor de Clasificación Inteligente de Contenidos.
- **PDF** – Formato de Documento Portátil *(Portable Document Format)*.
- **PPSINI** – Polo Productivo de Soluciones Informáticas para Internet.
- **SXP** – SCRUM + XP, representa la unión de ambas metodologías.
- **UCI** – Universidad de las Ciencias Informáticas.
- **URL** – Localizador Uniforme de Recurso, *(Uniform Resource Locator).*
- **WIRE** – Entorno de Recuperación de Información Web, *(Web Information Retrieval Environment)*.
- **XML** – Lenguaje de Marcado eXtensible, *(eXtensible Markup Language).*

RESUMEN

En Octubre de 2008, en la Universidad de las Ciencias Informáticas se crearon los proyectos productivos: Motor de Clasificación Inteligente de Contenidos, Generador de Estudios Webmétricos y Buscador Web, encargados de desarrollar los productos informáticos MOCIC, GEWEB y FARO respectivamente. Estos productos requieren procesar grandes volúmenes de contenido web. La naturaleza dinámica de la Web implica que sea prácticamente inviable realizar la recolección de su contenido de forma manual e independiente, al tomar en consideración el tiempo y los recursos que este proceso consumiría. Estas razones son la fuente de motivación del Grupo de Proyectos de Recuperación de Información para desarrollar un sistema informático capaz de recorrer automáticamente la Web y recolectar parte de su contenido. El propósito general del presente trabajo es mostrar el desarrollo de un sistema automatizado de recolección de contenido web (SINIBOT) para proporcionar a otros productos la información que requieren. Para ello, se revisó la información teórica relacionada con la recolección de contenido web; luego se definieron las características de SINIBOT y se procedió a su implementación. La aplicación fue desarrollada sobre la plataforma Larbin en su versión v2.6.3, obteniendo como resultado una versión mejorada de dicho sistema, capaz de satisfacer las necesidades de información de MOCIC, GEWEB y FARO, así como de futuros productos. El contenido recolectado se almacena en una base de datos diseñada según las necesidades de los productos antes mencionados. SINIBOT tiene además una interfaz web para la administración, configuración y gestión de cada uno de los recorridos efectuados. El presente trabajo representa una base teórica para el desarrollo de sistemas afines.

Palabras claves: automatizado, contenido web, Larbin, SINIBOT, recolección.

ABSTRACT

In October 2008 at the University of the Informatics Sciences, production projects such as Intelligent Content Classification Engine, Webmetrics Studies Generator and Web Seeker were created to develop products like MOCIC, GEWEB and FARO respectively. These products require processing large web-content volumes. The dynamic nature of the Web implies that the manual and independent collection of its contents is practically unfeasible when considering the time and resources needed by this process to be developed. This is the leitmotiv for the Team of Information Retrieval Projects to develop a system which can automatically traverse the Web and crawl some of the contents. The general purpose of this study is showing the development of an automated system for web-content crawling (SINIBOT) to provide other products with the information required. For this purpose it was necessary studying theoretical information related to web-content crawling, defining the characteristics of SINIBOT and implementing it. The application was developed based on Larbin v2.6.3 platform, getting as a result an improved version of that system, capable of satisfying information needs from MOCIC, GEWEG and FARO, as well as from future products. The crawled content is stored in a database designed to meet the needs of the above mentioned products. SINIBOT also have a web interface for administration, configuration and management of each of the journeys made. This work represents a theoretical base for the development of related systems.

Key words: automated, crawling, Larbin, SINIBOT, web content.

INTRODUCCIÓN

Internet representa en la actualidad uno de los espacios más utilizados por la población debido a los servicios que brinda. Dentro de sus servicios se encuentra la Web[1], la cual contiene un gran volumen de información que puede visualizarse principalmente por medio de texto, imágenes y videos. Este contenido aparece en varios idiomas y es accesible prácticamente desde cualquier parte, haciendo de la Web una de las principales fuentes de información a escala mundial.

El exponencial crecimiento de la Web, la diversidad de información que contiene y el no cumplimiento de los estándares para la organización de su contenido, hacen necesario el uso de herramientas que faciliten la búsqueda de la información solicitada por los usuarios. Entre estas herramientas se encuentran los Buscadores Web que, a través de una interfaz, muestran a los usuarios un conjunto de enlaces a documentos relacionados con la información que han solicitado. Para responder a las solicitudes de los usuarios las búsquedas realizadas por estas herramientas no se efectúan directamente sobre la Web, sino que se ejecutan sobre bases de datos que almacenan parte del contenido presente en la misma, el cual se ha recolectado con anterioridad.

La Recolección Masiva de Contenido Web es un caso peculiar de la Recuperación de Información, la cual se remonta a la década del '50. El término de Recuperación de Información fue acuñado en 1952 y fue ganando popularidad en la comunidad científica de 1961 en adelante. El comienzo de la década de los '90 señala el debut de la Web como un servicio disponible públicamente en Internet, dando inicio al vertiginoso crecimiento de información disponible en la red (MARTÍNEZ, 2002). Este período fue el escenario donde surgieron y se desarrollaron la mayoría de los

1 World Wide Web: sistema de documentos de hipertexto enlazados y accesibles a través de Internet.

actuales motores de búsqueda (RODRÍGUEZ, 2003).

Los principales métodos para recolectar contenido web son: (1) mediante sistemas automáticos de recolección y (2) manualmente, a través de un navegador web. A continuación se presentan algunos problemas relacionados con estos métodos:

- la recolección manual de contenido web constituye un proceso consumidor de tiempo y recursos, además de estar propenso a errores humanos.
- el contenido recolectado debe ser almacenado en un formato consecuente con las necesidades de quienes lo requieren.
- el contenido recolectado por una persona desde un navegador web no presenta toda la información que pudiera resultar útil para quienes lo requieren.
- al tener un grupo de personas donde cada una recolecte contenido web desde un navegador, es muy difícil garantizar que al menos dos de estas no accedan al mismo documento.

En la Universidad de las Ciencias Informáticas (UCI) se crearon en Octubre de 2008 los proyectos productivos: Motor de Clasificación Inteligente de Contenidos, Generador de Estudios Webmétricos y Buscador Web, encargados de desarrollar los productos informáticos MOCIC, GEWEB y FARO respectivamente. Estos productos requieren procesar grandes volúmenes de contenido web. La naturaleza dinámica de la Web implica que sea prácticamente inviable realizar la recolección de su contenido de forma manual e independiente, al tomar en consideración el tiempo y los recursos que este proceso consumiría.

Por lo anteriormente expuesto, al Grupo de Proyectos de Recuperación de Información (GPsRI[2]) se le asignó la tarea de desarrollar un producto informático que

2 Grupo de investigación y desarrollo (I+D) perteneciente al Polo de Soluciones Informáticas para Internet.

sea capaz de recorrer automáticamente la Web para recolectar y almacenar parte de su contenido. Por tal motivo surge el siguiente problema a resolver: ¿Cómo automatizar la recolección de contenido web para suministrar a otros productos la información que requieren?

La idea a defender, establecida como base de la investigación efectuada, se formula en los siguientes términos: El contenido web suministrado a productos informáticos que lo requieran debe ser recolectado mediante un sistema automatizado.

A partir de lo anterior, se define como objeto de estudio la Recuperación de Información y como campo de acción la Recolección Masiva de Contenido Web.

El objetivo general de este trabajo, derivado del problema a resolver, es el siguiente: Desarrollar un sistema informático de recolección de contenido web para suministrar a otros productos la información que requieren. El objetivo anterior puede articularse en los siguientes objetivos específicos:

- Revisar la teoría existente sobre recolección de contenido web para conocer qué se ha investigado del tema y reducir el ámbito de los hechos por tratar.
- Definir las características del Recolector de Contenido Web SINIBOT.
- Desarrollar y validar la solución propuesta.

Para dar cumplimiento a los objetivos propuestos se precisa de las siguientes tareas de investigación:

- Para revisar la teoría relacionada con la recolección de contenido web:
  - Analizar la estructura y el contenido de la Web.
  - Estudiar los principales métodos para recuperar información y recolectar contenido web.
  - Investigar sobre las tecnologías y herramientas adecuadas para el desarrollo

de SINIBOT, teniendo en cuenta: lenguaje de programación, sistema gestor de bases de datos y metodología de desarrollo; así como soluciones libres sobre las cuales sea factible continuar desarrollando.

- Para definir las características de SINIBOT:
  - Estudiar estrategias para modelar el problema a resolver a través del Modelo del Dominio.
  - Analizar una estructura adecuada para SINIBOT donde se identifiquen, se describan y se modelen las funcionalidades que ha de tener el mismo.
- Para desarrollar y validar la solución propuesta:
  - Estudiar los pasos para identificar las Tareas de Ingeniería que guiarán el desarrollo de SINIBOT.
  - Estudiar las estrategias para definir el Plan de *Releases* del producto.
  - Investigar sobre los métodos para confeccionar los Casos de Prueba de Aceptación que validarán las funcionalidades desarrolladas.

Durante una investigación científica el uso de métodos de investigación provee estrategias fundamentales para ahorrar tiempo y esfuerzo. Para guiar las tareas investigativas se hace uso de los siguientes métodos de investigación:

- Histórico-Lógico: se utiliza para analizar la trayectoria completa de la Recolección Masiva de Contenido Web al estudiar las etapas por las cuales ha transitado. Permitirá una mejor comprensión del estado actual de la teoría.
- Analítico-Sintético: se aplica para lograr el entendimiento de la Recolección Masiva de Contenido Web a través de sus características. Permitirá formular conclusiones basadas en la síntesis de los conocimientos y resultados obtenidos.
- Modelación: se emplea para modelar la estructura, relaciones internas y características de la solución a través de diagramas.

A continuación se describe brevemente la estructura del presente documento

(organización de la memoria), resumiéndose el contenido de cada capítulo:

- En el Capítulo 1 (Fundamentación Teórica) se hará una revisión de la Web. Se introducirá la Recuperación de Información y los métodos de obtención de información, describiendo la Recolección Masiva de Contenido Web y sus métodos principales, especificando los enfoques manual y automático. Sobre los robots web se abordará: surgimiento, políticas de comportamiento, recolectores en paralelo, el Protocolo Estándar para la Exclusión de Robots, ámbito social y tecnológico y las características ideales que deben tener estos sistemas. Se evidenciarán además las necesidades de información del Polo Productivo de Soluciones Informáticas para Internet. Finalmente, serán presentadas investigaciones similares a la desarrollada en este trabajo y las conclusiones que reflejarán la posición de los autores respecto a la teoría tratada.
- En el Capítulo 2 (Características del Sistema) se presentará el Modelo del Dominio con el objetivo de comprender el contexto del sistema. Se realizará la Descripción de la Propuesta del Sistema, describiendo cómo debe funcionar y destacando sus características distintivas. Se mostrará la Lista de Reserva del Producto, especificando sus Requisitos Funcionales y No Funcionales. Se elaborarán las Historias de Usuarios y el Diagrama de Componentes, donde se muestran los paquetes que conforman el sistema y las relaciones entre ellos.
- En el Capítulo 3 (Desarrollo y Validación de SINIBOT) se realizará el Plan de *Releases* para el control de las entregas inmediatas. Se presentarán las Tareas de Ingeniería que guiarán el proceso de desarrollo del sistema. Se mostrarán los Casos de Prueba a los cuales fue sometido SINIBOT en cada una de las iteraciones del proyecto. Por último, se exponen los resultados obtenidos y las funcionalidades alcanzadas durante el período de desarrollo.

Al final se presentan las Conclusiones, Recomendaciones, Referencias Bibliográficas, Anexos y el Glosario de Términos.

1. CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En el presente capítulo se hará una revisión de la Web. Se introducirá la Recuperación de Información y los métodos de obtención de información, describiendo la Recolección Masiva de Contenido Web y sus métodos principales, especificando los enfoques manual y automático. Sobre los robots web se abordará: surgimiento, políticas de comportamiento, recolectores en paralelo, el Protocolo Estándar para la Exclusión de Robots, ámbito social y tecnológico y las características ideales que deben tener estos sistemas. Se evidenciarán además las necesidades de información del Polo Productivo de Soluciones Informáticas para Internet. Finalmente, serán presentadas investigaciones similares a la desarrollada en este trabajo y las conclusiones que reflejarán la posición de los autores respecto a la teoría tratada.

1.2. La Web

A principios de la década de 1990, Tim Berners-Lee tuvo la idea de crear un sistema global de información con el objetivo de fusionar las tecnologías de computadoras personales, para satisfacer la demanda de intercambio automático de información entre los científicos que trabajaban en diferentes universidades e institutos de todo el mundo (CERN, 2008).

Con este sistema cada usuario en un ordenador podía navegar de forma automática por los nodos de una red, sin importar cómo funcionaban los sistemas de esos nodos. Este sistema global de información fue conocido como *World Wide Web*. En (GUTIÉRREZ *et al*., 2008) se expresan las siguientes palabras de Berners-Lee: “El concepto de la Web integró muchos sistemas de información diferentes, por medio

de la formación de un espacio imaginario abstracto en el cual las diferencias entre ellos no existían. La Web tenía que incluir toda la información de cualquier tipo en cualquier sistema."

La arquitectura lógica de la Web se sustenta en tres pilares fundamentales, los cuales son:

1. URI: Uniform Resource Identifier (Identificador Uniforme de Recurso). Para poder referenciar a todos los documentos en la Web es necesario que cada uno tenga su nombre propio, el cual se conoce como URI. Una versión más elemental de URI es URL: Uniform Resourse Locator (Localizador Uniforme de Recurso), el cual representa una dirección en la Web. Aunque la dirección es una de las formas de identificar un documento, es importante señalar que el concepto de identificador es más amplio que el de localizador, sobre todo para recursos móviles, los cuales no tienen una dirección fija.

2. HTML: Hyper Text Markup Language (Lenguaje de Marcas de Hipertexto). Lenguaje de marcado creado por Berners-Lee para la construcción de páginas web, a través del cual se define la estructura y el contenido de los documentos. Además de su simplicidad de uso, el hecho de ser un lenguaje de hipertexto permite redireccionar al lector desde un punto cualquiera del texto a otro objeto en la Web. Estos puntos de redireccionamiento son conocidos como *links* o enlaces.

3. HTTP: Hyper Text Transfer Protocol (Protocolo para la Transferencia de Hipertexto). Permite enviar y recibir información desde un lugar a otro en la Web, logrando así la comunicación entre clientes y servidores web. El cliente puede ser un navegador web o un agente, mientras que el servidor es quien almacena o crea los recursos requeridos por el cliente, tales como documentos HTML o imágenes, por solo citar algunos.

La popularidad de la Web como uno de los servicios de Internet ha provocado que en muchos casos sean confundidos ambos términos. Desde el punto de vista técnico son completamente diferentes, pues Internet es la red física que hace posible establecer la conexión entre distintos nodos, mientras que la Web representa la información que está publicada sobre dicha red.

El crecimiento de la Web es exponencial, debido principalmente a la facilidad y libertad de publicar contenido; cualquier persona sin distinción de tipo o condición, que posea conocimientos básicos sobre el lenguaje HTML, puede crear y publicar una página web (PÉREZ y RIPOLL, 2008). Se estima que más de la mitad de la Web ha sido creada o modificada en los últimos seis meses y que el contenido actual de la misma es de miles de terabytes[3], distribuidos principalmente en texto, imágenes, audio y video (GUTIÉRREZ et al., 2008).

1.2.1. Conceptos básicos

Un documento web es un fichero identificado por una URL y puede representar, entre otros formatos, una imagen, un archivo multimedia, un archivo de texto, un documento en formato PDF, un archivo comprimido o una página web. Por su parte, una página web es un documento codificado en lenguaje HTML que puede contener enlaces a otros documentos por medio de sus respectivas URLs (GUTIÉRREZ *et al.*, 2008).

Baeza-Yates en (GUTIÉRREZ *et al.*, 2008) define a la Web como compleja, basándose en las características de las páginas que la componen (ver Figura 1), expresando que existen páginas estáticas, dinámicas, públicas y privadas.

3 Terabyte: unidad de medida de almacenamiento de datos cuyo símbolo es Tb y equivale a 1024 Gb.

Las páginas estáticas son aquellas que están en un servidor web antes de ser solicitadas por un usuario, o sea, existen en todo momento, mientras que las páginas dinámicas no siempre están presentes en el servidor, sino que se crean justo en el instante que el usuario las solicita, principalmente por medio de una consulta al servidor o como resultado de llenar los datos de un formulario. Mientras que la cantidad de información que existe en la Web es finita, el número total de páginas web es potencialmente infinito, debido a que en algunos sitios se puede generar un número no acotado de páginas dinámicas. Actualmente, la mayor parte de la Web está constituida por páginas dinámicas. Las páginas públicas son accesibles para todos los usuarios, en tanto las privadas son aquellas que están protegidas por contraseñas o están dentro de una Intranet.

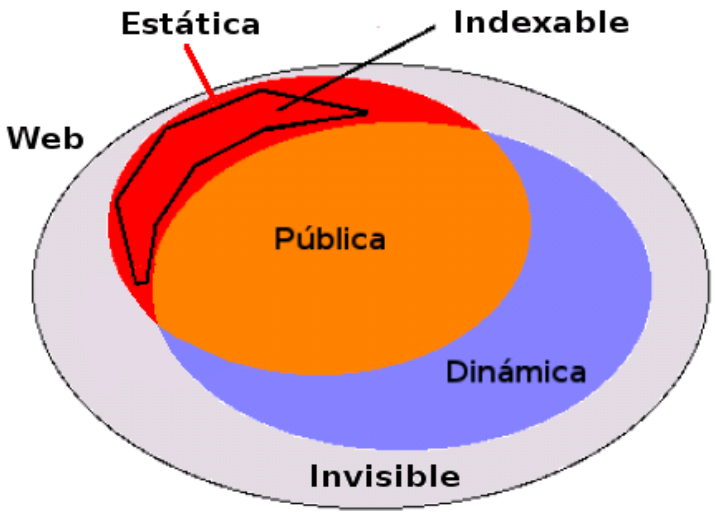


Figura 1: Principales características de la Web.

Tomado de (GUTIÉRREZ *et al.*, 2008)

En (BOJO *et al.,* 2004) los autores definen también los conceptos de Web Visible e Invisible, definiendo la parte visible como aquellos documentos cuyo contenido

puede ser mostrado a los usuarios por medio de buscadores web, mientras que la porción invisible se corresponde con los documentos no accesibles para estos buscadores, representando principalmente a contenidos recogidos en bases de datos u otros recursos informativos, los cuales necesitan ser consultados previamente para acceder a ellos, algo que los buscadores tradicionales no pueden realizar.

1.3. Recuperación de Información

En (PEÑAS, 2002) se define la Recuperación de Información como el término utilizado para obtener documentos relevantes al realizar una petición de información. En (CASTILLO, 2004) se expresa que es el área de la informática concerniente a obtener información sobre un tema en una colección de datos, donde no solo se determina cuáles documentos contienen la palabra clave de la consulta realizada por el usuario, sino que se satisfacen cualitativamente las necesidades de información del mismo.

Los Sistemas de Recuperación de Información según (BAEZA-YATES y RIBEIRO-NETO, 1999), deben de alguna manera interpretar el contenido de la información dentro de una colección de documentos y establecer con ellos un *ranking* de acuerdo al grado de relevancia que estos posean para las consultas de los usuarios.

Ricardo Baeza-Yates, en su artículo (BAEZA-YATES, 1999) presenta los principales problemas que, desde el punto de vista del contenido, enfrenta la Recuperación de Información:

- Distribuido: dada la estructura de la Web, los datos están en muchos computadores y sobre distintas plataformas. La topología de la red no está predefinida y el ancho de banda y confiabilidad de las conexiones es muy variable.
- Volátil: los nombres de dominio y páginas aparecen y desaparecen de la red

diariamente. Además, el volumen de los datos crece exponencialmente, doblando su tamaño en aproximadamente seis meses.

- Dinámico: actualmente la gran mayoría de las páginas se generan mediante una consulta a una base de datos y por ende es difícil recuperarlas sin conocer su estructura.
- Redundante: una porción de la Web está duplicada. La cantidad de información replicada, según Baeza-Yates en (GUTIÉRREZ *et al.*, 2008), es alrededor del 20% del total del contenido de la Web.
- Tipos heterogéneos: existen variedad de medios digitales, de cada medio hay distintos formatos (por ejemplo, HTML o Word para texto, o JPG y GIF para imágenes). Existen además diferentes lenguajes y alfabetos.
- Calidad heterogénea: la Web es un medio de publicación en el que los contenidos no pasan ningún tipo de proceso editorial. Por lo tanto la información de una página puede ser falsa, inválida, mal escrita, o presentar otros errores.

1.3.1. Métodos para recuperar información

Los autores de (BAEZA-YATES y RIBEIRO-NETO, 1999) definen que existen tres métodos principales para recuperar información en la Web: (1) mediante los motores de búsqueda o buscadores web, estos indexan una porción de los documentos existentes en la Web y permiten la localización de la información a través de una consulta; (2) haciendo uso de los directorios web temáticos, los cuales contienen documentos web previamente clasificados por categorías temáticas, facilitando así la localización de la información para los usuarios; y (3) a través de la explotación manual de la estructura de la Web, la cual no es una práctica extendida, debido principalmente a que los usuarios desconocen la ubicación exacta de la mayoría de la información que pudiera resultarles útil, es por ello que las formas de obtención de información más comunes y eficientes son los directorios web temáticos y los

buscadores web.

1.3.1.1. Directorios Web Temáticos

Los directorios web temáticos se definen en (MORENO, 2005) como sitios que contienen documentos organizados, según su categoría temática, en una estructura jerárquica mediante un proceso manual de selección editorial por un grupo de especialistas. Los directorios web más conocidos son Open Directory Project[4] (ODP) y el directorio de Yahoo![5].

