



UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
CENTRO DE INVESTIGACION DE SISTEMAS DE INFORMACIÓN CISI

Desarrollo del módulo de acceso y despliegue para la versión 2
del "Prototipo de Archivo Web de Venezuela".

**Trabajo Especial de Grado presentado ante la ilustre
Universidad Central de Venezuela por el
Br. Luis Daniel Castillo Cisnero, CI: 22756693
Tutora: Profa. Mercy Ospina**

12/07/2018

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 y despliegue para la versión 2 del "Prototipo de Archivo Web de Venezuela"." y presentado por el bachiller: Luis Castillo, 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 15 de Octubre de 2018, a las 3 pm horas, para que los autores lo defendieran en forma pública, lo que estos hicieron en la Sala B III 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 19 puntos.

En fe de lo cual se levanta la presente Acta, en Caracas el día 15 de Octubre de 2018



Prof. Mercy Ospina

(Tutora)



Prof. Andres Sanoja

(Jurado)



Prof. Franky Uzcátegui

(Jurado)

AGRADECIMIENTOS

Agradezco a mi madre, padre, hermano y a toda mi familia por todo el apoyo y el amor brindado.

Agradezco a la aldea por toda la pasión y el camino que hemos compartido en esta etapa, a las vivencias, a los buenos y los malos manejos.

Agradezco a la Universidad Central de Venezuela y a todos los profesores de los que tuve el honor de aprender durante todo mi camino vivido en ella.

Agradezco a King Crimson, el pensamiento crítico y la disconformidad por darme las ganas de mejorar.

RESUMEN

El progreso de la humanidad se debe a su capacidad intrínseca de razonar y transmitir conocimiento. Esta cualidad ha progresado a la par de la humanidad misma, desde las primeras estructuras sociales, cuando el humano era nómada y se transmitían conocimientos de técnicas de caza y elaboración de herramientas hasta el día de hoy donde gran porción de la humanidad tiene acceso al internet, y con él la capacidad de compartir información de manera casi instantánea y en cualquiera punto del planeta. Es importante la preservación de este conocimiento para hacerlo disponible para presentes y futuras generaciones, y es por eso su forma de representación ha evolucionado junto con el proceso de preservación en sí, desde la pintura rupestre, jeroglíficos, papiros hasta el internet. Este conocimiento, denominado patrimonio cultural ha sido clasificado como tangible tal como las esculturas, monumentos, entre otros, e intangible como por ejemplo las costumbres folklóricas, conocimiento científico, obras literarias, entre otros. Actualmente el tipo de patrimonio que más se genera es el intangible que gracias a las TICs y al Internet se encuentran cada día más en formato digital, de acá que la Unesco definiera el concepto de patrimonio digital y los lineamientos para su preservación. Para esto se han desarrollado, desde los 90s un conjunto de iniciativas de preservación web, llamadas Archivos Web. Estos sistemas presentan retos propios de la naturaleza del patrimonio preservado, desde la capacidad de almacenamiento, el rastreo y la recolección periódica de los elementos a preservar, su posterior indexación y finalmente la publicación (visualización) del material preservado, siendo este último reto el que concierne a este trabajo, el cual se enmarca dentro del desarrollo de un prototipo de Archivo Web propuesto por la Universidad Central de Venezuela, el cual se encuentra en su segunda fase que busca solucionar temas de rendimiento y escalabilidad que tenía la primera versión. Con esta nueva versión se emplean nuevas herramientas que hacen un mejor uso de los recursos disponibles, así como una nueva arquitectura que distribuye el trabajo en diferentes módulos. El objetivo de este Trabajo Especial de Grado es desarrollar el nuevo módulo de acceso, encargado de exhibir las páginas web preservadas que el usuario consulte a través de su buscador, para este se segmenta el módulo en un Front-end, encargado del despliegue de las páginas y un Back-end encargado de la comunicación los otros módulo del sistema y la transmisión de información entre este y el Front-end, cada uno desarrollado con una tecnología distinta, idónea para cada una de sus tareas y usando una adaptación de las metodologías de desarrollo AgisUS y AgilUP.

Palabras Clave: Archivo Web, módulo de acceso, react, django, preservación, AgilUS, AgilUP

Índice

RESUMEN.....	I
ÍNDICE	IV
ÍNDICE DE FIGURAS	VII
ÍNDICE DE TABLAS	IX
INTRODUCCIÓN.....	1
CAPÍTULO 1 PROBLEMA DE INVESTIGACIÓN	3
1.1. PLANTEAMIENTO DEL PROBLEMA.....	3
1.2. ALCANCE DE LA APLICACIÓN	5
1.3. OBJETIVOS DEL TRABAJO ESPECIAL DE GRADO	6
1.3.1. <i>Objetivo general</i>	6
1.3.2. <i>Objetivos específicos</i>	6
CAPÍTULO 2 MARCO CONCEPTUAL	7
2.1. PATRIMONIO CULTURAL.....	7
2.2. PRESERVACIÓN WEB	7
2.3. ARCHIVO WEB	9
2.4. FORMATO WARC.....	10
2.4.1. <i>Tipos de Registros WARC</i>	11
2.5. ANTECEDENTES DE ARCHIVADO WEB.....	12
2.5.1. <i>CyberCemetery</i>	14
2.5.2. <i>Internet Archive</i>	15
2.5.3. <i>Pandora</i>	16
2.5.4. <i>Archivo Web de la Biblioteca del Congreso</i>	18
2.5.5. <i>Consortio Internacional de Preservación de Internet (IIPC)</i>	19
2.6. PROCESO DE PRESERVACIÓN.....	19

2.6.1. Adquisición	19
2.6.2. Almacenamiento.....	20
2.6.3. Indexación y búsqueda.....	20
2.6.4. Acceso	25
2.7. APLICACIONES WEB	26
2.7.1. Arquitectura Cliente/Servidor	26
2.7.2. Estado de Transferencia Representacional (ReST).....	28
2.7.3. Patrón arquitectónico Modelo-Vista-Controlador.....	32
2.7.4. NodeJS y NPM	36
CAPÍTULO 3 MARCO METODOLÓGICO	38
3.1. METODOLOGÍAS ÁGILES	38
3.1.1. Ágil UP.....	38
3.1.2. AgilUS.....	41
3.1.3. Comparación de metodologías.....	45
3.2. DESARROLLO DE SOFTWARE BASADO EN COMPONENTES	46
3.2.1. Características	48
3.2.2. Modelos de componente	48
3.2.3. Composición	49
3.2.4. Ciclo de vida	49
3.2.5. Criterios	50
3.2.6. Problemas	50
CAPÍTULO 4 MARCO APLICATIVO	52
4.1. ARQUITECTURA	52
4.2. REQUERIMIENTOS	53
4.2.1. Funcionales.....	55
4.2.2. No funcionales	56
4.3. ADAPTACIÓN DE LA METODOLOGÍA.....	53

4.4. ITERACIÓN 0: CONCEPCIÓN	53
4.4.1. Requisitos.....	53
4.4.2. Etapa de análisis	56
4.4.3. Prototipaje.....	63
4.4.4. Entrega	64
4.5. SEGUNDA ITERACIÓN: REFINAMIENTO DEL FRONT-END	65
4.5.1. Requerimientos	65
4.5.2. Análisis	65
4.5.3. Prototipaje.....	65
4.5.4. Entrega	67
4.6. TERCERA ITERACIÓN: COMUNICACIÓN CON MÓDULO DE INDEXACIÓN	68
4.6.1. Requerimientos	68
4.6.2. Análisis	68
4.6.3. Prototipaje.....	69
Pruebas de integración	73
4.6.4. Entrega	81
CONCLUSIONES	82
REFERENCIAS BIBLIOGRÁFICAS	85

Índice de figuras

Figura 1 - Primera versión de Arquitectura del prototipo	4
Figura 2 - Nueva arquitectura del prototipo.....	5
Figura 3 - Definición de Archivo Web.....	9
Figura 4 Fases del proceso de preservación web	10
Figura 5 - Arquitectura Cliente/Servidor	27
Figura 6 - Arquitectura MVC.....	33
Figura 7 - Ciclo de vida AUP.....	39
Figura 8 - Ciclo de vida AgilUS	43
Figura 9 - Flujo DSBC.....	48
Figura 10 - Modelo de componente	49
Figura 11 - Ciclo de vida DSBC.....	50
Figura 12 - Arquitectura Local	52
Figura 13 - Nivel 0.....	56
Figura 14 - Nivel 1.....	57
Figura 15 - Nivel 2.....	59
Figura 16 - Flujo de la aplicación	61
Figura 17 - Colores principales	62
Figura 18 - Colores secundarios.....	62
Figura 19 - Logo.....	62
Figura 20 - Página principal	64
Figura 21 - Búsqueda desplegada y versión seleccionada.....	64
Figura 22 - Página principal - Barra de búsqueda.....	66
Figura 23 - Página principal - Enlaces.....	66

Figura 24 - Información equipo AWW	67
Figura 25 - Formulario de contacto	67
Figura 26 - Configuración de comunicación con SolrCloud	69
Figura 27 - Expresión regular	69
Figura 28 - Petición a SolrCloud.....	70
Figura 29 - Búsqueda por palabra clave.....	70
Figura 30 - Búsqueda por URL.....	71
Figura 31 - Servicio despachador de sitios	71
Figura 32 - Versión seleccionada Pt.1	72
Figura 33 - Versión seleccionada Pt.2	72
Figura 34 - Respuestas de primera sección Pt.1	76
Figura 35 - Respuesta de primera sección Pt.2	76
Figura 36 - Respuestas segunda sección.....	77
Figura 37 - Respuestas de tercera sección de encuesta	79
Figura 38 - Respuestas de cuarta sección de encuesta	80
Figura 39 - Respuestas quinta sección.....	81

Índice de tablas

Tabla 1 Distribución geográfica de las iniciativas de preservación web y espacio usado en disco	13
Tabla 1 - Terminología SolrCloud	22
Tabla 4 - Elementos ReST	29
Tabla 5 - Componentes REST	30
Tabla 6 - Elementos de datos	31
Tabla 7 - Tabla comparativa entre AUP y AUS	46
Tabla 6 - Análisis Wayback Machine	54
Tabla 7 - Análisis Oasis	55
Tabla 8 - Nivel 1. Realizar consulta	57
Tabla 9 - Nivel 1. Consultar información del equipo.....	58
Tabla 10 - Nivel 2. Ingresar datos.....	59
Tabla 10 - Primera sección de encuesta	75
Tabla 11 - Segunda sección de encuesta	77
Tabla 12 - Tercera sección de encuesta	78
Tabla 13 - Cuarta sección de encuesta	79
Tabla 14 - Quinta sección de encuesta	80

INTRODUCCIÓN

El fenómeno de la era de información actual ha hecho que la información se mueva a tasas sin precedentes, desde la generación de la información hasta la distribución, esto tiene como consecuencia que la información llegue de manera más efectiva a los usuarios a los que va dirigida pero también que sea más efímera. Esto debido a que los medios que la presentan, refiriendo específicamente a las páginas web, son altamente dinámicos y la información que se presenta en una sección en un momento dado puede no estar en esa misma sección tiempo después y posiblemente ser eliminada por completo.

Es por esto que surgen iniciativas para preservar la web, ya que la preservación es una tarea que se ha estado realizando desde tiempos antiguos, lo que varía ahora es el método para hacerlo porque el objeto que se desea preservar ahora es intangible. Se deben emplear diferentes herramientas, técnicas y metodologías para lograrlo. Diferentes organizaciones como entes gubernamentales, universidades y bibliotecas han hecho su aporte en el tema desarrollando sistemas de archivo web, proponiendo diferentes estrategias para hacerlo y desarrollando herramientas de apoyo para lograr una preservación web efectiva.

En la Universidad Central de Venezuela se está desarrollando el primer prototipo de Archivo Web de Venezuela, basado en los modelos, estándares y tecnologías aportados por las diferentes organizaciones. El prototipo ya cuenta con una primera versión desarrollada, pero esta se encuentra limitada tanto en capacidades como en eficiencia, ya que la arquitectura del prototipo se ha actualizado con módulos tecnológicos no incluidas para el momento en que se desarrolló su primera versión y estos cambios fueron hechos para mejorar el rendimiento de la aplicación y también para hacer una preservación más efectiva, con nuevos módulos que agregan funcionalidades.

En el caso particular de la presente investigación, se desea realizar un estudio de conceptos, herramientas y metodologías para el desarrollo de una nueva versión del módulo de acceso del prototipo, la primera versión del módulo de acceso (Martinez, 2013) tiene problemas de rendimiento que serán explicados con más detalle en el documento, por un lado y por otro ya no se adapta a la arquitectura en su estado actual, motivo para el que se desarrolle una nueva versión del mismo.

El documento se compone de cuatro capítulos. El primero, "Planteamiento del problema", describe la razón, los problemas y la situación del prototipo de Archivo Web, también expone el objetivo general y los objetivos específicos del proyecto.

El segundo capítulo, "Marco Conceptual", define los conceptos básicos del proceso de preservación web, y también los conceptos y las herramientas computacionales necesarios para comprender el proyecto.

El tercer capítulo, "Marco Metodológico", expone la metodología a usar para el desarrollo del proyecto y también la metodología que se está siguiendo para el desarrollo independiente de cada módulo que compone el prototipo de Archivo Web.

El cuarto capítulo, "Marco aplicativo", describe el proceso de desarrollo del y los resultados obtenidos durante este.

El quinto capítulo concluye el documento y aporta ideas para futuros proyectos relacionados.

CAPÍTULO 1

PROBLEMA DE INVESTIGACIÓN

1.1. Planteamiento del problema

Un Archivo Web es un sistema computacional cuyo objetivo es implementar el proceso de preservación web. Como todo sistema, este debe basarse en una arquitectura e implementar un conjunto de herramientas, estrategias y metodologías que cumplan con diferentes aspectos necesarios para lograr su objetivo. En el caso de los Archivos Web, este será visible por el usuario final como un sitio web con la funcionalidad de buscador de páginas web, que una vez seleccionado un objetivo se desplegará como la página web original, de la fecha escogida por el usuario.

Hasta la fecha se han desarrollado numerosas iniciativas de preservación web, el primer proyecto de este tipo que se conoce fue realizado por la Universidad del Norte de Texas en 1995 con el nombre de CyberCemetery, el mismo es un archivo de sitios web gubernamentales. Posteriormente se desarrolló el proyecto PANDORA de origen Australiano, apoyado por la Librería Nacional de Australia como el segundo en adoptar la iniciativa del archivo web y con el tiempo se sumaron otros proyectos a como es el caso de Suecia con Kulturarw3 en 1996, Nueva Zelanda en 1999 con el Archivo Web de Nueva Zelanda, República Checa con Webarchiv en el 2000, entre otros. Posteriormente en el 2003 la UNESCO (UNESCO, 2003) planteó un conjunto de directrices para la preservación digital, sentando bases para estándares y formatos que, a pesar de haberse puesto en práctica ya por la mayoría de los proyectos previos, se consolidan oficialmente como reglas de jure a través del Consorcio Internacional de Preservación de Internet (IIPC, por sus siglas en inglés) (IIPC, 2012).

Todo proyecto de preservación web debe seguir un flujo de tareas: localizar el sitio web previamente seleccionado, procesarlo para volverlo un archivo en formato estándar, indexar y almacenar dicho archivo, y finalmente dar acceso a los recursos almacenados, para ser consultados por los usuarios interesados, así como información asociada a estos, la cual es la función del módulo de acceso.

Específicamente el módulo de acceso es una aplicación web que permite el acceso a los contenidos preservados dentro del Archivo Web, a través de su interacción con un motor de búsqueda y el sistema de almacenamiento, teniendo además que desplegar estos contenidos representados en páginas web. El presente trabajo está enmarcado en la implementación de un prototipo de Archivo Web que se inició con el proyecto de "Desarrollo de un prototipo de Archivo Web para la preservación del patrimonio Web de Venezuela" de código PI-03-8139-2011/P de la Universidad Central de Venezuela, en la primera versión la cual está contenida en la Figura 1. En esta versión el procesamiento de los contenidos preservados era centralizado, por lo que el acceso implicaba tareas sobre archivos extensos que debían ser procesados cada vez que eran desplegados (ver sección 2.4.), esto demostró ser una tarea ineficiente, porque este procesamiento consume tiempo que se ve reflejado en espera al hacer la consulta, además de que este tiempo es proporcional al tamaño del archivo

que se procese, los cuales pesan entre 100 megabytes a más de 1 gigabyte, haciendo consultas con espera de hasta 10 minutos (Martínez, 2013).

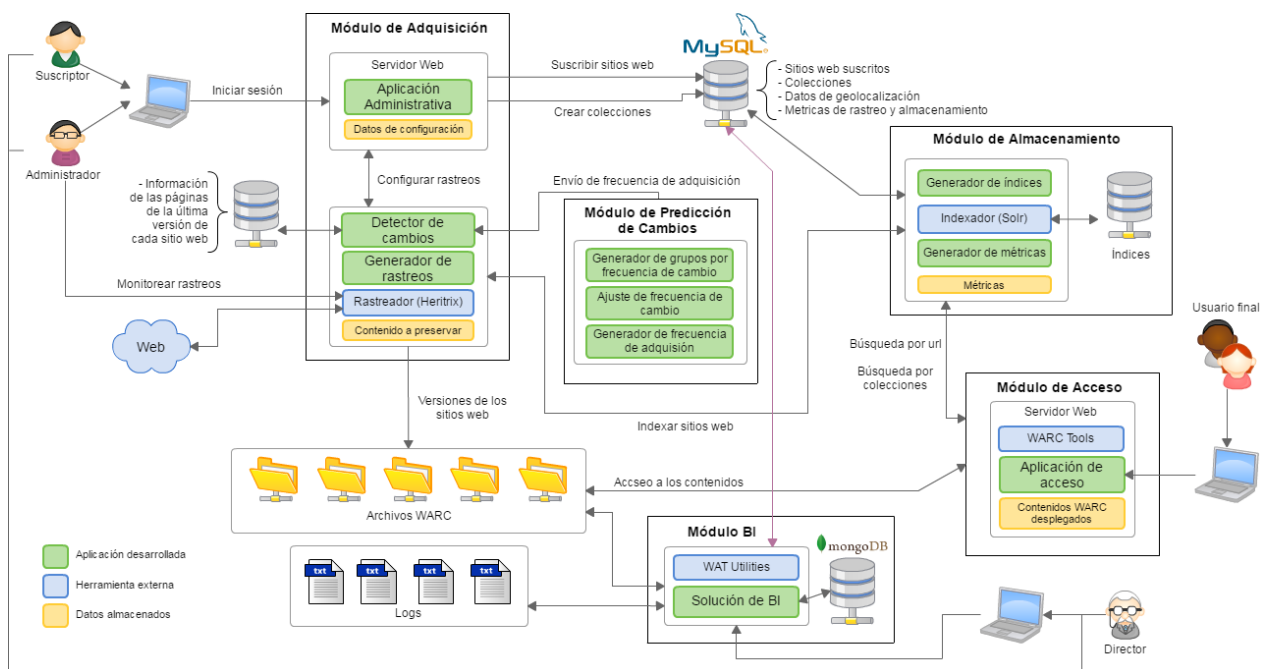


Figura 1 - Primera versión de Arquitectura del prototipo

Para resolver las limitaciones mencionadas previamente es necesario distribuir la carga de trabajo del módulo y agregar tareas que optimicen el trabajo, esta es la motivación para la versión 2 del Archivo Web de Venezuela, partiendo de una nueva arquitectura que se muestra en la Figura 2, que es la evolución de la primera, con la inclusión de nuevos módulos y herramientas para hacer un trabajo más complejo y rápido, entre estas está el almacenamiento y procesamiento distribuido que se traduce en mayor capacidad que a su vez es la manera en que se pretende conseguir mejorar los tiempos de respuesta al usuario final. En la nueva arquitectura el módulo de acceso ya no procesará los archivos WARC ni usará la herramienta WarcTools sino que debe comunicarse con el módulo de indexación para consultar por un recurso y con la respuesta obtenida consultar al módulo de almacenamiento para obtener así la información ya procesada de los archivos que anteriormente se procesaban en el módulo de acceso, como se muestra en la Figura 2 - Nueva arquitectura del prototipo. Además, el mismo módulo de acceso está dividido en dos sistemas altamente acoplados pero a su vez independientes entre sí, siguiendo la idea de distribuir las tareas, estos dos sistemas serán el Back-end y el Front-end, desarrollados en Django y ReactJS respectivamente, el primero se encargará de la comunicación entre el Front-end y el resto del sistema y el segundo se encargará de brindarle la interacción al usuario con la que hará solicitudes al archivo web.

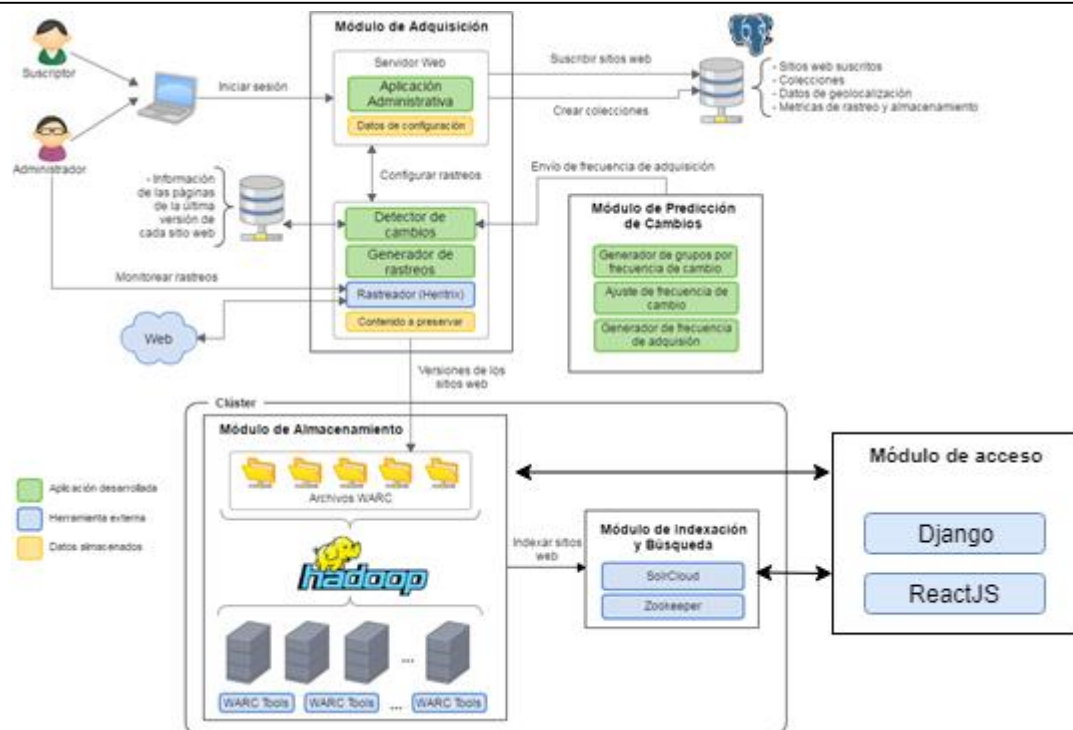


Figura 2 - Nueva arquitectura del prototipo

La finalidad del presente trabajo es describir y explicar todos los elementos que se tomaron en cuenta para el desarrollo del nuevo módulo de acceso para el prototipo de archivo web a la vez de hacer registro de su proceso de desarrollo. Vale la pena aclarar que la creación de este módulo dejará en desuso a su versión previa.

