UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
APLICACIONES CON LA TECNOLOGÍA INTERNET

Desarrollo del Módulo de Acceso a los Contenidos Preservados en formato WARC para un Prototipo de Archivo Web de Venezuela

Trabajo Especial de Grado presentado ante la ilustre
Universidad Central de Venezuela por los
Br. Mantura Kabchi, CI: 19223964
Br. Miguel Martínez, CI: 19294704
Para optar al título de Licenciado en Computación
Tutora: **Profa. Mercy Ospina**

Caracas, Marzo de 2014

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN

ACTA

Quienes suscriben, miembros del jurado designado por el Consejo de la Escuela de Computación, para examinar el Trabajo Especial de Grado titulado "**Desarrollo del Módulo de Acceso a los Contenidos Preservados en formato WARC para un Prototipo de Archivo Web de Venezuela**" y presentado por los bachilleres: Br. Mantura Teresa Kabchi Abchi, C.I: 19.223.964, y Br. Miguel Ángel Martínez Farías, C.I: 19.294.704, a los fines de optar al título de **Licenciado en Computación**, dejamos constancia de lo siguiente:

Leído como fue dicho trabajo, por cada uno de los miembros del jurado, se fijó el día 21 de Marzo de 2014, a las 11:00 horas, para que los autores lo defendieran en forma pública, lo que estos hicieron en la Sala PBIII de la Escuela de Computación, mediante una presentación oral de su contenido, luego de lo cual respondieron a las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobar con la nota de ____ puntos.

En fe de lo cual se levanta la presente Acta, en Caracas el día ___ de _____ de ____.

_____
Profa. Mercy Ospina
(Tutora)

_____
Profa Claudia León
(Jurado)

_____
Profa Concettina Di Vasta
(Jurado)

AGRADECIMIENTOS

A mis padres, Leticia y Armando, por siempre apoyar y respetar mis decisiones, por guiarme, darme y enseñarme lo necesario para cumplir cada meta que me he propuesto.

A mis hermanos, Alejandro, David y Alfredo porque siempre he podido contar con ellos y son ejemplo para mí.

A mi familia, a mis abuelos por ser ejemplos de perseverancia; a mi tío Carlos por estar pendiente de mí y darme uno de los instrumentos más importantes de mi carrera.

A Jeniree, por estar a mi lado durante 5 años de carrera, por ser mi mejor amiga, siempre motivándome a salir adelante. A la familia Sánchez por ser un segundo hogar.

A Mantura, una de mis mejores amigas, mi compañera de Seminario y Trabajo Especial de Grado, gracias por aguantarme y por todo el aprendizaje que juntos hemos logrado durante este año.

A Manuela, por estar desde el principio.

A mis amigos, Ernesto, Juan Manuel, Ciro, Scarlett y Antonio por ser como hermanos para mí; Cheto, Claudia, Luis Eduardo y Meilyn, por los años y momentos compartidos.

Miguel Martínez

A mis padres, Teresa y Eduardo, por ser A1, como dice mi viejo. La mejor guía que se puede tener.

A mis hermanos, Carlos, Emilio y David, por ser justamente eso, mis hermanos, aunque peleemos cuatro de cada cinco días.

A mi abuela por quererme como a su nieta favorita, por todas las veces que me regañó y me consintió.

A Migue, el mejor compañero de Seminario y Trabajo Especial de Grado. Ya es el final, y estamos hartos, pero nadie mejor que tu para haber pasado un año y medio en esta pesadilla.

A Karina y mi Morocho, mis mejores amigos. A ustedes cualquier cosa que les diga será redundante.

A mis amigos, Javi, Roberto, Ronnie, Gaby, Iñaki, Sheila, Alexander, Juan, Sarmi, Erika, Aintzane, Mariana, por todo lo que hemos compartido.

A mis padrinos, Salva y Odalys, por asumir con gusto la carga de ser mi segunda familia.

Mantura Kabchi

A Dios por permitirnos estar aquí cumpliendo esta meta.

A Mercy Ospina, por su tutoría, por habernos confiado este proyecto y habernos ayudado durante todo este tiempo.

A Concettina Di Vasta, por sus consejos y guías, siempre dispuesta y con ánimos.

A Lorena, por su disposición y ayuda incondicional.

A Roberto, por su asesoría creativa durante el Trabajo Especial de Grado.

A nuestros amigos de la universidad, Manuela, Oswaldo, Juan Raúl, José Alberto, Néstor, Daniela, Félix, Mitchell, Huáscar, Luis, por ser fieles desde el papelito y el tótem hasta el día de hoy.

Mantura y Miguel

Universidad Central de Venezuela.
Facultad de Ciencias
Escuela de Computación
Aplicaciones con la Tecnología Internet

Desarrollo del Módulo de Acceso a los Contenidos Preservados en formato WARC para un Prototipo de Archivo Web de Venezuela

Autor: Mantura Kabchi.
Miguel Martínez.

Tutora: Profa. Mercy Ospina.

Fecha: 10 de marzo de 2014.

RESUMEN

El patrimonio cultural es todo aquello que ha sido relevante para la historia de una sociedad, comunidad o pueblo y que forma parte de sus tradiciones y educación; este ha sido dividido en patrimonio tangible e intangible y dentro de este último se encuentra el patrimonio digital el cual abarca sitios Web, documentos, libros digitales, materiales multimedia, entre otros. Este patrimonio no es autopreservable, es decir tiende a desaparecer de la vista pública, por lo que se han venido desarrollando sistemas de información conocidos como Archivos Web cuyo propósito es preservar el patrimonio digital Web. Actualmente se está desarrollando un prototipo de Archivo Web en la Escuela de Computación de la Facultad de Ciencias de la Universidad Central de Venezuela, en el cual se están almacenando sitios web de prueba en un formato de archivo llamado *WARC file*, el cual es un contenedor de documentos Web estandarizado y desarrollado con la finalidad de soportar la preservación. El objetivo de este Trabajo Especial de Grado, el cual forma parte del desarrollo de este Prototipo, es crear una aplicación Web para el acceso y despliegue del contenido preservado en formato WARC, el cual puede ser accedido y navegado por los usuarios como si se encontraran en el sitio original, en su versión original. Para la lectura y extracción del contenido de dicho formato se utilizó una herramienta de código abierto denominada WARCTools, por lo que se considera que este trabajo es además un aporte para promover el uso del formato WARC y de los WARCTools.

Palabras Claves: Archivo Web, preservación Web, modelo OAIS, Formato WARC, despliegue.

Índice

Índice de Figuras

Índice de Tablas

Índice de Gráficas

Introducción

La preservación de información, bien sea materiales escritos o de transmisión oral, siempre ha sido de gran importancia para el ser humano y su desarrollo integral. De esta manera se puede conocer el pasado, el origen de aquello que nos rodea (cultura, tecnología, entre otros), documentar los nuevos descubrimientos, mantenerse al tanto de lo que ocurre en el día a día, y predecir comportamientos futuros de cualquier tipo de elemento que pueda progresar con el tiempo.

Tradicionalmente los instrumentos de preservación de contenido por definición son el papel y la transmisión oral a través de las generaciones. Las bibliotecas, entidades organizacionales dirigidas a la conservación de contenido, las tradiciones populares, las celebraciones, entre otros, permiten aún conservar y resguardar la información a través del tiempo. Pero, con la aparición de la World Wide Web y su crecimiento exponencial ha surgido la necesidad de almacenar y preservar el contenido que se ofrece en ésta. Este tipo de preservación es llamada “Preservación Web” y ha cobrado importancia para generaciones presentes y futuras.

Como el contenido en la Web es tan vasto, requiere de distintas estrategias, tecnologías y técnicas para lograr conservarlo y a la vez presentárselo a los millones de usuarios que puedan necesitarla en un mismo o en distintos momentos en el tiempo. Por ello no existe una iniciativa única de preservación web sino que se han desarrollado una serie de iniciativas en diversos países de Europa, Oceanía y Norte América que preservan una porción de la misma, siendo, además, respaldadas por la UNESCO (Gomes, Miranda, & Costa, 2010). Estas iniciativas presentan una serie de herramientas, tecnologías, estrategias y técnicas de adquisición, almacenamiento, indexación y acceso a la información que se pueden utilizar y combinar para lograr un alto rendimiento, eficiencia y eficacia al momento de preservar el contenido digital de los diversos sitios Web que lo requieran definiendo así el concepto de Archivo para la preservación Web.

El objetivo de este Trabajo Especial de Grado (TEG) es implementar el módulo de acceso que permita ingresar al contenido Web previamente almacenado y desplegarlo recreando una versión en el tiempo del sitio original. Este módulo forma parte del “Prototipo de Archivo Web para la Preservación del Patrimonio Web de Venezuela”, y será desarrollado siguiendo la metodología XP (eXtreme Programming). Este prototipo busca brindar una solución a la problemática que representa la pérdida de información que reside en la Web y que puede ser valiosa para el contexto de Venezuela.

El documento está dividido en cuatro capítulos; en el capítulo I “Planteamiento del Problema” se describe el problema al que se le da solución; en el capítulo II “Marco Teórico Conceptual” se definen todos los conceptos relevantes al proceso de preservación digital y a la Web, además de las diversas técnicas y estrategias para lograr la conservación del contenido; el capítulo III “Marco Aplicativo” describe la adaptación de la metodología XP al proyecto realizado, historias de usuario, iteraciones e implementación del mismo. Y por último Conclusiones y Referencias Bibliográficas.

CAPÍTULO I

PLANTEAMIENTO DEL PROBLEMA

1.1. Planteamiento del Problema

La Web y el Internet influyen de gran manera en el desarrollo de nuestra generación. Desde el nacimiento de la Web, la llegada de los documentos digitales, la aparición de la Web 2.0, entre otras tecnologías y herramientas originadas con el auge del Internet, la cantidad de información que se puede encontrar al navegar la red es muy extensa y variada por lo que es utilizada por niños, adolescentes y adultos para tanto estudiar e investigar como para socializar tomando la misma un lugar fundamental para las comunicaciones y la obtención de información en el mundo moderno. Las personas, empresas o instituciones la utilizan para buscar información, para expresar sus opiniones y sentimientos, para publicar investigaciones, relacionarse con otras personas, entre otras funciones. Por lo tanto, la información que puede ofrecer es extremadamente amplia y en muchos casos muy rica en contenido y en diversidad, por tanto, ésta es considerada un patrimonio cultural y debe ser tratado como tal.

El problema, en cuanto a la preservación Web, surge cuando la información comienza a desaparecer por limitaciones del sitio que la sirve o de la red (espacio, tiempos de respuesta y de transmisión, entre otros). Mucha información, que puede ser valiosa para un grupo o comunidad, para quien la necesite, desaparece porque o es suplantada por nueva, o el sitio en sí es eliminado por problemas con el dominio, o por asuntos internos de administración de la página, o quizás porque la organización asociada al mismo no continua con sus operaciones. Todo esto lleva a que la información se pierda y en muchos casos no pueda ser recuperada (Masanés, 2006).

Existen estudios que documentan la naturaleza efímera de los recursos Web derrotando la afirmación de que la Web es un medio de autopreservación (Spinellis, 2003). Estos estudios se centran en la disponibilidad de recursos en la misma URL, no los posibles cambios que puede sufrir. Algunos también verificaron el contenido y midieron la tasa de cambio. Cho y García Molina (2000) encontraron una vida media (período durante el cual la mitad de las páginas desaparecerán) de 50 días para las páginas Web, y Fetterly, Manasse, Najork, y Wiener (2003) mostraron cómo este tipo de cambio está relacionado con el tamaño y la ubicación del contenido.

Estas son algunas de las razones por lo que han surgido iniciativas en diversos países, que buscan preservar el contenido Web y permitirles a los usuarios acceder a él aunque el sitio que originó la información desaparezca o cambie, la otra razón es poder registrar los cambios en el tiempo que permitan hacer estudios sobre los mismos. Estas iniciativas están basadas en el concepto de archivo histórico y han sido desarrolladas por las bibliotecas, y en muchos casos son las bibliotecas nacionales

o universitarias las que promueven la preservación Web o digital. (Gomes, Miranda, & Costa, 2010) actualizado por (León, Ospina, 2013).

Todas las iniciativas estudiadas poseen características comunes y generales para poder conservar la información, las cuales pueden verse como las tareas de un archivo: adquirir el contenido y transformarlo a un formato estándar para luego poder almacenarlo, indexarlo para poder filtrar las búsquedas (por ejemplo, por categorías, por el nombre del sitio Web, entre otros) y finalmente crear una aplicación Web que permita al usuario el acceso y el despliegue de la información lo más cercano posible al sitio original.

En el año 2003 la Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura, o UNESCO por sus siglas en inglés, define las directrices para la preservación digital, por lo que a partir de ese momento surgen estándares, estrategias y formatos de archivo para la conservación del patrimonio digital, que han sido adoptados por las diversas iniciativas que nacieron antes y después de ese momento. Sin embargo, estas directrices no se han masificado y por lo tanto en muchos casos, carecen de documentación y de comercialización (UNESCO, 2003).

En Latinoamérica, y más específicamente en Venezuela, no se conocen iniciativas oficiales de preservación Web. Este TEG está enmarcado dentro del proyecto "Desarrollo del prototipo de Archivo Web para la preservación del patrimonio Web de Venezuela" de código PI-03-8139-2011/P financiado por el Consejo de Desarrollo Científico y Humanístico (CDCH) de la Universidad Central de Venezuela.

El desarrollo del prototipo de Archivo Web de Venezuela posee gran relevancia en el ámbito social ya que representa una fuente de conocimientos amplia, para las presentes y próximas generaciones, que en un futuro podría perderse si no surgen inquietudes hacia esta clase de iniciativas. Aunado a esto, está presente la motivación de contribuir a la masificación de formatos estándares para Archivos Web, como el formato de archivos WARC el cual es un formato publicado en el 2009 por la Organización Internacional de Estandarización (International Organization for Standardization, ISO), que permite concatenar y almacenar múltiples recursos u objetos de datos que contienen cabeceras de textos y bloques de datos en un solo archivo de gran tamaño y mantener la integridad de los contenidos almacenados, pero que ha presentado problemas para consolidar su uso por la dificultad que representa el acceso y despliegue del mismo.

Parte importante de este prototipo es permitir el acceso de los contenidos almacenados en estos formatos, que permitan visualizarlos en su formato original, sin embargo el acceso al WARC y su despliegue resulta una tarea compleja por varios factores, entre los cuales:

- Poca documentación de libre acceso, ya que existen los manuales de la ISO pero estos son pagos.

- Librerías para extracción de contenidos que se encuentran en desarrollo actualmente y además presentan poca documentación.
- No se consiguieron aplicaciones libres para el acceso que usen el formato WARC, por lo que existe poca experiencia documentada al respecto. Aquellas a las que se tuvo acceso soportan otros formatos que no son estándares ISO como el ARC (ver sección Antecedentes de Archivado Web del Capítulo II).

En este TEG se propone desarrollar un modulo que permita el acceso y navegación a contenidos web almacenados en formato WARC para que puedan ser visualizados en su forma original, usando para ello tecnología de software libre, como parte del prototipo de Archivo Web para sitios web en Venezuela.

1.2. Objetivo General y Específico:

El objetivo de este TEG es crear una aplicación Web para el acceso y despliegue de contenido Web almacenado en formato WARC, como parte del prototipo de Archivo Web de Venezuela.

Los objetivos específicos del TEG son:

1. Identificar los requerimientos principales (funcionales y no funcionales) del módulo de acceso y despliegue de archivos WARC para el prototipo de archivado de páginas Web en Venezuela.
2. Desarrollar el módulo de acceso y despliegue a los contenidos almacenados como una aplicación Web utilizando la metodología de desarrollo elegida y tecnología de software libre
3. Integrar el módulo desarrollado con el módulo de almacenamiento e indexación (García, J. y Rivero, L., 2013) para la búsqueda optimizada de los archivos y la recreación del sitio Web solicitado por el usuario.
4. Realizar pruebas unitarias, de aceptación y de validación.

1.3. Propuesta de la solución

Previo a esta investigación, se desarrollaron los módulos de Adquisición e Indexación del prototipo de Archivo Web (García, J. y Rivero, L., 2013) utilizando como herramientas para dichas funciones a Heritrix y Solr respectivamente, las cuales permiten almacenar e indexar los archivos de formato WARC. El hecho de contar con contenido ya almacenado permite proceder a la creación del módulo de acceso a estos contenidos, a través de una aplicación Web, que recree el sitio original de acuerdo a una versión en el tiempo.

Para ello el módulo de acceso debe interactuar con el módulo de indexación, este último recibe un URL y da como respuesta las versiones almacenadas de dicho URL. El usuario debe introducir el URL del sitio Web a buscar, luego recibe un listado de las versiones del sitio, las cuales pueden ser

representadas al usuario mediante un calendario, donde se le permita seleccionar la versión que desee, enviándole una nueva petición al módulo de Indexación el cual proveerá la ruta del archivo WARC que corresponde a la versión seleccionada.

Con esta ruta se procede a extraer el contenido del archivo WARC para mostrarlo a través del navegador web y que el usuario pueda navegar a través de sus enlaces y documentos. Este flujo de procesos se encuentra representado a través de la siguiente Arquitectura de Información:

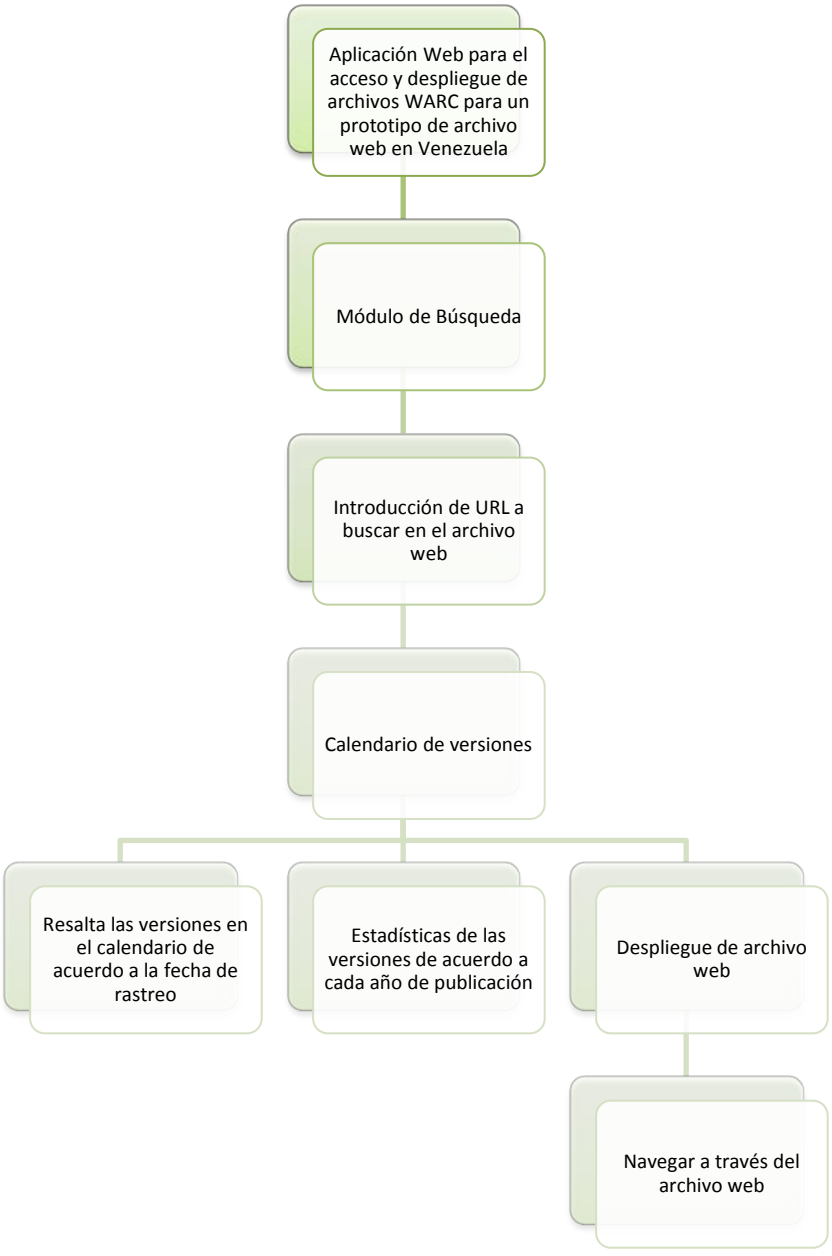


Figura 1. Arquitectura de Información

CAPÍTULO II

MARCO TEÓRICO CONCEPTUAL

En este capítulo se describen los conceptos que permiten entender el trabajo realizado. Se presenta el material referente a la Preservación y el concepto de Archivado Web así como también el modelo utilizado como base para el desarrollo del Archivado y la implementación del módulo de acceso.

2.1. Patrimonio Cultural y Digital

El patrimonio cultural está formado por los bienes que la historia le ha dejado a la nación y aquellos que en la actualidad son creados y que las comunidades o sociedades de un determinado pueblo o país les otorga un significado científico, histórico, estético o simbólico. El patrimonio cultural se divide en dos tipos: tangible (mueble e inmueble) e intangible (nagore.otsoa.net).

Patrimonio tangible mueble: Está compuesto por objetos arqueológicos, artísticos, etnográficos, tecnológicos y religiosos que constituyen hallazgos importantes para las ciencias del arte y de la tecnología y para la diversidad cultural del área. Por ejemplo: libros, manuscritos, fotografías, películas, entre otros.

Patrimonio tangible inmueble: este tipo de patrimonio cultural está constituido por los lugares, edificaciones, obras industriales y de ingeniería, zonas típicas y monumentos reconocidos desde el punto de vista arquitectónico que representan o tienen una significado histórico o simbólico y que han sido registrados como tales.

Patrimonio intangible: este patrimonio se refiere a aquel que es invisible y que reside en las creencias, modos de vivir, y rituales de las distintas culturas del mundo. Por lo tanto el patrimonio no solo se refiere a los bienes sino también a aquellas técnicas, saberes, espiritualidad que ha sobrevivido gracias a la tradición oral. Este patrimonio representa el “conjunto de rasgos distintivos, espirituales y materiales, intelectuales y afectivos que caracterizan una sociedad o un grupo social y que, más allá de las artes y de las letras, engloba los modos de vida, los derechos fundamentales del ser humano, los sistemas de valores, las tradiciones y las creencias”. Por ejemplo: mitos y leyendas, creencias religiosas, ritos, medicina tradicional, trajes que identifican a las regiones, fiestas propias del lugar, entre otros.

El patrimonio cultural puede ser preservado a través del tiempo, ya sea en museos, bibliotecas, álbumes, a través de la comunicación entre familia, amigos, con técnicas de conservación de estructuras u otros métodos que permiten que todo quede para las futuras generaciones y para que puedan aprender de su pasado.

Con la aparición de Internet se puede conseguir información acerca de una infinidad de temas actualizados ya sea basado en libros o en investigaciones hechas por personas, entes, universidades o empresas. Esto permite que Internet también se convierta en una herramienta que puede ser utilizada para preservar información relevante que puede ser aprovechado por generaciones futuras. El problema es que muchos sitios comienzan a desaparecer con el paso del tiempo o simplemente cambian porque así lo demanda la tecnología, teniendo que eliminar, por problemas de almacenamiento, información que podría ser útil para la formación de las personas. De aquí surge el interés de evitar que estas pérdidas sucedan y nacen los conceptos de patrimonio digital y preservación digital.

En marzo de 2003, la UNESCO publicó las Directrices para la Preservación del Patrimonio Digital preparado por la Biblioteca Nacional de Australia. El patrimonio de origen digital está formado por los materiales informáticos de valor perdurable dignos de ser conservados por generaciones futuras, por ejemplo publicaciones periódicas electrónicas, bases de datos distribuidas, páginas Web, entre otros. Todos estos ya forman parte del patrimonio cultural del mundo. (UNESCO, 2003)

También, en esta misma publicación hecha por la Biblioteca Nacional de Australia, surge el patrimonio de archivos digitales las cuales son aquellas colecciones digitales que pueden encontrarse en Internet y que poseen versiones históricas, son archivadas en bases de datos especializadas y con formatos estandarizados. Por ejemplo, una página Web que posea distintas versiones a través de los años y que pueda ser consultada hasta incluso encontrar su primera versión. Esto permite que la herencia cultural pueda ser accedida sin restricciones de tiempo o costo.

2.2. Preservación Digital y Archivado Web

Internet y la Web han influido de gran manera tanto en la informática como en la cultura y en la sociedad en general. En los últimos años se han roto las barreras de la distancia, la economía y se han abierto una gran variedad de oportunidades para todos aquellos que la utilicen buscando mejorar y progresar. De esta idea nace el concepto de preservación digital.

La preservación digital entonces es el conjunto de técnicas que pretenden recoger, almacenar y difundir información digital que perdure, esté disponible y pueda recuperarse a pesar de los inevitables cambios tecnológicos que se producen a velocidades vertiginosas. Esto requiere el compromiso de las instituciones y organizaciones que lleven a cabo proyectos de digitalización, las cuales deben tomar en cuenta el aspecto de la preservación a largo plazo. (REBUIN, 2009). Este tipo de preservación va más dirigido hacia aquel contenido que se encuentra bien estructurado y cuyo formato no posee variaciones marcadas (informes, trabajos de investigación, entre otros).

La preservación digital permite asegurar la accesibilidad y la recuperación de materiales ya sean producto de una digitalización o ya nacidos digitales. De aquí surge el concepto de archivado o archivos Web, los cuales son sistemas innovadores que adquieren, guardan y preservan información publicada en la Web. También permiten preservar documentos no-digitales que eventualmente fueron digitalizados y publicados en línea. (REBUIN, 2009)

Por su parte la preservación Web se refiere a aquella subclasificación de la preservación digital que va dirigida al contenido de páginas o sitios Web en Internet cuya estructura puede ser vista como la de un árbol y que puede variar de distintas maneras y estar formado por distintos tipos de documentos y formatos, como por ejemplo el Lenguaje de Marcado de Hipertexto (HTML, por sus siglas en inglés) o el de lenguaje Preprocesador de Hipertexto (PHP), lo que hace de su comportamiento un elemento impredecible ya que puede haber contenido que se genera dinámicamente de acuerdo a las peticiones del usuario.

La Web posee distintas características a nivel de preservación, las cuales se mencionarán a continuación (Masanés, 2006):

- **Cardinalidad:** se refiere al número de instancias de cada objeto o recurso Web a preservar. Aunque el servidor del sitio Web sea único, el número de copias del recurso es grande y aún más en la actualidad con los avances que se presentan en las tecnologías de almacenamiento. Por lo tanto la preservación estará supeditada a aquellas instancias de los recursos que han sido adaptadas para navegadores específicos o zonas geográficas localizadas.

- **Sistema de publicación activo:** la Web utiliza tres métodos para publicar su contenido: HTTP como protocolo de comunicación, HTML para la capa de presentación y URI para identificar los distintos recursos pertenecientes a un mismo sitio Web. Las publicaciones y su contenido dependen de la interacción permanente del autor de las mismas, el cual puede cambiar frecuentemente. Por lo tanto el archivado Web debe buscar la manera de separar el contenido de su creador y así las publicaciones puedan ser resistentes a la evolución de las tecnologías que causen que la información se vuelva obsoleta y sea eliminada. Para esto se debe copiar y archivar el contenido en una estructura separada a la que reside originalmente aunque esto limite su comportamiento ya que habrán funcionalidades del servidor que no se podrán replicar.

- **Bien cultural:** las publicaciones están disponibles desde cualquier lugar en el mundo que posea conexión de Internet, por lo tanto se puede decir que la Web posee aportes de distintas culturas y sociedades. Para la preservación de la Web se debe tomar en cuenta lo que pueda ser significativo y las consecuencias que esa información pueda tener para la comprensión futura de lo que ha sido la Web.

2.3. Antecedentes de Archivado Web

Las iniciativas de archivado Web poseen estándares tanto en tamaño como en alcance. En su gran mayoría los archivos contienen información acerca de su país de creación, aunque en algunos casos contienen información de entidades de distintos orígenes. La creación y mantenimiento de este tipo de archivadores es costosa y compleja. Se han realizado encuestas a bibliotecas para así justificar el uso de Archivos Web en las que se concluye que los beneficios son mayores que los costos. (Shiozaki, 2009)

La distribución aproximada de países que promueven y sirven estas iniciativas es la siguiente: 23 en Europa, 10 en Norte América, 6 en Asia y 3 en Oceanía. La mayoría de estos países gestionan 1 o 2 iniciativas a excepción de Estados Unidos que sirve 9 iniciativas de archivado Web. (Gomes, Miranda, & Costa, 2010) actualizado por (León, Ospina, 2013).

La primera de las iniciativas de archivado Web fue creada por Brewster Kale en el año 1996 en los Estados Unidos y llevó el nombre de Internet Archive (Archivo de Internet). Tres iniciativas le siguieron en ese mismo año: Australia's Web Archive (Archivo Web de Australia), Tasmanian's Web Archive (Archivo Web de Tasmania) y Kulturarw3 de Suecia. Solo otras cinco iniciativas surgieron en los siguientes 6 años: New Zealand WA (Nueva Zelanda, 1999), WebArchiv (República Checa, 2000), Library of Congress (EEUU, 2000), University of Michigan (EEUU, 2000) y OASIS (Korea, 2001). (Gomes, Miranda, & Costa, A survey on web archiving initiatives, 2010)

A continuación una comparativa entre algunas iniciativas de Archivado Web, basada en sus aspectos más relevantes. Las siguientes iniciativas fueron escogidas debido a su importancia en el ámbito de preservación Web. *Internet Archive* fue pionera para el desarrollo de la misma, al ser la primera librería digital oficial sin fines de lucro dedicada a almacenar la World Wide Web, además se encargó del desarrollo de *Wayback Machine*, herramienta de despliegue y acceso antecedente a la aplicación desarrollada durante este TEG. Por su parte PANDORA es una de las más antiguas y representa un estimado relevante de la cantidad de personas que pueden estar involucradas en un proyecto de este tipo, ya que es un programa colaborativo entre 11 agencias y además a partir de 2005 sus cosechas son conducidas por Internet Archive debido a la gran cantidad de información manejada. También se seleccionó *Library of Congress* por su sistema de búsqueda Avanzada basada en colecciones y metadatos, las cuales se quieren implementar en la aplicación a desarrollar para el presente TEG. Por último, se escogió para esta investigación al *Portuguese Web Archive* al ser una iniciativa de Archivo Web más actual, con una interfaz gráfica moderna y minimalista. Por lo tanto las iniciativas presentadas a continuación combinan entre ellas una serie de factores importantes que se quieren tomar en cuenta en conjunto para el desarrollo del Módulo de Acceso a los Contenidos Preservados en formato WARC para el Prototipo de Archivo Web de Venezuela.

Tabla 1. Comparativa entre iniciativas de Archivado Web

INICIATIVA / CARACTERÍSTICA	INTERNET ARCHIVE	PANDORA (PANDORA Australia's Web Archive)	LIBRARY OF CONGRESS WEB ARCHIVES	PORTUGUESE WEB ARCHIVE
PAÍS	Estados Unidos	Australia	Estados Unidos	Portugal
AÑO DE CREACIÓN	1996	1996	2000	2007
TECNOLOGÍAS	Heritrix, Wayback Machine, Nuthwax	Sistema digital de archivado PANDORA (PANDAS), HTTrack, DOSS	Heritrix, Wayback Machine, Digiboard (herramienta de permisos)	Heritrix, Wayback Machine, Nuthwax
FORMATO DE ARCHIVO	ARC/WARC	ARC/WARC	ARC/WARC	ARC
CONTENIDO DE ARCHIVADO (millones)	150.000	3.100	5	1.731
MÉTODOS DE ACCESO	Búsquedas por URL, Búsquedas por metadatos avanzada, Búsquedas de texto completo	Búsquedas por metadatos avanzada, Búsquedas de texto completo	Búsquedas por URL, Búsquedas por metadatos avanzada	Búsquedas por URL, Búsquedas por metadatos avanzada, Búsquedas de texto completo

Internet Archive utiliza una herramienta de acceso propia, como lo es Wayback Machine. Wayback es inicialmente una implementación del Wayback Machine de Internet Archive, de código abierto desarrollado en Java.

La versión actual en producción de Wayback Machine está implementada en Perl, y carece de mantenibilidad y extensibilidad, según Internet Archive (Internet Archive, 2011). Además, el código no es abierto. La principal motivación para la nueva versión en Java es resolver estos tres inconvenientes, permitiendo la distribución pública de la aplicación, y experimentación sencilla con nuevas funcionalidades y tecnologías de acceso.

La versión actual en Java del Wayback Machine soporta tres tipos de acceso: Modo "Archival Url", modo "Proxy", y modo "Domain Prefix".

El modo Archival URL provee al usuario una experiencia cercana a la versión actual de Wayback Machine que está en producción. Todas las consultas y peticiones de acceso de repetición pueden ser expresadas con URLs. En el modo de repetición Archival URL, el contenido archivado es modificado y retornado a los usuarios, intentando crear enlaces y contenido embebido referenciando nuevamente a Wayback Machine.

El modo Proxy URL permite retornar los documentos archivados dentro del navegador del cliente configurando el mismo para que redireccione todas las peticiones http a través del Wayback Machine. Esto tiene como gran ventaja que no se requiere Javascript para forzar al navegador del cliente a realizar peticiones de URLs adicionales y contenido embebido del Wayback Machine.

El modo DomainPrefix es similar al modo Archival URL, pero utiliza un esquema DNS para reescribir los URL, permitiendo que la sustitución de URL ocurra en el servidor. Este modo está considerado experimental.

La versión actual de Java puede operar en distintos modos de despliegue, desde aplicaciones individuales en un solo host que contiene todos los documentos e índices archivados, hasta sistemas altamente distribuidos donde los índices y el contenido archivado es desplegado a través de cientos de máquinas.

En el modo individual local, este software incluye la capacidad de escanear nuevo contenido archivado en una locación específica, y automáticamente indexar y servir el contenido como apareció originalmente. Hacer que el Wayback busque archivos WARC en el directorio donde la instancia del rastreador Web está escribiendo las salidas, debería proveer la capacidad para buscar contenido archivado.

Directrices para la preservación Web de la UNESCO (2003)

A partir del año 2003 se han creado aproximadamente 31 iniciativas, con un promedio de 6 por año. Esto se pudo deber a la preocupación adoptada por la UNESCO en relación a la preservación de herencia digital durante la trigésima segunda sesión de la Conferencia General de la UNESCO el 17 de Octubre de 2003 (UNESCO, 2003). En esta sesión se publicaron una serie de artículos concernientes al tema de preservación digital.

El artículo 2 destaca el propósito del mantenimiento de la herencia digital para que sea accesible al público. Se acordó que aquel material que sea del dominio público debe poder ser accedido sin ningún tipo de restricciones, mientras que aquel que sea privado debe estar protegido de cualquier intrusión. Para esto las instituciones u organizaciones en los distintos países deben llegar a un acuerdo de equilibrio entre los derechos de autores, otros derechos y el interés del público para poder acceder a material digital de herencia.

Los tres siguientes artículos de la Conferencia de la UNESCO destacan la preocupación por la pérdida de la herencia digital. El artículo 3 “The threat of loss” (la amenaza de la pérdida) menciona algunas de las razones por las que esto sucede:

- Lo rápido de la obsolescencia del hardware o el software que mantienen la información.
- La incertidumbre acerca de los recursos que se deben utilizar.
- La responsabilidad y los métodos de preservación y mantenimiento.

- La falta de soporte legislativo.
- La tecnología digital avanza muy rápidamente y los costos aumentan por lo que los gobiernos e instituciones no pueden desarrollar estrategias de preservaciones bien documentadas y permanentes en el tiempo.

El término amenaza por lo tanto se refiere a los beneficios de la herencia digital que tienden a perderse, como datos estadísticos históricos que permitan analizar mejor temas económicos, información cultural, social e intelectual.

En la sesión de la UNESCO también se habla acerca de las medidas que deben tomarse para poder solucionar el problema, como por ejemplo el artículo 6 que menciona la importancia del desarrollo de políticas y estrategias tomando en cuenta las circunstancias locales, nivel de urgencia, recursos disponibles y proyecciones futuras. El artículo 7 es muy relevante ya que menciona la importancia de la selección de aquello que debe preservarse. La selección varía de acuerdo al país, aunque lo más significativo a tomar en cuenta debe ser aquella información más relevante y más duradera en términos culturales, científicos, evidenciales o de valor. Las últimas secciones mencionan los roles y las responsabilidades que deberían participar en estos procesos, desde desarrolladores de hardware y software, publicistas, distribuidores, productores, museos, organizaciones y bibliotecas nacionales hasta el rol que debe tener la UNESCO.

Consorcio Internacional de Preservación de Internet (IIPC)

En el año 2003 fue fundado el Consorcio Internacional de Preservación de Internet (IIPC, por sus siglas en inglés) y está compuesto por distintas instituciones que trabajan en la preservación del Internet para las futuras generaciones. La misión principal de este consorcio es adquirir, preservar y hacer accesible el conocimiento e información de internet promoviendo intercambios globales y relaciones internacionales (Grotke, A; IIPC, 2008). Entre sus objetivos están: permitir una colección de contenidos de internet de cualquier parte del mundo que pueda ser preservado y archivado de forma segura y que pueda ser accedido a través del tiempo, promover el uso de herramientas comunes, técnicas y estándares que permitan la creación de archivos internacionales y por último fomentar y ayudar a las bibliotecas nacionales a que utilicen métodos de preservación digital en línea (IIPC, 2012). Entre algunos de sus miembros se encuentran La Biblioteca de Cataluña, La Biblioteca Nacional de Florencia, La Biblioteca Nacional de Francia, La Biblioteca Nacional de Alemania, La Biblioteca del Congreso; fundaciones sin fines de lucro como “Internet Archive” e “Internet Memory Foundation”; Universidades como Columbia y Harvard, entre otras; proveedores de servicio como Hanzo de Gran Bretaña y Archivos Nacionales como “The National Archives” de Gran Bretaña. (IIPC, 2012)

Todos los archivos Web seleccionan sitios específicos para el archivado. Estos sitios se escogen de acuerdo a factores como el consentimiento del autor o la relevancia de colecciones temáticas.

Algunos archivos Web se concentran en contenidos específicos que típicamente son grandes, como videos o archivos PDF o que simplemente están en constante crecimiento. Por lo tanto se requieren distintos formatos para almacenar que contienen meta-data o que utilizan métodos de compresión. De aquí surge el formato ARC que fue tomado por The Internet Archive como el estándar de almacenamiento (Burner & Kahle, 1996). En 2009 el formato WARC fue publicado por la Organización Internacional de Estandarización (ISO, 2009). ARC/WARC fue motivado por la creación del proyecto Archive-Access (Acceso-Archivo) que permite la descarga gratuita de herramientas de software libre que procesan este tipo de formatos (Ver Formato de archivo de este capítulo). (Internet Archive, 2008)

2.4. Arquitectura para archivar la Web

Para llevar a cabo la preservación Web hay que contar con un sistema que tenga las herramientas adecuadas para lograr dicha preservación, a grandes rasgos son necesarios tres módulos: uno que permita obtener los documentos Web, otro que permita almacenarlos en algún repositorio y uno que le permita a los usuarios acceder al contenido. Así mismo, para poder realizar el archivado Web con éxito entonces es necesario definir las funciones que se deben implementar: adquisición, almacenamiento e indexación y acceso

La obtención de los documentos Web se conoce como adquisición. Existen varios métodos de adquisición (Ospina Torres y León Luna, 2013). Uno de los más usados (método de adquisición del lado del cliente) requiere de un rastreador Web, también conocido como araña Web, el cual es un programa que se encarga de inspeccionar la World Wide Web de forma metódica y automatizada, con la finalidad de crear una copia de todas las páginas Web visitadas para posterior procesamiento.

Para describir el funcionamiento de los rastreadores es necesario definir como son identificados los recursos en Internet, por lo que se introducirá el concepto de URI (Uniform Resource Identifier), que es una cadena de caracteres que identifica inequívocamente un recurso en Internet, por ejemplo, http://www.ejemplo.com/usuarios/1.

Los rastreadores para comenzar la cosecha reciben una semilla, que es la URI a partir de la cual se va a comenzar el rastreo. Los rastreadores analizan cada página en busca de nuevas URIs las cuales serán encoladas para su rastreo, así el proceso continúa recursivamente. Para que este proceso no sea “infinito” es necesario establecer un alcance donde se indica hasta que profundidad de URIs descubiertas se va a hacer el rastreo (ver sección Adquisición de este capítulo).

Luego de que el rastreador coseche la página, dicha información se almacena en uno o varios archivos (ver sección Formatos de archivo de este capítulo) diseñados para soportar múltiples recursos

Web. Aquí es cuando entraría en escena el módulo de almacenamiento e indexación, este modulo aparte de almacenar las páginas Web cosechadas debe poder indexarlas para facilitar futuros accesos.

Por último, para poder tener acceso a los archivos Web almacenados en sus diversas versiones, es necesario contar con una aplicación, a través de la cual los usuarios pueden acceder a dichos archivos y desplegarlos.

A continuación se describirán un modelo de referencia para la creación de sistemas de preservación de contenido digital y su adaptación a Archivos Web el cual ha sido usado como base para el desarrollo del prototipo.

Modelo OAIS

El modelo Sistema de Información para Archivos Abiertos (*Open Archival Information System* OAIS), es un intento a proporcionar un marco de alto nivel para el desarrollo y la comparación de archivos digitales o repositorios, desarrollado por el Comité Consultivo de Sistema de Datos del Espacio *(Consultative Committe for Space Data System, 2002)* con la aprobación de la Organización Internacional de Normalización (ISO, 2003). Este modelo tiene como objetivo proporcionar un marco común que se puede utilizar para ayudar a entender los desafíos que representa los archivos Web. El valor del modelo es el proporcionar un lenguaje común de alto nivel que pueda facilitar la discusión a través de las comunidades interesadas en la preservación digital.

El modelo OAIS define dos roles esenciales como lo son el de "productor" y el de "consumidor". El productor es la persona o sistema cliente que se encarga de suministrar los archivos a ser conservados; mientras que el consumidor es la persona o sistema cliente que puede acceder a los archivos preservados. Este modelo también define la interacción entre los módulos y funciones de adquisición o ingreso, almacenamiento, gestión de datos y acceso. También define un módulo de Administración el cual maneja las operaciones cotidianas del Archivo y coordina las actividades de los demás componentes funcionales del OAIS y otro módulo llamado Planificación de preservación el cual se encarga de la vigilancia del medio ambiente de la OAIS para garantizar la conservación a largo plazo de los contenidos del Archivo. (Masanés, 2006). (Ver Figura 2)

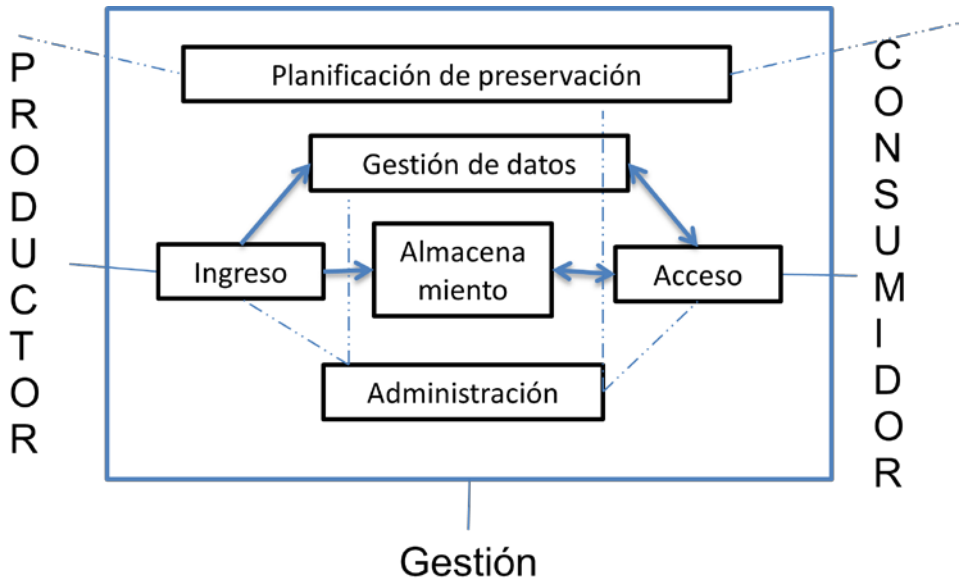


Figura 2.Modelo OAIS basado en el modelo funcional descrito

Arquitectura Funcional propuesta por el IIPC, basada en el modelo OAIS (Masanés, 2006)

La arquitectura funcional descrita por el IIPC (Ver Figura 3) es una adaptación del modelo OAIS y se puede relacionar con sus componentes de la siguiente manera:

- **Herramientas de Ingestión de Datos (*Data Ingest Tools*)** son las herramientas de adquisición (rastreador o araña Web) y además corresponden con el rol del Productor y con el componente funcional Ingreso, ya que se proveen las herramientas para aceptar contenidos y prepararlos para su almacenamiento.
- **Motor de Indexación y Búsqueda (*Index & Search Engine*)** es el que permite filtrar las peticiones del usuario Consumidor de acuerdo a criterios y además se comunica con la base de datos de índices para poder realizar sus tareas, por lo tanto se corresponde con el módulo de Gestión de datos del modelo OAIS ya que este maneja repositorios de los metadatos que describe la información almacenada en el Archivo como su mismo nombre lo indica.
- **Herramientas de Almacenamiento (*Storage Tools*)** son las que permiten almacenar los archivos en una base de datos dispuesta para esto, por lo tanto, en conjunto las herramientas y el repositorio forman el módulo de almacenamiento.
- **Herramientas de Acceso (*Access Tools*)** son las que le permiten al usuario ingresar y ver el contenido de los archivos almacenados de acuerdo a la búsqueda que haya realizado. Estas herramientas pueden ser aplicaciones Web y corresponden al módulo y función de Acceso.

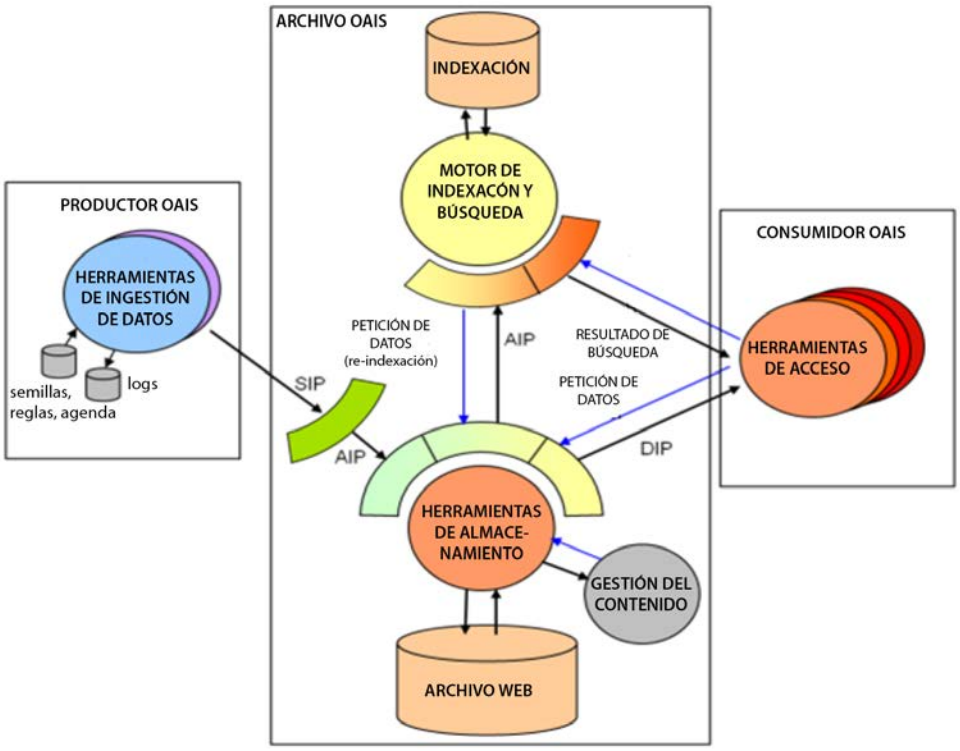


Figura 3. Arquitectura Funcional IIPC

2.5. Módulos de Acceso

El módulo de acceso, dentro de la arquitectura funcional descrita por el IIPC (Ver Figura 3), está constituido por una aplicación Web que en rasgos generales se encarga de realizar búsquedas de un sitio Web en específico almacenado en el archivo Web, mostrar las distintas versiones del mismo existentes y, finalmente, de acuerdo a la versión elegida por el usuario realizar la lectura y el despliegue del archivo WARC que lo contiene, recreando así lo más parecido posible el sitio original.

A continuación se describen tecnologías, arquitecturas y protocolos concernientes a las Herramientas de Acceso del Módulo de Acceso.

2.5.1. Aplicaciones Web

Aunque los inicios de Internet se remontan a los años 60, no es sino a partir de los 90 que este comienza su auge y extiende su uso a través del mundo. Este paso se da gracias a la aparición de la Web, la cual ha evolucionado enormemente a través de los años, pasando de páginas simples de contenido estático, como texto e imágenes, a páginas complejas con contenido dinámico proporcionado por bases de datos, formándose de esta manera el concepto de aplicación Web (Lujan Mora, 2002).

El concepto de Web está formado por tres elementos claves introducidos por Tim Bernes-Lee. Estos conceptos son:

- HTML: como lenguaje para crear contenidos Web.
- HTTP: como protocolo de comunicación entre computadores. También maneja la transferencia de datos y recursos.
- URL: como medio de direccionamiento de los distintos recursos de internet.

En el año 1990 Tim-Bernes-Lee escribió un programa que permitió crear enlaces entre nodos, lo que le da paso al desarrollo de una aplicación que comprendía un editor y un navegador gráfico de hipertexto al que llamó en un principio NextStep y que luego pasó a ser lo que conocemos hoy como World Wide Web.

2.5.2. Arquitectura Cliente / Servidor

El término cliente-servidor fue utilizado por primera vez en 1980 para referirse a los computadores en red. Este tipo de arquitectura implica una relación entre procesos o aplicaciones Web que solicitan servicios (clientes) y procesos que responden a dichas peticiones (servidores) (Ver Figura 6). Las ventajas de esto es que existirá una separación marcada entre roles y funcionalidades según el servicio que se ofrezca, además los niveles de seguridad aumentarán, la aplicación será modular siempre buscando que los servicios sean independientes entre sí, situando a cada uno en su plataforma más adecuada. Este término no implica la separación de computadores, es decir, el cliente y el servidor pueden residir en un mismo ordenador, pero esta arquitectura tiene como aspecto relevante el que permite desarrollar aplicaciones distribuidas con mayor facilidad que otras. En otras palabras, la arquitectura cliente-servidor es parte de la programación modular en la que un software es separado en módulos para facilitar su desarrollo y mantenimiento.

En una arquitectura de este tipo surge entonces las redes cliente-servidor, en la cual todos los clientes están conectados a un servidor en el que los diversos recursos y aplicaciones se encuentran centralizados y que son puestos a disposición de los usuarios cada vez que sea necesario (Ramírez, Hodgson, Reyes, & Coleman, 2010).

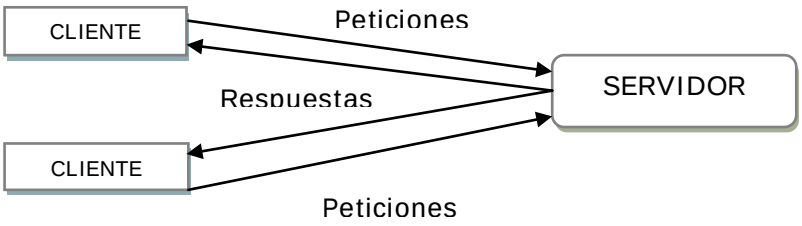


Figura 4. Arquitectura Cliente-Servidor

Características del cliente

El cliente dentro de una arquitectura cliente-servidor es quien inicia los requerimientos del servicio y posee las siguientes características:

- Es quien inicia las peticiones al servidor, por lo tanto su papel es activo y de suma importancia para determinar el comportamiento de la aplicación.
- Espera y recibe respuestas del servidor.
- Puede conectarse a uno o más servidores.
- Le presenta las respuestas del servidor al usuario usualmente a través de una interfaz gráfica.
- Por lo general se realizan validaciones del lado del cliente, como por ejemplo manejo de formatos en formularios.