Como método para recuperar información, los directorios web temáticos presentan las siguientes ventajas:

- La información disponible en los directorios es clasificada por especialistas de diversas materias, lo que la convierte en una fuente de recursos de reconocida calidad.
- Para usuarios no experimentados resultan de más fácil manejo que los buscadores.
- Permiten tener una visión global del contenido.

Como desventajas se pueden señalar:

- Cubren tan solo una pequeña fracción de los contenidos de la Web.
- Carecen de criterios homogéneos para la selección, clasificación y descripción del contenido.
- Muchos documentos dejan de ser útiles porque no se utilizan mecanismos para seguir los cambios en sus contenidos, direcciones, aparición o desaparición.

4 http://www.dmoz.org
5 http://www.yahoo.com

- Los documentos presentes no siempre son los más actuales.
- En ocasiones, para obtener un documento, es necesario navegar por varios niveles de profundidad en la estructura jerárquica del directorio.
- El crecimiento de los directorios es mucho más lento que el de la Web.

1.3.1.2. Buscadores Web

Los motores de búsqueda o buscadores web, son las herramientas de acceso a la información más populares y útiles en Internet, programadas para la localización y recuperación de información en la red (BOJO *et al*., 2004). Los buscadores extraen de los documentos las palabras o términos que mejor los representen a través de un proceso automático, con estas palabras generan un índice invertido (ver Figura 2) y almacenan esta información en una gigantesca base de datos que puede ser interrogada por los usuarios a través de una interfaz.

Cuando los usuarios efectúan una consulta, el motor busca en esta base de datos y devuelve, como respuesta, una lista con las URLs de aquellos documentos que se ajustan a los criterios establecidos en dicha consulta. Esta búsqueda no se efectúa directamente sobre la Web debido a su inmenso volumen y al tiempo requerido para dar respuesta a los usuarios.

En (BERGMAN, 2001) el autor expresa que el 85% de los usuarios de Internet utiliza motores de búsqueda para satisfacer sus necesidades de información. Los buscadores web, según Gonzalo Navarro en (GUTIÉRREZ *et al., 2008*) permiten escribir un par de palabras y obtener inmediatamente abundante información, en general muy relevante, que satisface las necesidades de los usuarios. Los representantes más conocidos de los buscadores web son Yahoo!, Google[6] y Microsoft MSN[7].

6 http://www.google.com
7 http://www.msn.com

Figura 2: Ejemplo de índice invertido para tres páginas web.

Tomado de (GUTIÉRREZ *et al*., 2008)

Sin un buscador los usuarios deberían conocer las direcciones en la Web de todos los sitios que pudieran resultarles interesantes o necesarios, perdiendo toda la información de los sitios que no conozcan. En un sentido muy real los buscadores conectan la Web, pues existen grandes porciones de esta a las que solo se llega utilizando un buscador. Según (GUTIÉRREZ *et al*., 2008), casi un tercio del tiempo que los usuarios pasan en Internet lo dedican a hacer búsquedas en la Web.

En (BAEZA-YATES *et al.*, 2007) se expresa que desde la perspectiva del usuario existen dos requisitos principales que deben cumplir todos los buscadores: (1) un

tiempo corto de respuesta y (2) una gran colección de documentos web disponibles en su índice. La calidad de un buscador reside en lo abundante, relevante y actualizada que sea su colección.

Estudios realizados aseguran que en el 2004 el 40% de los usuarios llegaban a los sitios web por primera vez tras seguir los resultados obtenidos por una consulta en un buscador (NIELSEN, 2004). Por otra parte en (COMSCORE, 2008) se asegura que en Junio de 2008 más del 91% del tráfico en las páginas pertenecientes a Estados Unidos estaba generado por los grandes buscadores, distribuidos de la siguiente forma: Google 61.5%, Yahoo! 20,9% y Microsoft MSN 9.2%.

En (CASTILLO, 2004) se explica que el diseño típico de un buscador web es en cascada (ver Figura 3), en el cual las operaciones son ejecutadas en el siguiente orden:

1) Recolección de contenido web.
2) Indexación del contenido recolectado.
3) Búsqueda de información por parte de los usuarios.

Donde se considera que el proceso de recolección es la etapa primaria y más importante dentro de la búsqueda en la Web.

Como método para recuperar información, los buscadores web presentan las siguientes ventajas:

- Realizan el descubrimiento del contenido de la Web mediante un proceso automático, lo que permite analizar una mayor cantidad de información.
- Mantienen sus bases de datos más actualizadas que los directorios.
- Asignan a la información recolectada un *ranking* de acuerdo a la calidad de su contenido.
- Permiten la realización de búsquedas avanzadas.

Como desventajas se pueden señalar:

- La clasificación de los contenidos es automática, por lo que carece de calidad en comparación con la de un directorio.
- La información encontrada no siempre responde a las necesidades reales de los usuarios.
- En ocasiones devuelven demasiadas URLs para una misma consulta, saturando de información al usuario. Esto provoca que se necesite examinar una amplia colección de documentos o formular una nueva consulta.

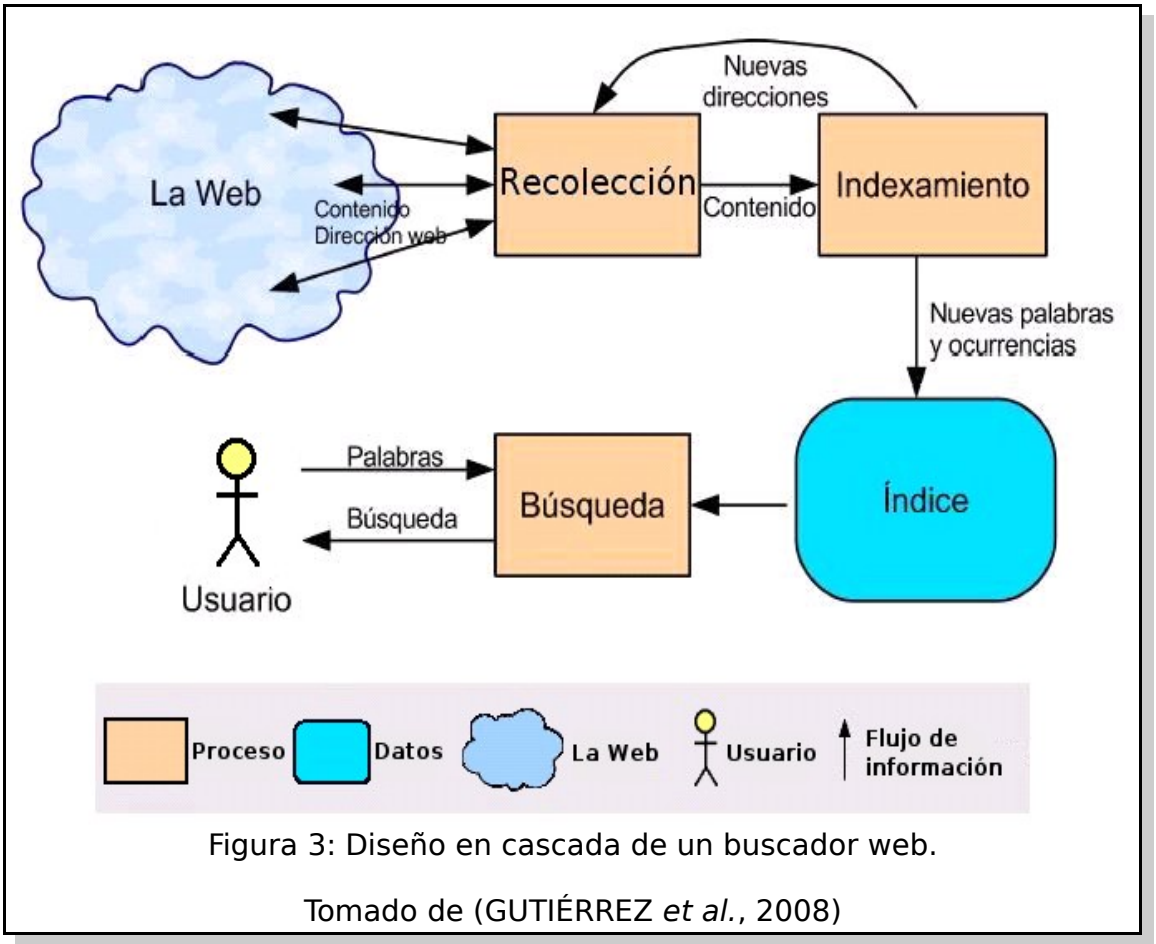


Figura 3: Diseño en cascada de un buscador web.

Tomado de (GUTIÉRREZ *et al.*, 2008)

1.4. Recolección Masiva de Contenido Web

En (BAEZA-YATES *et al.*, 2005) se define la Recolección Masiva de Contenido Web como el proceso mediante el cual se visita el mayor número posible de páginas web y se almacena su contenido. Este puede servir entre otros fines, para generar el índice de un buscador, o realizar un estudio de caracterización de la Web en general o de una región en específico.

Carlos Castillo menciona en (CASTILLO, 2004) dos características muy importantes de la Web que generan un escenario en el cual la Recolección Masiva de Contenido Web es muy difícil: (1) su inmensa cantidad de contenidos y (2) la velocidad de cambio que estos experimentan. La primera trae como consecuencia que solo sea posible descargar una parte de la Web, mientras que la segunda provoca que durante el proceso de recolección, mucha de la información ya almacenada ha cambiado, ha sido eliminada o se han agregado nuevas páginas a sitios que ya fueron visitados.

Además, las siguientes situaciones especiales definidas en (BAEZA-YATES *et al*., 2007) dificultan el proceso de recolección de contenido web:

- **Réplicas de contenidos:** constituye una práctica habitual en la Web tener varias copias distribuidas geográficamente de voluminosas colecciones de documentos. Esto trae como consecuencia que en ocasiones se recuperen páginas cuyo contenido se ha recolectado previamente. Por lo general, el diseño de las páginas replicadas no es el mismo, haciendo que su contenido no se detecte como duplicado durante el proceso de recolección.

- **Spam en la Web:** se refiere a acciones orientadas a engañar a los sistemas de búsqueda y recolección de contenido en la Web. Estas incluyen cambios en el texto en las cabeceras de las páginas o en sus enlaces.

1.4.1. Enfoque manual

La Recolección Masiva de Contenido Web en su enfoque manual, se define en (BAEZA-YATES et al., 2005), como las acciones realizadas por los usuarios, de forma independiente o colectiva y sin la utilización de un sistema automático, para localizar y descargar contenido de la Web.

En (GUTIÉRREZ *et al*., 2008) se expresa que en el año 2005 la Web tenía cerca de 11.500 millones de páginas web y que su volumen se duplica cada aproximadamente seis meses.

Tomando estos datos como referencia, los autores del presente trabajo estiman que el tamaño de la Web al concluir el año 2008 sería aproximadamente de 736.000 millones de páginas web (ver Figura 4), concluyendo que de realizarse una recolección manual de este volumen de contenido, utilizando para ello los 10.000 estudiantes de la UCI, con una conexión estable a Internet, trabajando todos los días del año durante 8 horas ininterrumpidas, asumiendo además que la Web no aumente su volumen (situación inconcebible) y que una página no sea descargada en más de una ocasión; si cada estudiante consume una media de 15 segundos para descargar una página web se necesitarían poco más de 100 años para completar esta tarea.

Debido al exponencial crecimiento que exhibe la Web y a la dinámica que presenta su contenido, no es viable encarar eficientemente la recolección de forma manual, ya que es prácticamente imposible mediante este proceso descargar millones de páginas web y mantener actualizada la colección resultante.

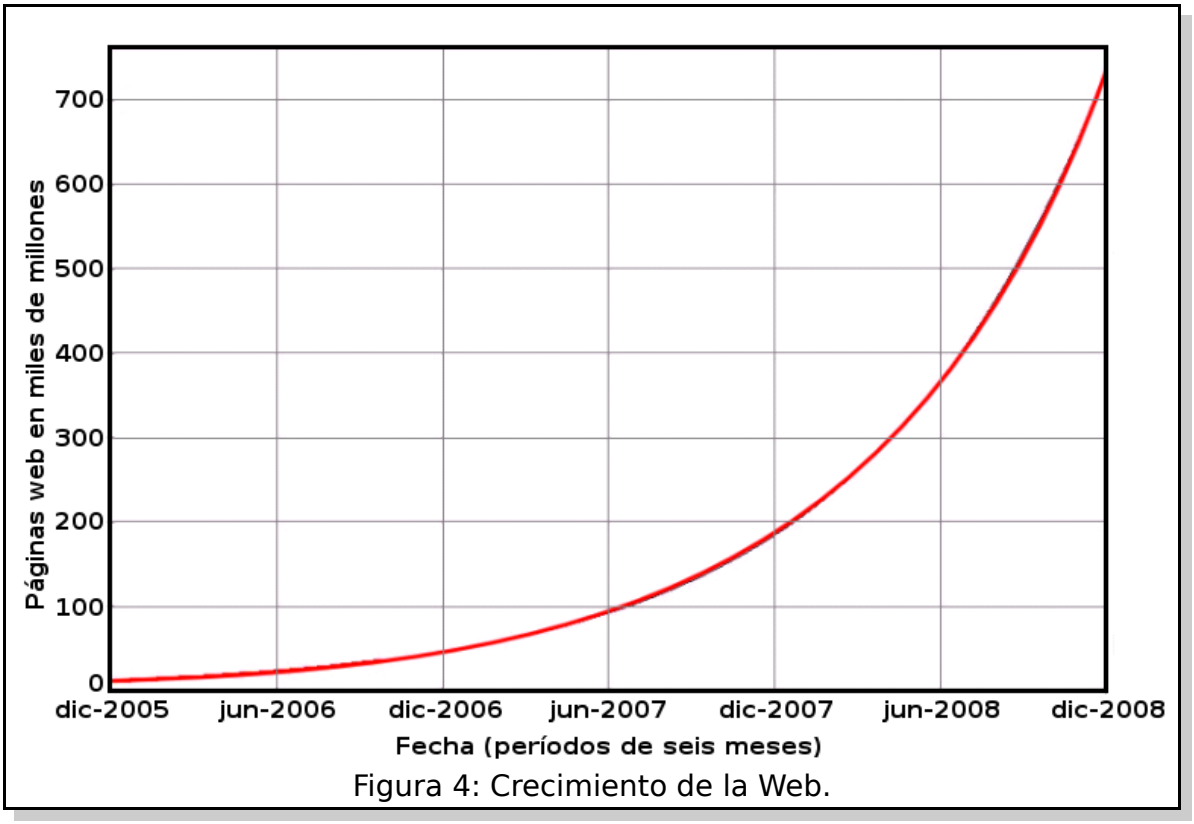


Figura 4: Crecimiento de la Web.

1.4.2. Enfoque automático

Para realizar consultas exitosas en un buscador web es necesario que este contenga información abundante y actualizada de la Web. Se explica en (GUTIÉRREZ *et al*., 2008) que los buscadores tienen programas en operación que están constantemente conectándose a los sitios web para descargar dicha información. A estos programas que usan los buscadores web para recolectar páginas y agregarlas a una base de datos se les denomina recolectores de contenido web (MORENO, 2005).

El contenido recolectado también puede utilizarse entre otros fines, para realizar

análisis estadísticos de la Web, validar el código HTML y los enlaces de las páginas, crear réplicas de contenidos y clasificar los contenidos de la Web.

Aún cuando los recolectores examinan la Web periódicamente, el crecimiento de esta imposibilita que se puedan recuperar todas sus páginas. Además, los constantes cambios que sufren muchas de las páginas existentes en la Web hacen que la colección obtenida se vuelva rápidamente obsoleta. Por otra parte el ancho de banda de la red y la capacidad de almacenamiento para los recolectores tienen un límite. Por estas razones, los buscadores solo almacenan una parte de la Web y la mantienen parcialmente actualizada (BAEZA-YATES *et al.*, 2005).

El enfoque automático de la Recolección Masiva de Contenido Web es aplicable solamente a la parte visible de la Web, excluyendo la denominada Web Invisible.

1.4.2.1. Recolectores de contenido web

Un recolector de contenido web es un caso particular de los agentes o robots web, cuya función consiste en recorrer de forma automática y recursiva la estructura de la web para descargar documentos (ROBOTSTXT, 2009). Estos programas también son conocidos como *spiders* (arañas) o *web crawlers* (rastreadores web).

El primer recolector de contenido web fue creado en el Instituto de Tecnología de Massachusetts (MIT por sus siglas en inglés) en 1993, por el estudiante de física Matthew Gray. Este recolector se llamó World Wide Web Wanderer, aunque también se conocía con el nombre de Wanderer y tenía como objetivo recorrer la Web en busca de sitios nuevos. Este recolector se mantuvo haciendo recorridos de forma regular en el período de Junio de 1993 hasta Enero de 1996, siendo la primera herramienta que se utilizó para determinar el crecimiento de la Web (GRAY, 1995).

Los recolectores de contenido web son una parte esencial de los motores de

búsqueda, por ello los detalles sobre sus algoritmos y arquitectura son mantenidos como secretos comerciales. En la mayoría de los casos, al publicarse el diseño de un recolector, se evidencia la falta de detalles que impiden el análisis del trabajo para su posterior reproducción.

El proceso de recolección de contenido web (ver Figura 5) efectuado por los recolectores se describe brevemente en (CAO *et al.*, 2003) en los siguientes pasos:

1) Tomar una lista inicial de URLs para comenzar el recorrido, estas URLs son agregadas a la lista de URLs encontradas.
2) Seleccionar una URL de la lista y descargar la página desde la Web.
3) Extraer los enlaces que contiene la página descargada y agregarlos a la lista de URLs encontradas.
4) Repetir los pasos 2 y 3 hasta que la lista de URLs se encuentre vacía o el proceso de recolección sea detenido.

Cada recolector tiene su propia estrategia para decidir qué páginas visitar. Generalmente parten de una lista inicial de URLs, esta lista inicial se puede obtener de páginas que contienen listas de servidores o listas de los sitios más populares de la Web. Además las URLs pueden haber sido registradas manualmente por los usuarios a través de un formulario o automáticamente por servicios que se dedican a promover páginas o sitios web.

Además de los robots cuyo objetivo es recolectar contenido de la Web, se pueden diferenciar otros tipos de robots web (DELGADO, 1998):

- *Knowbots*: Robots programados para localizar enlaces dirigidos hacia un documento o servidor en particular. Este tipo de robots permiten evaluar el impacto de las distintas aportaciones que engrosan las distintas áreas de conocimiento presentes en la red.
- *Wanderers*: Robots encargados de realizar entre otras tareas, análisis estadísticos

del crecimiento de la Red y registrar la cantidad de servidores.

- *Worms*: Robots encargados de tareas de duplicación (creación de *mirrors*) de directorios FTP para incrementar su utilidad a un número mayor de usuarios.
- *WebAnts*: Conjunto de robots físicamente alejados que cooperan para la consecución de un objetivo común.

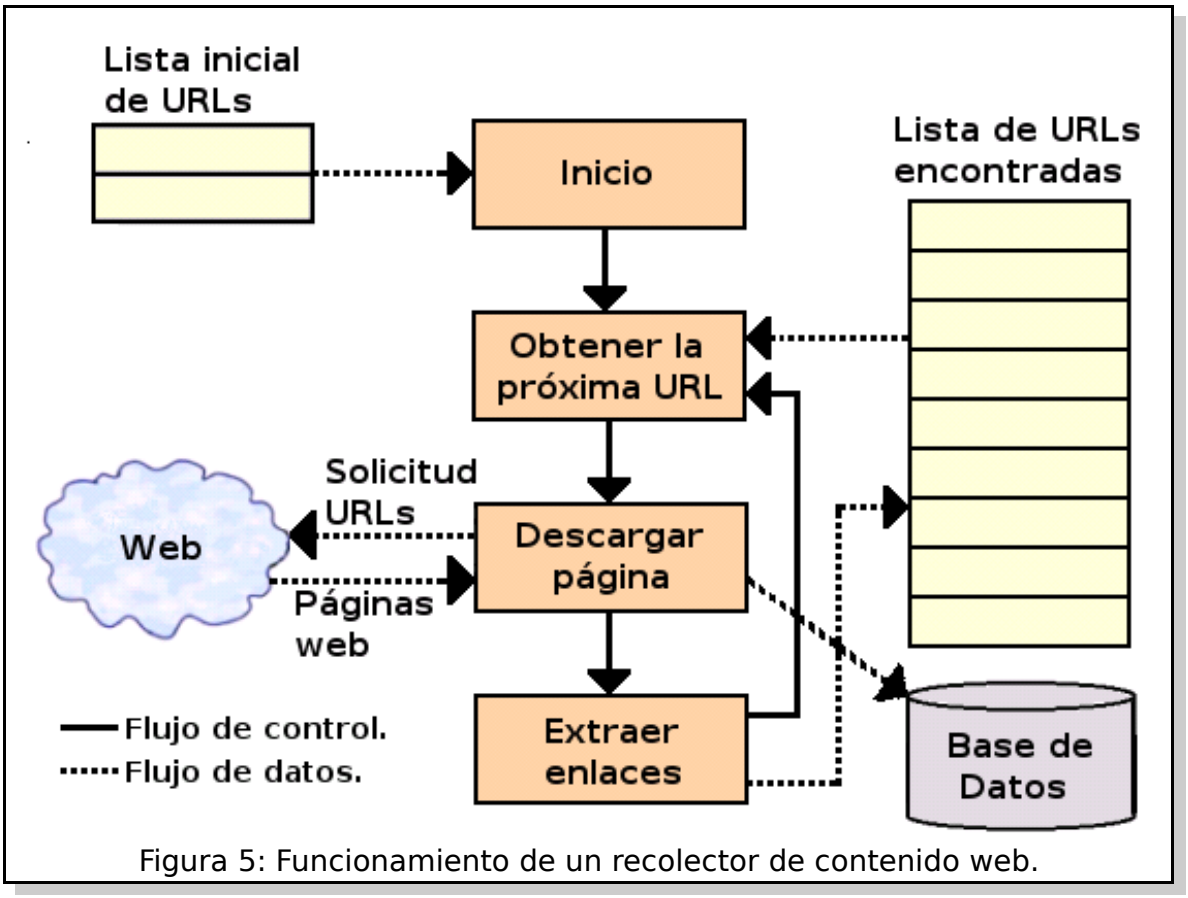


Figura 5: Funcionamiento de un recolector de contenido web.