1.2. Alcance de la aplicación

El proyecto permitirá a sus usuarios acceder al contenido Web previamente almacenado en formato WARC y desplegarlo recreando el sitio original de acuerdo a una versión en el tiempo seleccionada. El prototipo está orientado a almacenar sitios web de Venezuela y no se debe esperar encontrar sitios web de otra índole. Una limitante es la capacidad de reproducir fielmente el sitio archivado, ya sea por la naturaleza de la información presentada (streaming, información presentada en tiempo real u otra forma de despliegue compleja de presentar y almacenar) o por la manera en que fue almacenado, la capacidad de representación está limitada a la naturaleza técnica de los recursos y las herramientas de preservación. Otra limitante es de origen temporal, puede que el usuario desee acceder a una versión de un sitio web que no ha sido almacenada sencillamente porque esto sucedió previo al funcionamiento del sistema por lo que el mismo nunca alcanzó a almacenarlo. También existen limitaciones por parte de la información accesible, ya que los sistemas de archivos web son escasos, aunque antiguos, hay deficiencia del material disponible al que se puede referir.

1.3. Objetivos del Trabajo Especial de Grado

1.3.1. Objetivo general

Desarrollar el módulo del acceso y despliegue para la versión 2 del "Archivo web de Venezuela".

1.3.2. Objetivos específicos

1. Determinar los requerimientos funcionales y no funcionales del módulo de acceso y despliegue de archivos WARC.
2. Desarrollar el módulo empleando una adaptación de la metodología AUP con AgilUS y una arquitectura basada en componentes.
3. Integrar el módulo desarrollado al resto de los módulos pertenecientes a la arquitectura del prototipo de archivo web.
4. Realizar pruebas de usabilidad e integración del módulo.
5. Realizar pruebas de rendimiento para hacer comparación con los resultados del módulo de acceso de la versión 1 del prototipo.

CAPÍTULO 2

MARCO CONCEPTUAL

2.1. Patrimonio cultural

Una de las características primordiales a la que la humanidad debe su desarrollo es a la capacidad de transmitir información, tanto a sus coetáneos, pero, más importante aún, a través del tiempo, esto gracias a la lectura y escritura (con la cultura como base) y también a la voluntad de las personas a través del tiempo de preservar la información para su posterior uso. Los primeros instrumentos de preservación fueron la transmisión oral y luego la escritura desde las formas más primitivas hasta los libros. Las personas a diferentes niveles organizacionales son las que han tenido la tarea de hacerla preservación, desde las tradiciones populares, celebraciones, relatos hasta los archiveros, las bibliotecas y almacenes. Hasta la llegada de las computadoras todo este trabajo tenía una característica manual, pero con su desarrollo se les delegó cada vez más responsabilidades, desde la escritura de la información en un formato digital hasta su almacenamiento y su visualización.

En la era actual, el fenómeno de la globalización junto a las tecnologías de internet ha abierto el camino para la era de la información, en la cual, el fácil acceso y generación de contenido a través del internet además del abaratamiento de las computadoras cada vez mayor han dado cabida a una era en la cual constantemente se están generando grandes volúmenes de datos a una tasa sin precedente. Las páginas web es uno de los formatos más comunes en los que estos datos se presentan, además acoge a otros formatos y los presenta dentro de sí. Hay una característica preocupante de las páginas web porque la información que en ellas se presenta tiende a variar constantemente, estudios indican que la vida media de una página web es de dos años, esto solo considerando la disponibilidad de la URL, sin tomar en cuenta la información desplegada, Cho y García Molina (2000) encontraron una vida media de 50 días para las páginas, lo que coloca a información potencialmente valiosa en riesgo.

Debido a esto surge la iniciativa del archivo web, que, cabe destacar, refiere al término de archivar, es decir, almacenamiento histórico como el trabajo realizado en bibliotecas o compañías y no al término archivo como recurso computacional que almacena datos como un conjunto de bits.

2.2. Preservación web

Hoy en día se produce una cantidad de datos enorme a nivel mundial de origen digital y estos vienen en una gran variedad de formatos: textos, bases de datos, imágenes, audios, videos, programas informáticos, páginas web, entre otros. Para las instituciones y organizaciones que tienen a su cargo la recolección y preservación del patrimonio cultural, tratar de definir qué elementos deben conservarse para las generaciones futuras y cómo proceder en su selección y conservación, se está volviendo un problema apremiante y difícil.

La información digital producida hoy en día, en prácticamente todas las áreas de las actividades humanas, es concebida para ser consultada utilizando computadoras, por esta razón podría perderse si no se elaboran técnicas y políticas específicas para su conservación.

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)

La tarea de preservar un sitio web es más compleja de lo que parece ya que los elementos a guardar están almacenados en distintos formatos y además en diferentes servidores, independientes entre sí y por lo común, separados entre ellos. Adicionalmente, la capacidad que requiere preservar un sitio web es creciente porque se desea tener una copia fiel de cada versión del sitio, por lo que con cada actualización se creará una nueva versión del sitio almacenado, a todo esto, además se le debe agregar la inmensa cantidad de sitios web existentes, la cual se estima que es una cifra que ronda en los miles de millones. Estos factores hacen que la tarea de preservar la información digital inherentemente difícil.

Como las páginas web son sistemas que se están ejecutando en computadoras, cualquier aplicación o programa computacional es elementalmente un conjunto de archivos, incluyendo las páginas web, entonces preservar una página web en la práctica significa almacenar los archivos que la componen, específicamente serán archivos HTML, Hojas de estilo (CSS), JavaScript, imágenes y videos, los cuales además contienen metadatos del propio archivo como el tiempo de acceso, el tipo MIME y su longitud.

Los problemas asociados a la preservación digital no son únicamente de carácter técnico, sino que también tienen dimensiones sociales y de organización, es por ello que surge el interés de la UNESCO, pues su razón de ser es en parte, fomentar y permitir la preservación y el disfrute del patrimonio cultural, científico e informativo de los pueblos del mundo, y difícilmente podía ignorar el crecimiento y la vulnerabilidad del patrimonio digital (UNESCO, 2003)

La UNESCO ha elaborado una estrategia para promover la preservación digital, que se articula en torno a:

- Un amplio proceso de consultas con los gobiernos, responsables de la formulación de políticas, productores de información, instituciones y expertos encargados del patrimonio, fabricantes de programas informáticos y organismos de normalización.
- La difusión de directrices técnicas.
- La ejecución de proyectos piloto.

- La preparación de un proyecto de carta para la preservación del patrimonio digital, que se someterá a la aprobación de la Conferencia General en su 32ª reunión.

Para la UNESCO la preservación digital debe tener las siguientes características:

- Ser accesible a todo público.
- Disponer de un marco jurídico y técnico que proteja su autenticidad
- Garantizar la protección de información delicada o de carácter privado

2.3. Archivo web

Se define al archivado web como el proceso de coleccionar porciones de la World Wide Web para asegurar que esta información se preservará en un archivo para futuras investigaciones y el uso del público en general.

En un sentido práctico, un Archivo Web es un sistema de información que se compone de un conjunto de módulos orientados al almacenamiento, administración y respaldo de datos e información, estos módulos que integran el Archivo Web interactúan para procesar los datos y dan lugar a información más compleja, que se distribuye de la manera más adecuada posible en una determinada organización, en función de sus objetivos (Ospina Torres & León Luna, 2013). En la Figura 3 se ejemplifica el proceso que realiza un archivo web:



Figura 3 - Definición de Archivo Web

FUENTE: OSPINA, M. (2014) "UN MARCO DE REFERENCIA PARA LA IMPLEMENTACIÓN DE ARCHIVOS WEB"

Las fases del proceso de preservación propuestas por (Masanès, 2006) se muestran en la Figura 4 y son:

- Selección de los contenidos a preservar: Escogencia de subconjunto de páginas o sitios web que se preservarán el Archivo Web, puede ser definiéndolo previamente o por suscripción,
- Adquisición regular de los contenidos: Recolección de las páginas de manera periódica para poder capturar los cambios que se produzcan en estas. Representan la historia de la página.
- Almacenamiento: Asegurar la perdurabilidad de los contenidos adquiridos y sus datos descriptivos.
- Gestión del almacenamiento: Permite acceder de manera rápida y eficaz a los contenidos almacenados, bien sea por URL de la página o sitio web o por palabra clave,
- Acceso: Permite a los usuarios finales consultar y ver las páginas web almacenadas como si estuvieran en su servidor original, de manera que puedan observar su información y comportamiento, además se pueden mostrar métricas o indicadores importantes para los demás usuarios
- Administración: Permite al administrador monitorizar las distintas fases y asegurar el funcionamiento del archivo

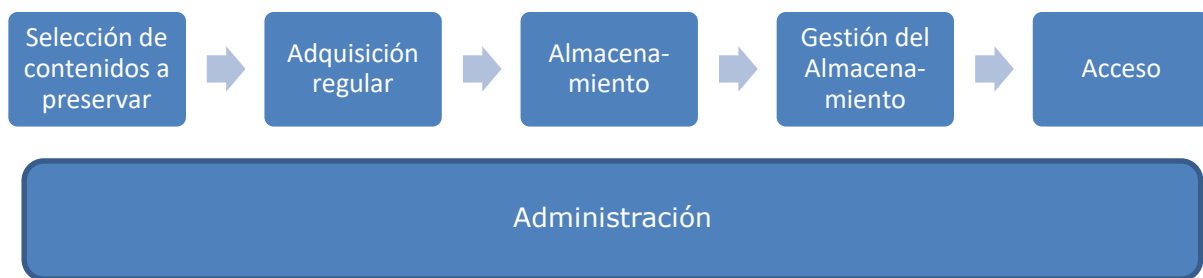


Figura 4 Fases del proceso de preservación web

Fuente: Adaptado de (Masanès, 2006)

Una vez definido el concepto de Archivo Web y el proceso de preservación es importante conocer los formatos de almacenamiento que estos pueden utilizar, el presente trabajo se enfocará en el formato WARC por haber sido escogido en etapas previas del desarrollo del proyecto. También es importante destacar que, en base a la Figura 4 Fases del proceso de preservación web, este proyecto se basa en desarrollar la última fase, el acceso al archivo web, pero su vez depende del almacenamiento y de la gestión de almacenamiento también llamada indexación para poder probarse.

2.4. Formato WARC

El formato WARC (Web ARChive) ofrece una convención para concatenar múltiples registros de recursos, cada uno consiste en un conjunto sencillo de cabeceras de texto y un bloque de datos en un largo archivo. El formato WARC es una revisión del formato ARC que ha sido tradicionalmente usado para guardar rastreadores web como secuencias de bloques de contenidos cosechados del internet.

El formato ARC original es usado internamente por Internet Archive para registrar secuencia de materiales capturados de la web. Cada captura es precedida por una cabecera de una línea que muy brevemente describe el contenido cosechado y su longitud. Este es seguido directamente por los mensajes de respuesta y contenido del protocolo de recuperación. La motivación para revisar el formato surgió de la discusión y las experiencias de el IIPC, cuyos miembros incluyen a IA y las bibliotecas nacionales de una docena de países. El formato revisado se espera que sea una manera estándar para estructurar, gestionar y almacenar billones de recursos colectados de la red. Por ejemplo, WARC será un formato de salida del software de cosecha, como Heritrix y un formato de entrada para un arreglo amplio de catalogado y herramientas de acceso.

El formato WARC generaliza a los formatos antiguos para un mejor apoyo a la cosecha, acceso y necesidades de cambio de las organizaciones de archivado. Además del principal contenido que actualmente registrado, la revisión acomoda contenido secundario relacionado, como los metadatos asignados, eventos de detección de duplicados abreviados y transformaciones de fechas tardías. La revisión puede ser útil para aplicaciones más generales que archivado web.

Un registro WARC va a contener una cabecera la cual va a estar seguida de un bloque de contenido. La cabecera contiene información como la versión del WARC, el tipo de registro, el tamaño de los datos y otros que dependen del tipo del registro.

Este formato permite relacionar y almacenar múltiples recursos u objetos de datos que contienen cabeceras de textos y bloques de datos en un solo archivo de gran tamaño.

Cada archivo WARC contiene uno o más objetos digitales y una vez que el objeto es escrito el archivo se cierra y solo se abre para lecturas, con el formato WARC se busca añadir nuevas características, especialmente en:

- El almacenamiento de las cabeceras de petición del protocolo HTTP.
- El almacenamiento de metadatos arbitrarios.
- La asignación de un identificador para cada archivo de contenido.

El formato WARC está hecho para almacenar todo tipo de contenido Web, ya sea obtenido por el protocolo HTTP o por cualquier otro protocolo. Entre sus principales características se tienen:

- Guarda contenido junto a la información de los protocolos de cosecha.
- Guarda metadatos enlazados a otros datos.
- Permite compresión de datos y preservar integridad de los registros.

2.4.1. Tipos de Registros WARC

Un registro WARC se puede clasificar en 8 tipos de registros:

- **Warcinfo:** describe los registros que le siguen hasta el final del archivo o hasta el siguiente registro del tipo warcinfo. Típicamente aparece al inicio de cada archivo WARC contiene información acerca de la cosecha que generó los siguientes registros. Para un Archivo Web, comúnmente contiene la descripción de un rastreo web, es decir, puede contener información como la duración, el propósito. El formato de la descripción puede contener datos como el tamaño máximo del archivo o la tasa de rastreo.
- **Resource:** contiene la respuesta completa de un protocolo, tal como una respuesta HTTP, incluyendo cabeceras y contenido-cuerpo, de una recuperación de Internet.
- **Request:** contiene una esquematización completa y específica de la petición al servidor incluyendo la información del protocolo de red.
- **Metadata:** registro creado para describir, explicar o acompañar a otro registro. Un registro "metadata" casi siempre se referirá a un registro de otro tipo, teniendo este otro registro el contenido original, o transformado, que había sido recolectado.
- **Revisit:** describe la acción de volver a visitar el contenido previamente archivado e incluye el cuerpo de contenido abreviado que tiene que estar enlazado a algún registro previo. Un registro "revisit" debería ser manejado únicamente cuando, al descifrar un registro, se requiere consultar un registro previo; otros tipos de registros deberían ser preferidos si el registro actual es descifrable por sí solo. No se requiere que algún "revisit" de un URI previamente visto utilice un "revisit", solo aquellos que se refieran a otros registros.
- **Conversion:** contiene una versión alternativa del contenido de otro registro 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 registros de este tipo deben estar relacionados a su correspondiente registro anterior de otros archivos WARC para crear el registro original de tamaño completo. Es usado cuando un registro hace que el WARC exceda el tamaño límite establecido y debe ser separado en segmentos. Un registro de continuación va a necesitar de los parámetros "Segment-Origin-ID" (ID-Segmento-Origen) y "Segment-Number" (Número-Segmento), y normalmente incluye el parámetro "Related-Record-ID" (IDRegistro-Relacionado).

Una vez definidos los elementos que definen y conforman un archivo web se procederá al estudio de antecedentes en el área de preservación web.

2.5. Antecedentes de Archivado Web

Las iniciativas de archivado Web son heterogéneas 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 de proyectos de archivos web se describe en la siguiente tabla:

Tabla 1 Distribución geográfica de las iniciativas de preservación web y espacio usado en disco

Continente / País	Cantidad	%	Espacio Usado en Disco (TB)
África	1	1,18%	
Egipto	1		1000
Asia	7	8,24%	
China	1		42
Israel	1		705
Japón	1		0,3007
Singapur	1		13
República Eslovaca	1		0,996
Taiwán	2		0,336
Europa	43	50,59%	
Austria	2		236
Bélgica	1		
Croacia	1		
República Checa	1		0,140
Dinamarca	1		
Estonia	1		23
Finlandia	1		0,661
Francia	4		22
Francia, Holanda	1		5874,2
Alemania	4		
Grecia	1		56
Islandia	1		
Irlanda	1		420
Letonia	1		
Luxemburgo	1		
Holanda	2		
Noruega	1		40
Portugal	1		20,6
Rusia	1		222,4
Serbia	2		
Eslovenia	1		

Tabla 1 Distribución geográfica de las iniciativas de preservación web y espacio usado en disco

Continente / País	Cantidad	%	Espacio Usado en Disco (TB)
España	3		360
Suecia	1		117,8
Suiza	2		
Suiza, Estados Unidos	2		
Ucrania	1		23
Reino Unido	3		150,587
Norte América	24	28,24%	
Canadá	4		19,435
Canadá, EEUU, Europa	1		
Estados Unidos	19		1859,1
Oceanía	3	3,53%	
Australia	1		260
Corea	1		700
Nueva Zelanda	1		3,8
Sur América	1	1,18%	
Chile	1		
Mundial	6	7,06%	
Mundial	6		
Total general	85	100%	12170,356

Fuente: (Ospina Torres & León Luna, 2013)

El primer proyecto de archivo Web que se halló es CyberCemetery creado en 1995 y que sigue en funcionamiento hoy en día, el año siguiente nacieron cuatro proyectos más, Internet Archive (Archivo de Internet) creado por Brewster Kale en Estados Unidos, Australia's Web Archive (Archivo Web de Australia), Tasmanian's Web Archive (Archivo Web de Tasmania) y Kulturarw3 de Suecia. Seguidos de estos más iniciativas se han ido agregando con el tiempo y ya en la era moderna existen múltiples archivos web a lo largo del mundo con diferentes propósitos.

Algunos de los Archivos Web más representativos serán explicados a continuación:

2.5.1. CyberCemetery

Creado por la Universidad del Norte de Texas y la U.S. Government Printing Office (Oficina Gubernamental de Impresión de los Estados Unidos) como parte del Programa Federal de la Biblioteca Depositorio, creando una asociación para proveer acceso público permanente a los sitios web y publicaciones de las agencias gubernamentales de los Estados Unidos y comisiones que han dejado de funcionar. Esta colección fue llamada "CyberCemetery"

El CyberCemetery es un archivo de sitios web gubernamentales que han cesado sus operaciones (usualmente sitios web de agencias gubernamentales y comisiones extintas que han publicado un reporte final). Esta colección ofrece una variedad de temas indicadores de la amplia naturaleza de la información gubernamental. En particular, esta colección ofrece sitios web que cubren distintos tópicos apoyando planes de estudio universitario y fortalezas de programas particulares. (CyberCemetery, 2009)

2.5.2. Internet Archive

Es una biblioteca digital sin fines de lucros que fue fundada el 12 de mayo de 1996 en San Francisco, su misión es proporcionar "acceso universal a todo el conocimiento" (Internet Archive, s.f.). Provee acceso público y gratis a colecciones de materiales digitalizados incluyendo sitios web, aplicaciones y juegos, música, películas/videos, imágenes en movimiento y cerca de tres millones de libros de dominio. Para 2016, Internet Archive tiene oficialmente 273 miles de millones de páginas web guardadas, tomando 15 petabytes de almacenamientos (Goel, 2016)

Fundado por Brewster Kahle, la organización provee acceso a más de 20.000 colecciones (Internet Archive, 2017), brevemente se mencionarán las principales, clasificadas en tres tipos:

- Colecciones principales:
 - Moving Image Archive
 - eBooks and Texts
 - Audio Archive
 - The Internet Archive Software Collection
 - Web Crawls
 - Open Educational Resources
 - Images
 - Data Collection
 - Our Media
- Colecciones de la comunidad:
 - Community Audio
 - Community Video
 - Community Texts
- Colecciones de imágenes:
 - Brooklyn Museum
 - Brooklyn Navy Yard
 - Community Wireless
 - Cover Art Archive
 - Flickr Commons Archive
 - Geograph Britain and Ireland

- Mathematics
- Occupy Archive
- Occupy Wall Street Flickr Archive
- SharedMap
- USGS Maps

Las tecnologías usadas por Internet Archive son Heritrix, Wayback Machine, NutchWAX y otras herramientas desarrolladas por Internet Archive.

2.5.2.1. Wayback Machine

Wayback Machine es el servicio que más destaca de Internet Archive. Es un archivo web, lanzado el 24 de octubre de 2001 por Brewster Kahle y Bruce Gilliat, y mantenido por Alexa Internet, en palabras de Kahle "Para la fecha, el archivo (Internet Archive) ha catalogado más de diez mil millones (10.000.000.000) ... Por nuestro quinto aniversario, estamos abriendo el archivo al público lanzando Wayback Machine, para que todos puedan regresar en el tiempo y mirar el Internet como si fuera el futuro pasado" (Myanmar Library, 2001)

Alexa Internet en cooperación Internet Archive, creó un índice tridimensional que permite navegar documentos web en múltiples periodos de tiempo y volvieron esta característica única Wayback Machine. (Internet Archive, 2014)

Limitaciones

El servicio presenta limitaciones técnicas, hecho que ha generado problemas legales con Internet Archive, estas limitaciones son:

- Limitaciones sobre la imagen del sitio desplegado.
 - Realiza cambios en los archivos HTML.
 - Las fechas no pueden no coincidir con lo presentado.
 - No puede alcanzar texto contenido dentro de imágenes.
 - En general, no puede ver páginas que dependan de scripts (con ciertas excepciones).
 - Puede mostrar una falsa representación del sitio web, si este depende de tecnologías dinámicas (i.e. tecnologías que producen un resultado luego de solicitar información a una base de datos o un script en el backend).
- Puede incluir enlaces al sitio actual. Cuando referencian objetos al sitio web existente, se está accediendo a información que existe hoy en día, no a la fecha que se está consultando desde Wayback Machine (Donahue, 2008)

2.5.3. Pandora

Es el archivo web nacional para la preservación de publicaciones en línea de Australia, la cual fue establecida inicialmente por la Biblioteca Nacional de Australia en el año 1996, y actualmente cuenta con la colaboración de nuevas bibliotecas estatales y organizaciones culturales australianas.

El nombre Pandora es un retro acrónimo que significa: Preserving and Accessing Networked Documentary Resources of Australia (Preservando y Accediendo Recursos Documentales en Red de Australia).

El archivo Pandora recopila, preserva, y pone a disposición para su visualización los recursos Web australianos. El acceso al archivo está a disposición del público a través del sitio Web de Pandora. Los sitios Web son seleccionados en base a su importancia cultural y el valor de la investigación.

Procedimiento general

Los siguientes tópicos son las pautas que sigue Pandora para su funcionamiento:

- Identificar y seleccionar.
- Contactar al publicador.
- Catalogar.
- Archivar.
- Comparación del publicador y el sitio archivado
- Estadísticas

PANDAS

PANDORA Digital Archiving System, conocidos como PANDAS, es un sistema desarrollado por la Biblioteca Nacional consecuencia de no haber encontrado un sistema comercial que proveyera una solución integrada, web de un sistema administrador de archivo web. Actualmente este sistema cuenta con tres versiones desarrolladas.

PANDAS fue diseñado para sostener el flujo de trabajo diseñado por el equipo de la sección de archivado web de la Biblioteca Nacional, este flujo de trabajo es:

- Identificar, seleccionar y registrar títulos candidatos.
- Pedir y grabar el permiso para archivarlo.
- Establecer los criterios de cosecha.
- Cosechar los archivos.
- Verificación del cumplimiento de los criterios de calidad.
- Realizar proceso de archivado.
- Organizar acceso, mostrar y descubrir rutas y metadatos para los recursos archivados. (PANDORA Australia's Web Archive, s.f.)

2.5.4. Archivo Web de la Biblioteca del Congreso

Es una colección de sitios web agrupados por tema, eventos o área temática. Creado en el 2000 a partir de un piloto para archivo web, se formó un equipo multidisciplinario con el fin de definir las pautas para estas tareas, como propósito de almacenar estos sitios web para futuras investigaciones. Desde el año 2003 forma parte del IIPC junto a las bibliotecas nacionales de Australia, Canadá, Dinamarca, Finlandia, Francia, Islandia, Italia, Noruega, Suecia, Reino Unido y el Internet Archive (EE.UU.), los objetivos de este consorcio es incluir la recolección de un rico conjunto de contenidos de Internet de todo el mundo y fomentar el desarrollo y uso de herramientas comunes, técnicas y estándares que permitan la creación de archivos internacionales.