Características del servidor

El servidor es aquel que se encarga de responder a las peticiones del cliente y posee las siguientes características:

- Desempeñan el papel pasivo en la comunicación ya que esperan por las peticiones del cliente, las cuales reciben, procesan para luego generar una respuesta.
- Aceptan conexiones desde múltiples clientes, por lo que permiten el multiprocesamiento y poseen mecanismos para manejar la concurrencia.
- No interactúan con los usuarios finales ya que de eso se encarga la aplicación cliente.
- Contienen el modelo de datos y la lógica para acceder a éste y modificarla o generarla según la petición del usuario.
- Poseen mecanismos de seguridad adicionales para la aplicación, como por ejemplo codificación MD5 o validaciones del modelo de datos.
- Por lo general los requerimientos de hardware son mayores para el servidor ya que deben ser capaces de responder en un tiempo aceptable a múltiples usuarios, además están encargados de guardar toda la información de los servicios y aplicaciones que soporta. Esto requiere mayor almacenamiento temporal y secundario, multiprocesamiento, mejores tiempos de respuesta y eficiencia.

Ventajas

Entre las ventajas que la arquitectura cliente-servidor ofrece, encontramos las siguientes:

- Las redes de computadores basados en arquitectura cliente-servidor permiten ejecutar partes distintas de una misma aplicación de forma concurrente.
- Permite la migración de aplicaciones de un procesador a otro con mínimos cambios en las aplicaciones, siempre y cuando se mantengan las tecnologías entre procesadores.

- Permite la escalabilidad tanto horizontal como vertical de la aplicación, es decir, permite aumentar el número de módulos o incluso disminuirlos sin afectar a otros servicios (horizontal) y además permite migrar a servidores más rápidos y de mayor rendimiento a medida que la aplicación crece (vertical).
- Permite el acceso de datos sin importar dónde se encuentre el usuario.
- Se puede duplicar la seguridad de una aplicación realizando validaciones tanto en el lado del cliente como en el lado del servidor.
- Permite centralizar el manejo de la información y separar las responsabilidades clarificando el diseño del sistema.

2.5.3. Estado de Transferencia Representacional (ReST)

ReST significa Representational State Transfer o Estado de Transferencia Representacional y es una arquitectura de software hecha para diseñar y para construir aplicaciones Web, y que se basa en el concepto de la representación de recursos. Un recurso puede ser cualquier concepto coherente al que se haga referencia como una URL. La representación de un recurso es un documento que capta el estado actual del mismo. Un ejemplo de esto es una página HTML, o HAML. ReST también utiliza todas las operaciones básicas o métodos del protocolo HTTP: GET, POST, DELETE, UPDATE. Esto evita tener que utilizar protocolos complejos como RPC, SOAP o COBRA.

ReST está basado además en protocolos de comunicación que pueden utilizar cache, que poseen una arquitectura cliente-servidor y que no tengan estado. Este tipo de arquitectura utiliza peticiones HTTP para crear, actualizar, leer y para eliminar datos. Por lo tanto las aplicaciones ReST utilizan el protocolo HTTP para las cuatro operaciones CRUD (Create/Read/Update/Delete). Las aplicaciones ReST son auto-contenidas, por lo que cada petición lleva consigo toda la información que requiera el servidor para completar sus operaciones.

En cuanto a programación, las aplicaciones ReST son alternativas ligeras a protocolos pesados y complejos como RPC o a servicios Web como SOAP o WSDL. Esto lo logra sin perder las propiedades de los servicios Web, como por ejemplo:

- Son independientes a la plataforma.
- Independencia del lenguaje de programación utilizado
- Está basado en estándares, en este caso HTTP o incluso HTTPS para encriptación.
- Pueden utilizarse en la presencia de Firewalls
- Para cuestiones de seguridad como nombre de usuario y claves utiliza tokens
- Los servicios ReST utilizan XML en sus respuestas como una manera de organizar datos, mientras que en las peticiones no se utiliza casi, esto debido a la simplicidad de dichas peticiones. En las peticiones simplemente se utiliza el método HTTP GET o el método HTTP

POST y sus parámetros en el URL sin comprometer la seguridad. Pero las aplicaciones ReST no tienen que utilizar obligatoriamente XML, al contrario de SOAP. También puede utilizar JSON o CSV.

Ventajas de utilizar ReST

La mayor ventaja que posee ReST es la facilidad de implementación, el orden y la rapidez del diseño y lo ligero de la aplicación frente a otros protocolos, además de su buen rendimiento. Este último se debe al soporte que posee de cache, peticiones simples y respuestas seguras y sencillas de parsear. ReST también permite reducción del tráfico de red ya que las operaciones se realizan con mayor rapidez y menos interacciones.

Su simplicidad, rendimiento y seguridad han hecho que el uso de ReST vaya aumentando con el paso de los años y sea utilizado por compañías de vanguardia como Twitter, Yahoo, Amazon, la gran mayoría de blogs que se encuentran en la Web, eBay, entre otros.

2.5.4. Patrón arquitectónico Modelo-Vista-Controlador

La arquitectura del software es el conjunto de guías y formas generales con base en las cuales se puede resolver un problema determinado. Estas indican la estructura, funcionamiento e interacción entre las partes del software. Por otro lado el diseño de una aplicación es un modelo del sistema, con un conjunto de técnicas y principios que permiten describirlo con el suficiente detalle para poder implementarlo. Estos principios y técnicas combinados con esquemas y estructuras de solución que pueden ser utilizados múltiples veces de acuerdo al contexto del problema forman un patrón de diseño.

Cada patrón puede ser utilizado de maneras distintas para resolver un mismo problema de acuerdo a su contexto. Además permiten que algunos aspectos del sistema puedan cambiar sin afectar otros aspectos.

Modelo-Vista-Controlador es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos y que se utiliza principalmente para manejar grandes cantidades de datos y de transacciones complejas donde se requiere una mejor separación de conceptos, de esta manera el desarrollo de la aplicación esté estructurado de manera ordenada, permitiendo la separación de roles y facilitando la programación en diferentes capas de manera paralela e independiente, evitando así los duplicados de código, la mezcla de la vista con el controlador y a su vez con el modelo de datos.

Este patrón fue descrito por primera vez por Trygve Reenskaug en 1979, y la implementación original fue realizada en Smalltalk en los laboratorios Xerox y se basa en la separación de la aplicación en tres capas: modelo, vista y controlador (Ver Figura 7).

- **Modelo:** es la representación de la información que maneja la aplicación, es decir, los datos, que en el contexto de la aplicación, proveen de información al usuario y a la aplicación misma. Es otra forma de llamar a la capa de dominio.
- **Vista:** es la representación del modelo en un formato que le permita al usuario interactuar con el mismo, usualmente a través de una interfaz gráfica. En el caso de una aplicación Web la Vista es una página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones.
- **Controlador:** es la capa encargada de manejar y responder las solicitudes del usuario procesando la información necesaria e incluso modificando el modelo si es necesario. Obtiene los datos del modelo y se los presenta al usuario como respuesta a sus peticiones a través de la vista.

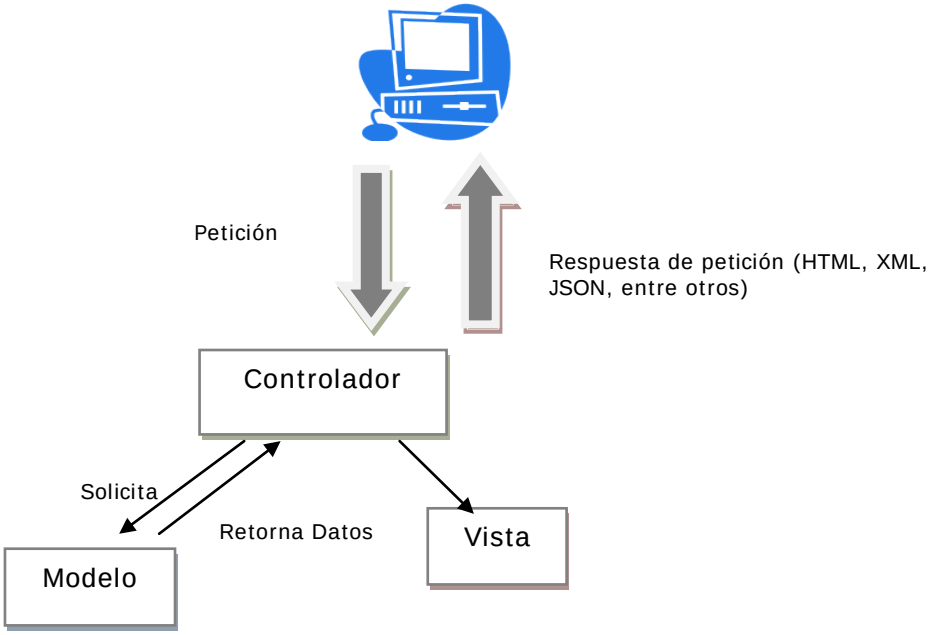


Figura 5. MVC

Ventajas de utilizar MVC

El patrón MVC ofrece muchas ventajas y facilidades para las distintas fases del proceso de ingeniería del software ya que permite la separación de roles, de módulos, la reutilización y un trabajo ordenado. Por lo tanto las ventajas del patrón MVC se relacionan con su misma arquitectura:

- Una separación marcada entre lógica de negocio y la capa de presentación. Esto permite que el acceso a los datos, agregar funcionalidades como validaciones o multilenguaje, la reutilización de módulos, entre otros, no requieran que la vista cambie, permitiendo que la aplicación sea más fácil de mantener y que la misma permanezca consistente.
- Se pueden agregar nuevas vistas que pueden ser manejadas por un mismo controlador.
- Se pueden crear controladores con vistas asociadas a él, de esta manera la aplicación es más fácil de organizar.
- Permite mejorar el rendimiento de la aplicación con solo modificar el código del controlador e incluso migrar la aplicación a nuevas tecnologías sin necesidad de unir las distintas capas, ya que se puede actualizar el framework MVC sin tener que cambiar la vista o se pueden utilizar nuevas versiones de HTML o CSS sin afectar el código del controlador.
- Las correcciones son más fáciles y rápidas de realizar, ya que sólo se deben hacer en un solo módulo de la aplicación y no en varios cómo sucedería si se mezclasen la vista y el código de control.
- Permiten la separación clara de roles en el equipo de desarrollo de la aplicación, ya que mientras unos se ocupan del modelo de datos, otros se ocupan de la lógica de negocio y otros de la presentación de la misma.

2.5.5. Frameworks basados en MVC

Un framework es una plantilla o diseño que puede ser reutilizable y que está compuesto por un conjunto de clases abstractas y las instancias de las mismas. Un framework es orientado a objetos y sus componentes facilitan la creación de aplicaciones Web ya que permiten abstraerse del protocolo HTTP y concentrarse en el desarrollo y en el mantenimiento.

Los frameworks fueron evolucionando con el paso de los años y el crecimiento de Internet. En un comienzo la Web fue estática, las respuestas a las peticiones eran archivos con el contenido solicitado. Luego con el desarrollo de Servlets se crea una abstracción de los mensajes que el servidor recibe, está abstracción se logra a partir de la creación de objetos abstractos de los que otras clases heredan para redefinir el comportamiento de las respuestas. Con la aparición de los Java Server Pages, aparecen dos modelos iniciales de frameworks; el primero definía las clases del modelo de datos a las que accedía la aplicación, se les aplica la lógica de negocio y se selecciona la vista en la que se mostrará el resultado; el segundo modelo define un controlador central que recibe las peticiones, selecciona los datos a utilizar, los procesa y elige la siguiente vista a utilizar.

Esta evolución entonces llevó a la creación del patrón MVC y por lo tanto a la creación de frameworks basados en el mismo buscando entonces separar los tres componentes: modelo, vista y controlador.

Ruby on Rails

Ruby on Rails, también conocido como RoR o Rails es un framework de aplicaciones Web de código abierto, escrito en el lenguaje de programación Ruby siguiendo el paradigma de la arquitectura Modelo Vista Controlador (MVC). Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros frameworks y con un mínimo de configuración. Además de esto se presenta con una sintaxis muy legible. Rails se distribuye a través de RubyGems, que es el gestor de paquetes para Ruby, el cual proporciona un formato estándar y autocontenido (llamado gem) para poder distribuir programas o librerías en Ruby, una herramienta destinada a gestionar la instalación de éstos, y un servidor para su distribución.

La estructura de una aplicación Ruby on Rails es la siguiente:

- Utiliza la arquitectura **Modelo Vista Controlador**

  Modelo: En las aplicaciones Web orientadas a objetos sobre bases de datos, el Modelo consiste en las clases que representan a las tablas de la base de datos. En Ruby on Rails, las clases del Modelo son gestionadas por ActiveRecord, que es la principal clase para la administración y funcionamiento de modelos. ActiveRecord es una implementación de este patrón de programación y está muy influenciada por la funcionalidad de su análoga en Ruby disponible en Rails. ActiveRecord proporciona la capa objeto-relacional que sigue rigurosamente el estándar ORM: Tablas en Clases, Campos en Atributos y Registros en Objetos. Facilita el entendimiento del código asociado a base de datos y encapsula la lógica especifica haciéndola más fácil de usar para el programador. El modelo representa las tablas de la base de datos, migraciones (expresan cambios en las BD), observadores.

  Vista: En MVC, *Vista* es la lógica de visualización, o cómo se muestran los datos de las clases del *Controlador*. Con frecuencia en las aplicaciones Web la vista consiste en una cantidad mínima de código incluido en HTML. El método de plantillas que se emplea en Rails por defecto es usar Ruby Embebido (archivos .rhtml, y a partir de la versión 2.x en adelante de RoR archivos.html.erb), que son básicamente fragmentos de código HTML con algo de código en Ruby (como HAML, Liquid, etc). En las vistas podemos usar los helpers, los cuales definen funciones que producen contenido para el HTML o Javascript de la aplicación; a través de estos se pueden crear formularios, campos y otros elementos.

  Es necesario escribir un pequeño fragmento de código en HTML para cada método del controlador que necesita mostrar información al usuario. El "maquetado" o distribución de los elementos de la página se describe separadamente de la acción del controlador y los fragmentos pueden invocarse unos a otros. Para ello se utilizan layouts.

Controlador: En MVC, las clases del *Controlador* responden a la interacción del usuario e invocan a la lógica de la aplicación, que a su vez manipula los datos de las clases del *Modelo* y muestra los resultados usando las *Vistas*. En las aplicaciones Web basadas en MVC, los métodos del *controlador* son invocados por el usuario usando el navegador Web. La implementación del *Controlador* es manejada por el ActionPack de Rails, que contiene la clase ApplicationController. Una aplicación Rails simplemente hereda de esta clase y define las acciones necesarias como métodos, que pueden ser invocados desde la Web, por lo general en la forma http://aplicacion/ejemplo/metodo, que invoca a la función *método* del controlador *ejemplo*, y presenta los datos usando el archivo de plantilla /app/views/ejemplo/método.html.erb, a no ser que el método redirija a algún otro lugar.

- Utiliza Gemas, las cuales son plugins y/o códigos añadidos a nuestros proyectos Ruby on Rails, que nos permiten nuevas funcionalidades como nuevos create, nuevas funciones pre-escritas (como login de usuarios) o nuevas herramientas para el desarrollo como puedan ser Haml y SASS (la primera es una nueva forma de plantilla basada en html pero más sencilla y potente, y la segunda es igual pero para el caso de las CSS).

- En cuanto a los servidores Web, para desarrollo y pruebas, se utiliza WEBrick (incluido con Ruby) o Unicorn (instalado a través de gemas). Para el modo producción se puede utilizar Nginx, Mongrel, Apache, Lighttpd con FastCGI o alguna combinación, por ejemplo utilizar Apache como proxy para los procesos WEBrick.

- En cuanto a las Bases de Datos, dado que la arquitectura Rails favorece el uso de estas se recomienda usar un Sistema Manejador de Bases de Datos Relacionales (SMBDR) para almacenamiento de datos. Rails soporta la biblioteca SQLite por defecto. El acceso a la base de datos es totalmente abstracto desde el punto de vista del programador ya que Rails gestiona los accesos a la misma automáticamente (aunque, si se necesita, se pueden hacer consultas directas en SQL). Rails intenta mantener la neutralidad con respecto a la base de datos, la portabilidad de la aplicación a diferentes sistemas de base de datos y la reutilización de bases de datos preexistentes. Sin embargo, debido a la diferente naturaleza y prestaciones de los SMBDR el framework no puede garantizar la compatibilidad completa. Se soportan diferentes SMBDR incluyendo MySQL, PostgreSQL, SQLite, IBM DB2 y Oracle.

- Cuando se realiza una petición en una aplicación Ruby on Rails, se cumplen los siguientes pasos (Ver Figura 8):

  (1) El navegador hace una petición, tal como http://mysite.com/video/show/15, la cual es recibida por un servidor Web (como Apache, WEBrick, mongrel, etc). Utiliza el archivo **routes** para averiguar qué controlador usar (por defecto la ruta es "/controller/action/id", como está definido en config/routes.rb). En este ejemplo, el controlador es "video", el método "show" y el id "15". El servidor Web a continuación

utiliza el Dispatcher o Despachador para (2) crear un nuevo controlador, llamar a la acción y pasarle los parámetros.

(3) Los controladores se encargan de analizar las solicitudes de usuario, presentaciones de datos, cookies, sesiones, entre otros. En nuestro caso, el método *show* en el controlador *video* sabe que debe buscar un video. Se le solicita al modelo obtener el video con identificador 15, y con el tiempo lo mostrará al usuario.

(4a) Puede que la acción a realizar requiera datos del modelo, por lo que los procesaría (insertar, consultar, actualizar, borrar) por medio de la librería ActiveRecord del modelo para generar la respuesta. En este caso, el modelo recupera el video 15 de la base de datos.

(4b) Puede que la acción utilice otros módulos de Rails como Action Mailer para enviar un correo por ejemplo.

(5a) El controlador devuelve el cuerpo de la respuesta (HTML, XML, etc) y metadatos (almacenamiento en caché de cabeceras, redirecciones) al servidor. El servidor combina los datos en bruto en una adecuada respuesta HTTP y lo envía al usuario. En nuestro ejemplo, el controlador proporciona el video 15 para la vista "show".

(5b) Puede que el controlador en cuestión redireccione a otro controlador para procesar la solicitud del usuario.

(6) Las vistas muestran lo que el controlador le envía al servidor y este reenvía al usuario en una respuesta HTTP. Las vistas son lo que el usuario ve: HTML, CSS, XML, Javascript, JSON. La vista genera el código HTML: divs, tablas, textos, descripciones, pies de página, etc, para el usuario.

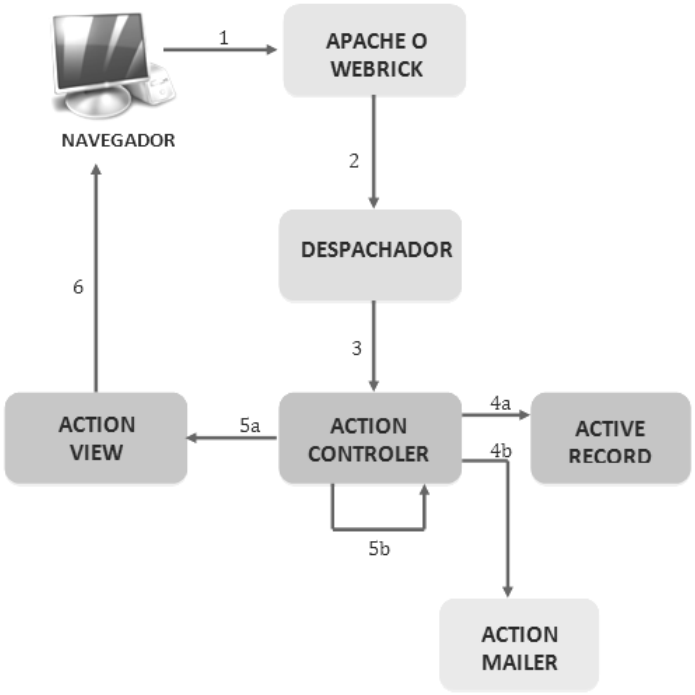


Figura 6. Estructura básica de una petición en Ruby on Rails

A continuación se presenta una tabla comparativa entre tres frameworks Web usados actualmente, CakePHP, Python Django y Ruby On Rails:

Tabla 2. Comparativa entre los frameworks Web CakePHP, Python Django y Ruby on Rails

Framework/ Característica	**CakePHP**	**Python Django**	**Ruby on Rails**
Manejo de contenido	Realiza las operaciones CRUD a través de una herramienta llamada Scaffolding que luego puede ser personalizada manipulando el código	Presenta una interfaz de administración que permite realizar las operaciones CRUD de forma sencilla	Realiza las operaciones CRUD a través de una herramienta llamada Scaffolding que luego puede ser personalizada manipulando el código
Sesiones de usuario	Soportada	Soportada	Soportada
Manejo de datos	Ofrece mecanismos poderosos de migración que permiten crear y actualizar los datos sin tener que destruir los anteriores ya que verifica los campos created, modified, updated.	Las migraciones de los datos son manuales. Actualizaciones de los datos pueden llevar a destruir los anteriores ya que no posee herramientas que automaticen este proceso.	Ofrece mecanismos poderosos de migración que permiten crear y actualizar los datos sin tener que destruir los anteriores ya que posee un manejador de versiones de migraciones
Mapeo Objeto Relacional	Si soporta. Es llamado *AppModel*	Si soporta. Es llamado *ORM Django*	Si soporta. Es llamado *ActiveRecord*

Usabilidad de las plantillas de interfaz	Sintaxis medianamente compleja ya que debe incluir código PHP embebido.	Sintaxis simple e incluye un lenguaje diseñado para plantillas de interfaz (Django Template Language)	Sintaxis medianamente compleja ya que debe incluir código Ruby embebido.
Herramientas de prueba	Provee una suite de pruebas que permiten probar modelos cargando data simple.	Presenta un modo de pruebas que permite configurar el ambiente para tal fin. Se crean scripts que llaman a funciones del modelo y del controlador.	Presenta un modo de pruebas que permite configurar el ambiente para tal fin. Se crean scripts que llaman a funciones del modelo y del controlador.

La elección de cualquiera de los *frameworks* Web mencionados anteriormente depende del contexto. Para el caso del módulo de acceso y despliegue de archivos WARC para el prototipo de Archivo Web en Venezuela, se requiere la integración de todos los módulos de la arquitectura funcional propuesta por el IIPC. Cada uno de estos utiliza distintas herramientas que están escritas en diversos lenguajes que por lo tanto requieren de un protocolo de comunicación que abstraiga el proceso de unificación, el cual para efectos del proyecto es HTTP por su simplicidad. Las operaciones propias de HTTP pueden ser implementadas a través de ReST. Ruby on Rails además de utilizar esta arquitectura de software, posee gemas como Solrsan y Sunspot que facilitan la integración con Solr, herramienta a través de la cual se realiza la indexación en el prototipo de Archivo Web de Venezuela. Estas razones hacen que Ruby on Rails se adapte efectivamente al contexto de esta iniciativa de preservación Web.

2.5.6. Formatos de Archivos

Los archivos a ser preservados deben almacenarse en un formato que permita poder manejar la gran cantidad de datos e información que puede ofrecer una página Web. El formato a utilizar es el Formato WARC, el cual fue seleccionado para el Prototipo de Archivo Web de Venezuela por un estudio previo (García, J. y Rivero, L., 2013).

1. Formatos WARC/ARC

Son formatos que permiten almacenar, describir y guardar recursos primarios de la Web, junto con los sucesivos cambios que dichos materiales puedan experimentar a lo largo de su exposición, en un sólo archivo.

2. Formato ARC

Introducido en el año 1996 por Mike Burner y Brewster Kahle para “Internet Archive”. Fue creado porque guardar millones de archivos en sistemas regulares de archivos sería extremadamente difícil de manejar. Por lo tanto los archivos ARC se utilizaron para almacenar datos agregados en un

solo archivo de gran tamaño. Los requerimientos para el almacenamiento deben ser los siguientes: el archivo debe ser auto-contenido, es decir, los objetos deben poder ser identificados y extraídos sin necesidad de revisar un archivo indexado aparte, debe ser posible obtener archivos de distintos protocolos de red y además debe ser posible concatenar múltiples archivos en una sola cadena de datos (Jack & Binns, Web Archive - ARC, 2012).

3. Formato WARC

Este formato permite concatenar y almacenar múltiples recursos u objetos de datos que contienen cabeceras de textos y bloques de datos en un solo archivo de gran tamaño. Es una extensión del formato ARC que permitía capturar imágenes de la Web. Cada archivo contiene uno o más objetos digitales y una vez que el objeto es escrito el archivo se cierra y solo se abre para lecturas. Mientras que el formato ARC guarda la información primaria, el formato WARC puede almacenar información como metadatos, abreviar eventos duplicados, transformaciones de fechas y la segmentación de grandes recursos. Puede controlar la información de protocolos de la capa de aplicación como HTTP, DNS y FTP, comprimir datos manteniendo su integridad. (ISO, 2009).

Un archivo de formato WARC es una concatenación de una o más colecciones WARC. Por lo general la primera colección describe las que le siguen. El contenido de una colección es usualmente el resultado de la adquisición de páginas Web, imágenes, información de redireccionamiento URL, resultados de búsqueda de nombres a través del protocolo DNS, archivos únicos o material sintetizado (metadatos, contenido transformado). Una colección WARC consiste de una cabecera seguida de un bloque de contenido. (ISO, 2009)

Toda colección en WARC tiene un tipo. Actualmente hay ocho tipos propuestos existentes, que se explican a continuación (ISO, 2009):

- **warcinfo:** una colección de este tipo describe las colecciones que le siguen hasta el final del archivo o hasta la siguiente colección del tipo warcinfo. Típicamente aparece al inicio de cada archivo WARC y usualmente contiene información acerca de la cosecha que generó las siguientes colecciones.
- **response:** una colección del tipo response debería contener una esquematización completa y específica de la respuesta del servidor incluyendo la información del protocolo de red.
- **resource:** una colección calificada como un recurso puede contener, por ejemplo, un archivo adquirido directamente de un repositorio localmente accesible o el resultado de una búsqueda en la red donde el protocolo de información ha sido descartado.

- **request:** una colección del tipo request debería contener una esquematización completa y específica de la petición al servidor incluyendo la información del protocolo de red.
- **metadata:** es una colección creada para describir, explicar o acompañar a otra colección. Esta colección siempre debe referirse a otra sea de contenido original o versionado. E.g.: warc/0.9 7141 resource file://webserver/htdoc/images/logo.jpg
- **revisit:** este tipo describe la acción de volver a visitar el contenido previamente archivado y puede incluir un cuerpo de contenido abreviado que tiene que estar enlazado a alguna colección previa. Esto se utiliza en lugar de el tipo response o del tipo request para indicar que el contenido visitado era un documento completo o un duplicado de un material previamente almacenado.
- **conversion:** una colección de conversion contiene una versión alternativa del contenido de otra colección que se creó como resultado del proceso de archivado. Cada transformación de una versión alternativa debe ser independiente de su versión original.
- **continuation:** los bloques de colecciones de este tipo deben estar concatenados a su correspondiente colección anterior (de otros archivos WARC) para crear la colección original de tamaño completo. Este es utilizado cuando una colección hace que el WARC exceda el tamaño límite deseado y debe ser separado en segmentos.

La siguiente es una descripción de los campos contenidos en un archivo WARC:

- **Línea de Cabecera:** consiste en una secuencia de siete segmentos de texto, cada uno separado por uno o más espacios en blanco:

  - **Warc-id**: es el identificador de la versión de warc utilizada: e.g. warc-09
  - **Longitud de los datos:** longitud en octetos de la colección.
  - **Tipos de colección:** incluye los tipos de colección (e.g. conversión)
  - **URI:** e.g**:** file://kb.dk/images/buildings/black_diamond.jpeg
  - **Fecha de creación de la colección:** en formato "YYYYMMDDhhmmss"
  - **Tipo de contenido:** e.g.: "image/gif".
  - **Record-**id: Identificador de la colección.

- Parámetros de nombres: utilizados para información acerca de la colección. Hay siete parámetros distintos, algunos de estos opcionales.

2.5.7. WARC Tools

Los WARC Tools es una herramienta de código abierto que proporciona una librería, un conjunto de líneas de comando de terminal, plug-ins para los servidores web y una documentación general para la manipulación de los archivos Web contenidos en WARCs. El objetivo principal de esta herramienta es facilitar y promover la adopción del formato de archivo WARC (visto anteriormente) para el almacenamiento de los archivos Web (IIPC, 2012).

El proyecto es liderado por Hanzo Archives, en colaboración con el equipo de trabajo del Internet Archive y soportado por el International Internet Preservation Consortium (IIPC).

Las herramientas que proporcionan los WARC Tools son manejadas a través de scripts que a su vez son programables, es decir que pueden ser modificadas según sean las necesidades de la organización que las utilice.

Para el presente TEG se hizo uso del conjunto de líneas de comando a través de la terminal del sistema y el uso de shell script. A continuación se dará a conocer los comandos que esta herramienta proporciona:

- **arc2warc:** Dado un archivo ARC, este comando crea un archivo WARC (aunque no es de buena calidad y está aún en desarrollo). Se dice que no es de calidad ya que este tipo de WARC no tendría los encabezados HTTP que son característicos en este.
- **warc2warc:** Da como resultado un archivo WARC, dando como resultado otro archivo WARC, hace la compresión a warc.gz
- **warcdump:** Lanza como resultado el contenido del archivo WARC por salida estándar
- **warcextract:** Lanza por salida estándar el contenido de la metadata que existe en el archivo WARC.
- **warcindex:** Crea un índice referencial del archivo WARC, en el podemos ver campos como identificador (warc-id), desplazamiento (offset), tipo de contenido (content-type), ruta (URI), tamaño (content-length), entre otros.
- **warcunpack_ia:** Lanza como resultado un directorio que se crea a partir de una regeneración de la estructura del sitio original, no necesariamente es la misma, mas sin embargo este mismo realiza la transformación de las direcciones relativas.
- **warcvalid:** Verifica la correctitud de un archivo WARC.

A continuación una tabla comparativa entre los formatos de archivo ARC y WARC (The Digital Formats):

Tabla 3. Comparación entre los formatos de archivo ARC y WARC.

Formatos de Archivo / Características	ARC	WARC
Fase de producción	Se utiliza para contenido Web accesible en un estado archivado, lo que representa la forma final de diseminación a través de Internet a un agente de usuario (navegador web).	
Metadatos de la cosecha	Almacena la información básica sobre el momento de la cosecha, la dirección IP de la máquina cosechadora, tipo MIME y código de respuesta para la operación de cosecha.	
Dependencias Externas	Acceso de los usuarios depende en gran escala de la indexación de los archivos. La misma puede permitir el acceso del usuario por URL o la fecha de versiones del sitio	
Contiene	Imágenes como GIF, JPEG, documentos digitales	Páginas HTML, imágenes como GIF, JPEG, etc.
Divulgación	Utilizado desde 1996. Desarrollado por el "Internet Archive" (Brewster Kahle). Se dispone, de manera gratuita, de documentación y herramientas para utilizar estos archivos.	Estándar abierto, documentado públicamente, desarrollado bajo los auspicios de la *International Internet Preservation Consortium.* Presentado en mayo de 2005 como un elemento de trabajo a través de la norma ISO TC46/SC4, fue aprobado como un estándar internacional en mayo de 2009. ISO TC46/SC4/WG12, convocada por la Bibliothèque nationale de France, es el grupo de trabajo encargado de mantenimiento.
Eficiencia a Escala	Excelente para el mejoramiento de las cosechas a granel y la indexación eficiente para el acceso de URL y la fecha. El uso de la ARC coordinada y archivos DAT es una forma de apoyar a la indexación eficiente para dicho acceso.	Excelente para mayor eficiencia de cosecha y eficaz indexación para el acceso por URL y por fecha. Los encabezados de registro estructurado pueden ser extraídos y almacenado por separado para una indexación eficiente. WARC apoya la eliminación de duplicados y la compresión para reducir el tamaño de los archivos para su almacenamiento, transmisión, e indexación después de la cosecha.
Versión	Es la primera versión de este tipo de formato de archivos, su siguiente versión es el formato WARC. Se utiliza para almacenamiento de archivos digitales	Es la siguiente etapa de los archivos de tipo ARC, para ahora incluir archivos de tipo Web.

2.6. Metodología de Desarrollo de Software Programación Extrema (XP)

XP o Programación Extrema (eXtreme Programming) es quizás la metodología más utilizado y conocido dentro de los procesos ágiles de desarrollo de software. Fue originalmente ideada por Kent Beck al final de los años 90. En este método los cambios en los requisitos o requerimientos durante el ciclo de desarrollo son naturales. XP está basado en una serie de principios, de estos se mencionarán los básicos (Ver Figura 9):

- Pruebas continuas unitarias (unión de módulos) y de aceptación (aprobación del módulos).
- Planificación del proceso, los usuarios y las actividades que se llevarán a cabo durante el desarrollo.
- Mejoras causadas por las continuas entregas.
- Sistema de metáforas, es decir, espacio de nombres claros.
- Diseño simple; solo se diseñan las funciones necesarias.
- Simplificación del código.
- El código es compartido por todos, por lo que se corrigen y se evitan más errores.
- Integración continua; debida a renovaciones de versiones en cada ciclo del proceso.
- Programación estandarizada. Se siguen estándares que todos deben cumplir para que al compartir el código no existan confusiones.
- Ritmo sostenible para no sobrecargar al equipo de trabajo.
- Relación con el cliente, haciéndolo parte del equipo.
- Programación en parejas. El desarrollo de módulos mejora su rendimiento y disminuye sus costes y tiempos de entrega si se realiza en parejas.

Figura 7. Ágil XP (Rengarajan, 2013)

Basada en estos principios, la programación extrema posee las siguientes características:

- **Desarrollo iterativo e incremental:** cada ciclo lleva consigo mejoras, cambios e inclusión de nuevos requerimientos. Cada ciclo lleva consigo una serie de pruebas de aceptación y unión de módulos por lo que la aplicación tiende a crecer progresivamente con cada ciclo.

- **Pruebas unitarias continuas:** se utiliza un código previamente preparado para realizar pruebas durante cada iteración del proceso.

- **Programación en parejas:** la discusión entre los miembros del equipo permiten solucionar problemas de forma más rápida e incluso evitarlos, de esta manera no hay pérdida en la productividad inmediata.

- **Frecuente interacción con el cliente:** el cliente o un representante del mismo forma parte del equipo. El equipo de trabajo puede verificar si todos los requerimientos de uno o de varios módulos se cumplen con el cliente a través del uso de prototipos.

- **Corrección de todos los errores:** antes de añadir cualquier funcionalidad los errores deben ser tratados.

- **Refactorización del código:** reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento.

- **Simplicidad del código:** código fácil de mantener, corregir y mejorar.

Ciclo de vida de un proyecto XP (Wells, 2009)

- **Exploración:** en esta etapa los clientes plantean los requerimientos de usuarios más importantes para una primera entrega. El equipo de trabajo tiene oportunidad de familiarizarse con las tecnologías a utilizar, los patrones y prácticas. Se comienza a plantear la posible arquitectura a utilizar.

- **Planteamiento:** durante esta etapa, el equipo de desarrollo otorga y estima prioridades y cantidad de esfuerzo que se debe aplicar para cada requerimiento de usuario. Estas estimaciones se pueden realizar basándose en el tiempo y alcance de cada requisito para así determinar cuántos requisitos se pueden entregar antes de una determinada fecha y el número de iteraciones que pueden ser necesitadas.

- **Iteraciones:** el plan de entrega debe estar compuesto por iteraciones con duración aproximada de tres semanas (desarrollo ágil). Se recomienda definir la arquitectura de la aplicación durante la primera iteración y así utilizarla a través del resto del proyecto. Esto no siempre es así ya que quien decide los requisitos a resolver por cada iteración es el cliente y quizás estos no sean suficientes para plantear una arquitectura definitiva. Los puntos a tomar en cuenta para realizar una iteración son:

    - Requerimientos que aún no son tratados.
    - Velocidad del proyecto a partir del número de iteraciones.
    - Pruebas de aceptación no superadas en la iteración anterior.
    - Tareas no terminadas previamente.

  Las fases de las iteraciones son:

  1. **Planificación del proyecto:**
     a. Captación de requerimientos del usuario (Historias de usuario)
     b. Liberar el plan de entrega (informarle al equipo de trabajo acerca del plan y la fecha de entrega).
     c. Comienzan las iteraciones.
     d. Se determina la velocidad del proyecto.
     e. Programación en parejas.
     f. Se realizan reuniones diarias para definir nuevos requerimientos. y presentar avances.

  2. **Diseño:**
     a. Diseños simples.
     b. Se hace un glosario de términos para que el equipo de trabajo pueda evitar confusiones.
     c. Se debe hacer análisis de riesgos para predecir cualquier eventualidad.
     d. Se deben tomar en cuenta las funcionalidades extra por cada ciclo.

  3. **Codificación**
     a. Código simple, legible y modular

  4. **Pruebas:**
     a. Pruebas unitarias
     b. Pruebas de aceptación
     c. Pruebas de validación

- **Producción:** pruebas adicionales y revisiones de rendimiento. Se deben definir nuevas funcionalidades a incluir y evaluarlas. Se pueden reducir los tiempos de cada iteración y documentar las ideas propuestas para una futura implementación.

- **Mantenimiento:** esta etapa se realiza mientras la iteración anterior se encuentra en modo producción. El sistema debe mantenerse en funcionamiento mientras se agregan nuevas iteraciones por lo que quizás se requiera de nuevo personal y de redefinición de estructuras.

- **Muerte del proyecto:** cuando ya no existan más requerimientos de usuario, se realizan mejoras de rendimiento y de confiabilidad, se genera la documentación final y no se realizan más cambios en la arquitectura del sistema.

Roles dentro de un proyecto XP

- **Cliente:** el cliente presenta los requisitos de usuario (Historias de usuario) y las pruebas funcionales de validación, asigna las prioridades a cada requerimiento y decide cuáles serán implementadas en cada iteración de acuerdo a la relevancia de las mismas.

- **Encargado de pruebas:** es quien ayuda al cliente a escribir las pruebas funcionales del sistema, las ejecuta y difunde los resultados a los miembros del equipo de trabajo.

- **Encargado de seguimiento:** es quien proporciona información de feedback o realimentación a los miembros del equipo a partir de las estimaciones realizadas en los tiempos de entrega propuestos de cada iteración y los tiempos reales abarcados. Evalúa también si los requisitos propuestos para una iteración pueden ser factibles y alcanzados durante la misma de acuerdo a factores como el tiempo y recursos necesarios.

- **Coach o Entrenador:** es quien revisa el proceso global y le proporciona información a los miembros de equipo de trabajo acerca de las prácticas XP asegurándose que se cumplan.

- **Consultor:** miembro externo del equipo que posee conocimientos amplios en un tema o área específica relacionada con el proyecto a realizar.

- **Gestor:** es quien permite el vínculo cliente-programador. Se asegura que el equipo trabaje efectivamente proporcionando el entorno adecuado y necesario para esto. Su labor principal es coordinar el trabajo.

CAPÍTULO III

DESARROLLO DE LA APLICACIÓN

En este capítulo se describirá la adaptación de la metodología de desarrollo de software Programación Extrema (eXtreme Programmin XP), para el desarrollo del Módulo de Acceso a los contenidos preservados en formato WARC.

Para el desarrollo del Módulo de Acceso a los Contenidos Preservados en formato WARC para el Prototipo de Archivo Web de Venezuela se implementaron las diferentes fases de la metodología de desarrollo de software Programación Extrema, la cual sigue los principios de la Modelación ágil, con la finalidad de que este se realizara de manera rápida, eficiente y con la documentación necesaria. También se tomó esta decisión por estar familiarizados con los artefactos o entregables UML. Ante esto, la planificación del proyecto se hizo tomando en cuenta los requerimientos principales del prototipo y todos aquellos que fueron surgiendo durante el desarrollo de la aplicación.

A continuación se expone el desarrollo de la aplicación del presente TEG: Exploración, Planteamiento e Iteraciones.

3.1. Exploración

En esta fase del ciclo de vida del proyecto, se plantean los requerimientos de usuarios (Requerimientos Funcionales) más importantes y aquellos planteados por el equipo de trabajo que debe poseer el software (Requerimientos No Funcionales) de manera de garantizar una implementación de calidad y funcional. Además, esta fase incluye información acerca de las tecnologías utilizadas, la arquitectura y los usuarios de la aplicación desarrollada. Así mismo se presentan los prototipos de diagramación de interfaz los cuales se hicieron a partir de los bocetos que fueron presentados y aprobados durante la propuesta del TEG.

3.1.1. Requerimientos Funcionales:

- El sistema debe permitir el registro opcional para los usuarios. Para esto se solicita al usuario una serie de datos (*nombre*, *correo*, *clave*, entre otros).
- El sistema debe permitir un inicio de sesión opcional para los usuarios, validando *clave* y *correo.* El sistema mediante este inicio de sesión debe proveer al usuario un historial de búsquedas y de un listado de favoritos correspondientes al usuario.

- El sistema debe proveer un campo de búsqueda donde el usuario introduce el URL del sitio Web que desee consultar.
- El sistema debe permitir al usuario la selección, a través de un calendario, de la versión en el tiempo del sitio Web solicitado de acuerdo a las almacenadas en el Archivo Web.
- El sistema debe desplegar el sitio Web solicitado por el usuario de modo que pueda navegar a través de él.
- El sistema debe proveer estadísticas al usuario acerca de las versiones cosechadas del sitio Web solicitado.
- El sistema debe proveer al usuario información del sitio como iniciativa de preservación Web.

3.1.2. Requerimientos No Funcionales:

- **Usabilidad:** El sistema debe ser fácil de aprender, intuitivo, y lo más importante, que ofrezca al usuario la satisfacción necesaria para seguir utilizándolo. Por lo tanto las terminologías utilizadas para describir las funcionalidades del sistema deben ser comunes al usuario. También debe ser capaz de orientar al usuario en su uso para que, en caso de error, éste sepa cómo reaccionar ante la situación. Cualquier proceso interno del software debe ser transparente para el usuario, de manera que no genere confusión a quien lo utiliza. El usuario debe poder rehacer los cambios hechos de manera que se sienta seguro y confiado con lo que está realizando e incluso pueda probar y ahondar más en el sistema sin miedo a ocasionar pérdidas o fallas importantes.

- **Confiabilidad:** Se manejarán los posibles errores de la interacción entre el usuario y la aplicación a través de validaciones del lado del cliente y del lado del servidor. Se asegura la correctitud, consistencia e integridad de los datos a través del control de concurrencia ofrecido por el sistema manejador de bases de datos. También se asegura la correctitud del sitio Web mostrado al usuario.

- **Seguridad:** El software contará con validación de usuario y correo. Cada quien tiene una tarea específica y no pueden interferir entre ellas. El sistema debe informar de los errores de alguna operación o por el contrario del éxito de la misma.

- **Eficiencia:** El software será creado utilizando los recursos de la mejor manera, tanto de tiempo de respuestas y precisión de costos, así como optimizar el uso de la memoria. También se utilizarán mecanismos óptimos para el parseo de archivos WARC.

- **Mantenibilidad:** El producto estará diseñado previendo posibles cambios para su mantenimiento correctivo o de remoción de errores, mantenimiento adaptativo o cambios en

el ambiente de trabajo y para mantenimiento perfectivo que puedan beneficiar a los usuarios, mejorar la competencia y el rendimiento. Las mismas pueden consistir tanto en la mejora de las funcionalidades ya existentes como en la adaptación del programa a nuevas funciones, a lo largo del tiempo de vida del mismo.

- **Portabilidad:** El software debe ser soportado por cualquier navegador, reflejando su independencia ante cualquier migración, sea cual sea el entorno.

3.1.3. Tecnologías usadas en la aplicación

Software:

- ReST como arquitectura de software para la transferencia de datos.
- HTTP como protocolo de comunicación y para la integración con el prototipo.
- Ruby como lenguaje de programación con su framework Rails.
- Estándar de formato de archivos WARC (ISO, 2009).

Hardware:

- Se utilizará un Servidor HP ProLiant ML110 G7 E3-1220 1P con un procesador Intel Quad Core Xeon de 3.1 GHz y 10 GB de RAM.

3.1.4. Arquitectura de la aplicación

Al tratarse de una aplicación Web, se basa en la arquitectura Cliente – Servidor, mostrada en la Figura 10.

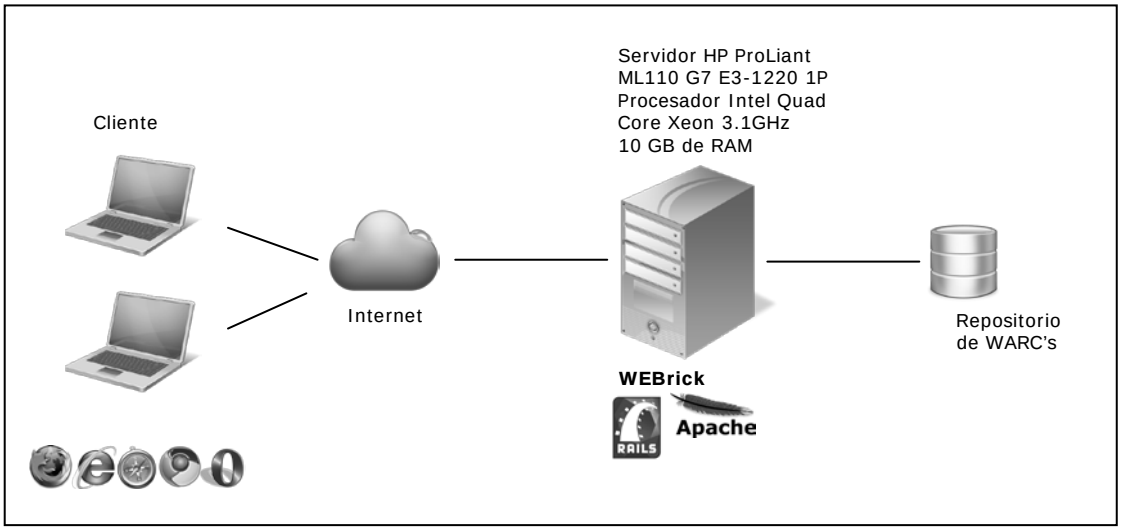
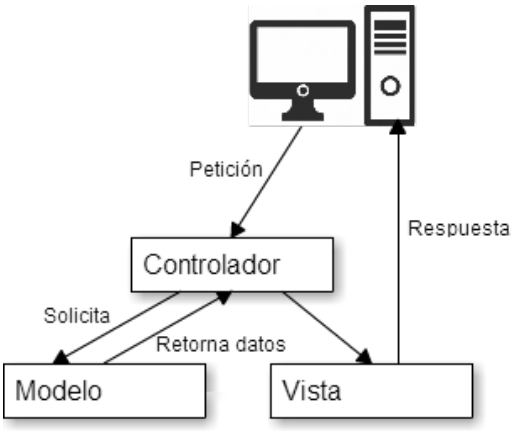


Figura 8. Arquitectura Cliente-Servidor

La aplicación desarrollada en este TEG está basada en una arquitectura Modelo – Vista – Controlador, mostrada en la Figura 11.



MODELO

- models
  - contacto.rb
  - favorito.rb
  - historial.rb
  - metricas.rb
  - pais.rb
  - profesion.rb
  - usuario.rb
  - warc.rb

CONTROLADOR

- controllers
  - application_controller.rb
  - despliegue_controller.rb
  - home_controller.rb
  - usuario_controller.rb

VISTA

- views
  - correos_usuario
    - correo_contacto.html.haml
    - correo_contacto.text.erb
    - correo_olvide_mi_clave.html.haml
    - correo_olvide_mi_clave.text.erb
  - despliegue
    - buscar.html.haml
    - buscar2.html.haml
  - home
    - _barra.html.haml
    - archivo_web.html.haml
    - busqueda_input.html.haml
    - busqueda_input_procesar.json
    - contacto.html.haml
    - despliegue_barra.html.haml
    - index.html.haml
    - prueba_calendar.html.haml
    - quienes_somos.html.haml
    - seleccionar_url.html.haml
  - layouts
    - _modal.html.haml
    - application.html.haml
    - externo.html.haml
    - otros.html.haml
    - otros2.html.haml
  - usuario
    - agregar_favorito.js.erb
    - cambiar_clave.html.haml
    - cambiar_datos.html.haml
    - eliminar_favorito.js.erb
    - listar_favoritos.html.haml
    - listar_historial.html.haml
    - olvide_clave.html.haml
    - registro.html.haml

Figura 9. Arquitectura MVC para el Prototipo de Archivo Web de Venezuela

3.1.5. Usuarios de la aplicación

El módulo de acceso y despliegue de archivos WARC para el Prototipo de Archivo Web en Venezuela, al ser una aplicación Web, está dirigido a investigadores, historiadores, académicos y al público en general. Por tal motivo, la aplicación cuenta con sólo un grupo de usuarios:

Usuario Público: corresponde a todo aquel usuario que accede a la aplicación a través de Internet. Este grupo se divide en dos tipos:

> **No registrado:** este tipo de usuario puede buscar el sitio que desea consultar, acceder y navegar el sitio desplegado por la aplicación, de acuerdo a la versión en el tiempo seleccionada por él del sitio consultado por medio de la URL del mismo, ver las estadísticas basadas en el número de versiones existentes del sitio en el Archivo Web.
>
> **Registrado:** este tipo de usuario cuenta con todas las funcionalidades del No registrado, y permite además almacenar en su cuenta sus sitios de interés etiquetándolos como *Favoritos* y un registro historial de los sitios consultados, de modo tal que pueda consultar estas listas cuando lo desee.

3.1.6. Prototipos de Diagramación de Interfaz

Durante la propuesta del TEG se presentan una serie de bocetos, los cuales fueron aprobados para continuar la aplicación, y a partir de estos se realizaron los prototipos de interfaz haciendo énfasis en la diagramación de las vistas de la aplicación.

En la Figura 12 se muestra el prototipo de la página principal de la aplicación, la cual presenta la cabecera que incluye en ella el logo del prototipo de Archivo Web de Venezuela, un menú con las opciones de la aplicación, el módulo de autenticación y la opción de registro. También posee una barra de búsqueda, una sección de contenido, una sección donde se muestra información relevante del prototipo, y por último el pie de página.

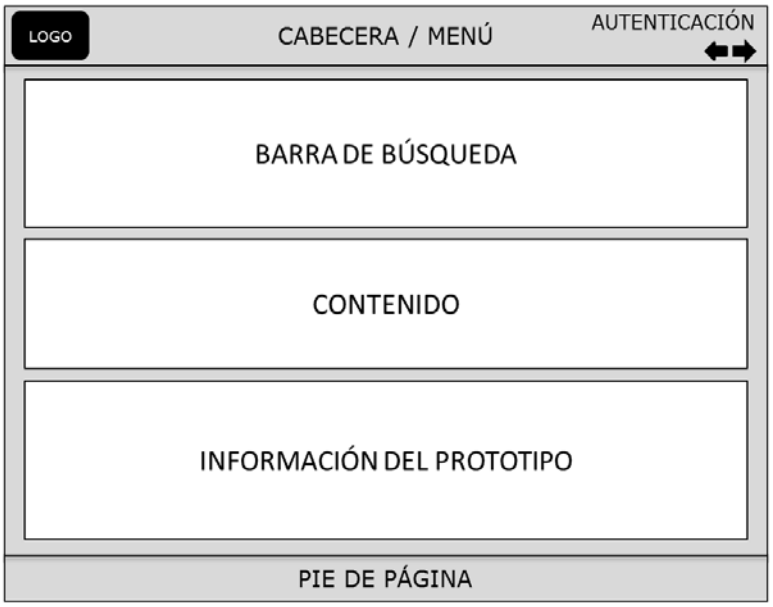


Figura 10. Prototipo de interfaz para la página principal de la aplicación

En la Figura 13 se muestra el prototipo de interfaz del resultado inicial de cualquier búsqueda que haya realizado el usuario, el cual es similar al anterior, pero con la barra de búsqueda más pequeña, y dos secciones de contenido: una para el resultado de la búsqueda como tal y otra que presenta opciones relevantes del prototipo; y el pie de página.

