Facultad 1

Sistema de Detección de Mirrors.

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

Autor: Lester Díaz González

Tutores: Msc. Yadiel Pérez Villazón.
Ing. Rubén Reynaldo Bonachea.

La Habana 2021

Declaración de Autoría

Yo Lester Díaz González el autor de la presente tesis que tiene por título:

“Sistema de Detección de Mirrors” Realizada para la dirección Proyectos Especiales reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los _10_ días del mes de _diciembre_ del año ___2021__.

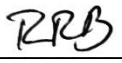
Lester Díaz González

Firma del Autor

Msc. Yadiel Pérez Villazón

Firma del Tutor

Ing. Rubén Reynaldo Bonachea



Firma del Tutor

Agradecimientos

En especial a mis padres que siempre me alentaron a estudiar una carrera para poder avanzar en la vida. A dios por poner en mi camino las posibilidades y ser sustento espiritual cuando lo necesité; a mis profesores, tutores y demás personal docente que me ayudo en la realización de este documento e investigación por la cual he dejado una deuda con la almohada de varias horas de sueño. A mis amigos los cuales siempre me tendieron una mano con todo lo que necesite aun cuando necesitaban ese tiempo para ellos, mi novia que gracias a su apoyo y su laptop logré avanzar con la edición del documento y la implementación de parte de la solución por último a esta Revolución por brindar la posibilidad a los jóvenes de estudiar en esta casa de altos estudios y así poder salir de la situación que algunos tienen en casa, en fin, a todos lo que tuvieron que ver directa o indirectamente con mi formación como informático.

Resumen

Con el creciente uso de las empresas cubanas de las plataformas digitales para la divulgación de sus productos y servicios, se busca que estos sitios web alcancen el mejor posicionamiento web posible; además de estas tendencias existe también el uso de sitios web espejos para mejorar la interacción del usuario con el contenido de las web de las empresas, esto crea la necesidad de conocer los sitios web espejos de los sitios indexados al monitor, por lo que se necesita una herramienta que lo haga de forma automática. El presente Trabajo de Diploma tiene como objetivo desarrollar una herramienta que permita identificar los espejos de un sitio web para lograr un mejor posicionamiento web de los mismos, para el desarrollo de este sistema se utilizó como Metodología de desarrollo XP, como lenguaje de programación: Java, lenguaje de modelado: UML, se usa como plataforma y framework de desarrollo: Spring boot, y entorno de desarrollo: NetBeans, como herramienta CASE: Visual Paradigma, para el control de versiones: Git (Gitlab de la Universidad), gestores de base de datos: MySQL e InfluxDB y gestor de colas de mensajes: RabbitMQ.

Palabras clave: espejos, posicionamiento web, buscadores, microservicios.

Abstract

With the growing use of Cuban companies of digital platforms for the dissemination of their products and services, it is sought that these websites achieve the best possible web positioning; In addition to these trends, there is also the use of mirror websites to improve user interaction with the content of the companies' websites, this creates the need to know the mirror websites of the sites indexed to the monitor, so it is necessary a tool that does it automatically. The present Diploma Work aims to develop a tool that allows to identify the mirrors of a website to achieve a better web positioning of them, for the development of this system it was used as XP development methodology, as a programming language: Java , modeling language: UML, used as a development platform and framework: Spring boot, and development environment: NetBeans, as a CASE tool: Visual Paradigm, for version control: Git (University's Gitlab), database managers data: MySQL and InfluxDB and message queue manager: RabbitMQ

Keywords: mirrors, web positioning, search engines, microservices.

CONTENIDO

INTRODUCCIÓN

Con el actual crecimiento del uso de las tecnologías de la información y la comunicación (TIC) por parte de las empresas y ciudadanos cubanos se ha ido desarrollando una revolución tecnológica en el país. El uso de la web por estas empresas y personas ha contribuido a una mayor interacción con el mundo de las redes sociales, así como una posibilidad de llegar a una mayor cantidad de personas y posibles clientes o potenciales inversionistas. Con el objetivo de lograr un mayor aprovechamiento de las tic se crea el Monitor de Sitios web cubanos, desarrollado por la Universidad de las Ciencias Informáticas (UCI) desde el año 2017. Este macroproyecto está compuesto por tres aplicaciones principales que permiten brindar el servicio de posicionamiento web o SEO (como es conocido según las siglas en inglés de *Search Engine Optimization*). El servicio se basa en un diagnóstico que se les realiza a los sitios web indexados por el monitor, permitiendo analizar el nivel de implementación de algunas técnicas SEO. Estas técnicas se dividen en variables e indicadores de medición, que les permite a los administrativos, webmaster y equipos editoriales de los medios digitales obtener una evaluación personalizada, ajustadas a las incidencias de sus sitios. La plataforma también apoya la toma de decisiones, pues brinda el nivel de complejidad y el impacto que tiene cada variable en cuestiones de posicionamiento web, dando a su vez un grupo de buenas prácticas que pudieran seguir, y que contribuyen a que los sitios alcancen mayor visibilidad en Internet. Actualmente la herramienta se ha centrado en el desarrollo de variables que permiten medir el nivel de integración del sitio web con las redes sociales, la optimización del sitio desde el propio desarrollo para que exista una buena comunicación del mismo con los motores de búsquedas, entre algunos elementos de usabilidad.

Llevando estos aspectos al contexto actual, cuando Cuba se encuentra en plena implementación del Gobierno Electrónico como parte de la política de informatización de la sociedad (2019), cada día es mayor el uso del Monitor de Sitio Web por parte de las entidades nacionales. Estas piden un servicio de revisión de sus sitios web y en el monitor se busca prestar el servicio lo más completo posible, es por la razón que se desea que al entrar un sitio web contar con una herramienta que automáticamente detecte si el sitio web recién indexado posee o no uno o varios sitios web espejos, esto es totalmente transparente al cliente ya que este no tiene por qué conocer el host de sus espejos o si el sitio los posee.

Actualmente la evaluación de estas variables solo se le realiza al sitio web principal, sin embargo, un sitio web puede estar alojado en diferentes servidores con características

diferentes, y que puede responder de diferentes formas, lo que trae consigo que hoy el evaluador de sitios webs solo tenga en cuenta la evaluación de sitio principal y no la de los otros sitios llamados *mirrors* (espejos). Esto provoca que no se logre correctamente la evaluación de las variables, ni se tengan en cuenta para las evaluaciones de disponibilidad y tiempo de respuesta; pudiendo ser que un espejo este respondiendo de forma lenta y el webmaster solo tenga información del sitio principal, no cumpliendo con la tarea de evaluar a estos *mirrors.*

Con la situación problemática antes descrita se puede llegar al siguiente **Problema de investigación:** ¿Cómo desarrollar una herramienta que permita conocer los mirrors de los sitios agregados al directorio de sitios web cubano?

Para la realización de esta investigación se define el **Objeto de estudio:** Posicionamiento Web y como **Campo de acción:** Sitios web espejos (*Mirrors*).

La investigación tiene como **Objetivo general:** Desarrollar un sistema que de manera automática obtenga los *mirrors* de los sitios web agregados al directorio de sitios web cubano.

Para poder cumplir el objetivo general se delimitan los siguientes **Objetivos específicos:**

- Establecer los fundamentos teórico-metodológicos para la construcción del marco teórico-conceptual de esta investigación.
- Realizar un estudio de las tendencias de estos sistemas a nivel nacional e internacional.
- Implementar el sistema de detección de *mirrors*.
- Validar la calidad de la herramienta implementada a través de pruebas de software y criterios de expertos.

Para desarrollar la investigación se definen las siguientes **Tareas de Investigación.**

1. Estudiar tanto a nivel nacional como internacional sobre el posicionamiento web y las aplicaciones de la tecnología usadas en la detección de espejos.

