Universidad de las Ciencias Informáticas

Facultad 2

Título: "Migración de la Plataforma de Seguridad en las Tecnologías de la Información a Xilema-Base-Web"

Trabajo de Diploma para optar por el título de Ingeniero Informático

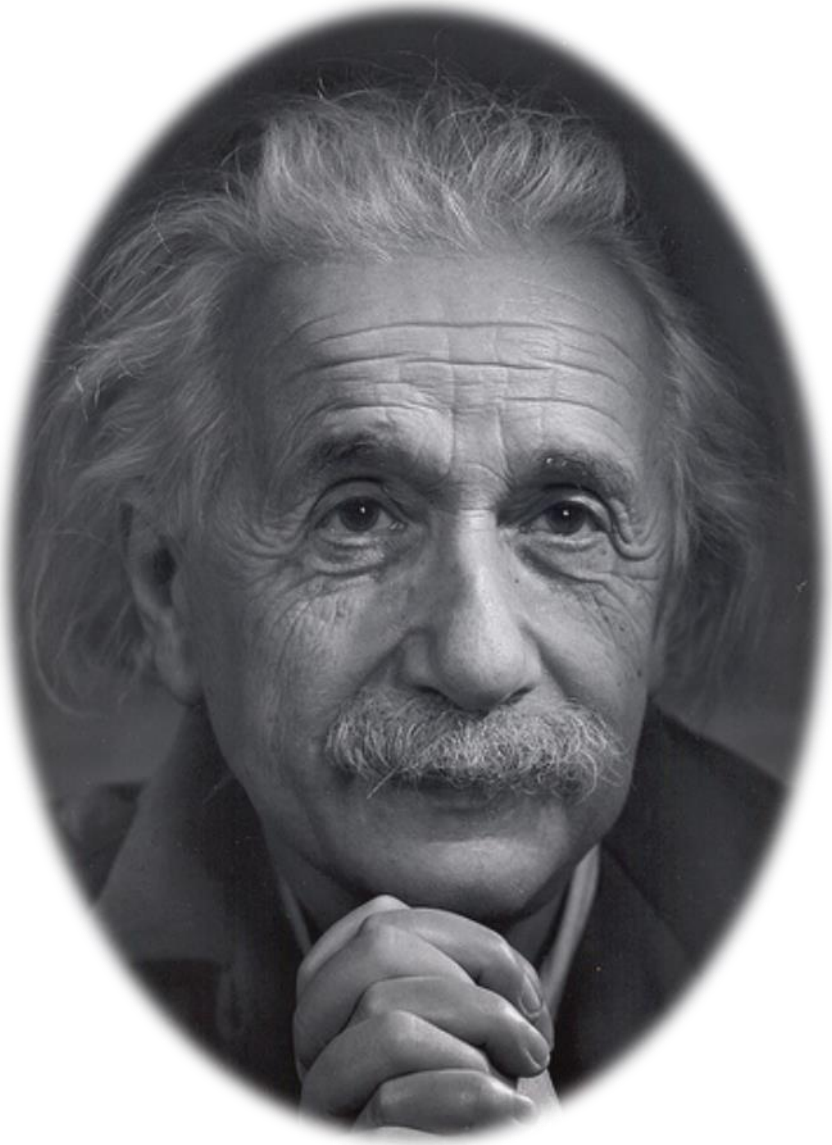
Autor(es): Kirenia Quesada Gé.
Jossue Amed Alfonso Canito.

Tutor(es): Ing. Luís Eduardo Gallardo Concepción

Ing. Acela María Sanamé Pérez

.

Junio, de 2018



"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber."

Albert Einstein.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis que tiene por título: Migración de la Plataforma de Seguridad en las Tecnologías de la Información a Xilema-Base-Web y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Kirenia Quesada Gé

_____

Firma del Autor

Jossue Amed Alfonso Canito

_____

Firma del Autor

Ing. Luís Eduardo Gallardo Concepción. Ing. Acela María Sanamé Pérez

_____ _____

Firma del Tutor

Firma del Tutor

DATOS DE CONTACTO

Síntesis de Tutor: Ing. Luis Eduardo Gallardo Concepción: Ingeniero en Ciencias Informáticas 2014. Desarrollador del proyecto “Gestor de Recursos de Hardware y Software (GRHS)”. Especialista B en Ciencias Informáticas /

Institución: Centro de Telemática (TLM) Facultad 2

Correo electrónico: legallardo@uci.cu

Síntesis del Tutor: Ing. Acela María Sanamé Pérez: Ingeniera en Ciencias Informáticas 2017. Recién Graduada en Adiestramiento

Institución: Centro de Telemática (TLM) Facultad 2

Correo electrónico: amsaname@uci.cu

AGRADECIMIENTOS

En general, agradezco a toda mi familia, porque siempre han estado presente a lo largo de esta carrera.