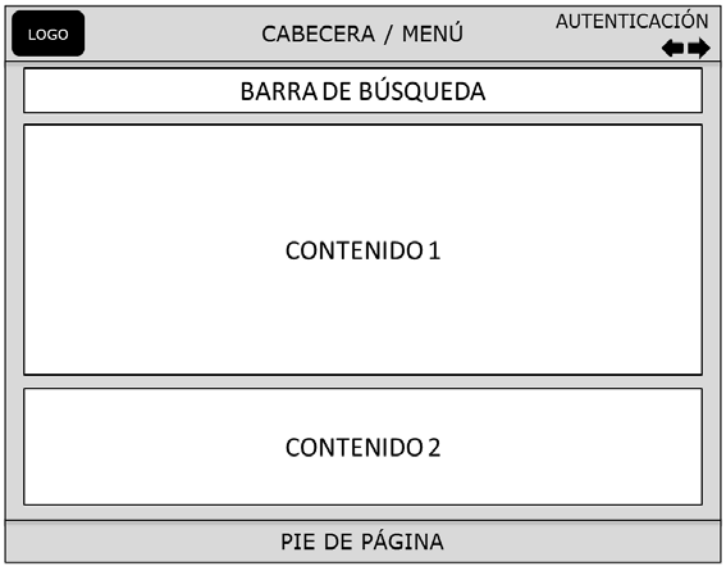


Figura 11. Prototipo de interfaz para el resultado inicial de búsqueda

En la Figura 14 se muestra el prototipo de interfaz del resultado final de una búsqueda, una vez seleccionada la versión de la página a desplegar, por lo que presenta la cabecera que se maneja en los dos prototipos anteriores, una sección de contenido desplegable por el usuario con información del

despliegue realizado, y una sección de contenido donde es desplegada como tal la versión que seleccionó el usuario del sitio buscado.

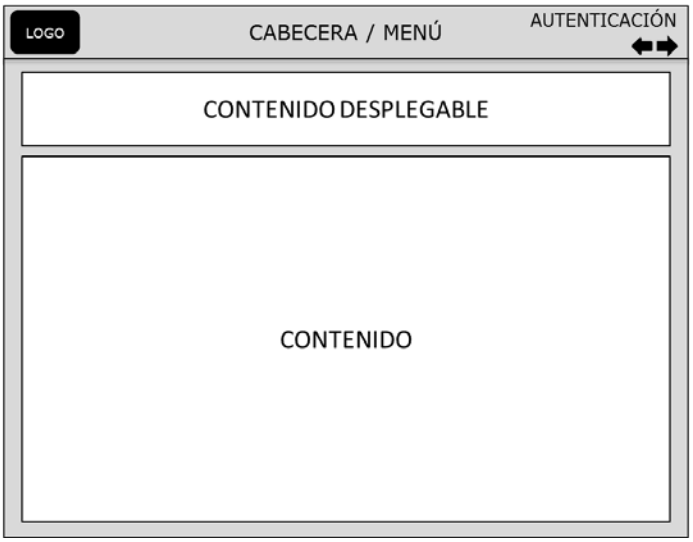


Figura 12. Prototipo de interfaz final de una búsqueda

3.2. Planteamiento:

En esta fase del ciclo de vida del proyecto, a través de las Historias de usuario se otorga y estima prioridades y cantidad de esfuerzo en cada uno de los requerimientos de la aplicación. Basado en estas, se realizó el diagrama de Casos de Uso y el Modelo de Datos, los cuales permiten definir las interacciones entre los usuarios y la aplicación, y como se relacionan los datos entre sí para formar la estructura sobre la que se sustenta aplicación.

3.2.1. Historias de usuarios

A continuación se presentan las Historias de usuario que surgieron a partir de los requerimientos funcionales para el desarrollo de la aplicación (Ver Tabla 4):

Tabla 4. Historias de usuario

Número: 1	**Nombre:** Diseño de la aplicación
Usuario: Mantura Kabchi – Miguel Martínez	**Iteración Asignada:** N/A
Tipo: Nueva	**Tiempo Estimado:** 2 meses
Descripción: Se realiza la captación de requerimientos y a partir de estos, el diseño de la aplicación Web que abarca la Arquitectura de la aplicación, Diagrama de Actividades, Casos de Uso	

Número: 2	**Nombre:** Instalación del ambiente de trabajo
Usuario: Mantura Kabchi – Miguel Martínez	**Iteración Asignada:** 0, 5, 9
Tipo: Nueva	**Tiempo Estimado:** 15 días
Descripción: Se procede a instalar en un ambiente local todas las herramientas necesarias para el desarrollo de la aplicación	

Número: 3	**Nombre:** Desarrollo de la Interfaz de Usuario
Usuario: Mantura Kabchi – Miguel Martínez	**Iteración Asignada:** 1, 4
Tipo: Nueva	**Tiempo Estimado:** 15 días
Descripción: Se evalúan diversas interfaces de usuario correspondientes a antecedentes de Archivado Web y a partir de esta evaluación se procede a desarrollar la interfaz propia de la aplicación	

Número: 4	**Nombre:** Registro de usuario
Usuario: Mantura Kabchi – Miguel Martínez	**Iteración Asignada:** 4
Tipo: Nueva	**Tiempo Estimado:** 10 días
Descripción: El sistema debe permitir el registro opcional para los usuarios de este. Se deben solicitar al usuario una serie de datos para completar el registro (*nombre*, *correo*, *clave*, entre otros)	

Número: 5	**Nombre:** Autenticación de usuario
Usuario: Mantura Kabchi – Miguel Martínez	**Iteración Asignada:** 4
Tipo: Nueva	**Tiempo Estimado:** 10 días
Descripción: El sistema debe permitir un inicio de sesión opcional para los usuarios de este. Se debe validar *clave* y *correo.* El sistema mediante este inicio de sesión debe proveer al usuario un historial de búsquedas y de un listado de favoritos correspondientes al usuario	

Número: 6	**Nombre:** Implementación de la Búsqueda por URL
Usuario: Mantura Kabchi – Miguel Martínez	**Iteración Asignada:** 7
Tipo: Nueva	**Tiempo Estimado:** 15 días
Descripción: El sistema debe proveer un campo de búsqueda donde el usuario introduce el URL del sitio Web que desee consultar	

Número: 7	**Nombre:** Selección de versión en el calendario
Usuario: Mantura Kabchi – Miguel Martínez	**Iteración Asignada:** 7

Tipo: Nueva	**Tiempo Estimado:** 15 días
Descripción: El sistema debe permitir al usuario la selección, a través de un calendario, de la versión en el tiempo del sitio Web solicitado de acuerdo a las cosechadas	

Número: 8	**Nombre:** Despliegue del sitio Web
Usuario: Mantura Kabchi – Miguel Martínez	**Iteración Asignada:** 2, 3, 8, 12
Tipo: Nueva	**Tiempo Estimado:** 3 meses
Descripción: El sistema debe desplegar el sitio Web solicitado por el usuario de modo que este pueda navegar a través de él	

Número: 9	**Nombre:** Gestionar Favoritos
Usuario: Mantura Kabchi – Miguel Martínez	**Iteración Asignada:** 6
Tipo: Nueva	**Tiempo Estimado:** 5 días
Descripción: El sistema debe permitir al usuario gestionar sus búsquedas etiquetadas como *Favoritos* pertenecientes a su sesión como usuario	

Número: 10	**Nombre:** Registro historial
Usuario: Mantura Kabchi – Miguel Martínez	**Iteración Asignada:** 6
Tipo: Nueva	**Tiempo Estimado:** 5 días
Descripción: El sistema debe permitir al usuario consultar a su registro historial de búsqueda perteneciente a su sesión como usuario	

Número: 11	**Nombre:** Estadísticas de versiones
Usuario: Mantura Kabchi – Miguel Martínez	**Iteración Asignada:** 7
Tipo: Nueva	**Tiempo Estimado:** 15 días
Descripción: El sistema debe proveer estadísticas al usuario acerca de las versiones cosechadas del sitio Web solicitado, usando gráficas para ello.	

Número: 12	**Nombre:** Información del sitio
Usuario: Mantura Kabchi – Miguel Martínez	**Iteración Asignada:** 10
Tipo: Nueva	**Tiempo Estimado:** 2 días
Descripción: El sistema debe proveer al usuario información del sitio como iniciativa de preservación Web.	

Número: 13	**Nombre:** Formulario de contacto

Usuario: Mantura Kabchi – Miguel Martínez	**Iteración Asignada:** 11
Tipo: Nueva	**Tiempo Estimado:** 1 día
Descripción: El sistema debe proveer al usuario un medio de comunicación vía correo electrónico con el administrador del sitio.	

Número: 14	**Nombre:** Buscar sitios por colecciones
Usuario: Mantura Kabchi – Miguel Martínez	**Iteración Asignada:** 13
Tipo: Nueva	**Tiempo Estimado:** 1 día
Descripción: El sistema debe proveer al usuario una búsqueda avanzada a través de colecciones de acuerdo al tema contenido en los archivos WARC	

3.2.2. Casos de Uso

A continuación se expone el diagrama de Casos de Uso realizado según los requerimientos de la aplicación Web (Ver Figura 15):

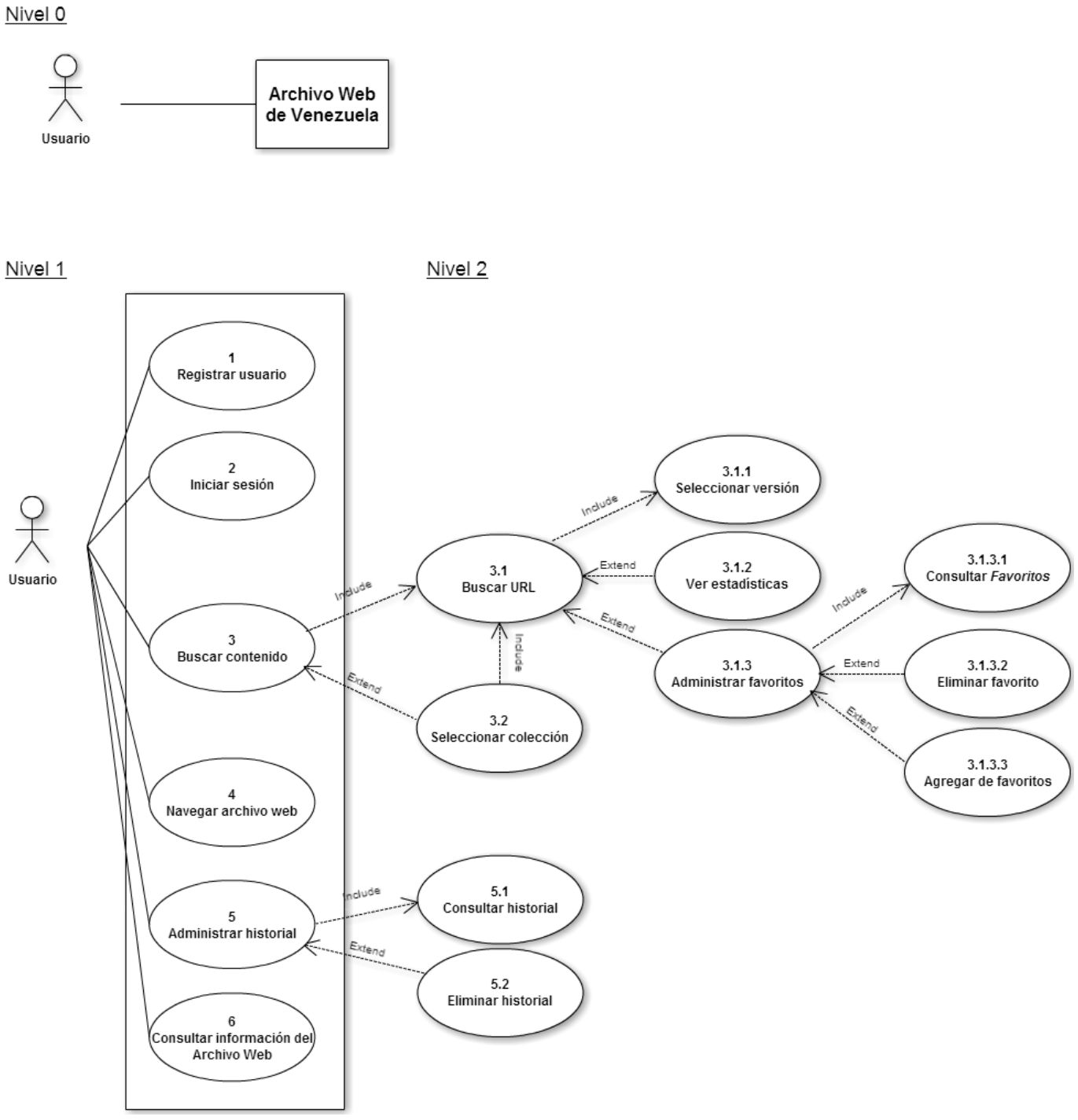


Figura 13. Diagrama de Casos de uso

Especificación de Casos de Uso

Tabla 5. Especificaciones de Casos de Uso

Especificación del caso de uso: Registrar usuario		
ID	AW-1	
Nombre	Registrar usuario	
Descripción	Permite registrar a nuevos usuarios en el sitio de manera opcional	
Actores	Usuario	
Precondiciones	El usuario no debe haberse registrado anteriormente	
Flujo normal de eventos	**Paso**	**Acción**
	1	El usuario ingresa en la opción de registrarse
	2	El usuario ingresa sus datos de registro
	3	Los datos son validados del lado del cliente
	4	Se envía la solicitud de registro del usuario
	5	Los datos son validados del servidor
	Si todos los datos introducidos son válidos se registra en la base de datos el nuevo usuario	
	Se redirecciona al usuario al inicio del sitio	
	Se envía un correo electrónico al usuario dándole la bienvenida al sitio	
Postcondiciones	El usuario queda registrado en la base de datos del sitio	
Excepciones	**Paso**	**Acción**
	1	Si el usuario no introduce los datos completos al momento de registrarse, se le indica qué datos le faltan y continúa completando los mismos
	2	Si alguno de los datos introducidos por el usuario no es correcto, se le indica el error y continúa introduciendo los datos
Anotaciones	Los datos a solicitar en el registro son: nombre, correo, clave, motivo de registro	

Especificación del caso de uso: Iniciar sesión		
ID	AW-2	
Nombre	Iniciar sesión	
Descripción	Permite al usuario autenticarse en el sitio de forma opcional	
Actores	Usuario	
Precondiciones	El usuario deber estar registrado y por ende existir en la base de datos del sitio.	
Flujo normal de eventos	**Paso**	**Acción**
	1	El usuario ingresa su correo y contraseña en el sistema y hace clic en el botón “Ingresar”
	2	Los datos son validados del lado del cliente
	3	Se envía la solicitud al servidor
	4	Los datos son validados en el servidor

	5	El sistema redirecciona al usuario al inicio mostrando nuevas opciones basadas en su perfil
Postcondiciones	El usuario puede interactuar con el sistema a través de nuevas opciones	
Excepciones	**Paso**	**Acción**
	1	Si el usuario no introduce los datos completos al momento de iniciar sesión, se le indica qué datos le faltan y continúa completando dichos datos para la autenticación
	2	Si alguno de los datos introducidos por el usuario al momento de iniciar sesión no es correcto, se le indica el error y puede volver a introducir los datos de la autenticación
Anotaciones	Los datos a solicitar para el inicio de sesión del usuario son: correo electrónico y clave.	

Especificación del caso de uso: Seleccionar contenido		
ID	AW-3	
Nombre	Seleccionar contenido	
Descripción	Permite al usuario realizar una búsqueda de un determinado sitio de su interés	
Actores	Usuario	
Precondiciones	N/A	
Flujo normal de eventos	**Paso**	**Acción**
	1	El usuario puede buscar por URL (Caso de uso 3.1 Buscar URL) o por colección (Caso de Uso 3.2 Seleccionar colección)
Postcondiciones		
Excepciones	N/A	
Anotaciones		

Especificación del caso de uso: Buscar URL		
ID	AW-3.1	
Nombre	Buscar URL	
Descripción	Permite al usuario realizar una búsqueda basada en el URL de un determinado sitio de su interés	
Actores	Usuario	
Precondiciones	N/A	
Flujo normal de eventos	**Paso**	**Acción**
	1	El usuario ingresa un URL en el campo de búsqueda y hace clic en “Buscar”
	2	Se valida el URL ingresado
	3	Se le da respuesta al usuario mostrando las distintas versiones del URL solicitado a través de un calendario
Postcondiciones	El usuario podrá elegir la versión que desea desplegar y navegar	
Excepciones	**Paso**	**Acción**
	1	Si el usuario no introduce un URL válido se mostrará un

		mensaje de error y se le permitirá ingresar nuevamente el URL
	2	Si el URL ingresado por el usuario no produce resultados en la búsqueda, ya que no existen rastreos del mismo, se le notificará al usuario y se le permitirá ingresar nuevamente otra URL.
Anotaciones	El botón "Buscar" permanecerá inactivo mientras el usuario no introduzca un URL válido	

Especificación del caso de uso: Seleccionar versión		
ID	AW-3.1.1	
Nombre	Seleccionar versión	
Descripción	Permite elegir una versión del sitio a través del URL solicitado	
Actores	Usuario	
Precondiciones	El usuario debe haber realizado la búsqueda de un URL	
Flujo normal de eventos	**Paso**	**Acción**
	1	El usuario selecciona la versión del sitio representado por el URL previamente buscado haciendo clic en la fecha señalada por el calendario
	2	Se despliega la versión del sitio web seleccionada
Postcondiciones	El usuario podrá navegar a través de la versión del sitio seleccionada	
Excepciones	N/A	
Anotaciones	Las versiones del sitio representado por el URL ingresado por el usuario estarán señaladas dentro de un calendario	

Especificación del caso de uso: Ver estadísticas		
ID	AW-3.1.2	
Nombre	Ver estadísticas	
Descripción	Permite consultar las estadísticas del sitio Web buscado por el usuario	
Actores	Usuario	
Precondiciones	El usuario debe haber realizado la búsqueda de un URL	
Flujo normal de eventos	**Paso**	**Acción**
	1	El usuario activa el evento "Ver estadísticas del sitio"
	2	Se despliega las estadísticas del sitio consultado
Postcondiciones	El usuario podrá conocer las estadísticas del sitio Web buscado	
Excepciones	N/A	
Anotaciones	Las estadísticas serán mostradas a través de gráficos de barras	

Especificación del caso de uso: Administrar *Favoritos*	
ID	AW-3.1.3
Nombre	Administrar *Favoritos*

Descripción	Permite al usuario consultar, agregar y eliminar sus búsquedas etiquetadas como *Favoritos* pertenecientes a su sesión como usuario	
Actores	Usuario	
Precondiciones	El usuario debe haberse autenticado	
Flujo normal de eventos	**Paso**	**Acción**
	1	El usuario selecciona una de las tres opciones: Consultar *Favoritos* (Ver caso de uso 3.1.3.1 Consultar *Favoritos*), Eliminar *Favorito* (Ver caso de uso 3.1.3.2 Eliminar *Favorito*), Agregar a *Favoritos* (Ver caso de uso 3.1.3.3 Agregar a *Favoritos*)
Postcondiciones	El usuario consultó, eliminó o agregó alguna de sus búsquedas a sus *Favoritos*	
Excepciones	N/A	
Anotaciones	Los *Favoritos* le permiten al usuario guardar un acceso directo a las búsquedas que sean de su interés	

Especificación del caso de uso: Consultar *Favoritos*		
ID	AW-3.1.3.1	
Nombre	Consultar *Favoritos*	
Descripción	Permite al usuario consultar sus búsquedas etiquetadas como *Favoritos* pertenecientes a su sesión como usuario	
Actores	Usuario	
Precondiciones	El usuario debe haberse autenticado	
Flujo normal de eventos	**Paso**	**Acción**
	1	El usuario activa el evento "*Favoritos*"
	2	Se despliegan las búsquedas marcadas como *Favoritos* del usuario
	3	
Postcondiciones	El ususario podrá consultar su lista de *Favoritos*	
Excepciones	**Paso**	**Acción**
	1	Si el usuario no posee ninguna búsqueda etiquetada como *Favorito*, obtendrá un mensaje indicándole que no posee ningún *Favorito* y que puede añadirlos al realizar sus búsquedas.
Anotaciones		

Especificación del caso de uso: Eliminar *Favorito*		
ID	AW-3.1.3.2	
Nombre	Eliminar *Favorito*	
Descripción	Permite al usuario eliminar sus búsquedas etiquetadas como *Favoritos* pertenecientes a su sesión como usuario	
Actores	Usuario	
Precondiciones	El usuario debe haberse autenticado, haber activado el evento "*Favoritos*" y poseer alguna búsqueda etiquetada como *Favorito*	
Flujo normal de eventos	**Paso**	**Acción**
	1	El usuario activa el evento "*Favoritos*"

	2	Se despliegan las búsquedas marcadas como *Favoritos* del usuario
	3	El usuario activa el evento "*Eliminar Favorito*" correspondiente a una búsqueda
	4	El usuario debe confirmar su acción
	5	Se despliegan nuevamente las búsquedas marcadas como *Favoritos* del usuario
Postcondiciones	El usuario eliminó una de sus búsquedas a sus *Favoritos*	
Excepciones	N/A	
Anotaciones		

Especificación del caso de uso: Agregar a *Favoritos*		
ID	AW-3.1.3.3	
Nombre	Agregar a *Favoritos*	
Descripción	Permite al usuario añadir una búsqueda etiquetándola como *Favorito,* perteneciente a su sesión como usuario	
Actores	Usuario	
Precondiciones	El usuario debe haberse autenticado, debe haber realizado una búsqueda y la búsqueda no debe haber sido etiquetada previamente como favorito	
Flujo normal de eventos	**Paso**	**Acción**
	1	El usuario activa el evento "Agregar a *Favoritos*"
	2	Se despliega una ventana pop-up donde el usuario podrá colocarle un nombre al *Favorito* que está añadiendo
	3	El usuario hace clic en "Aceptar" para cerrar el pop-up
Postcondiciones	El usuario agregó su búsqueda a sus *Favoritos*	
Excepciones	N/A	
Anotaciones		

Especificación del caso de uso: Seleccionar colección		
ID	AW-3.2	
Nombre	Seleccionar colección	
Descripción	Permite al usuario realizar una búsqueda basada en la selección de una colección de su interés	
Actores	Usuario	
Precondiciones	N/A	
Flujo normal de eventos	**Paso**	**Acción**
	1	El usuario selecciona en la lista desplegable la colección de su interés
	2	Se le da respuesta al usuario mostrando las distintas URLs de la colección que seleccionó
Postcondiciones	El usuario podrá elegir la URL que desee perteneciente a la colección que seleccionó	
Excepciones	N/A	
Anotaciones		

Especificación del caso de uso: Navegar archivo web		
ID	AW-4	
Nombre	Navegar archivo web	
Descripción	Permite al usuario navegar la versión del sitio Web que haya solicitado como si del sitio original se tratase	
Actores	Usuario	
Precondiciones	Haber realizado la búsqueda de un URL y haber seleccionado la versión que se quiere navegar del mismo	
Flujo normal de eventos	**Paso**	**Acción**
	1	El usuario navega en el sitio Web
Postcondiciones	El usuario navega en el sitio Web	
Excepciones	N/A	
Anotaciones		

Especificación del caso de uso: Consultar historial		
ID	AW-5.1	
Nombre	Consultar historial	
Descripción	Permite al usuario consultar su historial de búsquedas perteneciente a su sesión como usuario	
Actores	Usuario	
Precondiciones	El usuario debe haberse autenticado	
Flujo normal de eventos	**Paso**	**Acción**
	1	El usuario activa el evento "*Ver mi historial*"
	2	Se despliega un listado de su historial de búsquedas anteriores
Postcondiciones	El ususario podrá consultar su historial de búsquedas	
Excepciones	N/A	
Anotaciones		

Especificación del caso de uso: Eliminar historial		
ID	AW-5.2	
Nombre	Eliminar historial	
Descripción	Permite al usuario eliminar su historial de búsquedas perteneciente a su sesión como usuario	
Actores	Usuario	
Precondiciones	El usuario debe haberse autenticado y haber realizado alguna búsqueda estando autenticado	
Flujo normal de eventos	**Paso**	**Acción**
	1	El usuario activa el evento "*Ver mi historial*"
	2	Se despliega un listado de su historial de búsquedas anteriores
	3	El usuario activa el evento "*Eliminar del historial*" correspondiente a una búsqueda realizada previamente
	4	El usuario debe confirmar su acción

	5	Se despliegan nuevamente su historial de búsquedas
Postcondiciones	El usuario eliminó una de sus búsquedas de su historial	
Excepciones	N/A	
Anotaciones		

Especificación del caso de uso: Consultar información del Archivo Web		
ID	AW-6	
Nombre	Consultar información del Archivo Web	
Descripción	Permite al usuario consultar información sobre el Archivo Web	
Actores	Usuario	
Precondiciones	Haber realizado la búsqueda de un URL y haber seleccionado la versión que se quiere navegar del mismo	
Flujo normal de eventos	**Paso**	**Acción**
	1	El usuario navega en el sitio Web
Postcondiciones	El usuario navega en el sitio Web	
Excepciones	N/A	
Anotaciones		

3.2.3. Modelo de Datos

A continuación se presenta el Modelo de Datos Relacional, el cual expone de manera gráfica como se relacionan los datos entre sí, formando la estructura sobre la cual se soporta la aplicación creada.

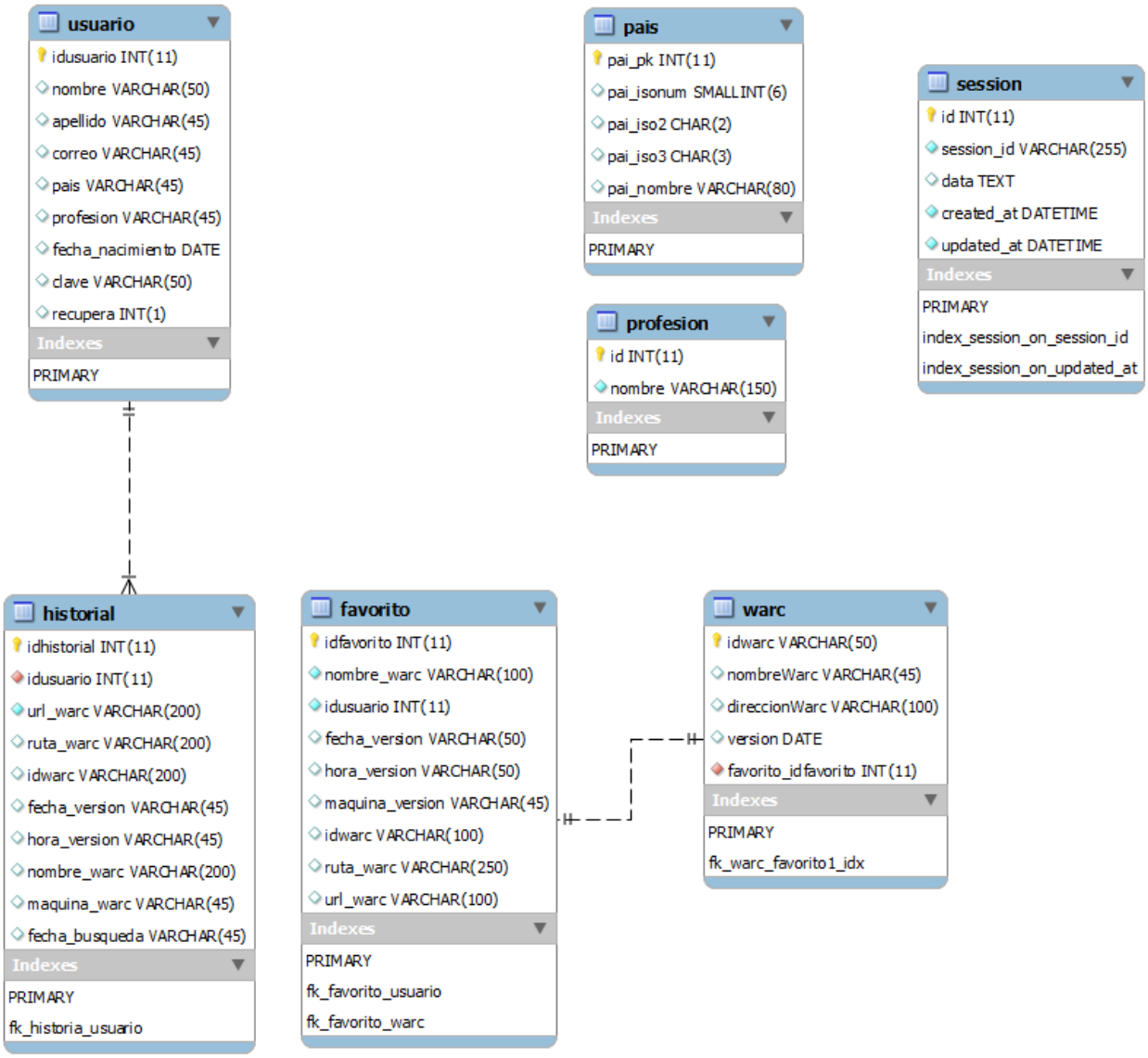


Figura 14. Modelo de datos

3.3. Iteraciones

Todo proyecto que siga la metodología XP se ha de dividir en iteraciones para lograr la evolución del proyecto de manera escalable. Para el desarrollo de este TEG se decidió que cada iteración representa una serie de historias de usuario relacionadas, que a lo sumo constituyen todo el desarrollo del módulo de acceso y despliegue de archivos WARC para el Prototipo de Archivo Web en Venezuela.

3.3.1. Iteración 0

- **Planificación:** En esta primera iteración el objetivo es la configuración del ambiente de desarrollo para la aplicación. También en esta iteración se analiza con mayor detalle las necesidades y requerimientos planteados en la propuesta de TEG. A partir de esto, se realizan los prototipos de Diagramación de Interfaces (ver punto 3.1.6 de este capítulo), las Historias de Usuario (ver punto 3.2.1 de este capítulo), el diagrama de Casos de Uso y sus respectivas especificaciones (ver punto 3.2.2 de este capítulo) y el Modelo de Datos (ver punto 3.2.3 de este capítulo).

Tabla 6. Planificación Iteración 0

Número de Iteración: 0	**Número de Historia:** 1 y 2
Fecha de Inicio: 2/05/2013	**Fecha de Fin:** 15/05/2013
Descripción: Configuración del ambiente de trabajo y diseño de la aplicación	**Tipo:** Desarrollo / Diseño

- **Descripción:** Para la configuración del servidor de desarrollo y de producción, en Linux Distro Ubuntu ambos, se instalan una serie de herramientas.

  En primer lugar, la instalación de cURL, que es una herramienta que permite transferir información con sintaxis de URL, soporta gran variedad de protocolos (HTTP, FTP, LDAP, entre otros) y funciona en gran variedad de sistemas operativos:

  ```
  $ apt-get install curl
  ```

  Una vez instalado cURL, se lleva a cabo la instalación de RVM (Ruby Version Manager) que es una herramienta que permite instalar más de una versión de Ruby, y además permite tener distintas librerías llamadas RubyGems y agruparlas por proyecto. La instalación de esto no es obligatoria para el desarrollo de la aplicación pero es preferible para mantener así el entorno de forma ordenada:

  ```
  $ bash < <( curl https://rvm.beginrescueend.com/install/rvm )
  ```

  Con el RVM 1.20.5 ya instalado, se prosigue a instalar la versión de Ruby actualmente estable 1.9.3 junto con el paquete OpenSSL para protocolos de seguridad a través del manejador RVM con el comando:

  ```
  $ rvm install 1.9.3 -C --with-openssl-dir=$HOME/.rvm/usr,--with- iconv-dir=$HOME/.rvm/usr
  ```

Al instalar Ruby ya se encuentra instalado el servidor Web WEBrick, el cual es usado para el desarrollo de la aplicación:
Al instalar el RVM, ya se encuentra instalado el RubyGems que es un gestor de paquetes de Ruby y se usa para distribuir todo tipo de programas y librerías, como por ejemplo el framework Rails:

```
$ gem install rails –version 3.2.13
```

De este modo se tiene instalado Ruby on Rails. Es creado un set de gemas para la aplicación en cuestión, es decir un gemset, que no es más que un directorio separado de gemas:

```
$ rvm gemset create gemas
```

Ya en este punto, se lleva a cabo la instalación de las gemas correspondientes a MySql que es el manejador de base de datos que se utilizará en el desarrollo de este proyecto:

```
$ gem install mysql2
```

Con todas estas herramientas instaladas, y una vez creado el proyecto es necesario modificar algunos archivos de configuración del proyecto, el archivo de rutas routes.rb, añadiéndole la siguiente línea para indicarle cual será la ruta de la página inicial: `root :to => 'home#index'`, y esta modificación `match ':controller(/:action(/:id))(.:format)'`, para indicar cuál es el formato de la peticiones a través del URL.

Además se agregan el resto de las gemas que vayan siendo necesarias a lo largo del desarrollo de la aplicación al archivo Gemfile, las cuales se instalan cuando se ejecuta el comando `bundle install` una vez modificado dicho archivo. Las gemas agregadas para la aplicación son:

bootstrap-sass (2.1.0.1)
coffee-rails (3.2.2)
coffee-script (2.2.0)
coffee-script-source (1.6.2, 1.4.0)
composite_primary_keys (5.0.9)
dynamic_form (1.1.4)
haml (3.1.7)
haml-rails (0.3.5)
hpricot (0.8.6)
jquery-rails (2.2.1, 2.1.3)
less-rails (2.2.5)
magic_encoding (0.0.2)
mysql2 (0.3.11)
nokogiri (1.5.5)
paperclip (3.3.1)
rails (3.2.13, 3.2.8)
sass-rails (3.2.6, 3.2.5)
therubyracer (0.10.2)
twitter-bootstrap-rails (2.1.4)
uglifier (2.0.1, 1.3.0)
unicorn (4.4.0)

La siguiente herramienta instalada para poder llevar a cabo el desarrollo de la aplicación, es Hanzo Warc Tools la cual se utiliza en este prototipo para acceder a los archivos WARC. La descarga de esta se hizo a través de: http://code.hanzoarchives.com/warc-tools/downloads
Una vez descargada se instalan las siguientes dependencias:

Python Setup tools python3-setuptools and python-setuptools

Python Unittest python-unittest2

Python 2.6 sudo apt-get install python2.6

Y finalmente se ejecutan los siguientes comandos:

./setup.py build

./setup.py install

- **Pruebas:** A continuación las pruebas realizadas en esta iteración

Tabla 7. Caso de prueba: 1. Iteración 0

Número de Caso de Prueba: 1	**Número de Historia de Usuario:** 2
Descripción: Imprimir un dato por pantalla a través de la consola de Ruby on Rails	
Resultado: 	

Tabla 8. Caso de prueba: 2. Iteración 0

Número de Caso de Prueba: 2	**Número de Historia de Usuario:** 2
Descripción: Crear una tabla en MySQL con un campo id	
Resultado:	

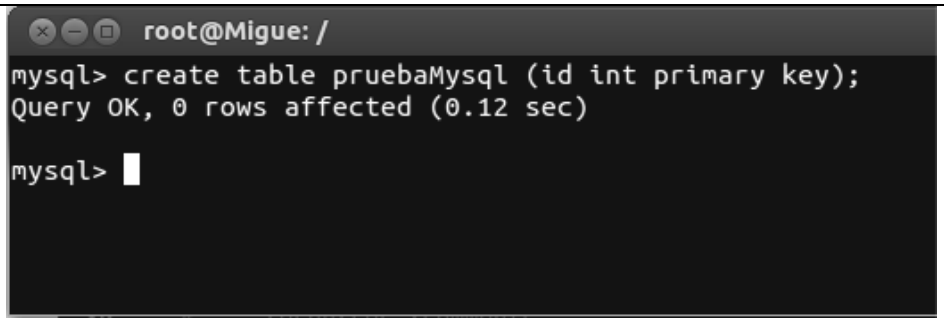
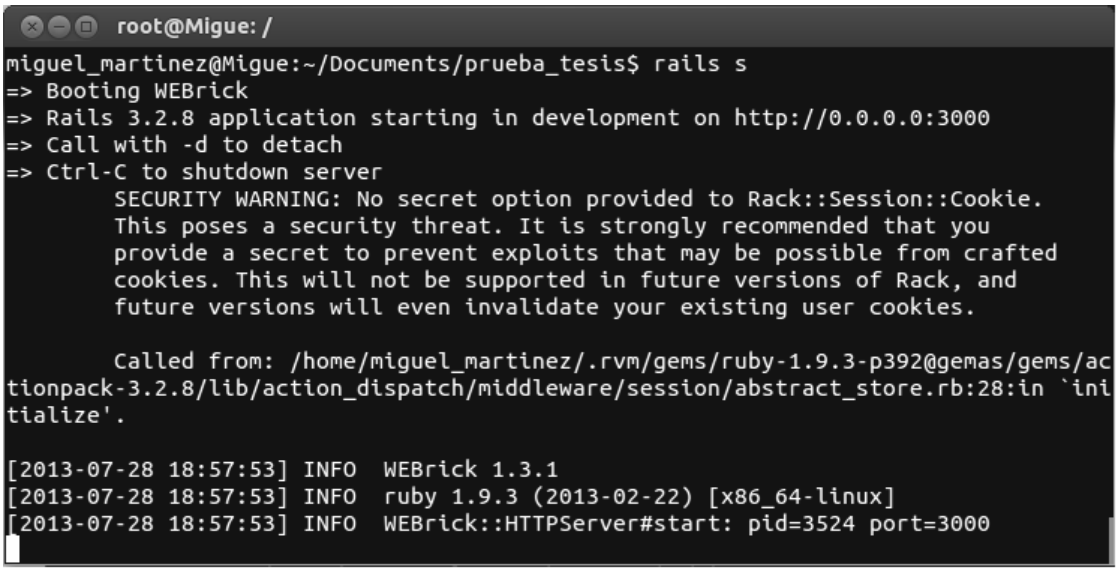


Tabla 9. Caso de prueba: 3. Iteración 0

Número de Caso de Prueba: 3	**Número de Historia de Usuario:** 2
Descripción: Levantar el servidor de rails a través de WEBrick	
Resultado:	



3.3.2. Iteración 1

- **Planificación:** En esta iteración el objetivo es desarrollar la interfaz gráfica de la aplicación basados en los prototipos de diagramación de interfaz realizados anteriormente, y de una evaluación de interfaces de los antecedentes.

Tabla 10. Planificación Iteración 1

Número de Iteración: 1	**Número de Historia:** 1 y 2
Fecha de Inicio: 17/05/2013	**Fecha de Fin:** 20/05/2013

Descripción: Desarrollo de la interfaz gráfica	**Tipo:** Desarrollo / Diseño

- **Diseño:** Previo a la diagramación de los prototipos de interfaz (ver punto 3.4.1.6 de este capítulo) se realiza una evaluación de interfaces antecedentes, la cual se muestra en la siguiente tabla:

Tabla 11. Comparativa entre iniciativas

Elementos a evaluar	Iniciativas			
	Internet Archive	**Pandora**	**Portuguese Web Archive**	**Library of Congress Web Archives**
Consistencia	La tipografía mostrada varía entre formato serif y formato sans-serif, lo que puede confundir al lector entre tratar de distinguir un enlace y un título. Los colores dentro del sitio varían de acuerdo al módulo, pero no de forma coherente. Esto también sucede con los formatos de tablas y la distribución de las mismas. Las metáforas utilizadas, aunque son pocas, identifican con claridad a lo que se refieren	La tipografía dentro del sitio no muestra variaciones al momento de enfatizar títulos y enlaces. Los primeros son presentados en un color azul en formato 'negrita', mientras que los segundos son presentados en color azul claro y subrayado. El formato de la página cambia totalmente al realizar una búsqueda	Los títulos y enlaces son diferenciados claramente. La tipografía no varía, solo en tamaño y colores para representar los títulos dentro del sitio. La distribución por cada pestaña no varía, pero los colores de los títulos cambian entre verde (subtítulos) y negro (títulos). El formato de párrafo no es justificado por lo que el texto puede verse desordenado. El tamaño de letra es consistente aunque muy pequeño.	La tipografía varía entre serif y sans-serif para diferenciar títulos y subtítulos, al igual que se utiliza subrayado para denotar un enlace. Las metáforas utilizadas no representan de forma clara que acciones realizan, por lo que si el usuario no sabe de inglés es muy complicado que comprenda el significado de las imágenes. La distribución de texto por cada módulo no es uniforme por lo que el usuario puede perderse y eventualmente cansarse.

Eficiencia	La interfaz no provee ejemplos de búsqueda visibles y la información mostrada en la página de inicio y enlaces subyacentes no es de gran utilidad para el propósito del sitio Web. Las búsquedas están muy bien implementadas y las respuestas iniciales son rápidas, por lo cual son satisfactorias aunque solo permitan URL. A partir de este punto la información proporcionada por el sitio está muy bien señalada. Las respuestas son satisfactorias	La interfaz no provee información inmediata y la que existe no posee mucha utilidad, por lo que hay que utilizar el módulo de Ayuda para saber cómo utilizar la barra de búsqueda. Los resultados tardan medianamente en encontrarse pero los mismos son diversos y detallados de acuerdo a los parámetros (palabra o frase) seleccionados. Aunque al buscar por URL los resultados no son específicos al sitio que se quiere encontrar. Las respuestas no son satisfactorias	La interfaz no provee ejemplos de búsqueda por lo que para asegurarse hay que utilizar el módulo de búsqueda avanzada. Hay muy poca información inicialmente, aunque el módulo de ayuda es bastante informativo. Las búsquedas devuelven resultados detallados de acuerdo a los parámetros, aunque mientras más parámetros se introduzcan mayor es el tiempo de respuesta. Una vez seleccionado el sitio tarda un tiempo en desplegarse y la navegación dentro del mismo es lenta. Las respuestas son medianamente satisfactorias.	La interfaz no ofrece información suficiente para la búsqueda de sitios. Pero una mejora que posee es el autocompletado y la elección del formato a buscar sin tener que ir a una interfaz de búsqueda avanzada. Las respuestas son rápidas con respecto a las otras iniciativas comparadas, y muy bien documentadas. Luego de este punto el orden de la información mejora haciéndola más comprensible. Al elegir el sitio a desplegar, los tiempos de respuesta aumentan aunque no considerablemente. Las respuestas son satisfactorias
Usabilidad	El sistema es de fácil aprendizaje. Es medianamente intuitivo por la falta de ejemplos al momento de realizar las búsquedas, además la barra no resalta. Una vez aprendido a utilizar el sistema, es bastante satisfactorio por su eficiencia. Por el lado estético deja mucho que desear por su inconsistencia y mal uso de espacios, además de no ser una interfaz moderna y llamativa. Es una interfaz estática y llena de texto que tiende a cansar y dar sensación de desorden.	El sistema no es de fácil aprendizaje. Poco intuitivo ya que el campo de búsqueda no destaca y el hecho de que cambie drásticamente de interfaces al realizar búsquedas puede confundir. Su baja eficiencia y consistencia en las búsquedas no permite que sea completamente satisfactorio. Estéticamente no es llamativa debido a que no se ha modernizado ni cambiado desde 2004. La interfaz es estática lo que tiende a disminuir la calidad y los tiempos de respuesta.	El sistema es de fácil aprendizaje. Es intuitivo por ser minimalista y dar completo énfasis en la búsqueda. El sistema es medianamente satisfactorio porque los tiempos de respuestas disminuyen su eficiencia. Estéticamente es la más llamativa por ser minimalista y con elementos modernos, lo que le permite tener una mejor distribución y no cansar al usuario con innumerables textos e imágenes.	El sistema es de fácil aprendizaje. Es medianamente intuitivo por el uso de metáforas. Una vez aprendido a utilizar el sistema, es bastante satisfactorio por su eficiencia. Estéticamente, aunque posee elementos dinámicos y llamativos, la interfaz está muy cargada lo que la hace ver desordenada y puede cansar al usuario.

Para desarrollar la interfaz gráfica se hace uso de Bootstrap, el cual es un framework de Twitter que ofrece una serie de plantillas CSS y ficheros Javascript. Para el retoque de las imágenes se usa la herramienta gráfica Adobe Photoshop CS6.

- **Codificación:** Dado que la aplicación se está desarrollando con Ruby on Rails, el cual utiliza el patrón arquitectónico MVC, las Vistas tienen un marco común llamado *layout*. Para el caso de estudio, se tienen tres *layouts:* externo (para todas las páginas cuando el usuario no ha iniciado

sesión), interno (para todas las páginas cuando el usuario ha iniciado sesión) y el application (para la página del despliegue propiamente). En la Figura 17 se muestran estos *layouts:*

```
!!!
%html
  %head
    %title Archivo Web de Venezuela
    = stylesheet_link_tag    "application", :media => "all"
    = javascript_include_tag "application"
    = csrf_meta_tags

  %body
    .navbar.navbar-inverse.navbar-fixed-top
      .navbar-inner
        .container
          =link_to "Archivo Web de Venezuela", {:action => "index", :controller => "home"},{:class => "brand"}
          %ul.nav
            %li.active
              %a{:href => ""} Inicio
            %li
              %a{:href => ""} Quiénes somos
            %li
              %a{:href => ""} Contacto
          %ul.nav.pull-right
            %li.dropdown
              %a{:class => "btn-action dropdown-toggle", "data-toggle" => "dropdown", :href => "#"}
                Iniciar sesión
                %b.caret
              %ul.dropdown-menu
                .espacio
                  %li
                    = form_tag :action => "index", :controller => "home" do
                      = text_field :cliente, :correo, :size => 15, :placeholder => "Correo"
                      = password_field :cliente, :clave, :size => 15, :placeholder => "Clave"
                      = submit_tag "Ingresar»", :class => "btn btn-primary"
                      =link_to "Olvidé mi clave"
                %li.divider
                %li
                  ¿No te has registrado?
                  %br
                  =link_to "Hazlo aquí"
            %li
              =link_to "Registrarse"
```

Figura 15. Código layout externo.html.haml. Iteración 1

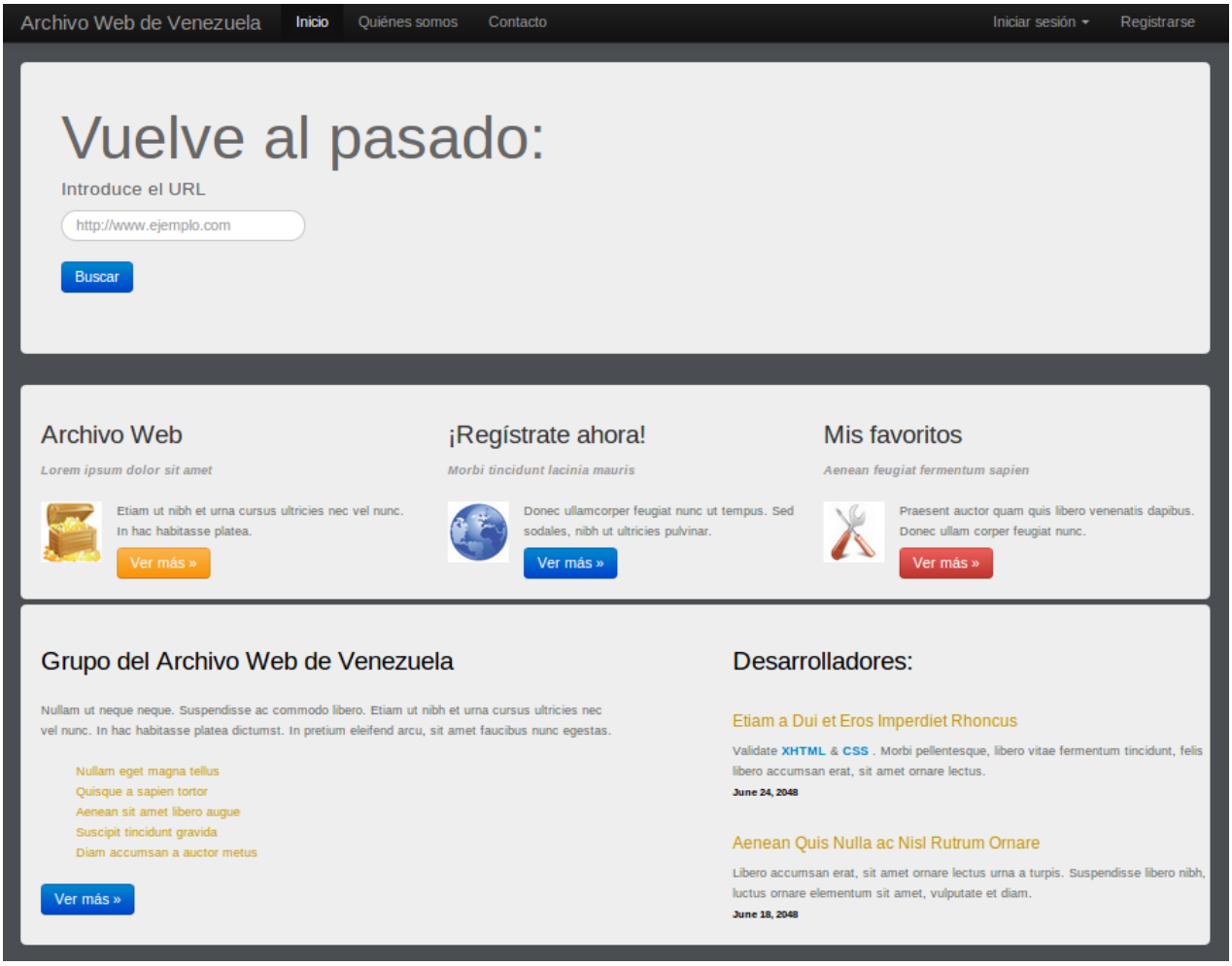


Figura 16. Interfaz externo.html.haml. Iteración 1

```
1  !!!
2  %html
3    %head
4      %title Archivo Web de Venezuela
5
6      = stylesheet_link_tag    "body"
7      = javascript_include_tag "application"
8      = csrf_meta_tags
9
10   %body
11     = render :partial => "home/barra"
12     .container
13       %br
14       %br
15       %br
16       = yield
17
18
19
20
```

Figura 17. Código application HAML. Iteración 1

El *layout* application.html.haml (Ver Figura 19) utiliza barra.html.haml, el cual es renderizado parcialmente a través de la opción de Rails = render :partial => "home/barra", de este modo el CSS del sitio desplegado no se ve afectado por el de la aplicación.