2. Consultar en el ámbito nacional como foráneo como se encuentran las herramientas de detección de mirrors.

3. Seleccionar las tecnologías y herramientas necesarias para el desarrollo de la aplicación.

4. Seleccionar los patrones necesarios para el desarrollo e implementación de la solución.

5. Implementar la solución de la aplicación.

6. Realizar pruebas y análisis necesarios para comprobar el correcto funcionamiento y que se hallan alcanzado los objetivos.

Se espera como **Resultado** un Sistema de Detección de *Mirrors* que brinde la información de estos sitios espejos para lograr conformar una base de datos con estos y las características de cada uno de ellos.

Para el desarrollo de la investigación se implementaron diferentes métodos científicos de investigación que permitieron comprender la necesidad de un sistema de detección de espejos para el evaluador de sitios web cubano.

Los **métodos teóricos** usados en la investigación:

Analítico-Sintético: Se usa para la recopilación de información requerida durante la realización del estudio del arte y para el desarrollo del trabajo mediante la revisión de documentos y artículos de donde se extrajeron los elementos más significativos relacionados con los sistemas de espejos, además del análisis de las diferentes metodologías, tecnologías y herramientas a utilizar para el desarrollo de la misma (Raul, 2009).

Histórico-lógico: Se usa para poder ver el desarrollo de los sitios espejos, así como su uso en la actualidad lo que ayuda a la comprensión de la necesidad de la aplicación.

Modelación: se usa en la representación mediante el uso de diagramas, de las características del sistema a desarrollar, relaciones entre objetos; y las actividades que intervinieron en el proceso de configuración del entorno colaborativo.

Los **métodos empíricos** usados fueron:

- **Entrevista:** realizada a los trabajadores del Monitor de Sitios Web Cubano para constatar las condiciones en las que se encuentra el posicionamiento web con respecto a los sistemas de espejos.
- **Observación:** se usará como punto de partida en el diseño de la investigación, ya que se pueden observar todos los procedimientos del monitor de sitios web cubanos en el tratamiento de los sitios espejos de estos. Se vera como resultado de esta observación la necesidad del desarrollo de un sistema de detección de *mirrors* al monitor de sitios web cubano.

Estructura de los capítulos:

El documento está compuesto por tres capítulos, las conclusiones generales, referencias bibliográficas y los anexos. Los capítulos están estructurados de la siguiente forma.

En **Capítulo 1(Fundamentación teórica)** se abordan los principales conceptos teóricos asociados a la investigación, se realiza un estudio de algunos sistemas con características idénticas o similares al módulo a desarrollar, además se describe la metodología de desarrollo a implementar, las herramientas informáticas y los lenguajes a utilizar en el desarrollo del módulo.

En el **Capítulo 2(Planificación y Diseño de la solución)** se realiza una descripción detallada de la propuesta de solución definiendo los elementos técnicos de la misma: los patrones, el diseño de clases, los medios empleados y las restricciones de diseño. Se representan mediante un modelo de dominio los principales conceptos que se manejan en el contexto del sistema. Se realiza una descripción de los requisitos funcionales.

En el **Capítulo 3(Implementación y pruebas del sistema)** se vislumbra la planificación de las pruebas que se le realizaran al sistema a desarrollar, para poder medir la calidad y eficiencia de acuerdo con las necesidades que requiere el cliente. Se representan los elementos físicos necesarios para un correcto despliegue de la aplicación. Se muestran componentes de la implementación. Se realiza la validación y prueba de la solución de acuerdo a los requisitos que debe cumplir para garantizar una calidad óptima, utilizando para ello las pruebas funcionales y criterios de expertos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se fundamentan los diferentes elementos teóricos que engloban la investigación, así como el desarrollo del tema. También se profundizan los aspectos y conceptos más importantes para la investigación. Además de un análisis sobre la metodología de desarrollo de software misma con la explicación por la cual se selecciona, las herramientas y las tecnologías que complementan el desarrollo e implementación de un sistema de detección de espejos en el Monitor de Sitios Web Cubanos.

1.1 Conceptos asociados a la investigación

En el presente epígrafe se estudian los conceptos fundamentales asociados a la investigación para una mejor comprensión de la misma.

Posicionamiento web

Posicionamiento web, posicionamiento en buscadores o posicionamiento SEO se refiere a las técnicas para que una página web aparezca en las primeras posiciones de los resultados en buscadores (como Google) para una serie de palabras o frases (Urosa Barreto, 2021).

El SEO trabaja en aspectos técnicos como la optimización de la estructura de una web, pero también se aplica a nivel de contenidos, con el objetivo de volverlos más útiles y relevantes para los usuarios.

El posicionamiento natural u orgánico es la que consigue una web de forma espontánea usando técnicas de posicionamiento web que tienen que ver con los contenidos de la misma (títulos, metas, palabras clave, etc), con los enlaces dirigidos a la página web y con la calidad del código, sin necesidad de que medie una campaña de publicidad o campañas de posicionamiento de pago por clic que paga directamente a los buscadores para aparecer como un enlace patrocinado: Google *Adwords*, Yahoo! *Search Marketing*, entre otras (Mousinho, 2020).

Search Engine Optimization (SEO)

El SEO orgánico o libre de pago consiste en la práctica activa de la optimización de un sitio web mediante la mejora de aspectos internos y externos con el fin de aumentar el tráfico que una página web recibe desde los motores de búsqueda sin necesidad de pagarles a los mismos. (Villacampa, 2018)

Características propias del SEO

- **Análisis de palabras clave:** Saber y comprender como las personas están buscando contenidos.
- **Web con una estructura adecuada:** Creación de páginas de destino que darán respuesta a todos los tipos de consultas que los usuarios están realizando y el contenido que buscan.
- **Contenido útil:** El objetivo específico debe ser que el contenido ayude al usuario sin importar si se trata de texto, video o una imagen.

Ventajas del uso de SEO

- **Posicionamiento a largo plazo:** Los sitios se mantienen más tiempo en las primeras páginas de los buscadores.
- **Aumento del tráfico objetivo:** Los sitios obtendrán más visitas de usuarios.
- **Aumentar ventas o conversiones:** Al tener más visitas los negocios virtuales tendrán más ventas.
- **Creación de buenos contenidos:** El sitio web tendrá mejor calidad y prestigio.

El SEO es una forma de verificar que tan bien hecho y optimizado esta un sitio web, lo cual ayuda a ganar usuarios y visitas a nuestros sitios con el uso de las tecnologías móviles y el crecimiento del comercio electrónico.

Motores de búsqueda

Los motores de búsqueda son mecanismos que organizan y distribuyen la información producida en la red a los usuarios que expresan sus dudas a través de *keywords* (palabras clave) en estos motores. (GIRALDO, 2017).

Relación SEO-motores de búsqueda

El posicionamiento en buscadores, optimización en motores de búsqueda o SEO (del inglés *Search Engine Optimization*), es un conjunto de acciones orientadas a mejorar el posicionamiento de un sitio web en la lista de resultados de Google, Bing, u otros buscadores de Internet. El SEO trabaja aspectos técnicos como la optimización de la estructura y los metadatos de una web, pero también se aplica a nivel de contenidos, con el objetivo de volverlos más útiles y relevantes para los usuarios.

Visibilidad de un sitio web

Cuando se habla de visibilidad desde el punto de vista del marketing digital, se habla de la capacidad de una marca, empresa, profesional o institución de llegar a ser visto y conocido por el mayor número de usuarios posible, particularmente aquellos que forman parte de su target objetivo (GARCÍA, 2006).

Visibilidad: capacidad de una marca empresa, negocio u otro tipo de institución de ser encontrado, visto y conocido por el mayor número de usuarios posible, particularmente aquellos que forman parte de su público objetivo. (Level, 2020).

1.2 Sistemas de mirror

Un sitio espejo (*mirror*) es un sitio web que es una réplica de un sitio ya existente, utilizado para reducir el tráfico de red (visitas a un servidor) o mejorar la disponibilidad del sitio original. Por eso, los sitios de espejo son útiles cuando el sitio original genera demasiado tráfico para que un solo servidor pueda dar soporte.