1.4.2.1.1. Políticas de comportamiento

Los recolectores de contenido web deben actualizar constantemente el contenido almacenado, por lo que visitan las páginas web en busca de modificaciones, al tiempo que encuentran nuevas páginas y desechan las que han desaparecido. También deben aprovechar al máximo el ancho de banda de la red y evitar la sobrecarga en los servidores que visitan. Además, se hace necesario la obtención de las páginas más relevantes aún sin conocimiento previo del contenido de las mismas. Para enfrentar estas contradicciones, los recolectores rigen su comportamiento por la combinación de las siguientes políticas (CASTILLO, 2004):

- **Política de Selección:** se encarga de seleccionar las páginas que serán descargadas. Debido al tamaño de la Web los robots solo pueden visitar y recolectar una parte de ella, la cual debe tener las páginas más relevantes. La importancia de una página se puede medir en función de la calidad de su contenido y la popularidad de la misma en términos de enlaces o visitas.

- **Política de Actualización:** determina cuando una página debe ser chequeada para descubrir cambios en la misma. La naturaleza dinámica de la Web permite que durante el proceso de recolección se creen nuevas páginas, mientras las ya existentes pueden ser modificadas o eliminadas. El objetivo de esta política es mantener actualizado el contenido recolectado.

- **Política de Cortesía:** el Protocolo Estándar para la Exclusión de Robots permite resolver parcialmente la sobrecarga en los servidores, pero este no toma en cuenta la frecuencia con la que un robot puede visitar un mismo sitio, siendo esta una de las formas más comunes de sobrecargar un servidor. Esta política establece que un robot debe esperar algunos segundos para efectuar peticiones repetidas a un mismo servidor. Muchos autores coinciden en que

este intervalo debe oscilar entre 10 y 60 segundos.

- **Política de Paralelismo:** coordina los procesos simultáneos de recolección con el objetivo de maximizar el ritmo de descargas y minimizar la superposición de contenido web. Además, analiza elementos como el consumo de recursos en los ordenadores y el tráfico en la red.

El correcto uso de estas políticas permite que se recuperen las mejores páginas en una fase temprana del proceso de recolección.

1.4.2.1.2. Recolectores en paralelo

A medida que el tamaño de la Web ha ido aumentando, se ha hecho más difícil recuperar una parte representativa de la misma mediante un único proceso de recolección (ver Figura 6). En (CHO y GARCÍA-MOLINA, 2005) se explica que para aumentar el ritmo de descargas de documentos desde la Web es necesario el empleo de múltiples procesos en paralelo. Los recolectores que implementan esta técnica son conocidos como recolectores en paralelo.

Para realizar la recolección en paralelo existen al menos dos variantes: (1) utilizando procesos simultáneos de recolección en un mismo ordenador (ver Figura 7) y (2) utilizando recolectores en diferentes ordenadores que cooperen entre ellos, los cuales pueden estar geográficamente distantes (ver Figura 8).

Uno de los mayores problemas que afronta la segunda variante es la necesidad de comunicación constante entre todos los recolectores, para evitar que descarguen los mismos documentos. Para ello se utilizan técnicas como la asignación estática y la asignación dinámica de URLs. En la primera se le asigna inicialmente a cada recolector una parte de la Web para que sea descargada; mientras que en la segunda

existe un sistema central encargado de asignar dinámicamente diferentes porciones de la Web durante la recolección.

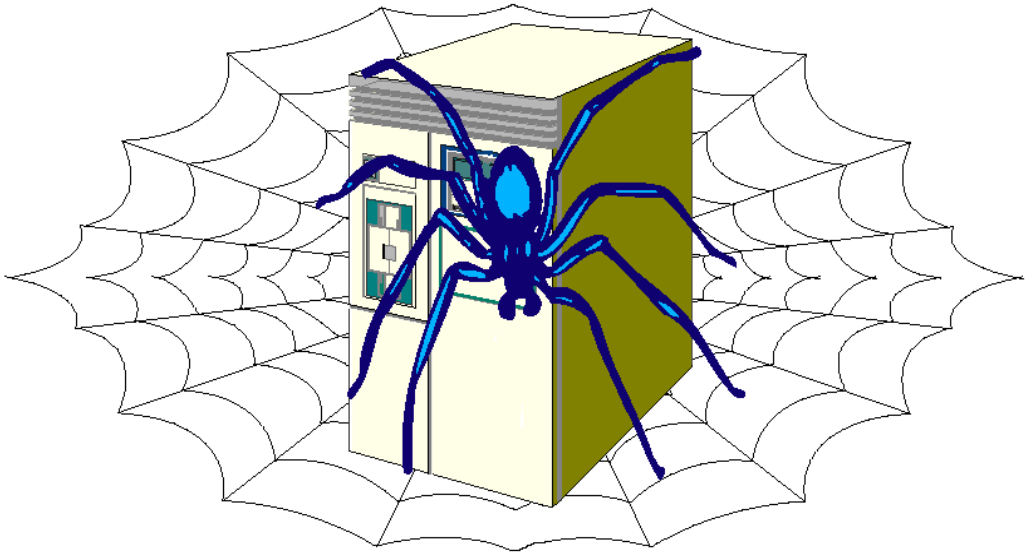


Figura 6: Recolección de contenido web mediante un único proceso.

Entre las ventajas que ofrece la recolección en paralelo de contenido web se encuentran:

- **Escalabilidad:** mayor capacidad del sistema para manejar grandes volúmenes de información.
- **Dispersión de la carga en la red:** la recolección se puede realizar en lugares geográficamente distantes, por lo que la carga de la red estaría dispersa en múltiples regiones.
- **Reducción de la carga en la red:** al realizar la recolección en lugares distantes geográficamente, la carga en la red se reduce solo a la zona donde se efectúa dicho proceso. Además, para trasladar el contenido descargado hacia un lugar centralizado, se utilizan técnicas que minimizan el tráfico en la red.

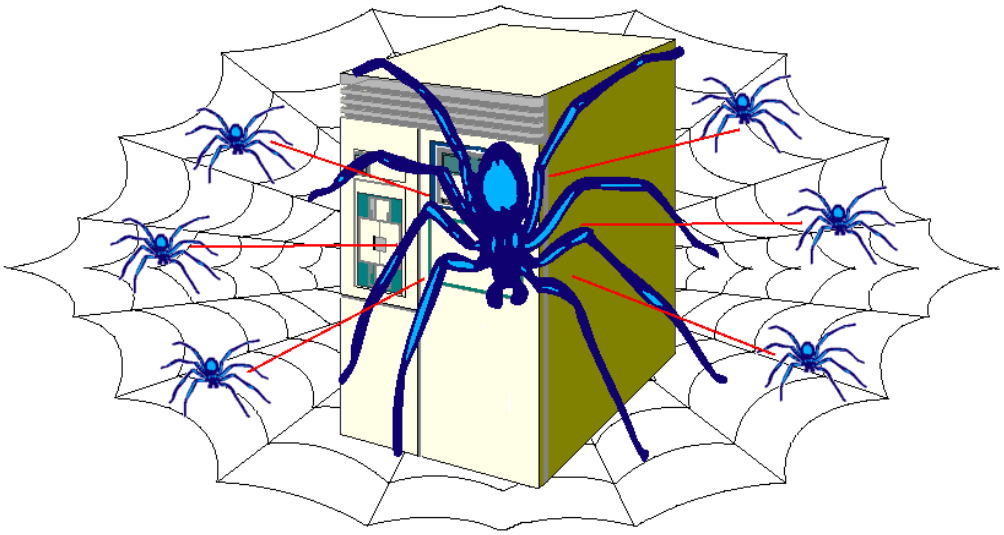


Figura 7: Recolección paralela en un mismo ordenador.

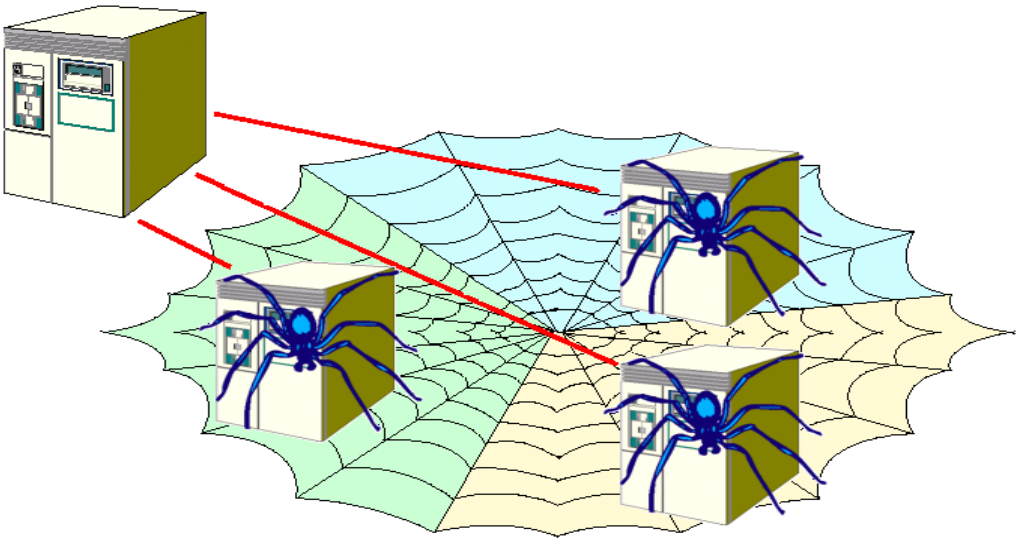


Figura 8: Recolección distribuida en diferentes ordenadores.

1.4.2.1.3. Protocolo Estándar para la Exclusión de Robots

Entre 1993 y 1994 existieron casos en que, debido principalmente a malas implementaciones, algunos robots no eran bienvenidos en los servidores web. Estos eran saturados con peticiones repetidas a los mismos archivos o los robots se desplazaban por lugares que no eran los más adecuados. Es así como se evidencia la necesidad de establecer mecanismos para indicarles a los robots qué parte de los servidores web no debían visitar, dando lugar al surgimiento del Protocolo Estándar para la Exclusión de Robots (KOSTER, 1996).

Este protocolo básicamente consiste en colocar en el servidor un fichero de texto llamado robots.txt. Lo primero que hace un robot al visitar un sitio es comprobar que existe este fichero, en caso afirmativo sigue las instrucciones del mismo, de lo contrario visita el servidor sin restricciones. Haciendo uso de esta técnica se puede restringir o denegar el acceso de un robot a un sitio web.

Otro método utilizado para controlar el acceso de los robots a los sitios web consiste en modificar las etiquetas en las cabeceras de los documentos HTML. Esta técnica es usada para indicarle a los robots si la página visitada puede ser indexada o no, y en caso afirmativo, si se deben seguir sus enlaces.

1.4.2.1.4. Robots web privativos

Estos recolectores brindan servicios a empresas privadas y sus dueños tienen la autoridad para no exponer públicamente ninguna especificación sobre el *software*. Son liberados bajo licencias privativas, restringiendo su uso exclusivamente a sus propietarios. De estos recolectores se desconoce casi todo, incluyendo aspectos importantes como: implementación de sus funcionalidades, algoritmos de búsqueda

o recolección y políticas de comportamiento. Como máximos exponentes en el ámbito internacional se destacan:

- **Googlebot.**
  Es el recolector de contenido web encargado de descargar la información para generar el índice del buscador Google. Está desarrollado sobre plataforma GNU/Linux y en su implementación se utilizó C/C++ como lenguaje de programación. El recolector y su código no se encuentran disponibles para su utilización o estudio. Para su funcionamiento se divide en dos componentes: Deepbot y Freshbot, el primero es un rastreador en profundidad que trata de seguir cualquier enlace en la Web y descargar todas las páginas posibles para la indexación, suele realizar esta tarea una vez al mes; el segundo, por su parte, recorre la Web buscando contenido fresco (por ejemplo los sitios de noticias), para ello visita las páginas que cambian frecuentemente y se ajusta a sus períodos de cambio. Googlebot es el recolector que lleva más tiempo recorriendo la Web, sin embargo al ser propietario sus funcionalidades no pueden ser aprovechadas libremente por otros.

- **Yahoo Slurp.**
  Este recolector es utilizado por el buscador web Yahoo para descubrir y descargar la porción de la Web que mostrará luego en sus índices. Fue comprado a la empresa Inktomi Corporation[8] y se encuentra actualmente en la versión Yahoo Slurp 3.0, está desarrollado sobre plataforma GNU/Linux, se utilizó el lenguaje de programación C/C++ para su implementación y cumple con el Protocolo Estándar de Exclusión de Robot. La documentación acerca del funcionamiento del robot es escasa y su uso es propiedad privada, haciendo que no se tenga acceso al producto o su código.

8 Compañía proveedora de *software* adquirida por Yahoo.

1.4.2.1.5. Robots web libres

Son aquellos que se liberan bajo una licencia que permita las libertades de ejecutar, estudiar, redistribuir y modificar el programa sin necesidad de pedir permiso o pagar por ello. De estos recolectores existe abundante documentación y se puede tener acceso tanto al programa como a su código fuente. Como máximos exponentes en el ámbito nacional e internacional se destacan:

- **WIRE**[9].
  El Entorno de Recuperación de Información Web (WIRE por sus siglas en inglés), fue desarrollado en el Centro de Investigaciones de la Web[10] en Chile. Constituye una suite de herramientas concebidas con el objetivo de realizar reportes sobre estudios estadísticos de la Web y diseñadas para trabajar con grandes volúmenes de información, han sido probadas con millones de documentos durante los estudios realizados a la Web de Chile, siendo además altamente configurables. En un recorrido experimental con duración de ocho horas efectuado en Diciembre de 2008 sobre la Web de la UCI, se detectaron 68.525 URLs de las cuales se descargaron exitosamente 49.342, las restantes 19.183 retornaron algún código de error al intentar descargarlas, dando como resultado un promedio de 6.167 documentos descargados por hora. WIRE se encuentra disponible en la Web bajo la licencia GPL[11], cuenta con abundante documentación acerca del producto, el lenguaje de programación usado para desarrollar la mayor parte de la aplicación fue C, aunque también se usó C++, empleando en su desarrollo aproximadamente 25.000 líneas de código. Además, respeta el Protocolo Estándar para la Exclusión de Robots y puede

9 http://www.cwr.cl/projects/WIRE/index.html
10 http://www.cwr.cl
11 http://www.gnu.org/copyleft/gpl.html

manejar varios procesos en paralelo sobre un mismo ordenador, funciona sobre una plataforma GNU/Linux y para ello requiere de las librerías adns[12] y libxml2[13], así como del motor de indexación swish-e[14]. Su última versión disponible es la v0.21 publicada el 28 de Noviembre de 2008.

- **Larbin[15].**
  Fue desarrollado en Francia por el equipo Verso del proyecto XYLEME perteneciente al INRIA[16], con la intención de recolectar páginas en formato XML que serían utilizadas en la base de datos de un buscador web orientado al mismo formato, aunque posteriormente se modificó para que descargara la mayoría de los formatos existentes en la Web. Se encuentra disponible bajo la licencia GPL, cuenta con abundante documentación acerca del producto y cada una de sus funcionalidades, es capaz de recolectar 5.000.000 de documentos diarios en un ordenador estándar, aunque este rendimiento depende principalmente de las características de la red, además de ser altamente configurable. En un recorrido experimental con duración de cuatro horas efectuado en Febrero de 2009 sobre la Web de la UCI, se detectaron 106.656 URLs de las cuales 98.824 pertenecían a documentos HTML, de estos se descargaron exitosamente 61.424, se detectaron 23.146 páginas web duplicadas, un total de 14.254 páginas no fueron descargadas por retornar algún código de error o no cumplir con las restricciones establecidas en los parámetros de configuración para dicho recorrido, dando como resultado un promedio de 15.356 documentos descargados por hora. Aunque Larbin es capaz de descargar la mayoría de los formatos existentes en la Web, para este recorrido se descargaron solo los documentos HTML. Además, Larbin no es solo

12 http://www.chiark.greenend.org.uk/~ian/adns
13 http://www.libxml.org
14 http://swish-e.org
15 http://larbin.sourceforge.net/index-eng.html
16 http://www.inria.fr

un producto final, sino una plataforma de desarrollo para crear recolectores personalizados. Para su implementación se utilizó como lenguaje de programación C/C++, funciona sobre una plataforma GNU/Linux y requiere de la librería estándar adns en su versión v1.4, puede gestionar varios procesos de recolección en paralelo en un mismo ordenador y respeta el Protocolo Estándar para la Exclusión de Robots. Su última versión disponible es la v2.6.3 publicada el 9 de Julio de 2003.

- **Guglebot.**
  Fue desarrollado en la UCI en el año 2006 con el objetivo de recolectar información para el buscador web Gugle. A pesar de ser liberado bajo la licencia GPL existe poca documentación acerca del producto y sus funcionalidades, debido a la interrupción del proyecto en el propio año. A la versión inicial Guglebot v0.1 (versión alfa) se le realizaron varias pruebas con duración de 11, 70, 180 y 600 minutos respectivamente, estas tenían como objetivo analizar el rendimiento del software y demostrar que era falsa la conjetura existente en aquel entonces de que la cantidad de páginas de la UCI era inferior a 20.000, lo cual quedó demostrado en el primer experimento, pues solo en 11 minutos Guglebot detectó 20.823 URLs. Posterior a estas pruebas iniciales se realizaron otros tres recorridos, con duración de una hora los dos primeros y tres horas el último, en el primero se detectaron 72.149 URLs y se desconoce cuántas páginas fueron descargadas; en el segundo recorrido se detectaron 71.101 URLs, de las cuales se descargaron exitosamente 20.498 páginas y 709 retornaron algún error cuando se intentó descargarlas, en el último recorrido se detectaron 128.638 URLs, se descargaron exitosamente un total de 52.481 páginas y retornaron error 759, evitando la descarga de las mismas, el promedio de páginas descargadas por hora fue de 17.614. Para el desarrollo de Guglebot se utilizó como lenguaje de programación Python[17] y

17 http://www.python.org

funciona sobre una plataforma GNU/Linux. Su última versión conocida es la v0.1 (versión alfa).

1.4.2.1.6. Características ideales

Teniendo en cuenta las características peculiares de la Web, los problemas inherentes a la recolección de su contenido y las necesidades de información de los usuarios, se exponen algunas de las características que deben estar presentes en los recolectores de contenido web para mejorar su calidad y eficiencia:

- implementados con un lenguaje potente de forma tal que pueda aprovechar al máximo los recursos de hardware que utiliza.
- diseñados para trabajar con grandes volúmenes de documentos.
- deben permitir cambiar la mayoría de los parámetros para los distintos procesos de recolección, de acuerdo a las necesidades específicas de los clientes.
- deben ser capaces de realizar la recolección en paralelo manejando varios procesos al mismo tiempo, entre los que debe existir una comunicación eficiente.
- deben respetar el Protocolo Estándar para la Exclusión de Robots.
- deben estar disponibles bajo una licencia que permita las libertades del software libre, además de contar con suficiente documentación que facilite su reutilización.

1.5. Necesidades de información de PPSINI

El Polo Productivo de Soluciones Informáticas para Internet (PPSINI), perteneciente a la Facultad 10 de la UCI, tiene como misión fundamental producir conocimientos,

productos y servicios informáticos para Internet que tributen a la informatización de la sociedad y al desarrollo económico del país; mediante la integración de los procesos de formación, investigación, producción y comercialización de la Universidad. Dentro del polo se identifican las siguientes líneas de investigación:

- Seguridad en Internet.
- Inteligencia Artificial Aplicada.
- Recuperación de Información.
- Cibermetría Aplicada.
- Matemática-Estadística Aplicada.
- Ingeniería y Gestión de Software.
- Base de Datos.

Por otra parte, las líneas de desarrollo de PPSINI se basan en crear conocimientos, productos y servicios informáticos acerca de:

- Seguridad en Internet.
- Inteligencia Artificial Aplicada.
- Recuperación de Información.
- Cibermetría Aplicada.

Para dar cumplimiento a los objetivos de estas líneas, el polo se divide en cuatro grupos de investigación y desarrollo, que a su vez se fraccionan en proyectos productivos, de los cuales, los mencionados a continuación, tienen como objetivo principal crear productos informáticos que requieren procesar grandes volúmenes de contenido web.

- **Motor de Clasificación Inteligente de Contenidos (MOCIC)[18].**
  Su objetivo general es desarrollar una herramienta informática que permita, de forma automatizada e inteligente, clasificar diferentes tipos de contenido. La diversidad de información en la Web, su facilidad de publicación y la ausencia de un proceso de selección y clasificación de la misma, ha implicado que esta aparezca mínimamente estructurada y abarque todos los temas posibles, incluyendo contenidos ilícitos[19] o nocivos[20] que atentan contra la integridad de los usuarios. El constante crecimiento de la Web hace necesario que se clasifique la mayor cantidad posible de documentos.

- **Generador de Estudios Webmétricos (GEWEB)[21].**
  Su objetivo general es desarrollar un producto informático que permita realizar estudios webmétricos para asistir en la toma de decisiones respecto a la Web analizada. Para realizar estos estudios es necesario procesar las características de una parte representativa de la Web. La veracidad de los resultados obtenidos depende del tamaño y nivel de detalle de la porción analizada de la Web.

- **Buscador Web (BUSCAWEB)[22].**
  Su objetivo general es desarrollar un producto informático que indexe archivos almacenados en servidores web para que los usuarios, a través de consultas, satisfagan sus necesidades de información. Para responder con éxito a dichas consultas, se precisa almacenar en la base de datos del buscador la mayor cantidad de información disponible en la Web, así como mantener actualizada dicha base de datos.