_barra.html.haml

```
!!!
%html
  %head
    %title Archivo Web de Venezuela

    = stylesheet_link_tag    "bootstrap"
    = javascript_include_tag "application"
    = csrf_meta_tags

  %body
    .navbar.navbar-inverse.navbar-fixed-top
      .navbar-inner
        .container
          %a.brand{:action => "index", :controller => "home"} Archivo Web de Venezuela
          %ul.nav
            %li.active
              %a{:href => ""} Inicio
            %li
              %a{:href => ""} Quiénes somos
            %li
              %a{:href => ""} Contacto
          %ul.nav.pull-right
            %li.dropdown
              %a{:class => "btn-action dropdown-toggle", "data-toggle" => "dropdown", :href => "#"}
                Iniciar sesión
                %b.caret
              %ul.dropdown-menu
                .espacio
                  %li
                    = form_tag :action => "index", :controller => "home" do
                      = text_field :cliente, :correo, :size => 15, :placeholder => "Correo"
                      = password_field :cliente, :clave, :size => 15, :placeholder => "Clave"
                      = submit_tag "Ingresar»", :class => "btn btn-primary"
                      =link_to "Olvidé mi clave"
                %li.divider
                %li
                  ¿No te has registrado?
                  %br
                  =link_to "Hazlo aquí"
            %li
              =link_to "Registrarse"
```

Figura 18. Código _barra.html.haml. Iteración 1

- **Pruebas:** A continuación las pruebas realizadas en esta iteración

Tabla 12. Caso de prueba: 4. Iteración 1

Número de Caso de Prueba: 4	**Número de Historia de Usuario:** 3
Descripción: Mantener el CSS de la aplicación y del sitio desplegado independientes entre ellos.	
Resultado: 	

3.3.3. Iteración 2

- **Planificación:** El objetivo principal de esta iteración es obtener los archivos que se encuentran en el WARC (ver sección Formatos de archivo del capítulo 2) utilizando la herramienta WARC Tools de Hanzo a través de la ejecución de comandos Shell Script mediante el lenguaje de programación Ruby. Dicha herramienta permite extraer los archivos que conforman el sitio Web solicitado por el usuario, por ejemplo, imágenes, archivos CSS, HTML, Javascript, entre otros, que permitirán en las próximas iteraciones simular el comportamiento del sitio original en un ambiente local.

Tabla 13. Planificación Iteración 2

Número de Iteración: 2	**Número de Historia:** 8
Fecha de Inicio: 24/05/2013	**Fecha de Fin:** 28/05/2013
Descripción: Extraer los archivos del WARC y organizarlos dentro de la aplicación de acuerdo al patrón MVC	**Tipo:** Desarrollo

- **Diseño:** Cuando el usuario introduce el URL del sitio de su preferencia y elige la versión del mismo que desea desplegar, la ruta del archivo WARC que lo contiene es tomada por el controlador. Este, a través de un script de WARC Tools llamado warcunpack_ia.py, extrae los archivos del WARC, ejecutándolo vía Shell Script; dicho script crea carpetas que contienen los archivos que conforman el sitio a desplegar. Posteriormente estos archivos se organizan en las diferentes carpetas de la aplicación siguiendo el patrón MVC para poder llevar a cabo correctamente el despliegue del sitio en cuestión.

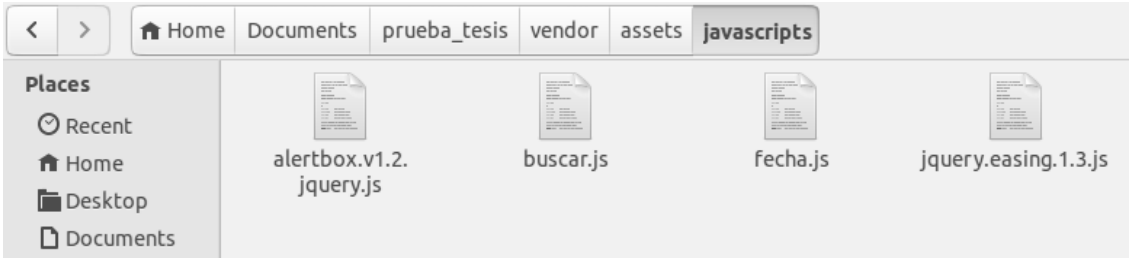
- **Codificación:** Los archivos contenidos en el WARC son extraídos a través del script warcunpack_ia.py, perteneciente a WARC Tools: *<ruta_del_WARC>./warcunpack_ia.py <nombre_WARC> -o <destino>*; estos archivos son analizados utilizando el comando *ls <nombre_carpeta>* para así obtener los nombres y formatos de archivo correspondientes a imágenes, CSS, JS, HTML, entre otros, y así almacenar en un arreglo dicha información. Este arreglo es recorrido para agrupar los nombres, en nuevos arreglos, según su formato (.css, .js, .html, .jpg, entre otros) y a su vez, de acuerdo al tipo de archivo, copiarlos a las respectivas carpetas de Rails que representan a los mismos siguiendo el patrón MVC: *vendor/assets/images* para imágenes, *vendor/assets/stylesheets* para archivos CSS, *vendor/assets/javascripts* para archivos JS y el resto de los archivos son copiados a la carpeta de las vistas correspondientes a un controlador llamado Despliegue; para la copia de los mismos se utilizó el comando de Shell Script *cp <nombre_archivo> <ruta_destino>*.

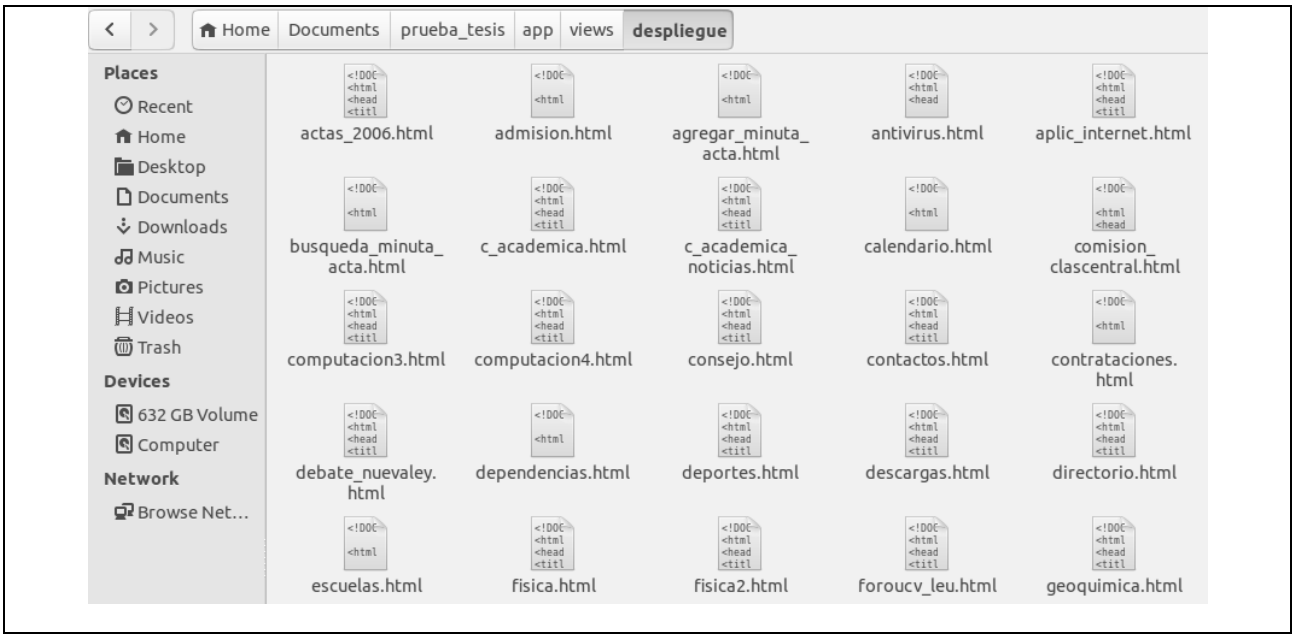
```
42  warc_unpack = `#{raiz}./warcunpack_ia.py #{raiz}#{file} -o #{raiz}`
43  file_name = "#{raiz}#{nombre_sitio}"
44
45  @ls = `ls #{file_name}`
46  @arreglo_ls = @ls.split
47  @arreglo_html = Array.new
48  @arreglo_files = Array.new
49  @arreglo_css = Array.new
50  @arreglo_js = Array.new
51  @arreglo_img = Array.new
52  @arreglo_dir_img = Array.new
53
54  @arreglo_ls.each do |separar|
55    if separar[/.html/] or separar[/.php/]
56      `cp #{file_name}#{separar} vendor/assets/stylesheets/`
57      @arreglo_html << separar
58    else
59      if separar[/.css/]
60        `cp #{file_name}#{separar} vendor/assets/stylesheets/`
61        @arreglo_css << separar
62      else
63        if separar[/.js/]
64          `cp #{file_name}#{separar} vendor/assets/javascripts/`
65          @arreglo_js << separar
66        else
67          if separar[/.png/] or separar[/.jpg/] or separar[/.jpeg/] or separar [/.gif/] or separar[/.jpe/]
68
69            `cp #{file_name}#{separar} vendor/assets/images/`
70            if separar[/.jpe/]
71              extension = separar.split(".")
72              `mv #{vendor}/#{separar} #{vendor}/#{extension[0]}.jpg`
73              separar = "#{extension[0]}.jpg"
74            end
75            @arreglo_img << separar
76            @arreglo_dir_img << "#{file_name}"
77          else
78            @arreglo_files << separar
79          end
```

Figura 19. Código despliegue_controller.rb. Iteración 2

- **Pruebas:** A continuación las pruebas realizadas en esta iteración

Tabla 14. Caso de prueba: 5. Iteración 2

Número de Caso de Prueba: 5	**Número de Historia de Usuario:** 8
Descripción: Extraer los archivos del WARC y copiarlos correctamente en sus respectivas carpetas de la aplicación siguiendo el patrón MVC	
Resultado:   	



3.3.4. Iteración 3

- **Planificación:** En esta iteración el objetivo es modificar los enlaces de las imágenes, de los archivos CSS, JS, HTML contenidos en los diferentes archivos HTML, de modo que la referencia a estos sea válida y el despliegue se haga correctamente.

Tabla 15. Planificación Iteración 3

Número de Iteración: 3	**Número de Historia:** 8
Fecha de Inicio: 1/06/2013	**Fecha de Fin:** 13/06/2013
Descripción: Modificar los enlaces de las imágenes, los CSS, JS, HTML contenidos en los archivos HTML	**Tipo:** Desarrollo

- **Diseño:** Los archivos extraídos del WARC en la iteración anterior, provienen del rastreo hecho al sitio original, por lo tanto los enlaces a archivos propios del sitio son direcciones relativas. Dado esto, los archivos deben estar en las carpetas que le corresponden (según el patrón MVC de Ruby on Rails) para que estos puedan ser encontrados por el navegador e interpretados correctamente por el mismo.

- **Codificación:** Para esta iteración se hace uso de los arreglos creados en la iteración anterior, los cuales se fueron recorriendo para buscar coincidencias de nombres dentro de los archivos HTML o CSS, en los que existen direcciones relativas que fueron modificadas de acuerdo a las

rutas correspondientes según el tipo de archivo, por ejemplo de *carpeta_ejemplo/ejemplo.css* a *vendor/assets/stysheets/ejemplo.css*, siguiendo el patrón MVC de la aplicación.

```
112     @arreglo_css.each do |css|
113       @arreglo_files.each do |dir|
114         line = File.read("vendor/assets/stylesheets/#{css}")
115         reemplazo = line.gsub(/\/#{dir}|\.\/#{dir}|#{dir}|\.\.\/#{dir}/, "../images")
116         File.open("vendor/assets/stylesheets/#{css}", "w") {|file| file.puts reemplazo}
117       end
118     end
119
120     @arreglo_html.each do |sub|
121       @arreglo_files.each do |dir|
122         line = File.read("#{file_name}#{sub}")
123         reemplazo = line.force_encoding('iso-8859-1').gsub(/\/#{dir}|\.\/#{dir}|#{dir}/, "/assets")
124         File.open("#{file_name}#{sub}", "w") {|file| file.puts reemplazo}
125       end
126
127       @arreglo_img.each do |img|
128         nombre = img.split(".")
129         line = File.read("#{file_name}#{sub}")
130         reemplazo = line.gsub(/\/assets\/#{nombre[0]}|#{nombre[0]}/, "/assets/#{nombre[0]}")
131         File.open("#{file_name}#{sub}", "w") {|file| file.puts reemplazo}
132       end
133
134       @arreglo_css.each do |css|
135         line = File.read("#{file_name}#{sub}")
136         reemplazo = line.gsub(/\/assets\/#{css}|#{css}/, "/assets/#{css}")
137         File.open("#{file_name}#{sub}", "w") {|file| file.puts reemplazo}
138       end
```

Figura 20. Código despliegue_controller.rb. Iteración 3

- **Pruebas:** A continuación las pruebas realizadas en esta iteración

Tabla 16. Caso de prueba: 6. Iteración 3

Número de Caso de Prueba: 6	**Número de Historia de Usuario:** 8
Descripción: Modificar los enlaces relativos a los CSS	

Resultado:
 	
Enlace original:
`1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">`
`2 <html>`
`3 <head>`
`4 <title>::Facultad de Ciencias::</title>`
`5 <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">`
`6 <!--`
`7 <script type="text/javascript" src="http://ajax./assets/googleapis.com/ajax/libs/jquery/1.3.1/jquery.min.js"></script>`
`8 -->`
`9`
`10 <script type="text/javascript" src="/assets/jquery.easing.1.3.js"></script>`
`11 <script type="text/javascript" src="/assets/alertbox.v1.2.jquery.js"></script>`
`12 <link rel="stylesheet" type="text/css" media="all" href="CSS/alertbox.css"/>`
`13`
`14 <script language="JavaScript" type="text/JavaScript">`	

Enlace modificado:

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <html>
3      <head>
4          <title>::Facultad de Ciencias::</title>
5          <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
6  <!--
7          <script type="text/javascript" src="http://ajax./assets/googleapis.com/ajax/libs/jquery/1.3.1/jquery.min.js"></script>
8  -->
9
10         <script type="text/javascript" src="/assets/jquery.easing.1.3.js"></script>
11         <script type="text/javascript" src="/assets/alertbox.v1.2.jquery.js"></script>
12         <link rel="stylesheet" type="text/css" media="all" href="/assets/alertbox.css"/>
13
14         <script language="JavaScript" type="text/JavaScript">
```

3.3.5. Iteración 4

- **Planificación:** Esta iteración plantea como objetivo la creación de los módulos de la aplicación referentes a los usuarios que quieran registrarse o que ya lo hayan hecho. En esta iteración se tratarán la autenticación, registro, cambio de datos y recuperación de clave por parte de los usuarios.

Tabla 17. Planificación Iteración 4

Número de Iteración: 4	**Número de Historia:** 3, 4, 5
Fecha de Inicio: 17/06/2013	**Fecha de Fin:** 30/06/2013
Descripción: Crear los módulos de autenticación, registro y modificación de datos de usuario, además de la recuperación de su clave en caso de haberla olvidado	**Tipo:** Desarrollo

- **Diseño:** para la fase de diseño se realizan una serie de diagramas de secuencia que constituyen los procesos de autenticación, registro, cambio de datos y recuperación de clave. Estos representan los pasos a seguir por cada operación y la interacción entre el modelo, las vistas y los controladores implicados en cada caso. Dichos diagramas son presentados a continuación:

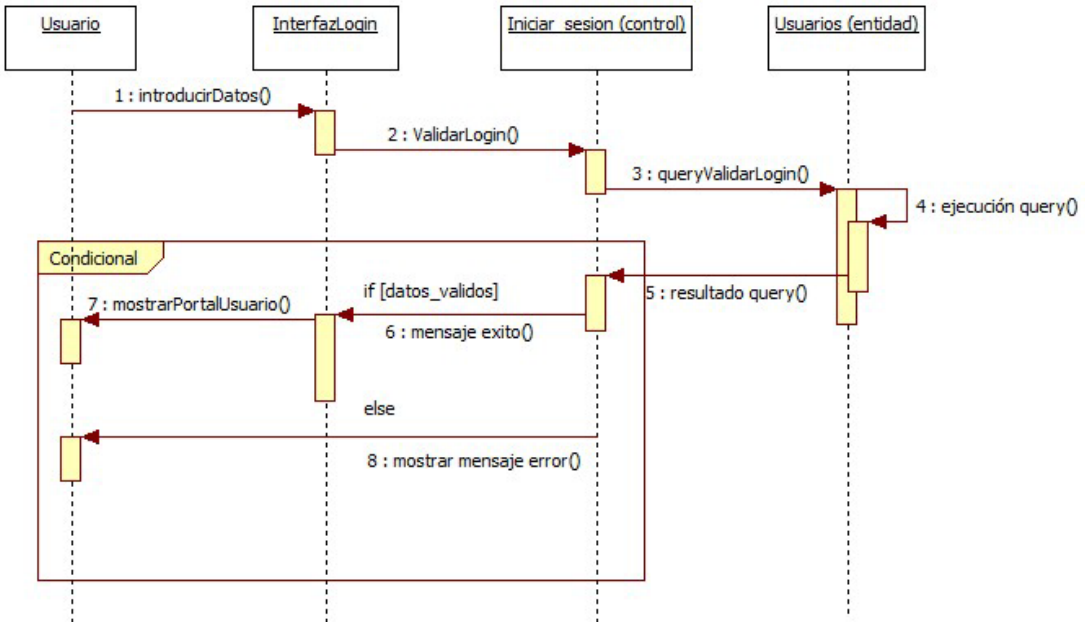


Figura 21. Diagrama de secuencia Autenticación. Iteración 4

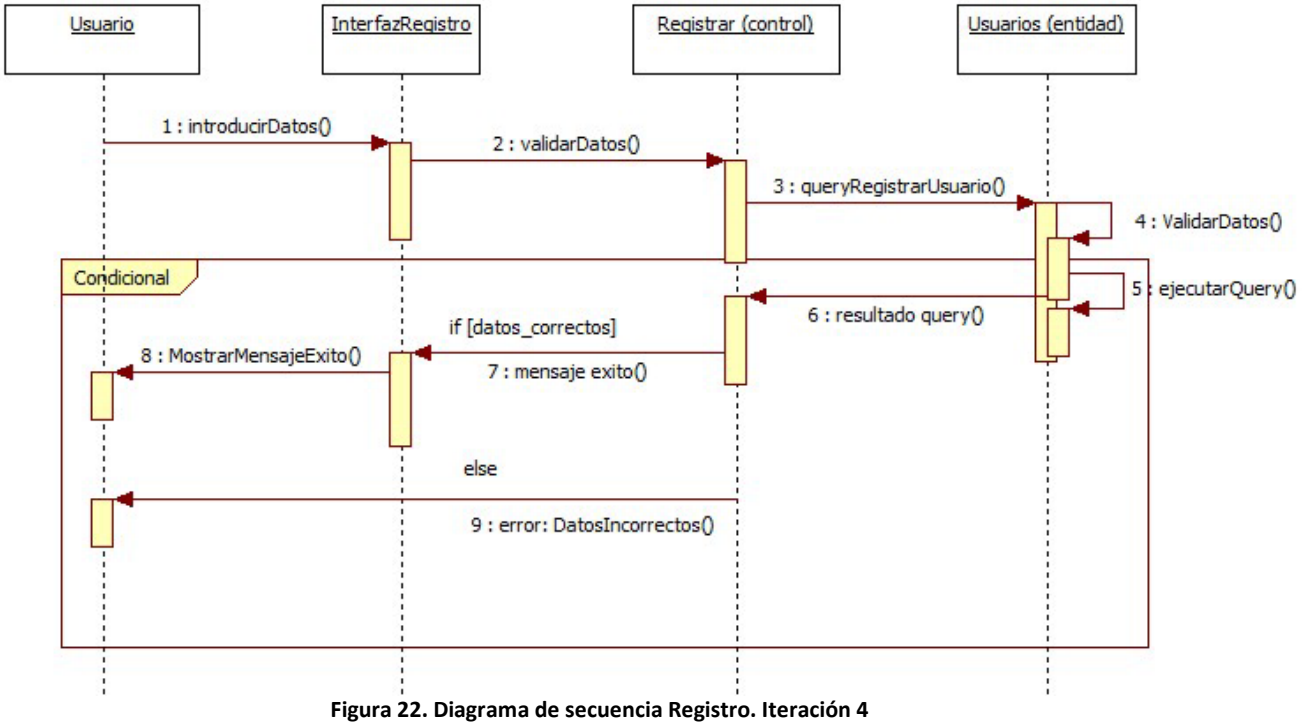


Figura 22. Diagrama de secuencia Registro. Iteración 4

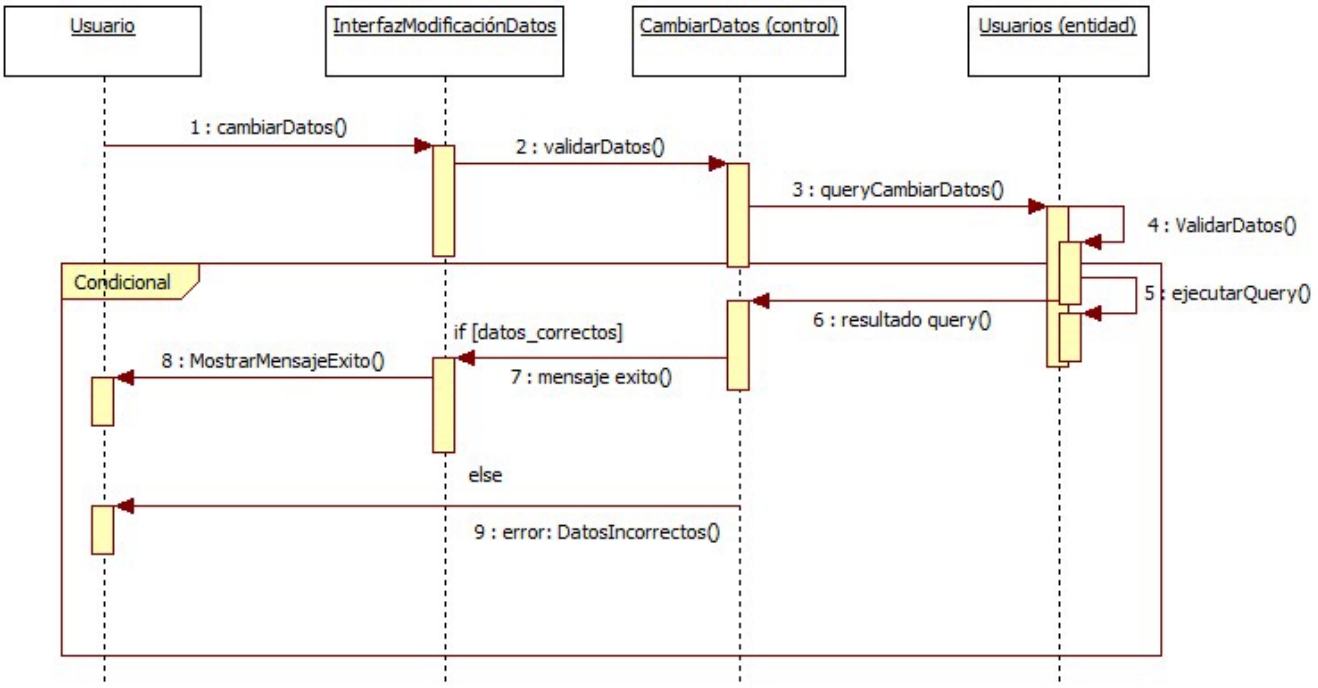


Figura 23. Diagrama de secuencia. Modificación de datos. Iteración 4

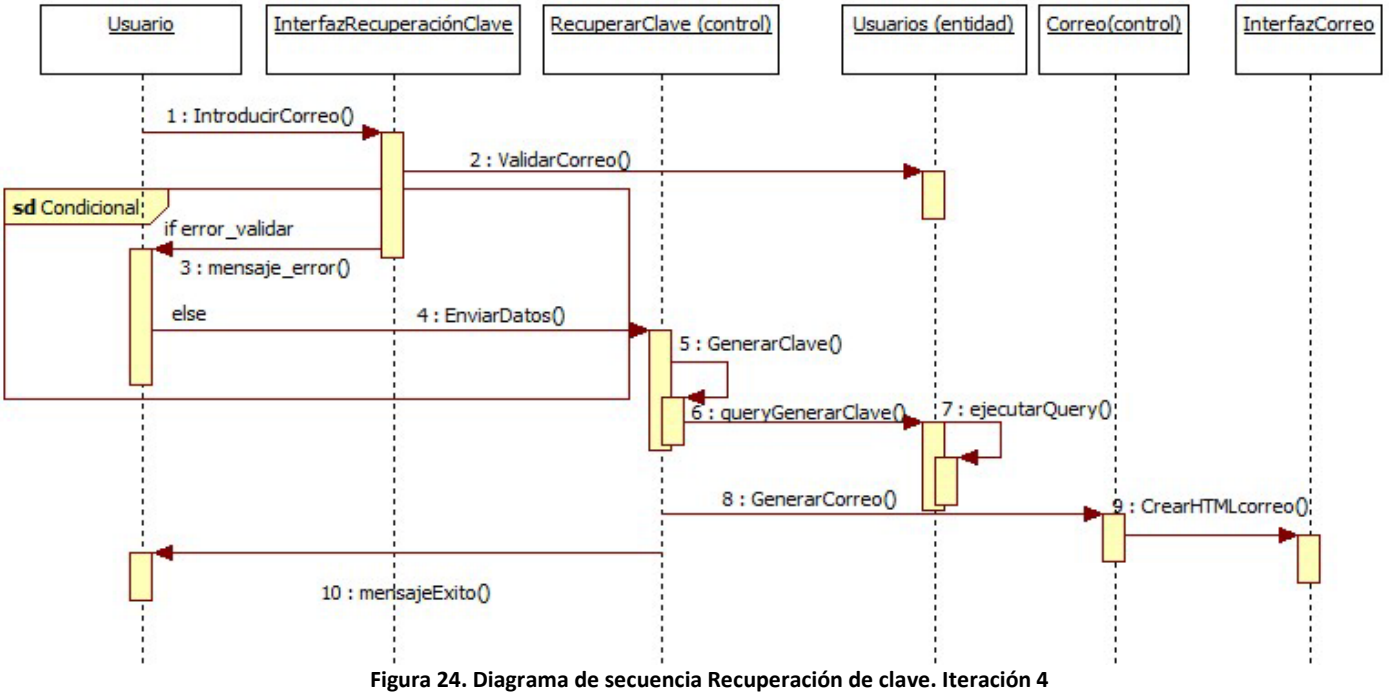


Figura 24. Diagrama de secuencia Recuperación de clave. Iteración 4

- **Codificación:** Dado que la aplicación se está desarrollando con Ruby on Rails, el cual utiliza el patrón MVC, los procesos desarrollados en esta iteración se encuentran manejados a través de un mismo controlador: **usuario_controller.rb** el cual interactúa con los modelos **Usuario.rb**, **Pais.rb** y **Profesión.rb**. A continuación se muestra el modelo Usuario.rb el cual permite realizar las validaciones para los usuarios tanto del lado del cliente como del lado del servidor:

```ruby
# -*- encoding : utf-8 -*-
class Usuario < ActiveRecord::Base

  attr_accessible :correo, :nombre, :apellido, :profesion,
                  :fecha_nacimiento, :pais, :clave
  attr_accessor :clave_confirmation, :clave_actual

  self.primary_key = "idusuario"

  validates :nombre, :presence => true
  validates :apellido, :presence => true
  validates :pais, :presence => true
  validates :profesion, :presence => true
  validates :correo, :presence => true,
                     :format => { :with => /^([^@\s]+)@((?:[-a-z0-9]+\.)+[a-z]{2,})$/i },
                     :uniqueness => true
  validates :clave, :presence => true, :length => {:minimum => 8}
  validates_confirmation_of :clave

  def nombre_completo
  "#{nombre} #{apellido}"
  end

  def self.autenticar(correo, clave)
      clave_digest = Digest::MD5.hexdigest(clave)
      Usuario.where(:correo => correo, :clave => clave_digest).first
  end
end
```

Figura 25. Código usuario. Iteración 4

En este modelo, en primer lugar tenemos los atributos attr_accesible y attr_accessor, los cuales permiten asignar datos masivamente a un registro. Los primeros, son los campos que corresponden con la base de datos; mientras que attr_accessor corresponde a variables que aunque no están en la base de datos se requieren para llevar a cabo algunas validaciones, como por ejemplo la confirmación de clave con validates_confirmation_of :clave, la cual compara contra el attr_accessor clave_confirmation. Por otra parte se tienen en este modelo las validaciones correspondientes a los campos de la tabla Usuario de la base de datos, por ejemplo validaciones de presencia, de unicidad, de longitud y/o de formato con una expresión regular que valida el formato correcto de un correo.

En todos los formularios utilizados en los diferentes procesos de esta iteración, se hacen validaciones del lado del servidor y del lado del cliente. Las validaciones del lado del servidor se hacen en el controlador usuario_controller.rb utilizando las validaciones expuestas anteriormente en el modelo Usuario.rb y se podrán apreciar en los extractos de código mostrados a continuación. En cambio, las validaciones del lado del cliente se llevan a cabo a través de la gema **client_side_validations** la cual utiliza javascript apoyándose en las mismas validaciones del modelo.

Autenticación:

Al solicitar al usuario su correo y su clave, estos son enviados a la función validar de usuario_controller.rb, el cual recibe estos parámetros y los compara con la base de datos a través de la función autenticar del modelo Usuario.rb. Si los datos coinciden con algún registro de la base de datos, se guarda un token de sesión a través de la variable session[:usuario] la cual se mantiene hasta que el usuario cierre sesión o cierre el navegador. Si los datos no coinciden, se le indica al usuario y se le redirecciona a la página de inicio.

```
11    def validar
12      reset_session
13      correo = params[:usuario][:correo]
14      clave = params[:usuario][:clave]
15      if usr = Usuario.autenticar(correo, clave)
16        session[:usuario] = usr
17        if session[:usuario].recupera == 0
18          redirect_to :action => "index", :controller => "home"
19          return
20        else
21          redirect_to :action => "cambiar_clave", :controller => "usuario"
22          return
23        end
24      end
25      flash[:mensaje] = "Datos Incorrectos"
26      redirect_to :action => "index", :controller => "home"
27      return
28    end
```

Figura 26. Código usuario_controller Autenticación. Iteración 4

Registro:

Para el registro se le solicita al usuario una serie de datos, los cuales son ingresados a través de un formulario del tipo form_for que pasa estos como parámetros a la función registro_procesar del usuario_controller.rb. En esta función se crea un registro en la base de datos con @usuario=Usuario.create, se toman estos parámetros, se validan del lado del servidor y se asocian a cada uno de los campos de @usuario, para almacenarlos posteriormente con @usuario.save.

Cabe destacar, que la clave ingresada por el usuario es transformada a MD5 antes de ser almacenada en la base de datos, buscando hacer la aplicación más segura.

```
37    def registro
38      @usuario = Usuario.new
39      @paises = Pais.find(:all)
40      @profesiones = Profesion.find(:all)
41    end
42
43    def registro_procesar
44      @paises = Pais.find(:all)
45      @profesiones = Profesion.find(:all)
46
47      if Usuario.where(:correo => params[:usuario][:correo]).first
48            flash[:mensaje] = "Este correo ya se encuentra registrado"
49            redirect_to :back
50            return
51      else
52          @usuario = Usuario.create
53          @usuario.nombre = params[:usuario][:nombre]
54          @usuario.apellido = params[:usuario][:apellido]
55          @usuario.fecha_nacimiento = params[:usuario][:fecha_nacimiento]
56          @usuario.pais = params[:usuario][:pais]
57          @usuario.profesion = params[:usuario][:profesion]
58          @usuario.clave = Digest::MD5.hexdigest(params[:usuario][:clave])
59          @usuario.correo = params[:usuario][:correo]
60          if @usuario.save
61            flash[:mensaje] = "Se ha registrado con éxito"
62            redirect_to :action => "index", :controller => "home"
63            return
64          else
65            render :action => "registro"
66          end
67        end
68    end
```

Figura 27. Código usuario_controller Registro. Iteración 4

Modificación de datos (datos del usuario y clave):

Para la modificación de datos del usuario o de la clave, el usuario debe estar autenticado. Ante esto, cuando el usuario desea realizar la modificación de estos y activa el evento correspondiente, se busca el usuario en cuestión a través de la variable de sesión y en la vista se le presentan los datos asociados a él en la base de datos. Una vez que este los modifica se llama a la función cambiar_datos_procesar o cambiar_clave_procesar de controlador usuario_controller.rb, donde se realiza un proceso similar al de registro, pero en lugar de crear un usuario nuevo, se modifica el que se buscó con la variable de sesión y al final se ejecutó @usuario.save para guardar los cambios. El formulario de modificación de datos, al igual que los otros, también se encuentra validado por las restricciones presentes en el modelo.

```ruby
  def cambiar_datos
    @usuario = Usuario.where(:idusuario => session[:usuario].idusuario).first
    @paises = Pais.find(:all)
    @profesiones = Profesion.find(:all)
  end

  def cambiar_datos_procesar
    @paises = Pais.find(:all)
    @profesiones = Profesion.find(:all)

    @usuario = Usuario.where(:idusuario => session[:usuario].idusuario).first
    @usuario.nombre = params[:usuario][:nombre]
    @usuario.apellido = params[:usuario][:apellido]
    @usuario.fecha_nacimiento = params[:usuario][:fecha_nacimiento]
    @usuario.pais = params[:usuario][:pais]
    @usuario.profesion = params[:usuario][:profesion]

    if @usuario.save
      session[:usuario] = @usuario
      flash[:mensaje] = "Se han modificado sus datos con éxito"
      redirect_to :action => "index", :controller => "home"
      return
    else
      render :action => "cambiar_datos"
    end
  end

  def cambiar_clave
        @usuario = Usuario.where(:idusuario => session[:usuario].idusuario).first
  end

  def cambiar_clave_procesar
    clave_act = params[:usuario][:clave_actual] #<%= label :usuario, :cedula, "Cedula:" %>
    clave = params[:usuario][:clave]
    confirm_new = params[:usuario][:clave_confirmation]

    @usuario = Usuario.where(:idusuario => session[:usuario].idusuario).first

    if @usuario.clave != Digest::MD5.hexdigest(clave_act)
      flash[:mensaje] = "Clave actual no coincide1"
      redirect_to :action => "cambiar_clave"
      return
    end

    if clave_act == confirm_new
      flash[:mensaje] = "La clave actual y la nueva claven deben ser diferentes"
      redirect_to :action => "cambiar_clave"
      return
    end

    @usuario.recupera = 0;
    @usuario.clave = Digest::MD5.hexdigest(clave)

    if @usuario.save
      session[:usuario] = @usuario #actualizar sesion porque el actual aun mantiene su clav
      flash[:mensaje] = "Cambio de clave exitosa"
      redirect_to :action => "index", :controller => "home"
      return
    else
      flash[:mensaje] = "No se pudo realizar el cambio de clave"
      redirect_to :action => "cambiar_clave"
      return
    end
  end
end
```

Figura 28. Código usuario_controller Modificación de datos. Iteración 4

Recuperación de clave:

Para recuperar la clave de un usuario, este debe ingresar su dirección de correo electrónico con el que se registró. En caso de no existir en la base de datos, se le indica a través de un mensaje de error. En el caso contrario, se genera una clave con la función SecureRandom provista por Ruby. Esta clave temporal es enviada al correo electrónico del usuario, quién debe autenticarse con ella, y podrá así cambiar su clave.

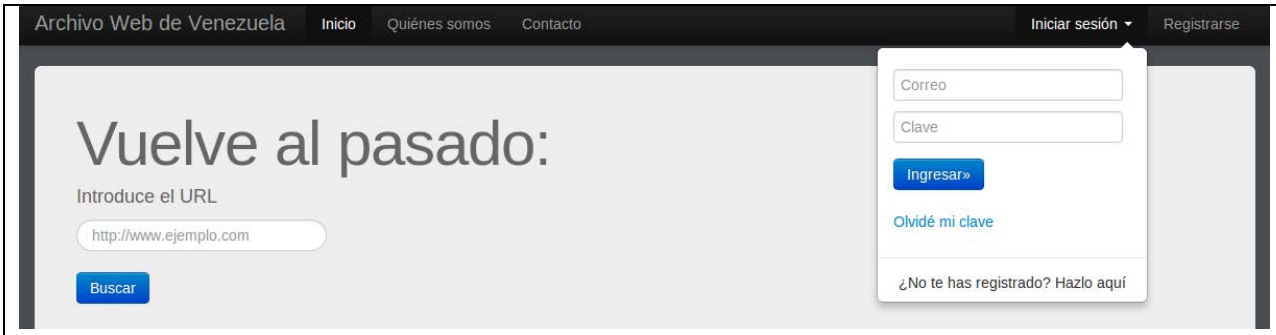
```
74    def recordar_contrasena_inicio
75      unless params[:usuario] &&
76        params[:usuario][:correo]
77        flash[:mensaje] = "El campo correo está vacío"
78        redirect_to :action => "olvide_clave"
79        return
80      end
81
82      correo = params[:usuario][:correo]
83      @usuario = Usuario.where(:correo => correo).first
84
85      unless @usuario
86        flash[:mensaje] = "Usted no se encuentra registrado"
87        redirect_to :action => "olvide_clave"
88        return
89      end
90      clave_nueva = SecureRandom.hex(10)
91      @usuario.recupera = 1;
92      @usuario.clave = Digest::MD5.hexdigest(clave_nueva)
93      @usuario.save
94      CorreosUsuario.correo_olvide_mi_clave(@usuario, clave_nueva).deliver
95      flash[:mensaje] = "Se ha enviado un mensaje a su correo electrónico con una clave
96                         temporal para que pueda acceder a su cuenta y cambiar de clave"
97      redirect_to :action => "index", :controller => "home"
98      return
99    end
```

Figura 29. Código Recuperación de clave. Iteración 4

- **Pruebas:**

Tabla 18. Caso de prueba: 7. Iteración 4

Número de Caso de Prueba: 7	**Número de Historia de Usuario:** 5
Descripción: Autenticarse con éxito en la aplicación	
Resultado: Antes de la autenticación	



Después de la autenticación

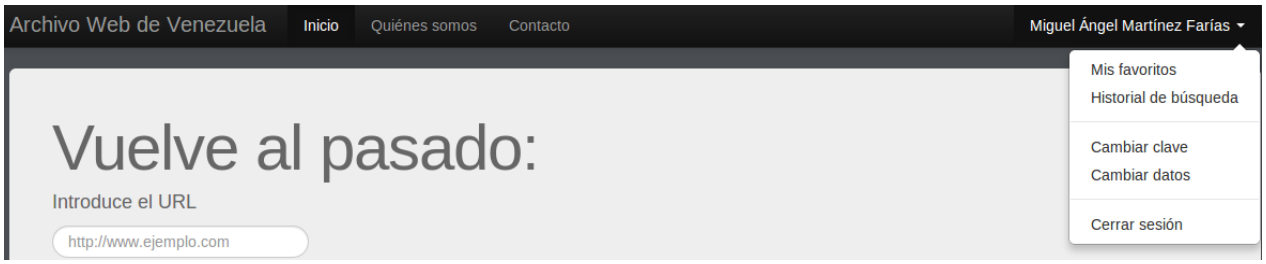
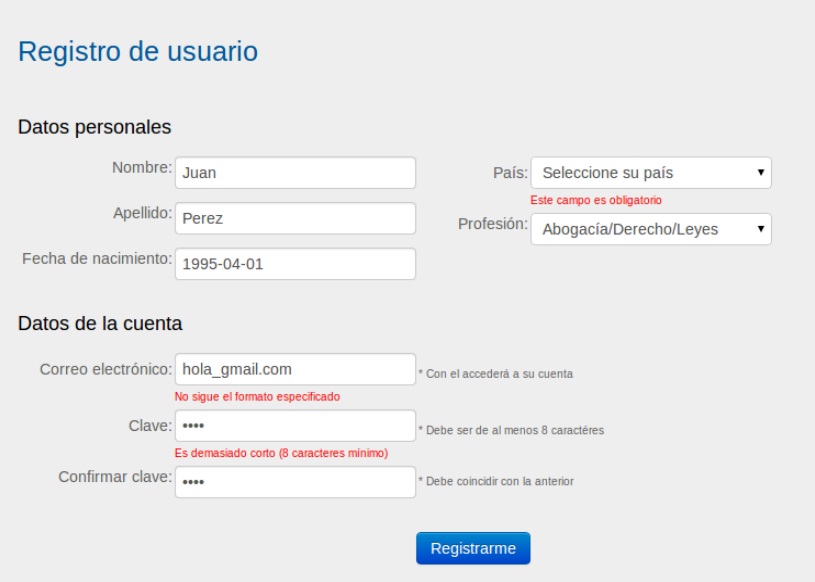


Tabla 19. Caso de prueba: 8. Iteración 4

Número de Caso de Prueba: 8	**Número de Historia de Usuario:** 4
Descripción: Verificar las validaciones en el proceso de registro	
Resultado:	

Tabla 20. Caso de prueba: 9. Iteración 4

Número de Caso de Prueba: 9	**Número de Historia de Usuario:** 4
Descripción: Verificar las validaciones en el proceso de registro, cambio de clave y cambio de datos	
Resultado:	



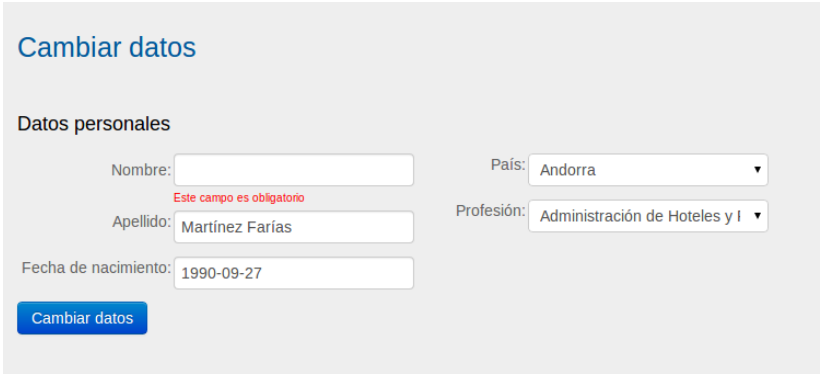
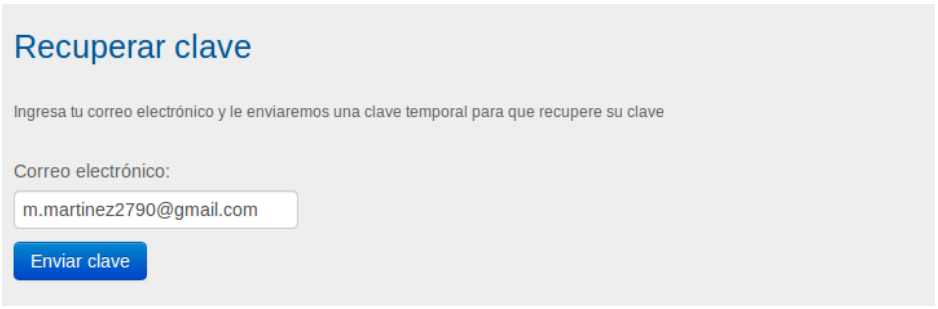


Tabla 21. Caso de prueba: 10. Iteración 4

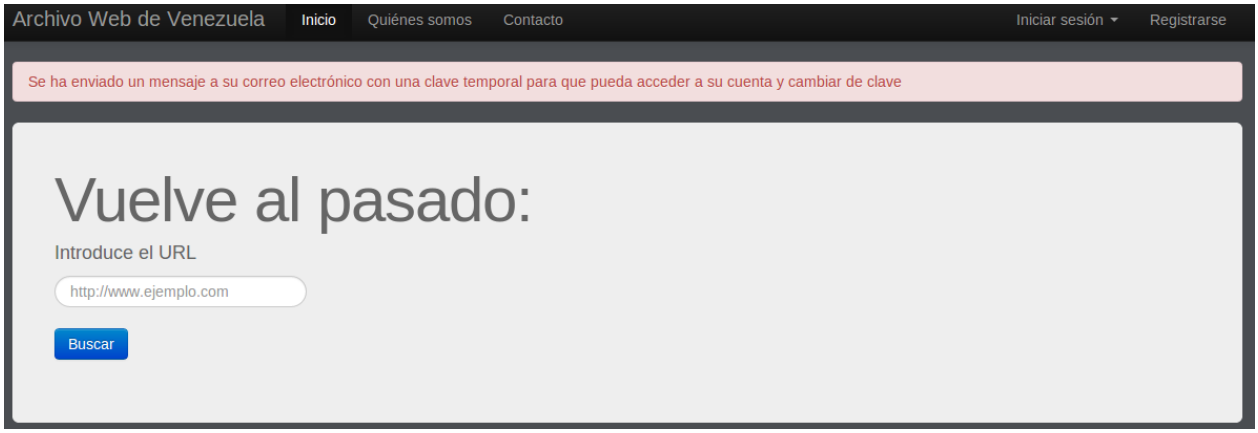
Número de Caso de Prueba: 10	**Número de Historia de Usuario:** 4
Descripción: Recuperar clave de forma exitosa	

Resultado:

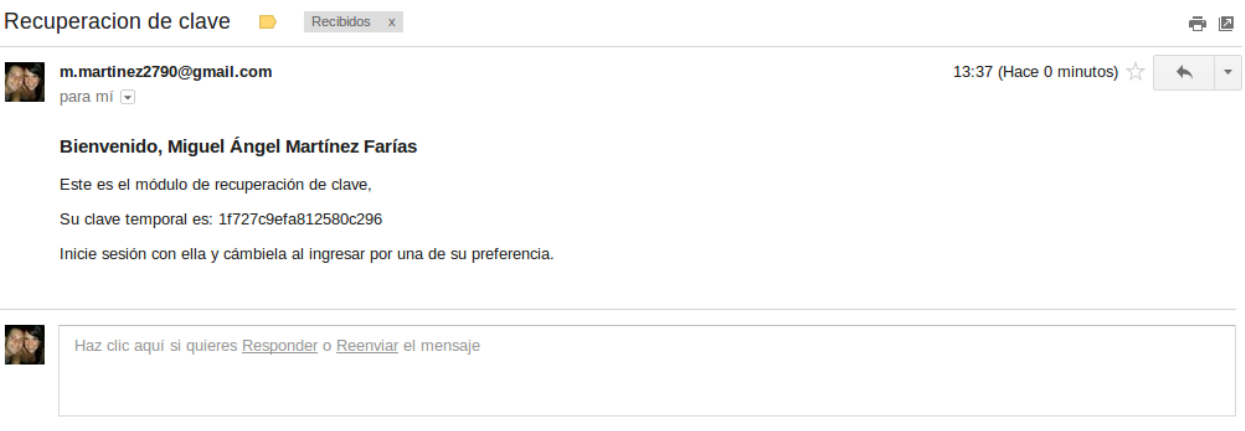
Ingresar correo



Mensaje de notificación



Correo con clave temporal



3.3.6. Iteración 5

- **Planificación:** En esta iteración el objetivo es desarrollar una actualización de la interfaz gráfica de la aplicación que se llevó a cabo en la Iteración 1 en busca de mejorar la experiencia del usuario al usar la aplicación.

Tabla 22. Planificación Iteración 5

Número de Iteración: 5	**Número de Historia:** 1 y 2
Fecha de Inicio: 2/07/2013	**Fecha de Fin:** 4/07/2013
Descripción: Actualización de la interfaz gráfica	**Tipo:** Desarrollo / Diseño

- **Diseño:** Para la actualización de la interfaz gráfica, se elabora un logotipo para el Prototipo de Archivo Web de Venezuela con el uso de las herramientas gráficas Adobe Photoshop CS6 y Adobe Illustrator CS6. De este logotipo se realizan dos versiones que se usan de acuerdo a la distribución del espacio donde son colocadas.

Figura 30. Logotipo de la aplicación

Para esta actualización de la interfaz gráfica, se sigue usando Bootstrap y se cambian los colores grises a tonos más claros y blanco en el fondo de la aplicación. Además de esto, se realizan cambios de dimensión en algunas de las secciones de la interfaz, y se agrega un pie de página que contiene el logo del prototipo, los logos correspondientes a la Universidad Central de Venezuela, a la Facultad de Ciencias y a la Escuela de Computación, así como la información de contacto de la UCV.

- **Codificación:** Siguiendo el patrón MVC, se modifican los layouts application, externo, otros y barra a través de los archivos CSS, cambiando en ellos lo necesario para lograr el estilo deseado.