Herramientas

La Biblioteca del congreso y sus socios han desarrollado un conjunto común de herramientas para el archivado web enfocado en los requerimientos de instituciones que son patrimonio cultural. Las áreas claves de funcionalidad dirigidas por estas herramientas incluyen selección, permisos, adquisición, almacenamiento y acceso.

- **Selección y permisos:** Digiboard, una herramienta que permite al equipo curador seleccionar los sitios web para archivarlos. Adicionalmente facilita la administración de permisos legales para capturar sitios web y/o acceder contenido externo y el proceso de revisión de calidad.
- **Adquisición:** Los archivos web son creados usando la herramienta de rastreo Heritrix.
- **Almacenamiento:** Los archivos web son empacados in paquetes Bagit-comformant usando la biblioteca BagIt.
- **Acceso:** La reproducción del archivo web es habilitada por una instalación local de Wayback Machine.

Requerimientos

- Extraer todo el código, imágenes, documentos, multimedia y otro archivo esencial para reproducir el sitio web tan completo como sea posible.
- Capturar y preservar metadatos técnicos de los servidores y de los rastreadores. Datos de tiempo y fecha son especialmente importantes para distinguir entre las capturas sucesivas de los mismos recursos.
- Guardar el contenido en exactamente la misma forma en que fue entregado. Los archivos HTML y otro código deben permanecer intactos; modificaciones dinámicas deben ser hechas en caliente durante la reproducción del sitio web.
- Mantener la independencia entre la plataforma y el sistema. Los metadatos técnicos no son guardados con algún mecanismo específico del sistema de archivos.

Estándares

Los archivos web son creados y guardados en formato ARC (legado) y en formato contenedor WARC.

Los sitios web son catalogados usando MODS (Metadata Object Description Schema). Palabras clave preliminares, título, y los metadatos de asunto son extraídos de los sitios archivados usando cURL luego evaluados y mejorados por los catalogadores que adicionalmente asignan un nombre controlado y sujetos a las cabeceras. (Library of Congress Web Archives, 2000)

2.5.5. 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) es una organización de miembros dedicados a crear y mejorar herramientas, estándares y buenas prácticas de archivado web mientras se promueve la colaboración internacional y el amplio acceso y uso de archivos web para la investigación y el patrimonio cultural. El estándar de archivo WARC, el rastreador Heritrix y las herramientas analíticas para archivos WARC son todos productos de los grupos de trabajo del IIPC. (IIPC, 2012)

Misión

La misión del IIPC es adquirir, preservar y hacer accesible el conocimiento e información para futuras generaciones en todo el mundo, promoviendo intercambio global y relaciones internacionales

Objetivos

Para cumplir con su misión, el IIPC está trabajando para lograr los siguientes objetivos:

- Habilitar la colección de un cuerpo enriquecido de contenido de internet de alrededor del mundo para ser preservado de una manera que pueda ser archivado, asegurado y accedido en el tiempo.
- Fomentar el desarrollo y uso de herramientas comunes, técnicas y estándares que permitan la creación de archivos internacionales.

Una vez establecido el estado del arte se describirá el proceso de preservación, destacando las actividades involucradas en el desarrollo de este trabajo

2.6. Proceso de preservación

2.6.1. Adquisición

En el proceso de preservación web, la fase de adquisición comprende la etapa en la que se selecciona el sitio web objetivo, se rastrea para seguir sus futuros cambios y se procesa para generar el archivo que será almacenado en la fase de almacenamiento.

2.6.2. Almacenamiento

La fase de almacenamiento comprende el momento en el que se guarda en la base de datos del sistema el archivo que se ha generado sobre la página web. También comprende la etapa en la que se modifica el archivo de un sitio web almacenado consecuencia de que se detectó un cambio en el sitio web objetivo, se volvió a procesar y debe volver a ser almacenado.

2.6.3. Indexación y búsqueda

En la actualidad, el tema de las búsquedas está ampliamente desarrollado y en constante investigación, la causa de la inversión que hay en esto es la necesidad de agilizar los tiempos de búsqueda en los sistemas de computación, consecuencia de esto han nacido los motores de búsquedas, que son en pocas palabras, sistemas especializados en búsquedas eficientes, esto lo logran empleando distintas herramientas como índices y pre procesamiento.

Un motor de búsqueda es un software que busca en documentos o en bases de datos las palabras clave y devuelve los resultados que cumplan con los criterios de búsqueda. Hoy en día, el lugar más empleado para estos sistemas es el internet, específicamente en sitios web como Google o Bing, sin embargo, también se usan implementaciones aisladas del internet, como las utilizadas por los sistemas operativos para la búsqueda de archivos internos. Por lo que se dice que existen múltiples tipos de motores de búsqueda y un gran número de ellos, cada uno con sus propias características. El primer motor de búsqueda que se haya desarrollado se considera Archie en 1990 por Alan Emtage, que fue utilizado para resolver el problema de datos esparcidos combinando un recolector de datos con un comparador de expresiones regulares. Hoy en día, el motor de búsqueda más popular es Google.

2.6.3.1. ¿Cómo funciona un motor de búsqueda?

Un motor de búsqueda Opera en el siguiente orden:

- Rastreador web
- Indexado
- Buscador

Los motores de búsqueda funcionan almacenando información sobre muchas páginas, que obtienen de los archivos HTML. Estas páginas son obtenidas a través de un rastreador, también llamados *arañas*, un navegador web automatizado que sigue cada enlace del sitio.

Los contenidos de cada página son entonces analizados para determinar cómo debe ser indexado.

Datos sobre las páginas visitadas son almacenados en una base de datos de índice para su uso en consultas posteriores. El propósito de un índice es permitir que la información pueda ser encontrada tan rápido como sea posible.

Cuando un usuario hace una consulta en el motor de búsqueda, típicamente usando palabras claves, el motor examina su índice y provee una lista de las páginas web que mejor cumplan con el criterio de búsqueda, usualmente con un resumen corto conteniendo el título del documento y algunas veces, partes del texto. El índice es construido desde la información guardada con los datos y el método con el que la información fue indexada. La mayoría de los motores de búsqueda soportan el uso de operadores booleanos *y*, *o*, *no*, para hacer más específica la consulta. El motor busca palabras o frases iguales a las ingresadas. Algunos motores proveen funciones avanzadas, llamadas *búsqueda de proximidad*, que permite a los usuarios definir la distancia entre las palabras claves. (Jawadekar, 2010)

2.6.3.2. Lucene

Lucene es una biblioteca de software de código abierto para la recuperación de información, permite tanto la indexación como la búsqueda de documentos. Originalmente, fue escrita por Doug Cutting, en septiembre del 2001, y pasó a formar parte de la familia de código abierto de la fundación Jakarta. Desde entonces, debido a su mayor disponibilidad, ha atraído a un gran número de desarrolladores, incluso, empresas como Hewlett Packard, FedEx, etc.

Características de Lucene

A continuación, se detallan algunas características que hacen de Lucene:

- **Origen de datos:** Muchas herramientas de indexación solo permiten indexar ficheros o páginas web, lo que supone un serio inconveniente cuando se tiene que indexar contenido almacenado en una base de datos.
- **Contenido etiquetado:** permite dividir el contenido de los documentos en campos y así poder realizar consultas con un mayor contenido semántico. Esto es, se pueden buscar términos en los distintos campos del documento concediéndole más importancia según el campo en el que aparezca.
- **Concurrencia:** gestiona que varios usuarios puedan buscar en el índice de forma simultánea, así como que un usuario modifique el índice al mismo tiempo que otro lo consulta.
- **Indexación:** Lucene trabaja con índices invertidos para conseguir velocidades de respuestas muy altas, este invierte la estructura de datos “centrada en páginas” en una estructura “centrada en palabras claves”.

Soluciones Basadas en Lucene

Lucene es una biblioteca de software, y por esta razón, no se puede usar instalándose en un computador y ejecutándolo, sino que debe ser utilizado de manera programática en un software, siendo invocado por este para utilizar sus funcionalidades. Se describirá brevemente tres implementaciones de esta, por su interés para el proyecto de prototipo de archivo web:

SOLR

Es una plataforma de búsqueda de código abierto, a partir del proyecto Apache Lucene. Sus características principales incluyen la búsqueda de gran alcance de texto, resaltado de búsqueda de facetas, la agrupación dinámica, la integración de bases de datos, documento enriquecido (por ejemplo, Word, PDF), manipulación y búsqueda geoespacial. SOLR está escrito en Java y se ejecuta como una búsqueda de texto completo del servidor independiente dentro de un contenedor de servlets como Tomcat. SOLR utiliza el Java Lucene, la colección de la búsqueda en su base para la indexación de texto completo y de búsqueda, y tiene como HTTP REST / XML y JSON APIs que hacen que sea fácil de usar desde prácticamente cualquier lenguaje de programación. La poderosa configuración externa de Solr permite tolerar a casi cualquier aplicación sin código Java y posee una extensa arquitectura cuando se requiere de personalización avanzada.

Algunas de las funcionalidades brindadas por Solr son las búsquedas facetadas, estas son subconjuntos de información asociados a los resultados de una consulta que nos permiten agrupar los mismos por distintos campos de los documentos. Como ejemplo de este tipo de búsqueda, se puede exponer el de productos en el que, en donde, además, de recuperar la información sobre cuántos de esos productos pertenecen a cada una de las categorías de productos existentes. Se podrá después afinar la búsqueda de productos, restringiendo los resultados a los productos asociados a dichas categorías.

SolrCloud

SolrCloud es el nombre de un conjunto de nuevas capacidades distribuidas en Solr. La nueva versión de Solr proporciona alta escalabilidad, tolerancia a fallos, las capacidades de indexación y búsqueda distribuida, búsqueda en tiempo real, configuración y gestión centralizada del clúster.

Tabla 2 - Terminología SolrCloud

Cluster	Conjunto de Solr nodos gestionados como una sola unidad
Nodo	Una instancia JVM corriendo Solr

Partición	subconjunto de toda la colección de documentos, Simplemente un solo núcleo Solr
Fragmento	Una partición tiene que ser almacenado en múltiples nodos como se especifica por el factor de replicación. Todos estos nodos forman colectivamente un fragmento. Un nodo puede ser una parte de múltiples fragmentos
Líder	Cada grupo tiene un nodo identificado como su líder. Todas las escrituras de documentos pertenecientes a una partición enrutan a través del líder
Colección	Uno o más fragmentos / Réplicas están dispuestos en Colección
Factor de replicación	Número mínimo de copias de un documento que mantiene el grupo
Registro de transacciones	Un registro sólo de adición de operaciones de escritura mantenida por cada nodo

FUENTE: "Apache Wiki", <https://wiki.apache.org/solr/SolrTerminology>

Ventajas de SolrCloud

Las ventajas que ofrece SolrCloud principalmente se centra en: Las ventajas que ofrece SolrCloud principalmente se centra en:

- **Sharding.** Es el acto de la división de un solo índice Solr en varios equipos, también conocido como fragmentos o rodajas en la terminología Solr. Sharding es más a menudo necesario porque un índice ha crecido demasiado para estar contenido en un solo servidor. Un fragmento dado puede contener múltiples servidores físicos / virtuales, lo que significa que todas las máquinas / réplicas en ese fragmento contienen los mismos datos de índice y sirven consultas para esos datos. A través sharding, puede dividir el índice a través de múltiples máquinas y seguir creciendo sin caer en problemas.
- **Distribución de Consultas y Balanceo de Carga.** Con SolrCloud, la búsqueda distribuida se maneja automáticamente por los nodos en la nube consultar cualquier nodo provocará que el nodo para enviar la consulta a un nodo en todos los demás

fragmentos, devolver una respuesta sólo cuando se ha agregado los resultados de todos los fragmentos. Además, ZooKeeper y todas las réplicas son conscientes de los nodos que no respondieron, y por lo tanto no van a dirigir las consultas a los nodos que se consideran caídos. En el caso de que un nodo caído aún no ha sido detectado por ZK y se envía una consulta, el nodo de consulta reportará el nodo como caído para ZK, y vuelva a enviar la consulta a otro nodo. De esta manera, las consultas a un SolrCloud son muy duraderas y tendrá casi nunca a ser interrumpidas por un nodo derribado.

SolrCloud puede manejar su propio equilibrio de la carga si se utiliza un cliente inteligente como SolrJ. SolrJ utilizará un sencillo equilibrador de carga *round robin*, distribuyendo las consultas de manera uniforme a todos los nodos en SolrCloud. Además, SolrJ es ZooKeeper conscientes y por lo tanto nunca se enviará una consulta a un nodo que se conoce como abajo.

- **Alta Disponibilidad.** En SolrCloud, cuando ZooKeeper detecta un líder caído, se iniciará el proceso de elección de líder de forma instantánea, la selección de un nuevo líder para comenzar a distribuir los documentos de nuevo. Desde el registro de transacciones se asegura de que todos los nodos del fragmento están sincronizados, todas las actualizaciones son. En algunos casos, si en una réplica han perdido demasiados cambios, se llevará a cabo una replicación estándar, así como reproducir el registro de transacciones antes de servir consultas. Se manejará automáticamente los cambios de recorrido y la recuperación cuando un nodo del clúster deja de funcionar, es decir, nunca se perderá las actualizaciones y la ruta automáticamente alrededor de los nodos que no responden.

- **Búsqueda en Tiempo Real y la Indexación Incremental.** SolrCloud cuenta con un líder en cada fragmento, pero el líder es en gran parte lo mismo que cualquier otra réplica, ambos documentos de indexación y de consultas que sirven. La única responsabilidad adicional del líder es la de distribuir los documentos a ser indexados a todas las demás réplicas en el fragmento, y para luego informar que todas las réplicas han confirmado la recepción de un documento dado. Cualquier documento enviado al SolrCloud es redirigido al líder del fragmento apropiado, que entonces lleva a cabo esta responsabilidad. Cuando una réplica recibe un documento, añade el documento a su registro de transacciones y se enviará una respuesta al líder. De esta manera, las actualizaciones de índices SolrCloud se llevan a cabo de una manera distribuida y son duraderas.

Una vez que el documento ha sido añadido a los registros de transacciones, está disponible a través de una *RealTimeGet*, pero no está disponible a través de la búsqueda hasta que se realice un *commit*. Un manual *commit* hará que todos los documentos del registro de transacciones estén disponibles para la búsqueda. También se pueden utilizar los parámetros de confirmación automática y *autoSoftCommit* para desencadenar *commit* de nodos individuales sobre una base regular.

Fragmentos y la Indexación de Datos en SolrCloud

Cuando los datos son demasiado grandes para un nodo, se puede dividir y almacenar las secciones mediante la creación de uno o más fragmentos. Cada uno es una parte del índice lógico o núcleo, y es el conjunto de todos los nodos que contienen esa sección del índice. Un fragmento es una manera de dividir un núcleo de más de un número de "servidores", o nodos. Por ejemplo, se podría tener un fragmento de datos que representa cada estado, o de diferentes categorías que pueden ser registrados de forma independiente, pero a menudo se combinan.

Antes de SolrCloud, Solr apoyó *Distributed Search*, que permite una consulta que se ejecutará a través de múltiples fragmentos, por lo que la consulta se ejecuta contra todo el índice Solr y no hay documentos ya que se perdía a partir de los resultados de búsqueda. Así que la división del núcleo a través de fragmentos no es exclusivamente un concepto SolrCloud. Hubo, sin embargo, varios problemas con el enfoque distribuido que requirió mejoras:

- La división del núcleo en fragmentos fuera algo manual.
- No existía apoyo para la indexación distribuida, lo cual se necesitaba para enviar explícitamente un documento a un fragmento específico; Solr no podía averiguar por sí mismo los fragmentos que se requería para enviar el documento.
- No había ningún equilibrio de carga o la conmutación por error

Con SolrCloud se solucionó estos problemas. Ahora si se cuenta con el apoyo para la distribución tanto en el proceso de índice y como en las consultas de forma automática, y ZooKeeper proporciona equilibrio de conmutación por error y la carga. Además, cada fragmento puede tener réplicas múltiples para robustez adicional.

Los líderes son elegidos de forma automática, en un principio en una base de orden de llegada de primera, y luego en base al proceso de Zookeeper. Si un líder falla, una de sus réplicas es elegida automáticamente como el nuevo líder. Como se inicia cada nodo, se asigna al fragmento con las réplicas de menor cantidad. Cuando hay un empate, se asigna al fragmento con el identificador de fragmento más bajo.

Cuando un documento se envía a una máquina para la indexación, el sistema determina primero si la máquina es una réplica o un líder.

- Si la máquina es una réplica, el documento se envía al líder para su procesamiento.
- Si la máquina es un líder, SolrCloud determina qué fragmento del documento debería ir, envía el documento al líder de ese fragmento, los índices del documento para este fragmento, reenvía la notación índice para sí mismo y para las réplicas.

2.6.4. Acceso

El acceso al archivo web será a través de un sitio web, éste será el módulo de acceso, que debe ser capaz de comunicarse con la base de datos, el módulo de indexación y búsqueda, enlazar los reportes generados por el módulo de inteligencia de negocios además de desplegar la información procesada de los archivos WARC para mostrar las diferentes versiones de los sitios web archivados. Este módulo será el responsable de ser el mediador entre el usuario final y prototipo de archivo web.

El módulo de acceso se implementará a través de una aplicación web, por lo que en la siguiente sección serán descritas con mayor detalle.

2.7. Aplicaciones Web

Una aplicación web es una aplicación de software cliente-servidor en el que el cliente se ejecuta en un navegador web.

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.

2.7.1. Arquitectura Cliente/Servidor

La arquitectura cliente/servidor es una arquitectura de red en el cual cada computadora o proceso en la red es un cliente o un servidor. Los servidores son computadores poderosos dedicados a gestionar archivos o tráfico de red. Los clientes son computadores convencionales en el cual los usuarios corren aplicaciones. Los clientes dependen de los servidores para recursos como archivos, dispositivos e incluso capacidad de procesamiento (Beal, s.f.)(ver **¡Error! No se encuentra el origen de la referencia.**). Lo usual es que el cliente y el servidor estén en equipos distintos y se comuniquen a través de Internet, sin embargo, esto no es excluyente y ambas partes pueden coexistir en el mismo equipo. 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 para que los servicios sean independientes entre sí, situando a cada uno en su plataforma más adecuada.

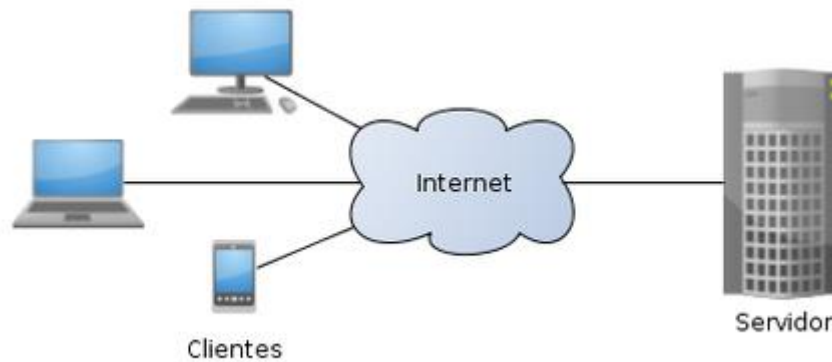


Figura 5 - Arquitectura Cliente/Servidor

FUENTE: <https://pt.wikipedia.org/wiki/Cliente-servidor>

2.7.1.1. Características del cliente

El cliente dentro de una arquitectura cliente-servidor 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.

2.7.1.2. Características del servidor

El servidor posee las siguientes características:

- Espera por las peticiones del cliente, las cuales reciben, procesan para luego generar una respuesta.
- Está orientado para manejar la concurrencia, ya que debe atender peticiones de múltiples clientes a la vez.
- Poseen mecanismos de seguridad adicionales para la aplicación.
- Por la naturaleza de la arquitectura, el servidor debe estar dotado de hardware poderoso dado que él debe responder a los múltiples clientes en tiempos aceptables,

hecho que se logra dotando de poder computacional a la máquina donde se aloja el servidor.

2.7.2. Estado de Transferencia Representacional (ReST)

La World Wide Web opera como un sistema de información interconectado que impone varias restricciones: agentes identifican objetos en el sistema, llamados recursos con identificadores de recursos uniformes (URI por sus siglas en inglés). Los agentes representan, describen y comunican estados de recursos por representaciones del recurso en una variedad de formatos de datos ampliamente entendidos (ejemplo: XML, HTML, CSS, JPEG, PNG). Los agentes intercambian representaciones vía protocolos que usan URIs para identificar y dirigir directa o indirectamente agentes y recursos (W3C, 2003)

Un estilo de arquitectura aún más restringido para aplicaciones web confiables conocido como REST (Representation State Transfer) ha sido propuesto por Roy Fielding. La web REST es el subconjunto de WWW (basado en HTTP) en que los agentes proveen *interfaces semánticamente uniformes* (esencialmente crear, recuperar, actualizar y eliminar) en vez de interfaces específicas a las aplicaciones, y manipulan recursos solo por el intercambio de *representaciones*. Además, las interacciones REST son “sin estado” en el sentido que el significado de un mensaje no depende del estado de la conversación. (W3C, 2004)

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.

Las propiedades de REST son:

- Desempeño: la interacción entre los componentes puede ser el factor dominante en el desempeño percibido por el usuario y la eficiencia de red.
- Escalabilidad: para soportar gran cantidad de componentes e interacciones entre componentes, el autor Roy Fielding ha descrito el efecto de la escalabilidad como sigue:
 - La separación en REST de lo que concierne al cliente-servidor simplifica la implementación de componentes, reduce la complejidad de la semántica de los conectores, mejora la efectividad del desempeño de la sintonización e incrementa la escalabilidad de componentes puros del servidor. Las restricciones de los sistemas en capas permiten a los intermediarios (proxys,

portones y muros de fuego) ser introducidos en varios puntos en la comunicación sin cambiar las interfaces entre componentes, permitiéndoles asistir en la traducción de la comunicación o mejorar el desempeño vía almacenamiento compartido de gran escala. REST permite procesamiento intermediario al limitar los mensajes para que sean auto-descriptivos: la interacción es sin estado entre las solicitudes, métodos estándar y los tipos de media son usados para indicar la semántica e intercambiar información, y las respuestas indican explícitamente almacenamiento.

- Simplicidad de una interfaz uniforme.
- Capacidad de modificar componentes para lograr las necesidades en constante cambio (incluso si la aplicación está en ejecución).
- Visibilidad de comunicación entre componentes por los agentes del servicio.
- Portabilidad de componentes al mover código del programa con datos.
- Confiabilidad es la resistencia a fallo en el nivel del sistema en la presencia de fallos internos de un componente, conectores o datos.

2.7.2.1. Elementos de la arquitectura ReST

ReST posee una serie de elementos importantes para su funcionamiento, estos son:

Conectores

Los conectores representan las actividades implicadas en acceder a recursos y transferir representaciones. Roles proveen una interfaz a ser implementada por los componentes. REST encapsula diferentes actividades de acceso y representaciones de transferencia en diferentes tipos de conectores.