Sirve para mejorar la velocidad de descarga ante un posible colapso del servidor original, ya que así el usuario puede acceder a un *mirror* más cercano a su localidad.

Al navegar por estos sitios, se accede al mismo contenido y a la misma información de manera duplicada, con la diferencia del dominio principal en la Url.

Sistema de detección de mirros

Un sistema de detección de espejos es una herramienta que sea capaz de detectar los *mirros* de los sitios webs, le permitirá al monitor de sitios webs cubano, no solo evaluar las estadísticas de sitio original sino también de los espejos de estos, siendo más certera la evaluación de los mismos. Esta herramienta funcionará tomando de los sitios web su dirección y haciendo con esta una búsqueda uno a uno de los IP relacionados con este DNS, al obtener una relación de los mismos guardará los datos de su ubicación, código del país, así como hará por países una relación de la cantidad de espejos de un mismo sitio alojados en él; además de la cantidad de sitios *mirrors* en general que existen en los distintos países. Con la información obtenida se confeccionará un *dashboard* que permitirá al monitor de sitios web acceder a esta información.

1.3 Sistemas homólogos

Una de las primeras etapas que debe desarrollarse dentro de una investigación es el estudio del estado del arte ya que permite determinar la forma como ha sido tratado el tema, cómo se encuentra el avance del conocimiento en el momento de realizar una investigación y cuáles son las tendencias existentes para el desarrollo de la temática o problemática que se va a llevar a cabo.

El estado del arte sirve al investigador como referencia para asumir una postura crítica frente a lo que se ha hecho y lo que falta por hacer en torno a una temática o problemática concreta (PALACIO, y otros, 2014).

Para el estudio de estos sistemas se usarà la siguiente relación de medidores

Tabla 1: Relación de medidores

No	
1	Capacidad de acoplarse al sistema.
2	Compatibilidad con los lenguajes y herramientas a usarse.
3	Satisfacción de los requisitos de obtención de la información que se necesita.

Al realizar el estudio del estado del arte de aplicaciones que pueden ser utilizadas como alternativas o herramientas para la detección de espejos se encontraron los siguientes softwares, que pueden ser utilizados mediante peticiones al DNS (*Domain Name Server*) para obtener todos los IP asociados a este dominio.

nslookup: Es una herramienta gratuita que podemos utilizar en Windous, Linux y MacOS. Sirve para resolver direcciones IP asociadas a un dominio. A través de un comando podemos obtener las direcciones IP si conocemos el nombre o al contrario conocer la dirección IP si conocemos el nombre del DNS.

ping: Este comando nos permitirá saber si llegamos a un determinado sitio de forma correcta, o si por el contrario no podemos llegar. Cuando ejecutamos una solicitud de ping, este envía una solicitud *ICMP Echo Request* al host de destino y cuando el host responde lo hace con un mensaje *ICMP Echo Reply*.

Existen también varios sitios web *online* que permiten realizar consultas DNS para obtener host de los sitios tales como:

https://dnslookup.es/ : En este sito podemos hacer consultas DNS a los sitios web y devuelve como resultado la cantidad de IP que corresponden a este dominio.

https://dnschecker.org/ : Este sitio al igual que el anterior trabaja sobre la herramienta nslookup para dar una respuesta a las consultas DNS.

Resultado del estudio de los sistemas homólogos

Una vez analizadas las aplicaciones o librerías existentes, que pueden ser utilizadas para la detección de *mirrors*, se concluye que ninguna puede ser utilizada completamente como solución, pues no cumplen con todos los requisitos necesarios para el Monitor de Sitios Webs, por lo que se llega a la conclusión de la necesidad de desarrollar un sistema para la detección de espejos de los sitios web del monitor de sitios web cubano, que cumpla con los requisitos necesarios. En la siguiente tabla se relacionan los resultados.

Tabla 2 Relación de medidores y homólogos

Homólogo	Medidor 1	Medidor 2	Medidor 3
nslookup	No es posible acoplarlo ya que es un comando	Pudiera usarse, pero no daría los resultados esperados.	No brinda la información necesaria para llenar el registro que se desea llenar.
ping	No es posible acolarlo ya que es un comando	Puede usarse, pero no brinda una respuesta que satisfaga lo que se desea obtener	No brinda la información necesaria para llenar el registro que se desea llenar.
https://dnslookup.es/	No se puede acoplar ya que no posibilita un consumo de api	No se puede usar	Solo brinda información sobre uno de los espejos
https://dnschecker.org/	No se puede acoplar ya que no posibilita un consumo de api	No se puede usar	Solo brinda información sobre uno de los espejos

1.3 Metodologías de desarrollo de software

Una metodología de desarrollo de software se puede definir como un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda en la construcción de un software. (Pressman, 2001)

En la actualidad existen varias metodologías divididas en dos grandes grupos, las tradicionales y las ágiles. Las primeras se centran en elaborar una documentación exhaustiva de todo el desarrollo del software y en cumplir con la planificación realizada en la fase inicial del proyecto, sin dar margen a posibles cambios, puesto que al ocurrir uno, se ven afectados varios componentes del proceso de desarrollo del software. Otro elemento que las caracteriza es la gran cantidad de participantes con los que cuenta, pues requiere de un equipo de trabajo capaz de administrar un proceso complejo en varias etapas.

Alternativamente, surgen las metodologías ágiles, caracterizadas por ser más orientadas al desarrollo del software, con bajos niveles de formalización en la documentación requerida y por ser, a diferencia de las tradicionales, más adaptables a los cambios, requerir de pequeños grupos de trabajo y por ser apropiadas para entornos volátiles (AMARO CALDERÓN, 2007). Según las características del software solicitado, dada la premura de tiempo en que se debería construir, los requisitos de alto nivel suministrados y con el fin de aprovechar los beneficios que proporciona los procesos de construcción rápida de software, se tomó la decisión de utilizar un modelo de desarrollo de tipo ágil; dado que estos métodos permiten la implementación del software con el mínimo de documentación, en el menor tiempo posible y teniendo al cambio como una constante.

La metodología que se escoge es la de Programación Extrema ya que además de brindar beneficios para el desarrollo del proyecto, todos los demás proyectos del departamento de Proyectos Especiales usan esta metodología, lo que ayuda a estar acorde a la documentación y tecnologías del centro y en relación con los demás proyectos.

Metodología de Programación Extrema (XP)

La metodología XP o Programación Extrema es una metodología ágil y flexible utilizada para la gestión de proyectos. *Extreme Programming* se centra en potenciar las relaciones interpersonales del equipo de desarrollo como clave del éxito mediante el trabajo en equipo, el aprendizaje continuo y el buen clima de trabajo. Esta metodología pone el énfasis en la retroalimentación continua entre cliente y el equipo de desarrollo y es idónea para proyectos con requisitos imprecisos y muy cambiantes. (Joskowicz, 2015).

Características

- Se considera al equipo de proyecto como el principal factor de éxito del proyecto
- Software que funciona por encima de una buena documentación.
- Interacción constante entre el cliente y el equipo de desarrollo.
- Planificación flexible y abierta.
- Rápida respuesta a cambios.

Roles

- **Cliente**: responsable de definir y conducir el proyecto, así como sus objetivos.
- **Programadores**: Estiman tiempos de desarrollo de cada actividad y programan el proyecto.
- **Tester**: Encargado de Pruebas.
- **Tracker**: Encargado de Seguimiento.
- **Coach**: Entrenador. Su papel es guiar y orientar al equipo.
- **Big Boss**: Gestor del proyecto, gerente del proyecto, debe tener una idea general del proyecto y estar familiarizado con su estado.