```
46  body {
47      color: #666;
48      background-color: #ffffff;
49  }
50  .footer{
51      /*background-color: black;*/
52      text-align: center;
53      color: black;
54  }
55
56  .templatemo_footer {
57      position: relative;
58      font-size: 9px;
59      margin-top: -90;
60      height: 90px;
61      clear: both;
62      margin: 0;
63      padding: 0;
64      padding-top: 10px;
65      background-color: #e2e2e2; /*cambiar por f2f2f2*/
66      line-height: 140%;
67        margin: 5px 5px 5px 5px;
68    border: 0;
69    border-top: 1px solid #DBD7D7;
70    border-bottom: 1px solid #FFFDFD;
71  }
72
73  .templatemo_footer a {
74      font-weight: normal;
75      color: #000;
76  }
77
78  .templatemo_footer .h5 {
79      font-weight: normal;
80      color: #666;
81      font-weight: bold;
82      padding-bottom: 5px;
83  }
84
85  .espacio{
86      padding: 15px;
87  }
88
89  .subr{
90      color: #005a9c;
91      }
```

Figura 31. Código templatemo_style CSS. Iteración 5

```
4411  .hero-unit {
4412    padding: 40px;
4413    padding-top: 18px;
4414    margin-bottom: 10px;
4415    font-size: 18px;
4416    font-weight: 200;
4417    line-height: 30px;
4418    color: inherit;
4419    background-color: #F7F8F3;
4420    -webkit-border-radius: 6px;
4421    -moz-border-radius: 6px;
4422    border-radius: 6px;
4423  }
4424  .hero-unit2 {
4425    padding-top: 18px;
4426    padding-left: 20px;
4427    padding-bottom: 0px;
4428
4429    margin-bottom: 10px;
4430    font-size: 12px;
4431    font-weight: 200;
4432    color: inherit;
4433    background-color: #F7F8F3;
4434    -webkit-border-radius: 6px;
4435    -moz-border-radius: 6px;
4436    border-radius: 6px;
4437  }
4438  .hero-unit-peq {
4439    padding-top: 18px;
4440    padding-left: 20px;
4441    margin-bottom: 10px;
4442    padding-bottom: 20px;
4443    font-size: 12px;
4444    font-weight: 200;
4445    color: inherit;
4446    background-color: #F7F8F3;
4447    -webkit-border-radius: 6px;
4448    -moz-border-radius: 6px;
4449    border-radius: 6px;
4450  }
4451  .p2 {
4452    text-align: right;
4453    font-size: 12px;
```

Figura 32. Código bootstrap CSS

El resultado de aplicar estos cambios e incluir el logo y el pie de página fue el siguiente:

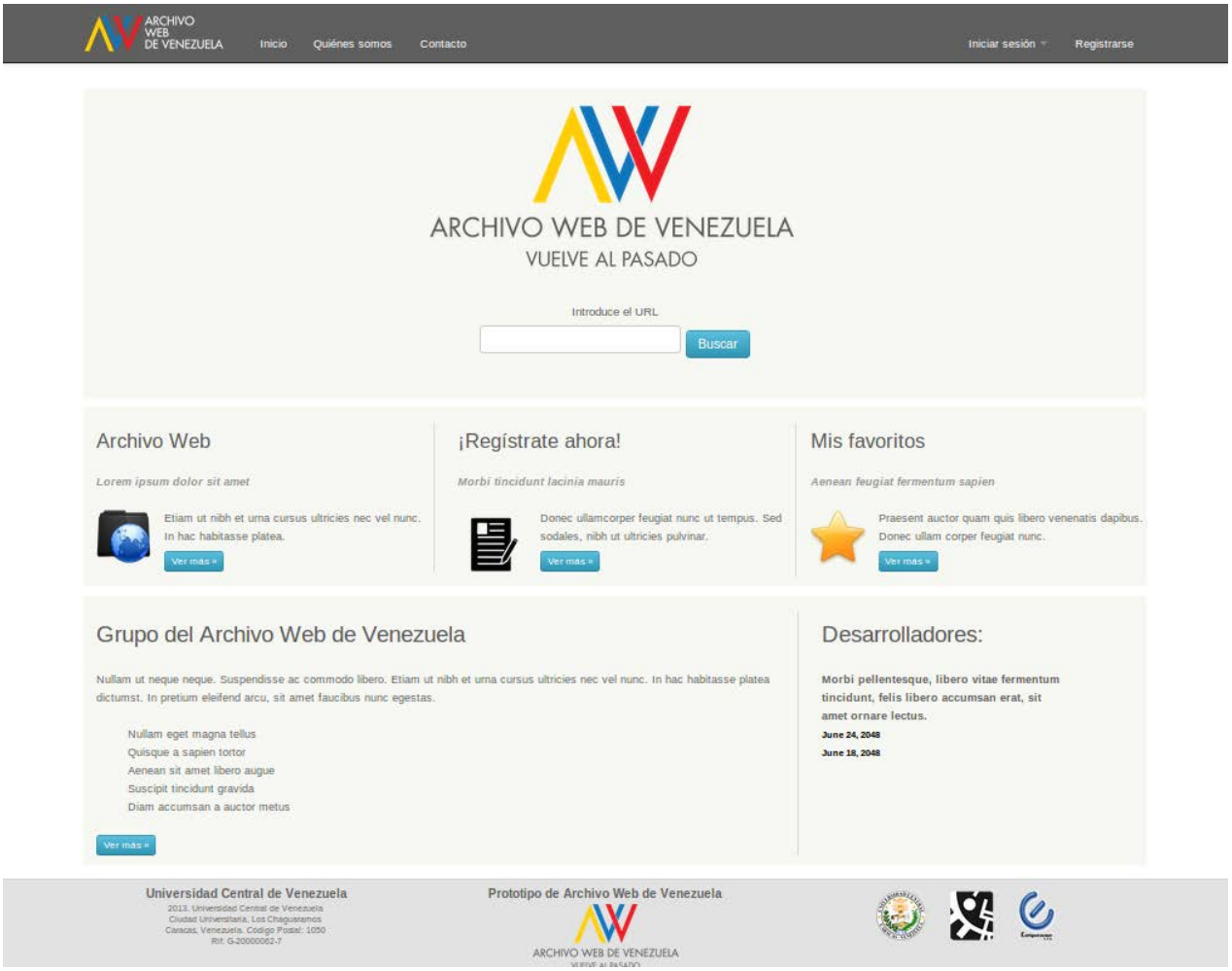


Figura 33. Interfaz inicial de la aplicación

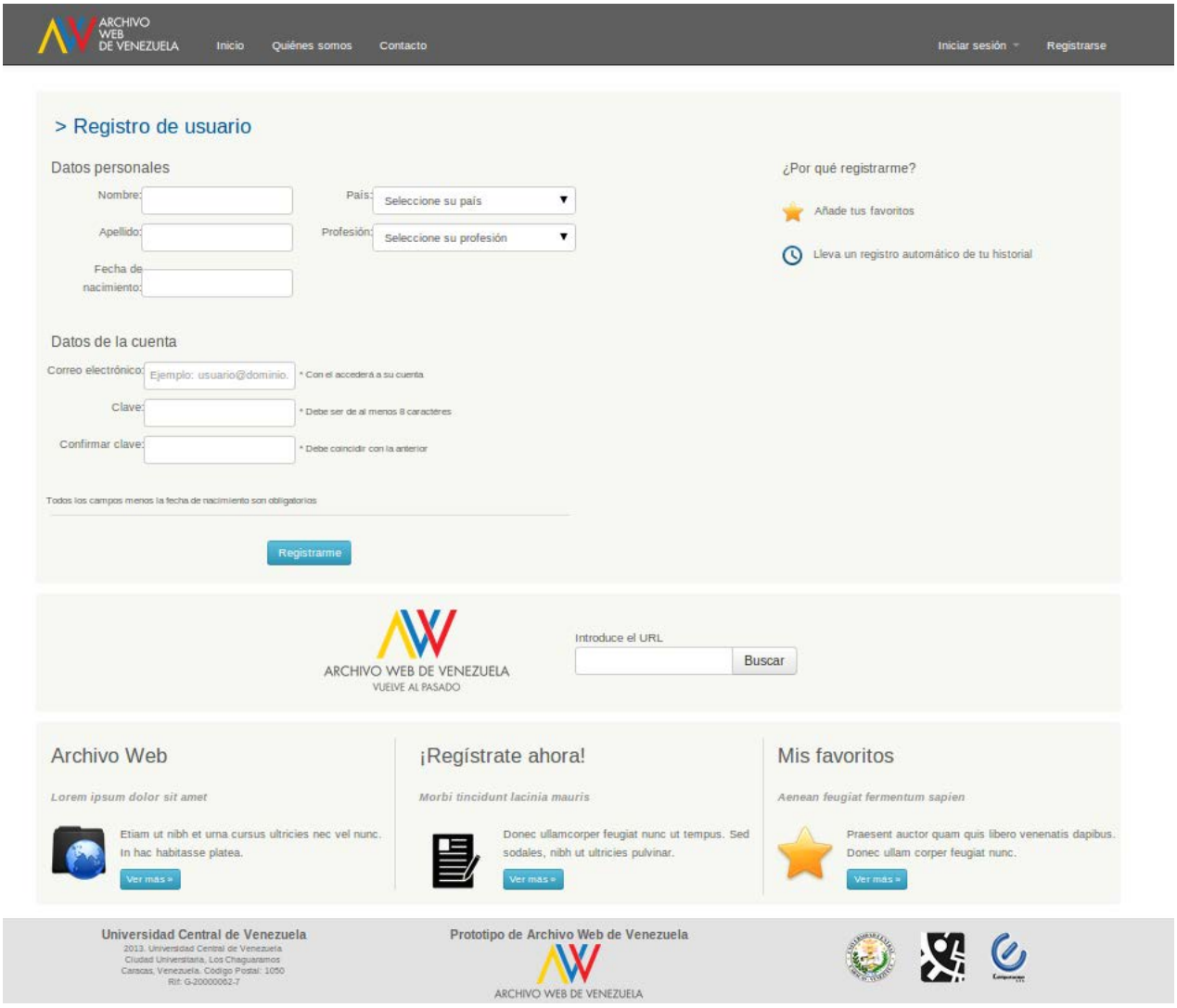


Figura 34. Interfaz secundaria de la aplicación

- **Pruebas:** En esta iteración no se requirieron pruebas.

3.3.7. Iteración 6

- **Planificación:** Esta iteración plantea como objetivo la creación de los módulos de Favoritos e Historial para brindarle al usuario la posibilidad de almacenar en su cuenta un acceso directo a las búsquedas que sean de su interés y llevar automáticamente un histórico de las búsquedas que ha realizado. Por tanto en esta iteración se tratará la administración de Favoritos y del Historial por parte de los usuarios.

Tabla 23. Planificación Iteración 6

Número de Iteración: 6	**Número de Historia:** 9, 10
Fecha de Inicio: 16/07/2013	**Fecha de Fin:** 26/07/2013
Descripción: Crear los módulos de Favoritos e Historial	**Tipo:** Desarrollo

- **Diseño:** Para la fase de diseño se realizan una serie de diagramas de secuencia que constituyen los procesos de gestión de Favoritos e Historial. Estos representan los pasos a seguir por cada operación y la interacción entre el modelo, las vistas y los controladores implicados en cada caso. Dichos diagramas son presentados a continuación:

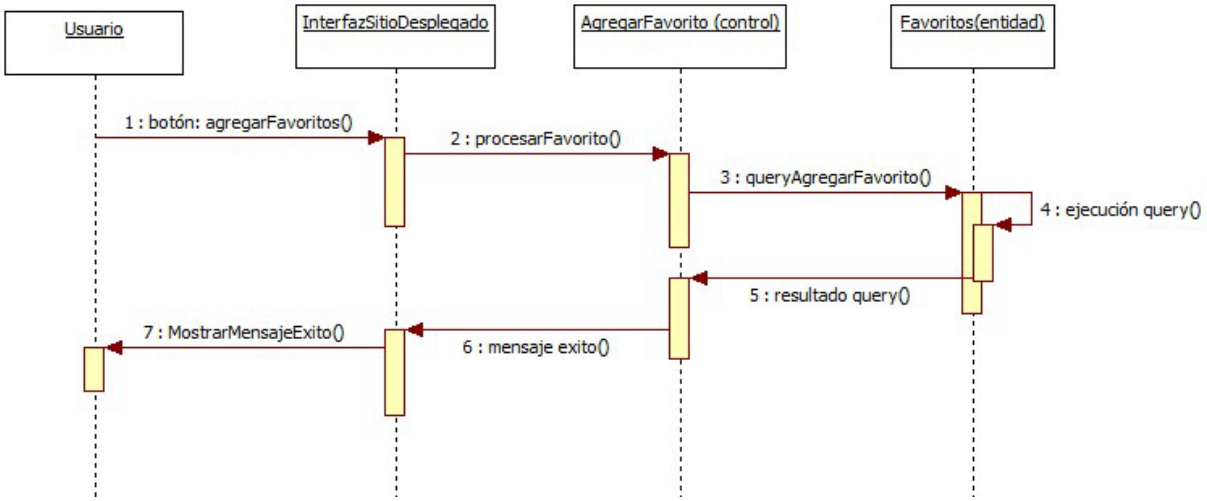


Figura 35. Diagrama de secuencia Agregar Favorito. Iteración 6

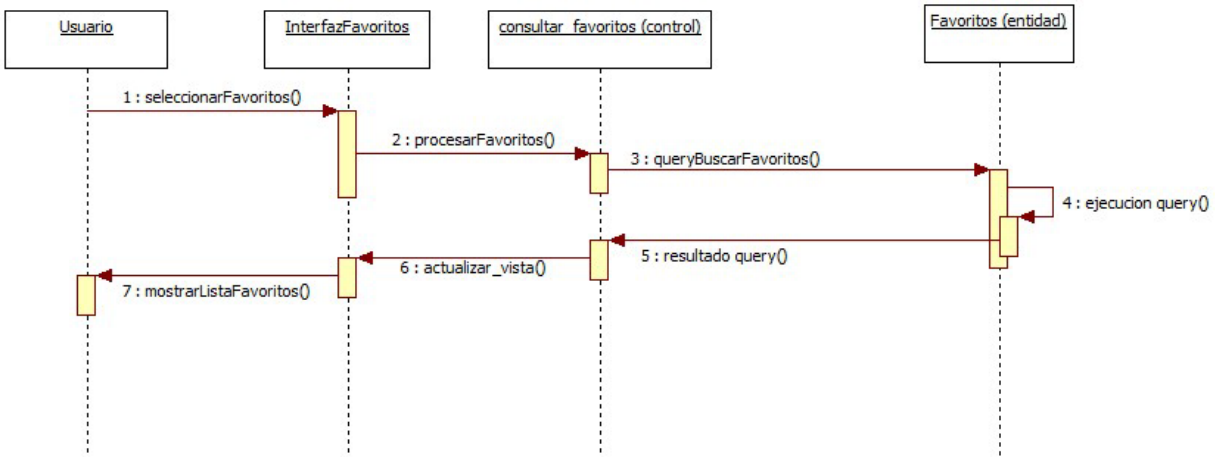


Figura 36. Diagrama de secuencia Consultar Favorito. Iteración 6

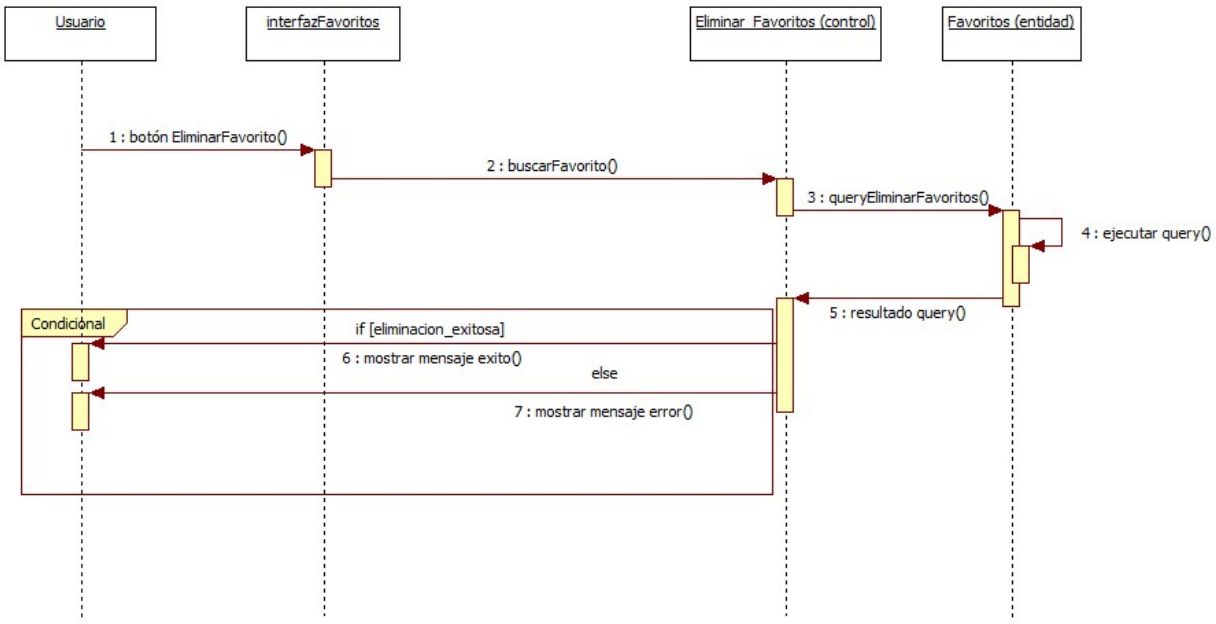


Figura 37. Diagrama de secuencia Eliminar Favorito. Iteración 6

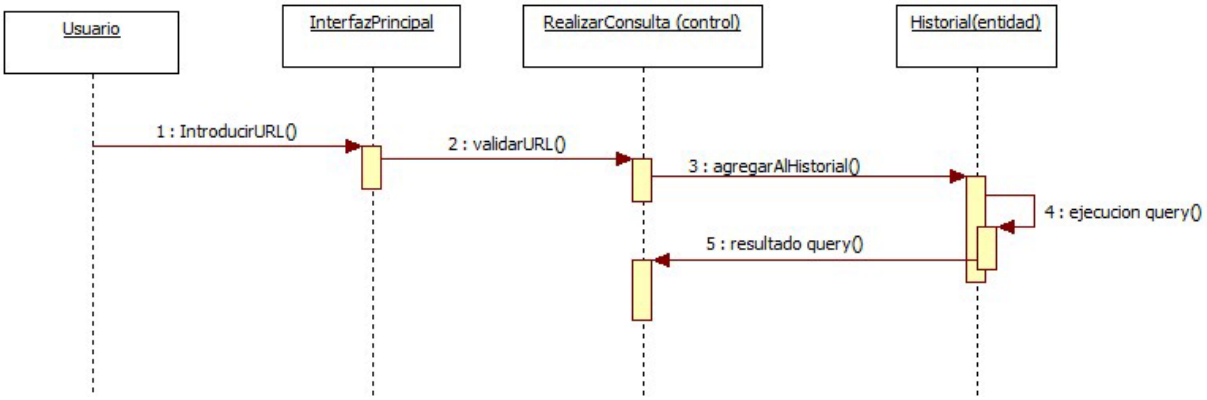


Figura 38. Diagrama de secuencia Agregar historial. Iteración 6

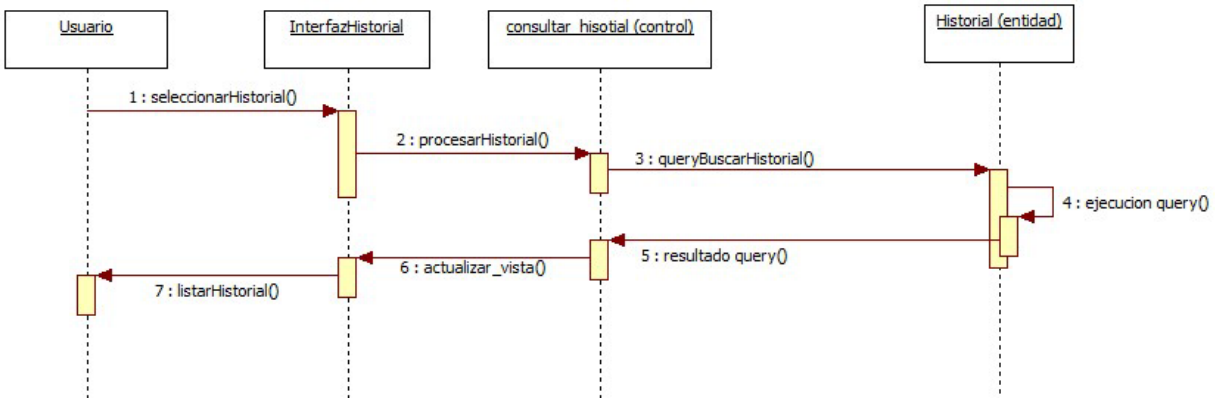


Figura 39. Diagrama de secuencia Consultar Historial. Iteración 6

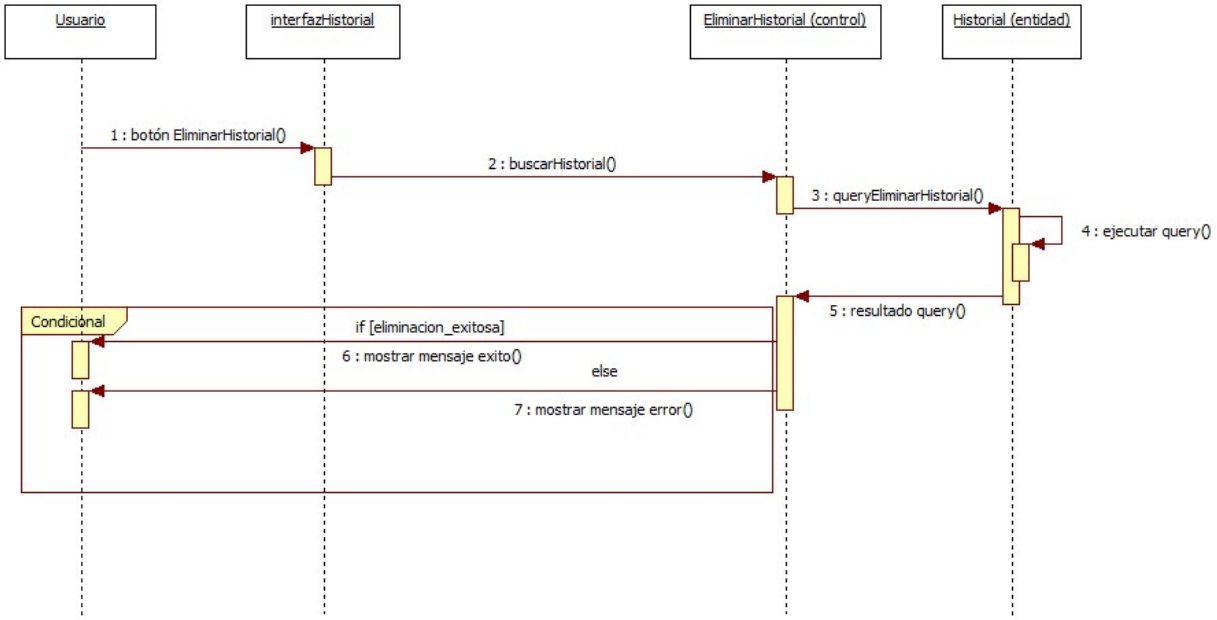


Figura 40. Diagrama de secuencia Eliminar historial. Iteración 6

- **Codificación:** Siguiendo el patrón MVC, los módulos desarrollados en esta iteración se encuentran manejados a través del controlador que se encarga de la lógica de negocio de los procesos vinculados al usuario: **usuario_controller.rb** el cual para el caso de estudio interactúa con los modelos **Usuario.rb**, **Favorito.rb** e **Historial.rb**.

Módulo de Favoritos:

En **usuario_controller.rb** se encuentran definidos una serie de métodos que permiten la gestión de los favoritos: **listar_favoritos**, **agregar_favorito**, **eliminar_favorito**, **eliminar_favorito_todo**.

```
193   def listar_favoritos
194     @usuario_favoritos = Favorito.paginate(:page => params[:page], :conditions => {:idusuario =>
195                                            session[:usuario].idusuario})
196   end
197
198   def agregar_favorito
199     warc = Warc.where(:idwarc => params[:idWarc]).first
200     f = Favorito.new
201     f.url_favorito = warc.nombreWarc
202     f.idusuario = session[:usuario].idusuario
203     f.url_version = warc.version
204     f.idwarc = warc.idwarc
205     f.save
206     @idwarc = params[:idWarc]
207     respond_to do |format|
208       format.js
209     end
210   end
211
212   def eliminar_favorito
213     favoritos = Favorito.where(:idwarc => params[:idWarc], :idusuario =>
214                                session[:usuario].idusuario).first
215     favoritos.destroy
216     @idwarc = params[:idWarc]
217     respond_to do |format|
218       format.js
219     end
220   end
221
222   def eliminar_favorito_todo
223     @entrada_favorito = Favorito.where(:idusuario => session[:usuario].idusuario)
224     @entrada_favorito.each do |fav|
225       fav.destroy
226     end
227     flash[:mensaje] = "Favoritos eliminados correctamente"
228     redirect_to :action => "listar_favoritos", :controller => "usuario"
229     return
230
231   end
```

Figura 41. Código usuario_controller. Iteración 6

Una vez desplegado la versión del sitio que el usuario seleccionó, si este se encuentra autenticado, se le presenta en la barra de la parte superior de la vista un botón “Agregar a favoritos” o “Eliminar de favoritos”, dependiendo de si dicha versión está o no en la lista de favoritos del usuario, ya que al realizar el despliegue se verifica esto con la base de datos (en **_barra.html.haml**) a través del ID del usuario y del ID del archivo WARC desplegado. Según sea el caso, es agregado (**agregar_favorito**) o eliminado (**eliminar_favorito**) el favorito de la base de datos por medio del **usuario_controller.rb**.

```
!!!
%html
  %head
    %title Archivo Web de Venezuela

    = stylesheet_link_tag    "bootstrap2"
    = javascript_include_tag "application"
    = csrf_meta_tags

  %body
    .navbar.navbar-inverse.navbar-fixed-top
      .navbar-inner
        .container
          %ul.nav
            %li
              %div{:style => "padding-top: 10px"}
                =image_tag ("awv_b3.png"), :width => '120', :alt => "Archivo Web de Venezuela"
            %li
              =link_to "Volver a las versiones", {:action => "prueba_calendar", :controller => "
                home"}
          %ul.nav.pull-right
            -if session[:usuario] != nil
              -if Favorito.where(:idwarc => @idwarc, :idusuario => session[:usuario].idusuario).
                first
                %li{:style => "padding-top: 10px"}
                  =link_to "Eliminar de favoritos", {:action => "eliminar_favorito", :idWarc => "#{
                    @idwarc}", :controller => "usuario"}, :remote => true
                %li{:style => "padding-top: 17px"}
                  =content_tag :i, '', :class => "icon-star icon-white"
              -else
                %li{:style => "padding-top: 10px"}
                  =link_to "Agregar a favoritos", {:action => "agregar_favorito", :controller => "
                    usuario", :idWarc => "#{@idwarc}"}, :remote => true
                %li{:style => "padding-top: 17px"}
                  =content_tag :i, '', :class => "icon-star-empty icon-white"

    %br
    %br
```

Figura 42. Código _barra HAML. Iteración 6

Cuando el usuario hace clic en “Agregar a favoritos” o “Eliminar de favoritos” de acuerdo sea el caso, **_barra.html.haml** que corresponde a la barra superior que se muestra en el despliegue, es renderizada nuevamente para actualizar así el botón de “Agregar a favoritos” o “Eliminar de favoritos”, a través del siguiente javascript:

```
$('.navbar.navbar-inverse.navbar-fixed-top').html("<%= escape_javascript(render(:partial =>
'home/barra',:locals => {:idWarc => @idwarc})).html_safe %>");


```

Figura 43. Código agregar_favorito JS ERB. Iteración 6

Cuando el usuario hace clic en “Mis favoritos” (opción que está disponible para el usuario cuando se encuentra autenticado), se le muestra una lista de sus versiones de los sitios que ha marcado como *Favoritos*. Esta opción se encuentra definida en **usuario_controller.rb** como **listar_favoritos**, y en ella se consulta a la tabla Favoritos de la base de datos con el ID del usuario.

Como parte de la gestión de los Favoritos, se le brinda al usuario la posibilidad de eliminar todos, o de forma individual, sus favoritos. Para eliminar uno de los favoritos (**eliminar_favorito** en **usuario_controller.rb**) se busca el ID correspondiente al Favorito a eliminar y se remueve de la base de datos con la función *destroy* de *ActiveRecord* de Ruby. Para eliminar todos los favoritos

(**eliminar_favorito_todo** en **usuario_controller.rb**), se hace una consulta que los busca, se almacena la respuesta en un arreglo, y se recorre el mismo para eliminar uno a uno cada favorito de la respuesta con la función *destroy*.

Módulo de Historial:

En **usuario_controller.rb** se encuentran definidos una serie de métodos que permiten la gestión de los favoritos: **listar_historial**, **eliminar_historial, eliminar_historial _todo**. Y en **application_controller.rb** está definida **historial** que es la que añade las búsquedas del usuario al historial. Esta se encuentra en dicho controlador, para que pueda ser llamada desde el controlador que realiza el despliegue.

```
8   def historial (nombreurl, version, direccion)
9       h = Historial.new
10      h.fecha = Time.now
11      h.idusuario = session[:usuario].idusuario if session[:usuario]
12      h.url_nombre = nombreurl
13      h.url_version = version
14      h.direccion_warc = direccion
15      h.save
16
17    end
```

Figura 44. Código application_controller. Iteración 6

Cuando el usuario selecciona la versión del sitio que desea desplegar, se llama entonces a la función **historial** pasándole el url y la versión del sitio, más el ID del warc. Aquí se crea una entrada en la tabla Historial de la base de datos con la fecha y hora actual, el ID del usuario que está autenticado, el url y la versión del sitio que seleccionó el usuario para desplegar y el ID del WARC.

Cuando el usuario hace clic el "Mi historial de búsqueda" (opción que está disponible para el usuario cuando se encuentra autenticado), se le muestra un listado de su historial. Esta opción se encuentra definida en **usuario_controller.rb** como **listar_historial**, y en ella se consulta a la tabla Historial de la base de datos con el ID del usuario.

Como parte de la gestión del Historial, se le brinda al usuario la posibilidad de eliminar todos, o de forma individual, su historial. Para eliminar una entrada de su historial (**eliminar_historial** en **usuario_controller.rb**) se busca en este el ID correspondiente a dicha entrada del historial a eliminar y se remueve de la base de datos con la función *destroy* de *ActiveRecord* de Ruby. Para eliminar todo el historial (**eliminar_historial_todo** en **usuario_controller.rb**), se hace una consulta que busca todas las entradas del mismo que corresponden al usuario autenticado, se almacena la respuesta en un arreglo, y se recorre el mismo para eliminar uno a uno cada entrada del historial de la respuesta con la función *destroy*.

```
170   def listar_historial
171     @usuario_historial = Historial.paginate(:page => params[:page], :conditions => {:idusuario =>
172                          session[:usuario].idusuario}).order ("fecha desc")
173   end
174
175   def eliminar_historial
176     @entrada_historial = Historial.find(params[:idhis])
177     @entrada_historial.destroy
178     flash[:mensaje] = "Entrada eliminada correctamente de su historial de búsqueda"
179     redirect_to :action => "listar_historial", :controller => "usuario"
180     return
181
182   end
183   def eliminar_historial_todo
184     @entrada_historial = Historial.where(:idusuario => session[:usuario].idusuario)
185     @entrada_historial.each do |his|
186       his.destroy
187     end
188     flash[:mensaje] = "Historial de búsqueda eliminado correctamente"
189     redirect_to :action => "listar_historial", :controller => "usuario"
190     return
191
192   end
```

Figura 45. Código usuario_controller. Iteración 6

- **Pruebas**

Tabla 24. Caso de prueba: 11. Iteración 6

Número de Caso de Prueba: 11	**Número de Historia de Usuario:** 9
Descripción: Al desplegar la versión del sitio seleccionado por el usuario, si fue marcado como *Favorito* previamente indicárselo al usuario en la barra superior	
Resultado: Sitio desplegado previamente marcado como *Favorito* 	

Tabla 25. Caso de prueba: 12. Iteración 6

Número de Caso de Prueba: 12	**Número de Historia de Usuario:** 9
Descripción: Al desplegar la versión del sitio seleccionado por el usuario, marcado como *Favorito* previamente, eliminar de los Favoritos del usuario	
Resultado: Sitio desplegado previamente marcado como *Favorito*  Eliminado de *Favoritos*  	

Tabla 26. Caso de prueba: 13. Iteración 6

Número de Caso de Prueba: 13	**Número de Historia de Usuario:** 9
Descripción: Listar exitosamente los Favoritos del usuario	
Resultado:	

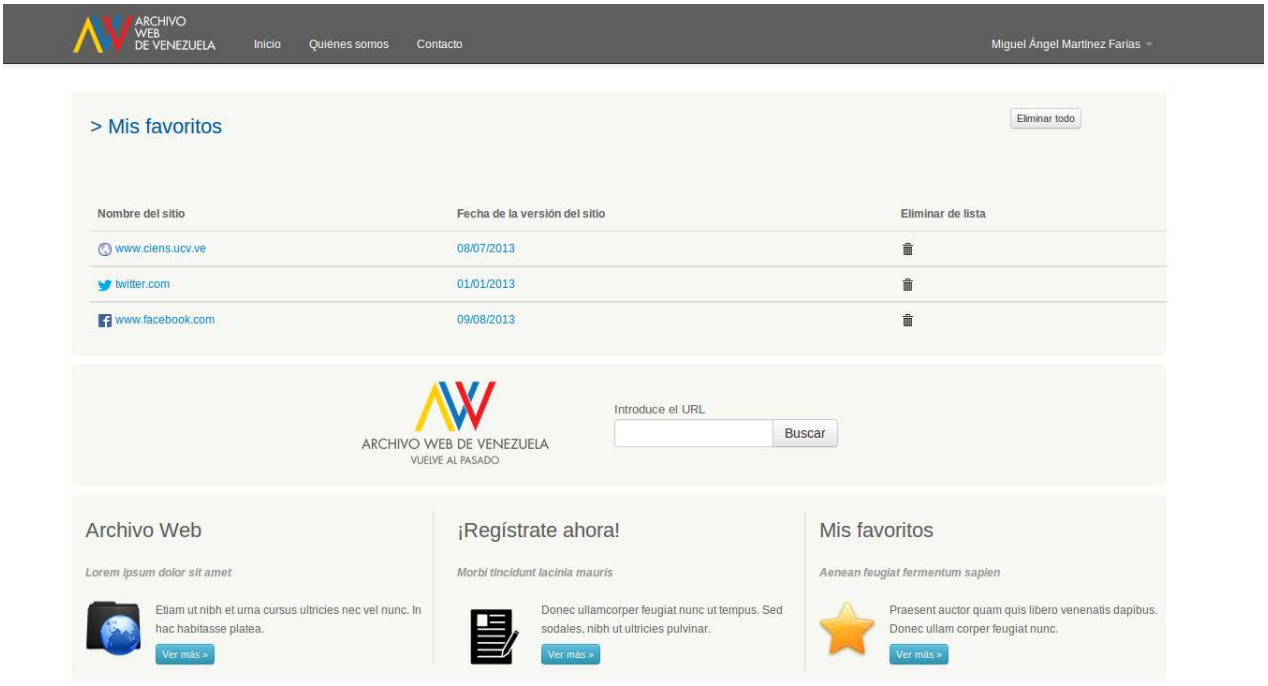


Tabla 27. Caso de prueba: 14. Iteración 6

Número de Caso de Prueba: 14	**Número de Historia de Usuario:** 10
Descripción: Listar exitosamente el historial del usuario	
Resultado:	

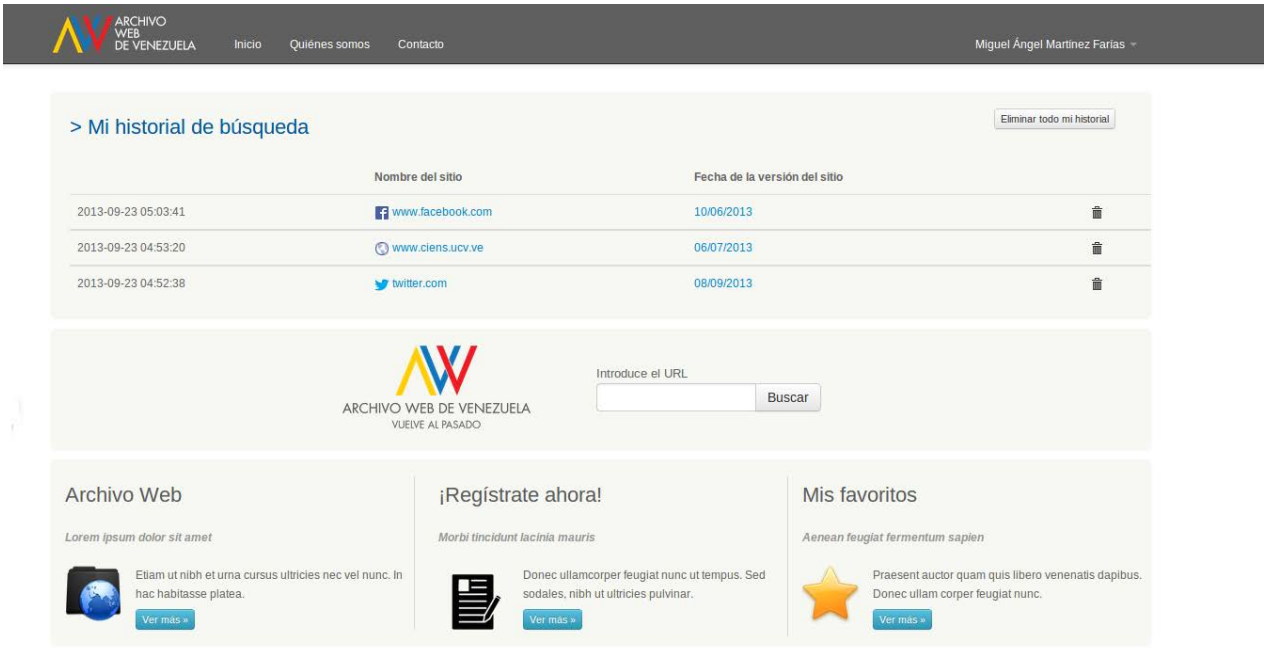
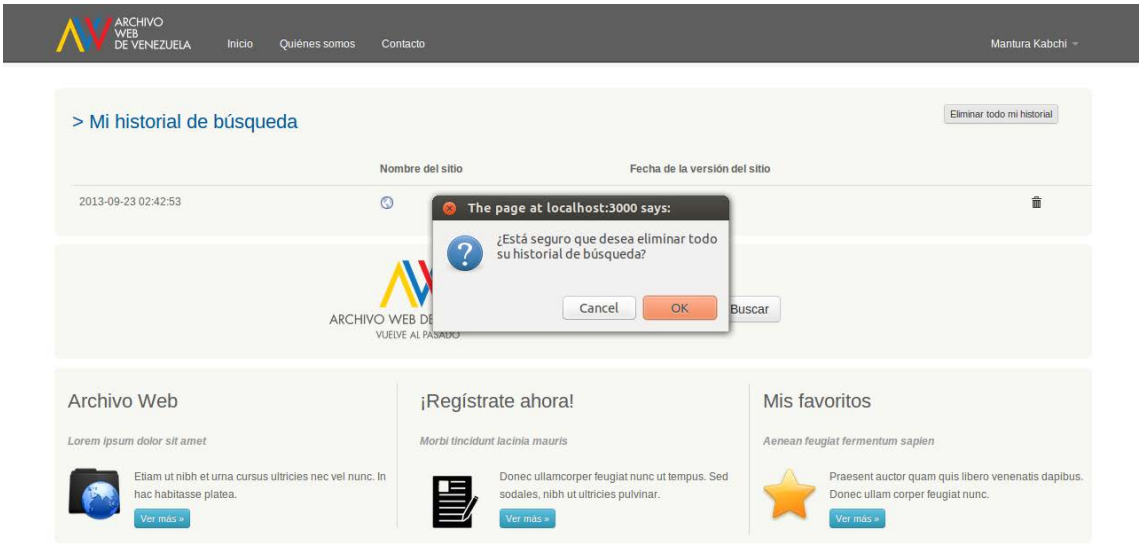
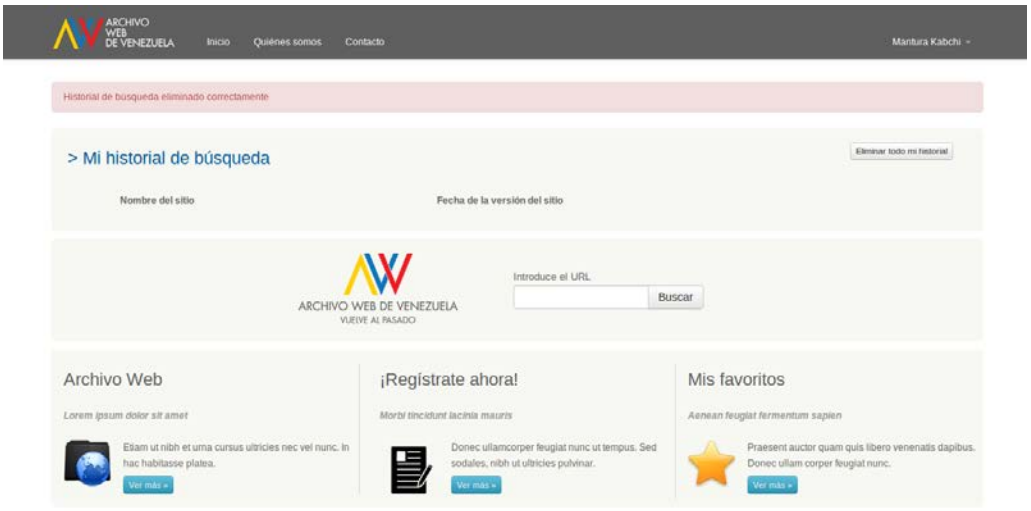


Tabla 28. Caso de prueba: 15. Iteración 6

Número de Caso de Prueba: 15	**Número de Historia de Usuario:** 10
Descripción: Eliminar todo el historial de búsqueda del usuario y notificárselo	
Resultado:	



Notificación al usuario:



3.3.8. Iteración 7

- **Planificación:** Esta iteración plantea como objetivo la creación del módulo de selección de versión del sitio web solicitado de acuerdo a las cosechadas a través de un calendario. Además de esto, proveer al usuario estadísticas acerca de las versiones cosechadas del sitio Web solicitado, usando para ello una gráfica.

Tabla 29. Planificación Iteración 7

Número de Iteración: 7	**Número de Historia:** 6, 7, 11
Fecha de Inicio: 20/08/2013	**Fecha de Fin:** 20/09/2013
Descripción: Creación del módulo de selección de versión a través de un calendario y proveer información estadística de las versiones cosechadas mediante un gráfico.	**Tipo:** Desarrollo

- **Diseño:** Para la fase de diseño se realiza un diagrama de secuencia que constituye el proceso de selección de versión a través de un calendario. A continuación dicho diagrama:

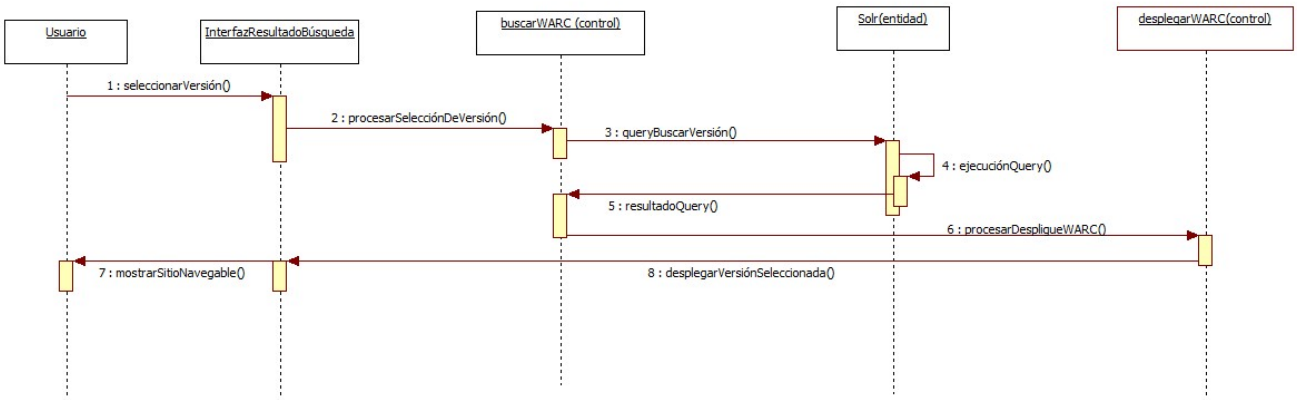


Figura 46. Diagrama de secuencia Selección de versión. Iteración 7

Para ello se adapta un selector de fechas con JavaScript provisto por jQuery UI, con el cual se genera el calendario anual que se le presenta al usuario, indicándole las fechas en la que se almacenó una versión del sitio que buscó.

- **Codificación:** Siguiendo el patrón MVC, el módulo desarrollado en esta iteración se encuentra manejados a través del controlador **home_controller.rb**. En él se define la acción **version_calendario** la cual es llamada cuando el usuario realiza una búsqueda de algún url. En esta acción se recibe el parámetro de la búsqueda y se procesa mediante una petición http al motor de búsqueda Solr, provista por el Módulo de Indexación y Rastreo del Prototipo de Archivado Web de Venezuela. En dicha petición se solicita la respuesta en un json y con el uso de la gema Httparty es parseado para luego ser trabajado. El json obtenido al consultar ww.ciens.ucv.ve:

```
{"responseHeader"=>
    {
        "status"=>0,
        "QTime"=>0,
        "params"=>{
            "indent"=>"on",
            "start"=>"0",
            "q"=>"URL:*ciens*",
            "wt"=>"json",
            "version"=>"2.2",
            "rows"=>"10"
        }
    },
 "response"=>
    {
        "numFound"=>3,
        "start"=>0,
        "docs"=>
            [{"warcid"=>"www.ciens.ucv.ve_20130413063513",
            "maquina"=>"190.169.69.155", "URL"=>"www.ciens.ucv.ve",
            "ruta"=>"/usr/local/heritrix-3.1.1/jobs/
            ciencias/20130413063513", "fecha"=>"2013-04-13T06:35:13Z",
            "nombreWarc"=>"WEB-20130413063516717-00000-3208~menorz~8443.
            warc.gz", "fecha_fin"=>"2012-11-15T21:10:16.237Z",
            "fecha_inicio"=>"2012-11-15T21:10:16.237Z"},

            {"warcid"=>"www.ciens.ucv.ve_20130514044121",
            "maquina"=>"190.169.69.155", "URL"=>"www.ciens.ucv.ve",
            "ruta"=>"/usr/local/heritrix-3.1.1/jobs/
            ciencias/20130514044121", "fecha"=>"2013-05-14T04:41:21Z",
            "nombreWarc"=>"WEB-20130514044127646-00000-1736~menorz~8443.
            warc.gz", "fecha_fin"=>"2012-11-15T21:12:07.66Z",
            "fecha_inicio"=>"2012-11-15T21:12:07.66Z"},

            {"warcid"=>"www.ciens.ucv.ve_20130618030138",
            "maquina"=>"190.169.69.155", "URL"=>"www.ciens.ucv.ve",
            "ruta"=>"/usr/local/heritrix-3.1.1/jobs/
            ciencias/20130618030138", "fecha"=>"2013-06-18T03:01:38Z",
            "nombreWarc"=>"WEB-20130618030214850-00000-2109~menorz~8443.
            warc.gz", "fecha_fin"=>"2012-11-15T21:13:41.599Z",
            "fecha_inicio"=>"2012-11-15T21:13:41.599Z"}]
    }
}
```


Figura 47. Código JSON generado por el módulo de indexación

De él tomamos los datos del response que nos interesan para procesarlo, como numFound para el número de rastreos encontrados y docs que presenta la información de los rastreos.

```
16  def version_calendario
17    #CONSULTA A SOLR
18
19    unless params[:URL] && params[:URL][:url]
20      flash[:mensaje] = "El campo de búsqueda está vacío"
21      redirect_to :action => "index"
22      return
23    end
24
25    @nombre_url = params[:URL][:url]
26    @query_curl= 'http://190.169.69.154:8080/solr/select/?wt=json&q=URL%3A*'+@nombre_url+
27                 '*&version=2.2&start='+'0'+'&rows=10&indent=on'
28    response = HTTParty.get(@query_curl)
29    data= response.body
30    @json_result = ActiveSupport::JSON.decode(data)
31    contador=10
32    while contador < @json_result["response"]["numFound"] do
33
34      @query_curl= 'http://190.169.69.154:8080/solr/select/?wt=json&q=URL%3A'+@nombre_url+
35                   '&version=2.2&start='+contador.to_s()+'&rows=10&indent=on'
36      response = HTTParty.get(@query_curl)
37      data= response.body
38      @json_resulttmp = ActiveSupport::JSON.decode(data)
39
40      @json_resulttmp["response"]["docs"].each do |nuevo|
41        @json_result["response"]["docs"] << nuevo
42      end
43      contador+= 10
44    end
45
46    if @json_result["response"]["numFound"] == 0
47      flash[:mensaje] = "Disculpe, no se han almacenado archivos correspondientes
48                         a su búsqueda"
49      redirect_to :action => "index"
50      return
51    end
52
53    @arreglo_json_url = Array.new
54    @arreglo_json_ruta = Array.new
55    @arreglo_json_maquina = Array.new
56    @arreglo_json_warc = Array.new
57    @arreglo_json_fecha = Array.new
58    @años = Array.new
59    @horas = Array.new
```

Figura 48. Código home_controller Obtención JSON. Iteración 7

Dichos datos son agrupados en diferentes arreglos como in_anio (contiene los años de los rastreos para la gráfica), grafica_contar (contiene el número de rastreos por año para la gráfica), fecha (contiene la fecha de los rastreos), maquina (contiene la máquina virtual en la que se encuentra el WARC), warc (contiene el nombre del WARC), ruta (contiene la ruta donde

se encuentra el WARC), url (contiene el nombre del URL al que pertece el WARC) y horas (contiene las horas de los rastreos). Estos arreglos son pasados al javascript que se encargas de desplegar en la vista toda esta información en forma de gráfica por un lado, y un calendario anual por otro. Esto se hace a través de la gema de Ruby on Rails llamada **gon,** pues permite enviar data al javascript directamente desde el controlador.

```
61  #Resultados del json
62      @json_result["response"]["docs"].each do |campos|
63        @arreglo_json_url<<campos["URL"]
64        @arreglo_json_ruta<<campos["ruta"]
65        @arreglo_json_maquina<<campos["maquina"]
66        @arreglo_json_warc<<campos["nombreWarc"]
67
68  #mes/dia/año
69        @fechasplit = campos["fecha"].split('T')
70        @fechasplit2 = @fechasplit[0].split('-')
71        @aux = @fechasplit[1].split('Z')
72        @aux2 = @fechasplit[1]
73        @horas << @aux2[0,8]
74
75        @fecha_completa = @fechasplit2[1]<<"/"<<@fechasplit2[2]<<"/"<<@fechasplit2[0]
76        @años << @fechasplit2[0]
77        @arreglo_json_fecha<<@fecha_completa
78      end
79
80  # contador para la grafica
81      @b = Hash.new(0)
82      @años.each do |v|
83        @b[v] += 1
84      end
85
86      @grafica_fecha = Array.new
87      @grafica_contar = Array.new
88
89      @b.each do |g|
90        @grafica_fecha<<g[0]
91        @grafica_contar<<g[1]
92      end
93
94  #Arreglos con los datos para calendar.js
95      gon.in_anio = @grafica_fecha.min.to_s
96      gon.grafica_contar = @grafica_contar
97      gon.grafica_fecha = @grafica_fecha
98      gon.fecha = @arreglo_json_fecha
99      gon.maquina = @arreglo_json_maquina
100     gon.warc = @arreglo_json_warc
101     gon.ruta = @arreglo_json_ruta
102     gon.url = @arreglo_json_url
103     gon.horas = @horas
104
105     render :layout => "otros2"
106   end
107 end
```

Figura 49. Código home_controller Parseo de JSON. Iteración 7

Teniendo estos datos en los diferentes arreglos, son recibidos en el javascript **calendar.js:**

```
(function($){
    $(document).ready(function(){
    var donde=[];
    var grafica_contar = gon.grafica_contar;
    var grafica_fecha = gon.grafica_fecha;
    var url = gon.url;
    var ruta = gon.ruta;
    var maquina = gon.maquina;
    var warc = gon.warc;
    var fecha = gon.fecha;
    var hora = gon.horas;
    var in_anio = gon.in_anio;
    var ruta_al_warc = new Array();
    var events = [];

```

Figura 50. Código calendario JS Declaración de variables. Iteración 7

En este javascript se tienen dos script principales, uno de ellos corresponde al calendario anual el cual se lleva a cabo mediante una adaptación del datepicker de jQueryUI. En el calendario se configura el número de meses que deseamos que aparezcan en la vista, el rango de años, nombre de los meses, entre otros. En la función BeforeShowDay son marcadas en amarillo las fechas que presentan una o más versiones del URL solicitado por el usuario. En la función OnSelect, que es la que contiene el comportamiento del calendario al seleccionar alguno de los días marcados en amarillo, se verifica si en dicha fecha seleccionada hay uno o más rastreos. Si es un solo rastreo, se hace el llamado al controlador despliegue_controller.rb pasándole por parámetro a la función buscar la ruta donde se encuentra el WARC, el nombre del WARC y la IP de la máquina donde se encuentra dicho WARC. De lo contrario, es decir, si hay más de un WARC, se le indica al usuario las horas de los rastreos de dicha fecha y el debe seleccionar entonces la versión según la hora que desea desplegar. Este despliegue se lleva a cabo de la misma forma explicada anteriormente.

```
53          onSelect: function(dateText) {
54              var masdeunwarc=0;
55              var date,
56              selectedDate = new Date(dateText),
57              i = 0,
58              event = null;
59
60              while (i < events.length && !event) {
61                  date = events[i].Date;
62                  if (selectedDate.valueOf() === date.valueOf()) {
63                      event = events[i];
64                  }
65                  i++;
66              }
67              if (event) {
68                  masdeunwarc = countItems(fecha, (event.Fecha));
69                  if (masdeunwarc > "1"){
70                      var w=0;
71                      donde.length = 0;
72                      for (var i = 0; i < fecha.length; i++){
73                          if (fecha[i] == event.Fecha) {
74                              donde[w]= i;
75                              ++w;
76                          }
77                      }
78                      f =0;
79                      for (var i = 0; i < donde.length; i++){
80                          ruta_al_warc[i] = ruta[donde[f]] + "," + warc[donde[f]] + "," + maquina[donde[f]];
81                              ++f;
82
83                      }
84                      $( "titulo" ).text("Versiones");
85                      $( "versiones" ).html(" ");
86                      $( "numversiones" ).text( "El " + event.Fecha + " se almacenaron " + masdeunwarc +
87                                                " versiones de " + url[donde[0]]);
88                      for (var t=0; t < donde.length; t++){
89                          $( "versiones" ).append('<a href="/despliegue/buscar?data=' + ruta_al_warc[t] +
90                                                  '">Hora: ' + hora[donde[t]] + "</a><br />");
91                      }
92                      top.location.href = '#top';
93                  }else{
94                      window.location.href = "/despliegue/buscar?data=" + ruta_al_warc;
95                  }
96              }
97          }
```

Figura 51. Código calendario JS validación del enlace. Iteración 7

El otro script principal de **calendario.js** es el que construye la gráfica que provee al usuario información sobre el número de rastreos por año del URL solicitado por él. De este modo, se crea un arreglo del tipo hash con los datos traídos de **home_controller.rb** para la gráfica que corresponden a los años y al número de rastreos. Con estos datos más la librería Amcharts (librería de javascript para realizar gráficos), se logró el gráfico requerido para proveerle al usuario información estadística de las versiones cosechadas.