18 http://gforge.f10.uci.cu/projects/mocic/
19 merecen una respuesta penal por quebrantar un bien jurídico (Ej: pornografía infantil y terrorismo).
20 no son lo suficientemente ofensivos como para merecer una respuesta penal (Ej: pornografía).
21 http://gforge.f10.uci.cu/projects/geweb/
22 http://gforge.f10.uci.cu/projects/buscaweb/

1.6. Herramientas a utilizar

Para lograr un desempeño óptimo en un sistema informático es vital la correcta selección de las herramientas que se utilizarán en su desarrollo. En el presente trabajo, para seleccionar las herramientas a utilizar se tuvo en cuenta la calidad de las prestaciones que ofrecen y la aceptación por parte de los desarrolladores en cuanto a conocimiento, dominio y popularidad de las mismas.

- **Sistema Operativo Debian GNU/Linux[23].**
  Este sistema es libre y dispone de más de 25.000 paquetes distribuidos de forma gratuita. Es una de las distribuciones de GNU/Linux más utilizadas en el mundo debido a la gran cantidad de colaboradores que posee, los cuales muestran calidad y estabilidad en su trabajo, dedicación al Contrato Social de Debian y un fuerte compromiso de ofrecer el mejor sistema operativo posible. Su última versión estable es la v5.0, también identificada con el nombre código Lenny, publicada el 14 de Febrero de 2009.

- **Entorno de Desarrollo Integrado Anjuta[24].**
  Es un proyecto libre creado para los lenguajes de programación C y C++, aunque también soporta otros lenguajes como Perl y Python, ofrece muchas características avanzadas de programación que incluyen un administrador de proyectos, asistentes, plantillas, depurador interactivo y un poderoso editor que verifica y resalta la sintaxis escrita. Su última versión estable es la v2.24.2 publicada el 25 de Noviembre de 2008.

23 http://www.debian.org
24 http://www.anjuta.org

- **Quanta Plus[25].**
  Es una herramienta libre utilizada en el desarrollo de páginas web que rápidamente se ha convertido en un editor con eficientes funcionalidades. Cuenta con una interfaz de múltiples documentos de mucha utilidad para los desarrolladores, además, puede incrementar la productividad a través del uso de acciones personalizadas, guiones y barras de herramientas, con lo que se puede automatizar casi cualquier tarea. Su ultima versión estable es la v3.5.9, liberada el 20 de Febrero de 2008.

- **CodeIgniter[26].**
  Es un poderoso framework para PHP desarrollado por Rick Ellis, director ejecutivo de Ellislab, Inc. El mismo resalta características como el bajo consumo de recursos del sistema, su rendimiento excepcional, es totalmente extensible y altamente compatible con gran variedad de versiones y configuraciones de PHP. Además, existe abundante documentación y posee una curva de aprendizaje muy corta. La última versión estable es la v1.7.1 publicada el 10 de Febrero de 2009.

- **Lenguaje de programación Perl[27].**
  Es un lenguaje de propósito general, originalmente desarrollado para el procesamiento de textos aunque en la actualidad es utilizado en diversos fines: administración de sistemas, desarrollo web, programación en red y desarrollo de interfaces gráficas, por solo citar algunos. Debe gran parte de su popularidad al hecho de que se trata de un lenguaje pseudo-compilado que se distribuye de forma gratuita, capaz de ejecutarse en cualquier plataforma que tenga un intérprete disponible, de fácil uso, eficiente y muy completo. El 26 de

25 http://quanta.kdewebdev.org
26 http://www.codeingniter.com
27 http://www.perl.org

Octubre de 1995 se creó la CPAN[28]: colección de sitios web que almacenan y distribuyen fuentes en Perl, binarios, documentación, scripts y módulos. Características importantes y algunas construcciones esenciales han sido añadidas a Perl, incluyendo un soporte importante para la Programación Orientada a Objetos. Su última versión estable es la v5.10.0, publicada el 18 de Diciembre de 2007.

- **Lenguaje de programación C/C++.**
  C surge como evolución del anterior lenguaje B, a su vez basado en BCPL. Está orientado a la implementación de sistemas operativos, concretamente Unix. Es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones. Es altamente eficiente ya que es posible utilizar sus características de bajo nivel para realizar implementaciones óptimas. De este lenguaje existen compiladores para casi todos los sistemas conocidos. C++ se creó para extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos, manteniendo una considerable potencia para la programación a bajo nivel, pero añadiendo elementos que le permiten también un estilo de programación con alto nivel de abstracción. Es un lenguaje muy popular y potente utilizado para el desarrollo de aplicaciones robustas y eficientes.

- **PHP[29].**
  Es un lenguaje script interpretado en el lado del servidor utilizado para la generación de páginas web dinámicas. La mayor parte de su sintaxis fue tomada de C, Java[30] y Perl, incorporando algunas cualidades propias. Es un

28 http://www.cpan.org
29 http://ww.php.net
30 http://www.java.sun.com

lenguaje multiplataforma que posee abundante documentación y permite la utilización de técnicas de Programación Orientada a Objetos. Es compatible con la mayoría de los sistemas gestores de bases de datos y permite la integración con bibliotecas externas para fines diversos. Es un producto de código abierto respaldado por una gran comunidad de desarrolladores que mejoran y amplían continuamente las capacidades del lenguaje. Su última versión estable es la v5.2.9 publicada el 26 de Febrero de 2009.

- **PostgreSql**[31]**.**
  Es un potente sistema gestor de bases de datos cuyo desarrollo está dirigido por una comunidad que, uniendo esfuerzos, ha hecho de este el sistema de base de datos de código abierto más avanzado y robusto del mundo. Soporta la mayoría de las transacciones SQL, posee control concurrente, puede ser ejecutado en casi todos los sistemas operativos, está completamente documentado y brinda soporte para los lenguajes más populares (C/C++, Java, Perl, Python y PHP). Su última versión estable es la v8.3, publicada el 4 de Febrero de 2008 bajo la Licencia BSD[32].

- **FreeNAS**[33]**.**
  Es un sistema operativo gratuito, basado en FreeBSD[34] que proporciona servicios de almacenamiento en red. Fue creado con el objetivo de simplificar la administración y mantenimiento de los servidores de archivos, para lo cual se basa en su gran escalabilidad, confiabilidad y disponibilidad, así como su buen funcionamiento. Es de fácil uso y proporciona datos heterogéneos, permitiendo automatizar y simplificar el mantenimiento de los datos. La última

31 http://www.postgresql.org
32 http://www.freebsd.org/copyright/
33 http://www.freenas.org
34 http://www.freebsd.org

versión estable es la v0.69 liberada el 17 de Enero de 2009.

1.7. Metodología de desarrollo

Para obtener mejores resultados en la organización y planificación del trabajo, así como un producto informático con calidad, que satisfaga tanto a clientes como a desarrolladores, es preciso usar una metodología de desarrollo de software.

La selección de una metodología de desarrollo implica un proceso de análisis de las necesidades y solicitudes de los clientes, las cualidades del producto y el control del proceso de desarrollo en cuestión. Además, se debe tener en cuenta el grado de implicación de los clientes en el proceso de desarrollo y la magnitud del mismo, así como la comunicación y colaboración entre los desarrolladores.

Las metodologías ágiles (CANÓS et al., 2007) forman parte del movimiento de desarrollo ágil de software. Se fundamentan en la adaptabilidad de los procesos de desarrollo para aumentar las probabilidades de éxito de un proyecto, en la simplicidad de sus reglas y prácticas, en la orientación a equipos de desarrollo de pequeño tamaño, la flexibilidad ante los cambios y su ideología de colaboración. Intentan evitar los burocráticos procesos de las metodologías tradicionales. Se basan además en los siguientes principios:

- realizar entregas continuas y funcionales en corto período de tiempo.
- dar la bienvenida a los cambios.
- los clientes forman parte del equipo de desarrollo.
- proporcionar el ambiente, el apoyo y la confianza suficiente para mantener motivados a los integrantes del equipo de desarrollo.
- la comunicación directa es el método más eficiente y efectivo para informar al equipo de desarrollo, intentando evitar otros medios como el teléfono, correo

electrónico y fax, por solo citar algunos.

- la medida principal de progreso es el funcionamiento del software.
- desarrollo sostenible, siendo indispensable la existencia de paz y armonía para el éxito del proyecto.
- buen diseño y calidad técnica.
- equipos auto-organizados que periódicamente se plantean ser más efectivos.

1.7.1. Metodología ágil SXP

Es una metodología compuesta por SCRUM y XP que ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan mejorar la actividad productiva, fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo y ayudando al líder del proyecto a ejercer mejor control sobre el mismo.

SCRUM se encarga de gestionar un equipo para que trabaje de forma eficiente, así como de medir siempre los procesos. XP es una metodología encaminada al desarrollo, consiste en una programación rápida o extrema cuya particularidad es tener como parte del equipo al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. Consta de cuatro fases principales: (1) Definición-Planificación, es donde se establece la visión, se fijan las expectativas y asegura el financiamiento del proyecto, (2) Desarrollo, se realiza la implementación del sistema y se obtiene un producto listo para ser entregado, (3) Entrega, se transfiere el producto al cliente y (4) Mantenimiento, donde se realiza el soporte para el cliente. En cada una de estas fases se realizan numerosas actividades entre las que se destacan: levantamiento de requisitos, priorización de la Lista de Reserva del Producto, definición de las Historias de Usuarios, implementación y prueba; durante estas actividades se generan los artefactos necesarios para documentar todo el proceso. Las entregas son frecuentes y existe una refactorización continua, lo que permite mejorar el diseño cada vez que

se añade una nueva funcionalidad.

SXP está especialmente indicada para proyectos de pequeños equipos de trabajo, con requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta una entrega rápida de resultados y una alta flexibilidad. Ayuda a que el equipo trabaje unido, en la misma dirección y con un objetivo claro. Permite además seguir de forma clara el avance de las tareas a realizar, ayudando a los líderes a ver día a día como funciona el trabajo (PEÑALVER, 2008).

1.8. Investigaciones similares

Se consideran investigaciones similares aquellas que fueron realizadas con el objetivo de resolver el problema que origina este trabajo; es decir, las que se plantearon la necesidad de investigar cómo automatizar el proceso de recolección de contenido web para diversos fines. En el ámbito internacional destacan, entre otros, los siguientes desarrollos:

- **Effective Web Crawling.**
  Investigación realizada en la Universidad de Chile[35] por Carlos Castillo en el año 2004 para optar por el grado de Ph.D. en Ciencias de la Computación. El autor analiza la Recolección Masiva de Contenido Web en diferentes niveles e incluye prácticas de implementación que son esenciales en este proceso. Presenta el modelo y la arquitectura de un robot web que integra el recolector con un motor de búsqueda, facilitando el acceso a los metadatos y a los enlaces de los documentos HTML, este modelo es implementado en el WIRE, recolector que proporciona un marco experimental para esta investigación. Propone el estudio y la implementación de algoritmos para lograr la recuperación de las páginas

35 htttp://www.uchile.cl

más importantes en una fase temprana del proceso de recolección. Además, modela y estudia el comportamiento de los usuarios en los sitios web, concluyendo que no es necesario ir más allá de cinco niveles para recuperar las páginas que realmente son visitadas por los usuarios. Por último, propone varios mecanismos para la cooperación con los servidores, con el objetivo de reducir el tráfico en la red.

- **Effective Web Crawlers.**
  Investigación realizada en la Universidad de Melbourne[36] en Australia por Halil Ali en el año 2008 para optar por el grado de Ph.D. en Ciencias de la Computación. El autor expone el problema de la inconsistencia en los resultados mostrados a un usuario, mediante un buscador web, cuando los documentos indexados se han modificado en la Web después de haber sido recolectados. Para mejorar el índice de coherencia entre los documentos existentes en la Web y los que se muestran a un usuario cuando efectúa una consulta a un buscador presenta cinco enfoques específicos, los cuales son: (1) detectar un cambio significativo, (2) medir el impacto de la recolección sobre una colección fresca desde la perspectiva del usuario, (3) desarrollar un framework para evaluar el rendimiento del recolector, (4) determinar la eficacia de varios sistemas de recolección y (5) la propuesta y evaluación de la efectividad de un enfoque dinámico de recolección. Presenta además un novedoso enfoque basado en estadísticas obtenidas durante la recolección que mantiene actualizada una colección de documentos, utilizando el texto de los enlaces que los une para determinar su probabilidad de cambio. Por último, presenta el diseño y la implementación de LARA, un recolector distribuido que brinda un marco experimental para esta investigación, el cual se utiliza en pruebas cuyos resultados son utilizados como retroalimentación.

36 http://www.rmit.edu.au

En el ámbito nacional, no existen o se desconocen investigaciones similares que sustenten el desarrollo de sistemas afines. No obstante, se han desarrollado programas para recolectar contenido web, como es el caso de Guglebot en la UCI. Otro ejemplo es Carlota: software que actualmente se desarrolla en la Oficina para la Informatización[37] de Cuba con el objetivo de mantener actualizada la base de datos del buscador web Dos por Tres[38]. Este sistema ha sido descartado por hallarse en una fase prematura de su desarrollo, además de no haberse definido la licencia bajo la cual será liberado el producto.

1.9. Conclusiones

Después de estudiar las principales investigaciones similares efectuadas y los inconvenientes de la recolección manual de contenido web, se evidencia la necesidad de efectuar la recolección de dicho contenido mediante un sistema automatizado. Para ello, luego de analizar las características ideales que deben tener los sistemas recolectores de contenido web, las propiedades de los principales recolectores libres en el ámbito nacional e internacional y los resultados obtenidos en los recorridos efectuados con cada uno de ellos, se decide utilizar Larbin como plataforma para el desarrollo de SINIBOT, obteniendo como resultado una versión mejorada de dicho sistema, capaz de satisfacer las necesidades de información de MOCIC, GEWEB y FARO, así como de futuros productos que requieran procesar grandes volúmenes de contenido web.

Tras abordar aspectos de las metodologías ágiles y de las ventajas que brinda la utilización de las mismas, se considera apropiado usar la metodología SXP para guiar el desarrollo de SINIBOT.

37 http://www.infosoc.cu
38 http://www.2x3.cu

Con la realización del presente capítulo se ha definido la posición de los autores con respecto a la teoría existente sobre la Recolección Masiva de Contenido Web y se materializó la fundamentación teórica que servirá de sustento para la toma de decisiones en los capítulos siguientes.

2. CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1. Introducción

Para guiar el desarrollo de SINIBOT es necesario comprender su contexto e identificar las condiciones o capacidades que debe cumplir. En este capítulo se presentará el Modelo del Dominio con el objetivo de comprender el contexto del sistema. Se realizará la Descripción de la Propuesta del Sistema, describiendo cómo debe funcionar y destacando sus características distintivas. Se mostrará la Lista de Reserva del Producto, especificando sus Requisitos Funcionales y No Funcionales. Se elaborarán las Historias de Usuarios y el Diagrama de Componentes, donde se muestran los paquetes a utilizar y las relaciones entre ellos.

2.2. Modelo del Dominio

Para el planeamiento del presente sistema se realizó un modelado del dominio. El objetivo de este proceso consiste en capturar, comprender y describir los elementos más importantes dentro del contexto del sistema, así como las relaciones existentes entre ellos (ver Figura 9). El Modelo del Dominio junto al Diccionario del Dominio, ayudan a los usuarios, clientes, desarrolladores y otros interesados a utilizar un vocabulario común.

A continuación se muestra el Diccionario del Dominio, el cual sirve como un glosario de términos en el que se describen textualmente los elementos identificados durante el modelado del dominio del problema.

- URL: representa la dirección en la Web de un único Documento.

- Documento: archivo disponible en la Web identificado por una URL y puede representar, entre otros formatos, una imagen, un archivo multimedia, un archivo de texto o una Página Web.
- Página Web: archivo de texto codificado en lenguaje HTML, puede contener enlaces a otros documentos por medio de sus respectivas URLs.
- Lista de URLs de Partida: contiene una o varias URLs que son utilizadas por el Recolector para comenzar un recorrido.
- Lista de URLs Encontradas: representa las URLs detectadas por el Recolector al extraer los enlaces de las Páginas Web que visita durante el recorrido.
- Recolector: dada una Lista de URLs de Partida comienza un recorrido en el que detecta nuevas URLs, las cuales son agregadas a la Lista de URLs Encontradas para ser visitadas posteriormente. Al visitar una URL se obtiene el Documento en cuestión.

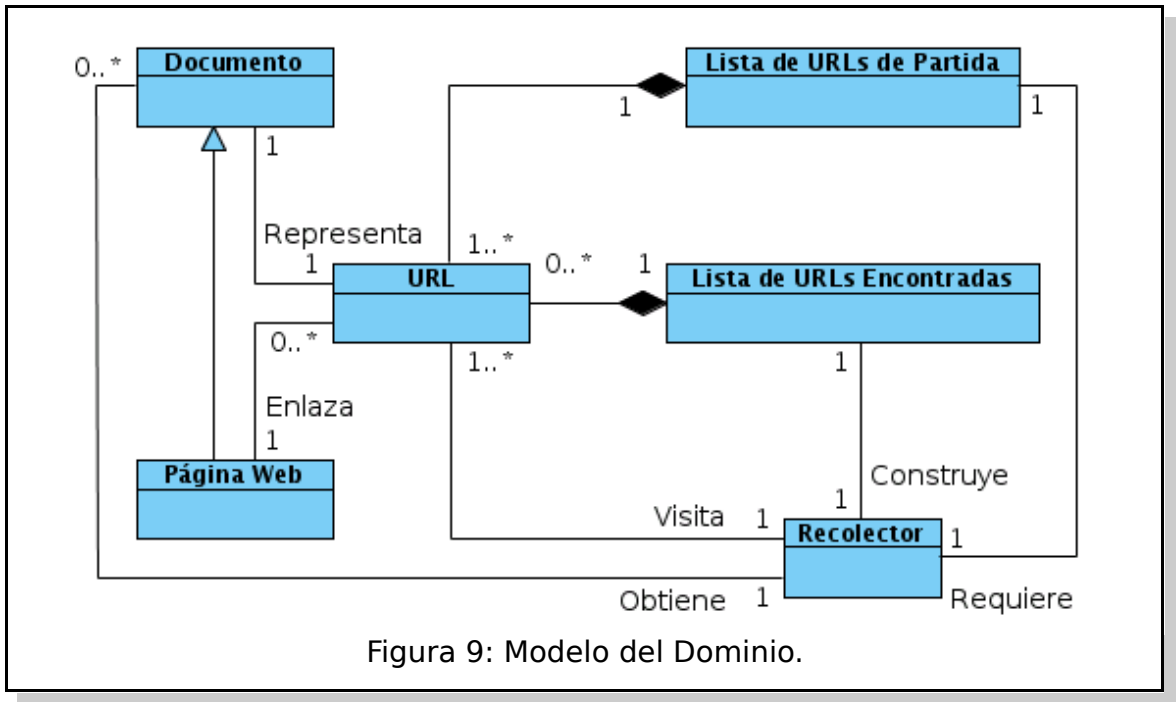


Figura 9: Modelo del Dominio.

En sentido general, al comenzar el proceso de recolección el Recolector obtiene la Lista de URLs de Partida, visita cada URL y obtiene el Documento representado por la misma, si el Documento es una Página Web y enlaza a otros Documentos, las URLs que representan dichos enlaces son extraídas y agregadas a la Lista de URLs Encontradas, la cual se construye durante el proceso de recolección y contiene además las URLs de la lista de partida.

Una vez realizado el modelado del dominio se comprende el contexto de SINIBOT, lo cual posibilita estar en condiciones de hacer la propuesta del sistema y ayuda a determinar los requisitos que se desprenden de dicho contexto.

2.3. Descripción de la Propuesta del Sistema

Larbin es un recolector de contenido web que, a modo general, presenta las siguientes funcionalidades: (1) recorre de forma automática y recursiva la estructura de la Web a partir de una lista inicial de URLs, (2) descarga el documento visitado y si este es una página web procede a extraer sus enlaces, los cuales adiciona a la lista de URLs encontradas, (3) si un documento no puede ser descargado exitosamente entonces se le asigna a la URL un código de error que indica la naturaleza del problema, (4) realiza recorridos mediante procesos en paralelo sobre un mismo ordenador y (5) durante el proceso de recolección, genera una página web en la cual se muestran estadísticas correspondientes al recorrido y a las URLs que visita.

La instalación de Larbin se realiza paso a paso, de forma manual por el usuario, debiendo este gestionar las librerías y dependencias necesarias para que se instale con éxito, ejemplo de ello es que la librería adns que se incluye en el paquete de la versión 2.6.3 de Larbin está obsoleta, debiendo ser sustituida por la versión 1.4 de la misma librería.

El contenido web recolectado se almacena en un directorio, siguiendo un criterio muy pobre en cuanto a organización, esto imposibilita la eficiente gestión de la colección obtenida. Además, dicha colección será sobrescrita si no es replicada manualmente antes del próximo recorrido de Larbin, perdiendo así parte del contenido recolectado en el último recorrido.

La modificación de las opciones de configuración para comenzar los recorridos, así como la introducción de la lista de URLs iniciales, se realiza escribiendo directamente los parámetros deseados sobre los ficheros de configuración. Esto hace necesario que el usuario deba tener total dominio de la localización de los ficheros que debe modificar dentro de la estructura de directorios del paquete de Larbin.

Por otra parte, no existe un mecanismo para programar los horarios de inicio y fin de los recorridos, debiendo el usuario introducir los comandos para iniciar y detener los procesos de recolección de forma manual y en tiempo real.

Se propone modificar y agregar las funcionalidades necesarias para que la aplicación cumpla con los requerimientos previstos por los clientes, aumentando la calidad de las prestaciones del producto final. Cuando el proceso de desarrollo de la versión v0.1 de SINIBOT culmine, la aplicación tendrá integrada una interfaz web que permita administrar el recolector de forma eficiente; el contenido recolectado será almacenado en una base de datos, facilitando la gestión de la colección obtenida en cada recorrido y la aplicación incluirá además un instalador automatizado con interfaz de consola.