En especial a mi mamá Migdalia y a mi papá Carlos por ser las personas más importantes de mi vida, por estar siempre conmigo, por siempre luchar por mí y estar dispuestos a darme más de lo que podían solo por verme feliz. Por estar presente en cada momento de mi vida y demostrarme que sí se puede siempre que nos lo propongamos.

A mi hermana por ser sobre todas las cosas mi amiga y confidente, por estar siempre cuando la necesito .A mis tíos Oscar, Laida, Yordanka, Agnery Maritza. A mi prima Dayana que más que prima es mi hermana de corazón. A mis hermanas del alma Laura y Maybel que siempre estuvimos juntas no importara lo que fuera, tanto en los buenos como malos momentos. A todos mis amigos que de una forma u otra formaron parte de mi vida durante mi tiempo de estudiante en especial a la Pitufa, Idenis, Denia, Xiuny, Vero, Erick, Manuel Zaldívar, Alejandro, China, Lidia, White, Lara, Anabel, Marlon, Ronald, Joel, Yoanny y otros muchos más. A mis compañeras de apartamento Marian, Neivis, Idalis, Yeni y Karel por los buenos días que vivimos en el 3408 y el 79203.A mi novio Yasser por regalarme los días más lindos vividos y por estar siempre para mí cuando más lo necesitaba y a su familia por acogerme como otra hija más. A mi amiga Madelis por creer y ver lo bueno en mí en todo momento. A mi cuñado Miguel por ser como un hermano para mí y extenderme siempre su mano cuando lo necesito. A mi Sobrina Melanie. A mi perrita Panda. A mi otra familia Candy, Alexis y Miguel por quererme tanto o más que a su propia sangre. A mi compañero de tesis y amigo Jossué por arriesgarse conmigo y lanzarnos en esta batalla sin saber cuál sería el verdadero resultado. Gracias por compartir los trabajos de la tesis y bueno por estar aquí en estos momentos. Gracias a mis tutores, a Ramón y a Fernando por la ayuda para la

confección de este proyecto. En general a todos, los que de una forma u otra contribuyeron a mi formación como ingeniera en ciencias informática.

Kirenia Quesada Gé

A mi madre, por el gran amor y la devoción que tienes a tus hijos, por el apoyo ilimitado e incondicional que siempre me has dado, por tener siempre la fortaleza de salir adelante sin importar los obstáculos, por haberme formado como un hombre de bien, y por ser la mujer que me dio la vida y me enseñó a vivirla… no hay palabras en este mundo para agradecerte, mamá.

A mi padre, por el valor y el coraje que has tenido para levantarte ante cualquier adversidad, por las enseñanzas que me has dado, y por darme ánimos siempre diciéndome lo orgulloso que te sientes de tus hijos, muchas gracias, papá.

A mis abuelas Mirella y Gladys, porque han sido y serán siempre un ejemplo incuestionable de fortaleza, sabiduría y responsabilidad, por apoyarme incondicionalmente en todo momento, no tengo palabras para agradecerles.

A toda mi familia, por sus palabras de aliento y sus buenos deseos, especialmente a mis tíos Julio Canito, Rosa Elvira y Miralys Canito.

A mis hermanos Enmanuel y Frank por esas sonrisas y el amor incondicional que me brindan.

A mi compañera de tesis Kirenia que sin ella no hubiera sido posible llegar al final.

A todos mis amigos y compañeros de la universidad por su apoyo y las buenas y malas experiencias que vivimos juntos, sé que nos seguiremos viendo.

A todos aquellos profesores que contribuyeron a mi formación como ingeniero en ciencias informáticas, mi más sincero agradecimiento por el apoyo y constancia.

Todos aquellos familiares y amigos que no recordé al momento de escribir esto.

Ustedes saben quiénes son.

Jossué A Alfonso Canito

DEDICATORIA

Les dedico esta tesis a todos mis abuelos en especial mi abuelo Carlos y a mis abuelas Ana y Elba los cuales ya no están presentes entre nosotros y no pudieron verme realizar uno de sus sueños. A mis padres Carlos y Migdalia por luchar a mi lado durante todos estos años de estudios universitarios y por ser las personas más importantes de mi vida.

A mi hermana y mi sobrina para que cuando crezca siga el ejemplo de su tía y se forme como universitaria algún día. A todas las personas que de una forma u otra hicieron posible este triunfo brindándome su apoyo, amor y confianza.

Kirenia Quesada Gé

Jehová, por darme la oportunidad de vivir y por estar conmigo en cada paso que doy, por fortalecer mi corazón e iluminar mi mente y por haber puesto en mi camino a aquellas personas que han sido mi soporte y compañía durante todo este periodo de estudio.