Tabla 3 - Elementos ReST

Tipo de conector	Descripción	Ejemplo
Cliente	Enviar solicitudes, recibir respuestas.	Biblioteca HTTP
Servidor	Recibir solicitudes, enviar respuestas.	Api del servidor web

Caché	Puede ser localizado en el conector del cliente o servidor para almacenar respuestas, puede ser compartida entre varios clientes.	Caché del navegador
Resolver	Transforma el identificador de los recursos en direcciones de red.	Bind (biblioteca de búsqueda DNS)
Túnel	Relé de solicitudes, cualquier componente puede pasar de un comportamiento activo a un comportamiento de túnel.	SOCKS, SSL luego de conexión HTTP

FUENTE: "MR. BOOL", <http://mrbool.com/rest-architectural-elements-and-constraints/29339>

Componentes

Tabla 4 - Componentes REST

Rol de componente	Descripción	Ejemplo
Servidor de origen	Usa el conector del servidor para recibir la solicitud y es la fuente definitiva para representar sus recursos. Cada servidor porvee una interfaz genérica a sus servicios como una jerarquía de recursos.	Apache httpd, Microsoft IIS
Usuario agente	Usa un conector del cliente para iniciar una solicitud y se vuelve el recipiente último de la respuesta	Cualquier navegador como Google Chrome o Mozilla Firefox
Puerta de acceso	Actua como ambos, cliente y servidor para pasar sus solicitudes y respuestas	Squid, CGI, Proxy inverso
Proxy	Actua principalmente como enmascarador del cliente, aunque también puede enmascarar al servidor y cumple la misma tarea de la puerta de acceso	Cualquier prestador del servicio de Proxy,

Rol de componente	Descripción	Ejemplo
		como el proxy del CERN o Gauntlet

FUENTE: "MR. BOOL", <http://mrbool.com/rest-architectural-elements-and-constraints/29339>

Elementos de datos

El aspecto clave de REST es el estado de los elementos de datos, sus componentes se comunican al transferir representaciones del estado deseado de los elementos de datos. Estos son:

Tabla 5 - Elementos de datos

Elemento de datos	Descripción	Ejemplo
Recurso	Cualquier información que pueda ser nombrada es un recurso. Un recurso es un mapeo conceptual de un conjunto de entidades, no la entidad misma. Dicho mapeo puede cambiar a través del tiempo.	En general, un recurso RESTful es cualquier cosa direccionable sobre la web.
Identificador del recurso	Todo recurso debe tener un nombre que lo identifique unívocamente. Bajo HTTP estos son llamados URIs, el cual en un sistema RESTful es un hiperenlace a un recurso. Es el único medio para los clientes y servidores intercambiar representaciones de recursos. La relación entre URIs y recursos es muchos a uno, un recurso puede tener múltiples URIs que proveen diferente información sobre la localización de un recurso	Formato estandarizado de la URI.
Metadatos del recurso	Estos describen el recurso. Los metadatos proveen información adicional como la localización, identificadores alternos del recurso para diferentes formatos o	Enlace de la fuente

Elemento de datos	Descripción	Ejemplo
	información de etiquetas de entidad sobre el recurso mismo	
Representación	Es algo que enviado de ida y vuelta entre clientes y servidores. Entonces, nunca se envía y recibe información, solo sus representaciones. Una representación captura el estado actual o deseado de un recurso. Un recurso particular puede tener múltiples representaciones	Secuencias de bytes, HTML, documentos.
Metadatos de la representación	Describe la representación	Cabeceras
Datos de control	Esto define el propósito de un mensaje entre componentes, como una acción siendo solicitada.	Si-modificado-desde, Si-es-igual

FUENTE: "MR. BOOL", <http://mrbool.com/rest-architectural-elements-and-constraints/29339>

2.7.3. Patrón arquitectónico Modelo-Vista-Controlador

Modelo-vista-controlador (MVC) es un patrón de diseño que asigna a los objetos en la aplicación uno de los tres roles: modelo, vista o controlador. El patrón no solo define los roles que juegan los objetos en la aplicación, define la manera en que estos objetos se comunican entre sí. Cada uno de los tres tipos de objeto está separado de los otros por límites y se comunican con los otros tipos de objetos al otro lado de estos límites (Apple Inc, 2015). El patrón de diseño MVC desacopla estos componentes principales permitiendo reuso eficiente del código y desarrollo paralelo.

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 **iError! No se encuentra el origen de la referencia.**).

- **Modelo:** es el componente central del patrón, expresa el comportamiento de la aplicación en términos de dominio del problema, independiente a la interfaz de usuario-

Gestiona directamente los datos, la lógica y reglas de la aplicación. Un modelo almacena datos que es obtenida de acuerdo a los comandos del controlador y mostrada en la vista.

- **Vista:** genera nuevas salidas para el usuario basada en cambios en el modelo. Puede ser cualquier representación de salida de información, como cuadros o diagramas. Múltiples vistas de la misma información son posibles, como un cuadro de barra para gestionar y una vista tabular para los contadores. 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 recibiendo entradas de información y volviéndola comandos para el modelo o la vista. Obtiene los datos del modelo y se los presenta al usuario como respuesta a sus peticiones a través de la vista. Puede enviar comandos al modelo para actualizar su estado. También puede enviar comandos a sus vistas asociadas para cambiar la presentación de la vista del modelo.

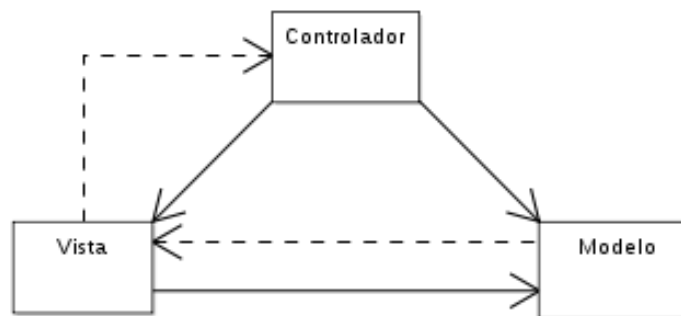


Figura 6 - Arquitectura MVC

FUENTE: <https://es.wikipedia.org/wiki/Modelo-vista-controlador>

2.7.3.1. Objetivos de MVC

- **Desarrollo simultáneo:** como MVC desacopla los distintos componentes de una aplicación. Los desarrolladores son capaces de trabajar en paralelo en diferentes componentes sin impactar o bloquear otro.
- **Reúso de código:** al crear componentes que son independientes entre sí, los desarrolladores son capaces de reusar componentes rápida y fácilmente en otras aplicaciones.

2.7.3.2. Ventajas y desventajas

Ventajas

- **Desarrollo simultáneo:** como ya se explicó anteriormente, permite a múltiples desarrolladores trabajar simultáneamente en el modelo, vista y controlador.
- **Alta cohesión:** MVC permite agrupamiento lógico de acciones relacionadas en un controlador. Las vistas de un modelo específico también se agrupan juntas.
- **Bajo acoplamiento:** la propia naturaleza de MVC es que haya bajo acoplamiento entre modelos, vistas y controladores.
- **Facilidad de modificación:** por la separación de responsabilidades, futuro desarrollo o modificación es más fácil.

Desventajas

- **Navegabilidad del código:** la navegación del patrón puede ser compleja porque agrega nuevas capas de abstracción y requiere que los desarrolladores se adapten al criterio de descomposición de MVC.
- **Consistencia multi-artefacto:** descomponer una característica entre causa esparcimiento por lo que requiere que los desarrolladores mantengan la consistencia de múltiples representaciones juntas.
- **Curva de aprendizaje pronunciada:** conocimiento de múltiples tecnologías se hace norma.

2.7.3.3. Frameworks basados en MVC

En programación, un *framework* es una abstracción en la cual el software que provea funcionalidades genéricas puede ser modificado selectivamente por código adicional escrito por el desarrollador, haciéndolo un software de una aplicación específica, es decir, sacrificando la característica genérica por especialización, dicho en otras palabras, es una plantilla o diseño que tiene el fin de ser reutilizado como base para desarrollar aplicaciones especializadas y que está compuesto por un conjunto de clases abstractas y las instancias de las mismas.

Los *frameworks* fueron evolucionando con el paso de los años y sobretodo con 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, esta abstracción se logra a partir de la creación de objetos abstractos de los que otras clases heredan para redefinir el comportamiento de las respuestas.

Python Django

Django es un Framework Web alto nivel y código abierto que utiliza el lenguaje de programación Python. Su objetivo principal es facilitar el desarrollo de sitios web complejos manejados por bases de datos. Django enfatiza en la reusabilidad y conectividad de componentes, desarrollo rápido y en el principio de *no te repitas* (DRY) que significa "Cada

pieza del conocimiento debe tener una sola, no ambigua, representación autoritaria dentro de un sistema, esto, a nivel de aplicación Web, quiere decir que en lo posible no debe haber duplicados del código del software en cada una de sus capas, por ejemplo, código de control que debe ejecutarse en cada vista y que se repite a través de las misma.

Python Django no se considera un framework que siga el patrón arquitectónico modelo-vista-controlador, sino modelo-plantilla-vista (MTV por sus siglas en inglés), la razón de esto es que en su interpretación de MVC, la vista describe los datos que son presentados al usuario. No es necesariamente como se *ven* los datos, sino *qué* datos se están presentando. La vista describe que datos se ven, no como son vistos. Es una distinción sutil.

Entonces, en este caso, la “vista” es la función de retorno de Python para una URL en particular, porque la función de retorno describe que datos son presentados.

Además, es sensible a separar contenido de presentación, que es donde la plantilla viene a hacer efecto. En Django, una “vista” describe que datos son presentados, pero una vista normalmente delega a una plantilla, que describe como los datos son presentados.

¿Dónde cabe el “controlador entonces? En el caso de Django, es probablemente el framework en sí: la maquinaria que envía una solicitud a la vista apropiada, de acuerdo a la configuración URL de Django. (Django Software Foundation, 2013)

Ahora bien, esta es la interpretación conceptual que tiene el equipo de Django sobre su arquitectura y para fines prácticos, hay que definir formalmente las funciones de cada elemento de este patrón:

- **Modelo:** es la capa de acceso. Esta capa contiene todo sobre los datos: como accederla, como validarla, qué comportamiento tiene y las relaciones entre los datos.
- **Plantilla:** la capa de presentación. Esta capa contiene las decisiones relacionadas con la presentación: como debería ser mostrada la página web u otro tipo de documento.
- **Vista:** la capa de lógica de negocio. Esta capa contiene la lógica que accede al modelo y difiere a la plantilla apropiada. Se puede ver como un puente entre los modelos y las plantillas. (Django Book, 2016)

Una aplicación en Python posee los siguientes archivos que deben ser comprendidos:

- **__init__.py:** es un archivo vacío que le dice a Python que debe tomar un determinado directorio como un paquete Python. Un paquete es una manera de estructurar el espacio de nombres de Python utilizando nombres de módulos con puntos, es decir, un módulo A.B designa un submódulo llamado B dentro de un paquete llamado A.

- **manage.py:** contiene código que permite interactuar con el proyecto de Django de varias maneras. A través de este se puede instalar y crear el modelo físico de la Base de datos definido por un modelo creado con lenguaje Python.
- **views.py:** en este archivo se declaran las funciones del controlador que permiten procesar los datos. Son funciones escritas en Python.
- **settings.py:** este archivo contiene las configuraciones necesarias para el proyecto.
- **urls.py:** es un archivo que posee las rutas del proyecto y es manejado por URLconf.
- **models.py:** es el archivo en el que se declaran las clases del modelo, definiendo tipos de datos y relaciones de pertenencia.

Otras características importantes de Python Django son las siguientes:

- Django posee un cache que se integra con la cache de memoria (memcached) y otros sistemas similares.
- Posee un framework de sindicación que permite crear feeds RSS a través de clases Python.
- Puede generar interfaces de administración automáticamente, de esta manera las funciones CRUD que solo puede ejecutar el administrador de la aplicación se vuelven mucho más fáciles. Esto además ofrece como ventaja que el usuario Administrador no tiene que ser un experto en programación.

2.7.4. NodeJS y NPM

Desarrollado en 2009 por Ryan Dahl, NodeJS es un ambiente de ejecución código abierto que ejecuta código JavaScript del lado del servidor (Training.com, 2016). Históricamente, JavaScript era usado principalmente para *scripts* del lado del cliente, los cuales eran embebidos en el documento HTML del sitio web y ejecutados por el motor JavaScript en el navegador del cliente. NodeJS permite a los desarrolladores usar JavaScript del lado del servidor, lo cual produce contenido dinámico en las páginas web **antes** de ser enviado por el navegador del usuario.

NPM a su vez es un gestor de paquetes para JavaScript. NPM es incluido como una característica del instalador de Node.js. Consiste en una aplicación cliente a través de la línea de comandos que interactúa con un registro remoto, permitiéndole al usuario consumir y distribuir módulos que están disponibles en el registro en línea.

2.7.4.1. ReactJS

Es una librería de JavaScript para construir interfaces de usuario desarrollada y mantenida por Facebook. Se basa en componentes y las principales características son:

- Enlace de datos en una unidireccional: las propiedades son pasadas a un componente desde el padre, estas propiedades son recibidas como un conjunto único de valores inmutables. Siempre que alguno de estos valores la función de renderizado del componente es llamada permitiendo al componente mostrar sus cambios.
- Virtual DOM: el uso de un “Modelo de objetos del documento virtual” permite a React crear una estructura de datos caché, computar la diferencia resultante y luego actualizar el DOM del navegador eficientemente (ReactJS, s.f.). Esto permite a los desarrolladores escribir código como si la página entera es mostrada en cada cambio, mientras que la librería solo actualiza los subcomponentes que realmente cambian.
- Métodos del ciclo de vida: estos métodos son ganchos que permiten la ejecución de código en ciertos puntos durante el ciclo de vida del componente.
- JSX: JavaScript XML es una extensión de la sintaxis de JavaScript (Facebook, s.f.), similar a HTML provee una manera de estructurar el despliegue del componente usando sintaxis similar.

Para lograr el concepto de React de flujo de datos unidireccional, la arquitectura Flux presenta una alternativa popular a la arquitectura MVC.

2.7.4.2. Flux/Redux

Flux presenta acciones que son enviadas a través del *despachador central* a una *tienda* y estos cambios son propagados nuevamente a la vista. Flux es una variante del patrón de diseño observador (Johnson, s.f.) . El propósito del comportamiento de Flux es la inmutabilidad, a través de las acciones, transmiten el mensaje a la tienda la cual actualiza su estado con los datos recibidos en la acción, el estado se actualiza creando una nueva estructura de datos con los datos nuevos y asignando esta nueva estructura a la tienda, es decir, no modifica los datos que tiene, sino que crea una copia de estos más la nueva información, siendo así inmutable.

Redux es a su vez una librería de JavaScript que es una implementación de Flux, en síntesis, ofrece una API limitada para gestionar el estado de la aplicación. Está influenciado por la programación funcional.

CAPÍTULO 3

MARCO METODOLÓGICO

El siguiente capítulo describe las metodologías utilizadas en el proyecto y las compara para luego ser aplicadas en el proceso de desarrollo y ser descrita su aplicación en el marco aplicativo.

3.1. Metodologías ágiles

El desarrollo de software ágil es un término paraguas para un conjunto de métodos y prácticas basadas en los valores y principios expresados en el Manifiesto ágil.

Las soluciones evolucionan a través de colaboraciones entre equipos auto organizados, interfuncionales utilizando las prácticas apropiadas para su contexto. (Agile Alliance, s.f.)

Principios

1. La mayor prioridad es satisfacer al cliente a través de entregas tempranas y continuas de software valioso.
2. Los requerimientos cambiantes son bienvenidos, incluso en desarrollo avanzado. Los procesos ágiles aprovechan el cambio para la ventaja competitiva del cliente.
3. Entrega de software funcional constante, de un par de semanas a un par de meses con la preferencia de la menor escala posible.
4. Personal de negocios y desarrolladores deben trabajar juntos diariamente durante todo el proyecto.
5. Construir proyectos alrededor de individuos motivados. Suministrar el ambiente y apoyo que necesitan y confiar que harán el trabajo.
6. El método más eficiente y efectivo de transmitir información hacia y entre un equipo de desarrollo es en una conversación cara a cara.
7. Software funcional es la principal métrica de progreso.
8. Los procesos ágiles promueven desarrollo sostenible. Los patrocinantes, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante indefinidamente.
9. Atención continua a excelencia técnica y buen diseño mejora la agilidad.
10. La simpleza – el arte de maximizar el monto de trabajo hecho – es esencial.
11. Las mejores arquitecturas, requerimientos y diseños emergen de equipos auto organizados.
12. En intervalos regulares, el equipo refleja como volverse más efectivo, entonces se afina y ajusta su comportamiento acorde a ello.

En la siguiente sección se describirán y compararán dos metodologías de desarrollo ágil

3.1.1. Ágil UP

En un principio, es una versión simplificada de la metodología RUP (Rational Unified Process) la cual describe un acercamiento sencillo al desarrollo de aplicaciones software de negocio utilizando técnicas y conceptos ágiles manteniéndose fiel a RUP (Ambler, 2005).

La Figura 7 muestra el ciclo de vida de Ágil UP:

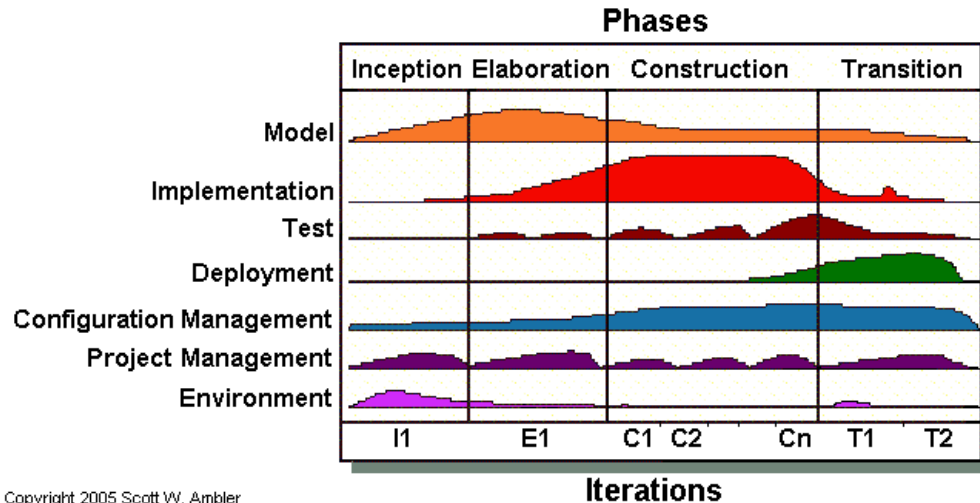


Figura 7 - Ciclo de vida AUP

FUENTE: "Ambysoft", <http://www.ambysoft.com/unifiedprocess/agileUP.html>

3.1.1.1. Fases

- **Concepción:**

Identificar el alcance inicial del proyecto, una arquitectura potencial para el sistema, obtener la financiación inicial del proyecto y la aceptación de las partes interesadas.

- **Elaboración:**

Probar la arquitectura del sistema.

- **Construcción:**

Elaborar el software ejecutable de manera incremental cumpliendo con las necesidades de mayor prioridad del proyecto.

- **Transición:**

Validar y desplegar el sistema desarrollado en un ambiente de producción.

3.1.1.2. Principios

La metodología Ágil UP está definida sobre los siguientes principios

- **Tu equipo sabe lo que hace:**

Los miembros del equipo no van a leer la documentación detallada del proceso, pero recibirán una guía de alto nivel o el entrenamiento necesario

- **Simpleza:**

Todo debe ser descrito concisamente usando las palabras mínimas necesarias

- **Agilidad:**

Se basa en los principios de desarrollo ágil:

- Desarrollo iterativo e incremental
- Pruebas unitarias continuas
- Programación en parejas
- Frecuente interacción con el cliente
- Corrección de todos los errores
- Refactorización del código
- Simplicidad del código

- **Enfoque en actividades de alta prioridad:**

Se enfoca en las actividades que son importantes, no cada posible caso que pudiera suceder en el proyecto.

- **Independencia de las herramientas:**

Se puede utilizar cualquier herramienta que sea necesaria, la sugerencia es usar las herramientas que mejor se adecuan para el trabajo, que suelen ser herramientas simples o de código abierto.

- **Producto a la medida:**

Debe ser fácilmente ajustable vía cualquier herramienta de edición HTML, sin necesitar la compra de alguna herramienta especial.

3.1.1.3. Disciplinas

Las disciplinas son realizadas de manera iterativa, definiendo las actividades que realizan los miembros del equipo para construir, validar y entregar el software funcionar que cumpla las necesidades. Estas son:

- **Modelo:**

El objetivo es comprender la organización del negocio, el problema que aborda el proyecto e identificar una solución viable al mismo.

- **Implementación:**

El objetivo de esta disciplina es transformar el modelo en código ejecutable y realizarle pruebas básicas y en particular, unitarias.

- **Prueba:**

Realizar una evaluación de objetivos para asegurar calidad. Esto incluye encontrar defectos, validar que funcione como es debido y verificar que los requerimientos se cumplen.

- **Despliegue:**

Planear la entrega del sistema y ejecutar un plan para hacer el sistema disponible para usuarios finales.

- **Gestión de configuración:**

Gestionar el acceso para los artefactos del sistema. Esto no solo incluye seguimientos de versiones sino controlar y gestionar los cambios hacia ellos.

- **Gestión de proyecto:**

Dirigir las actividades que toman lugar en el proyecto. Esto incluye gestión de riesgos, dirigir miembros y coordinar con personas y sistemas fuera del alcance del proyecto para asegurar que será entregado a tiempo y dentro del presupuesto.

- **Ambiente:**

Apoyar el resto del esfuerzo asegurando que el acceso apropiado, la guía y las herramientas están disponibles para el equipo cuando se necesitan

3.1.2. AgilUS

La metodología AgilUs nace como línea de investigación del Centro Ingeniería de *Software* y Sistemas (ISYS) de la Escuela de Computación de la Universidad Central de Venezuela. Creada por Alecia Acosta, se basa en el concepto de usabilidad. Se fundamenta en el análisis centrado en el usuario y en la participación de especialistas, con el objetivo de evolucionar el *software*, a fin de que éste alcance el mayor grado de usabilidad una vez culminado su desarrollo. (Acosta Alecia, 2011)

AgilUs es un método de desarrollo iterativo e incremental que pone el mayor peso del desarrollo en la consecución de la usabilidad, haciendo énfasis en el desarrollo de las interfaces desde etapas iniciales. Esta metodología reduce la cantidad de actividades y artefactos que se generan, propicia la participación del usuario y se adapta a los cambios que puedan surgir en

el desarrollo. Está orientada al desarrollo de sistemas con alto grado de interactividad con los usuarios (*front-end*) y menos complejidad en la lógica de la aplicación (*back-end*).

AgilUs provee un conjunto de “buenas prácticas” para el desarrollo de *software* desde una perspectiva ágil, donde requiere involucrar al usuario de manera activa durante el proceso de desarrollo. Algunas de estas “buenas prácticas” son:

- **Diseño centrado en el usuario (DCU):** es un enfoque de diseño y desarrollo que se centra en los deseos, limitaciones y necesidades de los usuarios finales de un *software*.
- **Diseño basado en prototipos:** el desarrollo de *software* es guiado por la construcción de prototipos de alta fidelidad y la evaluación de los mismos por los usuarios y por especialistas en usabilidad.
- **Desarrollo ágil, incremental e iterativo:** el desarrollo es lo más simple posible, provee resultados sin necesidad de esclarecer todos los requerimientos al inicio del desarrollo y permite regresar a etapas anteriores una vez recibida la retroalimentación de las evaluaciones realizadas.
- **Usabilidad como atributo de la calidad:** se siguen las recomendaciones del estándar ISO/IEC 9126-1 para la producción de *software* usable, ya que la usabilidad es considerada un atributo de la calidad interna y externa del *software*.
- **Interacción continua con el usuario:** la presencia constante y participativa del usuario es fundamental. La usabilidad del sistema sólo puede ser determinada por el usuario.