La interfaz web de administración contará con el Módulo de Gestión de Usuarios, en el cual se realizarán las acciones básicas concernientes a estos como son: autenticar, adicionar, editar y eliminar usuarios. Se establecerán los roles de Administrador y Consultor, donde el primero tendrá acceso a todas las funcionalidades disponibles desde la interfaz, mientras que el segundo solo podrá, desde el Módulo de Gestión de

Información, consultar la información referente a los recorridos efectuados. Dicha interfaz también presentará el Módulo de Gestión de Recorridos, a través del cual, el Administrador del sistema podrá iniciar un nuevo recorrido, pudiendo fácilmente insertar la lista de URLs iniciales y modificar los parámetros de configuración que utilizará SINIBOT en su versión v0.1. Además, se podrán programar los horarios de inicio y fin de los recorridos, así como detenerlos en caso de ser necesario.

El instalador se encargará de gestionar librerías, dependencias, componentes y permisos para el correcto funcionamiento de los subsistemas de Almacenamiento, Recolección y Administración Web de SINIBOT.

La base de datos está diseñada para almacenar de cada recorrido la lista de URLs iniciales, las opciones de configuración, la fecha y hora de inicio y de terminación, las estadísticas correspondientes a las páginas visitadas y las principales características del hardware sobre el cual se ejecuta la aplicación. De los documentos recolectados se almacenarán los siguientes datos: contenido de los mismos, cabecera HTTP, lista de enlaces a otros documentos si es una página HTML y la fecha y hora en que se recolectó, si el documento tratado no es descargado exitosamente se registra el código asociado al error. Con el fin de simplificar el mantenimiento de los datos, la información referente al contenido de los documentos recolectados será almacenada utilizando la tecnología FreeNAS.

2.4. Modelo de Datos

A continuación se presenta el Modelo de Datos empleado por SINIBOT en su versión v0.1. El mismo se muestra en dos imágenes que representan, en la base de datos, las tablas utilizadas por los módulos de Gestión de Información (ver Figura 10) y Gestión de Usuarios (ver Figura 11).

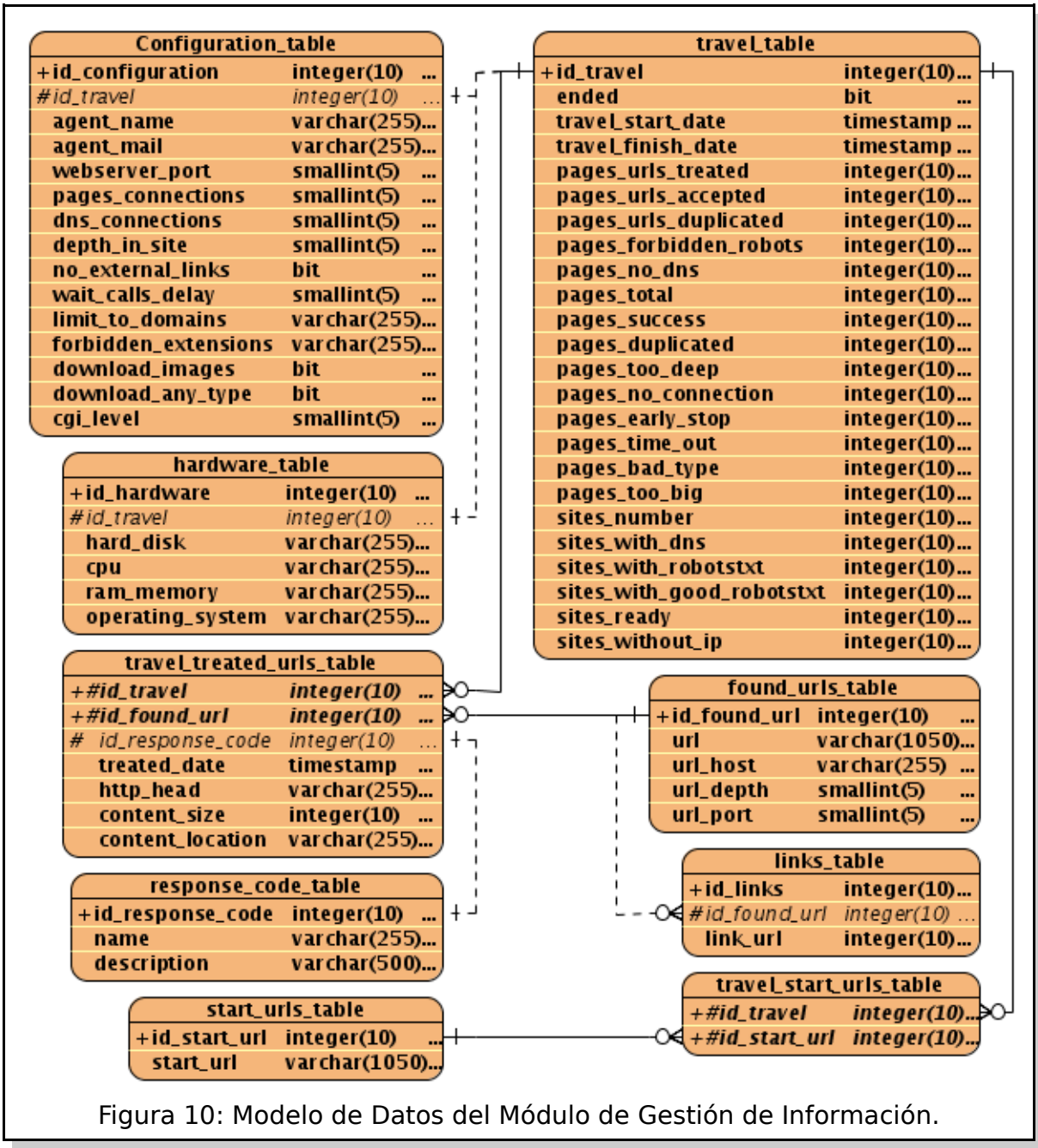


Figura 10: Modelo de Datos del Módulo de Gestión de Información.

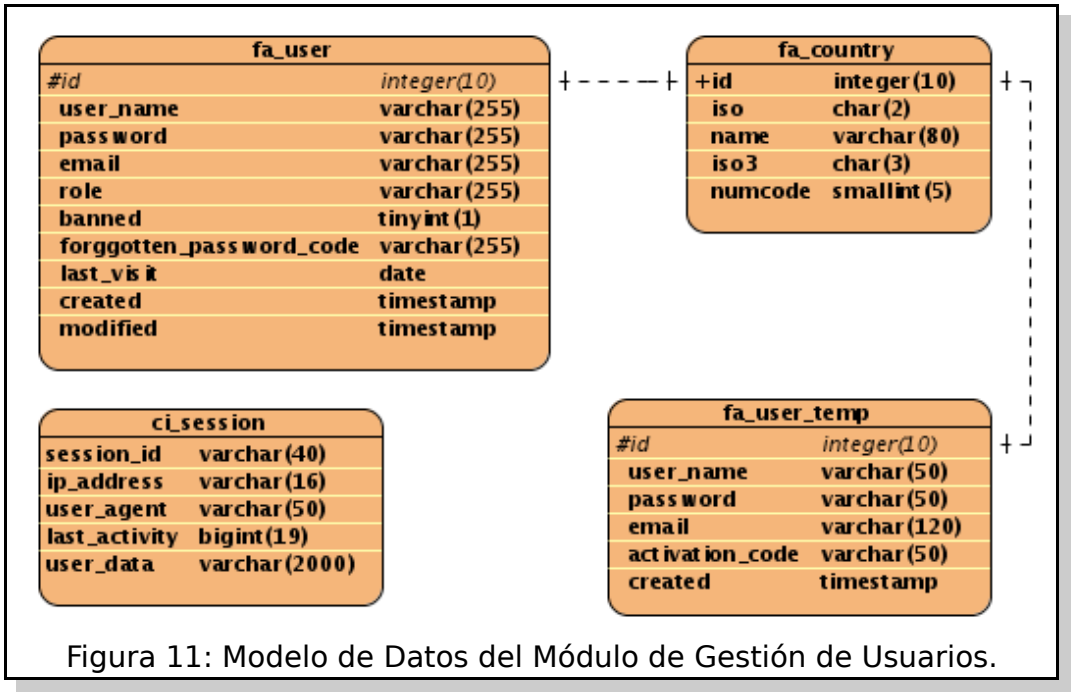


Figura 11: Modelo de Datos del Módulo de Gestión de Usuarios.

2.5. Lista de Reserva del Producto

Al iniciar el desarrollo del producto es difícil tener definidos todos los requerimientos del software, sin embargo suelen surgir los más importantes. Estas actividades priorizadas son recogidas en el artefacto Lista de Reserva del Producto y definen el trabajo que se va a realizar en el proyecto.

La lista puede crecer y modificarse a medida que se obtiene más conocimiento acerca del producto o varían las necesidades del cliente, con la restricción de que solo puede cambiarse entre las iteraciones establecidas para el desarrollo de la aplicación. El objetivo de este artefacto es asegurar que el producto obtenido sea el más correcto, útil y eficiente posible (PEÑALVER, 2008).

A continuación se muestra la lista priorizada de la versión v0.1 de SINIBOT. De cada requerimiento se muestra su descripción, el tiempo en semanas estimado para su desarrollo y el responsable de dicha estimación (ANA: Analista, PRO: Programador).

Ítem *	Descripción	Estimación	Estimado por
Prioridad: Alta			
1	Comenzar el recorrido usando una lista inicial de URLs.	1	PRO
2	Extraer la próxima URL a visitar.	1	PRO
3	Recolectar las características de los documentos tratados.	1	PRO
4	Detectar el motivo del error si una URL no es tratada.	1	PRO
5	Extraer los enlaces de las páginas web y agregarlos a la lista de URLs encontradas.	1	PRO
6	Recolectar el contenido web mediante procesos en paralelo.	2	PRO
7	Almacenar en la base de datos toda la información obtenida durante los recorridos.	5	PRO
8	Permitir al administrador establecer desde la interfaz web la lista de URLs de partida para cada recorrido.	1	PRO
9	Permitir al administrador modificar desde la interfaz web las opciones de configuración para cada recorrido.	1	PRO
10	Programar fecha y hora de inicio y fin de un recorrido.	1	PRO
11	Detener un recorrido.	1	PRO
12	Revisar información de los recorridos.	1	PRO

Ítem *	Descripción	Estimación	Estimado por
13	El sistema tendrá un instalador automatizado con interfaz de consola.	3	PRO
14	El sistema contará con una Interfaz web de administración.	4	PRO
Prioridad: Media			
1	Generar estadísticas durante el proceso de recolección.	1	PRO
2	Obtener y almacenar las características del *hardware* sobre el cual se ejecuta cada recorrido.	1	PRO
3	Autenticar usuario.	1	PRO
4	Adicionar usuario.	1	PRO
5	Editar los datos del usuario.	1	PRO
6	Eliminar usuario.	1	PRO
7	Asignar roles a los usuarios para controlar el nivel de acceso a las distintas funcionalidades.	1	ANA
8	Tratamiento de errores.	1	PRO
Requisitos No Funcionales (RNF)			
1	El sistema debe estar bien documentado para minimizar el tiempo de mantenimiento en caso de ser requerido.	1	ANA
2	El sistema debe funcionar sobre una plataforma GNU/Linux.	2	PRO
3	El sistema debe ser extensible permitiendo agregar nuevas funcionalidades en un futuro.	2	PRO

Tabla 1: Lista de Reserva del Producto.

2.6. Historias de Usuario

Constituyen la técnica para especificar las funcionalidades del software. Son escritas en un lenguaje natural y su construcción depende principalmente de la habilidad que tenga el cliente para definirlas. Permiten responder rápidamente a los requerimientos cambiantes y son la base para las pruebas funcionales del sistema (PEÑALVER, 2008). A continuación se muestran las Historias de Usuario que rigen el desarrollo de la versión v0.1 de SINIBOT.

Historia de Usuario	**Número:** U-SINIBOT-1
Nombre: Comenzar Recorrido.	
Modificación de Historia de Usuario Número:	
Usuario(s): Saimel Sáez Estrada.	**Iteración Asignada:** 1
Prioridad en Negocio: (**Alta** / Media / Baja)	**Puntos Estimados:** 1
Riesgo en Desarrollo: (**Alto** / Medio / Bajo)	**Puntos Reales:** 1
Descripción: El sistema comenzará el recorrido siguiendo una lista de URLs de inicio.	
Observaciones: Funcionalidad implementada en la plataforma Larbin, utilizada como base para el desarrollo del presente sistema.	

Tabla 2: Historia de Usuario Comenzar Recorrido.

Historia de Usuario	**Número:** U-SINIBOT-2
Nombre: Extraer URL a Visitar.	
Modificación de Historia de Usuario Número:	
Usuario(s): Saimel Sáez Estrada.	**Iteración Asignada:** 1
Prioridad en Negocio: (**Alta** / Media / Baja)	**Puntos Estimados:** 1
Riesgo en Desarrollo: (**Alto** / Medio / Bajo)	**Puntos Reales:** 1
Descripción: Para continuar con el recorrido, se obtendrán automáticamente la próxima URL de la Lista de URLs Encontradas.	
Observaciones: Funcionalidad implementada en la plataforma Larbin, utilizada como base para el desarrollo del presente sistema.	

Tabla 3: Historia de Usuario Extraer URL a Visitar.

Historia de Usuario	**Número:** U-SINIBOT-3
Nombre: Recuperar Características y Contenido.	
Modificación de Historia de Usuario Número:	
Usuario(s): Saimel Sáez Estrada.	**Iteración Asignada:** 1
Prioridad en Negocio: (**Alta** / Media / Baja)	**Puntos Estimados:** 1
Riesgo en Desarrollo: (**Alto** / Medio / Bajo)	**Puntos Reales:** 1
Descripción: Al visitar una URL, se recuperarán del documento tratado las características requeridas por el usuario, así como el contenido de dicho documento.	
Observaciones: Funcionalidad implementada en la plataforma Larbin, utilizada como base para el desarrollo del presente sistema.	

Tabla 4: Historia de Usuario Recuperar Características y Contenido.

Historia de Usuario	**Número:** U-SINIBOT-4
Nombre: Detectar Error.	
Modificación de Historia de Usuario Número:	
Usuario(s): Saimel Sáez Estrada.	**Iteración Asignada:** 1
Prioridad en Negocio: (**Alta** / Media / Baja)	**Puntos Estimados:** 1
Riesgo en Desarrollo: (**Alto** / Medio / Bajo)	**Puntos Reales:** 1
Descripción: Si al visitar una URL, esta no puede ser tratada, se detectará el motivo del error, asociando un código a cada error posible.	
Observaciones: Funcionalidad implementada en la plataforma Larbin, utilizada como base para el desarrollo del presente sistema.	

Tabla 5: Historia de Usuario Detectar Error.

Historia de Usuario	**Número:** U-SINIBOT-5
Nombre: Extraer Enlaces.	
Modificación de Historia de Usuario Número:	
Usuario(s): Saimel Sáez Estrada.	**Iteración Asignada:** 1
Prioridad en Negocio: (**Alta** / Media / Baja)	**Puntos Estimados:** 1
Riesgo en Desarrollo: (**Alto** / Medio / Bajo)	**Puntos Reales:** 1
Descripción: Si el documento tratado es una página web y posee enlaces, estos serán extraídos y agregados a la Lista de URLs Encontradas.	
Observaciones: Funcionalidad implementada en la plataforma Larbin, utilizada como base para el desarrollo del presente sistema.	

Tabla 6: Historia de Usuario Extraer Enlaces.

Historia de Usuario	**Número:** U-SINIBOT-6
Nombre: Realizar Recolección en Paralelo.	
Modificación de Historia de Usuario Número:	
Usuario(s): Saimel Sáez Estrada.	**Iteración Asignada:** 1
Prioridad en Negocio: (**Alta** / Media / Baja)	**Puntos Estimados:** 2
Riesgo en Desarrollo: (**Alto** / Medio / Bajo)	**Puntos Reales:** 1
Descripción: El sistema realizará la recolección de contenido web mediante procesos en paralelo sobre un mismo ordenador.	
Observaciones: Funcionalidad implementada en la plataforma Larbin, utilizada como base para el desarrollo del presente sistema.	

Tabla 7: Historia de Usuario Realizar Recolección en Paralelo.

Historia de Usuario	**Número:** U-SINIBOT-7
Nombre: Generar Estadísticas.	
Modificación de Historia de Usuario Número:	
Usuario(s): Saimel Sáez Estrada.	**Iteración Asignada:** 1
Prioridad en Negocio: (Alta / **Media** / Baja)	**Puntos Estimados:** 1
Riesgo en Desarrollo: (Alto / **Medio** / Bajo)	**Puntos Reales:** 1
Descripción: Durante el proceso de recolección, el sistema debe generar estadísticas correspondientes al recorrido y a las URLs que visita.	
Observaciones: Funcionalidad implementada en la plataforma Larbin, utilizada como base para el desarrollo del presente sistema.	

Tabla 8: Historia de Usuario Generar Estadísticas.

Historia de Usuario	**Número:** U-SINIBOT-8
Nombre: Almacenar Información.	
Modificación de Historia de Usuario Número:	
Usuario(s): Saimel Sáez Estrada.	**Iteración Asignada:** 2
Prioridad en Negocio: (**Alta** / Media / Baja)	**Puntos Estimados:** 5
Riesgo en Desarrollo: (**Alto** / Medio / Bajo)	**Puntos Reales:** 5
Descripción: El sistema almacenará en una base de datos las estadísticas generadas durante el recorrido, las características de los documentos tratados, los enlaces de las páginas web y las opciones de configuración para cada recorrido.	
Observaciones: El contenido de los documentos recolectados será almacenado utilizando la tecnología FreeNAS.	

Tabla 9: Historia de Usuario Almacenar Información.

Historia de Usuario	**Número:** U-SINIBOT-9
Nombre: Establecer Lista de URLs Iniciales.	
Modificación de Historia de Usuario Número:	
Usuario(s): Anlí Wilfredo Martínez Gallardo.	**Iteración Asignada:** 2
Prioridad en Negocio: (**Alta** / Media / Baja)	**Puntos Estimados:** 1
Riesgo en Desarrollo: (**Alto** / Medio / Bajo)	**Puntos Reales:** 1
Descripción: El sistema permitirá al Administrador establecer desde la interfaz web, las URLs de partida para cada recorrido.	
Observaciones: Dicha lista de URLs se puede establecer escribiendo cada URL en un cuadro de texto o cargándolas desde un fichero.	
Prototipo de interfaces: Ver Anexo 1.	

Tabla 10: Historia de Usuario Establecer Lista de URLs Iniciales.

Historia de Usuario	**Número:** U-SINIBOT-10
Nombre: Configurar Recorrido.	
Modificación de Historia de Usuario Número:	
Usuario(s): Anlí Wilfredo Martínez Gallardo.	**Iteración Asignada:** 2
Prioridad en Negocio: (**Alta** / Media / Baja)	**Puntos Estimados:** 1
Riesgo en Desarrollo: (**Alto** / Medio / Bajo)	**Puntos Reales:** 1
Descripción: El sistema permitirá al Administrador modificar desde la interfaz web las opciones de configuración para cada recorrido.	
Observaciones: Existen parámetros que deben ser objeto de atención pues sobrecargan en gran medida el tráfico en la red.	
Prototipo de interfaces: Ver Anexo 1.	

Tabla 11: Historia de Usuario Configurar Recorrido.

Historia de Usuario	**Número:** U-SINIBOT-11
Nombre: Gestionar Estado de los Recorridos.	
Modificación de Historia de Usuario Número:	
Usuario(s): Saimel Sáez y Anlí W. Martínez.	**Iteración Asignada:** 2
Prioridad en Negocio: (**Alta** / Media / Baja)	**Puntos Estimados:** 2
Riesgo en Desarrollo: (**Alto** / Medio / Bajo)	**Puntos Reales:** 2
Descripción: El sistema permitirá al Administrador realizar desde la interfaz web las siguientes acciones: programar fecha y hora de inicio y fin de un recorrido y detenerlo en caso de ser necesario.	
Observaciones: Cuando se programa la fecha y hora de inicio de un recorrido, el sistema impedirá que se ejecute o programe un nuevo recorrido.	
Prototipo de interfaces: Ver Anexo 1.	

Tabla 12: Historia de Usuario Gestionar Estado de los Recorridos.

Historia de Usuario	**Número:** U-SINIBOT-12
Nombre: Gestionar Información Recolectada.	
Modificación de Historia de Usuario Número:	
Usuario(s): Anlí Wilfredo Martínez Gallardo.	**Iteración Asignada:** 2
Prioridad en Negocio: (**Alta** / Media / Baja)	**Puntos Estimados:** 1
Riesgo en Desarrollo: (**Alto** / Medio / Bajo)	**Puntos Reales:** 1
Descripción: El sistema permitirá a todos los usuarios revisar la información referente a los recorridos realizados, mientras que el Administrador podrá además eliminar dicha información.	
Prototipo de interfaces: Ver Anexo 2.	

Tabla 13: Historia de Usuario Gestionar Información Recolectada.

Historia de Usuario	**Número:** U-SINIBOT-13
Nombre: Gestionar Usuarios.	
Modificación de Historia de Usuario Número:	
Usuario(s): Anlí Wilfredo Martínez Gallardo.	**Iteración Asignada:** 2
Prioridad en Negocio: (Alta / **Media** / Baja)	**Puntos Estimados:** 2
Riesgo en Desarrollo: (Alto / **Medio** / Bajo)	**Puntos Reales:** 1
Descripción: El sistema permitirá al Administrador realizar desde la interfaz web las siguientes acciones: autenticar usuario, agregar usuario, editar información de un usuario y eliminar un usuario.	
Observaciones: El sistema establecerá los roles de Administrador y Consultor, los cuales tendrán diferentes privilegios dentro de la aplicación.	
Prototipo de interfaces: Ver Anexo 3.	