1.2 Lenguajes

Un lenguaje de programación es un lenguaje formal que mediante una serie de instrucciones le permite a un programador escribir un conjunto de órdenes y algoritmos para de esta forma controlar el comportamiento físico y lógico de una máquina. Los lenguajes de programación se clasifican en dos tipos principales: lenguaje de bajo nivel y lenguaje de alto nivel; otras clasificaciones pueden ser lenguaje de programación del lado del cliente y lenguaje de programación del lado del servidor.

Java 11

Java es un lenguaje de programación y plataforma de computación lanzado por primera vez por Sun Microsystems en 1995. Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado, y más se crean todos los días. Java es rápido, seguro y confiable. Desde computadoras portátiles a centros de datos, consolas de videojuegos a supercomputadoras científicas, teléfonos celulares a Internet.

Este lenguaje será muy útil ya que es un lenguaje sobre le cual corren la mayoría de los proyectos de la Dirección de proyectos especiales, además de se muy simple y legible que otros lenguajes dará una mayor facilidad a la hora de corrgir los errores que surjan durante el desarrollo de la alpicacion.

UML

UML (*Unified Modeling Language*) o Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos y describe la semántica esencial de estos diagramas y los símbolos en ellos utilizados (Marcotte, 2010).

Este lenguaje resultará de gran utilidad durante el desarrollo del entorno colaborativo, pues con él se realizarán los diagramas para una mayor comprensión y una mayor organización durante el desarrollo del proyecto.

1.3 Herramientas y tecnologías de desarrollo informático

Las herramientas de desarrollo de software han desarrollado un importante papel desde sus inicios en el desarrollo de aplicaciones. Como parte de la ingeniería de software estas herramientas también han experimentado varios cambios, consecuencia esto del avance tecnológico propio de los últimos años.

Estos cambios de las herramientas de desarrollo de software ha sido un factor con especial influencia en el des arrollo de la ingeniería de software y de otras disciplinas relacionadas, por lo que provoca la alta tasa de cambio de las mismas.

Visual Paradigm

Visual Paradigm es una herramienta para desarrollo de aplicaciones utilizando modelado UML ideal para Ingenieros de Software, Analistas de Sistemas y Arquitectos de Sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos.

Visual Paradigm for UML Enterprise Edition (VP-UML EE): Es la edición top de la línea de productos, lo que representa lo más moderno y agrega valor en términos de modelado de datos orientado a objetos, hace posible la documentación del proyecto, mapeo relacional de objetos para Java, .NET y PHP, reduciendo costos y aumentando su productividad (Marcotte, 2010). Posibilita la representación gráfica de los diagramas permitiendo ver el sistema desde diferentes perspectivas como: componentes, despliegue, secuencia casos

de uso; clase, actividad, estado, entre otros. Además, identifica requisitos y comunica información, se centra en cómo los componentes del sistema interactúan entre ellos, sin entrar en detalles excesivos, además permite ver las relaciones entre los componentes del diseño y mejora la comunicación entre los miembros del equipo usando un lenguaje gráfico. Facilita licencias especiales para fines académicos.

MySQL

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

MySQL fue inicialmente desarrollado por MySQL AB (empresa fundada por David Axmark, Allan Larsson y Michael Widenius). MySQL AB fue adquirida por Sun Microsystems en 2008, y ésta a su vez fue comprada por Oracle Corporation en 2010, la cual ya era dueña desde 2005 de Innobase Oy, empresa finlandesa desarrolladora del motor InnoDB para MySQL.

InfluxDb

InfluxDb es una base de datos de series de tiempo de código abierto (TSDB) desarrollada por la empresa InfluxData. Está escrito en el lenguaje de programación Go para el almacenamiento y la recuperación de datos y series de tiempo en ampos como monitoreo de operaciones, métrica de aplicaciones y análisis en tiempo real.

RabbitMQ

Es un sistema de código abierto que funciona como un *middleware* de mensajería eficiente que sirve como intermediario entre productores y consumidores, en nuestro caso el monitor de sitios web y el sistema de detección de espejos a desarrollar, implementa un protocolo de capa de aplicación AMQP (Protocolo de mensaje avanzado de *queueing*) el cual esta enfocado en la comunicación de mensajes asíncronos con garantía de entrega. En otras palabras, RabbitMq define las colas que van a almacenar los sitios web indexados al monitor hasta que la aplicación del detector los analice.

Git

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

Spring Boot

Spring Boot es una herramienta que nace con la finalidad de simplificar aún más el desarrollo de aplicaciones basadas en el ya popular framework Spring Core. Spring Boot busca que el desarrollador solo se centre en el desarrollo de la solución, olvidándose por completo de la compleja configuración que actualmente tiene Spring Core para poder funcionar.

NetBeans

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE1 es un producto libre y gratuito sin restricciones de uso.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos (Actualmente Sun Microsystems es administrado por Oracle Corporation).

1.4 Conclusiones del capítulo

Mediante el estudio del estado del arte se analizaron los principales elementos que se consideran importantes para la propuesta de solución y se llega a la siguiente conclusión.

El estudio previo de las tecnologías y tendencias, permitió escoger para guiar el desarrollo de la aplicación se utiliza la herramienta Visual Paradigm para el modelado. El IDE de desarrollo que se utilizará para apoyar la programación es NetBeans. Para el control de versiones se usará Gitlab de la universidad.

CAPÍTULO 2: PLANIFICACIÓN Y DISEÑO DE LA SOLUCIÓN

El presente capítulo está dedicado a la identificación de las principales funcionalidades de la solución propuesta. En el mismo, se expondrán los artefactos generados por la puesta

en práctica de la metodología de desarrollo de software XP en sus fases de Exploración, Planificación y Diseño. Se abordará sobre cómo se le dará solución al problema de investigación antes descrito en el capítulo anterior.

2.1 Propuesta de Solución

Dada la situación problemática planteada en el presente trabajo de diploma se tiene como propuesta de solución un sistema que le permitirá al monitor de sitios web cubano detectar los sitios espejos de los sitios web, es decir un sistema de detección de estos sitios espejos.

El proceso comenzará obteniendo del directorio todos los sitios web clasificados para luego con la herramienta rabbitMQ organizar una cola con los sitios lo cual permitirá procesar estos uno a uno y hacer el proceso más eficiente sin necesidad de tener que realizar consultas constantes al directorio de sitios. El paso siguiente es detectar con nuestro sistema si tienen o no sitios espejos, en caso de resultar *true* se guardará la información de estos en una base de datos con los parámetros de los *mirrors*, luego de este paso se enviará un resumen a influxDB. En este punto se creará una *dashboard* para poder monitorizar estos *mirrors website,* este tablero permitirá que no sea necesario acoplar la aplicación al monitor de sitios web cubano, sino que este consumirá de la aplicación en el tablero(*dashboard*) antes mencionado.

2.1.1 Requisitos Funcionales

Un requisito funcional (RF) define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. Los RF pueden ser cálculos, detalles técnicos, manipulación de datos entre otras funciones que un sistema debe cumplir.

RF1: Detección de los *mirrors* de un sitio web.

RF2: Detección el IP del sitio web espejo.

RF3: Detección el código del país donde se encuentra el espejo.

RF4: Enviar resumen de los sitios web espejos a InfluxDB.

RF5: Implementación de un *endpoint* para el consultar estado de los espejos en un sitio web.

RF6: Creación de un *dashboard* para la monitorización de los sitios web y sus *mirrors.*

RF7: Envío de notificaciones a equipos de seguimiento con la información de los nuevos espejos detectados.

RF8: Guardar información de los espejos de un sitio web en MySQL.

RF9: Integración del Sistema de detección de espejos con el evaluador de variables.

2.1.2 Historias de Usuario

La metodología de desarrollo ágil de software XP tiene como uno de sus artefactos a las historias de usuario (HU), la cual consiste en una técnica para representar y especificar los requisitos del software mediante un conjunto de tablas en las que se hace una descripción breve de las características que debe tener el sistema. Estas se descomponen en tareas de programación las cuales serán implementadas durante una iteración. Son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar (Joskowicz, 2015).