Acosta (2011) explica que AgilUs centra el desarrollo de *software* en los siguientes principios:

- **Integra la Interacción Humano Computador (IHC) y la Ingeniería de Software (IS):** IS e IHC son complementarias, no son disciplinas excluyentes. Un diseño centrado en el usuario impacta positivamente en la calidad del *software* (ISO 9126-1).
- **La usabilidad debe considerarse desde el principio del desarrollo:** la usabilidad aumenta la calidad del *software*, entonces es conveniente incluirla desde el principio en el desarrollo como uno de los requerimientos fundamentales para impactar positivamente en la calidad del producto final.
- **La usabilidad determina la utilidad:** un *software* se considera útil en la medida que pueda ser usado a fin de producir resultados, en forma eficiente, intuitiva y satisfactoria para los usuarios.
- **El usuario determina la usabilidad:** la usabilidad no es una propiedad abstracta. Un *software* sólo será considerado usable en un contexto específico y por un tipo de usuario específico. El objetivo es lograr que todos los usuarios del *software* encuentren usables las tareas que pueden realizar.

3.1.2.1. Ciclo de vida

“El ciclo de vida de AgilUs hace énfasis en la importancia del usuario y sus evaluaciones. Está basado en el desarrollo iterativo e incremental de prototipos de alta fidelidad hasta que se convierten en el producto final para entrega” (Acosta Alecia, 2011).

AgilUs está formado por cuatro (4) etapas diferentes: Requisitos, Análisis, Entrega y Prototipaje, y en cada una de ellas se incluyen actividades y artefactos poco costosos para la construcción de la usabilidad, como se muestra en la Figura 8.



Figura 8 - Ciclo de vida AgilUS

Copyright 2011 Alecia Acosta.

3.1.2.2. Etapas

A continuación, se describe las cuatro (4) etapas de AgilUs y las técnicas de evaluación de usabilidad utilizadas (Acosta Alecia, 2011):

- **Requisitos:** se realiza el análisis global del problema a solucionar, se estudian productos similares existentes, se genera un perfil de usuario, y se define la lista de requerimientos a desarrollar. Esta etapa es importante en el desarrollo del *software*, ya que un mal análisis de requerimientos traería como consecuencia un *software* que no cumple con las necesidades del usuario.
Las técnicas de evaluación de usabilidad (indagación) que se utilizan en esta etapa son:

- **Tormenta de ideas:** técnica de grupo para la generación de ideas nuevas y útiles, que permitan mediante reglas sencillas aumentar las probabilidades de innovación y originalidad.
 - **Encuesta:** conjunto de preguntas normalizadas dirigidas a una muestra representativa de la población para obtener información estadística acerca de sus opiniones o hechos específicos.
 - **Cuestionario:** lista de preguntas que se distribuye entre los usuarios para obtener información de sus opiniones. Este instrumento se puede utilizar en cualquier etapa del proceso de desarrollo, dependiendo de las preguntas que estén formuladas.
 - **Entrevista:** dialogo entre un entrevistador y un entrevistado, donde el entrevistador hace una serie de preguntas al entrevistado con el fin de conocer sus opiniones e ideas.
 - **Evaluación de sistemas existentes:** consiste en la revisión de versiones anteriores del mismo sistema, así como sistemas de la competencia o afines, con el objetivo de identificar ventajas, desventajas y problemas de usabilidad.
 - **Perfil de usuario:** describe a los usuarios previstos del sistema, detallando sus características más relevantes.
 - **Requerimientos funcionales y no Funcionales:** recopilación de condiciones o capacidades que debe cumplir el sistema.
- **Análisis:** se lleva a cabo el análisis de la solución a desarrollar, se emplean diagramas de casos de uso y modelo de objetos del dominio, siguiendo la notación UML, para definir las funcionalidades que tendrá el producto a desarrollar.

Las técnicas de evaluación de usabilidad que se utilizan en esta etapa son:

- **Prototipo en papel:** bosquejo del sistema que se desea realizar. Resulta muy útil cuando el cliente no tiene claros algunos puntos.
- **Guía de estilo:** documento que recoge normativas y patrones básicos relacionados con el aspecto de la interfaz para su aplicación en el desarrollo de nuevas pantallas dentro de su entorno.
- **Modelo de casos de uso:** modelo que permite especificar las funcionalidades y el comportamiento del sistema mediante su interacción con usuarios u otros sistemas.
- **Modelo de objetos del dominio:** modelo que describe y prioriza cada uno de los objetos identificados en el sistema.
- **Objetos de interfaz:** modelo que permite comprender la parte visual de la aplicación.
- **Patrones de interacción:** describen aspectos concernientes a la interfaz de usuario; están orientados a presentar soluciones apropiadas a problemas recurrentes que se les presentan a los usuarios cuando utilizan las aplicaciones interactivas.

- **Prototipaje:** se implementa un prototipo rápido de la interfaz de usuario a partir de los patrones de interacción, el cual va evolucionando hasta convertirse en el producto final y se realizan evaluaciones de usabilidad apropiadas a esta etapa: evaluaciones heurísticas y listas de comprobación.

Las técnicas de evaluación de usabilidad que se utilizan en esta etapa son:

- **Prototipo rápido:** técnica basada en ordenador que pretende reducir el ciclo iterativo de desarrollo. Los prototipos iterativos desarrollados podrán ser rápidamente reemplazados o modificados según los informes procedentes de otras evaluaciones a medida que se evoluciona en el desarrollo de las tareas a realizar. Existen muchas herramientas para la generación de prototipos rápidos, siendo habituales una secuencia de imágenes en Microsoft PowerPoint o Visual Basic.
- **Evaluación heurística:** se basa en una inspección donde los especialistas juzgan si cada elemento de la interfaz sigue los principios de usabilidad establecidos.
- **Lista de comprobación:** consta de la verificación del cumplimiento de una lista de principios que el equipo de desarrollo acuerda para el diseño de la interfaz de usuario.
- **Pensamiento en voz alta:** técnica donde se les pide a los usuarios que expresen en voz alta sus pensamientos, sentimientos y opiniones mientras que interaccionan con el sistema o un prototipo del mismo.
- **Prototipo ejecutable:** técnica donde el prototipo utilizado para las pruebas se corresponde con la interfaz real en la mayor medida posible. Es utilizado para maquetar la interfaz y en él son aceptadas entradas desde ratón o teclado, tal como lo haría la interfaz real, a su vez que responde a esos eventos de idéntica forma (mostrando ventanas, mensajes de respuesta, entre otros).
- **Entrega:** se aplican las pruebas al sistema para certificar que la aplicación desarrollada sea un *software* usable y sin errores, finalmente se pone en producción la aplicación.

Las técnicas de evaluación de usabilidad que se utilizan en esta etapa son:

- **Pruebas de aceptación:** pruebas que aportan datos tanto cuantitativos como cualitativos sobre usuarios reales que llevan a cabo tareas reales con el sistema. Los profesionales de la usabilidad pueden evaluar algunos aspectos de la accesibilidad utilizando protocolos normalizados de pruebas de usabilidad.
- **Protocolo de preguntas:** consiste en la formulación de preguntas directas al usuario acerca del sistema para detectar qué partes de la interfaz resultan obvias y qué otras resultan oscuras.

3.1.3. Comparación de metodologías

	AUP	AUS
--	-----	-----

Método de desarrollo	Iterativo	Iterativo e incremental
Iteración	Disciplinas: 1. Modelo 2. Implementación 3. Prueba 4. Despliegue 5. Gestión de configuración 6. Gestión de Proyecto Ambiente	Etapas: 1. Requisitos 2. Análisis 3. Prototipaje 4. Entrega
Pasos	Fases: 1. Concepción 2. Elaboración 3. Construcción 4. Transición	No está establecido específicamente.
Principios	• Tu equipo sabe lo que hace • Simpleza • Agilidad • Enfoque en las actividades de alta prioridad • Independencia de herramientas • Producto a la medida	• Interacción Humano-Computador • Ingeniería del software • La usabilidad debe considerarse desde el inicio del desarrollo • La usabilidad determina la utilidad El usuario determina la usabilidad

Tabla 6 - Tabla comparativa entre AUP y AUS

3.2. Desarrollo de software basado en componentes

Hay que aclarar que esta metodología es fundamental para el desarrollo del prototipo de archivo web en general, ya que este lo componen múltiples módulos desarrollados por múltiples miembros del equipo por lo que cada módulo debe seguir lineamientos generales con el fin de encausarse todos hacia un mismo objetivo con un orden ideal.

El desarrollo de software basado en componentes (DSBC) depende del reúso de software, surge de la incapacidad del desarrollo orientado a objetos de dar soporte a un reúso efectivo. Las clases con un único objeto son muy detalladas y específicas por lo que los componentes son más abstractos y pueden ser considerados como proveedores de servicios autónomos. Para aclarar esto, un componente se define como una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio (Szyperski, 1998).

El DSBC depende entonces de los componentes, pero adicionalmente los componentes deben ser los más generales posibles y reusables, pero esto implica un costo y un conjunto de características que deben, estos pueden no cumplir con estas y deben pasar por un proceso que los adapte a estas para ser aceptados.

Para construir componentes reusables a partir de componentes comunes se les debe aplicar lo siguiente:

- Remover métodos específicos de la aplicación.
- Cambiar los nombres para hacerlos generales.
- Agregar métodos para ampliar la cobertura.
- Hacer el manejo de excepciones consistente.
- Agregar una interfaz de configuración para la adaptación del componente.
- Integrar componentes requeridos para reducir las dependencias.

Una vez pasan por estas transformaciones, para evaluar si es considerado reusable debe cumplir con:

- Debe reflejar abstracciones de un dominio estable.
- Debe ocultar representaciones de estado.
- Debe ser lo más independiente posible.
- Debe mostrar excepciones a través de sus interfaces.

Entre la usabilidad y la reusabilidad debe existir un balance porque a medida de que la interface es más general, mayor es su reusabilidad pero esto lo hace más compleja y por lo tanto menos usable.

Los componentes genéricos pueden ser menos eficientes en espacio y tomar mayores tiempos de ejecución que sus equivalentes no genéricos, es por esto que su coste de desarrollo puede ser mayor que el costo de componentes específicos equivalentes. El coste de esta característica adicional de reusabilidad debe ser una organización y no un coste del proyecto.

El proceso entonces de DSBC sigue un flujo como se muestra en la Figura 9:

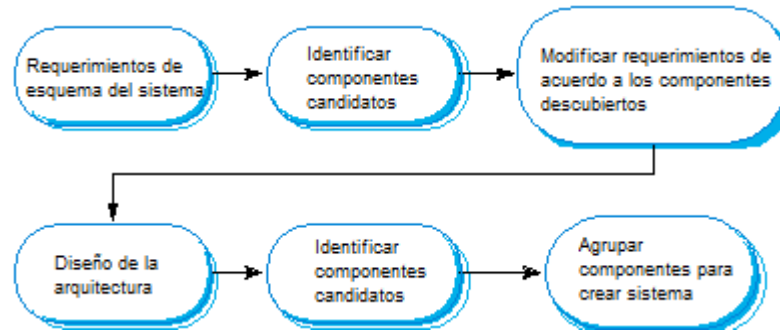


Figura 9 - Flujo DSBC

FUENTE: Sommerville, I. (2005). "Ingeniería del Software"

3.2.1. Características

Esencialmente, los componentes deben cumplir con las siguientes características definidas por Ian Sommerville en el 2004:

- **Estandarizado:** un componente que es usado en DSBC debe cumplir con el modelo estandarizado de componentes. Este modelo debe definir las interfaces del componente, sus meta datos, documentación, composición y despliegue.
- **Independiente:** un componente debe ser capaz de desplegarse sin depender de otro componente específico, en caso de requerir un servicio específico debe ser explícitamente expuesto en las especificaciones de los requerimientos de la interfaz.
- **Componible:** para que un componente sea componible todas las interacciones externas se deben llevar a cabo por a través de interfaces definidas públicamente. Adicionalmente, debe proveer acceso externo para información sobre si mismo como sus métodos y sus atributos.
- **Desplegable:** un componente debe ser autocontenido y debe ser capaz de operar en una como una entidad autónoma en alguna plataforma de componente que implemente el modelo de componente.
- **Documentado:** los componentes deben estar completamente documentados para que los potenciales usuarios puedan decidir si cumple o no sus necesidades. La sintaxis e idealmente, la semántica de todas las interfaces del componente debe ser especificadas.

3.2.2. Modelos de componente

Es una definición de estándares para la implementación, documentación y despliegue. Ejemplos de esto son el modelo Enterprise Java Beans, COM+ model y Corba Component Model. Este especifica cómo deben ser definidas las interfaces y los elemento que deben ser explicados en la definición de las interfaces.

En la Figura 10 se muestran los elementos de los modelos de componente.

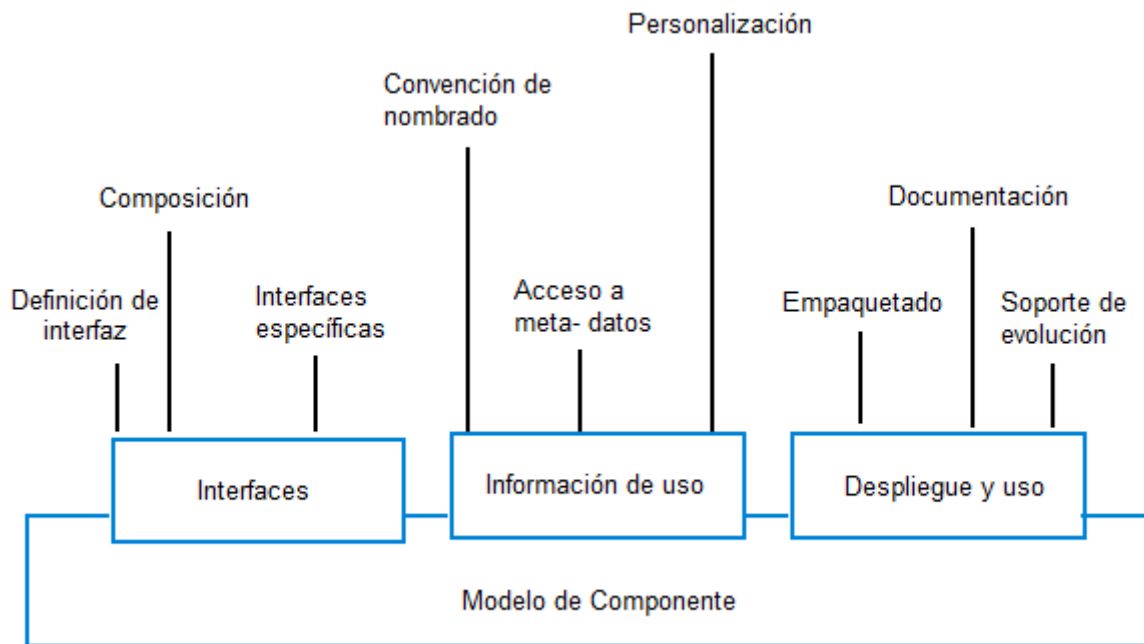


Figura 10 - Modelo de componente

FUENTE: Sommerville, I. (2005). "Ingeniería del Software"

3.2.3. Composición

El proceso de composición implica integrar los diferentes componentes entre si y con la infraestructura. Normalmente esto implica escribir "código de pegamento" que una los componentes. Los diferentes tipos de composición que se pueden realizar son:

- **Secuencial:** donde los componentes son ejecutados en secuencia.
- **Jerárquica:** donde un componente llama el servicio de otro(s).
- **Aditiva:** donde las interfaces de dos componentes son puestas juntas para crear un nuevo componente.

3.2.4. Ciclo de vida

El ciclo de vida de la metodología se resume en la Figura 11:

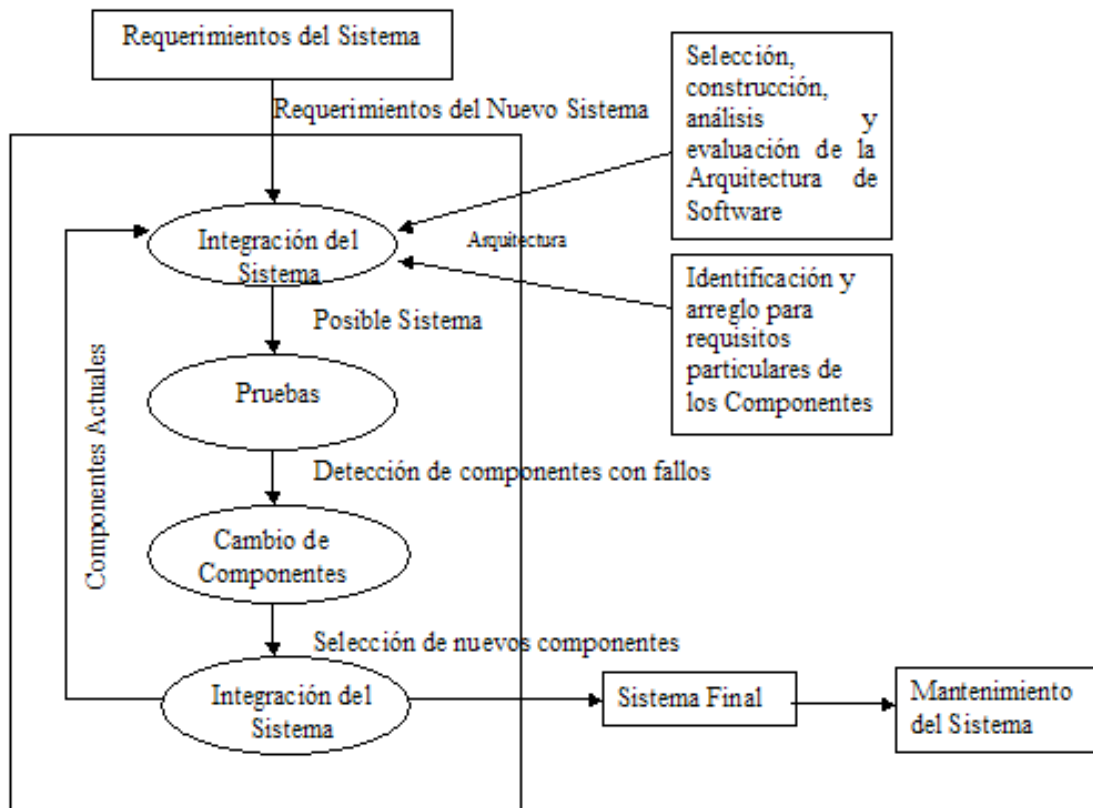


Figura 11 - Ciclo de vida DSBC

FUENTE: (Ospina Torres & León Luna, 2013)

3.2.5. Criterios

Los siguientes 7 criterios son definidos por Meyer en 1999:

- Puede ser usado por otros elementos de software.
- Puede ser usado por los clientes sin la necesidad de la intervención del desarrollador.
- Incluye las especificaciones de todas las dependencias.
- Incluye documentación de las funcionalidades que ofrece.
- Se puede entender su funcionamiento en base a las especificaciones.
- Se puede acoplar a otros componentes.
- Puede ser incorporado a un sistema de manera suave y rápida.

3.2.6. Problemas

- **Confianza:** se debe confiar en el proveedor del componente. Un componente no confiable puede no operar como lo anuncia en el mejor de los casos y en el peor de los casos puede comprometer la seguridad de todo el sistema

- **Requerimientos:** diferentes grupos de componentes satisfarán diferentes requerimientos
- **Validación:**
 - Las especificaciones del componente pueden no ser lo suficientemente detalladas para permitir desarrollar pruebas comprensivas.
 - Los componentes pueden tener funcionalidades indeseadas y esto puede interferir con las pruebas sobre la aplicación.

CAPÍTULO 4

MARCO APLICATIVO

Ya hablado de las tecnologías, referencias, motivación y metodologías que se usarán en el desarrollo del proyecto, es momento de desarrollar el sistema en sí. En el siguiente capítulo se describirá detalladamente todo el proceso de desarrollo. Comenzando por el alcance, la justificación y la arquitectura siguiendo por la adaptación de las metodologías en el proceso de desarrollo y terminando con la aplicación de estas en el desarrollo en sí.

4.1. Arquitectura

En la Figura 12 - Arquitectura Local se muestra la arquitectura que se desarrolló en este proyecto. Particularmente el Front-end es un sistema desarrollado en ReactJS con el paquete *create-react-app* desarrollado por Facebook quien también es el responsable de ReactJS y que está disponible en NPM. Este se comunica con el Back-end desarrollado en Django, que está disponible en el manejador de paquetes *pip* y a su vez este sistema se comunica con el módulo de indexación y el módulo de almacenamiento, que salen del alcance de este proyecto.

Es importante destacar que para el desarrollo de este proyecto no se probó con el módulo de acceso oficial del Archivo Web, porque se encuentra desarrollo y por lo tanto no es accesible para su uso. Sino que se desarrolló una aplicación adicional en Django que simula el funcionamiento del mismo y esto lo usando la herramienta de WarcTools para la descompresión de los archivos en formato WARC y almacenando los datos obtenidos de la descompresión en una base de datos SQLite. Se sabe que así no será el módulo de almacenamiento cuando esté listo, pero para fines prácticos, el back-end es independiente de esto por lo que el cambio solo consistirá en cambiar la dirección a quien se hacen las peticiones una vez el módulo de acceso esté listo.

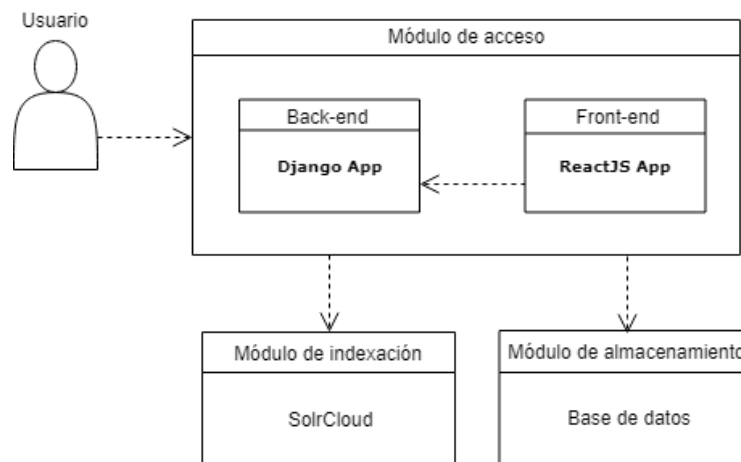


Figura 12 - Arquitectura Local

4.2. Requerimientos

El proyecto tiene requerimientos de dos naturalezas, funcionales y no funcionales. A continuación, se describirán cada uno puntualmente.

4.3. Adaptación de la metodología

Para el desarrollo de este Trabajo Especial de Grado se realizó una adaptación de AgilUs agregando partes de AgilUp. Básicamente ambas son metodologías iterativas por lo que se escogió utilizar las etapas de AgilUs, pero también las fases de iteración de AgilUp, aplicando así iteraciones constituidas por cuatro fases y a la vez estas iteraciones están orientadas en etapas de AgilUp.

Los artefactos y técnicas que se seleccionaron para cada de la metodología fueron los siguientes:

- **Requisitos:** tormenta de ideas, entrevista, evaluación de sistemas existentes, análisis de perfiles de usuarios y lista de requerimientos a desarrollar.
- **Análisis:** prototipos en papel, guía de estilo y modelos de casos de uso.
- **Prototipaje:** prototipos ejecutables, lista de comprobación y pensamientos en voz alta.
- **Entrega:** pruebas de aceptación.

De igual manera, es importante señalar que las etapas de la metodología no se aplicaron de forma exhaustiva en cada iteración, sino que se emplearon según lo necesario, dependiendo de los objetivos planteados para dicha iteración y los resultados obtenidos de las anteriores.

4.4. Iteración 1: Concepción

En esta primera iteración el propósito fue obtener una perspectiva consolidada de lo que se quería lograr y una primera idea de cómo se debía ver esto, por este motivo las fases fundamentales en esta iteración son la de requisitos y la análisis, el prototipaje fue orientado a una discusión entre el desarrollador y el dueño del producto de cómo se podría realizar lo requerido luego de analizarlo y evaluar los recursos que se disponían.