Mi madre Mercedes Canito, por darme la vida, amarme mucho, creer en mi y porque siempre me apoyaste. Mamá gracias, todo te lo debo a ti.

Mi padre Josué Alfonso por los ejemplos de perseverancia y constancia que lo caracterizan, por el valor mostrado para salir adelante y por su amor.

Mis abuelas Mirella Marrero y Gladys Ramos, por quererme y apoyarme siempre, esto también se lo debo a ustedes.

Mis tíos, Julio Canito, Rosa Alfonso y Miralys Canito por estar conmigo y apoyarme siempre, los quiero mucho.

Jossué A Alfonso Canito

RESUMEN

El software en su ciclo de vida atraviesa diferentes etapas esenciales y una de estas etapas es la de su migración. Muchos son los motivos por los que se realiza la misma tales como mejorar su funcionamiento o adaptar las tecnologías existentes a un nuevo entorno de trabajo. La Universidad de las Ciencias Informáticas (UCI) cuenta con varios centros desarrolladores como es el caso del Centro de Telemática (TLM), donde se desarrolló un software llamado Plataforma de Seguridad en las Tecnologías de la Información (Xilema-PlatSI). Este sistema se encarga de detectar posibles vulnerabilidades en aplicaciones web y para ello ejecuta herramientas de auditorías web (W3af, Nikto, Zap, Acunetix y Arachni).Esta aplicación fue desarrollada en el framework Symfony2 y PHP como lenguaje de programación para la parte web. El centro TLM en su nuevo proceso de estandarización informático define como lenguaje de programación Python y como marco de trabajo Xilema-Base-Web y no al viejo framework en el que se desarrolló el producto. El objetivo de este trabajo de diploma fue realizar una migración con el fin de adaptar a Xilema-PlatSI a los estándares tecnológicos establecidos por el centro Telemática. El proceso fue guiado por la metodología ágil AUP-UCI, Python como lenguaje de programación y Xilema-Base-Web como marco de desarrollo.

PALABRAS CLAVE

auditorías web, framework, mantenimiento adaptativo, mantenimiento de software, Xilema-Platsi

ABSTRACT

The software in its life cycle goes through different essential stages and one of these stages is its migration. There are many reasons why it is carried out, such as improving its operation or adapting existing technologies to a new work environment. The University of Computer Science (UCI) has several development centers such as the Telematics Center (TLM), where a software called Security Platform in Information Technologies (Xilema-PlatSI) was developed. This system is in charge of detecting possible vulnerabilities in web applications and for this purpose, web audit tools (W3af, Nikto, Zap, Acunetix and Arachni). This application was developed in the framework Symfony2 and PHP as a programming language for the web part. The TLM center in its new process of computer standardization defines the Python programming language and Xilema-Base-Web work framework and not the old framework in which the product was developed. The aim of this diploma work was to carry out a migration in order to adapt Xilema-PlatSI to the technological standards established by the Telematics center. The process was guided by the agile AUP-UCI methodology, Python as a programming language and Xilema-Base-Web as a development framework.

KEYWORDS

web audits, framework, adaptive maintenance, software maintenance, Xilema-Platsi

TABLA DE CONTENIDOS

Índice de Tabla

Índice de Figura

INTRODUCCIÓN

Las tecnologías de la información y las comunicaciones (TIC) son cada vez más usadas para el apoyo y la automatización de todas las actividades de las empresas. Gracias a ellas, las organizaciones han conseguido adquirir importantes beneficios como la obtención de mayor cantidad de clientes y un conocimiento más profundo acerca de las necesidades para brindar un mejor servicio. Las diferentes empresas siempre van a estar expuestas a amenazas, riesgos y vulnerabilidades en sus sistemas por lo que lleva consigo que la migración de los mismos se realice de forma continua pues en el mercado existe mucha demanda sobre la calidad del software. Muchos son los motivos por los cuales se realiza la migración, tales como mejoras en el funcionamiento, la reducción del costo al optar por un software libre, pues el factor económico relativo al pago de licencias y las dificultades para adquirir el software propietario es un elemento que afecta muchas empresas hoy día.

El software primitivo no es una vía fiable para el desarrollo tecnológico y ha atravesado escenarios dentro de distintos tipos de migración caracterizada por éxitos y fracasos. Sobre esta materia se ha pronunciado el Partido Comunista de Cuba (PCC) a través de sus lineamientos 223 y 226, aprobados el 18 de abril del 2011 durante su VI Congreso. Estos lineamientos se pronuncian por "elevar la soberanía tecnológica…" así como "ejecutar inversiones en la industria electrónica y de informática y comunicaciones…" (1). Por estas razones, ha sido de vital importancia la ejecución y formación de un proceso de informatización de todo el país.