Las HU son representadas mediante tablas divididas por las siguientes secciones:

• **Número:** Esta sección representa el número, incremental en el tiempo, de la historia de usuario que se describe.

• **Nombre de Historia de Usuario**: Identifica la HU que se describe entre los desarrolladores y el cliente.

• **Programador:** Rol del usuario que realiza la funcionalidad.

• **Prioridad en negocio:** Se le otorga una prioridad (Alta, Media, Baja) a las HU de acuerdo a la necesidad de desarrollo.

• **Riesgo en Desarrollo:** Se le otorga una medida de (Alto, Medio, Bajo), a la ocurrencia de errores en el proceso de desarrollo de la HU.

- Esta sección no aplica en nuestro sistema, aunque se pondrá en la tabla de la HU para cumplir con el formato de las mismas.

• **Tiempo Asignado:** Número de la iteración donde va a desarrollarse la HU.

• **Tiempo Estimado:** Es el tiempo estimado en semanas ideales(40horas) que se demorará el desarrollo de la HU.

Tiempo Real: Es el tiempo real que tomó el desarrollo de la HU

Prioridad en Negocio: Es la prioridad que tiene la HU para el negocio.

• **Descripción:** Breve descripción de la HU.

• **Observaciones:** Señalamiento o advertencia del sistema.

Tabla 3: HU1

Historia de Usuario	
Número: 1	**Usuario:** Desarrollador
Nombre: Detección de *mirrors* de un sitio web	
Programador: Lester Díaz González	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Tiempo Estimado:**
Riesgo en desarrollo: No aplica	**Tiempo Real:**
Descripción: El sistema detecta los sitios espejos del sitio web en estudio	

Tabla 4: HU2

Historia de Usuario	
Número: 2	**Usuario:** Desarrollador
Nombre: Detección del IP del sitio web espejo	
Programador: Lester Díaz González	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Tiempo Estimado:**
Riesgo en desarrollo: No aplica	**Tiempo Real:**
Descripción: Se obtiene la dirección IP correspondiente al sitio web espejo.	

Tabla 5: HU3

Historia de Usuario	
Número: 3	**Usuario:** Desarrollador
Nombre: Detección del código del país del sitio web espejo	
Programador: Lester Díaz González	**Iteración Asignada:** 1
Prioridad en negocio: Alta	**Tiempo Estimado:**
Riesgo en desarrollo: No aplica	**Tiempo Real:**
Descripción:	

Tabla 6: HU4

Historia de Usuario	
Número: 4	**Usuario:** Desarrollador

Nombre: Enviar resumen de los sitios web espejos a InfluxDB	
Programador: Lester Díaz González	**Iteración Asignada:** 2
Prioridad en negocio: Alta	**Tiempo Estimado:**
Riesgo en desarrollo: No aplica	**Tiempo Real:**
Descripción: Se envían los datos de los sitios web espejos (un resumen de espejos por países, los sitios y su cantidad de espejos y la cantidad de espejos en un país) al gestor de bases de datos.	

Tabla 7: HU5

Historia de Usuario	
Número: 5	**Usuario:** Desarrollador
Nombre: Implementación de un *endpoint* para el consultar estado de los espejos en un sitio web.	
Programador: Lester Díaz González	**Iteración Asignada:** 2
Prioridad en negocio: Alta	**Tiempo Estimado:**
Riesgo en desarrollo: No aplica	**Tiempo Real:**
Descripción: Con esto se garantiza que cuando se termine con los sitios espejos se detenga el consultor de tareas	

Tabla 8:HU6

Historia de Usuario	
Número: 6	**Usuario:** Desarrollador
Nombre: Creación de un *dashboard* para la monitorización de los sitios web y sus *mirrors.*	
Programador: Lester Díaz González	**Iteración Asignada:** 2
Prioridad en negocio: Alta	**Tiempo Estimado:**
Riesgo en desarrollo: No aplica	**Tiempo Real:**
Descripción: Se crea este tablero que va a mostrar los sitios web, así como sus espejos para su monitorización.	

Tabla 9: HU7

Historia de Usuario

Número: 7	**Usuario:** Desarrollador
Nombre: Envío de notificaciones a equipos de seguimiento con la información de los nuevos espejos detectados.	
Programador: Lester Díaz González	**Iteración Asignada:** 3
Prioridad en negocio: Alta	**Tiempo Estimado:**
Riesgo en desarrollo: No aplica	**Tiempo Real:**
Descripción: En caso de que a un sitio se le detecten espejos se enviara una notificación a los equipos de seguimiento por cada ves que se detecte un nuevo espejo.	

Tabla 10: HU8

Historia de Usuario	
Número: 8	**Usuario:** Desarrollador
Nombre: Guardar información de los espejos de un sitio web en MySQL.	
Programador: Lester Díaz González	**Iteración Asignada:** 3
Prioridad en negocio: Alta	**Tiempo Estimado:**
Riesgo en desarrollo: No aplica	**Tiempo Real:**
Descripción: Se guardan en el gestor los datos de los sitios espejos como son su identificador o ID, dirección u URL, dirección IP, código del país y la fecha en que se creó este sitio	

Tabla 11: HU9

Historia de Usuario	
Número: 9	**Usuario:** Desarrollador
Nombre: Integración del Sistema de detección de espejos con el evaluador de variables.	
Programador: Lester Díaz González	**Iteración Asignada:** 3
Prioridad en negocio: Alta	**Tiempo Estimado:**
Riesgo en desarrollo: No aplica	**Tiempo Real:**
Descripción: Así se garantiza que el evaluador de variables pueda hacer un estudio de los sitios web espejos.	

2.1.3 Plan de iteraciones

Todo proyecto que siga la metodología XP se ha de dividir en iteraciones de aproximadamente 3 semanas de duración. Al comienzo de cada iteración los clientes deben seleccionar las HU que serán implementadas.

Tabla 12: Plan de iteraciones

Iteración	Duración (Semanas)	Historias de Usuarios
1	**3**	**1-3**
2	**3**	**4-6**
3	**3**	**7-9**

Iteración 1: En esta iteración se realizará la exploración pertinente a los conceptos asociados a la investigación a desarrollar, se definirán las metodologías de desarrollo de software, así como los lenguajes y herramientas a utilizar para dar solución a la problemática.

Iteración 2: En la iteración se presenta una propuesta de la solución con un diseño, una planificación y las vías para llegar a la solución. Además, se verá cumplido lo estipulado en el capítulo 2 del documento.

Iteración 3: En esta iteración se realizarán las pruebas a la aplicación y se someterá a los criterios de expertos para así asegurar un correcto funcionamiento de la misma, al concluir estas pruebas con un resultado satisfactorio se hará entrega del programa al cliente.

2.1.4 Plan de entrega

El plan de entregas es el compromiso final del equipo de trabajo con los clientes. Esta es una cuestión de vital importancia para ambas partes, ya que la entrega tardía o temprana de la solución, repercute notablemente en la economía y la moral de los implicados (Abreu, 2019).

Tabla 13: Plan de entrega.

Plan de entrega		
Iteración 1	**Iteración 2**	**Iteración 3**
17/11/2021	**17/11/2021**	**25/11/2021**

2.2 Modelo conceptual

Un modelo conceptual es una representación de un sistema, hecho de la composición de conceptos que se utilizan para ayudar a las personas a conocer, comprender o simular un tema que represente el modelo, incluye las entidades importantes y las relaciones entre ellos.

El modelo conceptual relaciona las entidades que intervienen en el proceso de la detección y almacenamiento de los espejos, estas son:

El **directorio de sitios web** que es de donde nuestro sistema tomará los sitios a analizar.

El **gestor de colas** que se encargara de organizar la lista de los sitios obtenidos del monitor.

El **sistema de detección de *mirror*** quien es nuestra entidad más importante pues es quien se encarga de realizar el proceso de detección de los espejos de los sitios web analizados.