Tabla 14: Historia de Usuario Gestionar Usuarios.

Historia de Usuario	**Número:** U-SINIBOT-14
Nombre: Almacenar Características de *Hardware*.	
Modificación de Historia de Usuario Número:	
Usuario(s): Anlí Wilfredo Martínez Gallardo.	**Iteración Asignada:** 2
Prioridad en Negocio: (Alta / **Media** / Baja)	**Puntos Estimados:** 1
Riesgo en Desarrollo: (Alto / **Medio** / Bajo)	**Puntos Reales:** 1
Descripción: El sistema obtendrá y almacenará en la base de datos las principales características del *hardware* sobre el cual se ejecuta cada recorrido.	

Tabla 15: Historia de Usuario Almacenar Características de *Hardware*.

Historia de Usuario	**Número:** U-SINIBOT-15
Nombre: Crear Instalador.	
Modificación de Historia de Usuario Número:	
Usuario(s): Saimel Sáez Estrada.	**Iteración Asignada:** 3
Prioridad en Negocio: (**Alta** / Media / Baja)	**Puntos Estimados:** 3
Riesgo en Desarrollo: (**Alto** / Medio / Bajo)	**Puntos Reales:** 2
Descripción: El sistema contará con un instalador automatizado para la gestión de librerías, dependencias, componentes y permisos para el funcionamiento de los subsistemas de Almacenamiento, Recolección y Administración Web.	
Observaciones: El instalador tendrá una interfaz de consola.	

Tabla 16: Historia de Usuario Crear Instalador.

Historia de Usuario	**Número:** U-SINIBOT-16
Nombre: Realizar Tratamiento de Errores.	
Modificación de Historia de Usuario Número:	
Usuario(s): Saimel Sáez y Anlí W. Martínez.	**Iteración Asignada:** 3
Prioridad en Negocio: (Alta / **Media** / Baja)	**Puntos Estimados:** 2
Riesgo en Desarrollo: (Alto / **Medio** / Bajo)	**Puntos Reales:** 2
Descripción: Los subsistemas de Instalación, Recolección y Administración Web tendrán implementado mecanismos para el tratamiento de errores, capaces de gestionar las posibles excepciones generadas durante la ejecución de los mismos.	
Observaciones: En caso de capturarse una excepción durante la ejecución, debe mostrarse un mensaje indicando la naturaleza de la misma.	

Tabla 17: Historia de Usuario Realizar Tratamiento de Errores.

2.7. Diagrama de Componentes

En este artefacto se muestran las organizaciones y dependencias lógicas entre los componentes del *software*, ya sean estos ficheros de código fuente, binarios o ejecutables. Los elementos de modelado que lo conforman son los componentes y paquetes, los que muestran la estructura de alto nivel del modelo de implementación (FERNÁNDEZ, 2001).

El Diagrama de Componentes de la versión v0.1 de SINIBOT (ver Figura 12) muestra tres componentes principales: (1) el Paquete Servidor (*Server Package*), (2) el Paquete de Recolección (*Crawl Package*) y (3) la Base de Datos db_sinibot (*database*). El primero presenta un fichero *(file)* con las URLs iniciales y el Paquete Interfaz, el cual se compone por el *framework* CodeIgniter y la librería (*library*) FreakAuth_ligth. CodeIgniter utiliza la librería FreakAuth_ligth mientras que la

Interfaz, considerada un paquete por su complejidad, se encarga de crear el fichero con las URLs iniciales.

Por su parte, el Paquete de Recolección se compone del ejecutable (executable) sinibot, el fichero (file) sinibot.conf y las librerías (library) adns, pthread y pqxx. Desde el Paquete Interfaz se modifica el fichero de configuración sinibot.conf y se ejecuta sinibot, este último utiliza las librerías antes mencionadas y obtiene las opciones de configuración desde sinibot.conf.

El Paquete de Recolección y el Paquete Interfaz son los encargados de la interacción con la Base de Datos db_sinibot.

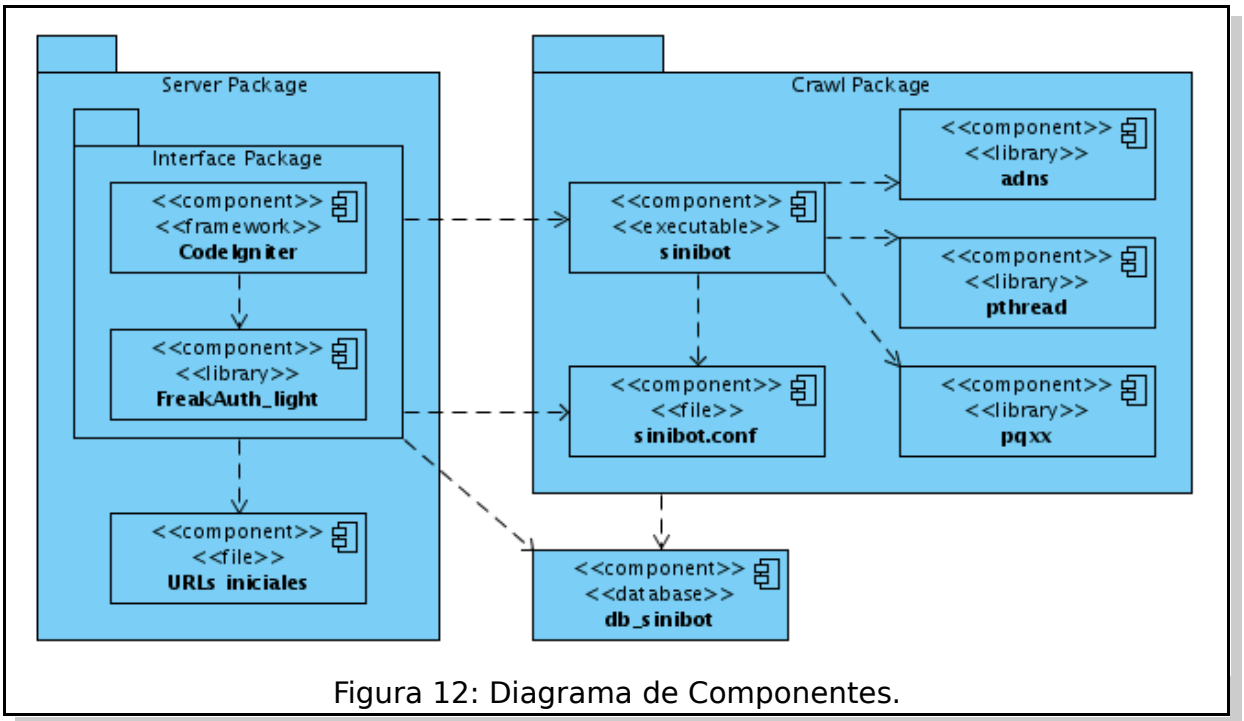


Figura 12: Diagrama de Componentes.

2.8. Conclusiones

Después de presentar las características del sistema, se evidencia la efectividad del proceso de investigación efectuado, en el cual se concluyó utilizar Larbin como plataforma base para el desarrollo de SINIBOT, decisión que permitió adquirir implementadas, de las quince Historias de Usuario definidas anteriormente, las siete más importantes para el proceso de recolección.

Al concluir el presente capítulo, se han creado las condiciones para efectuar el desarrollo y validación de la versión v0.1 de SINIBOT.

3. CAPÍTULO 3: DESARROLLO Y VALIDACIÓN DE SINIBOT

3.1. Introducción

Un producto listo para ser entregado, requiere el completo desarrollo y validación de las funcionalidades previamente definidas. En el presente capítulo se presentará el Plan de Releases para el control de las entregas inmediatas. Se muestran las Tareas de Ingeniería que guiarán el proceso de desarrollo del sistema. Se presentan los Casos de Prueba a los cuales fue sometido SINIBOT en cada una de las iteraciones del proyecto. Finalmente, se exponen los resultados obtenidos y las funcionalidades alcanzadas durante el período de desarrollo.

3.2. Plan de Releases

En el artefacto Plan de Releases se recogen las iteraciones a realizar con sus características, además del orden de las Historias de Usuario con su planificación estimada para ser entregadas. El objetivo de cada iteración es obtener una versión funcional del sistema que, aunque no cuente con todos los requerimientos definidos, constituye un resultado de valor para el cliente (PEÑALVER, 2008). A continuación se muestra el Plan de Releases de la versión v0.1 de SINIBOT.

Release	Descripción de la iteración	HU a implementar	Duración total
Iteración 1	Comprende el período de investigación realizado, donde se concluyó utilizar Larbin como plataforma para el desarrollo de SINIBOT.	U-SINIBOT-1 U-SINIBOT-2 U-SINIBOT-3 U-SINIBOT-4 U-SINIBOT-5 U-SINIBOT-6 U-SINIBOT-7	01/11/08-27/12/08
Iteración 2	Definir, implementar y probar las principales funcionalidades de la versión v0.1 de SINIBOT.	U-SINIBOT-8 U-SINIBOT-9 U-SINIBOT-10 U-SINIBOT-11 U-SINIBOT-12 U-SINIBOT-13 U-SINIBOT-14	05/01/09-27/02/09
Iteración 3	Crear el instalador de la versión v0.1 de SINIBOT, validar el tratamiento de excepciones y probar las funcionalidades desarrolladas.	U-SINIBOT-15 U-SINIBOT-16	01/03/09-31/05/09

Tabla 18: Plan de Releases de la versión v0.1 de SINIBOT.

3.3. Tareas de Ingeniería

En este artefacto se recogen las principales tareas a realizar por cada Historia de Usuario. Las siete primeras Historias de Usuario definidas para SINIBOT no generan tareas, ya que están implementadas en la plataforma Larbin. A continuación se

presentan las Tareas de Ingeniería de la versión v0.1 de SINIBOT.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-8-01
Número Historia de Usuario: U-SINIBOT-8	
Nombre Tarea: Estudiar Almacenamiento de Contenido en Larbin.	
Tipo de Tarea: (Desarrollo / **Estudio**)	**Puntos Estimados:** 0.1
Fecha Inicio: 05/01/2009	**Fecha Fin:** 06/01/2009
Programador Responsable: Saimel Sáez Estrada.	
Descripción: Realizar estudio de la documentación y el código de Larbin para conocer cómo se almacena el contenido recolectado, de esta manera se podrá modificar dicho proceso y se almacenará el contenido en una base de datos.	

Tabla 19: Tarea de Ingeniería Estudiar Almacenamiento de Contenido en Larbin.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-8-02
Número Historia de Usuario: U-SINIBOT-8	
Nombre Tarea: Estudiar Librería pqxx.	
Tipo de Tarea: (Desarrollo / **Estudio**)	**Puntos Estimados:** 0.1
Fecha Inicio: 07/01/2009	**Fecha Fin:** 08/01/2009
Programador Responsable: Saimel Sáez Estrada.	
Descripción: Realizar estudio de la librería pqxx para manipular las conexiones entre el subsistema de recolección y la base de datos.	

Tabla 20: Tarea de Ingeniería Estudiar Librería pqxx.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-8-03
Número Historia de Usuario: U-SINIBOT-8	
Nombre Tarea: Crear Base de Datos.	
Tipo de Tarea: (**Desarrollo** / Estudio)	**Puntos Estimados:** 0.3
Fecha Inicio: 09/01/2009	**Fecha Fin:** 11/01/2009
Programador Responsable: Saimel Sáez Estrada y Anlí W. Martínez Gallardo.	
Descripción: Diseñar y crear la base de datos de SINIBOT que almacenará el contenido recolectado y las estadísticas generadas durante los recorridos.	

Tabla 21: Tarea de Ingeniería Crear Base de Datos.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-8-04
Número Historia de Usuario: U-SINIBOT-8	
Nombre Tarea: Almacenar Contenido Complementario.	
Tipo de Tarea: (**Desarrollo** / Estudio)	**Puntos Estimados:** 2.0
Fecha Inicio: 12/01/2009	**Fecha Fin:** 25/01/2009
Programador Responsable: Saimel Sáez Estrada.	
Descripción: Definir e implementar las funcionalidades necesarias para manejar las conexiones a la base de datos, así como almacenar en la misma las estadísticas de los recorridos y las características de los documentos tratados, como son: cabecera HTTP, enlaces, URL y código de error si no pudo ser tratado, por solo citar algunas.	

Tabla 22: Tarea de Ingeniería Almacenar Contenido Complementario.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-8-05
Número Historia de Usuario: U-SINIBOT-8	
Nombre Tarea: Estudiar FreeNAS.	
Tipo de Tarea: (Desarrollo / **Estudio**)	**Puntos Estimados:** 0.5
Fecha Inicio: 26/01/2009	**Fecha Fin:** 29/01/2009
Programador Responsable: Saimel Sáez Estrada.	
Descripción: Realizar estudio de la documentación de FreeNAS para conocer sus principios de funcionamiento.	

Tabla 23: Tarea de Ingeniería Estudiar FreeNAS.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-8-06
Número Historia de Usuario: U-SINIBOT-8	
Nombre Tarea: Instalar FreeNAS.	
Tipo de Tarea: (**Desarrollo** / Estudio)	**Puntos Estimados:** 0.5
Fecha Inicio: 30/01/2009	**Fecha Fin:** 01/02/2009
Programador Responsable: Saimel Sáez Estrada.	
Descripción: Proceder a instalar y configurar FreeNAS con el objetivo de utilizar esta tecnología para almacenar el contenido de los documentos recolectados.	

Tabla 24: Tarea de Ingeniería Instalar FreeNAS.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-8-07
Número Historia de Usuario: U-SINIBOT-8	
Nombre Tarea: Almacenar Contenido de los Documentos.	
Tipo de Tarea: (**Desarrollo** / Estudio)	**Puntos Estimados:** 1.5
Fecha Inicio: 02/01/2009	**Fecha Fin:** 10/02/2009
Programador Responsable: Saimel Sáez Estrada.	
Descripción: Definir e implementar las funcionalidades necesarias para manejar las conexiones al depósito en FreeNAS, así como almacenar en el mismo el contenido de los documentos tratados.	

Tabla 25: Tarea de Ingeniería Almacenar Contenido de los Documentos.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-9-01
Número Historia de Usuario: U-SINIBOT-9	
Nombre Tarea: Estudiar Obtención de URLs de Partida.	
Tipo de Tarea: (Desarrollo / **Estudio**)	**Puntos Estimados:** 0.3
Fecha Inicio: 12/01/2009	**Fecha Fin:** 13/01/2009
Programador Responsable: Anlí W. Martínez Gallardo.	
Descripción: Realizar estudio de la documentación y de la estructura de los ficheros de configuración de Larbin para conocer cómo se obtiene la Lista de URLs Iniciales, de esta manera se podrá modificar dicho proceso y las URLs iniciales serán proporcionadas desde la interfaz web de administración.	

Tabla 26: Tarea de Ingeniería Estudiar Obtención de URLs de Partida.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-9-02
Número Historia de Usuario: U-SINIBOT-9	
Nombre Tarea: Interfaz para Establecer las URLs de Partida.	
Tipo de Tarea: (**Desarrollo** / Estudio)	**Puntos Estimados:** 0.3
Fecha Inicio: 14/01/2009	**Fecha Fin:** 15/01/2009
Programador Responsable: Juan M. Álvarez Tur y Anlí W. Martínez Gallardo.	
Descripción: Diseñar e implementar una interfaz dentro del Módulo de Gestión de Recorridos para que el Administrador pueda establecer la Lista de URLs de Partida.	

Tabla 27: Tarea de Ingeniería Interfaz para Establecer las URLs de Partida.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-9-03
Número Historia de Usuario: U-SINIBOT-9	
Nombre Tarea: Almacenar URLs de Partida.	
Tipo de Tarea: (**Desarrollo** / Estudio)	**Puntos Estimados:** 0.4
Fecha Inicio: 16/01/2009	**Fecha Fin:** 18/01/2009
Programador Responsable: Anlí W. Martínez Gallardo.	
Descripción: Almacenar en la base de datos las URLs de partida para cada recorrido.	

Tabla 28: Tarea de Ingeniería Almacenar URLs de Partida.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-10-01
Número Historia de Usuario: U-SINIBOT-10	
Nombre Tarea: Estudiar Opciones de Configuración.	
Tipo de Tarea: (Desarrollo / **Estudio**)	**Puntos Estimados:** 0.3
Fecha Inicio: 19/01/2009	**Fecha Fin:** 20/01/2009
Programador Responsable: Anlí W. Martínez Gallardo.	
Descripción: Realizar estudio de la documentación y de la estructura de los ficheros de configuración de Larbin para conocer cómo modificar las opciones de configuración, de esta manera se podrá modificar dicho proceso y los parámetros de configuración serán proporcionados desde la interfaz web de administración.	

Tabla 29: Tarea de Ingeniería Estudiar Opciones de Configuración.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-10-02
Número Historia de Usuario: U-SINIBOT-10	
Nombre Tarea: Interfaz para Configurar los Recorridos.	
Tipo de Tarea: (**Desarrollo** / Estudio)	**Puntos Estimados:** 0.3
Fecha Inicio: 21/01/2009	**Fecha Fin:** 22/01/2009
Programador Responsable: Juan M. Álvarez Tur y Anlí W. Martínez Gallardo.	
Descripción: Diseñar e implementar una interfaz dentro del Módulo de Gestión de Recorridos para que el Administrador pueda modificar las opciones de configuración.	

Tabla 30: Tarea de Ingeniería Interfaz para Configurar los Recorridos.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-10-03
Número Historia de Usuario: U-SINIBOT-10	
Nombre Tarea: Almacenar Opciones de Configuración.	
Tipo de Tarea: (**Desarrollo** / Estudio)	**Puntos Estimados:** 0.4
Fecha Inicio: 23/01/2009	**Fecha Fin:** 25/01/2009
Programador Responsable: Anlí W. Martínez Gallardo.	
Descripción: Almacenar en la base de datos las opciones de configuración para cada recorrido.	

Tabla 31: Tarea de Ingeniería Almacenar Opciones de Configuración.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-11-01
Número Historia de Usuario: U-SINIBOT-11	
Nombre Tarea: Gestionar Estado de los Recorridos.	
Tipo de Tarea: (**Desarrollo** / Estudio)	**Puntos Estimados:** 2.0
Fecha Inicio: 25/01/2009	**Fecha Fin:** 07/02/2009
Programador Responsable: Anlí W. Martínez Gallardo.	
Descripción: Diseñar la interfaz e implementar las funcionalidades necesarias para el que, a través del Módulo de Gestión de Recorridos, el Administrador pueda iniciar, detener o programar un recorrido.	

Tabla 32: Tarea de Ingeniería Gestionar Estado de los Recorridos.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-12-01
Número Historia de Usuario: U-SINIBOT-12	
Nombre Tarea: Gestionar Información Recolectada.	
Tipo de Tarea: (**Desarrollo** / Estudio)	**Puntos Estimados:** 1.0
Fecha Inicio: 08/02/2009	**Fecha Fin:** 14/02/2009
Programador Responsable: Anlí W. Martínez Gallardo.	
Descripción: Diseñar la interfaz e implementar las funcionalidades necesarias para que, a través del Módulo de Gestión de Información, los usuarios puedan revisar la información recolectada.	

Tabla 33: Tarea de Ingeniería Gestionar Información Recolectada.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-13-01
Número Historia de Usuario: U-SINIBOT-10	
Nombre Tarea: Estudiar Librería FreakAuth_light.	
Tipo de Tarea: (Desarrollo / **Estudio**)	**Puntos Estimados:** 0.4
Fecha Inicio: 15/02/2009	**Fecha Fin:** 17/02/2009
Programador Responsable: Anlí W. Martínez Gallardo.	
Descripción: Realizar estudio de la documentación de la librería FreakAuht_light del *framework* CodeIgniter con el objetivo de conocer sus principios de funcionamiento.	

Tabla 34: Tarea de Ingeniería Estudiar Librería FreakAuth_light.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-13-02
Número Historia de Usuario: U-SINIBOT-10	
Nombre Tarea: Configurar Librería FreakAuth_light.	
Tipo de Tarea: (**Desarrollo** / Estudio)	**Puntos Estimados:** 0.6
Fecha Inicio: 18/02/2009	**Fecha Fin:** 21/02/2009
Programador Responsable: Anlí W. Martínez Gallardo.	
Descripción: Configurar la librería FreakAuth_ligth, la cual permite realizar acciones como: autenticar, adicionar, editar y eliminar usuarios, así como establecer entre ellos roles para diferenciar los privilegios que tendrán los Consultores y el Administrador.	

Tabla 35: Tarea de Ingeniería Configurar la Librería FreakAuth_light.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-14-01
Número Historia de Usuario: U-SINIBOT-14	
Nombre Tarea: Obtener Características de *Hardware*.	
Tipo de Tarea: (Desarrollo / **Estudio**)	**Puntos Estimados:** 0.4
Fecha Inicio: 11/02/2009	**Fecha Fin:** 13/02/2009
Programador Responsable: Saimel Sáez Estrada.	
Descripción: Estudiar los comandos necesarios para obtener las principales características del *hardware* sobre el cual se realiza cada recorrido.	

Tabla 36: Tarea de Ingeniería Obtener Características de *Hardware*.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-14-02
Número Historia de Usuario: U-SINIBOT-14	
Nombre Tarea: Almacenar Características de *Hardware*.	
Tipo de Tarea: (**Desarrollo** / Estudio)	**Puntos Estimados:** 0.6
Fecha Inicio: 14/02/2009	**Fecha Fin:** 17/02/2009
Programador Responsable: Saimel Sáez Estrada.	
Descripción: Implementar las funcionalidades necesarias para almacenar en la base de datos las principales características del *hardware* sobre el cual se realiza cada recorrido.	