La Universidad de las Ciencias Informáticas (UCI) fue creada en el 2002 por nuestro Comandante en Jefe Fidel Castro Ruz, y es única de su tipo en el país vinculando el proceso docente con la investigación y desarrollo de software. Dicha institución cuenta con una amplia gama de centros productivos dedicados a la producción de software tanto nacional como internacional. Dentro de estos centros se encuentra el Centro de Telemática (TLM) el cual se especializa en la implementación de software en la rama de las Telecomunicaciones y la Seguridad Informática. En TLM se desarrolló un proyecto llamado Plataforma de Seguridad en las Tecnologías de la Información(Xilema-PlatSI) el cual tiene como objetivo integrar herramientas de seguridad, que faciliten la detención de vulnerabilidades que puedan originar un fallo en la Confidencialidad, Integridad, Autenticidad y/o Disponibilidad de la información. Dicho sistema se desarrolló a solicitud de la empresa de Telecomunicaciones de Cuba (ETECSA) para la realización de auditorías a las aplicaciones y sistemas de información que son alojados en sus Centros de Datos. La aplicación cuenta con un flujo de trabajo que permite la gestión de auditorías, ejecución de pruebas de

seguridad, análisis y correlación de resultados y la conformación de un informe final con las evidencias encontradas y el nivel de riesgo asociados a las mismas.

En el Centro de Telemática se definió para realización de los proyectos a Python como lenguaje de programación, a Django como marco de trabajo y se creó el framework Xilema-Base-Web que define la arquitectura del centro. A pesar del buen funcionamiento de la plataforma, actualmente la tecnología en la que está desarrollada no cumple con los estándares que son utilizados en el centro. Otro inconveniente, es que los nuevos integrantes del proyecto reciben una capacitación enfocada a dicha estandarización y no al viejo framework en el que se desarrolló el producto.

A partir de la situación antes descrita, se plantea el siguiente **problema de investigación**: ¿Cómo adaptar la Plataforma de seguridad en las tecnologías de la información (PLATSI) a Xilema-Web?

Se define como **objeto de estudio**: Los procesos de migración de sistemas informáticos. Para dar solución al problema anteriormente planteado se define como **objetivo general**: Migrar la Plataforma de Seguridad en la Tecnología y la Información para que cumpla con los estándares tecnológicos establecidos por el centro Telemática, teniendo como **campo de acción** el proceso de migración de la Plataforma de Seguridad en la Tecnología y la Información.

Para cumplir con el objetivo anteriormente planteado es necesario desarrollar las siguientes **tareas de investigación**:

- Estudios de los estándares que sigue el Centro Telemática, así como de los elementos a tener en cuenta para un proceso de migración de software.
- Selección de las tecnologías, herramientas y metodología a emplear para desarrollar la migración del software.
- Estudios de las deficiencias de la arquitectura actual del sistema para diseñar e implementar las funcionalidades del sistema.
- Investigación de los diferentes tipos de pruebas de software para su posterior aplicación sobre el sistema.

Para el desarrollo del presente trabajo investigativo se utilizaron **métodos científicos de investigación** con el fin de obtener un conocimiento del estado del arte del problema si como una recolección de datos de las herramientas a estudiar, los métodos son:

Métodos teóricos:

Analítico-Sintético: Este método posibilitó el análisis de documentos y conceptos sobre los tipos de mantenimientos existentes y sus características para luego seleccionar el más adecuado para dar solución al presente trabajo.

Métodos empíricos:

- **Observación:** Este método se utilizó para observar de forma detalla el sistema en su framework anterior y analizar toda la documentación referente al mismo.

El presente trabajo está estructurado de la siguiente forma:

Capítulo1. Fundamentación teórica de la investigación: Se abordan los aspectos y conceptos fundamentales relacionados con la migración de sistemas. De igual forma se explica y expone la metodología de desarrollo de software a utilizar y las herramientas y tecnologías necesarias para realizar la adaptación de Xilema-PlatSI a los estándares del centro de TLM.

Capítulo 2. Características del sistema: En el capítulo se presenta la propuesta de solución donde se detallan características para un mejor entendimiento del desarrollo. Se describen además una serie de artefactos definidos por la misma, tales como: Requisitos funcionales y no funcionales, casos de usos del sistema, así como sus descripciones, diagramas de clases, de casos de usos y prototipos de interfaz de usuario.

Capítulo 3. Implementación y evaluación de la solución propuesta: El capítulo contiene todo el diseño del sistema propuesto, se definen los patrones de diseño y de arquitectura y se expone todo lo relacionado con la implementación del proyecto, así como las pruebas realizadas al sistema. Además se expone todo lo relacionado con la realización de pruebas al sistema.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS DE LA INVESTIGACIÓN.