Y por último los **gestores de bases de datos** quienes se encargan de almacenar la información de los sitios web espejos obtenidos por el sistema de detección de espejos

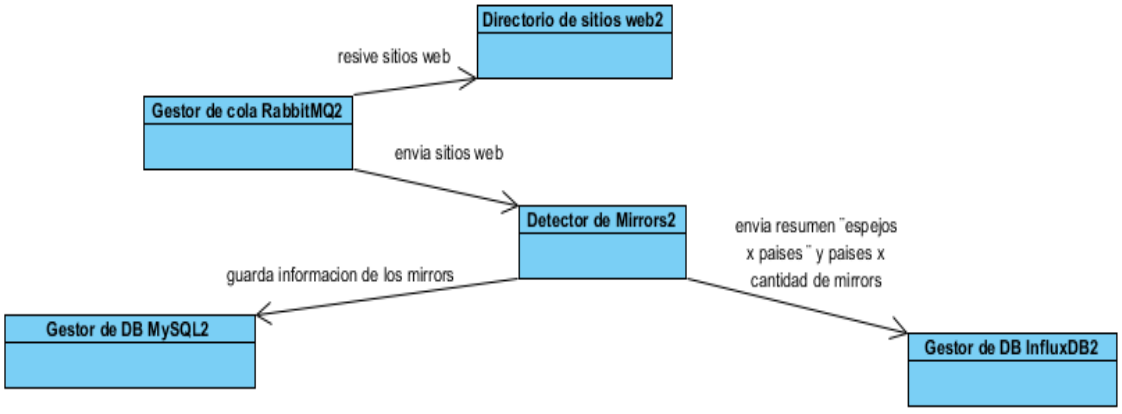


Imagen 1: modelo conceptual

2.2.1 Arquitectura de software

Para el desarrollo de los procesos es necesario definir la arquitectura de software a utilizar que se puede definir como: la estructura de un sistema o bien la forma en que va a estar

organizado este; incluye los elementos que lo conforman, sus propiedades visibles desde el exterior y las relaciones que existen entre ellos (Pressman, 2002).

La arquitectura representa la clave para comprender, organizar y comunicar un sistema, además, facilita la evolución de la solución. Es diseñada para satisfacer los requerimientos funcionales y no funcionales establecidos por los usuarios, clientes y proveedores del sistema. Un software que no posee un correcto diseño arquitectónico puede funcionar de forma muy deficiente o simplemente no funcionar generando consecuencias para la organización que sirve. Para las empresas que dependen de los sistemas de información las arquitecturas de software son fundamentales para el logro de sus objetivos organizacionales, lo que incluye el poder evolucionar rápidamente según las condiciones altamente cambiantes de los mercados actuales (Villegas, 2015).

Patrones de Arquitectura

Ofrecen soluciones a problemas de determinada arquitectura de software. Facilitan una descripción de los elementos y el tipo de relación que poseen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico formula un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. En comparación con los patrones de diseño, los patrones arquitectónicos tienen un nivel de abstracción mayor (Pressman, 2002). Algunos de estos patrones son: programación por capas, tres niveles, pipeline, invocación implícita, arquitectura dirigida por eventos, arquitectura orientada a servicios, modelo vista controlador, entre otros patrones. Para el desarrollo de la solución se seleccionó la arquitectura de micro servicios.

Arquitectura de microservicios

Los micro servicios son un tipo de arquitectura que sirve para diseñar aplicaciones. Lo que distingue a la arquitectura de micro servicios de los enfoques tradicionales y monolíticos es la forma en que desglosa una aplicación en sus funciones principales. Cada función se denomina servicio y se puede diseñar e implementar de forma independiente. Esto permite que funcionen separados sin afectar a los demás. Las aplicaciones complejas se componen de procesos independientes pequeños que se comunican entre sí mediante API que no utilizan el mismo idioma. (Vargas, 2019).

Patrones de diseño

Son valorados debido a que imponen reglas sobre la arquitectura y expresan esquemas para solucionar problemas de un mismo tipo que pueden presentarse durante el desarrollo de la aplicación. Constituyen la base para realizar la búsqueda de soluciones a problemas que se presentan en el desarrollo de software y otros marcos del diseño de interacción. Una solución es considerada un patrón de diseño cuando posee ciertas características entre las cuales se encuentran: su efectividad debe haberse comprobado resolviendo problemas similares en otras ocasiones, debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (Pressman, 2011)

Para el diseño de la propuesta de solución se tuvieron en cuenta los patrones Generales de Software para Asignación de Responsabilidades (GRASP). Entre ellos se encuentran experto en información, bajo acoplamiento, creador, alta cohesión, controlador.

Experto en información

Este patrón plantea que se debe asignar una responsabilidad al experto en información, en otras palabras, a la clase que cuenta con los datos necesarios para cumplir la responsabilidad. De esta forma, se conserva el encapsulamiento de la información, puesto que los objetos ejecutan las tareas que le corresponden de acuerdo a la información que poseen, lo que da lugar a sistemas más robustos y fáciles de mantener (Larman, 2004). Este patrón se encuentra representado en la clase *download.java*, la cual tiene la información que se descarga de un sitio web.

Bajo acoplamiento

El patrón bajo acoplamiento impulsa la asignación de responsabilidades de manera que su localización no incremente el acoplamiento hasta un nivel que lleve a los resultados negativos que puede producir un acoplamiento alto (Larman, 2004).

Alta cohesión

Este patrón plantea que se debe asignar una responsabilidad de modo que la cohesión siga siendo alta (una clase tiene responsabilidades moderadas). Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas, que no realicen un trabajo enorme. Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo.

Creador

Este patrón plantea que se debe asignar a una clase X la responsabilidad de crear una instancia de una clase Y. La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. Este patrón es el encargado de guiar la asignación de responsabilidades relacionadas con la creación de objetos.

2.3 Modelo de datos

En el modelo de datos se permite describir las estructuras de la base de datos, el tipo de datos que hay en la BD y la forma en que se relacionan. El modelado de datos describe la composición de los datos y los atributos que describen el objeto.

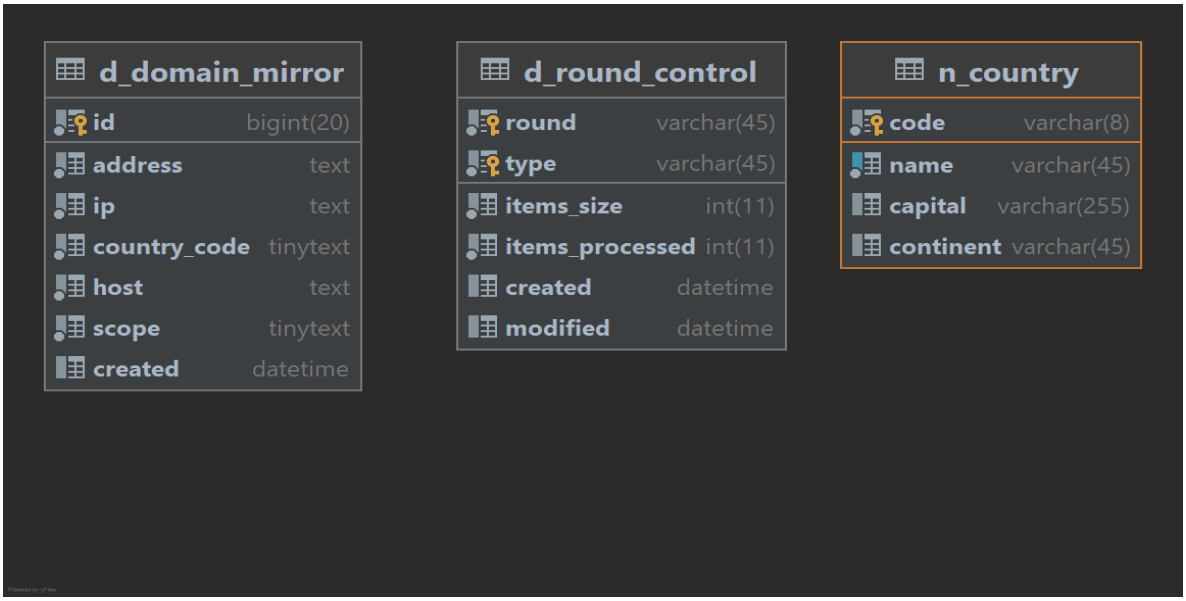


Imagen 2: Modelo de datos

2.4 Diagrama de despliegue

No es más que en un diagrama estructurado que muestra la arquitectura del sistema desde el punto de vista del despliegue o distribución de los artefactos del software en los destinos en que se propone el mismo. Define a los artefactos como representaciones de elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo. Ejemplos de artefactos son los archivos ejecutables, bibliotecas, archivos, esquemas de base de datos y archivos de configuración. Cuando se hace mención de destino de despliegue no es más que un nodo el cual es un dispositivo de hardware o un entorno de ejecución de software (Sarmiento, 2016).