4.4.1. Requisitos

Los requisitos se obtuvieron de una entrevista con el dueño del producto y la evaluación de las referencias previas.

4.4.1.1. Entrevista

Las primeras ideas planteadas fueron:

- Diseñar interfaces fáciles de usar e intuitivas.

- Permitir al usuario consultar la información guardada en el módulo de almacenamiento.
- Tener una sección en la que se explique el motivo del proyecto y cualquier otra información pertinente, además de una sección de contacto.

4.4.1.2. Evaluación de sistemas existentes

Se realizó un análisis de sistemas afines existentes con la finalidad de analizar su funcionamiento, su proceso de negocio y usarlo como referente en el desarrollo del proyecto; así como minimizar las desventajas o errores que estos pudieran tener. Los sistemas estudiados fueron: Wayback Machine(internet archive) y Oasis(archivo web de Korea). También se analizaron otros sistemas de archivo web pero aportaban poca o nula información adicional luego del análisis de las referencias antes mencionadas.

Tabla 7 - Análisis Wayback Machine

Wayback Machine	
Tópico	Descripción/Observaciones
Descripción	Es un portal que permite buscar y desplegar las páginas almacenadas. Pertenecer a la organización Internet Archive
Funcionalidades	Permite registrar usuario. Buscar y desplegar las páginas almacenadas en la versión que el usuario escoja.
Apariencia	La aplicación presenta interfaces consistentes e intuitivas pero muy cargadas y se complican rápidamente.
Personalización	Los usuarios registrados son capaces de guardar sus búsquedas.
Aspectos de accesibilidad	No cuenta con ninguna funcionalidad para usuarios con limitaciones físicas, como por ejemplo atajos de teclado, configuración de tamaño de letra, contraste de colores, entre otros.
Idiomas	Sólo inglés, aunque las páginas archivadas pueden estar en cualquier idioma.
Soporte en línea	Cuenta con un formulario para contactar con personal autorizado. No dispone de un manual de usuario dentro de la misma aplicación, sin embargo la interfaz es lo suficientemente intuitiva.
Opinión como usuario	Mejor ejemplo de archivo web existente, es tanto el más grande como el más completo

(Castillo, Autoria propia, 2018) - <https://web.archive.org/>

Tabla 8 - Análisis Oasis

Oasis	
Tópico	Descripción/Observaciones
Descripción	Igual que Wayback Machine, permite al usuario consultar las colecciones almacenadas en el archivo.
Funcionalidades	Permite registrar usuario. Buscar y desplegar las páginas almacenadas en la versión que el usuario escoja, además de eso permite consultar colecciones de otras índoles.
Apariencia	Interfaz muy cargada pero bien distribuida, presenta una paleta de colores agradable a la vista y animaciones. No es intuitiva pero está lo suficientemente documentada como para entender qué hace cada sección.
Personalización	No cuenta con funcionalidades para personalizar la aplicación.
Aspectos de accesibilidad	No cuenta con ninguna funcionalidad para usuarios con limitaciones físicas, como por ejemplo atajos de teclado, configuración de tamaño de letra, contraste de colores, entre otros.
Idiomas	Coreano
Soporte en línea	Permite contactar con el administrador del sistema mediante el envío de un correo electrónico, formulario de contacto o por teléfono local.
Opinión como usuario	Es el archivo web con la mejor interfaz, a pesar de ser cargada tiene buen aspecto y tiene el suficiente detalle como para entender la tarea de cada elemento

(Castillo, Autoria propia, 2018) - <http://www.oasis.go.kr/>

4.4.1.3. Requerimientos

Una vez realizada la entrevista y la evaluación de sistemas afines, se dedujeron los siguientes requerimientos:

4.4.2. Funcionales

4.4.3. No funcionales

Requerimientos funcionales

- Consultar información ingresado en el módulo de indexación
- Mostrar respuesta obtenida del módulo de indexación
- Seleccionar una versión de la respuesta obtenida por el módulo de indexación
- Solicitar versión seleccionada al módulo de almacenamiento
- Desplegar la página web respondida por el módulo de almacenamiento

Requerimientos no funcionales

- Interfaz escueta e intuitiva.
- Sección de información de los implicados en el sistema del Archivo Web de Venezuela, motivación y otros trabajos especiales de grado pertenecientes al sistema.
- La estructura del código será consistente para lograr mantenibilidad para los próximos desarrolladores a futuro.
- Desarrollar código documentado para una buena legibilidad
- Etapa de análisis

4.4.3.1. Casos de uso

A continuación, se presentan los diferentes niveles que componen el modelo de casos de uso de la aplicación objetivo de este Trabajo Especial de Grado.

- **Nivel 0:** En la Figura 13 - Nivel 0 se puede observar la interacción de los usuarios con el sistema. A este nivel solo existe un tipo de usuario que es el que consulta sobre el módulo.

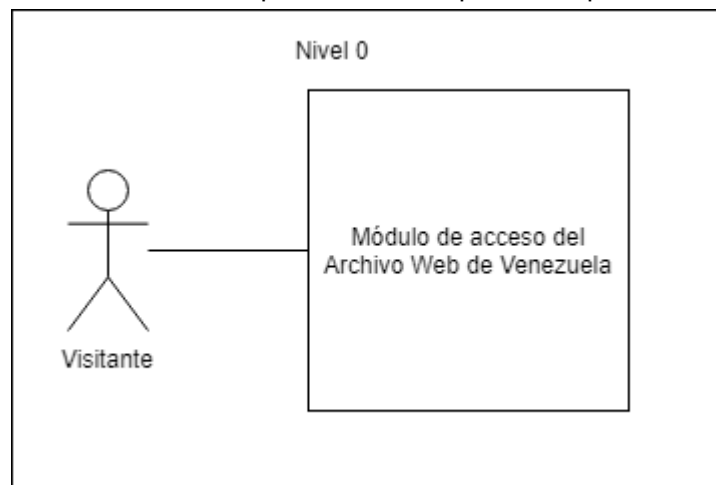


Figura 13 - Nivel 0

- **Nivel 1:** En la Figura 14 - Nivel 1 se especifica el conjunto de casos de uso que engloban las funcionalidades planteadas para la aplicación y son descritos en las Tabla 9 - Nivel 1. Realizar consulta, **¡Error! No se encuentra el origen de la referencia.** y **¡Error! No se encuentra el origen de la referencia.**

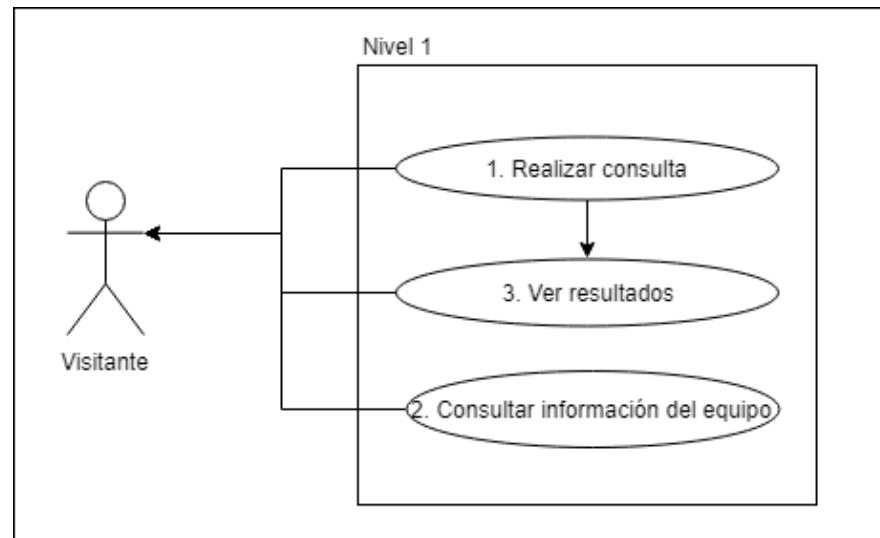


Figura 14 - Nivel 1

Tabla 9 - Nivel 1. Realizar consulta

Identificador y nombre	1. Realizar consulta
Actor(es)	Visitante.
Descripción	Permite al actor llenar un consultar las páginas archivadas .
Precondición	Ninguna
Postcondición	Luego de consultar se listarán los resultados obtenidos de su búsqueda, en este punto el actor debe escoger uno de los resultados para ser mostrado.
Flujo normal	1. El usuario ingresa datos que se buscan en el sistema, se listan los resultados obtenidos y este selecciona uno 2. Se despliega el sitio web que se quiere y el usuario consulta esta información
Flujo alternativo	Ninguno.
Puntos de extensión	- Casos de uso Include: ninguno - Casos de uso Extend: ninguno.

(Castillo, Autoria propia, 2018)

Tabla 10 - Nivel 1. Consultar información del equipo

Identificador y nombre	2. Consultar información del equipo
Actor(es)	Visitante
Descripción	Permite al actor obtener toda la información referente al equipo encargado de desarrollar los diferentes módulos.
Precondición	Ninguno.
Flujo Normal	1. El usuario selecciona la sección de información del proyecto y ve dicha información
Postcondición	Ninguna.
Puntos de extensión	- Casos de uso Include: ninguno. - Casos de uso Extend: ninguno.

(Castillo, Autoria propia, 2018)

- **Nivel 2:** En la Figura 15 - Nivel 2 se observan las funcionalidades que se extienden de los casos de uso expuestos en el nivel 1 y se describen a mayor profundidad sus funcionalidades

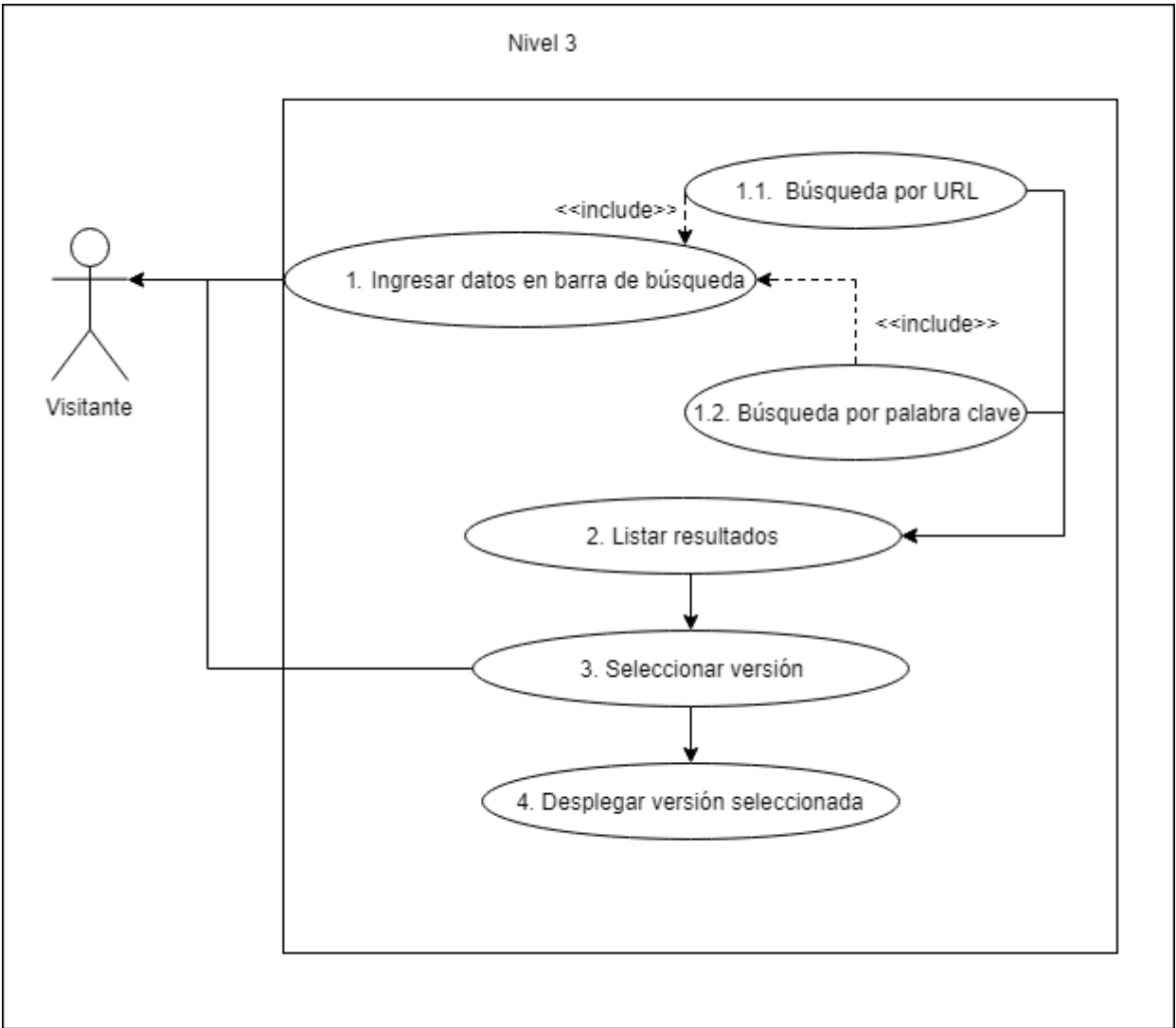


Figura 15 - Nivel 2

Tabla 11 - Nivel 2. Ingresar datos

Identificador y nombre	1. Ingresar datos en la barra de búsqueda
Actor(es)	Visitante.
Descripción	Permite al actor ingresar los datos que quiere consultar para ser filtrados y buscados por el servicio de búsqueda.
Precondición	Ninguna
Postcondición	Ninguna

Flujo normal	1. El usuario ingresa información en la barra de búsqueda 2. El sistema determina qué tipo de búsqueda es, ya sea por URL o por palabras clave 3. El sistema realiza la consulta al módulo de almacenamiento 4. El sistema lista las respuestas obtenidas por el módulo de almacenamiento 5. El usuario escoge una de las respuestas listadas por el módulo. 6. El sistema consulta la información solicitada al módulo de acceso. 7. El sistema despliega la respuesta obtenida del módulo de acceso al usuario.
Flujo alternativo	Ninguno.
Puntos de extensión	- Casos de uso Include: Búsqueda por URL, Búsqueda por palabra clave. - Casos de uso Extend: ninguno.

(Castillo, Autoria propia, 2018)

En la Figura 16 - Flujo de la aplicación se describe el proceso de interacción previsto entre el usuario y la aplicación. Cabe destacar que la responsabilidad del módulo de acceso llega hasta

el Back-end, el módulo de indexación y el de almacenamientos son sistemas independientes a este y forman parte del archivo web de Venezuela, pero no competen al proyecto actual.

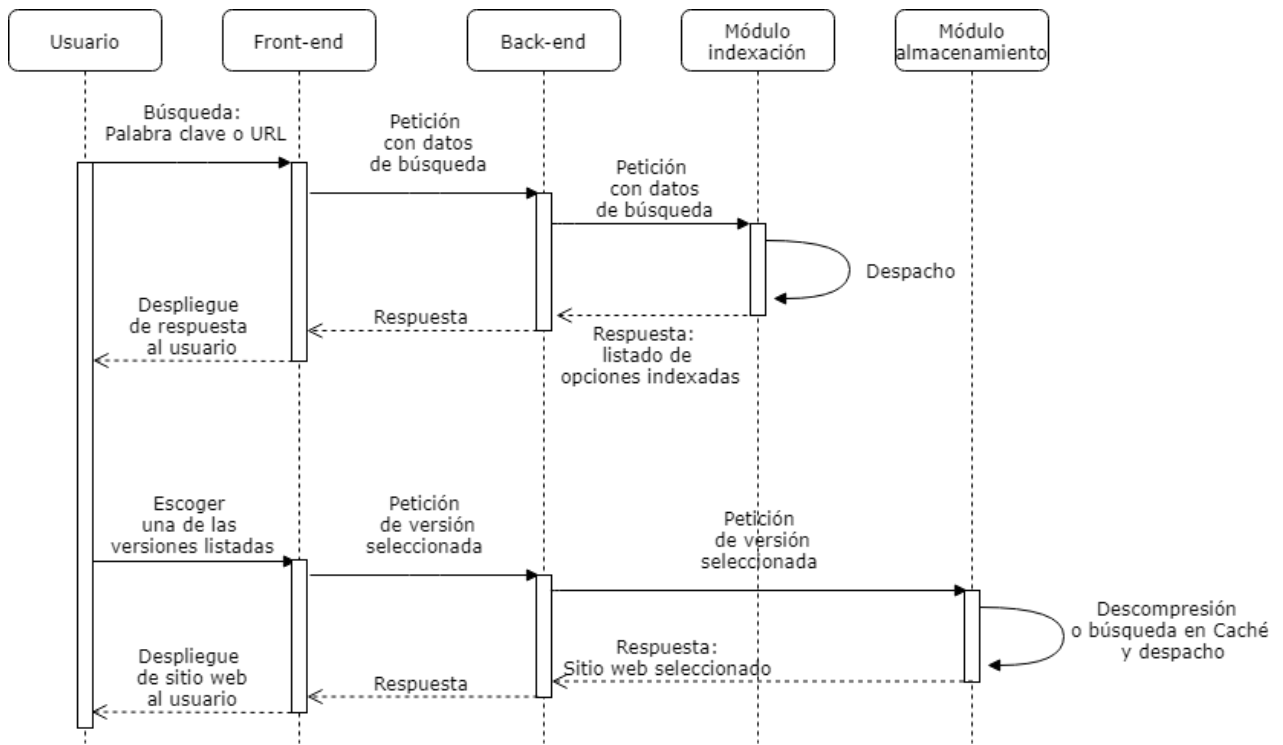


Figura 16 - Flujo de la aplicación

4.4.3.2. Guía de estilos

En la guía de estilo se definió la gama de colores utilizados en la aplicación, la tipografía y color de fuente, características de los botones y logos utilizados. Es importante destacar que existe un logo para el Archivo Web de Venezuela desde antes de la concepción del proyecto, el cual es de los colores de la bandera nacional, por lo que los colores a escoger deben hacer cohesión con esta precondition.

- **Gama de colores**

- **Colores principales:** son los colores básicos de la aplicación, los que la definen y ocupan la gran mayoría de la misma. En este caso se utilizaron colores azules (ver Figura 17 - Colores principales) para lograr una sensación de comodidad, tranquilidad y serenidad en el usuario cuando navegara por la aplicación.

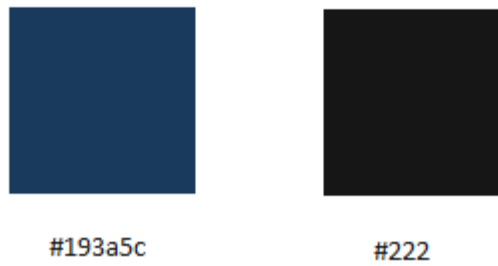


Figura 17 - Colores principales

- **Colores secundarios:** son los colores que complementan a los principales, en su mayoría son empleados como colores de fondo (ver Figura 18 - Colores secundarios)



Figura 18 - Colores secundarios

- **Tipografía y color de fuente**

La tipografía que se utilizó en toda la aplicación fue "Generica" para títulos y enlaces destacable mientras que para los textos se utilizó "Open-Sans". Todos los textos usan el color blanco.

- **Logo**



Figura 19 - Logo

4.4.4. Prototipaje

La arquitectura de la aplicación es de tipo cliente-servidor. Para el desarrollo de la aplicación se utilizaron dos tecnologías diferentes, la primera fue ReactJS para el lado del Front-end y la otra fue Django para el Back-end.

La manera en que estas tecnologías comprenden la arquitectura del sistema es de la siguiente: el front-end, hecho en ReactJS se encarga de las mostrar interfaces, estas reciben y envían información del back-end, hecho en Django, el cual a su vez se conecta con el módulo de indexación y con el módulo de almacenamiento con los cuales también ocurre intercambio bidireccional de información.

Las razones por la que se seleccionaron estas tecnologías fueron las siguientes:

- Tanto Django como ReactJS son tecnologías que van con el estado del arte de sus ámbitos pertinentes.
- Warctools, la herramienta que procesa los archivos de extensión .warc está desarrollado en Python.
- Django es una herramienta orientada al desarrollo rápido, mantenible y escalable, todas estas características son deseables para el proyecto.
- Una característica fundamental de ReactJS es su DOM Virtual, el cual se quiere aprovechar particularmente en este proyecto para el despliegue de las versiones de las páginas porque este aumenta la eficiencia del despliegue al calcular todos los cambios que se realicen en el DOM Virtual por separado y actualizan el DOM real solo cuando se quiere.

4.4.4.1. Interfaces

En la Figura 4.23 se pueden observar las interfaces desarrolladas en esta primera iteración.



Figura 20 - Página principal

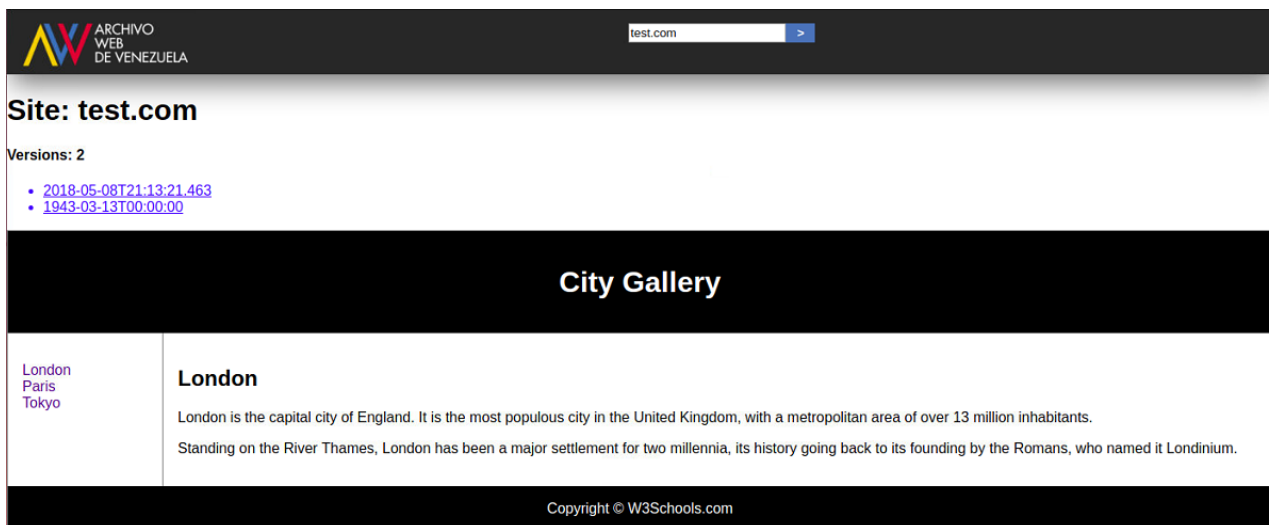


Figura 21 - Búsqueda desplegada y versión seleccionada

4.4.5. Entrega

Para esta entrega se desarrollaron los primeros artefactos del proyecto, todos están sujetos a cambios y se irán refinando a medida que se avanza el proceso de desarrollo.

La primera entrega consistió de un repositorio hecho en ReactJS que muestra las interfaces

que se observan en la etapa de prototipaje y un repositorio hecho de Django que tiene dos servicios, el primero que recibe la búsqueda y la filtra por tipo entre palabra clave y URL. El segundo servicio devuelve un HTML de prueba que es el que se muestra en la **iError! No se encuentra el origen de la referencia.**, este servicio se refinará para ser el que devuelve el HTML de la versión seleccionada por el usuario en las próximas iteraciones.

4.5. Iteración 1: Refinamiento del Front-end

4.5.1. Requerimientos para esta iteración

- Mejorar el diseño de las vistas del front-end.
- Enriquecer la información referente al equipo del Archivo Web de Venezuela mostrada en el front-end.
- Refinar el despliegue de resultado, implementando un tipo de respuesta diferente para cada tipo de búsqueda (URL y palabra clave)

4.5.2. Análisis

Básicamente en esta iteración lo que debe ser analizado es cómo se verá la nueva versión de la interfaz, cómo deben cambiar los elementos presentes, qué puede agregarse y qué puede quitarse. Para esto se hace análisis de las referencias escogidas y cómo pueden integrarse los elementos aprovechables al sistema que se tiene.