Para la realización del sistema informático es de vital importancia realizar un estudio de las herramientas, tecnologías y metodologías ms adecuadas a utilizarse. En este capítulo se abordan los distintos conceptos relacionados con el problema a resolver, así como la metodología, herramientas, tecnologías y lenguaje de programación empleados.

1.1 Plataforma de Seguridad y las Tecnologías de la Información.

Xilema-PlatSI es una herramienta informática confeccionada a la medida y necesidad del departamento de seguridad informática de ETECSA. La plataforma está pensada para contar con tres Módulos de auditorías: Módulo de Auditorías a Aplicaciones Web, Auditorías a Sistemas Operativos y Auditorías a Sistemas Gestores de Bases de Datos y actualmente solo cuenta con el módulo de auditorías a aplicaciones web.

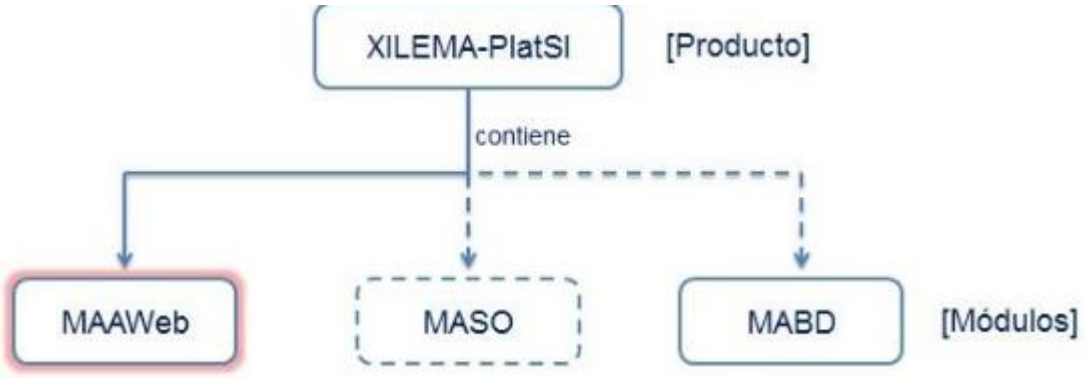


Figura 1. Estructura de Xilema-Platsi

En el mismo se integran herramientas especializadas en la realización de pruebas a aplicaciones web para verificar la seguridad de las mismas, la gestión de la ejecución de estas herramientas está dirigida por un distribuidor de tareas (Bambú). Permite además relacionar las vulnerabilidades encontradas por cada una de las herramientas en una base de datos común. Xilema-Platsi cuenta con un módulo de administración en el que se gestionan los usuarios y permisos en el sistema.

Esta aplicación web distribuye las tareas de cada una de las herramientas de forma remota y automática, posibilitando la múltiple ejecución de las mismas a través de hilos capaces de ejecutar las herramientas correspondientes a las solicitudes realizadas. Está basada en una arquitectura cliente-servidor. Las tareas son asignadas a cada uno de los agentes escogidos, y los resultados de los mismos son enviados de vuelta al servidor. Las herramientas existentes hoy en la plataforma son:

Nikto: es un código abierto, es escáner de servidor web que realiza pruebas exhaustivas a servidores web por varios elementos, incluyendo archivos potencialmente peligrosos, programas, controles de versiones no actualizadas de servidores, y los problemas específicos de la versión sobre servidores. También comprueba si hay elementos de configuración del servidor, tales como la presencia de múltiples archivos de índices, opciones de servidor HTTP, y tratará de identificar los servidores web instalados y software. Artículos de escaneo y *plugins* se actualizan con frecuencia y se pueden actualizar automáticamente. (2)

Acunetix: es una herramienta automatizada de pruebas de seguridad de aplicaciones Web. Comprueba diferentes vulnerabilidades (por ejemplo, inyección de SQL, Cross Site Scripting). Hasta la fecha Acunetix comprueba sobre más de 500 tipos diferentes de vulnerabilidades. Puede escanear cualquier sitio Web que es accesible a través del protocolo HTTP / HTTPS, básicamente, si el sitio Web se puede ver en un navegador, Acunetix puede escanearlo. También proporciona herramientas de pruebas de penetración manuales que aumentan y contribuyen a las pruebas automatizadas, así como ayudar con la prueba de vulnerabilidades. (3)

Zap: Una de las herramientas más potentes del programa **OWASP** es ZAP (**Zed Attack Proxy**). Esta plataforma está diseñada especialmente para monitorizar la seguridad de las aplicaciones web de las compañías, siendo una de las aplicaciones del proyecto más activas en cuanto a auditorías de seguridad. (4)

W3af: (Web Application Attack and Audit Framework) es una herramienta *open source* de auditoría que permite detectar vulnerabilidades web y explotarlas. Es bastante sencilla de utilizar y muy útil para automatizar diferentes análisis en un sólo proceso (5).