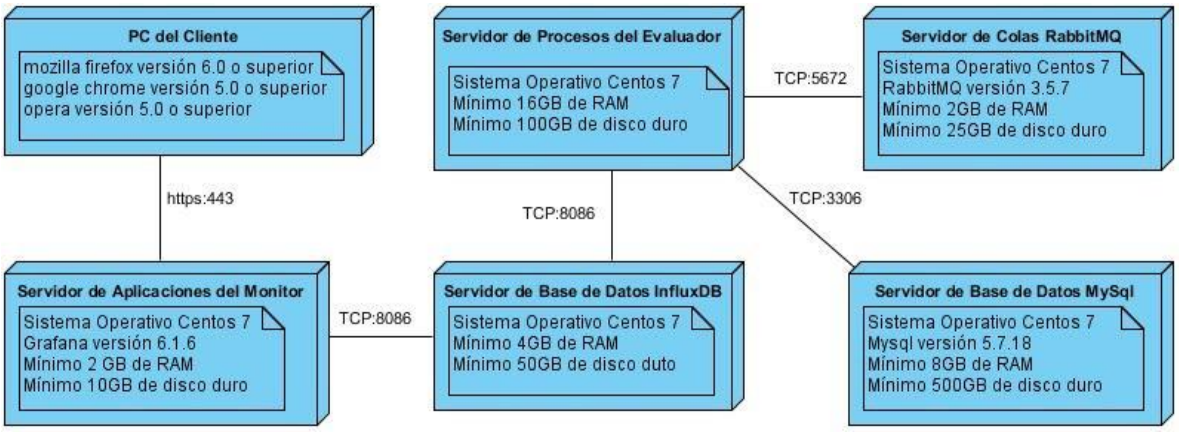


Imagen 3: Diagrama de despliegue

2.4.1 Nombre del procesador: funcionalidad y capacidad del nodo

Nodo: Elementos del proceso con al menos un procesador, memoria.

Ejemplos

- **PC del Cliente:** Son todos los dispositivos que pueden acceder a la aplicación del monitor para consultar la evaluación de un sitio web.
- **Servidor de Aplicaciones del Monitor:** Se encarga del procesamiento y control de la información que se encuentre en la aplicación del Grafana.
- **Servidor de Base de Datos InfluxDB:** Se encarga de almacenar las evaluaciones de los sitios web que fueron evaluados por el evaluador, para que sean utilizadas por el servidor de aplicaciones del monitor.
- **Servidor de Base de Datos MySQL:** se encarga de almacenar las evaluaciones de los sitios web que fueron evaluados por el sistema de detección de espejos.
- **Servidor de Procesos del Evaluador:** se encarga de ejecutar todos los procesos de evaluación de las variables.
- **Servidor de Colas RabbitMQ:** es un servidor que se encarga de definir colas, en el cual las aplicaciones se pueden conectar y transferir/leer mensajes entre ellas.

2.4.2 Nombre del tipo de conexión: Características físicas

Conectores: Expresa el tipo de conector o protocolo que relaciona dos elementos.

Ejemplos

- **Https:443:** protocolo que se utiliza para conectar el nodo PC del cliente y el servidor de aplicaciones del monitor.

- **TCP:8086:** protocolo que se utiliza para conectar los nodos servidor de aplicaciones del monitor y el servidor de procesos del evaluador con el servidor de base de datos InfluxDB.
- **TCP:5672:** protocolo que se utiliza para conectar el nodo servidor de procesos del evaluador con el servidor de colas RabbitMQ.
- **TCP:3306:** protocolo que se utiliza para conectar el nodo servidor de procesos del evaluador con el servidor de base de datos MySQL.

2.5 Conclusiones del capítulo

Se evidenció la propuesta de solución para la problemática planteada, detallada a través del modelo de dominio realizado a partir de los procesos identificados permitió conocer todos los términos y conceptos presentes en el entorno, los requisitos funcionales y las historias de usuario. Se identificaron 9 requisitos funcionales y 9 historias de usuario, en las cuales se realizaron las descripciones de los requisitos funcionales. Mediante la identificación de la arquitectura se pudo lograr una mayor organización de los elementos que darán forma a la aplicación. Se diseñaron el modelo de datos respondiendo a las exigencias y a las clases que posee la solución propuesta y el diagrama de despliegue, que permitió identificar las distintas topologías de hardware sobre las que se ejecuta el sistema.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA

Al culminar la etapa de diseño, que sienta las bases estructurales del sistema, se inicia la fase de implementación y pruebas. Se codifican las pruebas unitarias a realizar a la aplicación, el sistema no es necesario la realización de pruebas de aceptación ya que no existe un cliente en específico para que apruebe el diseño de alguna interfaz ya que la aplicación no cuenta con estas. La etapa de implementación y pruebas constituye el período con más resultados tangibles y sustanciales, pues la misma culmina con un sistema concluido y validado que garantice la solución a la problemática planteada. En este capítulo se tratan aspectos tales como los estándares de codificación que rigen el código generado, las iteraciones realizadas para la implementación de las HU, las pruebas aplicadas y sus resultados.

3.1 Estándares de codificación

Los estándares de codificación comprenden todos los aspectos de generación de código en un proyecto y permiten entender de manera rápida, fácil y sencilla el código empleado en el desarrollo de un software. Garantizan un mantenimiento óptimo del código por parte de los programadores, independientemente de si son los creadores del mismo. Reflejan un estilo armonioso como si un único programador hubiera escrito todo el código de una sola vez. La confección de estos estándares debe ser definida al comienzo de la implementación para garantizar que todos los programadores trabajen de manera coordinada. A continuación, describen algunos de los estándares seguidos para el desarrollo de la solución.

- Utilizar la siguiente estructura de organización del código:
    - Comentario de comienzo.
    - Sentencias de importación de paquetes.
    - Declaración de las clases e interfaces.
- Codificación UTF–8 en todos los módulos.
- Evitar usar espacios en blanco innecesarios.
- Utilizar el estilo *Camel Case* para nombrar clases, métodos y variables.
- Las variables declaradas como contantes deben ir totalmente en mayúsculas con las palabras separadas con una barra baja _

- Las variables de una sola letra deben ser evitadas salvo los casos comúnmente reconocidos como iteradores (i, j, k).
- Usar líneas en blanco para separar secciones, entre definición de clases e interfaces, entre métodos, entre variables y la primera sentencia.
- Usar un espacio entre una palabra clave (if, for, while) y el paréntesis a continuación. Todos los operadores excepto el punto, los de incremento ++ y decremento -- deben separarse de sus operandos con un espacio.

3.2 Pruebas

La metodología XP enfatiza en la realización de pruebas a lo largo de todo el desarrollo del software, con el fin de lograr un producto con calidad, reduciendo el número de errores y disminuyendo el tiempo transcurrido entre la aparición de un error y su corrección.