Tabla 37: Tarea de Ingeniería Almacenar Características de *Hardware*.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-15-01
Número Historia de Usuario: U-SINIBOT-15	
Nombre Tarea: Estudiar Proceso de Instalación.	
Tipo de Tarea: (Desarrollo / **Estudio**)	**Puntos Estimados:** 0.2
Fecha Inicio: 01/03/2009	**Fecha Fin:** 03/03/2009
Programador Responsable: Saimel Sáez Estrada.	
Descripción: Estudiar los pasos necesarios para la instalación por separado de los productos obtenidos en los subsistemas de Almacenamiento, Recolección y Administración Web.	

Tabla 38: Tarea de Ingeniería Estudiar Proceso de Instalación de Larbin.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-15-02
Número Historia de Usuario: U-SINIBOT-15	
Nombre Tarea: Integración de los Componentes de SINIBOT.	
Tipo de Tarea: (**Desarrollo** / Estudio)	**Puntos Estimados:** 0.5
Fecha Inicio: 04/03/2009	**Fecha Fin:** 07/03/2009
Programador Responsable: Saimel Sáez Estrada y Anlí W. Martínez Gallardo.	
Descripción: Integrar en un único paquete los productos obtenidos en los subsistemas de Almacenamiento, Recolección y Administración Web.	

Tabla 39: Tarea de Ingeniería Integración de los Componentes de SINIBOT.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-15-03
Número Historia de Usuario: U-SINIBOT-15	
Nombre Tarea: Crear Instalador.	
Tipo de Tarea: (**Desarrollo** / Estudio)	**Puntos Estimados:** 1.2
Fecha Inicio: 08/03/2009	**Fecha Fin:** 15/03/2009
Programador Responsable: Indira Garcés Pérez y Saimel Sáez Estrada.	
Descripción: Implementar un instalador con interfaz de consola, capaz de gestionar todas las librerías, dependencias, componentes y permisos para el correcto funcionamiento del sistema.	

Tabla 40: Tarea de Ingeniería Crear Instalador.

Tarea de Ingeniería	**Número Tarea:** U-SINIBOT-16-01
Número Historia de Usuario: U-SINIBOT-16	
Nombre Tarea: Realizar Tratamiento de Errores.	
Tipo de Tarea: (**Desarrollo** / Estudio)	**Puntos Estimados:** XX
Fecha Inicio: 16/03/2009	**Fecha Fin:** 29/03/2009
Programador Responsable: Saimel Sáez Estrada y Anlí W. Martínez Gallardo.	
Descripción: Implementar en cada subsistema mecanismos para el tratamiento de posibles excepciones generadas durante la ejecución del sistema.	

Tabla 41: Tarea de Ingeniería Realizar Tratamiento de Errores.

3.4. Casos de Prueba de Aceptación

En este artefacto se describen las pruebas realizadas a cada Historia de Usuario con el objetivo de validar la versión del sistema obtenida en cada iteración, logrando así un producto con calidad, listo para ser entregado.

El cumplimiento satisfactorio de estas pruebas constituye el hito para avanzar hacia la próxima iteración (PEÑALVER, 2008). A continuación se muestran los Casos de Pruebas de Aceptación definidos en la versión v0.1 de SINIBOT.

Caso de Prueba de Aceptación	**Código Caso de Prueba:** U-SINIBOT-1-01
Historia de Usuario: U-SINIBOT-1	**Probador:** Saimel Sáez Estrada.
Descripción de la Prueba: Modificar los ficheros de configuración de Larbin y establecer las URLs de partida.	
Condiciones de Ejecución: Tener instalado Larbin en su versión v2.6.3.	
Entrada / Pasos de ejecución: Configurar Larbin e iniciar recorrido.	
Resultado Esperado: Las primeras URLs tratadas coinciden con las URLs de partida.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 42: Caso de Prueba de Aceptación U-SINIBOT-1-01.

Caso de Prueba de Aceptación	**Código Caso de Prueba:** U-SINIBOT-2-01
Historia de Usuario: U-SINIBOT-2	**Probador:** Saimel Sáez Estrada.
Descripción de la Prueba: Se configura el paquete Larbin y se establece una única URL de partida, la cual representa un documento HTML con enlaces.	
Condiciones de Ejecución: Tener instalado Larbin en su versión v2.6.3.	
Entrada / Pasos de ejecución: Configurar Larbin e iniciar recorrido.	
Resultado Esperado: Una vez que la URL de partida es tratada, Larbin continúa el recorrido a través de los enlaces encontrados.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 43: Caso de Prueba de Aceptación U-SINIBOT-2-01.

Caso de Prueba de Aceptación	**Código Caso de Prueba:** U-SINIBOT-3-01
Historia de Usuario: U-SINIBOT-3	**Probador:** Saimel Sáez Estrada.
Descripción de la Prueba: Configurar Larbin para que recolecte el contenido de las URLs que visita.	
Condiciones de Ejecución: Tener instalado Larbin en su versión v2.6.3.	
Entrada / Pasos de ejecución: Configurar Larbin e iniciar recorrido.	
Resultado Esperado: El contenido y demás características de los documentos tratados es recolectado.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 44: Caso de Prueba de Aceptación U-SINIBOT-3-01.

Caso de Prueba de Aceptación	**Código Caso de Prueba:** U-SINIBOT-4-01
Historia de Usuario: U-SINIBOT-4	**Probador:** Saimel Sáez Estrada.
Descripción de la Prueba: Realizar un recorrido cuyas URLs de partida no puedan ser tratadas por diversos motivos.	
Condiciones de Ejecución: Tener instalado Larbin en su versión v2.6.3.	
Entrada / Pasos de ejecución: Configurar Larbin e iniciar recorrido.	
Resultado Esperado: Para cada URL que no puede ser tratada se detectará el motivo del error.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 45: Caso de Prueba de Aceptación U-SINIBOT-4-01.

Caso de Prueba de Aceptación	**Código Caso de Prueba:** U-SINIBOT-5-01
Historia de Usuario: U-SINIBOT-5	**Probador:** Saimel Sáez Estrada.
Descripción de la Prueba: Se configura el paquete Larbin y se establecen URLs de partida que representan documentos HTML con enlaces.	
Condiciones de Ejecución: Tener instalado Larbin en su versión v2.6.3.	
Entrada / Pasos de ejecución: Configurar Larbin e iniciar recorrido.	
Resultado Esperado: La cantidad de documentos tratados excede a la cantidad de URLs de partida, evidenciando que los enlaces encontrados son extraídos y agregados a la Lista de URLs Encontradas.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 46: Caso de Prueba de Aceptación U-SINIBOT-5-01.

Caso de Prueba de Aceptación	**Código Caso de Prueba:** U-SINIBOT-6-01
Historia de Usuario: U-SINIBOT-6	**Probador:** Saimel Sáez Estrada.
Descripción de la Prueba: Para las mismas URLs de partida, efectuar un recorrido A donde a los parámetros 'pagesConnexions' y 'dnsConnexions' se le asigne valor 1; efectuar un recorrido B donde estos parámetros tengan valor 5. Ambos recorridos serán interrumpidos pasado 1 minuto de ejecución.	
Condiciones de Ejecución: Tener instalado Larbin en su versión v2.6.3.	
Entrada / Pasos de ejecución: Configurar Larbin e iniciar recorrido.	
Resultado Esperado: La cantidad de documentos recolectados en el recorrido B es mayor que la cantidad recolectada en el recorrido A.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 47: Caso de Prueba de Aceptación U-SINIBOT-6-01.

Caso de Prueba de Aceptación	**Código Caso de Prueba:** U-SINIBOT-7-01
Historia de Usuario: U-SINIBOT-7	**Probador:** Saimel Sáez Estrada.
Descripción de la Prueba: Configurar Larbin para que ejecute el servidor web por el puerto 8081.	
Condiciones de Ejecución: Tener instalado Larbin en su versión v2.6.3.	
Entrada / Pasos de ejecución: Configurar Larbin e iniciar recorrido.	
Resultado Esperado: Al visitar desde un navegador la URL http://localhost:8081/ se debe mostrar la página generada por Larbin donde se muestran las estadísticas del recorrido.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 48: Caso de Prueba de Aceptación U-SINIBOT-7-01.

Caso de Prueba de Aceptación	**Código Caso de Prueba:** U-SINIBOT-8-01
Historia de Usuario: U-SINIBOT-8	**Probador:** Saimel Sáez Estrada.
Descripción de la Prueba: El Administrador accede a la interfaz de gestión de recorridos, establece las URLs iniciales y los parámetros de configuración e inicia un recorrido.	
Condiciones de Ejecución: ■ Conectar SINIBOT con el servidor de base de datos y el depósito en FreeNAS. ■ El usuario debe estar autenticado como Administrador.	
Entrada / Pasos de ejecución: El Administrador establece la configuración para el recorrido y hace clic sobre el botón Comenzar Recorrido.	
Resultado Esperado: La base de datos y el depósito en FreeNAS deben contener la información recolectada durante el recorrido.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 49: Caso de Prueba de Aceptación U-SINIBOT-8-01.

Caso de Prueba de Aceptación	**Código Caso de Prueba:** U-SINIBOT-9-01
Historia de Usuario: U-SINIBOT-9	**Probador:** Anlí W. Martínez Gallardo.
Descripción de la Prueba: El Administrador desde la interfaz proporciona un conjunto de URLs donde solo algunas cumplen con el formato establecido.	
Condiciones de Ejecución: El usuario debe estar autenticado como Administrador.	
Entrada / Pasos de ejecución: El Administrador establece las URLs de partida a través del cuadro de texto o cargándolas desde un fichero y hace clic sobre el botón Comenzar Recorrido.	
Resultado Esperado: Las URLs que no cumplen con el formato establecido no son suministradas al subsistema de recolección.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 50: Caso de Prueba de Aceptación U-SINIBOT-9-01.

Caso de Prueba de Aceptación	**Código Caso de Prueba:** U-SINIBOT-10-01
Historia de Usuario: U-SINIBOT-10	**Probador:** Anlí W. Martínez Gallardo.
Descripción de la Prueba: El Administrador desde la interfaz establece los parámetros de configuración para un recorrido donde solo algunos cumplen con la sintaxis establecida.	
Condiciones de Ejecución: El usuario debe estar autenticado como Administrador.	
Entrada / Pasos de ejecución: El Administrador establece los parámetros de configuración para un recorrido y hace clic sobre el botón Comenzar Recorrido.	
Resultado Esperado: Se muestra un mensaje de error indicando los parámetros cuya sintaxis es incorrecta, impidiendo así que comience el recorrido.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 51: Caso de Prueba de Aceptación U-SINIBOT-10-01.

Caso de Prueba de Aceptación	**Código Caso de Prueba:** U-SINIBOT-11-01
Historia de Usuario: U-SINIBOT-11	**Probador:** Anlí W. Martínez Gallardo.
Descripción de la Prueba: El Administrador programa la fecha y hora de inicio y fin de un recorrido.	
Condiciones de Ejecución: El usuario debe estar autenticado como Administrador.	
Entrada / Pasos de ejecución: El Administrador especifica desde la interfaz la fecha y hora de inicio y fin del recorrido y hace clic sobre el botón Comenzar Recorrido.	
Resultado Esperado: El recorrido comienza y termina en la fecha y hora programada.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 52: Caso de Prueba de Aceptación U-SINIBOT-11-01.

Caso de Prueba de Aceptación	**Código Caso de Prueba:** U-SINIBOT-11-02
Historia de Usuario: U-SINIBOT-11	**Probador:** Anlí W. Martínez Gallardo.
Descripción de la Prueba: El Administrador detiene un recorrido en ejecución.	
Condiciones de Ejecución: ■ El usuario debe estar autenticado como Administrador. ■ Debe existir un recorrido en ejecución.	
Entrada / Pasos de ejecución: El Administrador hace clic en el botón Detener Recorrido.	
Resultado Esperado: El recorrido es detenido, se almacena el estado actual del mismo para reanudarlo posteriormente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 53: Caso de Prueba de Aceptación U-SINIBOT-11-02.

Caso de Prueba de Aceptación	**Código Caso de Prueba:** U-SINIBOT-12-01
Historia de Usuario: U-SINIBOT-12	**Probador:** Anlí W. Martínez Gallardo.
Descripción de la Prueba: Los usuarios acceden al Módulo de Gestión de Información con el objetivo de revisar y eliminar la información almacenada.	
Condiciones de Ejecución: ■ El usuario que revisa la información debe estar autenticado en el sistema. ■ Debe existir información sobre al menos un recorrido.	
Entrada / Pasos de ejecución: El usuario accede al Módulo de Gestión de Información y hace clic sobre el recorrido que desea revisar.	
Resultado Esperado: Los Consultores solo pueden visualizar la información almacenada, mientras que el Administrador puede además eliminarla.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 54: Caso de Prueba de Aceptación U-SINIBOT-12-01.

Caso de Prueba de Aceptación	**Código Caso de Prueba:** U-SINIBOT-13-01
Historia de Usuario: U-SINIBOT-13	**Probador:** Anlí W. Martínez Gallardo.
Descripción de la Prueba: Adicionar un nuevo usuario.	
Condiciones de Ejecución: El usuario debe estar autenticado como Administrador.	
Entrada / Pasos de ejecución: El Administrador accede al Módulo de Gestión de Usuarios, llena los campos necesarios para adicionar un nuevo usuario y hace clic en el botón Adicionar Usuario.	
Resultado Esperado: Solo el Administrador tiene privilegios para adicionar a un nuevo usuario. Si la información proporcionada por el Administrador es correcta y el nuevo usuario no ha sido registrado previamente, entonces es añadido al sistema.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 55: Caso de Prueba de Aceptación U-SINIBOT-13-01.

Caso de Prueba de Aceptación	**Código Caso de Prueba:** U-SINIBOT-14-01
Historia de Usuario: U-SINIBOT-14	**Probador:** Anlí W. Martínez Gallardo.
Descripción de la Prueba: Se realizan recorridos sobre distintas arquitecturas de *hardware*.	
Condiciones de Ejecución: Tener instalado la versión v0.1 de SINIBOT.	
Entrada / Pasos de ejecución: Se inicia un nuevo recorrido.	
Resultado Esperado: Se almacenan las principales características del *hardware* sobre el cual se ejecutan los recorridos.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 56: Caso de Prueba de Aceptación U-SINIBOT-14-01.

Caso de Prueba de Aceptación	**Código Caso de Prueba:** U-SINIBOT-15-01
Historia de Usuario: U-SINIBOT-15	**Probador:** Saimel Sáez Estrada.
Descripción de la Prueba: Instalar la versión v0.1 de SINIBOT sobre un ordenador en el cual no estén instalados los paquetes, librerías y dependencias requeridas.	
Condiciones de Ejecución: Quien instala la aplicación debe estar autenticado como usuario *root* en el ordenador.	
Entrada / Pasos de ejecución: Ejecutar el instalador de SINIBOT.	
Resultado Esperado: El instalador debe gestionar los paquetes, librerías, permisos y dependencias requeridas para el correcto funcionamiento de la aplicación.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 57: Caso de Prueba de Aceptación U-SINIBOT-15-01.

3.5. Resultados obtenidos

A continuación se muestran los resultados obtenidos por el equipo de desarrollo. El fruto principal de este trabajo es la versión v0.1 SINIBOT, producto que comenzará a utilizarse en PPSINI y se espera que se le agreguen nuevas funcionalidades en futuras versiones.

3.5.1. Acerca del tiempo de desarrollo

La inexistencia de un sistema automatizado de recolección de contenido web para los productos: MOCIC, GEWEB y FARO de PPSINI, constituía una situación problemática surgida en Octubre de 2008, debido a las necesidades de los productos antes mencionados de procesar grandes volúmenes de contenido web.

Al GPsRI se le asignó la tarea de desarrollar un sistema informático capaz de suplir las necesidades de información de PPSINI, comenzando a inicios de Noviembre de 2008 a dar los primeros pasos para solucionar el problema planteado. Se efectuó un intenso proceso de investigación donde se revisó la información teórica relacionada con la Recolección Masiva de Contenido Web. Durante dicha investigación se analizaron las características de varios sistemas afines, priorizando los que fueron liberados bajo una licencia de software libre, optando finalmente por la utilización de Larbin en su versión v2.6.3 como plataforma base para implementar SINIBOT. A inicios de Enero de 2009 comienza el proceso de desarrollo, guiado por la metodología ágil SXP. En Mayo de 2009 se obtiene la versión v0.1 de SINIBOT, la cual está sujeta a mejoras ya que se le agregarán nuevas funcionalidades.

Para la creación de este producto se realizaron estudios de las librerías pqxx y FreakAuth_light, del framework CodeIgniter y de la documentación de Larbin. Se

comenzó a implementar sin experiencia previa de los conceptos y elementos más importantes para desarrollar un sistema automatizado de recolección de contenido web, aún así, el tiempo de desarrollo fue relativamente corto, cumpliéndose todas las expectativas planteadas en la planificación inicial.

3.5.2. Acerca de las funcionalidades obtenidas

A continuación se muestran las principales funcionalidades que posee SINIBOT en su versión v0.1.

- Generar estadísticas del recorrido.
- Almacenar en la base de datos las características de los documentos tratados.
- Almacenar el contenido de los documentos tratados usando la tecnología FreeNAS.
- Almacenar para cada recorrido las estadísticas generadas, las URLs de partida, las opciones de configuración y las principales características del hardware sobre el cual se ejecuta el recorrido.
- Posee una interfaz web que permite, de manera sencilla y amigable, gestionar los usuarios, la información almacenada en la base de datos y el estado de ejecución de un recorrido, así como establecer las opciones de configuración y la lista de URLs de partida para cada recorrido.
- Posee un instalador automatizado con interfaz de consola, capaz de gestionar librerías, dependencias, componentes y permisos necesarios para el correcto funcionamiento de la aplicación.

3.6. Conclusiones

Tras realizar el desarrollo y validación de SINIBOT, se ha obtenido un producto capaz

de recorrer la Web de forma automatizada y recolectar parte de su contenido, que presenta además una interfaz web de administración y un instalador automatizado con interfaz de consola. Las pruebas efectuadas al producto permitieron corregir algunos errores, mejorando así la calidad de la solución. Se evidencia además que la metodología ágil SXP provee los mecanismos adecuados para guiar eficientemente el proceso de desarrollo de un proyecto.

CONCLUSIONES GENERALES

Una vez realizada la fundamentación teórica que sustentó el presente trabajo, definidas las características de SINIBOT y efectuado su desarrollo y validación, se obtuvieron resultados que permiten a los autores del mismo presentar las siguientes conclusiones:

- El análisis de las principales características de la Web reflejó que su dinamismo, gigantesco tamaño actual y exponencial ritmo de crecimiento eran aspectos a tener en cuenta al definir las características de SINIBOT, puesto que indicaban el gran volumen de información que este debía tratar.

- El estudio de investigaciones similares demostró que en el ámbito nacional existe déficit cognoscitivo sobre la Recolección Masiva de Contenido Web, mientras que en el ámbito internacional, existe abundante conocimiento sobre el tema, ya que aparecen tanto productos privativos cuyas características son mantenidas como secretos comerciales, como soluciones libres, bien documentadas, sobre las cuales es factible continuar desarrollando.

- Recolectar contenido web mediante procesos en paralelo constituye una mejor práctica que la utilización de un único proceso de recolección, debido a que se aprovechan eficientemente los recursos de hardware disponibles.

- Aspectos como la replicación de contenido, el spam, la gran cantidad de documentos en la Web y la velocidad de cambio que estos experimentan, son los principales desafíos que enfrenta en la actualidad la Recolección Masiva de Contenido Web.

- Debido a las características y ritmo de crecimiento de la Web, es imposible recolectar su contenido manualmente, por lo que este proceso debe realizarse de forma automatizada.

- La metodología ágil SXP, empleada para guiar el proceso de desarrollo, se integró perfectamente al mismo, optimizando el tiempo de implementación y aumentando la calidad del producto final.
- Dadas las características distintivas de SINIBOT, se concluye que podrá cumplir satisfactoriamente su labor dentro de PPSINI, así como en otros marcos de trabajo donde sea necesario recolectar grandes volúmenes de contenido web.

En resumen, los principales aportes de este trabajo son:

- Base teórica para el desarrollo de sistemas afines.
- Desarrollo y validación de SINIBOT, un sistema automatizado de recolección de contenido web capaz de satisfacer las necesidades de información de los productos MOCIC, GEWEB y FARO, así como de futuros productos que lo requieran.

Finalmente, los autores del presente Trabajo de Diploma consideran que la idea que se propuso defender fue ratificada; es decir, el contenido web suministrado a productos informáticos que lo requieran debe ser recolectado mediante un sistema automatizado.

RECOMENDACIONES

- Desarrollar en futuras versiones de SINIBOT las siguientes funcionalidades:
  - navegar a través de un proxy que requiera autenticación.
  - establecer desde la interfaz web todos los parámetros de configuración que brinda Larbin.
  - leer las parámetros de configuración directamente desde la base de datos, eliminando así la actual dependencia de ficheros de configuración.
  - continuar un recorrido detenido.
- Desarrollar un sistema distribuido de recolección de contenido web, utilizando para ello la versión v0.1 de SINIBOT o una versión superior.
- Desarrollar una nueva versión del instalador que presente una interfaz gráfica, haciéndolo más amigable para el usuario.
- Desarrollar una nueva versión de la interfaz web que permita programar varios recorridos por diferentes administradores.
- Probar el producto a gran escala con el objetivo de asegurar su correcto funcionamiento.