```
104     //Data del grafico
105     var chart;
106     var chartData = [];
107     w = 0;
108     while (w < grafica_contar.length) {
109         chartData[w] =  { anios: grafica_fecha[w], numero_rastreos: grafica_contar[w]};
110         w++;
111     }
112     //Elaboración del gráfico
113     AmCharts.ready(function () {
114
115         chart = new AmCharts.AmSerialChart();
116         chart.colors = ["#0D8ECF"];
117         chart.zoomOutButton = {
118             backgroundColor: "#000000",
119             backgroundAlpha: 0.15
120         };
121         chart.dataProvider = chartData;
122         chart.categoryField = "anios";
123
124         // Ejes
125         //Categoria
126         var categoryAxis = chart.categoryAxis;
127         categoryAxis.gridAlpha = 0.07;
128         categoryAxis.axisColor = "#DADADA";
129         categoryAxis.startOnAxis = true;
130         // Valor
131         var valueAxis = new AmCharts.ValueAxis();
132         valueAxis.gridAlpha = 0.07;
133         valueAxis.integersOnly= true;
134         chart.addValueAxis(valueAxis);
135         // Grafo
136         var graph = new AmCharts.AmGraph();
137         graph.type = "column";
138         graph.title = "Versiones";
139         graph.valueField = "numero_rastreos";
140         graph.balloonText = "[[value]]";
141         graph.lineAlpha = 0.6;
142         graph.fillAlphas = 0.6;
143         chart.addGraph(graph);
144         // Cursor
145         var chartCursor = new AmCharts.ChartCursor();
146         chartCursor.zoomable = false;
147         chartCursor.cursorAlpha = 0;
148         chart.addChartCursor(chartCursor);
149         // Escribir el gráfico
150         chart.write("chartdiv");
151     });
```

Figura 52. Código calendario JS visualización del calendario. Iteración 7

- **Pruebas:** A continuación las pruebas de esta iteración.

Tabla 30. Caso de prueba: 16. Iteración 7

Número de Caso de Prueba: 16	**Número de Historia de Usuario:** 7
Descripción: Al realizar una búsqueda de determinado sitio, mostrarle al usuario las diferentes versiones existentes de dicho sitio mediante un calendario para que el usuario pueda seleccionar la versión a desplegar.	

Resultado: El resultado es un calendario con las diferentes versiones existentes en el archivo web del sitio buscado por el usuario, marcadas en amarillo. Cuando el usuario hace clic en alguna de ellas, dicha versión es desplegada.

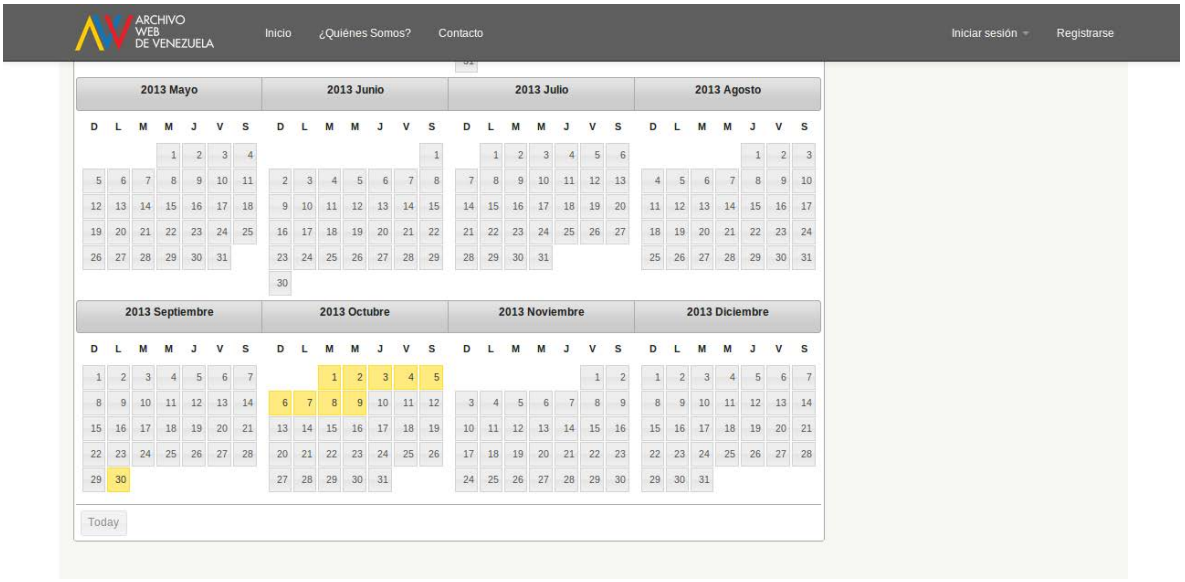
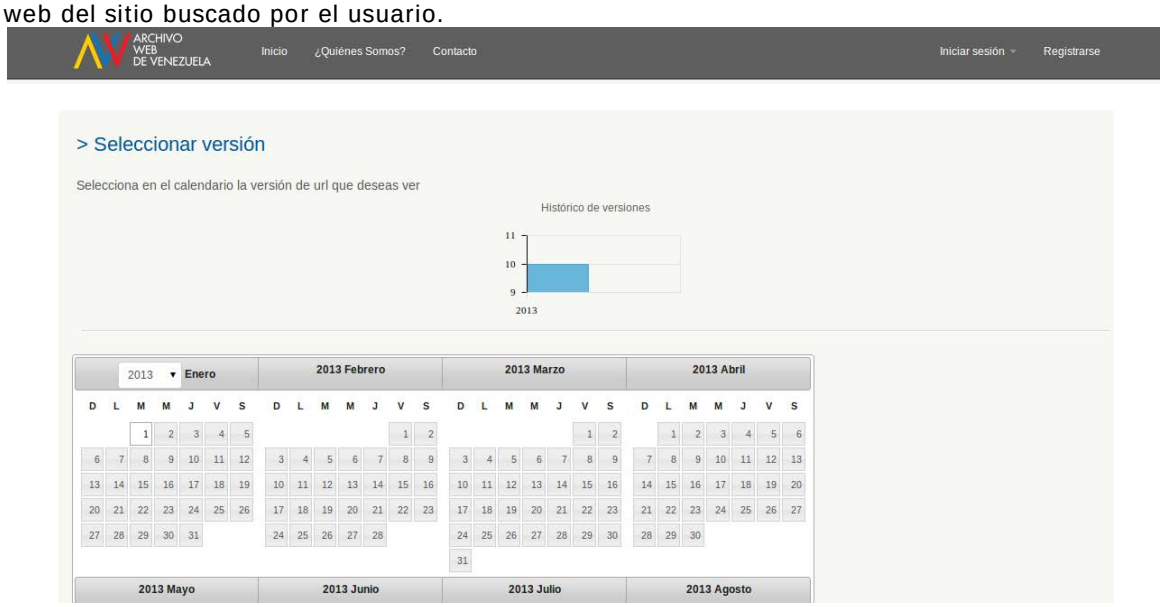


Tabla 31. Caso de prueba: 17. Iteración 7

Número de Caso de Prueba: 17	**Número de Historia de Usuario:** 11
Descripción: Al realizar una búsqueda de determinado sitio, proveer al usuario información estadística de las versiones cosechadas de dicho sitio mediante un gráfico.	
Resultado: El resultado es una gráfica que muestra por año cuántas versiones existen en el archivo web del sitio buscado por el usuario.	



3.3.9. Iteración 8

- **Planificación:** En esta iteración el objetivo es modificar el proceso de despliegue buscando optimizarlo y generalizar aún más la solución al problema de desigualdad en la formación de archivos HTML.

Tabla 32. Planificación Iteración 8

Número de Iteración: 8	**Número de Historia:** 8
Fecha de Inicio: 30/09/2013	**Fecha de Fin:** 4/12/2013
Descripción: Modificación del módulo de despliegue de archivos WARC: optimización del código y generalización de la solución	**Tipo:** Desarrollo

- **Diseño:** En el nuevo diseño del módulo de despliegue se utilizará la carpeta *public*, propia de Ruby on Rails, para almacenar las versiones de los archivos WARC que son descargadas desde el servidor. En el servidor se realiza la extracción del archivo WARC correspondiente a la versión escogida por el usuario y luego la misma es copiada en la carpeta *public* mencionada anteriormente. Además el archivo de rutas de Rails es modificado para que reconozca la nueva carpeta como válida.

  El flujo de datos de la aplicación es el siguiente:

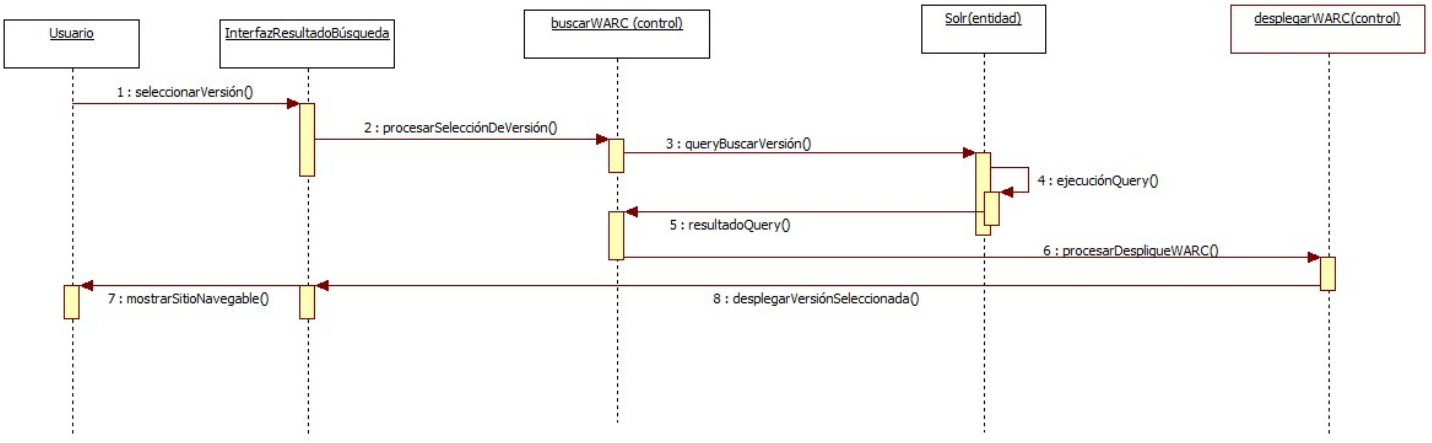


Figura 53. Diagrama de secuencia Despliegue. Iteración 8

Esta iteración se encarga de resolver los problemas de los pasos procesarDespliegueWarc, desplegarVersionSeleccionada y mostrarSitioNavegable.

- **Codificación:** El usuario selecciona la versión del archivo que quiere desplegar y navegar, su opción es procesada, se verifica si el archivo se encuentra en la carpeta *public* y en ese caso se realiza el despliegue del sitio sin necesidad de manipular los archivos, ya que si se encuentra en dicha carpeta significa que previamente fueron tratados. En el caso contrario, es decir, el archivo no se encuentra en la carpeta *public* porque no ha sido desplegado anteriormente, se procede a buscar el mismo en el servidor. Cuando el usuario realiza una búsqueda, la aplicación realiza la consulta a *Solr* y una vez que es seleccionada la versión del sitio a desplegar, la respuesta trae consigo todos los datos del WARC;y en la función buscar estos son separados y organizados en diferentes variables.

```
6   def buscar
7     session[:return_to] ||= request.referer
8     ruta_warc = params[:data].split(",")
9     nombre_warc_aux = ruta_warc[1].split(".")
10    nombre_warc_def = nombre_warc_aux[0]
11
12    file = "#{ruta_warc[0]}/warcs/#{nombre_warc_def}"
13
14    if ruta_warc[2] == "190.169.69.155"
15      raiz_warctools = "/root/paquetes/warctools/"
16    else
17      raiz_warctools = "/root/warctools/"
18    end
19
20    @idwarc = ruta_warc[5]
21    horawarc = ruta_warc[4]
22    fechawarc = ruta_warc[3]
23    warc_nombre = nombre_warc_def
24    warc_ruta = ruta_warc[0]
25    maquina_warc = ruta_warc[2]
26    url_warc = ruta_warc[6]
```

Figura 54. Código despliegue_controller. Iteración 8

Después de esto, la aplicación ingresa a la carpeta *reports* propia del rastreo de la versión a desplegar donde hay un archivo generado por el rastreo llamado seeds-report.txt donde obtenemos la ruta del archivo index.html del sitio a desplegar, siendo este el archivo principal del sitio.

```
43    url_seed = `ssh root@#{ruta_warc[2]} sed '2p'
44                #{ruta_warc[0]}/reports/seeds-report.txt
45                | cut -d ' ' -f 3`
```

Figura 55. Código obtención de directorio principal. Iteración 8

Luego de esto, la aplicación añade al archivo de rutas del proyecto **routes.rb** la ubicación de la nueva carpeta que será agregada en *public* con la versión del sitio a desplegar, siempre y cuando dicha ruta no esté en el archivo.

```
40  warc_nombre = url_warc.split(".")
41  @match_nombre = "#{warc_nombre[1]}_#{@idwarc}"
42
```

Figura 56. Código Obtención de ruta dinámica. Iteración 8

Si la ruta se encuentra en el archivo **routes.rb** se despliega el sitio.

```
71  line = File.read("config/routes.rb")
72  if line["match '/#{@match_nombre}'"]
73      redirect_to :action => "despliegue_barra",
74                  :controller => "home", :el_match => @match_nombre
75      return
76  end
```

Figura 57. Código redirección a WARC. Iteración 8

De lo contrario, la ruta es añadida al final del archivo **routes.rb** y se ejecuta el comando 'rake routes' para añadir la ruta nueva a las rutas válidas de Rails.

```
78  last_line = 0
79  file = File.open('config/routes.rb', 'r+')
80  file.each {  last_line = file.pos unless file.eof? }
81  file.seek(last_line, IO::SEEK_SET)
82  file.write "match '/#{@match_nombre}', :to =>
83              redirect('/#{@idwarc}/#{ultimo_dir_http}
84              /crawlerdefault.html')\nend"
85  file.close
86  `rake routes`
```

Figura 58. Código Escritura en archivo de rutas. Iteración 8

En este punto, se desempaqueta el archivo utilizando la función *gunzip* propia de la librería de comandos de Linux, se extraen los archivos utilizando un script escrito en *python* llamado *warcunpack_ia.py* el cual forma parte de los *warctools* y se crea la carpeta en *public* donde son copiados los archivos resultantes de la extración, cuyo nombre es el dominio del sitio y la fecha y hora de la versión. Seguidamente se elimina el archivo descomprimido y el extraído, dejando el servidor de rastreos exactamente como se encontraba, evitando así duplicados en el almacenamiento.

```
94      `ssh root@#{ruta_warc[2]} "gunzip -c
95       #{ruta_warc[0]}/warcs/#{nombre_warc_def}.warc.gz >
96       #{ruta_warc[0]}/warcs/#{nombre_warc_def}.warc"`
97      `ssh root@#{ruta_warc[2]} #{raiz_warctools}./warcunpack_ia.py
98       #{ruta_warc[0]}/warcs/#{nombre_warc_def}.warc -o
99       #{ruta_warc[0]}/warcs/#{nombre_warc_def}`
100     `ssh root@#{ruta_warc[2]} rm
101      #{ruta_warc[0]}/warcs/#{nombre_warc_def}.warc`
102     `mkdir public/#{@idwarc}`
103     `scp -r root@#{ruta_warc[2]}:#{carpeta} public/#{@idwarc}/`
104     `ssh root@#{ruta_warc[2]} rm -Rf
105      #{ruta_warc[0]}/warcs/#{nombre_warc_def}`
```

Figura 59. Código para desempaquetar archivo WARC. Iteración 8

Posteriormente, se obtienen y colocan en un arreglo las rutas de todos los archivos html, y en otro arreglo las de las imágenes, mediante la función **glob** propia de Ruby.

```
108   @directorios_html = Dir.glob("public/#{@idwarc}/**/*.html")
109   @directorios_html.each do |separar|
110     @arreglo_html << separar
111     @dir_aux_html = separar.split("#{@idwarc}/#{ultimo_dir_http}/")
112     @aux_html = @dir_aux_html[1].split("/")
113     @ultimo_html= @aux_html.length - 1
114     @ultimo_directorio_html = @dir_aux_html[1].chomp(@aux_html[@ultimo_html])
115     @nombre_html = @aux_html[@ultimo_html].split(".")
116     @arreglo_html_sub << "#{@ultimo_directorio_html}#{@nombre_html[0]}"
117   end
118   @directorios_images = Dir.glob("public/#{@idwarc}/**/*.{jpg,jpe,gif,png,jpeg}")
119
120   @directorios_images.each do |separar|
121     if separar[/\.jpg/] or separar[/\.gif/] or separar[/\.jpe/] or
122        separar[/\.jpeg/] or separar[/\.png/]
123       @dir_aux = separar.split("#{@idwarc}/#{ultimo_dir_http}/")
124       if separar[/\.jpe/]
125         @aux_img = @dir_aux[1].split("/")
126         @ultima_img = @aux_img.length - 1
127         @ultimo_directorio_img = @dir_aux[1].chomp(@aux_img[@ultima_img])
128         @ultimo_directorio_separar = separar.chomp(@aux_img[@ultima_img])
129         @nombre_imagen = @aux_img[@ultima_img].split(".")
130         `mv #{separar} #{@ultimo_directorio_separar}#{@nombre_imagen[0]}.jpg`
131         @arreglo_img << "#{@ultimo_directorio_img}#{@nombre_imagen[0]}.jpg"
132       else
133         @arreglo_img << @dir_aux[1]
134       end
135     end
136   end
```

Figura 60. Código de tratamiento archivo WARC. Iteración 8

Teniendo los html y las imágenes guardadas en arreglos, es llamada la función recorrido_html, la cual se encarga de realizar las sustituciones dentro de cada html adaptando las rutas a la nueva ubicación del sitio (carpeta *public* del Módulo de Acceso y Despliegue).

```
137   @arreglo_html.each do |ruta|
138     recorrido_html(ruta, @arreglo_img, ultimo_dir_http, @idwarc, @arreglo_html_sub)
139   end
```

Figura 61. Código para llamada al método de tratamiento del WARC. Iteración 8

En primer lugar se realiza la lectura del archivo html sobre el cual se trabajarán las sustituciones de rutas. Se recorre el arreglo de imágenes y si en el archivo html se encuentra una ruta que coincida con alguna de las que se encuentran en el arreglo de imágenes se reemplaza la ruta que está escrita en la página original por la que hace referencia a *public* y a la carpeta de la versión seleccionada por el usuario.

```
260 def recorrido_html(ruta, arreglo_imagenes, ultimo_dir_http, idwarc, arreglo_html_sub)
261   arreglo_imagenes.each do |img|
262     line = File.read(ruta)
263     begin
264       reemplazo = line.gsub(/\"\/#{img}"/, "\"\/#{idwarc}/#{ultimo_dir_http}/#{img}\"")
265       File.open(ruta, "w") {|file| file.puts reemplazo}
266     rescue ArgumentError
267       reemplazo = line.force_encoding('iso-8859-1').gsub(/\"\/#{img}"/,
268                                 "\"\/#{idwarc}/#{ultimo_dir_http}/#{img}\"")
269       File.open(ruta, "w") {|file| file.puts reemplazo}
270     end
271   end
```

Figura 62. Código para modificar rutas de imágenes. Iteración 8

De igual modo se realiza este proceso con el arreglo de los html, buscando en cada uno de los archivos html coincidencias de rutas con las de dicho arreglo y reemplazándolas con la ruta que hace referencia a *public* y a la carpeta de la versión seleccionada por el usuario. Asimismo, se modifica las etiquetas de hipervínculos **a** que especifican que el enlace debe abrir en una página o pestaña nueva para que abra en el mismo marco desde el cual fue activado el evento (modificamos los target="_blank" por target="_self"), de modo de mantener el *header* de la aplicación en el tope de la página, ya que el sitio desplegado es renderizado en un marco que se encuentra bajo dicho *header.*

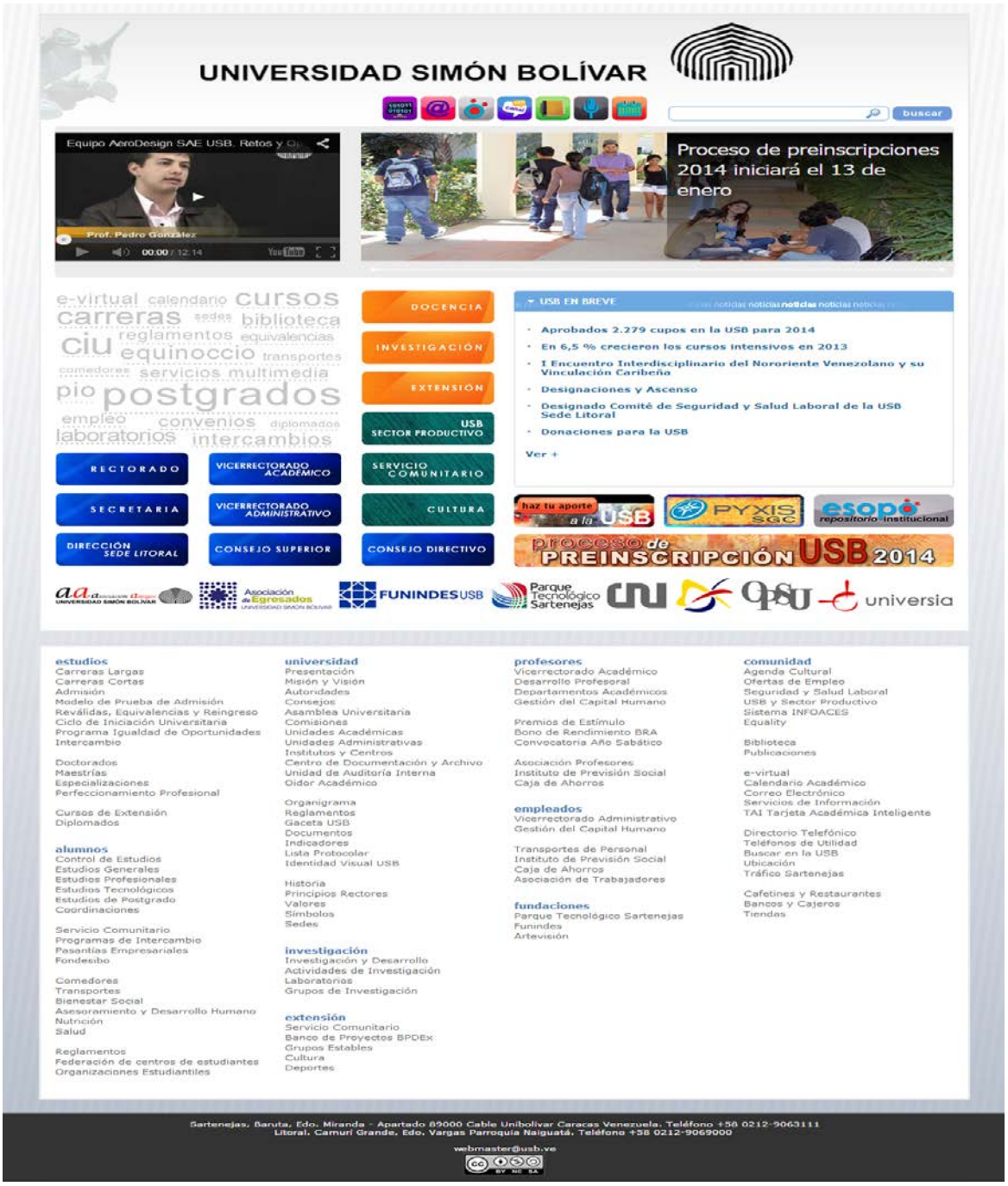
```
arreglo_html_sub.each do |html|
line = File.read(ruta)
begin
  reemplazo = line.gsub(/src\s*\=\s*\"http:\/\/#{ultimo_dir_http}\//,
              "src=\"\/#{idwarc}/#{ultimo_dir_http}/")
  reemplazo2 = reemplazo.gsub(/href\s*\=\s*\"http:\/\/#{ultimo_dir_http}\//,
               "href=\"\/#{idwarc}/#{ultimo_dir_http}/")
  reemplazo3 = reemplazo2.gsub(/\"\/#{html}\"/,
               "\"\/#{idwarc}/#{ultimo_dir_http}/#{html}\"")
  reemplazo4 = reemplazo3.gsub(/\"\/#{html}\.html\"/,
               "\"\/#{idwarc}/#{ultimo_dir_http}/#{html}\.html\"")
  reemplazo5 = reemplazo4.gsub(/\"_blank\"/, "\"_self\"")
  reemplazo6 = reemplazo5.gsub(/\"http:\/\/#{ultimo_dir_http}\/\"|\"http:\/\/#{ultimo_dir_http}\"/,
               "\"\/#{idwarc}/#{ultimo_dir_http}/crawlerdefault.html\"")
  File.open(ruta, "w") {|file| file.puts reemplazo6}
rescue ArgumentError
  reemplazo = line.force_encoding('iso-8859-1').gsub(/src\s*\=\s*\"http:\/\/#{ultimo_dir_http}\//,
              "src=\"\/#{idwarc}/#{ultimo_dir_http}/")
  reemplazo2 = reemplazo.force_encoding('iso-8859-1').gsub
               (/href\s*\=\s*\"http:\/\/#{ultimo_dir_http}\//,
                "href=\"\/#{idwarc}/#{ultimo_dir_http}/")
  reemplazo3 = reemplazo2.force_encoding('iso-8859-1').gsub(/\"\/#{html}\"/,
               "\"\/#{idwarc}/#{ultimo_dir_http}/#{html}\"")
  reemplazo4 = reemplazo3.force_encoding('iso-8859-1').gsub(/\"\/#{html}\.html\"/,
               "\"\/#{idwarc}/#{ultimo_dir_http}/#{html}\.html\"")
  reemplazo5 = reemplazo4.force_encoding('iso-8859-1').gsub(/\"_blank\"/, "\"_self\"")
  reemplazo6 = reemplazo5.force_encoding('iso-8859-1').gsub
               (/\"http:\/\/#{ultimo_dir_http}\/\"|\"http:\/\/#{ultimo_dir_http}\"/,
               "\"\/#{idwarc}/#{ultimo_dir_http}/crawlerdefault.html\"")
  File.open(ruta, "w") {|file| file.puts reemplazo6}
end
```

Figura 63. Código para modificar rutas CSS, JS y HTML. Iteración 8

Finalmente, una vez que la función recorrido_html termina, ya se han modificado entonces todos los enlaces de los html que hacen referencia a imágenes o a otro html, para que correspondan a su nueva ubicación en la carpeta *public* de la aplicación de Acceso y Despliegue del prototipo, y específicamente a la carpeta de la versión del sitio seleccionada por el usuario; de este modo es renderizado el sitio desde dicha carpeta y el usuario puede navegarlo como si del sitio original se tratara.

- **Pruebas:** A continuación las pruebas de esta iteración.

Tabla 33. Caso de prueba: 18. Iteración 8

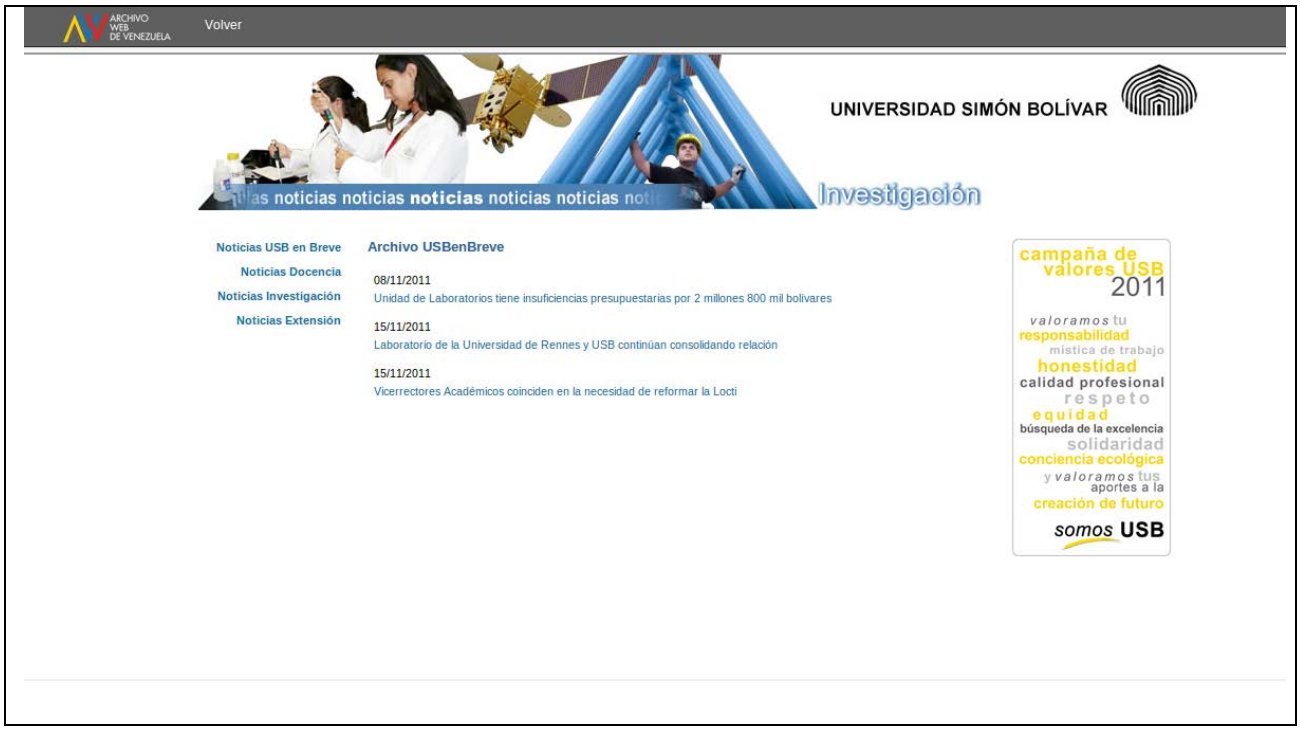
Número de Caso de Prueba: 18	**Número de Historia de Usuario:** 8
Descripción: Al realizar una búsqueda de determinado sitio y seleccionar la versión más reciente que se tenga en el Archivo Web, desplegar el sitio y compararla con la actual (que debe ser prácticamente igual o al menos muy similar)	
Resultado: Sitio www.usb.ve en línea del día 4 de diciembre de 2013. 	

Versión del sitio www.usb.ve desplegado del Archivo Web del día 24 de octubre de 2013.

Es posible ver la similitud entre el despliegue de la versión y el sitio que se encuentra en línea actualmente. Las fechas son de 34 días de diferencia, por lo que no hay muchos cambios y se puede apreciar que el despliegue se está realizando de forma correcta.

Tabla 34. Caso de prueba: 19. Iteración 8

Número de Caso de Prueba: 19	**Número de Historia de Usuario:** 8
Descripción: Al realizar una búsqueda de determinado sitio y seleccionar la versión más reciente que se tenga en el Archivo Web, desplegar el sitio y navegar siguiendo varios enlaces, para verificar que estos funcionen. Además de esto, verificar que se muestran las imágenes del sitio. Con esta prueba se verifica que los enlaces se estén modificando correctamente de acuerdo a la nueva ruta del sitio en la carpeta *public.*	
Resultado: 	



3.3.10. Iteración 9

- **Planificación:** En esta iteración el objetivo es desarrollar una actualización de la interfaz gráfica de la aplicación que se llevó a cabo en la Iteración 5 en busca de mejorar la experiencia del usuario al usar la aplicación.

Tabla 35. Planificación Iteración 9

Número de Iteración: 9	**Número de Historia:** 1 y 2
Fecha de Inicio: 9/12/2013	**Fecha de Fin:** 10/12/2013
Descripción: Actualización de la interfaz gráfica	**Tipo:** Desarrollo / Diseño

- **Diseño:** Para esta actualización de la interfaz gráfica se realiza un diagrama con los tres pasos principales para utilizar el Archivo Web de Venezuela, de modo de colocarlo en el inicio de la aplicación para facilitarle al usuario el uso del mismo. Dicho diagrama se muestra a continuación:

Figura 64. Interfaz de ayuda a usuario. Iteración 9

- **Codificación:** Para esta actualización se modifica la vista **index** del controlador **home**, donde se le añade el diagrama y en pocas palabras la descripción de los pasos.

```
.row{:style=>"padding-left:70px"}
  .span2
  .span3
    =image_tag ("p1.png"), :width => '300px', :alt => "Archivo Web de
      Venezuela"
  .span3
    .h5 Introduce el URL
    .input
      = form_tag({:action=> "analizar_url", :controller => "home"}) do
        .input-append
          =text_field :URL, :url, :id => "appendedInputButtons span2",
            value => ""
          = submit_tag "Buscar", :class => "btn btn-primary"
    .span3
.clear
.row{:style=>"padding-left:70px"}
  .span2
  .span2
    =image_tag ("foto1.png"), :width => '180px', :alt => "Archivo Web
      de Venezuela", :class => "img-polaroid"
    .textom
      1. Introduce el
      %span.label
        URL
      que deseas buscar, o parte de él
  .span2
    =image_tag ("foto2.png"), :width => '180px', :alt => "Archivo Web
      de Venezuela", :class => "img-polaroid"
    .textom
      2. Selecciona a través del
      %span.label
        calendario
      la versión del sitio que deseas navegar
  .span2
    =image_tag ("foto3.png"), :width => '180px', :alt => "Archivo Web
      de Venezuela", :class => "img-polaroid"
    .textom
      3. ¡Navega tal como en el
      %span.label
        pasado!
```

Figura 65. Código Index HAML. Iteración 9

El resultado de añadir este diagrama en la página de inicio fue el siguiente:

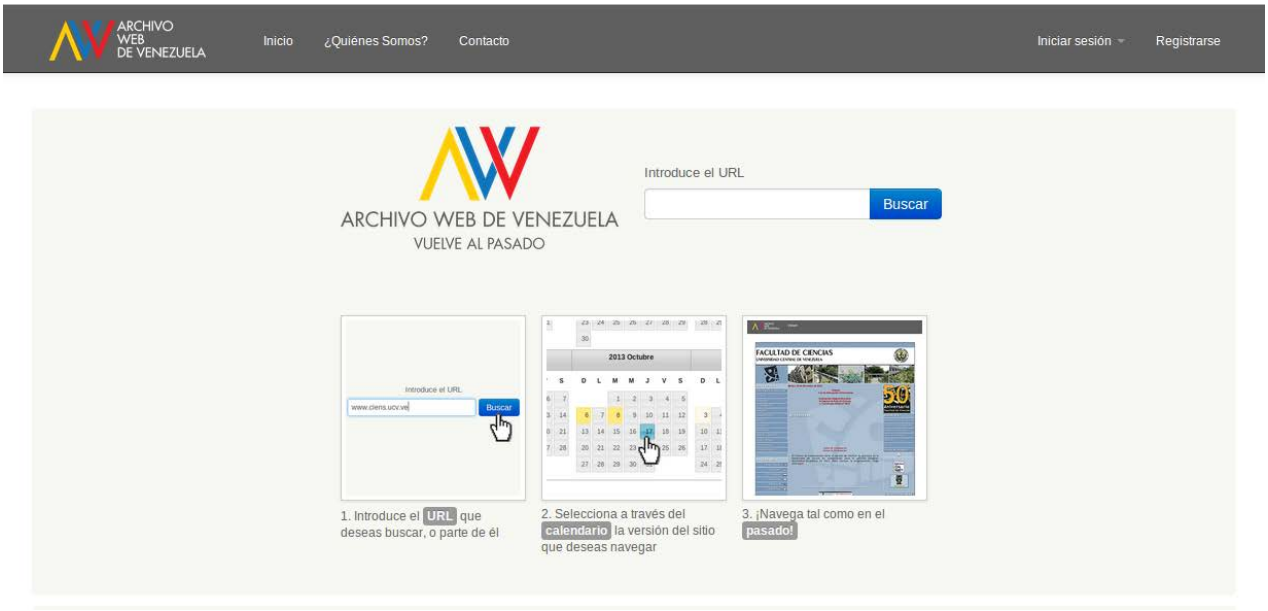


Figura 66. Interfax principal. Iteración 9

- **Pruebas:** En esta iteración no se requirieron pruebas.

3.3.11. Iteración 10

- **Planificación:** En esta iteración el objetivo es crear dos páginas informativas: quiénes somos y Archivo Web, con el fin de brindarle al usuario mayor información sobre el prototipo.

Tabla 36. Planificación Iteración 10

Número de Iteración: 10	**Número de Historia:** 12
Fecha de Inicio: 11/12/2013	**Fecha de Fin:** 12/12/2013
Descripción: Creación de páginas de información del sitio	**Tipo:** Desarrollo / Diseño

- **Diseño:** Para la creación de estas páginas informativas se diagrama la información sobre el equipo del prototipo de Archivo Web de Venezuela, y la información sobre la arquitectura del mismo; para posteriormente crearlas.

- **Codificación:** Para la creación de las dos páginas informativas se crea la vista **quienes_somos.html.haml** y la vista **archivo_web.html.haml**, las cuales se muestran a continuación:

```
.row-fluid
  .span10
    %h3.subr
      > ¿Quiénes Somos?
  .span12
    %h4.subr
      > Grupo de Archivo Web de Venezuela
.row-fluid
  .span12
    %table.table.table-hover
      %table.table{:style => "font-size:15px"}
        %thead
          %h3
            Líderes de proyecto
          %tr
          %tbody
            %tr
              %td
                =image_tag ("female.png"), :width => '150', :class=>"img-circle"
              %td
                %p
                  Profesora Claudia León
                  %h5{:style => "font-size:12px"}
                    %br
                    %b
                      Profesión:
                    Licenciada en Computación
                    %br
                    %b
                      Nivel Académico:
                    Doctor en Computación
                    %br
                    %b
                      Correo Electrónico:
                    claudia.leon@ciens.ucv.ve, claudia.leon@gmail.com
                    %br
                    %b
                      Institución:
                    Universidad Central de Venezuela. Escuela de Computación
                    %br
                    %b
                      Grupo de Investigación:
                    Centro de Computación Paralela y Distribuida

              %td
                =image_tag ("female.png"), :width => '150', :class=>"img-circle"
              %td
```

Figura 67. Código Quienes somos. Iteración 10

```
.row-fluid
  .span10
    %h3.subr
      > Archivo Web de Venezuela
  .span10
    %p{:style=>"line-height:24px; font-size:14px"}
      El prototipo de Archivo Web de Venezuela está enmarcado dentro del
      proyecto “Desarrollo de un prototipo de Archivo Web para la preservación
      del patrimonio Web de Venezuela” de código PI-03-8139-2011/P financiado
      por el Consejo de Desarrollo Científico y Humanístico (CDCH) de la
      Universidad Central de Venezuela.

      El prototipo de Archivo Web de Venezuela posee gran relevancia en el
      ámbito social ya que representa una fuente de conocimientos amplia, para
      las presentes y próximas generaciones, que en un futuro podría perderse
      si no surgen inquietudes hacia esta clase de iniciativas. Aunado a
      estpo, en Latinoamérica, y más específicamente en Venezuela, no se
      conocen iniciativas oficiales de preservación Web.


.row-fluid
  .span12
    %h3.subr
      > Estructura del archivo
  .span10
    =image_tag ("estructura2.png"), :class=>"img-polaroid"
```

Figura 68. Código Interfaz Archivo Web. Iteración 10

El resultado de añadir ambas páginas es el siguiente:

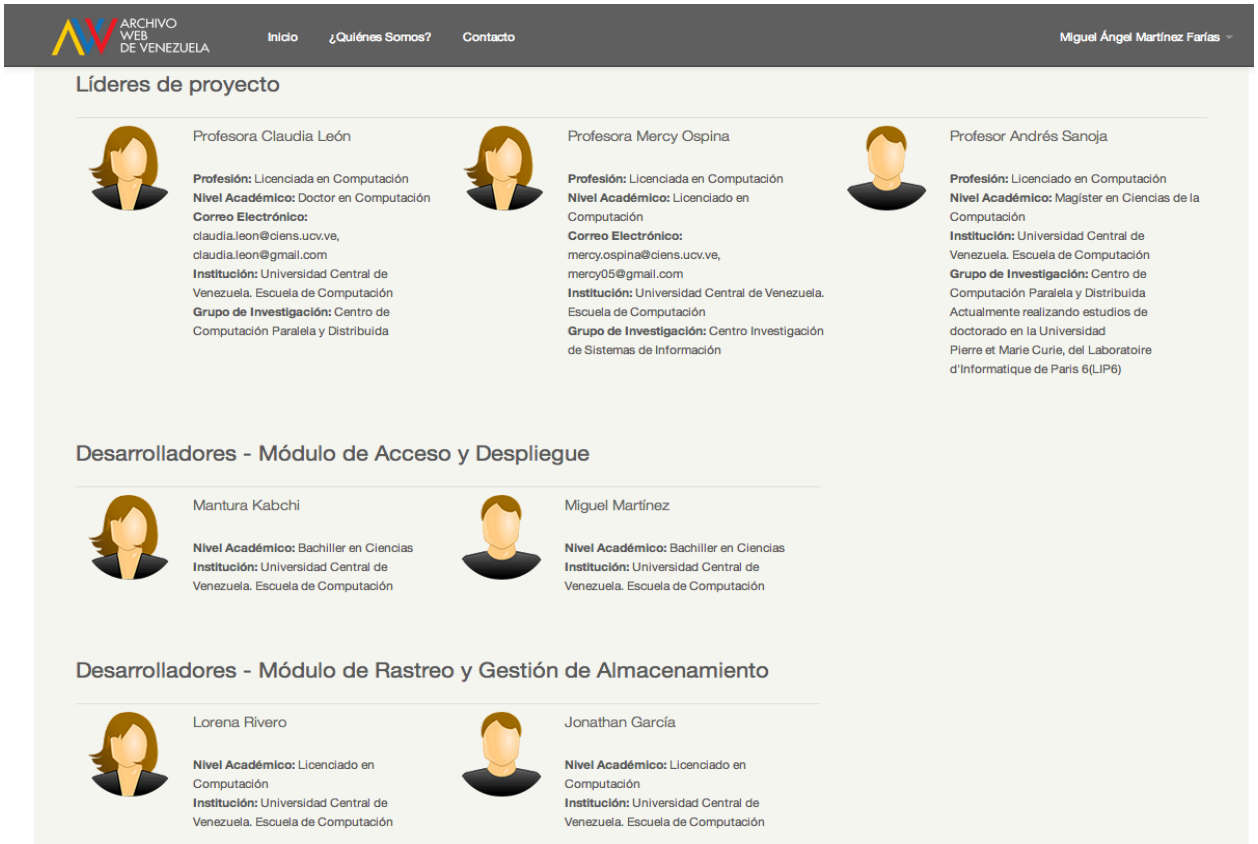


Figura 69. Interfaz Quienes Somos. Iteración 10

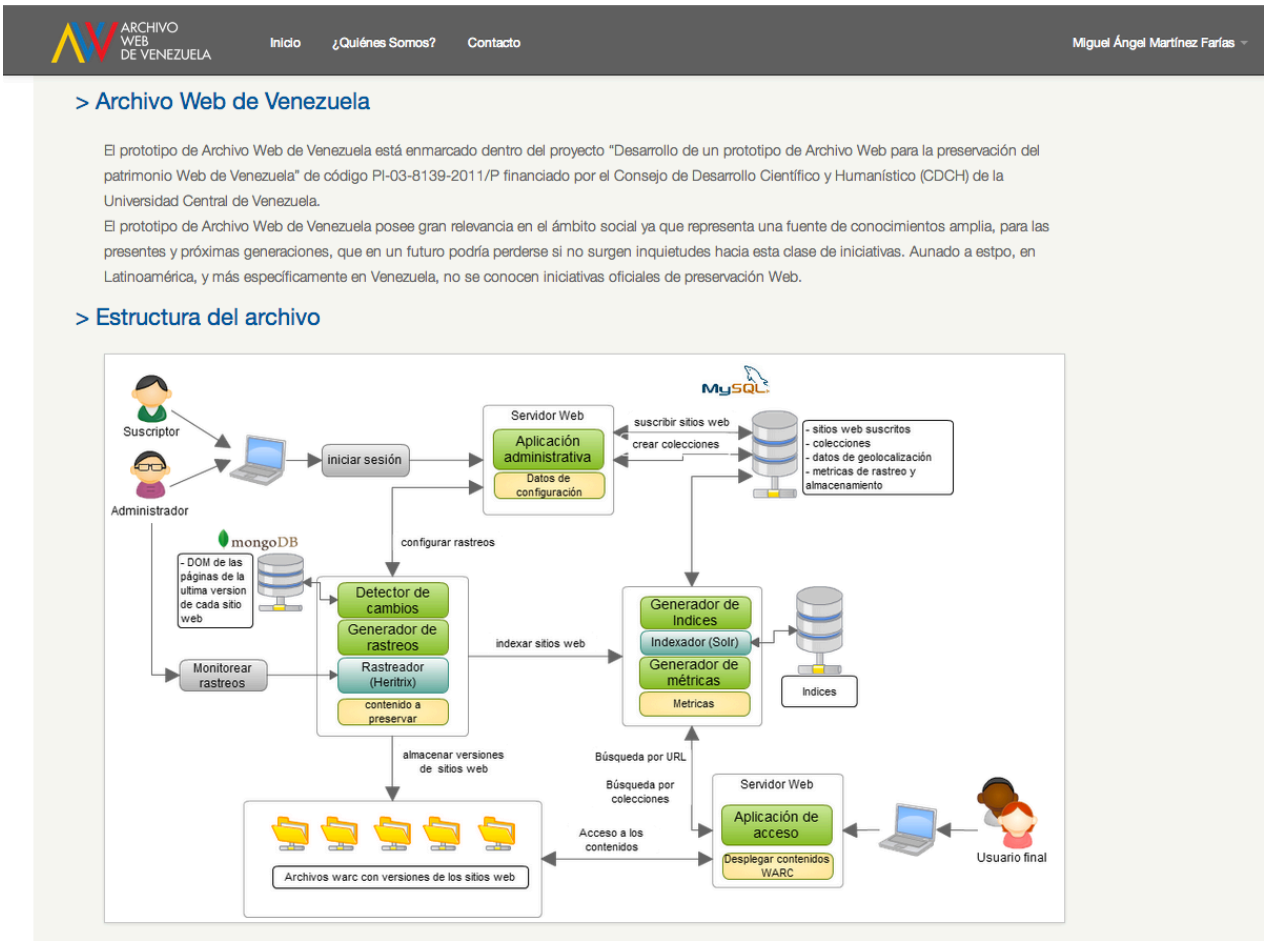


Figura 70. Interfaz Archivo Web. Iteración 10

- **Pruebas:** En esta iteración no se requirieron pruebas.

3.3.12. Iteración 11

- **Planificación:** En esta iteración un formulario de contacto para brindar un medio de comunicación entre los usuarios y el administrador del prototipo.

Tabla 37. Planificación Iteración 11

Número de Iteración: 11	**Número de Historia:** 13
Fecha de Inicio: 9/01/2014	**Fecha de Fin:** 10/01/2014
Descripción: Creación de formulario de contacto del sitio	**Tipo:** Desarrollo / Diseño

- **Diseño:** Para la creación de estas páginas informativas se diagrama el formulario de contacto del prototipo de Archivo Web de Venezuela.

- **Codificación:** Para la creación del formulario de contacto se crea la vista **contacto.html.haml** donde se encuentran los campos nombre, correo y mensaje. Si el usuario se encuentra autenticado, el campo nombre y correo están deshabilitados pues corresponden a su sesión de usuario. Posteriormente se configuran los datos del servidor de correo que se está utilizando para enviar los correos electrónicos, agregándolos al final del archivo **development.rb**, ubicado en la ruta dentro del proyecto **/config/environments**:

```
ActionMailer::Base.smtp_settings = {
:address              => "stryx.ciens.ucv.ve",
:port                 => 465,
:domain               => "ciens.ucv.ve",
:user_name            => "archivo.webve",
:password             => "suscr1pc10n",
:authentication       => "plain",
:enable_starttls_auto => true
}
```

Figura 71. Código para configurar correo. Iteración 11

Una vez configurado el servidor de correos en el proyecto, se crea la función encargada de la gestión de los correos electrónicos en la clase **CorreosUsuarios** ubicado en la ruta del proyecto **/mailers/**, donde se define la función **correo_contacto** que recibe los datos de **home_controller** quien llama a dicha función:

```
def correo_contacto(nombre_usr, correo_usr, mensaje_usr, fecha_usr)
  @usuario = nombre_usr
  @correo = correo_usr
  @mensaje = mensaje_usr
  @fecha = fecha_usr
  mail(:to => "archivo.webve@ciens.ucv.ve", :subject => "Mensaje de Contacto")
end
```

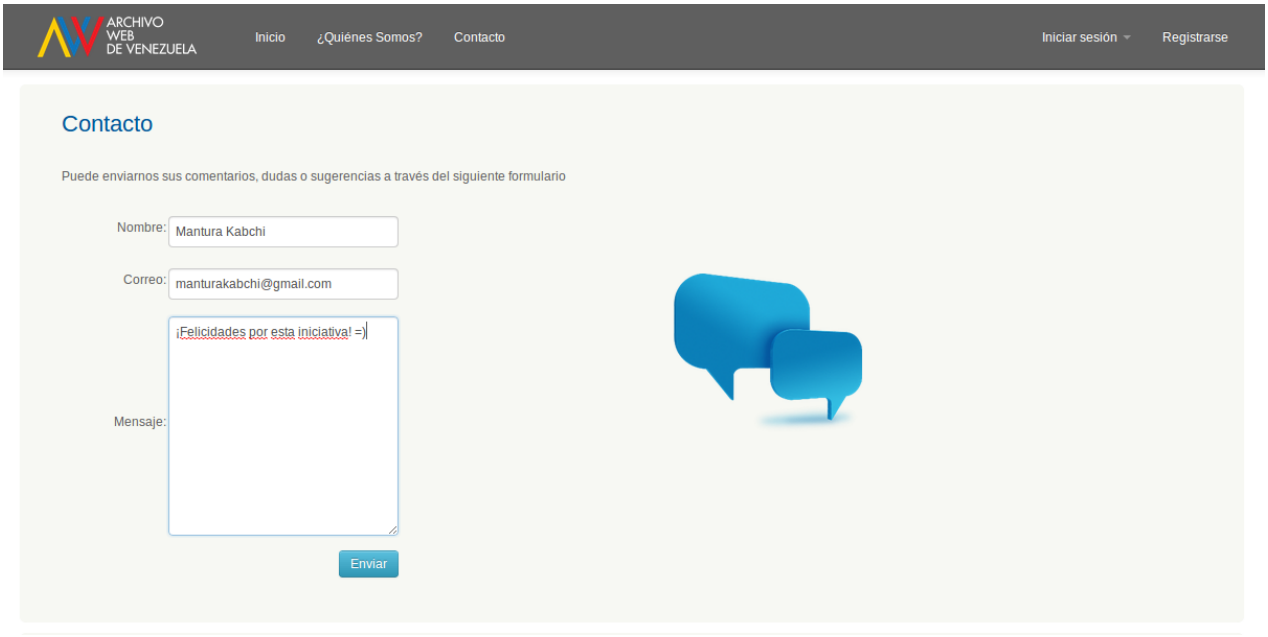
Figura 72. Código definición de correo. Iteración 11

- **Pruebas:** A continuación las pruebas de esta iteración.

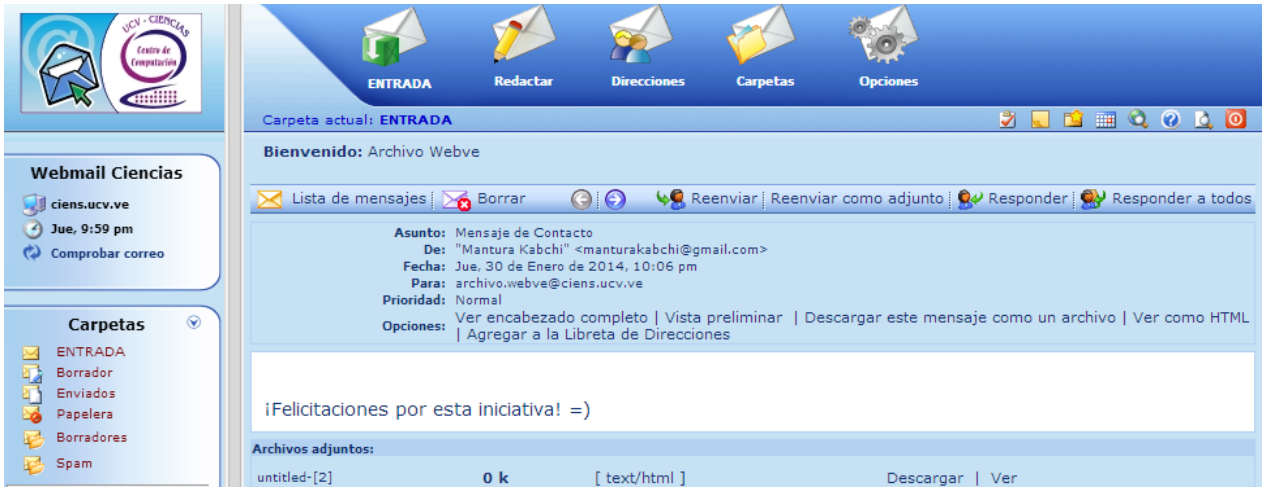
Tabla 38. Caso de prueba: 20. Iteración 11

Número de Caso de Prueba: 20	**Número de Historia de Usuario:** 13
Descripción: Enviar un mensaje de contacto a través del formulario de contacto	
Resultado:	

Formulario de contacto



Correo recibido



3.3.13. Iteración 12

- **Planificación:** esta iteración tiene como objetivo independizar el desempaquetado, la extracción y manipulación de los enlaces y nombres de archivos de cada uno de los WARC de la aplicación Web, para que este proceso se realice una vez que han sido rastreados los sitios y no cuando el usuario solicita el despliegue como se realizó en iteraciones anteriores; de modo que el tiempo de espera del mismo al desplegar un sitio sea notablemente reducido. El componente a desarrollar en esta iteración permite, de acuerdo a la arquitectura propuesta por el IIPC, la transformación de los AIP (Archive Information Package), es decir los archivos WARC, en la información que se le otorga a los usuarios finales que es el DIP (Delivery Information Package), es decir el archivo WARC manipulado para que sea navegable por él.

Tabla 39. Planificación Iteración 12

Número de Iteración: 12	**Número de Historia:** 8
Fecha de Inicio: 10/01/2014	**Fecha de Fin:** 22/01/2014
Descripción: Independizar de la aplicación Web la transformación de AIP a DIP mediante la creación de un componente que realice esto.	**Tipo:** Desarrollo

- **Diseño:** Para la creación de este componente se migrará el controlador que se encarga de desempaquetar el WARC, extraer de él los archivos que conforman el sitio, y manipular estos archivos modificando los enlaces y nombres de archivos para que el sitio pueda ser navegable por el usuario, a un script independiente de la aplicación Web. Este script se ejecutará en segundo plano, dos veces por día. De este modo cuando el usuario solicite algún despliegue, no hay que procesarlo en ese instante, sino buscarlo y renderizarlo; disminuyendo el tiempo de espera del usuario. A continuación el diagrama de componentes del Módulo de Acceso a los Contenidos Preservados en formato WARC para el Prototipo de Archivo Web de Venezuela:

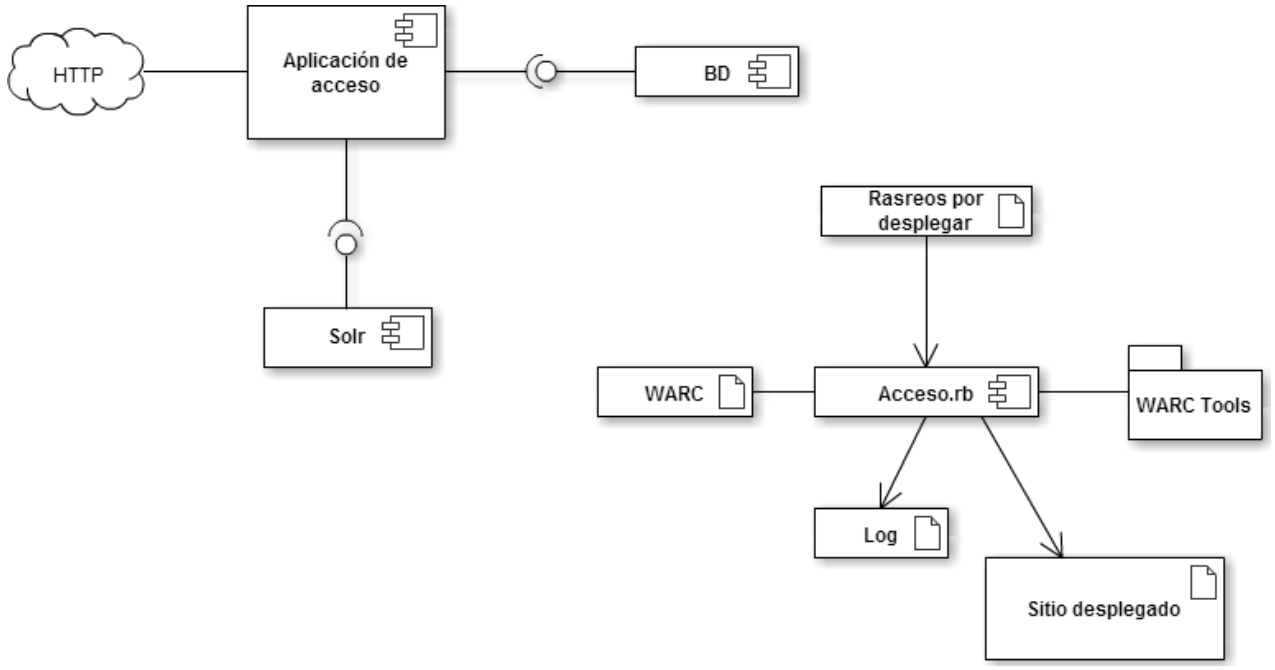


Figura 73. Diagrama de componentes. Iteración 12

Componentes:

- **Aplicación de acceso:**

  Para el componente Aplicación de acceso, se utiliza el protocolo http para que los usuarios realicen búsquedas de contenido por URL o por colección. Utiliza un mapeador objeto relacional para acceder a la base de datos appacceso. Solicita mediante los servicios REST realiza solicitudes a Solr para obtener lista de colecciones, URL por colección, versiones asociadas a un URL. Este componente recibe por parte del usuario la colección seleccionada (para búsquedas por colección) o el URL de interés (para búsquedas por URL), y la versión del sitio que desea navegar. Este componente despliega la versión del sitio que el usuario seleccionó para que él navegue en ella. Permite al usuario realizar búsquedas (por URL y por colecciones), seleccionar la versión que desea navegar del sitio que buscó, desplegar el sitio y navegarlo, ver estadísticas del sitio buscado (cantidad de versiones por año).

- **Acceso.rb:**

  Este componente es un script, que utiliza la lista de rastreos (WARC) para desplegar. Genera una carpeta por WARC con el contenido procesado para navegar, y actualiza la lista de rastreos para desplegar. Este componente verifica periódicamente la lista de rastreos por desplegar, los cuales descomprime,

desempaqueta su contenido en una carpeta pública usando la librería WARC Tools y procesa los documentos html modificando enlaces y extensiones para que pueda ser navegable, y deja el repositorio de archivos WARC en su estado original.

En este diagrama de componentes podemos observar que en esta iteración se añade un componente más, además de la aplicación de acceso, Acceso.rb, el cual en iteraciones anteriores era parte de un controlador de la aplicación. Este nuevo componente ahora es un script que se ejecuta de forma independiente a la aplicación.

- **Codificación:** Para esta iteración, se toma el controlador **despliegue_controller** de la Iteración 8, y sus funciones se colocan en un archivo independiente llamado **accesoruby.rb**, el cual es llamado por el script **Despliegue.sh**, q a través de CRON del servidor donde se encuentra el prototipo, es ejecutado dos veces al día.

Este script recorre todas las máquinas virtuales del servidor del prototipo, y lee el archivo llamado con la IP de la máquina y la extensión txt. Toma la primera línea que contiene una ruta por procesar, de dicha ruta se obtienen los parámetros necesarios para hacer el llamado a la función **buscar** que está en el **accesoruby.rb**, y se procede a llamar a dicha función. Además, se crea un log por día, para almacenar el comportamiento del script.

```
#!/bin/bash
source /home/acceso/.profile
cd /home/acceso

echo "Inicio de despliegue"
maquinas[0]='190.169.69.155'
maquinas[1]='190.169.69.156'
maquinas[2]='190.169.69.157'
maquinas[3]='190.169.69.158'

for i in "${maquinas[@]}"
do
    FILENAME=$i.txt
    echo "FILENAME $FILENAME"
    ssh root@$i "cat /home/$FILENAME" | while read LINE
    do
        maquina=$i
        RUTA=$LINE
        nombreWarc=`ssh -n root@$maquina ls $RUTA/warcs`
        fechatmp=`echo $LINE | awk -F/ '{print $NF}'`
        fecha=`echo $fechatmp | cut -c 5-6`/`echo $fechatmp
               | cut -c 7-8`/`echo $fechatmp |cut -c 1-4`
        hora=`echo $fechatmp | echo $fechatmp |
              cut -c 9-10`:`echo $fechatmp |
              cut -c 11-12`:`echo $fechatmp | cut -c 13-14`
        warcid=`ssh -n root@$maquina sed -n '2p'
                $LINE/reports/seeds-report.txt |
                cut -d ' ' -f 3|  cut -d '/' -f 3`_$fechatmp
        URL=`ssh -n root@$maquina sed -n '2p'
             $LINE/reports/seeds-report.txt | cut -d ' ' -f 3
             |  cut -d '/' -f 3`
```

```
33          echo "maquina $maquina" >>
34              logs/Despliegue.$(date +%Y-%m-%d).log
35          echo "RUTA $RUTA" >>
36              logs/Despliegue.$(date +%Y-%m-%d).log
37          echo "nombreWarc $nombreWarc" >>
38              logs/Despliegue.$(date +%Y-%m-%d).log
39          echo "fecha $fecha" >>
40              logs/Despliegue.$(date +%Y-%m-%d).log
41          echo "hora $hora" >>
42              logs/Despliegue.$(date +%Y-%m-%d).log
43          echo "warcid $warcid" >>
44              logs/Despliegue.$(date +%Y-%m-%d).log
45          echo "URL $URL" >>
46              logs/Despliegue.$(date +%Y-%m-%d).log
47
48          ./accesoruby.rb $RUTA $nombreWarc $maquina
49                          $fecha $hora $warcid $URL
50
51          if [ $? -eq 0 ]
52          then
53              echo "**** Despliegue de archivo $warcid EXITOSO ****"
54                  >> logs/Despliegue.$(date +%Y-%m-%d).log
55              `ssh -n root@$maquina "echo $RUTA >> /home/porIndexar/$FILENAME"`
56          else
57              echo "**** Despliegue de archivo $warcid FALLIDO ****"
58                  >> logs/Despliegue.$(date +%Y-%m-%d).log
59          fi
60      done
61  done
```

Figura 74. Código para ejecutar script de acceso. Iteración 12

- **Pruebas:** A continuación las pruebas de esta iteración.

Tabla 40. Caso de prueba: 21. Iteración 12

Número de Caso de Prueba: 21	**Número de Historia de Usuario:** 8
Descripción: Mostrar el log tras correr el script Despliegue.sh	
Resultado:	

```
maquina 190.169.69.155
RUTA /data/heritrix-3.1.1/jobs/usb-2013-09-30-02-11-12/20140124230506
nombreWarc WEB-20140124230507-00000-Heritrix1.warc.gz
fecha 01/24/2014
hora 23:05:06
warcid www.usb.ve_20140124230506
URL www.usb.ve
********** Despliegue de archivo www.usb.ve_20140124230506 EXITOSO **********
maquina 190.169.69.156
RUTA /data/heritrix-3.1.1/jobs/uc-2013-09-30-02-30-35/20131010070202
nombreWarc WEB-20131010070204-00000-localhost.localdomain.warc.gz.open
fecha 10/10/2013
hora 07:02:02
warcid _20131010070202
URL
********** Despliegue de archivo _20131010070202 FALLIDO **********
maquina 190.169.69.156
RUTA /data/heritrix-3.1.1/jobs/ula-2013-09-30-02-28-50/20131013070005
nombreWarc WEB-20131013070007-00000-localhost.localdomain.warc.gz
fecha 10/13/2013
hora 07:00:05
warcid www.ula.ve_20131013070005
URL www.ula.ve
********** Despliegue de archivo www.ula.ve_20131013070005 EXITOSO **********
```

3.3.14. Iteración 13

- **Planificación:** en esta iteración el objetivo es desarrollar una búsqueda de sitios Web preservados a través de una clasificación o "Colección", ya que los archivos WARC pueden clasificarse según su contenido y objetivos, para lo cual es necesario conectarse con el módulo de indexación para obtener la metadata necesaria. De esta manera los usuarios podrán tener un rango de búsqueda mayor.

Tabla 41. Planificación Iteración 13

Número de Iteración: 13	**Número de Historia:** 14
Fecha de Inicio: 23/01/2014	**Fecha de Fin:** 24/01/2014
Descripción: Desarrollo de búsquedas por colección.	**Tipo:** Desarrollo / Diseño

- **Diseño:** para la búsqueda por colecciones el usuario deberá seleccionar el tema de su preferencia de una lista (tecnológico, educativo, científico, entre otros). A partir de esto se mostrará una colección de archivos WARC, en una lista de URL's, de acuerdo a la selección del usuario. El usuario procede a escoger la URL de su preferencia lo cual lo llevará al módulo de calendario de versiones que le permitirá desplegar la versión de su preferencia. A continuación el diagrama de secuencia de la búsqueda por colecciones:

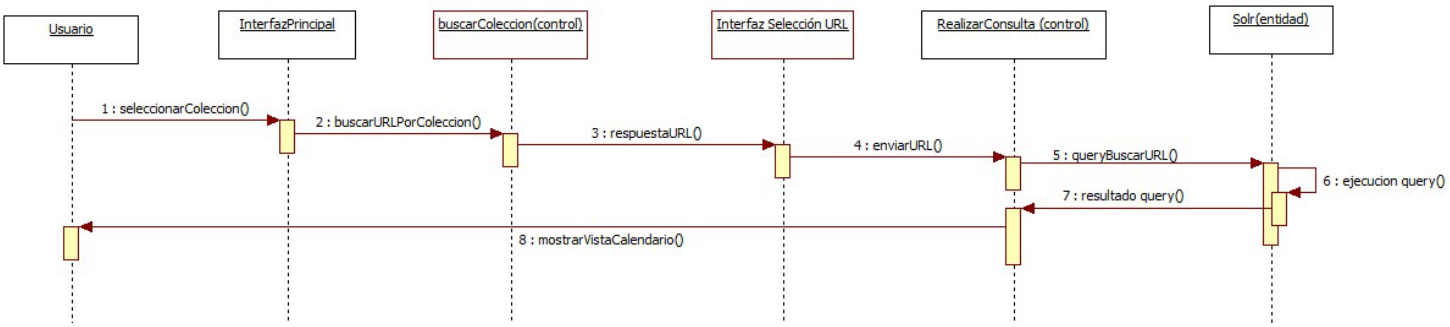


Figura 75. Diagrama de secuencia Búsqueda por colecciones. Iteración 13

- **Codificación:** para el desarrollo de la búsqueda por colecciones se agrega un combo box con las colecciones o clasificaciones de archivos WARC contenidos en el sistema. Para obtener esta lista se solicita al módulo de indexación que envíe los metadatos de los archivos WARC con un campo nuevo llamado colección, la respuesta es procesada como una variable Hash que posee el identificador del WARC y la colección asociada.

```
@arreglo_html = params[:nombre_html]

@query_curl= 'http://190.169.69.154:8080/solr/select/?wt=json&q=coleccion_nombre%3A***&version=2.2&start='+'0'+'&rows=10&indent=on'
response = HTTParty.get(@query_curl)

data= response.body
@json_result = ActiveSupport::JSON.decode(data)
contador=10
while contador < @json_result["response"]["numFound"] do

  @query_curl= 'http://190.169.69.154:8080/solr/select/?wt=json&q=coleccion_nombre%3A*&version=2.2&start='+contador.to_s()+'&rows=10&indent=on'
  response = HTTParty.get(@query_curl)
  data= response.body
  @json_resulttmp = ActiveSupport::JSON.decode(data)

  @json_resulttmp["response"]["docs"].each do |nuevo|
    @json_result["response"]["docs"] << nuevo
  end

  contador+= 10
end

  if @json_result["response"]["numFound"] == 0
    flash[:mensaje] = "Disculpe, no se han almacenado archivos correspondientes a su búsqueda"
    redirect_to :action => "index"
    return
  end
  #----------------
  @hash = Hash.new
  @json_result["response"]["docs"].each do |campos|
      @hash["#{campos["coleccion"]}"] = campos["coleccion_nombre"]
  end
  @hash.delete_if{|k,v| v.empty?}
  @hash.sort
```

Figura 76.Código home_controller RB. Iteración 13

Posteriormente se configura el combo box para que tome el valor de la colección como atributo visible y el identificador para obtener los archivos WARC correspondientes.