Arachni: es un framework modular y de alto rendimiento está desarrollado en Ruby y sirve para escanear vulnerabilidades en un sitio web. Está pensado para ayudar a los profesionales en los análisis y pruebas de penetración, también es muy útil para administradores de servidores o *webmasters* que evalúan la seguridad de las aplicaciones web modernas. Arachni es de código libre y gratuito, es multiplataforma, compatible con todos los principales sistemas operativos Windows, Mac OS y Linux. (6).

La solución se desarrolló con el marco de trabajo Symfony, utilizando como lenguaje de programación php para la aplicación web y python para el distribuidor de tareas (Bambú). Como middleware o lógica de intercambio de información entre aplicaciones, se utilizó ICE (Internet Communication Engine) para el envío de las pruebas desde la aplicación web hasta los agentes. Además, se usa como gestor de base de datos PostgreSQL. Para el manejo de la aplicación intervienen varios usuarios que son diferenciados por roles.

Entre ellos se encuentran, el solicitante que se encarga de realizar la solicitud de una auditoría, el planificador encargado de crear las auditorías, con la posibilidad de ejercer como solicitante también y asigna dichas auditorías a los especialistas. Por otra parte, se encuentran los especialistas, que ejecutan las herramientas según las auditorías que le fueron asignadas y gestionan las vulnerabilidades detectadas. Por último, se encuentra el responsable que redacta el informe final de las auditorías.

1.2 Migración de software

Las migraciones son una adaptación de los sistemas y una evolución en la innovación empresarial. Traen consigo numerosos beneficios que repercuten directamente en la productividad de las compañías: mejora los procesos de negocio, la colaboración, el rendimiento y la optimización del puesto de trabajo. (7)

Los entornos migrados (datos y aplicaciones) deberán ofrecer finalmente la misma eficiencia y operatividad que ofrecían en el entorno anterior, como así también por la necesidad de hacer mínimo el impacto en todos los niveles de la organización. (8)

1.3 Estándares informáticos del centro de Telemática a los que se migrará PlatSI

1.3.1 Django

Django es un marco de trabajo Web implementado sobre el lenguaje de programación Python, pertenece a la licencia Distribución de software de *Berkeley* (BSD). Django brinda estructura al código fuente, fomentando las buenas prácticas de desarrollo Web, lo que promueve un código legible y fácil de mantener. La implementación del patrón de diseño *Model Template View* (MTV) es una característica propia que contiene el marco de trabajo, la cual contribuye a la organización de las distintas partes de la aplicación, y a modificar estas sin afectar cualquier otra pieza del software. Su alta escalabilidad le posibilita manejar el crecimiento continuo de trabajo de manera fluida sin perder calidad en los servicios (9)

1.3.2 Python

Python es un lenguaje de programación interpretado, orientado a objetos y de alto nivel con semántica dinámica. Sus estructuras de datos integradas de alto nivel, combinadas con el tipado dinámico y el enlace dinámico, lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como también para usarlo como *scriptingo* lenguaje de pegado para conectar componentes existentes. La sintaxis simple y fácil de

aprender de Python enfatiza la legibilidad y, por lo tanto, reduce el costo del mantenimiento del programa. Python admite módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización de código. El intérprete de Python y la extensa biblioteca estándar están disponibles en formato fuente o binario sin cargo para todas las plataformas principales, y se pueden distribuir libremente (10)

1.3.3 Xilema-Base-Web

Es un marco de trabajo para la parte del cliente desarrollado en el Centro de TLM que está constituido por Django como framework base, librerías de JavaScript como son *Jquery* y *Backbone*. Además cuenta con las pautas de diseño de la UCI.

1.4 Metodología de desarrollo AUP-UCI

Toda metodología debe ser adaptada a las características de cada proyecto donde se utiliza por lo que se decide realizar una adaptación de la metodología *Agile Unified Process* (AUP) que se adaptara al ciclo de vida definido para la actividad productiva de la UCI. Por tal motivo se desarrolla la AUP-UCI para de esta forma asegurar la calidad del software que se produce en la universidad.

De las cuatro fases de AUP (Inicio, Elaboración, Construcción y Transición) la AUP-UCI decide mantener la de **Inicio**, donde se realiza un estudio con el cliente para obtener información acerca del alcance, tiempo costo y esfuerzo del desarrollo y se decide si es viable pasar a la siguiente fase que es Ejecución. En esta fase de **Ejecución** se integran las fases de Elaboración, Construcción y Transición anteriormente mencionadas de la AUP donde se ejecutan las actividades requeridas para desarrollar el software, se modela el negocio, se obtienen los requisitos, se elabora la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es entregado al cliente y se realizan capacitaciones sobre el uso del software. En la metodología AUP-UCI se decide añadir una nueva fase, **Cierre**, en la cual se analizan los resultados del producto y se llevan a cabo todos los procedimientos formales del cierre del proyecto.