Los elementos que se escogieron de las referencias para seguir de ejemplo fueron:

- Barra de búsqueda de Oasis
- Calendario de Wayback Machine
- Formato de imágenes de Oasis

Esto quiere decir que se adaptarán dichos artefactos al sistema actual, integrando los colores escogidos para los estilos. Se verán a detalle estos elementos en la etapa de prototipaje.

4.5.3. Prototipaje

En esta etapa se observarán los elementos analizados previamente, además de las nuevas funcionalidades estipuladas en los requerimientos.

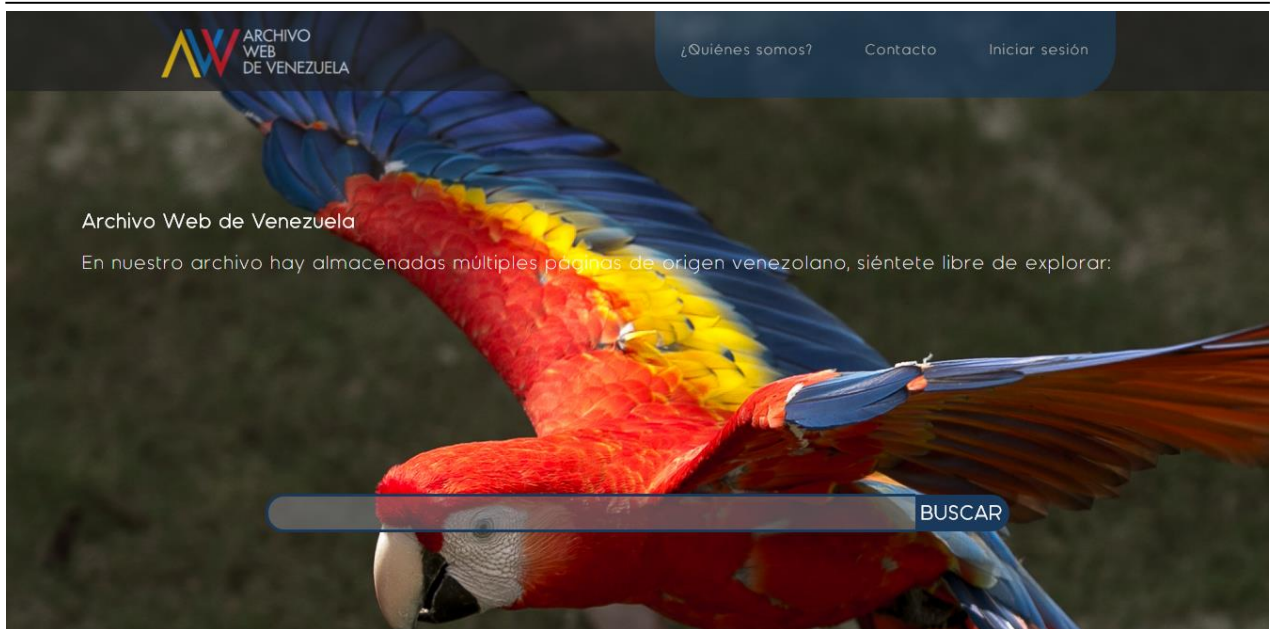


Figura 22 - Página principal - Barra de búsqueda

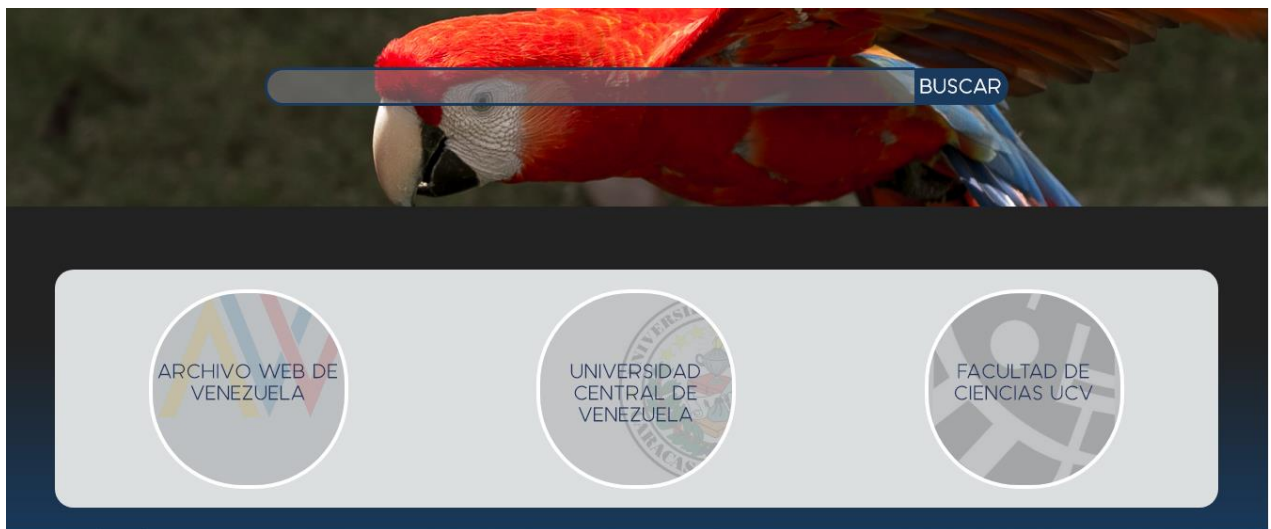


Figura 23 - Página principal - Enlaces



Figura 24 - Información equipo AWV

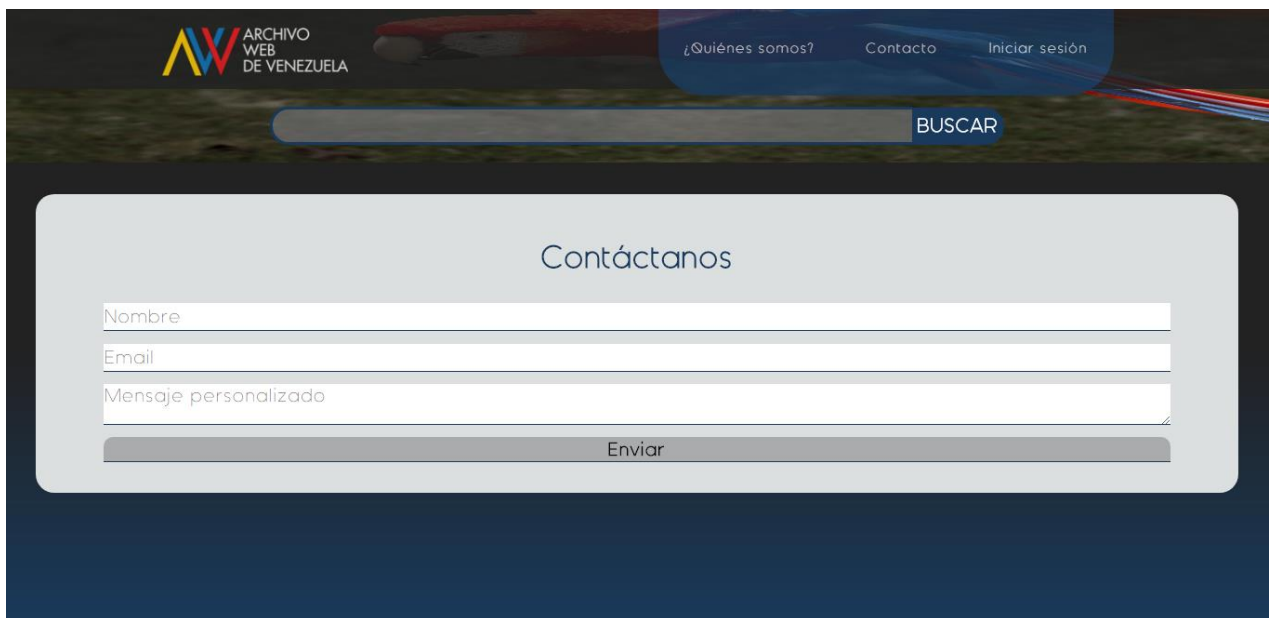


Figura 25 - Formulario de contacto

4.5.4. Entrega

En esta iteración se entrega un sistema con vistas mucho mejor logradas, y qué están en mayor sintonía con el estado del arte de la experiencia de usuario, considerando fondos con gradiente, secciones de estilos planos, bordes redondeados y animaciones de transición entre vistas. También se adoptó una manera híbrida para el despliegue de resultados, usando un

calendario como Wayback Machine, para las búsquedas de UL, y a su vez listando los resultados para la búsqueda por palabra clave como en Oasis.

4.6. Tercera iteración: Comunicación con módulo de indexación

Una vez desarrolladas interfaces satisfactorias estéticamente y que incluyen toda la información que se desea mostrar, es en esta iteración que se desea integrar ahora la comunicación con el módulo de indexación, funcionalidad fundamental para lograr el objetivo del proyecto, que es desplegar las páginas web indexadas en el sistema.

Hasta este punto el back-end respondía con datos estáticos a todas las peticiones, es decir, siempre daba el mismo resultado sin importar qué datos tenía la petición, es momento ahora de cambiar esta respuesta por datos dinámicos que reaccionen a la información que proporcione el módulo de indexación.

4.6.1. Requerimientos

- Usar la SolrCloudPy para comunicar con SolrCloud (módulo de indexación).
- Procesar la respuesta recibida por SolrCloud.
- Enviar los datos procesados con el formato requerido por el front-end.
- Integrar la respuesta de SolrCloudPy pasada del back-end al front end al calendario, señalando las fechas recibidas en el mismo para ser escogidas y cuando el usuario seleccione alguna, hacer una nueva petición hacia el módulo de almacenamiento que despachará la versión solicitada

4.6.2. Análisis

Para empezar, no se puede integrar una herramienta de comunicación a cierta plataforma, en este caso SolrCloudPy, si no se tiene la plataforma con la que se quiere comunicar. Entonces lo primero que se debe hacer es instalar y configurar esta plataforma. Logrado esto se puede proceder a usar SolrCloudPy, que es una librería de Python.

Ya cumplidas las dependencias se comienza con el verdadero desarrollo. En este punto ya el sistema tiene la capacidad de dilucidar qué tipo de búsqueda se ha solicitado, entonces lo que se debe de hacer es incluir en esta selección la llamada a la API de SolrCloud. Lo que ocurre es que el front-end envía la solicitud al back-end, este filtra el tipo de búsqueda y de cualquiera del tipo que sea hace una petición a SolrCloud con la forma necesario dependiendo a su tipo, SolrCloud entonces responde con una lista ya sea de URL distintas o de versiones categorizadas por fecha de la misma URL, esta respuesta la formatea el back-end y la envía al front-end. El mismo entonces despliega esta lista ya sea con un calendario o con una lista simple, en este punto el usuario selecciona uno de los resultados mostrados, esto se traduce en otra petición para el back-end, esta vez con el identificador de la página que él quiere, esta no va hacia

SolrCloud sino al módulo de almacenamiento, el cual devuelve la página que se solicitó en código HTML, CSS, y JavaScript, el back-end lo recibe y lo envía al front-end, el cual lo despliega.

4.6.3. Prototipaje

Se comienza por dotar al backend la capacidad de comunicarse con la plataforma de SolrCloud, instalado la librería SolrCloudPy a través del manejador de paquetes *pip* con el comando *pip install solrcloudpy*, esto asumiendo que la plataforma ya ha sido instalada y configurada en el equipo en uso. Hecho ésto luego se llama a esta biblioteca en el servicio de comunicación existente en el back-end, como se observa en Figura 26 - Configuración de comunicación con SolrCloud.

```
from django.http import HttpResponse, JsonResponse
from django.views.decorators.csrf import csrf_exempt
from rest_framework.renderers import JSONRenderer
from rest_framework.parsers import JSONParser
from deliverer.models import Website
from deliverer.serializers import WebsiteSerializer
from django.http import HttpResponse
from solrcloudpy.connection import SolrConnection

import datetime
import json
import re

conn = SolrConnection(server=["localhost:8987", "localhost:7578"], detect_live_nodes=False, user=None, password=None, timeout=10)
coll = conn['archiveWebVenezuela']
```

Figura 26 - Configuración de comunicación con SolrCloud

Con esta configuración ya se puede hacer peticiones a la plataforma, esto se integra al servicio que responde al front-end como se observa en Figura 28 - Petición a SolrCloud. Las respuestas dependen del tipo de petición que haya solicitud, esto es detectado por la petición a través de una expresión regular como se muestra en Figura 27 - Expresión regular. Si se solicitó una URL entonces la respuesta va a ser un arreglo de fechas que se integra al calendario del front-end, por el contrario, si la solicitud son palabras claves la respuesta será un arreglo de URL que coincidan con las palabras ingresadas, esto se listará en el front-end con una lista como se observa en Figura 29 - Búsqueda por palabra clave y Figura 30 - Búsqueda por URL. Llegado a este punto el usuario debe ahora escoger una de las versiones listadas

```
def matchSite(site):
    result = re.match(r'^((http://www|.https://www|.http://|https://)?[a-z0-9]+([\-\.\.]{1}[a-z0-9]+)*\.[a-z]{2,5}(:[0-9]{1,5})?(\./'
    return bool(result)
```

Figura 27 - Expresión regular

```
@csrf_exempt
def jsontest(request):
    data = json.loads(request.body)
    query = data['query']
    if matchSite(query):
        extRequest= 'URL: *'+query+'*'
        response = coll.search({'q':extRequest})
        response = jsonify(response)
    else:
        extRequest= 'content: *'+query+'*'
        response = coll.search({'q':extRequest})
        response = jsonify(response)
    return JsonResponse(response)
```

Figura 28 - Petición a SolrCloud

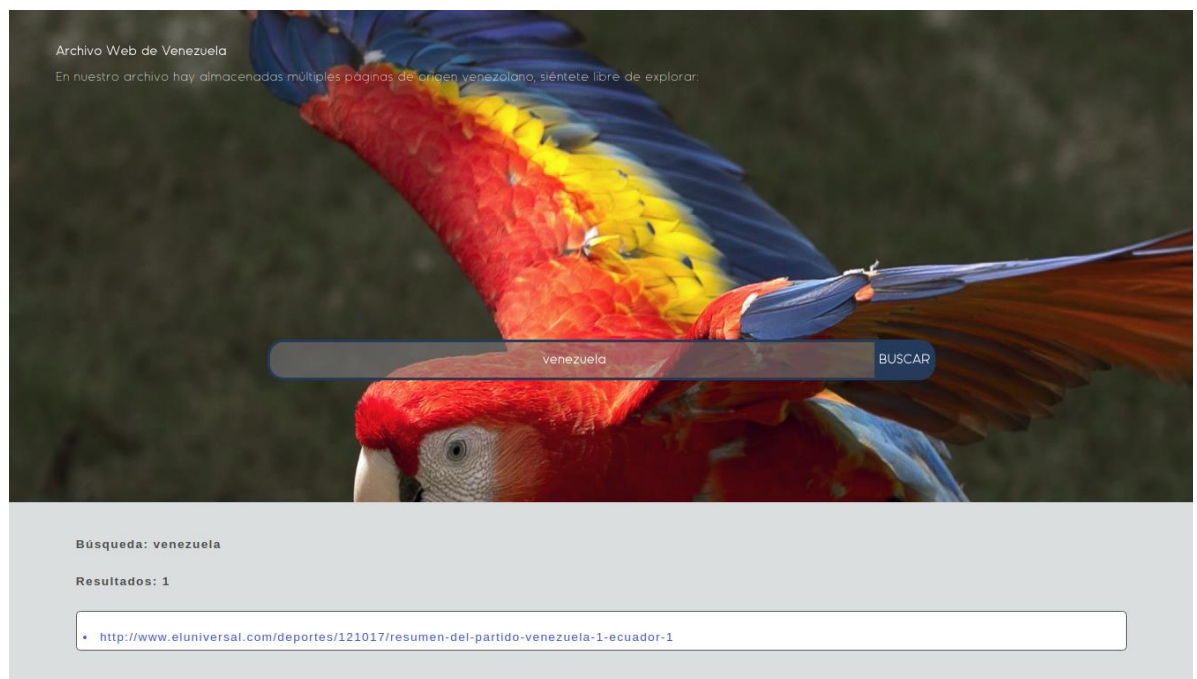


Figura 29 - Búsqueda por palabra clave



Figura 30 - Búsqueda por URL

Esta selección se traduce en una solicitud al back-end, este a su vez hará una solicitud al módulo de almacenamiento con el identificador de la URL que esté almacenada, en la Figura 31 - Servicio despachador de sitios se observa como el servicio que recibe la solicitud despacha la información solicitada con los archivos HTML, CSS y JavaScript que componen el sitio web solicitado, el back-end los recibe y los transmite al front-end, que los despliega, mostrando efectivamente la página web que solicitó el usuario como se muestra en la Figura 32 - Versión seleccionada Pt.1 y Figura 33 - Versión seleccionada Pt.2.

```
@csrf_exempt
def siteretrieve(request):
    data = json.loads(request.body)
    query = ''.join(data['site_version'])
    aux = query[(query.rfind('/')+1):]
    aux = aux[:40] + '.html'

    if aux == '.html':
        aux = 'index.html'

    route = '../../indexadorsolr/scriptIndexador/' + query.replace('http://', 'http/').replace('.warc.gz', '') + aux
    print(route)

    F = open(route, 'rb')
    doc = BeautifulSoup(F, "html.parser")

    return HttpResponse(doc)
```

Figura 31 - Servicio despachador de sitios

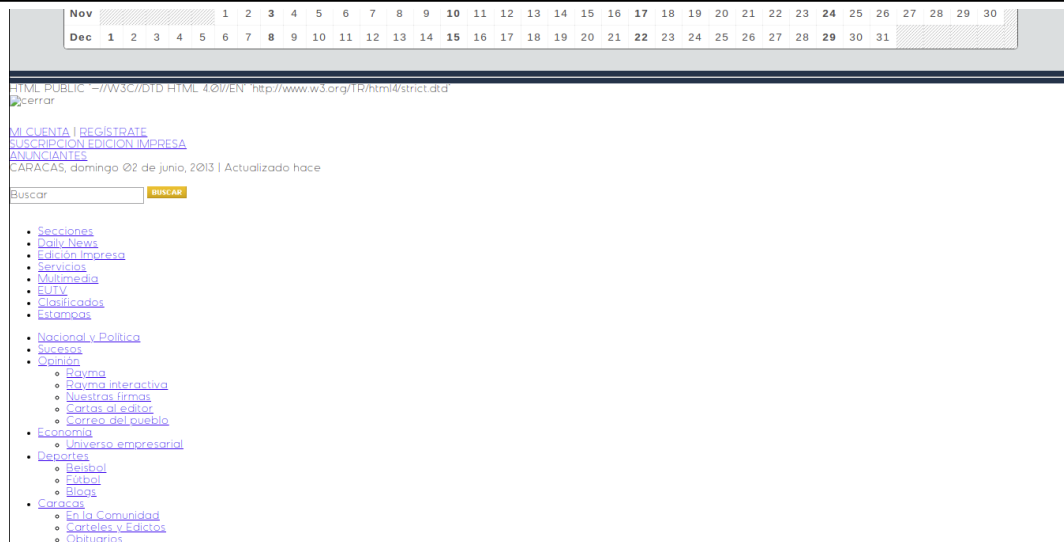


Figura 32 - Versión seleccionada Pt.1



Figura 33 - Versión seleccionada Pt.2

4.6.3.1. Proceso de integración

La integración con el resto del archivo web hay que aclarar que fue parcial, inicialmente porque el archivo web está en desarrollando y para el momento solo se cuenta con el módulo de indexación listo, que además no se encuentra en ejecución en alguna plataforma sino que se dispone del código en un repositorio, por lo que para usarlo se debe instalar de manera local. El proyecto requerirá además del módulo de almacenamiento que, como ya se dijo, no está disponible, por lo que se usará una aplicación que se encargará de guardar la información descomprimida de los archivos .warc en una base de datos, como esta será desarrollada en Django también, se usará una base de datos SQLite por conveniencia, aunque se sabe que el verdadero módulo no usará esta.

Aclarado esto se procede ahora a describir el proceso de integración:

1. Se busca el módulo de indexación en la plataforma (BitBucket) donde esta almacenado el repositorio, se clona y se procede a instalar como se indica en su archivo *readme*.
2. Instalada entonces se procede a indexar información en él. Se cuenta con un script desarrollado también en Python para esto. Ahora bien, el script no tiene la configuración necesaria para lograr su tarea sin ser modificado.
3. Se procede a analizar este script y a hacer las modificaciones necesarias para lograr su tarea. Se determinan cuáles son las configuraciones que deben cambiarse, específicamente son los puertos de los fragmentos de SolrCloud, los que están configurados son los usados por defecto pero el archivo *readme* determina usar otros, posteriormente hay que modificar las rutas establecidas de los archivos *.warc*, las configuradas son rutas absolutas de los equipos donde fue desarrollado este módulo inicialmente, se hace el cambio por rutas absolutas del equipo actual.
4. Logrado el funcionamiento de Solr se procede ahora a realizar la consulta a través del back-end. Obtenida la respuesta se sabe que ahora es capaz de comunicarse con Solr por lo que se ha integrado el módulo de indexación al back-end del proyecto. Se procede entonces a desarrollar la aplicación que simulará el módulo de acceso. Esta aplicación es una base de datos en SQLite que almacena los archivos generados cuando se procesan los archivos WARC para su indexación. Su comportamiento es el siguiente:
 - a. Va hacia la ruta donde están los *.warc* descomprimidos.
 - b. Crea una entrada por cada directorio que representa un *warc* y sus campos son cada uno de los archivos internos de este.
5. Desarrollada esta nueva aplicación se procede ahora a realizar una consultar nuevamente en Solr y ahora con la respuesta obtenida se hace la solicitud en esta nueva aplicación, con la respuesta que es el documento HTML del sitio consultado se despliega el sitio en el front-end

Pruebas de integración

Las pruebas realizadas se basan lograr una comunicación efectiva entre el módulo de acceso y los módulos de indexación y almacenamiento.

- **Pruebas con módulo de indexación:** con la configuración suministrada en el *readme* la primera prueba falla, ya que la información dada no es suficiente para lograr ejecutar dicho módulo.
 - Las direcciones para descargar la versión requerida de Solr está expirada por lo que se debe buscar por iniciativa esta versión en plataformas de búsqueda (Google)

- Obtenida e instalada la versión adecuada de Solr se sigue con la configuración dada. El resto de la instalación transcurre sin problemas pero al momento de indexar, utilizando un script proporcionado también por el repositorio del módulo vuelve a fallar, ya que las configuraciones del script están adaptadas al equipo en el que se desarrolló y no son generales, también faltan módulos de pip por instalar que no están estipulados en el readme, específicamente BeautifulSoup4.
- Corregidas estas configuraciones ahora si se puede establecer una comunicación efectiva entre SolrCloud y el back-end del proyecto.
- **Pruebas con el módulo de almacenamiento:** el módulo de acceso presenta menos retos ya que es una simulación parcializada a lo que requiere el proyecto y está hecha a la medida. Sin embargo, presenta retos inherentes a la naturaleza del sistema:
 - La tarea de almacenar en la base de datos es sencilla pero si se intenta consultar un sitio web en base a la respuesta obtenida, esta falla por lo que hay que determinar cómo será la identificación de la información requerida, esto es crucial ya que hay que tener en cuenta que es SolrCloud quien determina como se llama la versión del sitio web que se desea consultar, en este caso la respuesta que proporciona es la URL de la información solicitada pero esta no es la estructura de los archivos descomprimidos aunque es muy similar, por lo que la identificación que se les dará a las entradas que representa a los warc en la base de datos será esta URL que se obtiene haciendo un análisis gramatical de la ruta relativa de los archivos y agregando las palabras "<http://www>." y ".com" en ciertas partes de esta ruta, dándole la forma de un enlace a un sitio web.
 - Hecho este procesamiento se puede ahora solicitar con URL en la base de datos y responde efectivamente, dando como respuesta los archivos requeridos para un despliegue efectivo de las páginas web solicitadas.