```
.input
  = form_tag({:action=> "analizar_coleccion", :controller => "home", :hash => @hash}) do
    .input-append
      = collection_select(:field, :field, @hash, :first, :last)
      = submit_tag "Buscar", :class => "btn btn-default btn-mini"
```

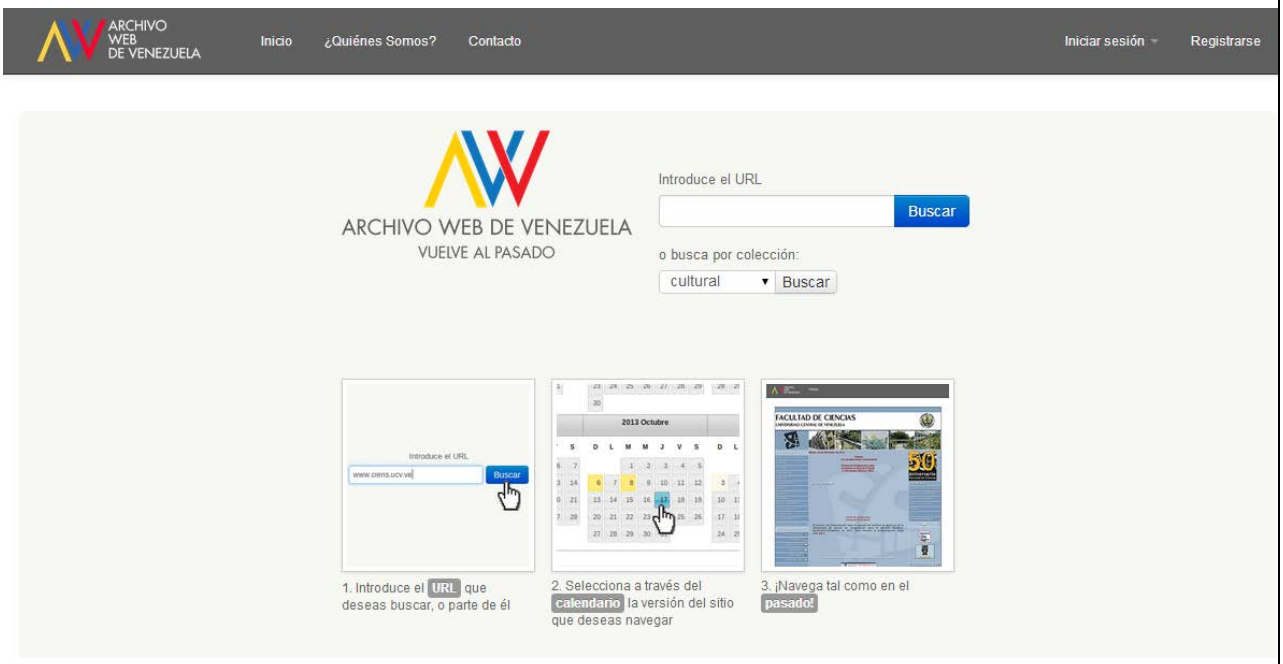
Figura 77. Código index HAML. Iteración 13

- **Pruebas:** A continuación las pruebas de esta iteración.

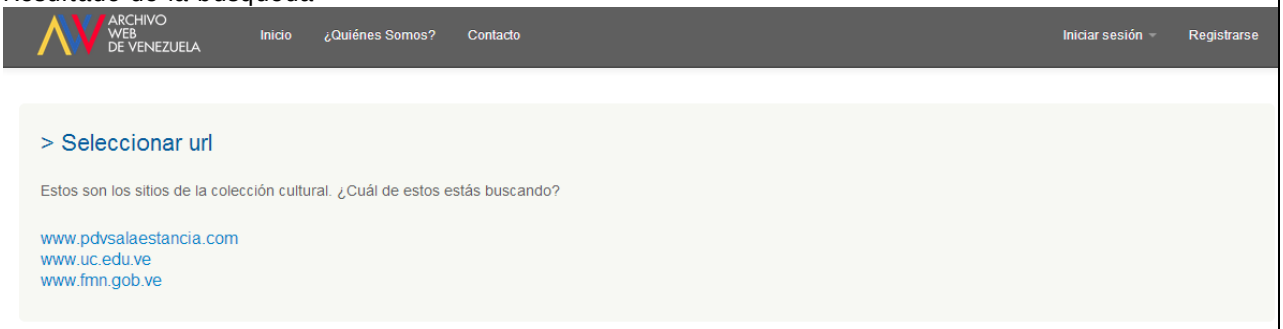
Tabla 42. Caso de prueba: 22. Iteración 13

Número de Caso de Prueba: 22	**Número de Historia de Usuario:** 14
Descripción: Buscar un archivo WARC por colección cultural	
Resultado:	

Campo de búsqueda



Resultado de la búsqueda



3.4. Pruebas de Usabilidad

Para culminar con el desarrollo del Módulo de Acceso a los Contenidos Preservados en formato WARC para el Prototipo de Archivo Web de Venezuela fueron aplicadas pruebas de usabilidad con el objetivo de corroborar que el módulo desarrollado es un producto de software usable, tolerante a fallas y que cumple con el funcionamiento esperado para finalmente ponerlo en producción. Las pruebas de usabilidad consistieron en un cuestionario donde se empleó la escala de Likert (Fernadez Nogales, 2004), en ésta se plantean enunciados, ante los cuales el individuo debe mostrar su acuerdo o desacuerdo, se utilizaron cinco alternativas de respuestas para cada enunciado: totalmente de acuerdo, de acuerdo, ni de acuerdo ni en desacuerdo, en desacuerdo y totalmente en desacuerdo. El cuestionario se realizó mediante un formulario digital hecho a través de Google Drive y fue aplicado a 20 personas que variaban entre profesores y estudiantes de la Escuela de Computación, y personas que no pertenecen al área de Computación.

A continuación el cuestionario utilizado:

Cuestionario - Archivo Web de Venezuela

Trabajo Especial de Grado: "Desarrollo del Módulo de Acceso a los Contenidos Preservados en formato WARC para el Prototipo de Archivo Web de Venezuela"

Responda las siguientes preguntas de acuerdo a su experiencia con la aplicación

Ingresar en http://190.169.69.154/

Los URL de los sitios y las versiones que se pueden consultar por el momento son:

URL	Año	Fecha
www.pdvsalaestancia.com	2013	13 Octubre
www.rena.edu.ve	2014	11 Enero
www.ula.ve	2013	13 Octubre
www.usb.ve	2013	15 Diciembre
www.fmn.gob.ve	2013	04 Octubre
www.ing.ucv.ve	2014	02 Enero

* Required

Preguntas sobre la Interfaz

Con respecto a la interfaz gráfica *

	Totalmente en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Totalmente de acuerdo
Es fácil comprender las acciones que se pueden realizar	○	○	○	○	○
Es fácil de aprender (en poco tiempo se conoce las funciones)	○	○	○	○	○
Es fácil de usar (las acciones tienen bajo nivel de complejidad)	○	○	○	○	○
Los colores son agradables	○	○	○	○	○
Es consistente en todas las páginas (estructura, disposición de imagenes, entre otros)	○	○	○	○	○
La experiencia con la interfaz fue positiva	○	○	○	○	○

Observaciones con respecto a la interfaz

Con respecto a la información que provee la interfaz *

	Totalmente en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Totalmente de acuerdo
Es comprensible	○	○	○	○	○
Es útil para lograr llevar a cabo un objetivo	○	○	○	○	○
Las funciones del menú son claras	○	○	○	○	○
Los mensajes aportan significado	○	○	○	○	○

Observaciones sobre la información de la interfaz

El logotipo de la aplicación se adecúa al tema de la misma *

	1	2	3	4	5	
Totalmente en desacuerdo	○	○	○	○	○	Totalmente de acuerdo

Preguntas de opciones para el usuario

En cuanto al registro de usuario en el sistema *

	Totalmente en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Totalmente de acuerdo
Sencillo de llevar a cabo	○	○	○	○	○
El sistema ayudo en caso de errores	○	○	○	○	○
El inicio de sesión se hace de manera fácil	○	○	○	○	○

En cuanto a la sección de favoritos *

	Totalmente en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Totalmente de acuerdo
Es fácil de administrar	○	○	○	○	○
Facilita las busquedas	○	○	○	○	○
Guarda con éxito mis selecciones	○	○	○	○	○

En cuanto al historial de búsqueda *

	Totalmente en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Totalmente de acuerdo
Es fácil de administrar	○	○	○	○	○
Facilita las busquedas	○	○	○	○	○
Es útil	○	○	○	○	○

Funciones de la aplicación

Con relación a las búsquedas *

	Totalmente en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Totalmente de acuerdo
La búsqueda por URL responde adecuadamente	○	○	○	○	○
Da asistencia en caso de errores	○	○	○	○	○
La búsqueda por colecciones es clara	○	○	○	○	○
Las búsquedas dan resultados esperados	○	○	○	○	○
El calendario de versiones es sencillo de entender	○	○	○	○	○

Mensaje que se muestra al seleccionar una version *

Indique cual mensaje le parece mas apropiado

- ○ Vuelve al pasado
- ○ Regresando en el tiempo
- ○ Conoce tu pasado
- ○ Conoce tu historia
- ○ Ninguno de estos

Proponer un mensaje

Con relación a los sitios web desplegados *

	Totalmente en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Totalmente de acuerdo
Los enlaces funcionan como esperaba	○	○	○	○	○
La navegación por el sitio ha sido satifactorio	○	○	○	○	○
La información estadística le parece útil	○	○	○	○	○
Las búsquedas dan resultados esperados	○	○	○	○	○

Observaciones con respecto a los sitios web desplegados

La aplicación me ha permitido *

	Totalmente en desacuerdo	En desacuerdo	Ni de acuerdo ni en desacuerdo	De acuerdo	Totalmente de acuerdo
Conocer que es un Archivo Web	○	○	○	○	○
Saber de que trata el Archivo Web de Venezuela	○	○	○	○	○
Ponerme en contacto con los administradores	○	○	○	○	○
Conocer detalles técnicos del sistema	○	○	○	○	○

Observaciones con respecto a la información del sistema

Estaría dispuesto a recomendar esta aplicación *

	1	2	3	4	5	
Totalmente en desacuerdo	○	○	○	○	○	Totalmente de acuerdo

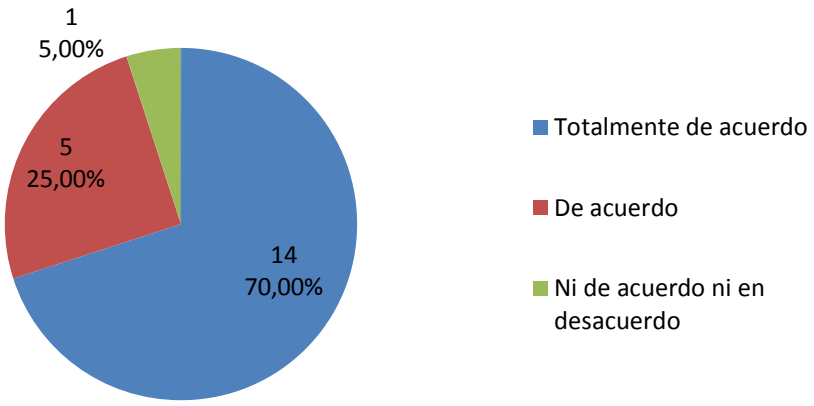
Qué recomendaría para mejorar la experiencia del usuario que utilice el Archivo Web de Venezuela

3.4.1. Resultados de la Prueba de Usabilidad

Del cuestionario presentado anteriormente, se pudieron obtener los siguientes resultados:

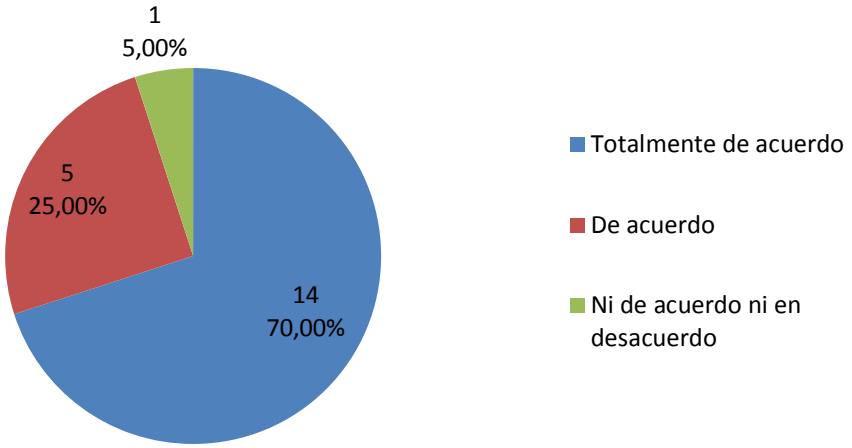
Las primeras seis gráficas corresponden a la interacción del usuario con la Interfaz Gráfica de la aplicación.

La Gráfica 1 muestra que las acciones que se pueden realizar mediante la aplicación son fáciles de comprender ya que el 95% de las personas encuestadas estuvieron entre "Totalmente de acuerdo" (70%) y "De acuerdo" (25%) con esto.

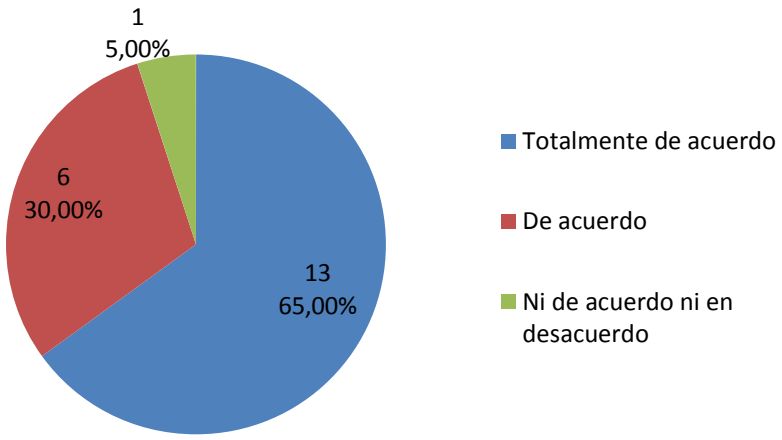


Gráfica 1. Es fácil comprender las acciones que se pueden realizar

La Gráfica 2 y 3 indican que la aplicación es tanto fácil de aprender como fácil de usar; puesto que en ambas el 95% de las respuestas oscilan entre "Totalmente de acuerdo"(70%) y "De acuerdo"(25%).

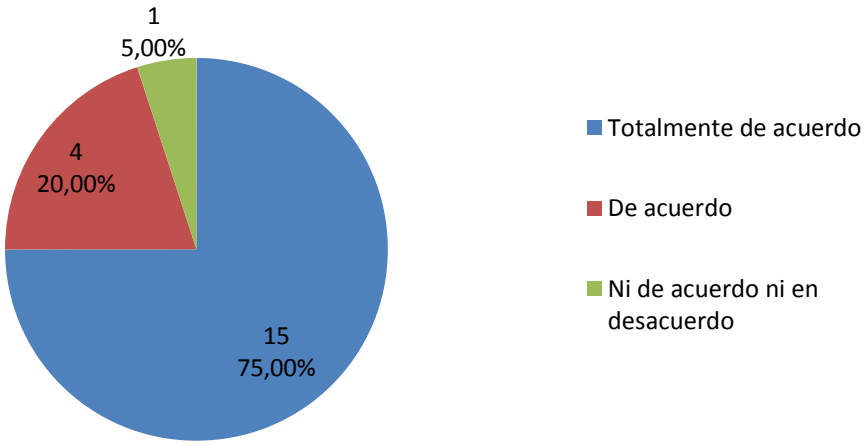


Gráfica 2. Es fácil de aprender (en poco tiempo se conoce las funciones)



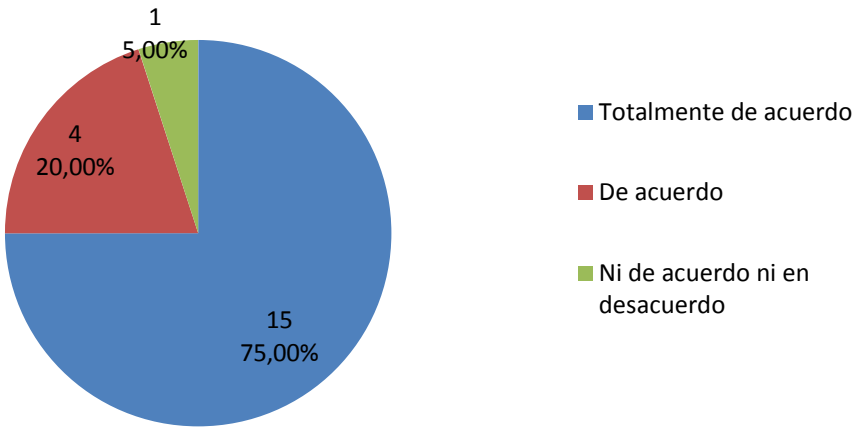
Gráfica 3. Es fácil de usar (las acciones tienen bajo nivel de complejidad)

La Gráfica 4 muestra como la mayoría de las personas concordaron en que los colores de la aplicación les resultan agradables.



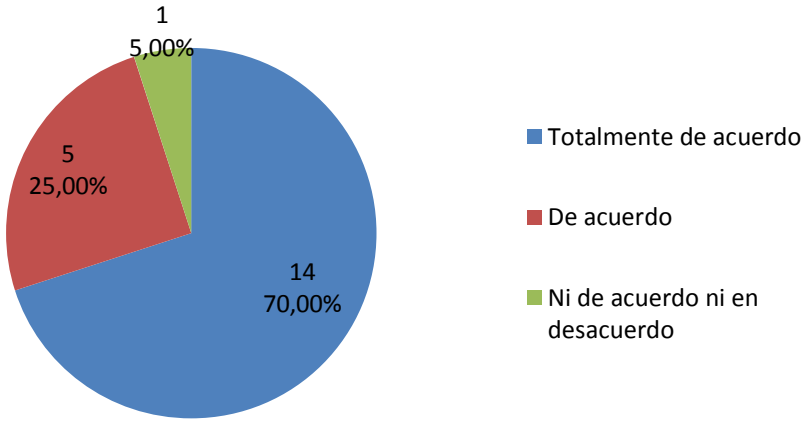
Gráfica 4. Los colores son agradables

La Gráfica 5, soporta el resultado mostrado en las gráficas anteriores con respecto a la interacción del usuario con la aplicación. Esta gráfica evalúa la consistencia en todas las páginas en cuanto a estructura, disposición de imágenes, entre otros. La importancia de este tópico radica en sus beneficios: genera una navegación más fluida, evita comportamientos inesperados, atenúa el riesgo de desorientación, transmite estabilidad, tranquilidad y confianza al utilizar la aplicación. Como se puede observar el 95% de los usuarios encuestados coincidieron que las interfaces son consistentes.



Gráfica 5. Es consistente en todas las páginas (estructura, disposición de imágenes, entre otros)

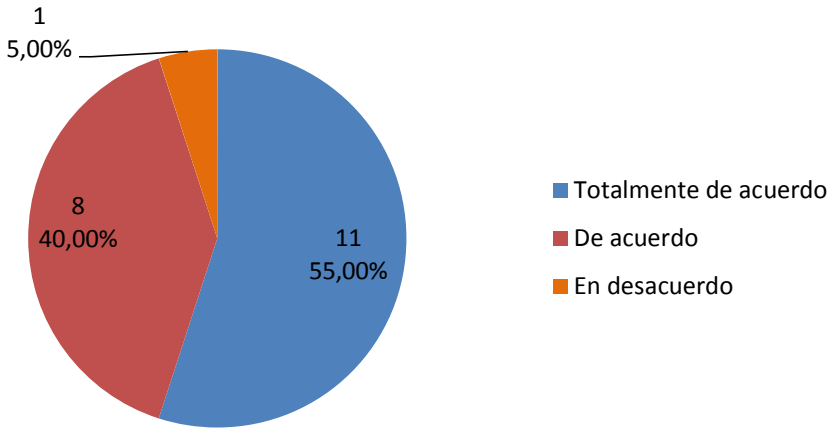
Internet es un medio de comunicación, donde la interfaz tiene un papel fundamental y hace que el producto sea o no competitivo, por lo tanto una Interfaz Gráfica consistente, usable, eficiente, entre otras características, permite atraer a los usuarios a querer utilizar la aplicación. La Gráfica 6 muestra que el 95% de los encuestados tuvieron una experiencia positiva con la interfaz de la aplicación; siendo este resultado consistente con los resultados previamente analizados referentes a la interacción del usuario con la aplicación y con la interfaz gráfica de la misma.



Gráfica 6. La experiencia con la interfaz fue positiva

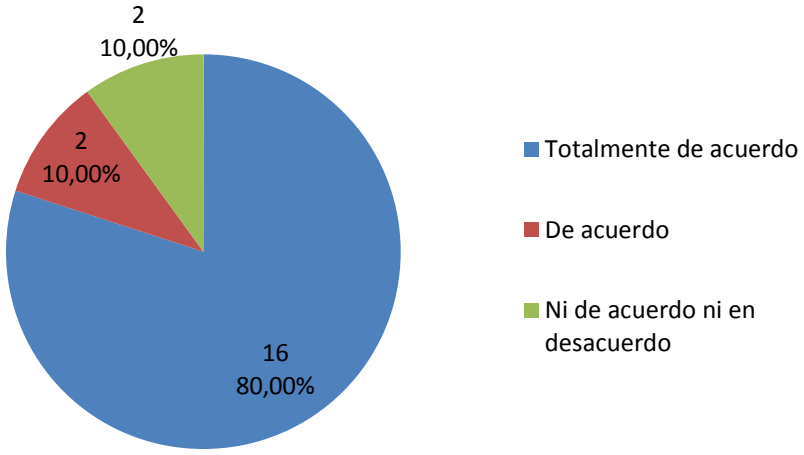
La información que provee la interfaz juega un papel relevante en el logro de los objetivos del usuario dentro de la aplicación, pues mediante ellos se guía al mismo en las diversas operaciones que puede realizar y se le notifica el resultado de sus acciones al interactuar con la aplicación. Las siguientes cuatro gráficas evalúan la efectividad de la información provista a través de la interfaz al usuario.

La Gráfica 7 muestra que la información que provee la interfaz es comprensible para el usuario, puesto que el 55% de los encuestados están totalmente de acuerdo con esto, el 40% estuvo de acuerdo, y sólo un 5% representado por una persona estuvo en desacuerdo.



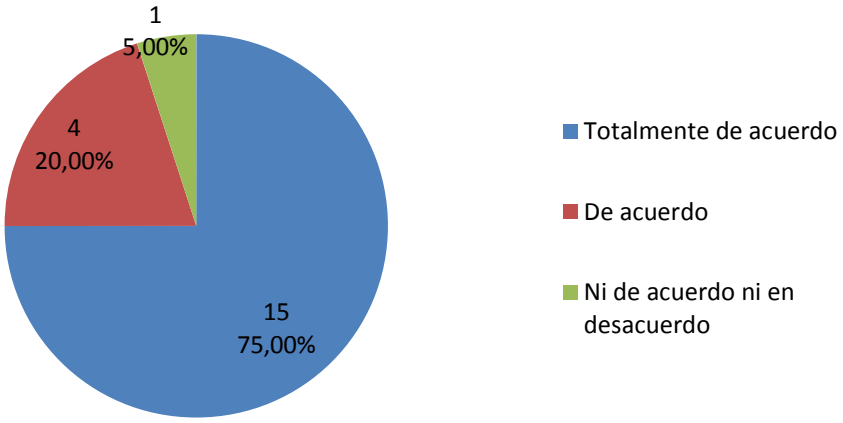
Gráfica 7. Es comprensible

La Gráfica 8 muestra que la información que provee la interfaz le resulta útil a los usuarios para llevar a cabo sus objetivos dentro de la aplicación, ya que el 90% de los encuestados estuvo entre "Totalmente de acuerdo" (80%) y "De acuerdo" (10%) con esto.



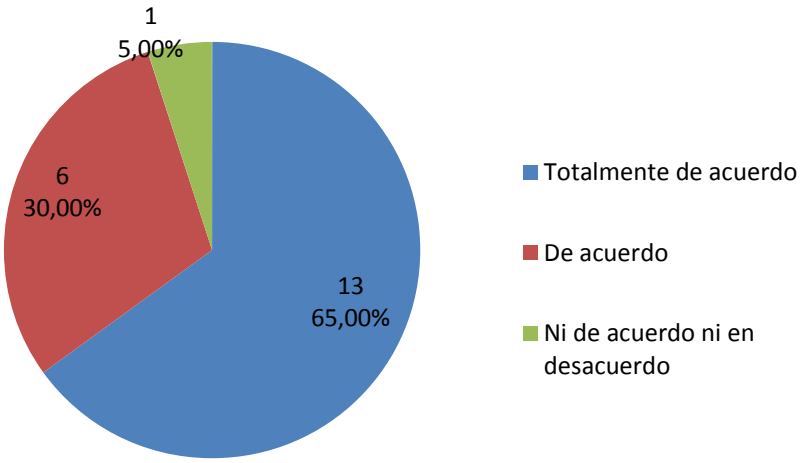
Gráfica 8. Es útil para lograr llevar a cabo un objetivo

La Gráfica 9 muestra que las funciones del menú son claras, ya que el 95% de los encuestados estuvo entre "Totalmente de acuerdo" (75%) y "De acuerdo" (20%) con esto.



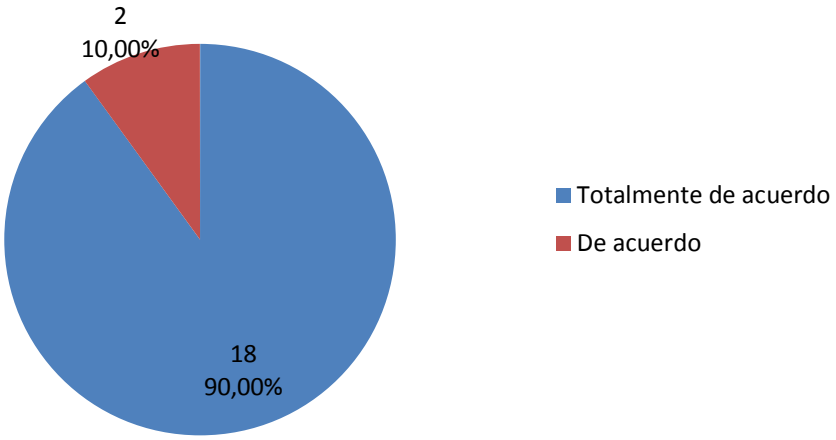
Gráfica 9. Las funciones del menú son claras

La Gráfica 10 muestra que los mensajes que provee la interfaz aportan significado a los usuarios, ya que el 95% de los encuestados estuvo entre "Totalmente de acuerdo" (65%) y "De acuerdo" (30%) con esto.



Gráfica 10. Los mensajes aportan significado

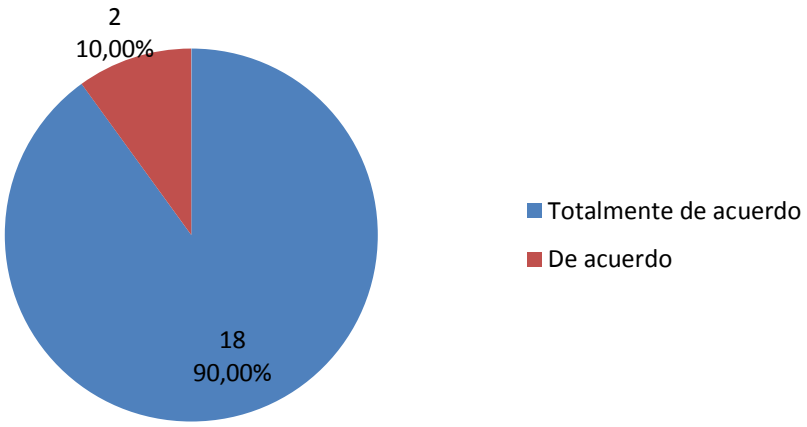
La Gráfica 11, tiene relación con el logotipo de la aplicación, donde se evalúa la adecuación del mismo con el tema del Prototipo de Archivo Web de Venezuela; de acuerdo a las personas encuestas el logotipo logra su objetivo.



Gráfica 11. El logotipo de la aplicación se adecúa al tema de la misma

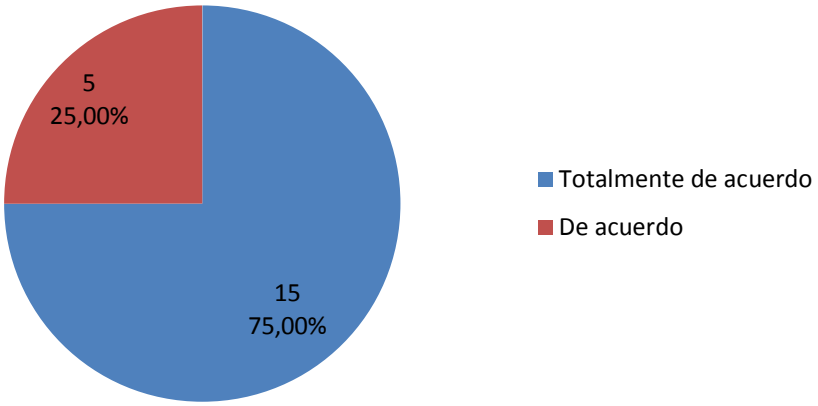
Las siguientes gráficas evalúan los diversos módulos de la aplicación, comenzando con el Módulo de Registro y Autenticación de Usuarios.

De acuerdo a la Gráfica 12, para el 100% de los encuestados fue sencillo llevar a cabo su registro como usuario del sistema, puesto que el 90% estuvo “Totalmente de acuerdo” con esto y el 10% estuvo “De acuerdo”.



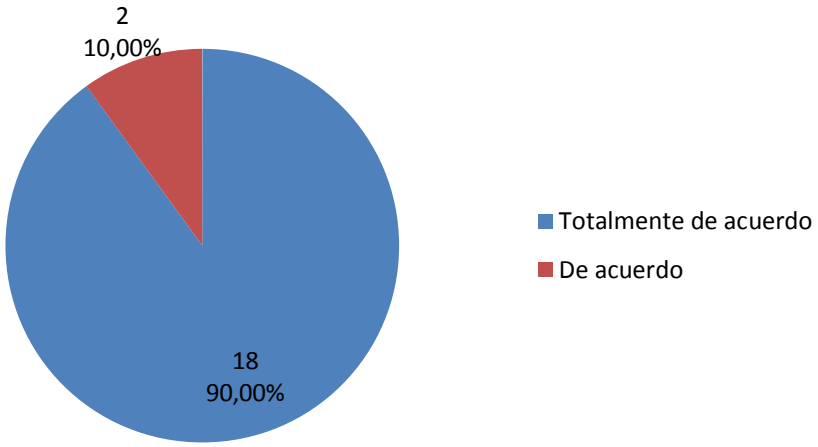
Gráfica 12. Sencillo de llevar a cabo

El manejo de errores dentro de una aplicación es relevante para los usuarios ya que cuando este comete un error es importante que no sienta que ha fracasado y que el sistema lo ayude a corregirlo y superarlo. De acuerdo a la Gráfica 13, el 75% de los usuarios estuvo “Totalmente de acuerdo” con que la aplicación le ayudó cuando cometió un error y el 25% restante estuvo “De acuerdo” con esto.



Gráfica 13. El sistema ayudó en caso de errores

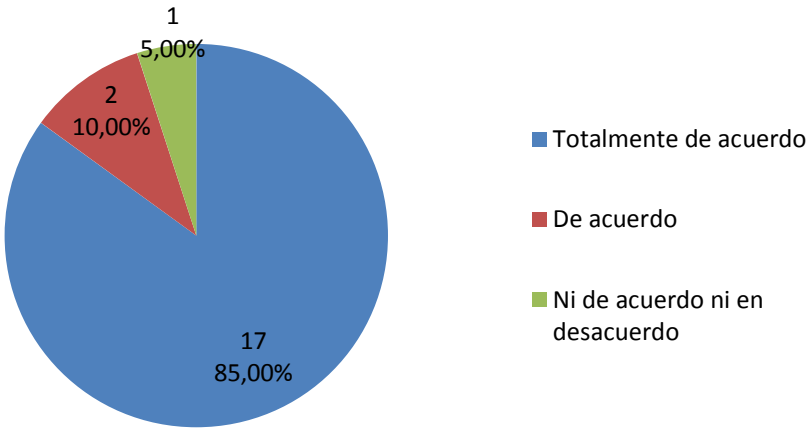
Para el 100% de los encuestados fue fácil autenticarse en el sistema, puesto que el 90% estuvo "Totalmente de acuerdo" con esto y el 10% estuvo "De acuerdo", como es posible observar en la Gráfica 14.



Gráfica 14. El inicio de sesión se hace de manera fácil

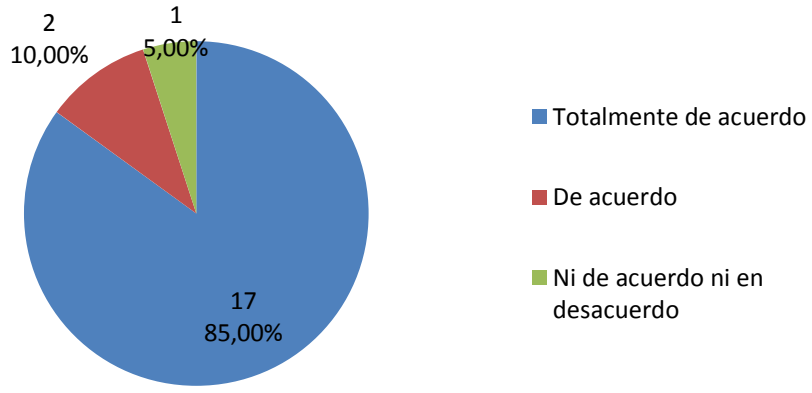
Las siguientes tres gráficas evalúan el Módulo de Favoritos. Este módulo fue pensado como un atajo para aquellos usuarios que requieran visitar un mismo sitio en repetidas ocasiones, de modo que el usuario pueda gestionar sus *Favoritos* a su gusto.

De acuerdo a la Gráfica 15, el 95% de los encuestados encontró fácil de administrar sus *Favoritos*, puesto que el 85% estuvo "Totalmente de acuerdo" con esto, el 10% estuvo "De acuerdo" y sólo el 5% no estuvo "Ni de acuerdo ni en desacuerdo".



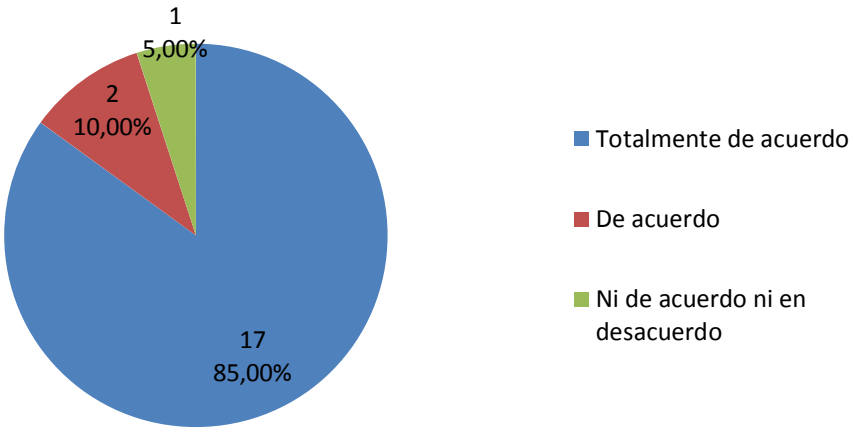
Gráfica 15. Es fácil de administrar

Con respecto a la utilidad del Módulo de Favoritos, se observa que el 95% de los encuestados concuerda en que este módulo le facilita sus búsquedas, puesto que el 85% estuvo "Totalmente de acuerdo" con esto, el 10% estuvo "De acuerdo" y sólo el 5% correspondiente a una sola persona de la población encuestada no estuvo "Ni de acuerdo ni en desacuerdo".



Gráfica 16. Facilita las búsquedas

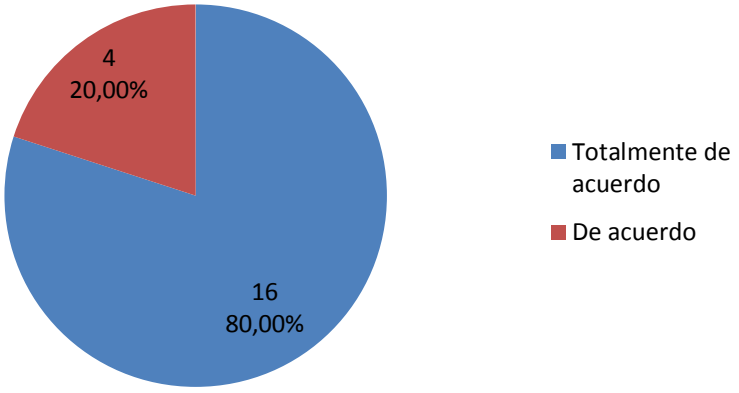
La Gráfica 17 evalúa el éxito al guardar un sitio desplegado como *Favorito*. Con este resultado se observa que la mayoría de los usuarios encuestados logró guardar exitosamente sus sitios favoritos.



Gráfica 17. Guarda con éxito mis selecciones

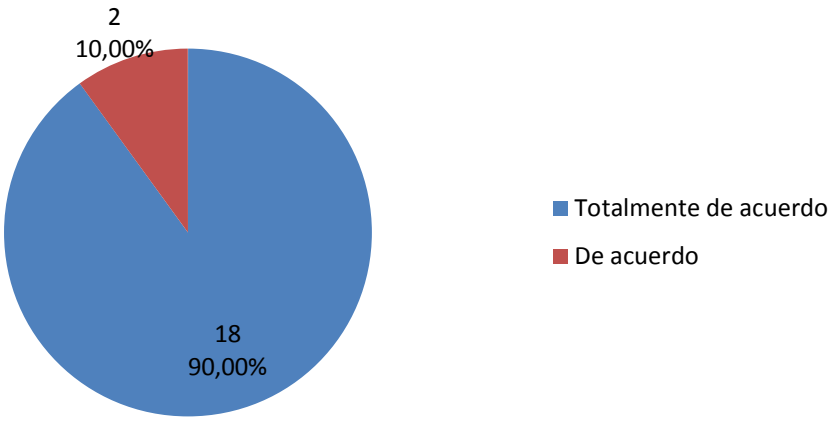
Las siguientes tres gráficas evalúan el Módulo de Historial de Búsqueda. Este módulo fue pensado como un atajo para aquellos usuarios que requieran visitar los últimos sitios consultados por él.

De acuerdo a la Gráfica 18, el 100% de los encuestados encontró fácil de administrar su Historial de Búsqueda, puesto que el 80% estuvo "Totalmente de acuerdo" con esto y el 20% estuvo "De acuerdo", arrojando resultados netamente positivos.



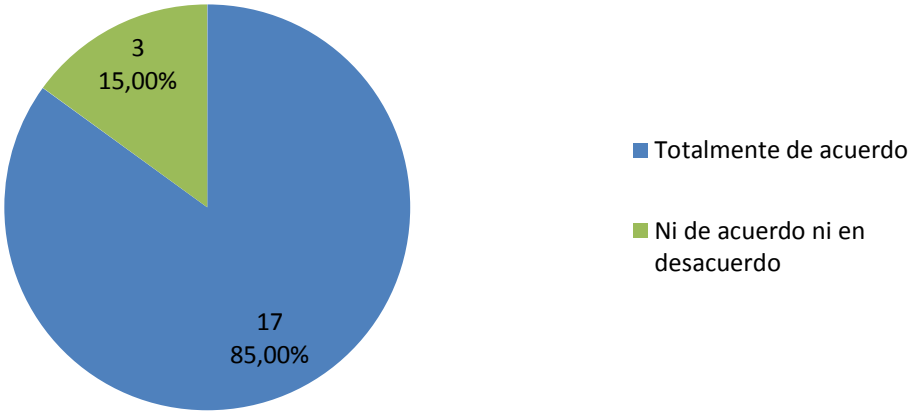
Gráfica 18. Es fácil de administrar

Con respecto a la utilidad del Módulo de Historial de Búsqueda, se observa en la Gráfica 19 que el 100% de los encuestados concuerda en que este módulo le facilita sus búsquedas, puesto que el 90% estuvo "Totalmente de acuerdo" con esto y el 10% estuvo "De acuerdo".



Gráfica 19. Facilita las búsquedas

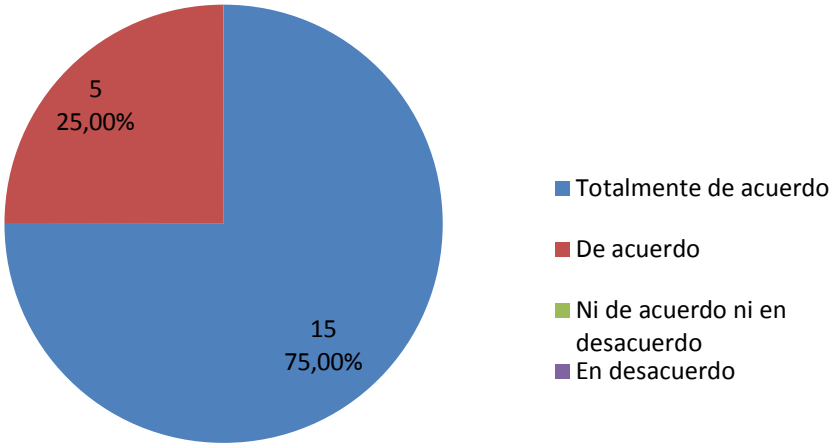
La Gráfica 20 evalúa la utilidad de la posibilidad de almacenar las búsquedas realizadas por el usuario automáticamente. Con este resultado se observa que la mayoría de los usuarios encuestados encontró útil este módulo.



Gráfica 20. Es útil

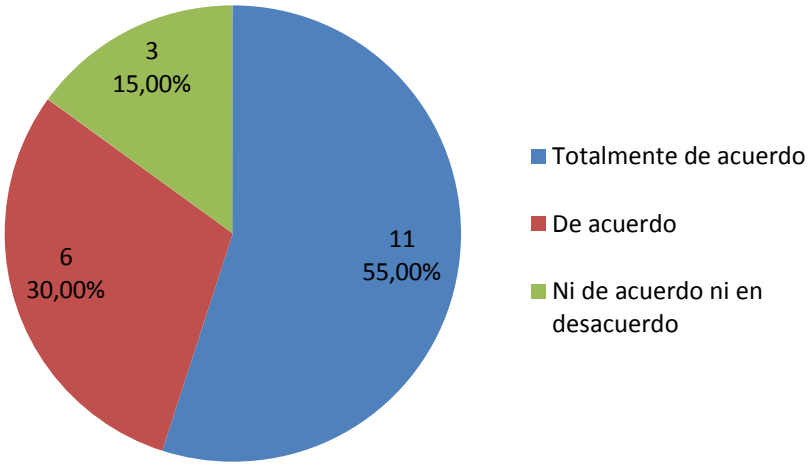
Las siguientes cinco gráficas evalúan el Módulo de Búsqueda. Este módulo es parte del objetivo principal del Prototipo de Archivo Web de Venezuela, puesto que a través de éste el usuario ubica el sitio Web que desea desplegar.

En cuanto a la modalidad de búsqueda basada en URL, la Gráfica 21 muestra como todos los usuarios encuestados perciben que la aplicación responde de forma adecuada a sus búsquedas. Como se puede observar, un 75% de los encuestados estuvieron “Totalmente de acuerdo” con esta premisa, mientras que el restante 25% estuvo “de acuerdo”.



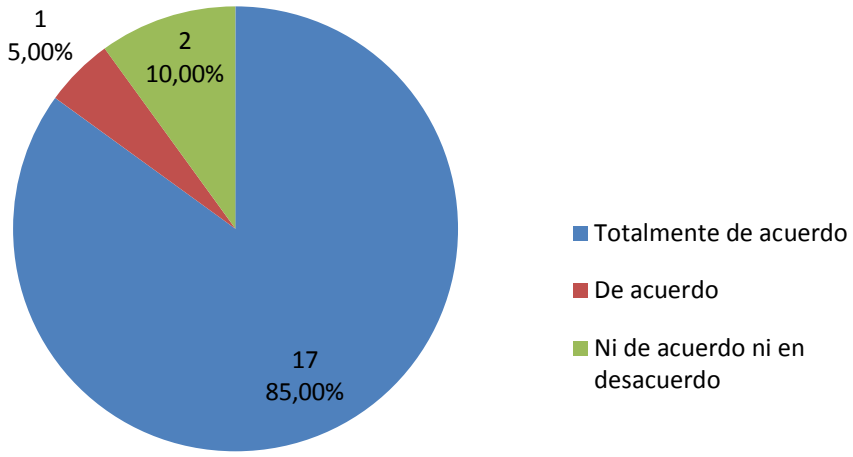
Gráfica 21. La búsqueda por URL responde adecuadamente

De acuerdo a la Gráfica 22, el 55% de los usuarios estuvo "Totalmente de acuerdo" con que la aplicación le ayudó cuando cometió un error, el 30% estuvo "De acuerdo" con esto y un 15%, correspondientes a un total de tres personas, respondieron que estuvieron "Ni de acuerdo ni en desacuerdo". Por lo tanto, para el manejo de errores en el módulo de búsqueda la evaluación arrojó resultados positivos.



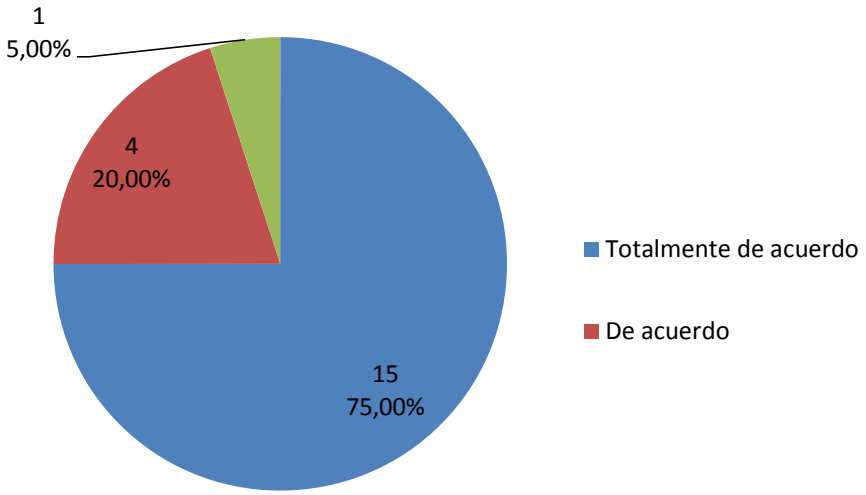
Gráfica 22. Da asistencia en caso de errores

En cuanto a la modalidad de búsqueda por colecciones, la Gráfica 23 muestra que el 85% de los usuarios estuvo "Totalmente de acuerdo" con que la búsqueda por colecciones es clara, es decir, sencilla de identificar, un 10% estuvo "De acuerdo" con esto y el restante 5% estuvo "Ni de acuerdo ni en desacuerdo".



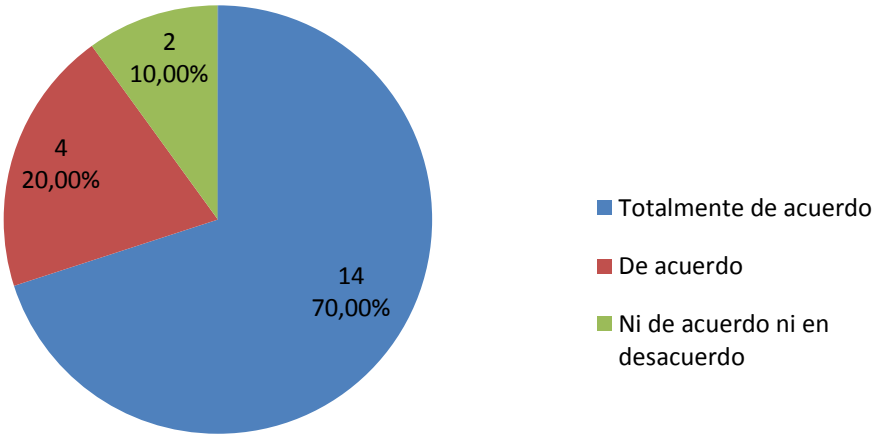
Gráfica 23. La búsqueda por colecciones es clara

El éxito de la aplicación Web para el Prototipo de Archivo Web de Venezuela depende en gran parte de la experiencia del usuario con la misma, ya que si los resultados de sus búsquedas no son los que espera entonces no tendrá motivos para volver a utilizarla o simplemente buscará una alternativa distinta. La Gráfica 24 muestra resultados positivos ya que un 95% de los encuestados se encuentran entre las opciones "Totalmente de acuerdo" (75%) y "De acuerdo" (20%). Con estos resultados se denota que ambas modalidades de búsqueda arrojaron resultados positivos.



Gráfica 24. Las búsquedas dan resultados esperados

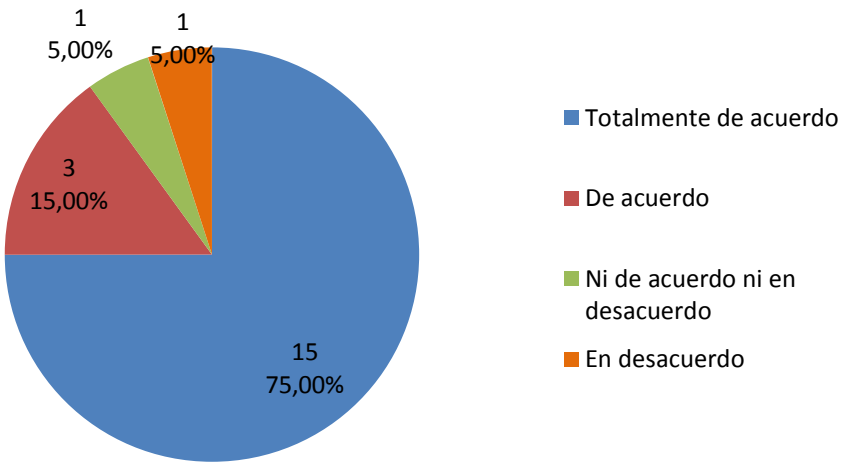
Un elemento importante dentro del Módulo de Búsqueda es el calendario de versiones, por lo que la facilidad de uso y su entendimiento son claves para el despliegue de sitios preservados. La Gráfica 25 muestra que el calendario de versiones es bastante claro entre las personas encuestadas, ya que un 70% estuvo “Totalmente de acuerdo” con esto, un 20% estuvo “De acuerdo” y solo un 10% representado por 2 personas estuvieron “Ni de acuerdo ni en desacuerdo”.



Gráfica 25. El calendario de versiones es sencillo de entender

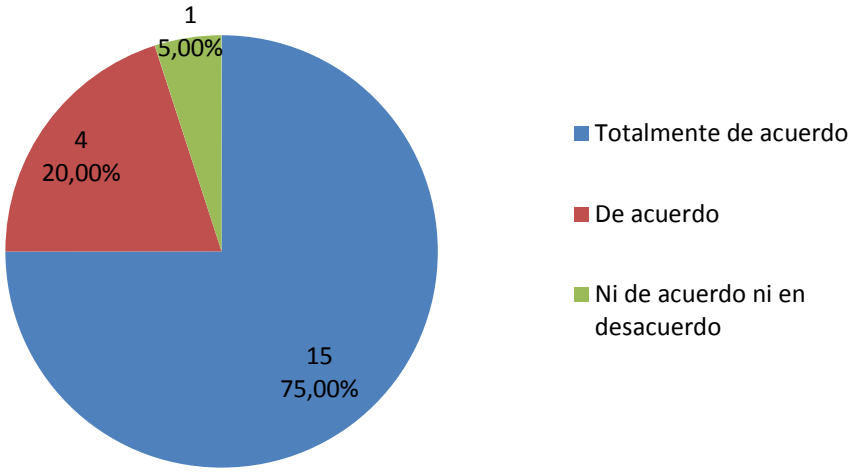
Las siguientes cuatro gráficas corresponden al Módulo de Sitios Web Desplegados. Este módulo representa el núcleo central de la aplicación y el objetivo principal de este TEG, por lo que su correcto funcionamiento es primordial dentro de la aplicación Web desarrollada.

Para lograr una navegación fluida y clara a través de los sitios Web desplegados los enlaces deben funcionar adecuadamente, redireccionando de forma correcta y siempre manteniendo a los usuarios dentro del Archivo Web. La Gráfica 26 muestra que el 75% de los usuarios encuestados respondieron que están "Totalmente de acuerdo" con que los enlaces funcionan como esperaban mientras que un 15% respondió que están "De acuerdo". Esto significa que los enlaces, para un 90% de los usuarios, respondieron de forma correcta. Como el restante 10% no realizó observaciones con respecto a este tópico se desconocen las razones por las que pudo fallar algún enlace. En general los resultados fueron satisfactorios.



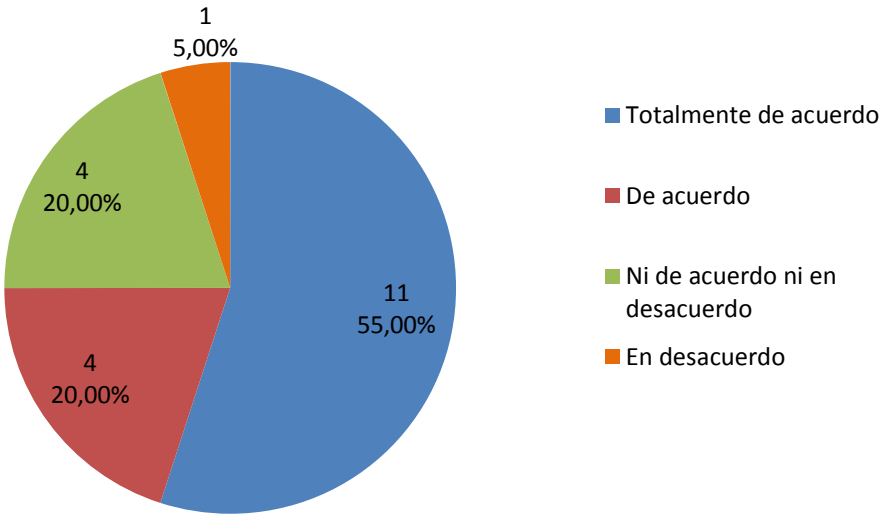
Gráfica 26. Los enlaces funcionan cómo esperaba

La Gráfica 27 es una evaluación basada en la experiencia del usuario navegando dentro de un sitio desplegado. Esta gráfica es un complemento de la anterior, ya que si los enlaces funcionan de forma adecuada lo esperado es que la navegación sea satisfactoria. La gráfica muestra que un 95% de los usuarios, divididos entre las respuestas "Totalmente de acuerdo" (75%) y "De acuerdo" (20%), pudieron navegar el sitio Web desplegado correctamente.



Gráfica 27. La navegación por el sitio ha sido satisfactoria

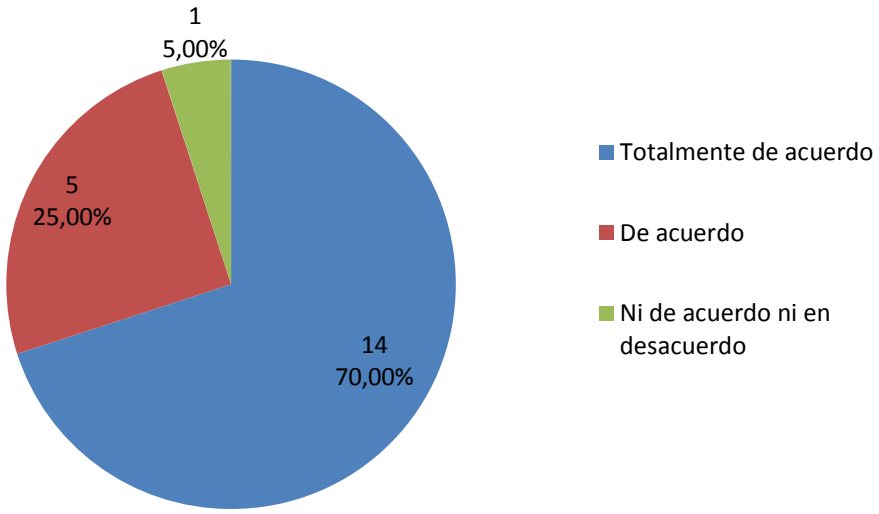
La Gráfica 28 evalúa la utilidad de la información estadística proporcionada por la interfaz. Los resultados muestran como un total de quince personas que representan el 75% de los usuarios encuestados están entre las opciones de "Totalmente de acuerdo" (55%) y "De acuerdo" (20%). Pero para el 25% restante la información estadística puede no ser relevante ya que cuatro personas (20%) respondieron que están "Ni de acuerdo ni en desacuerdo" y una persona (5%) respondió que está "En desacuerdo".



Gráfica 28. La información estadística le parece útil

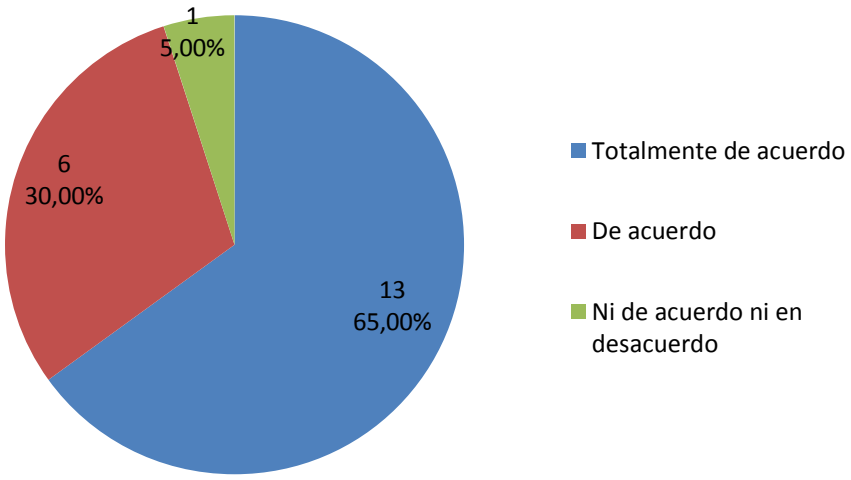
Las siguientes cuatro gráficas se enfocan en lo que la aplicación le ha permitido al usuario. Al ser una iniciativa cuya importancia radica en el ámbito de la educación es esencial que permita a los usuarios comprender lo que es un Archivo Web y los beneficios que éste puede aportar.

La Gráfica 29 evalúa si el usuario logró entender y conocer lo que es un Archivo Web. Como se puede apreciar una gran mayoría de los encuestados lograron comprenderlo, ya que un 70% estuvo "Totalmente de acuerdo" y un 25% estuvo "De acuerdo" con esta premisa.



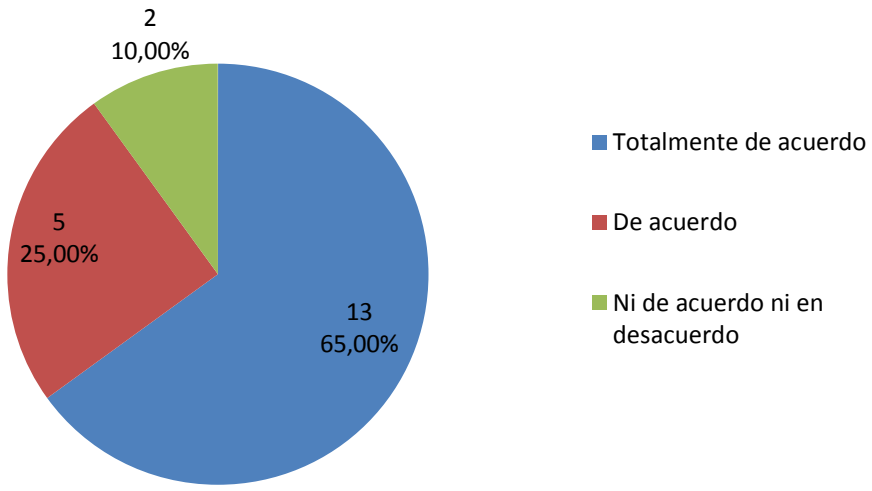
Gráfica 29. Conocer que es un Archivo Web

La Gráfica anterior permitió saber si los usuarios encuestados lograron comprender lo que es un Archivo Web, para la siguiente se evalúa si dichos usuarios lograron saber, aún más específicamente, de qué trata el Archivo Web de Venezuela. La Gráfica 30 muestra que un 65% estuvo "Totalmente de acuerdo" con esto, mientras que un 30% estuvo "De acuerdo" y el 5% restante estuvo "Ni de acuerdo ni en desacuerdo".



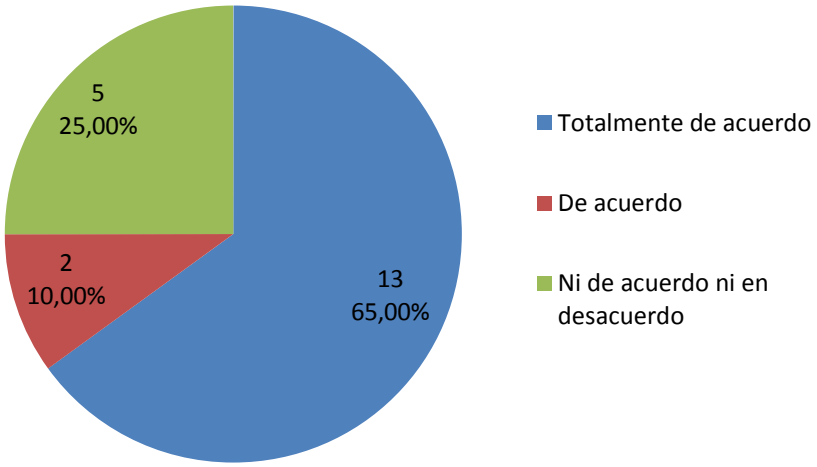
Gráfica 30. Saber de qué trata el Archivo Web de Venezuela

La Gráfica 31 evalúa si los usuarios pudieron contactar a través de un mensaje directo a los administradores del sistema. Se puede observar cómo un 90% de las personas encuestadas tuvieron éxito para lograr el objetivo, ya que 65% de ellas contestaron “Totalmente De acuerdo” y un 25% estuvo “De acuerdo”.



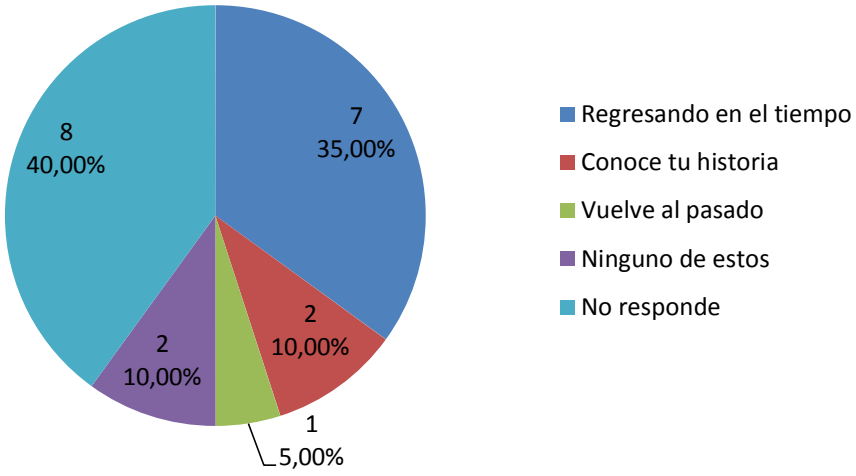
Gráfica 31. Ponerme en contacto con los administradores

La Gráfica 32 evalúa si la aplicación les ha permitido a los usuarios conocer detalles técnicos del sistema. En este caso se puede observar cómo aunque el 65% de los usuarios estuvieron “Totalmente de acuerdo” con esto, un 25% estuvo “Ni de acuerdo ni en desacuerdo”. Esto se puede deber a que la arquitectura de este tipo de sistemas es compleja para aquellas personas no familiarizadas con la misma.



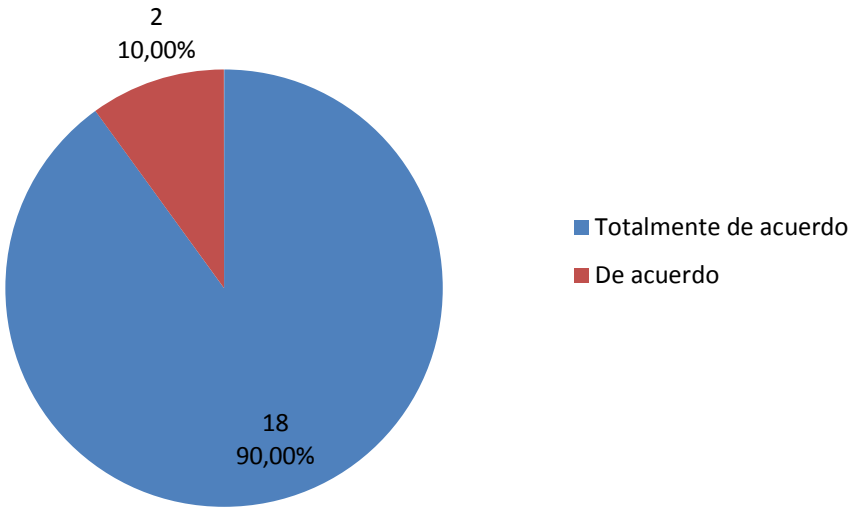
Gráfica 32. Conocer detalles técnicos del sistema

Junto al logotipo del Prototipo de Archivo Web de Venezuela se colocó un mensaje que resumiera y representara la utilidad del mismo en una sola frase. La Gráfica 33 muestra que la preferencia de los encuestados en cuanto a dicho mensaje, se inclinó a "Regresando en el tiempo"; por lo cual este fue en mensaje adoptado en el prototipo.



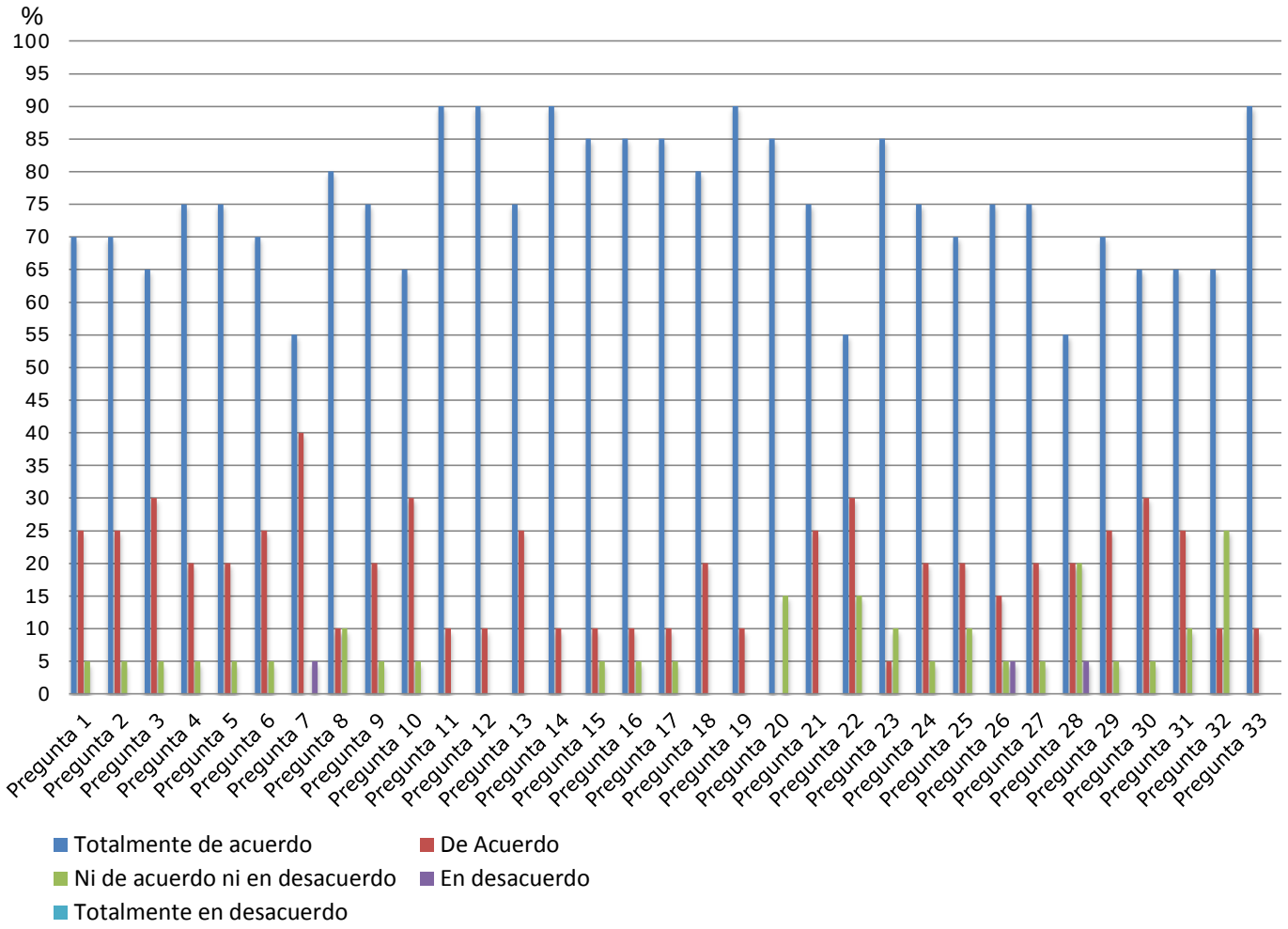
Gráfica 33. Mensaje que se muestra junto al logo del prototipo

Finalmente, la Gráfica 34 muestra que la aplicación tuvo una aceptación del 100% por parte de los encuestados puesto que están dispuestos a recomendarla.



Gráfica 34. Estaría dispuesto a recomendar la aplicación

A continuación se presenta un gráfico que engloba todas las preguntas realizadas en la encuesta y las resume en uno solo:



Gráfica 35. Gráfico resumen de la encuesta

Preguntas:

Sobre la interfaz

Con respecto a la interfaz gráfica:

Pregunta 1	Es fácil comprender las acciones que se pueden realizar
Pregunta 2	Es fácil de aprender (en poco tiempo se conoce las funciones)
Pregunta 3	Es fácil de usar (las acciones tienen bajo nivel de complejidad)
Pregunta 4	Los colores son agradables
Pregunta 5	Es consistente en todas las páginas (estructura, disposición de imágenes, entre otros)
Pregunta 6	La experiencia con la interfaz fue positiva

Con respecto a la información que provee la interfaz:

Pregunta 7 Es comprensible

Pregunta 8 Es útil para lograr llevar a cabo un objetivo

Pregunta 9 Las funciones del menú son claras

Pregunta 10 Los mensajes aportan significado

Pregunta 11 El logotipo de la aplicación se adecúa al tema de la misma

Opciones para el usuario

En cuanto al registro del usuario:

Pregunta 12 Sencillo de llevar a cabo

Pregunta 13 El sistema ayudó en caso de errores

Pregunta 14 El inicio de sesión se hace de manera fácil

En cuanto a la sección de favoritos:

Pregunta 15 Es fácil de administrar

Pregunta 16 Facilita las búsquedas

Pregunta 17 Guarda con éxito mis selecciones

En cuanto al historial de búsqueda:

Pregunta 18 Es fácil de administrar

Pregunta 19 Facilita las búsquedas

Pregunta 20 Es útil

Funciones de la aplicación

Con relación a las búsquedas:

Pregunta 21 La búsqueda por URL responde adecuadamente

Pregunta 22 Da asistencia en caso de errores

Pregunta 23 La búsqueda por colecciones es clara

Pregunta 24 Las búsquedas dan resultados esperados

Pregunta 25 El calendario de versiones es sencillo de entender

Con relación a los sitios Web desplegados:

Pregunta 26 Los enlaces funcionan cómo esperaba

Pregunta 27 La navegación por el sitio ha sido satisfactoria

Pregunta 28 La información estadística le parece útil

La aplicación me ha permitido:

Pregunta 29 Conocer que es un Archivo Web

Pregunta 30 Saber de qué trata el Archivo Web de Venezuela

Pregunta 31 Ponerme en contacto con los administradores

Pregunta 32 Conocer detalles técnicos del sistema

Pregunta 33 Estaría dispuesto a recomendar la aplicación

A través de estos resultados es posible afirmar que la aplicación presentó un alto grado de aceptación por parte de los encuestados, dado que todas las preguntas tuvieron resultados positivos. Sin embargo, además de las respuestas proporcionadas, se les solicitó a los usuarios que ofrecieran recomendaciones o resaltaran aspectos, que a su parecer, deben ser corregidos en la aplicación. Algunas de estas recomendaciones fueron:

- **Con respecto a la interfaz gráfica:**
  - Cambiar el color de los mensajes de éxito ya que se mostraban en color rojo, el cual es usualmente utilizado para representar errores. La solución fue colocarlos en color verde en el caso de éxito y rojo en el caso de que ocurra algún error.

  - Al momento de desplegar el sitio, la palabra *Volver,* que le permite al usuario regresar a la interfaz desde la cual seleccionó la versión del sitio a desplegar, debería ser mostrada en una zona que resalte más. La solución fue colocarla en el extremo izquierdo de la barra, justo a un lado del logotipo, y también mostrar una flecha que asemeje a la del navegador que permite regresar a la página anterior.

  - Cuando el usuario selecciona *Agregar a Favoritos* debería mostrarse un mensaje de éxito. La solución fue colocar el mensaje *¡Añadido A Mis Favoritos!*

- **Con respecto a la información mostrada por la interfaz:**
  - La frase *Introduce el URL* no es lo suficientemente intuitiva para el usuario, ya que no especifica cómo debería introducirse o qué formato seguir, por lo tanto, para solucionarlo se agregaron ejemplos con las distintas maneras de ingresar datos para así poder generar resultados.

  - El calendario de versiones no muestra el nombre del sitio que fue buscado, lo que puede desorientar al usuario. La solución fue simplemente colocar la URL del sitio en el tope de la interfaz, justo sobre el calendario.

  - La sección de Archivo Web de Venezuela no es clara en cuanto al concepto y propósito del mismo. Por lo tanto, se extendió la información presentada agregando las causas por las que se crea el archivo Web y su concepto general.

- Otra recomendación fue la de agregar una ayuda para el usuario en línea que permita guiarlo paso a paso hacia el logro de sus objetivos dentro del Archivo Web de Venezuela. Esto fue solucionado desarrollando una guía interactiva que enumera los pasos a seguir para buscar y desplegar un sitio Web preservado, utilizando textos, animaciones e imágenes.

CONCLUSIONES

Al culminar este trabajo se considera que se cumplieron los objetivos planteados debido a que se implementó una aplicación Web para el acceso y despliegue de contenido Web preservado en formato WARC, la cual se integró con el módulo de almacenamiento e indexación y forma parte del prototipo de Archivo Web de Venezuela y así poder contribuir al uso del formato WARC para la preservación de sitios Web. Dicho objetivo guió la investigación hacia la revisión de aplicaciones Web antecedentes que tuvieran como función principal la preservación de archivos Web. Las aplicaciones que fueron tomadas en cuenta utilizan como herramienta de acceso y despliegue Wayback Machine, por lo que el principal requerimiento se convirtió en el desarrollo de una alternativa propia de la Universidad Central de Venezuela y del prototipo de Archivo Web de Venezuela. Para cumplir el objetivo principal del trabajo de grado y los objetivos derivados del mismo, se utilizó ágil XP como metodología de desarrollo y Ruby on Rails como lenguaje de programación obteniendo resultados satisfactorios.

La realización del TEG presentó una serie de retos a nivel de programación, debido a la escasa documentación del formato de archivo WARC con el cual se trabajó; y a nivel de investigación que nos ayudaron como grupo de trabajo a mejorar aspectos profesionales, personales y de organización. La metodología XP nos permitió seguir un patrón específico de tareas para poder agilizar y mejorar el proceso de desarrollo, mientras que la elección de lenguaje y del framework que lo acompaña permitió ordenar la aplicación de acuerdo a una arquitectura MVC, para así mantener un espacio de nombres relacionados y un flujo de datos uniforme. Por lo tanto a nivel técnico fue una experiencia de crecimiento y aprendizaje.

Todos estos retos fueron incentivos para cumplir con los objetivos planteados innovando soluciones y buscando una identidad propia para el prototipo de Archivo Web de Venezuela, sin dejar a un lado la importancia de la educación como base para el desarrollo de un país, siendo este proyecto un avance en el área. El contenido Web es muy vasto y conlleva a saber el origen de aquello que nos rodea (cultura, tecnología, entre otros), documentar los nuevos descubrimientos, mantenerse al tanto de lo que ocurre en el día a día, y predecir comportamientos futuros de cualquier tipo de elemento que pueda progresar con el tiempo. Por esta razón el proyecto sumó además una motivación extra, que le da una validez mayor al esfuerzo realizado para lograr obtener un prototipo que satisfaga las necesidades de los usuarios a nivel educativo y personal.

Por medio de la encuesta realizada, se obtuvo opiniones, comentarios y sugerencias sobre la aplicación realizada. Los resultados obtenidos a través de las encuestas son alentadores para la continuación del proyecto, pues tuvo un alto grado de aceptación ya que todas las preguntas tuvieron resultados positivos. Además de esto, la retroalimentación conseguida a través de las personas encuestadas brindó un valioso aporte para la mejora de la aplicación.

Durante la elaboración del prototipo de Archivo Web de Venezuela surgieron limitaciones debido a distintas razones: la escasa documentación para trabajar con archivos WARC, no se consiguieron precedentes oficiales que utilizaran Ruby on Rails o gemas para trabajar archivos de este formato, por lo que se utilizó Warc-Tools para el procesamiento de los archivos trayendo consigo limitaciones propias de las librerías que se utilizan, por ejemplo, de tiempo, ya que la extracción de los archivos utilizando la librería *warcunpack_ia.py* toma un tiempo considerable; otras limitaciones se dieron por la naturaleza de los sitios rastreados, es decir, la formación de los archivos HTML, ya que es no son uniformes en su estructura, porque dependen de quien desarrolle el sitio y de las tecnologías y metodologías utilizadas.

Para solucionar estas limitaciones se propone que se lleven a cabo las siguientes acciones en trabajos futuros:

- En la solución planteada en este TEG, el tiempo de respuesta de la aplicación a las peticiones de despliegue de los usuarios se ven reducidas puesto que el archivo WARC se encuentra extraído y procesado para su posterior despliegue, y esto implica una mayor necesidad de almacenamiento por la duplicación de los archivos, por lo tanto se recomienda que la aplicación de despliegue posea un servidor propio separado de aquellos que son necesarios para los rastreos y almacenamiento y en este se encuentren los archivos WARC ya listos para ser accedidos.

- Se recomienda que al igual que en la solución planteada en este trabajo se extraiga el WARC una vez rastreado el sitio, pero la manipulación de cada archivo extraído (entiéndase el manejo de los enlaces y los nombres de archivos) se realice cuando el usuario solicite dicho archivo, ya que el tiempo de la manipulación de los archivos depende de la cantidad de archivos que html modo podría reducirse notablemente.

- En cuanto a los enlaces que redireccionan a sitios externos al Archivo Web, se recomienda tomar en cuenta los demás directorios dentro del archivo WARC, puesto que en el caso de estudio solo fue tomado en cuenta el directorio principal del sitio.

- En cuanto a la búsqueda de contenido, se propone brindarle al usuario la posibilidad de filtrar sus búsquedas de acuerdo a información contenida en el WARC. Para esto se requiere modificaciones tanto en el Módulo de Acceso a los Contenidos Preservados en formato WARC, como en el Módulo de Indexación, para que el usuario pueda hacer una consulta más amplia y/o específica sobre los contenedores.

REFERENCIAS

BIBLIOGRÁFICAS Y DÍGITALES

Archive-it. (2006). *Home: Archive-it*. Recuperado el 2012, de Archive-it Web site: http://www.archive-it.org/

Burner, M., & Kahle, B. (Septiembre de 1996). *WWWArchive File Format Specification*. Obtenido de http://pages.alexa.com/company/arcformat.html

Cho, J., & García-Molina, H. (2000). *The evolution of the web and implications for an Incremental Crawler. Paper presented at the Proceedings of the 26th International Conference on Very Large Data Bases.*

Consultative Committe for Space Data System. (2002). *Reference Model for an Open Archival Information System .* Recuperado el Enero de 2013

Fernández Nogales, A. (2004). *Investigación y Técnicas de Mercado.* ESIC Editorial.

Fetterly, D., Manasse, M., Najork, M., & Wiener, J. (2003). *A large-scale study of the evolution of web pages.* Budapest.

García, J. y Rivero, L. (2013). Implementación de los módulos de adquisición y almacenamiento de un prototipo para el archivado de sitios Web en Venezuela. Trabajo Especial de Grado. Universidad Central de Venezuela.

Gomes, D., Miranda, J., & Costa, M. (2010). *A survey on web archiving initiatives.* Portugal.

Gomes, D., Nogueira, A., Miranda, J., & Costa, M. (s.f.). *Introducing the Portuguese web archive initiative.* FCCN-Fundação para a Computação Científica Nacional.

Grotke, A; IIPC. (2008). *Member Profile Survey Results.* Australia.

Hunt, A. (13 de Diciembre de 2011). *Don´t Repeat Yourself.* Recuperado el Diciembre de 2012, de http://c2.com/cgi/wiki?DontRepeatYourself

(a) IIPC . (2012). *IIPC*. Obtenido de http://www.netpreserve.org/about-us/mission-goals

(b) IIPC. (2012). *IIPC-Members*. Obtenido de http://www.netpreserve.org/about-us/members

(c) IIPC. (2012). *IIPC-WarcTools Project*. Obtenido de http://netpreserve.org/projects/warc-tools-project

Internet Archive. (s.f.). *About: About the Internet Archive*. Recuperado el 2012, de Internet Archive: http://archive.org/about/

Internet Archive. (2011). *Internet Archive*. Recuperado el 18 de 09 de 2013, de http://archive-access.sourceforge.net/projects/wayback/

Internet Memory Foundation. (2010). *Web Archiving in Europe*. Obtenido de http://internetmemory.org/images/uploads/Web_Archiving_Survey.pdf, 2010.

ISO. (2003). *Nomras ISO 14721*. Obtenido de http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=24683

ISO. (2009). *ISO. 28500 Information and documentation-WARC file format.* Nueva Zelanda.

Jack, P., & Binns, A. (12 de Junio de 2012). *Web Archive - ARC*. Obtenido de https://webarchive.jira.com/wiki/display/Heritrix/ARC+File+Format

Library of Congress Web Archives. (s.f.). *Home: Library of Congress Web Archives*. Recuperado el 2012, de Library of Congress Web Archives Web site: http://lcweb2.loc.gov/diglib/lcwa/html/lcwa-home.html

Lujan Mora, S. (31 de Octubre de 2002). *Universidad de Alicante. Departamento de Lenguajes y Sistemas Informáticos.* Obtenido de http://rua.ua.es/dspace/bitstream/10045/16995/1/sergio_lujan-programacion_de_aplicaciones_web.pdf.

Masanés, J. (2006). *Web Archives.* New York.

Meghini, C. (13 de Octubre de 2008). *The Reference Model for an Open Archival Information System*. Recuperado el 17 de Enero de 2013, de http://www.digitalpreservationeurope.eu: http://www.digitalpreservationeurope.eu/preservation-training-materials/files/oais-reference-model.pdf

nagore.otsoa.net. (s.f.). Recuperado el Noviembre de 2012, de nagore.otsoa.net/memhn/patrimonio.htm

National Library of New Zealand; British Library; IIPC. (2012). *Web Curator*. Obtenido de http://webcurator.sf.net/

Ospina Torres, M.H. y León Luna, C.P. (2013). Una arquitectura basada en software libre para archivos web. En la Revista Venezolana de Información, Tecnología y Conocimiento, 10 (1), 53-72, recuperado en junio de 2013. Disponible en: http://netpreserve.org/http://revistas.luz.edu.ve/index.php/enlace/article/viewFile/12755/12326.

(a) PANDORA Australia's Web Archive. (s.f.). *About PANDORA DIGITAL ARCHIVING SYSTEM (PANDAS): Overview*. Recuperado el 2012, de PANDORA Australia's Web Archive Web site: http://pandora.nla.gov.au/pandas.html

(b) PANDORA Australia's Web Archive. (s.f.). *About PANDORA: An Overview*. Recuperado el 2012, de PANDORA Australia's Web Archive Web site: http://pandora.nla.gov.au/overview.html

Plekhanova, J. (Septiembre de 2009). *Institute for Business and Information Technology.* Recuperado el Febrero de 2013, de http://ibit.temple.edu/wp-content/uploads/2011/03/IBITWebframeworks.pdf

Portuguese Web Archive. (s.f.). *Home: Portuguese Web Archive*. Recuperado el 2012, de Portuguese Web Archive Web site: http://sobre.arquivo.pt/

Proyecto Minerva. (2012). Recuperado el 2013, de http://lcweb2.loc.gov/diglib/lcwa/html/lcwa-home.html

Ramírez, H., Hodgson, J., Reyes, J., & Coleman, K. (16 de Noviembre de 2010). *Arquitectura Cliente Servidor.* Recuperado el Diciembre de 2012, de http://www.slideshare.net/NoeGonzalezMendoza/arquitectura-cliente-servidor

REBUIN. (2009). *Preservación digital: Guía de recursos.* España.

Rengarajan, M. S. (2013). *Technology Association of Georgia.* Recuperado el Febrero de 2013, de http://www.tagonline.org/files/33_ExtremeProgramming_RR.pdf

Shiozaki, R. (2009). *Role and justification of web archiving by national libraries a questionnaire survey.* Recuperado el Noviembre de 2012, de http://lis.sagepub.com/content/41/2/90

Spinellis, D. (2003). *The decay and failures of web references.* Communications of ACM.

The Digital Formats. (s.f.). *The Digital Formats*. Recuperado el Marzo de 2013, de The Digital Formats: http://www.digitalpreservation.gov/formats/fdd/fdd000236.shtml

Wells, D. (2009). *Extreme Programming*. Recuperado el 2013, de http://www.extremeprogramming.org

(a) UNESCO. (2003). *Charter on the Preservation of the Digital Heritage.* Australia.

(b) UNESCO.