Se escoge AUP-UCI como metodología de desarrollo dado que el proyecto lo desarrollan dos integrantes que ejecuta diversos roles, con tiempo limitado para el desarrollo de la solución por lo que es necesario un enfoque ágil. Se emplean las tres fases definidas: de inicio, ejecución y cierre; y de los escenarios permitidos, el 02 que define el desarrollo del proyecto mediante el empleo de casos de uso para la representación del sistema. AUP-UCI por procesos de estandarización a nivel UCI es la metodología que actualmente se utiliza en el centro por lo que su utilización en el presente trabajo de diploma garantiza que la documentación y artefactos generados estén de acorde a la generada en el mismo.

1.5 Herramientas y tecnologías utilizadas para realizar la migración de PlatSI

1.5. 1 Lenguaje de programación Python 2.7

Python es un lenguaje de programación que le permite trabajar más rápido e integrar sus sistemas de manera más efectiva. (11) Es un lenguaje de programación multiparadigma, permite varios estilos: programación orientada a objetos, programación estructural y funcional. Se desarrolla como un proyecto de código abierto, administrado por la Fundación de Software Python. Tiene gran soporte y permite la integración con otros lenguajes y herramientas.

Se utiliza Python como lenguaje de programación, siendo una de las tecnologías pertenecientes a la estandarización del centro TLM. Entre las ventajas de este lenguaje se encuentran:

- Rápido de desarrollar.
- Sencillez y velocidad.
- Sus bibliotecas hacen gran parte del trabajo. (12)

1.5.2 Marco de Trabajo: Django 1.8

Django es un framework web escrito en Python que impulsa el desarrollo de código limpio al promover buenas prácticas de desarrollo web y permite construir aplicaciones web más rápidas y con menos código que, se centra en automatizar todo lo posible y se adhiere al principio DRY (Don't Repet Yourself). (13) Este marco de trabajo da la posibilidad de adaptar y manipular de una manera sencilla los formularios de acuerdo a las necesidades de la aplicación. Además, pueden ser manipulados los datos enviados a través de los formularios y separa las interfaces evitando el cambio de la lógica del negocio en caso de variaciones en su apariencia.

1.5.3 Lenguaje de Marcas de Hipertexto (HTML5)

HTML5 es un lenguaje de marcas empleado para estructurar y presentar contenido en la web. Como su nombre lo indica es la quinta revisión del estándar HTML. (14). Es el lenguaje que utiliza Django en sus plantillas.

1.5.4 Xilema-Base-Web

En este trabajo se utiliza para darle la arquitectura de TLM al Módulo de Gestión de Auditoría Web de Xilema –PlatSI v2.0, además de utilizar el módulo de gestión de usuarios y grupos que ya incorpora.

1.5.5 Sistema Gestor de Base de Datos: PostgreSQL 9.1

PostgreSQL es un sistema de administración de bases de datos de propósito general y objeto relacional, el sistema de base de datos de código abierto más avanzado. Es un software gratuito y de código abierto. Su código fuente está disponible bajo la licencia PostgreSQL una licencia libre de código abierto. Permite agregar personalizado a funciones desarrolladas usando diferentes lenguajes de programación como Java. Está diseñado para ser extensible. Se pueden definir personalizaciones en los tipos de datos, tipos de índice, idiomas funcionales. (15)

Se utiliza en la solución para la gestión de usuarios, así como para el almacenamiento de las vulnerabilidades y su tratamiento.

1.5.6 Entorno de desarrollo integrado (IDE): PyCharm 2016.3.2

PyCharm es un entorno de desarrollo integrado (IDE) que proporciona la finalización inteligente de códigos, inspecciones de códigos, resaltado de errores sobre la marcha y soluciones rápidas, junto con refactorizaciones automáticas de códigos y capacidades de navegación avanzadas. Brinda soporte de primera clase para Python, JavaScript, TypeScript, CSS. Domina código con reconocimiento de idioma, detección de errores y arreglos de código sobre la marcha. Contiene una búsqueda inteligente para saltar a cualquier clase, archivo o símbolo, o incluso cualquier ventana de herramientas. Las refactorizaciones específicas del lenguaje y del marco ayudan a realizar cambios en todo el proyecto. (16)