Debido a que al momento de entrega de este documento la aplicación no estaba concluida en un 100% se realiza una estrategia de pruebas para realizar a la futura aplicación ya terminada. Este software por las características de no poseer interfaces para que el usuario interactúe con el sistema no necesita pruebas de aceptación, las pruebas a realizar serán los test unitarios, así como las pruebas de acoplamiento para poder ir viendo el nivel que posee la aplicación hasta el nivel en que se encuentra.

3.2.1 Pruebas Unitarias

En la presente investigación se utilizó el nivel de pruebas unitarias utilizando la técnica de caja blanca. Estas pruebas permiten determinar si un módulo del sistema está listo y correctamente terminado. A la hora de aplicarlas se tienen en cuenta algunos parámetros, los cuales se muestran a continuación:

• Los datos de entrada son conocidos por el Encargado de Pruebas y estos deben ser preparados con minuciosidad, puesto que el resultado de las pruebas depende de estos.

• Se debe conocer qué componentes interactúan en cada caso de prueba.

• Se debe conocer de antemano, qué resultados debe devolver el componente según los datos de entrada utilizados en la prueba.

Finalmente se deben comparar los datos obtenidos en la prueba con los datos esperados, si son idénticos el módulo supera la prueba. El requisito mínimo que se establece para culminar una iteración es que todas las funcionalidades implementadas superen cada uno de sus test unitarios.

En las siguientes imágenes se muestra el resultado de los test unitarios de las funcionalidades

- Detección del IP
- Detección del país
- Detección del código del país
- Detección de la ciudad donde se aloja

Estos parámetros se evidencian en un solo test.

Imagen 4: Resultado test country code detector

Imagen 5: Resultado test IP detector

3.2.2 Pruebas de Integración

Las pruebas de integración están diseñadas para probar la interacción entre los distintos componentes de un sistema, también pueden certificar el funcionamiento o integración entre dos o más sistemas. Su objetivo es tomar los componentes probados en unidad y construir una estructura acorde a la que propone el diseño. Estas se realizan al concluir las pruebas unitarias y estas estén aprobadas

Para comprobar la correcta integración de todos los microservicios desarrollados se analizaron los procesos de conectar la herramienta al monitor para obtener los sitios indexados, se resume el resultado en la siguiente tabla:

Tabla 12: Resultado prueba de integración del Detector de Mirrors con Monitor de Sitios Web Cubano

Escenario	Descripción	Manual de Configuración	Valor Esperado	Respuesta de la Aplicación
Conectar https:443	El sistema intenta comunicarse con el monitor de sitios web cubanos para poder recibir los sitios web a evaluar.	El monitor debe estar disponible antes de realizar la evaluación.	1 si es correcta, y 0 si es incorrecta la evaluación.	El sistema verifica la conexión con el monitor para luego obtener los sitios web y enviarlos al gestor RabbitMq para que los organice.

3.3 Conclusiones del capítulo

En este capítulo se reflejan la ejecución de las pruebas al sistema dándoles cumplimiento al cuarto objetivo específico de la investigación, conste que las pruebas realizadas se le aplicaron a la parte desarrollada hasta el momento, por lo que no son contundentes a la hora de afirmar un correcto funcionamiento del total de la aplicación. El empleo de los estándares de codificación definidos facilitó la lectura, comprensión y mantenimiento del código para los desarrolladores. Con el objetivo de validar el producto se aplicaron pruebas unitarias y pruebas de acoplamiento. Las pruebas unitarias sirvieron para comprobar internamente las funciones de los módulos, facilitando la detección de errores que pudieran conllevar a futuras no conformidades. Las pruebas de acoplamiento sirvieron para certificar el correcto acoplamiento de la herramienta al Monitor.

CONCLUSIONES

Con la realización del presente Trabajo de Diploma se arribaron a las siguientes conclusiones:

- La elaboración del marco teórico de la investigación permitió sentar las bases conceptuales del sistema informático a desarrollar y estableció la naturaleza del comportamiento de sus funcionalidades.
- El análisis del estado del arte, las tendencias actuales del uso de los sitios espejos, del posicionamiento web permitió ver la necesidad existente del análisis de *mirrors* para así poder mediante las técnicas de posicionamiento SEO lograr un buen posicionamiento de los mismos.
- El uso de la metodología XP, orientada a equipos de desarrollo pequeños, permitió la obtención de un producto que responde a los objetivos de esta investigación.
- Las tecnologías y herramientas seleccionadas resultaron ser adecuadas para el desarrollo del producto facilitando el proceso de desarrollo de la solución a la problemática.
- Se desarrolló un módulo informático para el Monitor de Sitios Web Cubanos de la Universidad de las Ciencias Informáticas. El sistema desarrollado ayuda en el posicionamiento web, ya que pudiendo analizar los mirrors se identifican las debilidades de estos sitios web.
- Con el desarrollo del módulo propuesto se cumplió con el objetivo planteado en la investigación, desarrollándose una herramienta para la detección de sitios web espejos pertenecientes a los sitios web indexados al monitor de sitios web cubano.

RECOMENDACIONES

A la Dirección de Proyectos Especiales se les recomienda el despliegue de la aplicación cuando ya esté concluida y posterior a que el sistema sea liberado por calidad de software y la aplicación de las pruebas faltantes, las pruebas de seguridad, terminar de realizar los test unitarios faltantes para verificar el correcto funcionamiento de todos los componentes del software para así lograr que el despliegue de la solución se haga con la aplicación ya en estado de terminada. Además, se recomienda un estudio más profundo del uso de los sitios espejos por las empresas cubanas.

REFERENCIAS

Abreu, Marta Infante. 2019. *Conocimiento conceptual para el modelado de procesos de negocios: Revisión de la literatura. Revista Cubana de Transformación Digita.* 2019.

AMARO CALDERÓN, Sarah Dámaris. 2007. *Metodologías ágiles.* 2007.

GARCÍA, Alejandro Gonzalo Bravo. 2006. *Accesibilidad web. Un problema pendiente.* 2006.

GIRALDO, Valentina. 2017. Descubre qué son los motores de búsqueda, qué tipos existen y para qué sirve cada uno de ellos. N/A, de rockcontent. [En línea] 2017. https://rockcontent.com/es/blog/motores-de-busqueda.

Joskowicz. 2015. *Reglas Y Prácticas En Extreme Programming .* 2015.

Larman, Craig. 2004. *Agile and Iterative Development: A Manager's Guide.* 2004.

Level, Human. 2020. Human Level. [En línea] 2020. https://www.humanlevel.com.

Marcotte, Ethan. 2010.*Responsive web design.* 2010.

2019. Ministerio de comunicaciones. [En línea] 2019. https://www.mincom.gob.cu/es/gobierno-electronico.

Mousinho, Andre. 2020. rockcontent. [En línea] 2020. https://rockcontent.com/es/blog/que-es-seo/.

PALACIO, Olga Lucía Londoño, GRANADOS, Luis Facundo Maldonado y VILLAFÁÑEZ, LC Calderón. 2014. *Guía para construir estados del arte.* s.l. : International Corporation of Networks of Knowledge,, 2014. Vol. 5.

Pressman. 2002. *Ingeniería del Software.* 2002.

Pressman. 2011. *Engenharia de Software - 7.ed.* 2011.

Pressman. 2001. *INgeniería de software. Un enfoque práctico.* 2001.

Raul, Eliseo. 2009. [En línea] 2009. http://niveldostic.blogspot.com/2009/06/metodo-analitico-sintetico.html.

Sarmiento, Johana. 2016. [En línea] 2016. http://umldiagramadespliegue.blogspot.com/.

Urosa Barreto, Félix. 2021. *Posicionamiento orgánico en buscadores (SEO).* 2021.

Vargas, Ricardo González. 2019. *Arquitectura de microservicios basada en contenedores.* 2019.

Villacampa, Óscar. 2018. *Libros de SEO, el posicionamiento en buscadores al descubierto.* 2018.

Villegas, Miguel. 2015. *Hacia una arquitectura informacional .* 2015.