REFERENCIAS BIBLIOGRÁFICAS

BAEZA-YATES, R. 1999. *Desenredando la Madeja.*
[fecha de consulta: 15-Febrero-2009].
Disponible en:
http://www.ati.es/novatica/2000/145/ricbae-145.pdf

BAEZA-YATES, R., CASTILLO, C. y GRAELLS, E. 2007. *Características de la Web Chilena 2006*. [en línea].
Santiago, Chile: Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas, Departamento de Ciencias de la Computación. 2007.
Disponible en:
http://www.ciw.cl/material/web_chilena_2006/Web_Chilena2006.pdf

BAEZA-YATES, R., CASTILLO, C., MARÍN, M. y RODRÍGUEZ, A. 2005. *Crawling a Country: Better Strategies than Breadth-First for Web Page Ordering.*
[fecha de consulta: 13-Febrero-2009].
Disponible en:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.6585&rep=rep1&type=pdf

BAEZA-YATES, R y RIBEIRO-NETO, B. 1999. *Modern Information Retrieval.* [en línea].
Santiago, Chile: Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas, Departamento de Ciencias de la Computación, 1999.
Disponible en:
http://sunsite.dcc.uchile.cl/irbook/

BERGMAN, M. K. 2001. *White Paper: The Deep Web: Surfacing Hidden Value.*
Ann Arbor, Michigan, Estados Unidos: Universidad de Michigan, Departamento de Publicaciones, 2001. [fecha de consulta: 20-Febrero-2009].
Disponible en:
http://dx.doi.org/10.3998/3336451.0007.104

BOJO, C. [et al]. 2004. *Internet Visible e Invisible: búsqueda y selección de recursos de información en Ciencias de la Salud.* [en línea].
Madrid, Instituto de Salud Carlos III. 2004.
Disponible en:
http://bvs.isciii.es/mono/pdf/BNCS_01.pdf

CANÓS J. H., LETELIER P y PENADÉS M. C. 2007. *Metodologías Ágiles en el desarrollo de Software.* [fecha de consulta: 10-Marzo-2009]
Disponible en:
http://www.willydev.net/descargas/prev/TodoAgil.Pdf

CAO, F., JIANG D. y SINGH J. P. 2003. *Scheduling Web Crawl for Better Performance and Quality.* [fecha de consulta: 5-Febrero-2009].
Disponible en:
ftp://ftp.cs.princeton.edu/techreports/2003/682.pdf

CASTILLO, C. *Effective Web Crawling.* Tesis Doctoral, Santiago, Chile: Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas, Departamento de Ciencias de la Computación, 2004. [191] p.
Disponible en:
http://www.chato.cl/papers/crawling_thesis/effective_web_crawling.pdf

CERN (2008). *Where the web was born.* [fecha de consulta: 12-Febrero-2009].
Disponible en:
http://public.web.cern.ch/public/en/About/Web-en.html

CHO, J. y GARCÍA-MOILNA, H. 2002. *Parallel Crawlers.*
[fecha de consulta: 27-Febrero-2009].
Disponible en:
http://oak.cs.ucla.edu/~cho/papers/cho-parallel.pdf

COMSCORE (2008). *comScore Releases June 2008 U.S. Search Engine Rankings.*

[fecha de consulta: 15-Febrero-2009].
Disponible en: http://www.comscore.com/press/release.asp?press=2337

DELGADO, A. M. 1998. *Mecanismos para la recuperación de información en la WWW.* Memoria de Investigación para la obtención de la suficiencia investigadora. Palma de Mallorca. España: Universidad de las Islas Baleares, 1998. [84] p.
Disponible en:
http://servidorti.uib.es/adelaida/tice/modul6/memfin.pdf

FERNÁNDEZ, A. 2001. *Diagrama de Componentes.*
[fecha de consulta: 05-Marzo-2009].
Disponible en:
http://tvdi.det.uvigo.es/~avilas/UML/node49.html

GRAY, M. 1995. *Measuring the Growth of the Web.*
[fecha de consulta: 22-Febrero-2009].
Disponible en:
http://www.mit.edu/people/mkgray/growth/

GUTIÉRREZ, C. [et al]. 2008. *Cómo funciona La Web.* [en línea]. Santiago, Chile: Universidad de Chile, Centro de Investigación de la Web.
Disponible en:
http://www.ciw.cl/libroWeb-NV.pdf

KOSTER, M. 1996. *A Standard For Robot Exclusion*.
[fecha de consulta: 25-Febrero-2009]
Disponible en:
http://www.robotstxt.org/orig.html

MARTÍNEZ-MÉNDEZ, F. J. *Propuesta y desarrollo de un modelo para la evaluación de la recuperación de información en Internet*. Tesis Doctoral. Murcia, España: Universidad de Murcia, 2002. [298] p.
Disponible en:

http://www.tdr.cesca.es/TESIS_UM/AVAILABLE/TDR-0919106-141210//tesis-javiermartinez.pdf

MORENO JIMENEZ, P. M. 2005. *Estrategias y mecanismos de búsqueda en la web invisible.* [en línea].
Disponible en:
http://www.umng.edu.co/www/resources/web%20invisible.pdf

NIELSEN, J. 2004. *Statistics for Traffic Referred by Search Engines and Navigation Directories to Useit.* [fecha de consulta: 04-Febrero-2009].
Disponible en:
http://www.useit.com/about/searchreferrals.html

PEÑALVER, G. 2008. *MA-GMPR-UR2 Metodología ágil para proyectos de software libre.* Tesis Diploma. Cuidad de La Habana, Cuba: Universidad de las Ciencias Informáticas, Facultad 10, 2008. [94] p.
Disponible en:
http://bibliodoc.uci.cu/TD/TD_1309_08.pdf

PEÑAS, A. *Un sistema interactivo y multilingüe de búsqueda textual basado en técnicas lingüísticas.* Tesis Doctoral. Madrid, España: Universidad Nacional de Educación a Distancia, 2002. [257] p.
Disponible en:
http://nlp.uned.es/~anselmo/tesis/

PÉREZ, Y y RIPOLL, D. A. *Asignación automatizada de categorías temáticas al contenido textual de documentos HTML*. Tesis Diploma. Ciudad de La Habana, Cuba: Universidad de las Ciencias Informáticas, Facultad 10, 2008. [94] p.
Disponible en:
http://bibliodoc.uci.cu/TD/TD_1258_08.pdf

ROBOTSTXT (2009). The Web Robots Page. *Frequently Asked Questions.* [fecha de consulta: 17-Febrero-2009].

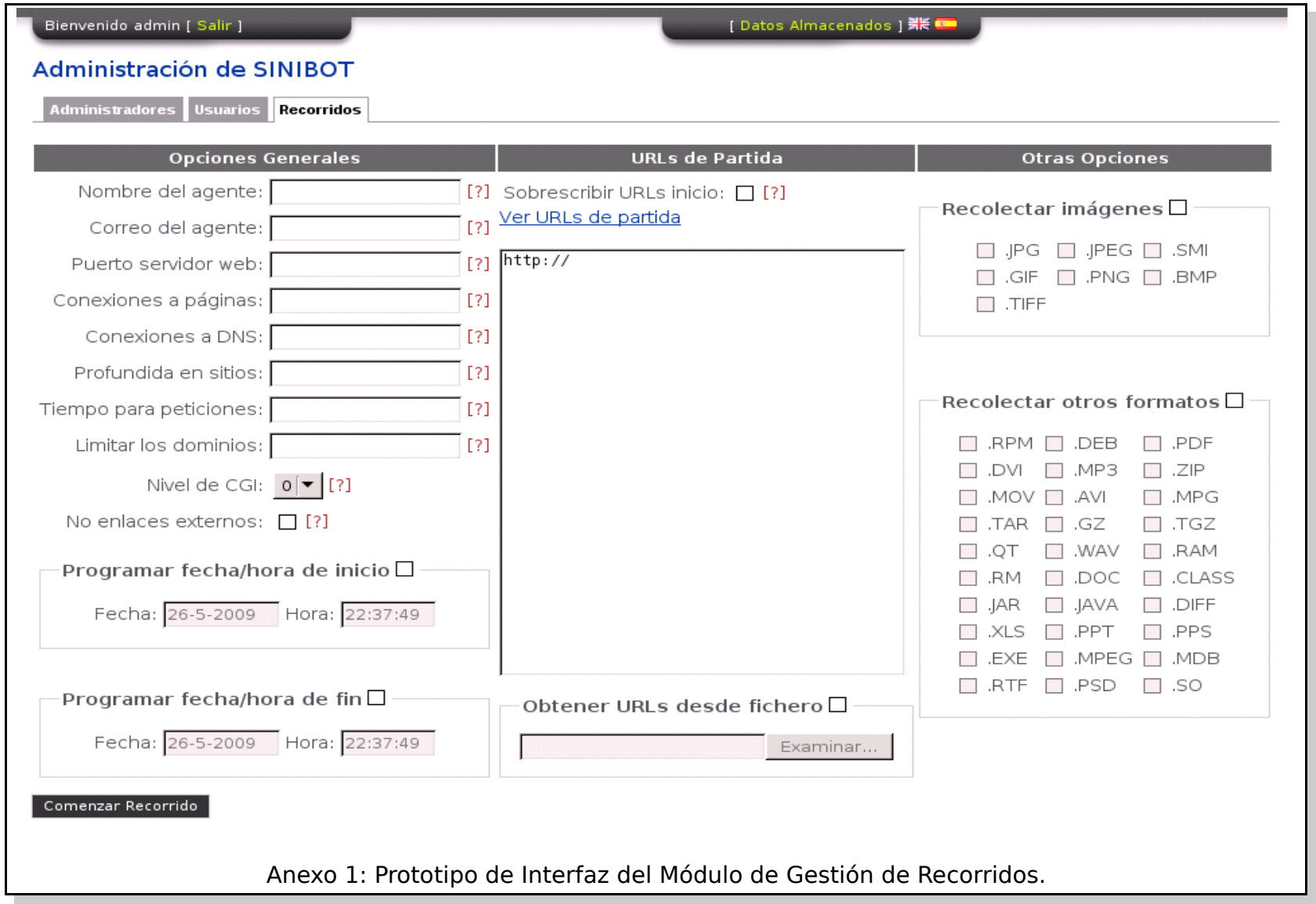
Disponible en:
http://www.robotstxt.org/faq.html

RODRIGUEZ, M. 2007. *Introducción de procedimientos ágiles en la producción de software en la Facultad 7 de la Universidad de las Ciencias Informáticas.* Tesis Diploma. Cuidad de La Habana, Cuba: Universidad de las Ciencias Informáticas, Facultad 7, 2007. [99] p.
Disponible en:
http://bibliodoc.uci.cu/TD/TD_0693_07.pdf

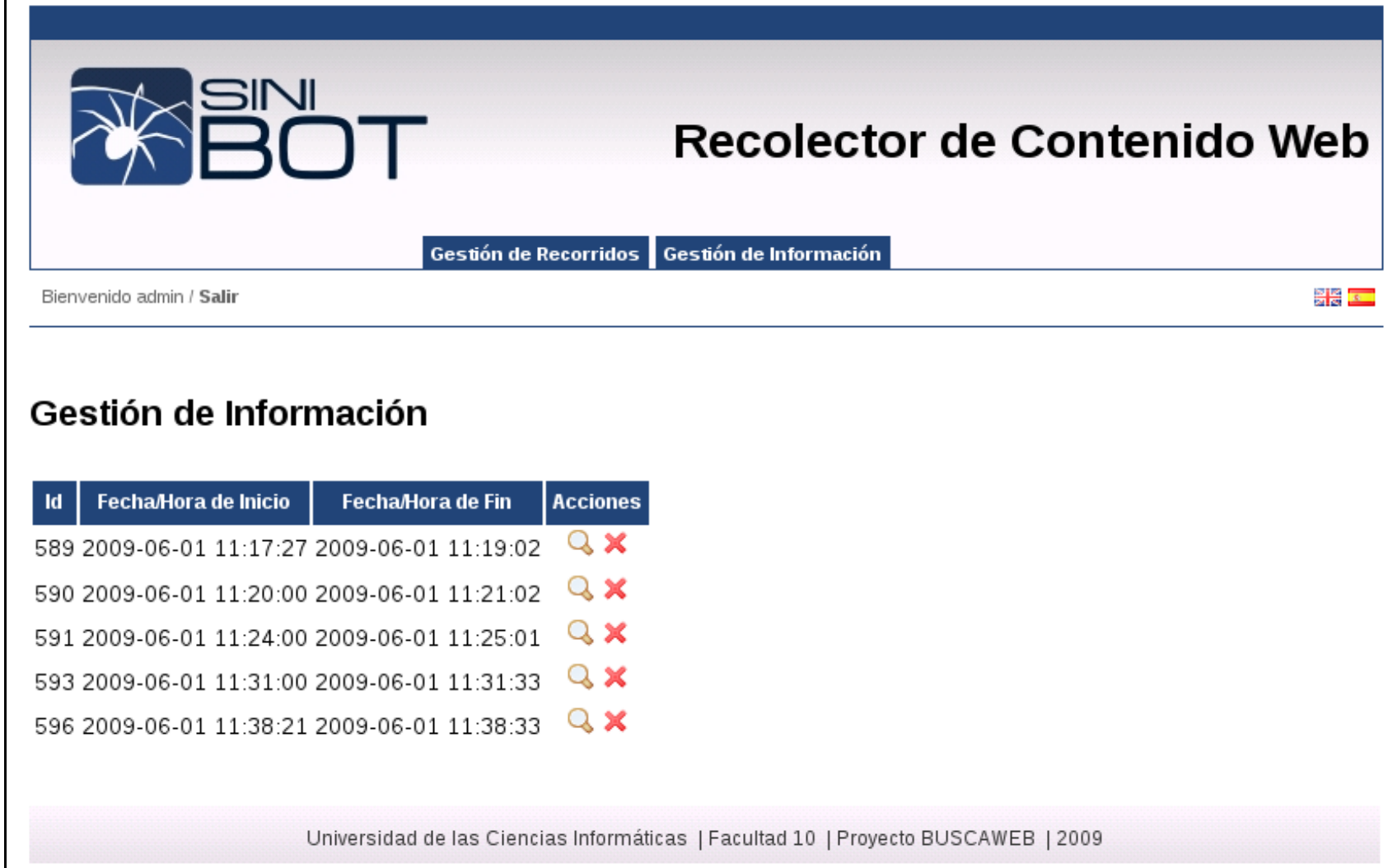
RODRÍGUEZ, R. (2003). *Motores de búsqueda sobre salud en Internet.* [fecha de consulta: 18-Noviembre-2008].
Disponible en:
http://bvs.sld.cu/revistas/aci/vol11_5_03/aci02503.htm

W3C (2009). World Wide Web Consortium. *Guía Breve de Web Semántica.* [fecha de consulta: 5-Febrero-2009].
Disponible en:
http://www.w3c.es/divulgacion/guiasbreves/WebSemantica

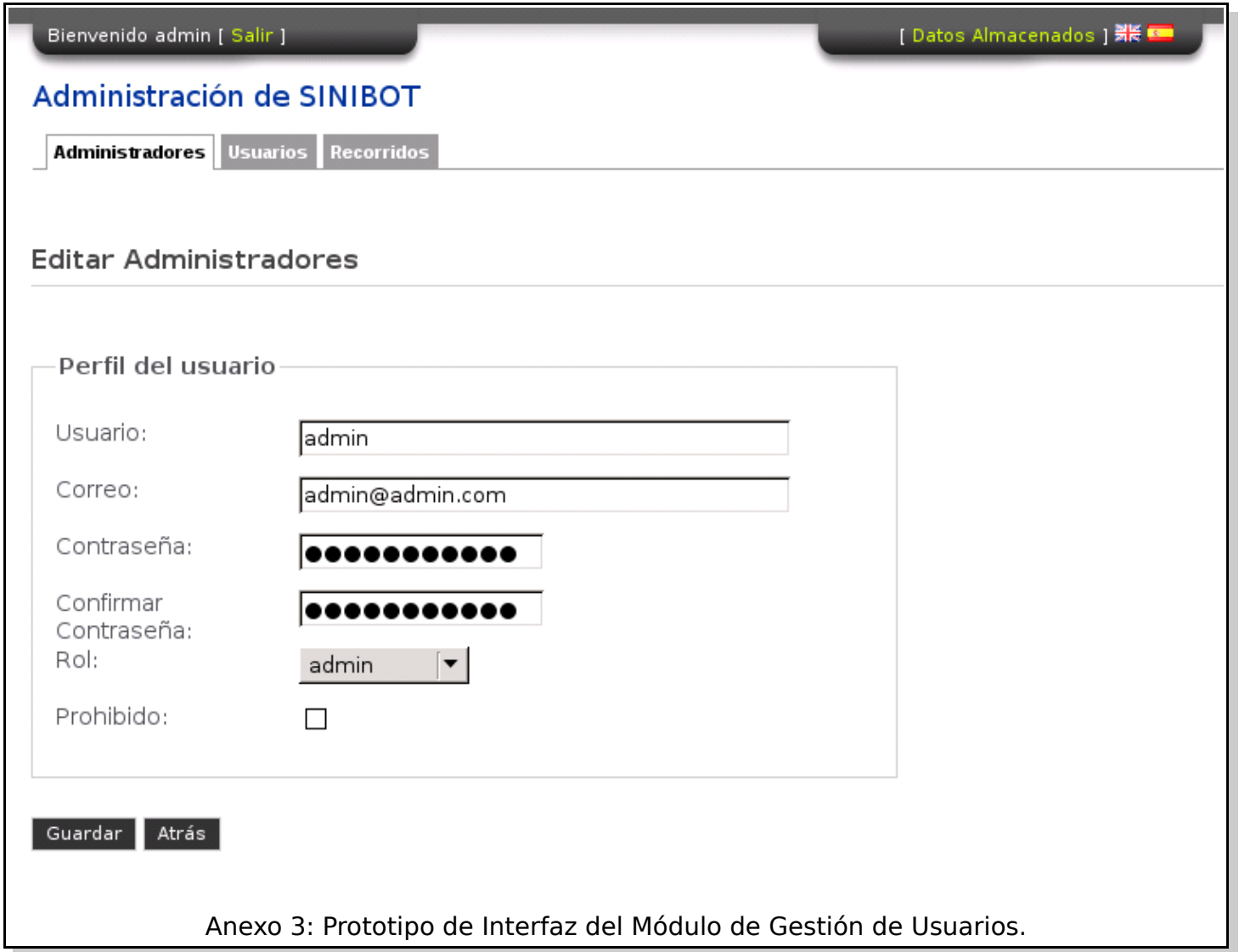
ANEXOS



Anexo 1: Prototipo de Interfaz del Módulo de Gestión de Recorridos.



Anexo 2: Prototipo de Interfaz del Módulo de Gestión de Información.



Anexo 3: Prototipo de Interfaz del Módulo de Gestión de Usuarios.

GLOSARIO DE TÉRMINOS

- **Artefacto** – Producto tangible resultante del proceso de desarrollo de *software* y se refiere, en ocasiones, a un producto terminado, como el código o el ejecutable, pero habitualmente representa la documentación generada a lo largo del desarrollo del producto, en lugar del producto en sí.
- **Contenido web** - Todo documento, imagen, animación, sonido, video, que pueda ser transmitido y ejecutado a través de un navegador.
- **Enlace** - Elemento de una página web que hace referencia a otro documento.
- ***Framework*** – En el desarrollo de *software,* es una estructura de soporte definida, mediante la cual otro proyecto puede ser organizado y desarrollado. Puede incluir soporte de programas, librerías y un lenguaje interpretado, entre otras herramientas, para ayudar a desarrollar y unir los componentes de un proyecto.
- **GNU/Linux** – Sistema operativo similar a Unix que utiliza como base las herramientas de sistema de GNU y el kernel de Linux.
- ***Hardware*** – Corresponde a todas las características físicas y tangibles de una computadora: sus componentes eléctricos, electrónicos, electromecánicos y mecánicos; sus cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado.
- **Internet** – Red de ordenadores a nivel mundial que ofrece distintos servicios, como el envío y recepción de correo electrónico, la posibilidad de ver información en las páginas web, de participar en foros de discusión, de enviar y recibir ficheros, de charlar en tiempo real, entre otros.
- **Intranet** - Red de ordenadores privados que utiliza tecnología Internet para compartir de forma segura cualquier información o programa del sistema operativo, evitando que cualquier usuario de Internet pueda ingresar.
- **Iteraciones** – En el contexto de un proyecto, se refieren a la técnica de desarrollar y entregar componentes incrementales de funcionalidades. Una iteración resulta en uno o más paquetes atómicos y completos de trabajo del proyecto que pueda realizar alguna función tangible. Múltiples iteraciones contribuyen a crear un producto completamente integrado.

- **Metodología ágil** – Paradigma de desarrollo de *software* basado en procesos ágiles que evita los tortuosos y burocráticos caminos de las metodologías tradicionales, enfocándose para ello en la gente y los resultados.
- **Metodología de desarrollo** – Versión amplia y detallada de un ciclo de vida completo de desarrollo de *software* que incluye reglas, procedimientos, métodos, herramientas, funciones individuales y en grupo por cada tarea, productos resultantes y normas de calidad.
- **Requisitos** – Capacidades, condiciones o cualidades que el sistema debe cumplir y tener.
- ***Script*** – Guión o conjunto de instrucciones que permite la automatización de tareas creando pequeñas utilidades. Usualmente es un archivo de texto ejecutado por un intérprete, muy utilizados en la administración de sistemas Unix.
- ***Software* libre** – *Software* o programa que brinda libertad a los usuarios sobre el producto adquirido y por tanto, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente**.**
- ***Software* privativo** – *Software* o programa al que por medio de una licencia no libre, se le ha privado de ciertas libertades de uso, modificación o distribución.
- ***Suite* de herramientas** – Conjunto de *software* o aplicaciones destinadas a una actividad específica.
- **Unix** – Sistema operativo portable, multitarea y multiusuario, desarrollado en 1969 por un grupo de empleados de los laboratorios Bell de AT&T.
- **URL –** Secuencia de caracteres de acuerdo a un formato estándar, que se usa para nombrar recursos como documentos o imágenes en Internet, según su localización.
- **Usuario *root*** - En sistemas operativos del tipo Unix, *root* es el nombre convencional de la cuenta de usuario que posee todos los derechos de lectura, escritura y ejecución sobre cualquier componente del sistema.
- ***World Wide Web*** – Sistema de documentos de hipertexto, enlazados y accesibles a través de Internet.