4.6.3.2. Prueba de aceptación

Logrado el objetivo de integrar el sistema con los módulos requeridos y a su vez ser capaz de listar, despachar y desplegar el sitio web consultado es ahora el momento de ser evaluado por el usuario. Esto es en pocas palabras dejarlo usar el sistema y luego hacerlo contestar una encuesta con la que en base a sus respuestas se puede cuantificar entonces el sistema.

La población entrevistada consistió de personas que tenían amplia experiencia con el uso de computadoras y están al día con las tecnologías modernas, específicamente consistió

de 14 estudiantes, específicamente 8 estudiantes de computación, 5 de biología y 1 de física, 4 licenciados en computación, 2 licenciados en biología y un ingeniero civil. Siendo un total de 21 personas entrevistadas.

Esto se realizó desarrollando una encuesta con cinco preguntas, tres de las cuales usan la escala de Likert y las otras dos son binarias. Sus respuestas se describen como sigue:

La primera sección tiene le siguiente formato:

Respecto a la interfaz *

	Me parece desagradable	No me gusta pero es aceptable	No me es relevante	Me parece aceptable	Me gusta mucho
¿Cuál es su primera impresión al ingresar al sistema?	☐	☐	☐	☐	☐
Es fácil comprender las acciones que se pueden realizar	☐	☐	☐	☐	☐
¿Le parecen que los colores usados son agradables?	☐	☐	☐	☐	☐
El estilo es consistente en todas las páginas	☐	☐	☐	☐	☐
La experiencia con la interfaz fue positiva	☐	☐	☐	☐	☐

Tabla 12 - Primera sección de encuesta

Y los usuarios contestaron de la siguiente manera:

Respecto a la interfaz

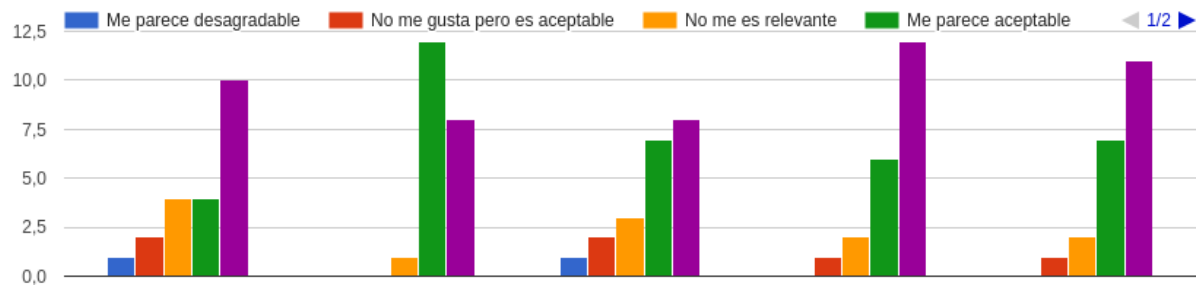


Figura 34 - Respuestas de primera sección Pt.1

Respecto a la interfaz

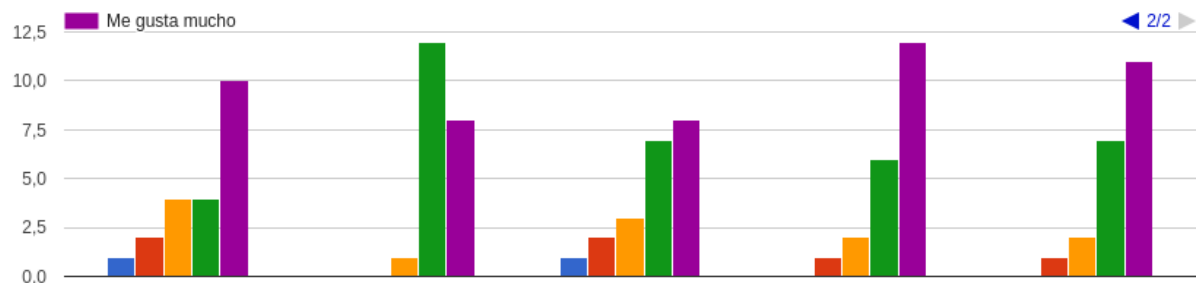


Figura 35 - Respuesta de primera sección Pt.2

Donde cada sección corresponde a cada pregunta en el orden mostrado en la Tabla 12
- Primera sección de encuesta.

La segunda sección tiene el siguiente formato:

Respecto a la información mostrada: *

	No	Me es irrelevante	Sí
¿Le parece relevante la información referente al desarrollo de la página?	☐	☐	☐
Al ingresar al sistema ¿entendió que debía hacer?	☐	☐	☐
Le parece que falta información referente al equipo de desarrollo	☐	☐	☐

Tabla 13 - Segunda sección de encuesta

Y las respuestas fueron como se muestra en

Respecto a la información mostrada:

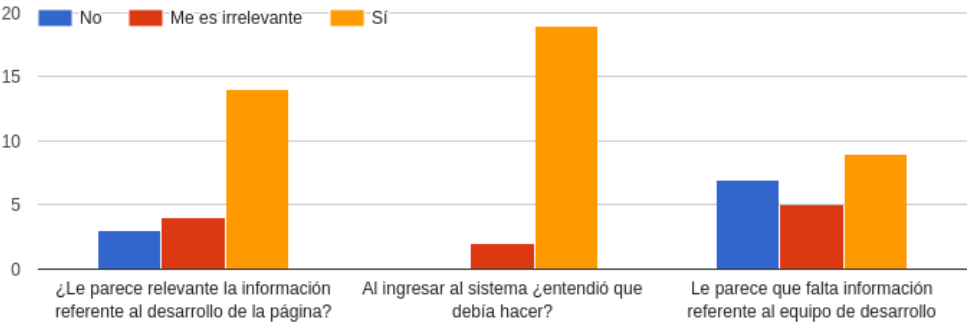


Figura 36 - Respuestas segunda sección

La tercera sección tiene el siguiente formato:

Respecto a las búsquedas: *

	No	Sí
Al ingresar al sistema ¿entendió qué debía hacer para realizar una búsqueda?	☐	☐
Al realizar la búsqueda ¿entendió los resultados mostrados?	☐	☐
Al escoger una versión de las listadas ¿le pareció que la recreación del sitio fue adecuada?	☐	☐
Durante todo el proceso de consulta ¿le pareció que los resultados fueron los esperados?	☐	☐
Al momento de ingresar datos ¿supo que información debía ingresar?	☐	☐
Cuando apareció el calendario ¿entendió como debía interactuar con él?	☐	☐
Cuando se listaron diferentes sitios web ¿entendió como debía interactuar con dicha lista?	☐	☐

Tabla 14 - Tercera sección de encuesta

Y las respuestas proporcionadas se observan en Tabla 14 - Tercera sección de encuesta

Respecto a las búsquedas:

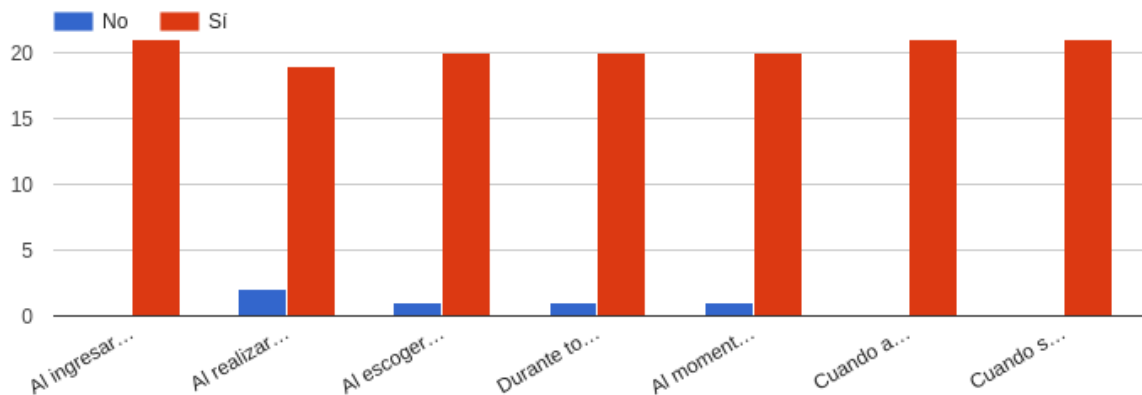


Figura 37 - Respuestas de tercera sección de encuesta

La cuarta sección tiene el siguiente formato:

Respecto a los sitios web desplegados: *

	No	Si
La navegación por el sitio fue satisfactoria	☐	☐
Las búsquedas dan resultados esperados	☐	☐
¿Le parece que la lista desplegada en caso de búsqueda por palabra clave y el calendario en caso de búsqueda por URL son la manera adecuada de mostrar los resultados?	☐	☐

Tabla 15 - Cuarta sección de encuesta

Y las respuestas proporcionadas se observan en Figura 38 - Respuestas de cuarta sección de encuesta

Respecto a los sitios web desplegados:



Figura 38 - Respuestas de cuarta sección de encuesta

La quinta sección tiene el siguiente formato:

¿Quisiera ver esta aplicación como un sistema público y gratuito? *

☐ Sí

☐ No

☐ No público pero si gratuito (para personal especializado)

☐ No gratuito pero si público (cualquiera podría usarlo si esta dispuesto a pagarlo)

☐ Otro: _____

Tabla 16 - Quinta sección de encuesta

Y sus respuestas se observan en Figura 39 - Respuestas quinta sección

¿Quisiera ver esta aplicación como un sistema público y gratuito?

21 respuestas



Figura 39 - Respuestas quinta sección

4.6.4. Entrega

En esta iteración se entregan dos sistemas con funcionalidades añadidas, el back-end tiene la funcionalidad de comunicarse con la plataforma de SolrCloudPy y con el módulo de almacenamiento, esto la da la responsabilidad de ser la zona de intercambio entre el front-end y el resto del Archivo Web de Venezuela. El front-end a su vez adquiere la funcionalidad de listar y desplegar las respuestas recibidas por el back-end, específicamente integrar fechas al calendario, listar URLs y desplegar internamente archivos HTML, CSS y JavaScript.

4.6.4.1. Cierre de la iteración

Llegado a este punto se culmina el proceso de desarrollo del módulo de acceso del Archivo web de Venezuela, siendo capaz de lo que se había estipulado desde el seminario que es listar respuestas y desplegar la página web solicitada por el usuario a través de una amalgama de tecnologías integradas a lo largo del proceso de desarrollo.

CONCLUSIONES

El objetivo principal es el desarrollo del módulo de acceso para la nueva versión del Archivo Web de Venezuela incorporando la nueva arquitectura propuesta que busca distribuir tareas para mejorar tiempo de respuesta y agregar funcionalidades al Archivo Web de Venezuela, para el momento de concluir el proceso de desarrollo se cuenta con un sistema capaz de realizar las funcionalidades requeridas, principalmente consultar y desplegar los sitios web archivados con tiempos de respuesta en el orden de los segundos, muy diferente de la primera versión del Archivo Web donde el tiempo de respuesta está en el orden de los minutos, razón principal que hizo necesaria la nueva versión. Adicionalmente, esta nueva implementación también hace mejor uso de la memoria, porque no es necesario desempaquetar los archivos WARC siempre que se consulten por ellos, sino que al ser indexados son también almacenados en la base de datos que simuló el módulo de acceso, ahorrando memoria y capacidad de procesamiento. En este sentido el objetivo principal fue alcanzado y es momento de analizar los objetivos específicos y cuál es el estado particular de cada uno para el final del proceso de desarrollo:

- **Determinar los requerimientos funcionales y no funcionales del módulo de acceso y despliegue de archivos WARC:** Los requerimientos funcionales y no funcionales fueron definidos e incluidos en el proceso de desarrollo del sistema. Principalmente en la primera iteración del marco aplicativo se determinó cuales eran las funcionalidades que se requerían para este módulo, a través de una entrevista con la encargada del Archivo web y evaluando las referencias más relevantes, que fueron WayBack Machine y Oasis
- **Desarrollar dicho módulo empleando una adaptación de la metodología AUP con AgilUS y una arquitectura basada en componentes:** La metodología que se empleó se basó en las iteraciones de AUP pero con las fases de AUS así que fue efectiva su implementación y aprovechada en el proceso de desarrollo. Al ser una metodología ágil se aprovechó su flexibilidad para incorporar secciones en el proceso de desarrollo a medida de que fueran solicitadas, esta característica agilizó el proceso de desarrollo porque permitió que el proyecto se segmentara de manera incremental y a la vez estableció específicamente los requerimientos en cada iteración, permitiendo que se mantuviera el foco a lo largo de todo el proceso de desarrollo
- **Integrar el módulo desarrollado al resto de los módulos pertenecientes a la arquitectura del prototipo de archivo web:** básicamente el back-end fue el responsable de esta integración, su función en términos generales fue de servir de área intermedia entre las peticiones del front-end y las respuestas dadas por los módulos de indexación y almacenamiento. Es importante destacar que para el momento de desarrollo el resto del sistema no estaba realmente disponible para una integración real por lo que el módulo de indexación se probó con una instalación local en el mismo equipo donde se alojó el resto del módulo de acceso y, más importante aún, el módulo de almacenamiento no estaba disponible por lo que se simuló como una segunda aplicación en Django con la capacidad de descomprimir archivos de formato WARC y

una base de datos en la que se buscaban los archivos WARC descomprimidos que serían consultados.

- **Realizar pruebas de usabilidad e integración del módulo:** por lo descrito en el punto previo, la integración a pesar de ser eficaz no fue realista, por lo que se sabe que este módulo es capaz de integrarse en un módulo de almacenamiento, sin embargo, no se hizo en uno como tal, por lo que la integración solo fue parcial. Este objetivo no se considera completado a cabalidad. También los resultados arrojados por las encuestas son en su mayoría positivos, con un bajo porcentaje de desacuerdo. Con esto se puede concluir que el sistema logra el objetivo de ser fácil e intuitivo, logrando que cualquier persona con los conocimientos mínimos sea capaz de usar el sistema y ubicarse en cada una de las secciones que era una de las principales metas establecidas para poder decir que el proyecto era realmente práctico, a lo que se concluye en base a las respuestas obtenidas de que sí lo fueron.
- **Realizar pruebas de rendimiento para hacer comparación con los resultados del módulo de acceso de la versión 1 del prototipo:** los tiempos de respuesta obtenidos son positivos, hay que entender que el funcionamiento actual no es el funcionamiento final del proyecto, sin embargo es lo más cercano que se puede lograr para el momento, por lo que el tiempo de respuesta no debe variar. Con la nueva arquitectura y comportamiento los tiempos de respuesta son casi instantáneos y dependen casi únicamente del tiempo de respuesta entre las solicitudes del back-end a los otros módulos. Con cualquier consulta la respuesta es casi inmediata ya que solo se basa en consultar a dos plataformas en donde ya está almacenada la información requerida y su respuesta depende de su capacidad intrínseca para responder y no al procesamiento de un archivo, que era la razón por la cual en la versión anterior los tiempos de respuesta eran muy altos.

Para concluir este sistema logró la mayoría de los objetivos propuestos y aquellos que no se lograron fueron parcialmente realizados. Para trabajos futuros se recomienda agregar tecnologías más modernas principalmente para la gestión de los diferentes módulos que componen al Archivo Web, es decir, aprovechar la naturaleza distribuido del Archivo Web de Venezuela para implementar diferentes servicios como Kubernetes, Docker, Jenkins, entre otros, para una gestión más efectiva del sistema. Para trabajos futuros la tarea es ahora de integración con el archivo web una vez desarrollados el resto de sus módulos, por la manera en que fue desarrollado este proyecto el propósito es que sea sencilla la integración con el módulo de BI y de almacenamiento real, ya que están documentadas cada una de las secciones que lo componen y se es claro dónde se realiza cada tarea, por lo que la comunicación con otras API's depende de la experticia del desarrollador con las herramientas, lo que se requiere es agregar los nuevos puntos de acceso para estas y

llamar a los servicios. En caso del módulo de almacenamiento, se debe realizar una nueva configuración que se adapte a las condiciones de este módulo una vez esté listo, es sabido que el acceso a una base de datos depende a cual plataforma se esté usando por lo que esta .

REFERENCIAS BIBLIOGRÁFICAS

- Agile Alliance. (s.f.). *Agile Alliance*. Obtenido de Agile 101: <https://www.agilealliance.org/agile101/>
- Ambler, S. (2005). *Ambyssoft*. Recuperado el Enero de 2013, de <http://www.ambyssoft.com/unifiedprocess/agileUP.html>
- Apache Software Foundation & Stone Circle Production. (2007). *ApacheCon*. Recuperado el Enero de 2013, de <http://www.apachecon.com/eu2007/materials/solr-talk.pdf>
- Apple Inc. (2015). *Developer apple*. Obtenido de Developer Apple: <https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html>
- Archive-it. (2006). *Home: Archive-it*. Recuperado el 2012, de Archive-it Web site: <http://www.archive-it.org/>
- Beal, V. (s.f.). *Webopedia*. Obtenido de Webopedia client/server architecture: http://www.webopedia.com/TERM/C/client_server_architecture.html
- Burner, M., & Kahle, B. (Septiembre de 1996). *WWWArchive File Format Specification*. Obtenido de <http://pages.alexacompany.com/arcformat.html>
- Castillo, L. (20 de Julio de 2018). Autoria propia. Caracas, Venezuela.
- 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*.
- CyberCemetery. (2 de Febrero de 2009). *CyberCemetery Web site*. Obtenido de CyberCemetery : <https://govinfo.library.unt.edu/>
- Django Book. (2016). *Django Book*. Obtenido de Django Book: <http://djangobook.com/model-view-controller-design-pattern/>
- Django Software Foundation. (2013). *Django FAQ*. Obtenido de Django: <https://docs.djangoproject.com/en/1.11/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names>
- Donahue, L. (28 de Diciembre de 2008). *Practice*. Obtenido de Practice: <https://web.archive.org/web/20100629050840/http://www.practice.com/2008/12/29/debunking-the-wayback-machine>

- Facebook. (s.f.). *JSX*. Obtenido de JSX: <https://facebook.github.io/jsx/>
- Fowler, A. (2015). *NoSQL . For Dummies*.
- Gamez, V. (28 de 09 de 2014). *Iniciación a Elasticsearch*. Obtenido de <http://vgrdominik.es/elasticsearch/iniciacion/>
- Goel, V. (23 de Octubre de 2016). *Internet Archive*. Obtenido de Internet Archive Blog: <https://blog.archive.org/2016/10/23/defining-web-pages-web-sites-and-web-captures/>
- 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.
- IIPC. (2012). *IIPC About*. Obtenido de IIPC: <http://www.netpreserve.org/about-us/mission-goals>
- IIPC. (2012). *IIPC-Members*. Obtenido de <http://www.netpreserve.org/about-us/members>
- IIPC. (2012). *Netpreserve*. Obtenido de Netpreserve: <http://netpreserve.org/>
- Internet Archive. (Marzo de 2008). *Nutchwax - Home Page*. Obtenido de <http://archive-access.sourceforge.net>
- Internet Archive. (16 de Abril de 2014). *Internet Archive FAQ*. Obtenido de Internet Archive: <https://archive.org/about/faqs.php>
- Internet Archive. (2 de Junio de 2017). *Internet Archive Search Engine*. Obtenido de Internet Archive: <https://archive.org/search.php?query=mediatype%3Acollection>
- Internet Archive. (s.f.). *About: About the Internet Archive*. Recuperado el 2012, de Internet Archive: <http://archive.org/about/>
- ISO. (2009). *ISO. 28500 Information and documentation-WARC file format*. Nueva Zelanda.
- Jawadekar, W. S. (2010). *Knowledge Management*. McGraw-Hill Education.
- Johnson, N. (s.f.). *nicholasjohnson.com*. Obtenido de <http://nicholasjohnson.com/react/course/exercises/flux/>
- Krugler, K. (2012). *Solr-Lucene Zone. Dzone*. Recuperado el Enero de 2013, de <http://architects.dzone.com/articles/solr-hadoop-big-data-love>

- Library of Congress Web Archives. (2000). *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>
- Martínez, K. y. (2013). *Desarrollo de Modulo de Acceso a los contenidos preservados en formato WARC para un prototipo de Archivo Web de Venezuela*.
- Masanès, J. (2006). *Web Archive*. New York, USA: Springer-Verlag.
- MongoDB. (2015). *Data Model Design*. Recuperado el 07 de 2015, de <http://docs.mongodb.org/manual/core/data-model-design/>
- MrBool. (s.f.). *MrBool*. Obtenido de MrBool: <http://mrbool.com/rest-architectural-elements-and-constraints/29339>
- Myanmar Library. (25 de Octubre de 2001). *Online Burma/Myanmar Library*. Obtenido de Online Burma/Myanmar Library: <http://www.burmalibrary.org/reg.burma/archives/200110/msg00079.html>
- Ospina Torres, M., & León Luna, C. (2013). Una arquitectura basada en software libre para archivos web. *Enl@ce Revista Venezolana de Información, Tecnología y Conocimiento*, 10 (1), 53-72.
- Ospina, M., Martínez, M., Kabchi, M., & León, C. (2014). *Desarrollo de una Aplicación para Acceder a*. Caracas.
- P. Hunt, F. P. (2010). *Zookeeper: Wait-free coordination for internet-scale systems*, distributed systems.
- 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>
- 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>
- Portuguese Web Archive. (s.f.). *Home: Portuguese Web Archive*. Recuperado el 2012, de Portuguese Web Archive Web site: <http://sobre.arquivo.pt/>
- Ramírez Valenzo, M., Cuevas Valencia, R., & Martínez Castro, J. (11 de 05 de 2011). *Integración de búsquedas de texto completo en Bases de Datos NoSQL*. Obtenido de <http://revistavinculos.udistrital.edu.co/files/2012/12/integracionbusqueda.pdf>
- ReactJS. (s.f.). *ReactJS Documentation*. Obtenido de <https://reactjs.org/docs/refs-and-the-dom.html>

- 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>
- T. A. S. Foundation. (2010). *Apache zookeeper*. Obtenido de <http://zookeeper.apache.org>
- The Apache Software Foundation. (2012). *Hadoop*. Obtenido de hadoop.apache.org
- The Apache Software Foundation. (2012). *Lucene Apache*. Recuperado el Diciembre de 2012, de <http://lucene.apache.org/>
- Training.com. (11 de 9 de 2016). *Training.com*. Obtenido de <http://blog.training.com/2016/09/about-nodejs-and-why-you-should-add.html>
- UNESCO. (2003). *Charter on the Preservation of the Digital Heritage*. Australia.
- UNESCO. (2003). DIRECTRICES PARA LA PRESERVACIÓN DEL PATRIMONIO DIGITAL. Australia: Biblioteca Nacional de Australia.
- Voutssás Márquez, J. (2009). *Preservación del patrimonio documental digital en México: UNAM*. Centro Universitario de Investigaciones Bibliotecológicas.
- W3C. (9 de Diciembre de 2003). *Architecture of the World Wide Web*. Obtenido de W3C: <https://www.w3.org/TR/2003/WD-webarch-20031209/>
- W3C. (11 de Febrero de 2004). *Web Services Architecture*. Obtenido de W3C: <https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest>