1.5.7 Visual Paradigm UML 8.0

Visual Paradigm es una herramienta UML. La herramienta está diseñada para una amplia gama de usuarios, incluidos ingenieros de software, analistas de sistemas, analistas de negocios y arquitectos de sistemas, o para cualquier persona interesada en la construcción fiable de sistemas de software de gran escala con un enfoque orientado a objetos. Además, admite los últimos estándares de notación UML. Es un proveedor líder de soluciones de software que

permiten a las organizaciones desarrollar aplicaciones de calidad de manera más rápida, mejor y más económica. Visual Paradigm está dedicado a desarrollar y entregar continuamente software, servicios y asociaciones para ayudar a los clientes a transformar con precisión los requisitos de su sistema en soluciones de software de alta calidad, todas con un riesgo mínimo y un máximo retorno de la inversión. Todos los productos de Visual Paradigm están diseñados y desarrollados para eliminar la complejidad, mejorar la productividad y comprimir los plazos de tiempo de desarrollo de software de los clientes. (17) Se selecciona Visual Paradigm for UML en su versión 8.0 como la herramienta que se utilizará para realizar los diagramas y modelos necesarios en el desarrollo del software definidos en el lenguaje de modelado UML.

.

1.6 Conclusiones del capítulo

En el presente capítulo se abordaron las principales características y el funcionamiento actual de la plataforma Xilema-PlatSI. Se analizaron los diferentes conceptos de migración de software, así como los estándares a lo que se migrará PlatSI. Se define como metodología a utilizar AUP-UCI para guiar el mantenimiento siendo la misma la implantada en la universidad adaptándose al desarrollo productivo y como parte de marco de trabajo de la investigación se especificó la metodología, tecnologías y estándares de desarrollo a utilizar.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.

Introducción

En el presente capítulo se presenta la propuesta de solución donde se detallan características para un mejor entendimiento del desarrollo. Teniendo en cuenta la metodología escogida para guiar el desarrollo del presente trabajo, se describen una serie de artefactos definidos por la misma, tales como: requisitos funcionales y no funcionales, casos de usos del sistema, así como sus descripciones, diagramas de clases, de casos de usos y prototipos de interfaz de usuario.

2.1 Propuesta de solución

La propuesta de solución está enmarcada en realizar una migración de la Plataforma de Seguridad en las Tecnologías de la Información (PlatSI). Consta de una parte web a desarrollar con Django como marco de trabajo, Python como lenguaje de programación y sobre la arquitectura Xilema-Base-Web como estrategia para garantizar la calidad de sus productos. Por otro lado la plataforma PlatSI fue implementada con un sistema de autenticación por usuarios y definidos por roles, que son los encargados de realizar solicitudes, planificar, ejecutar tareas de las herramientas contenidas, además de gestionar los resultados de las mismas así como generar informes con los resultados de las auditorías realizadas, todo ello no fuera posible sin la presencia de Bambú que se encarga de manejar la ejecución de las tareas en cada uno de los clientes por medios de plugins desarrollados para cada herramienta. A continuación se muestra una imagen para un mejor entendimiento de la parte de PlatSI que se quiere migrar.

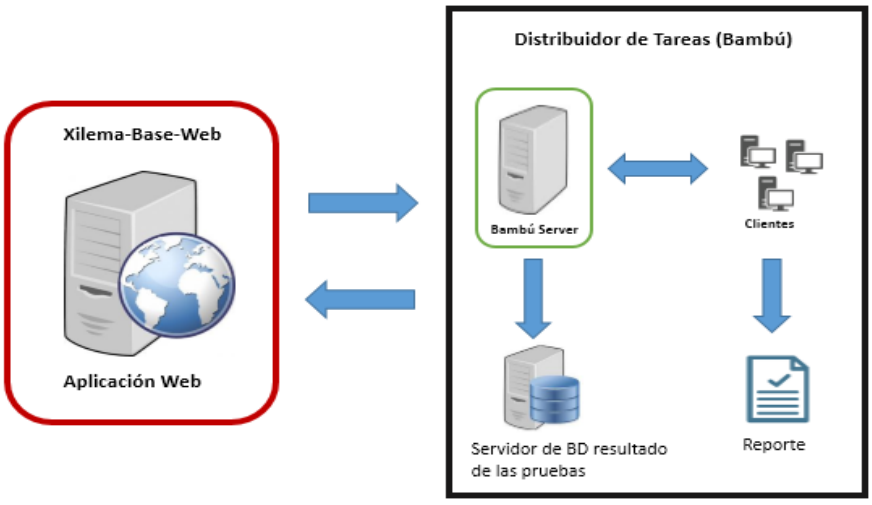


Figura 2. Propuesta de solución.

2.2 Requisitos funcionales

Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema. Un requerimiento es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. Es una definición detallada y formal de una función del sistema. (18)

Requisitos funcionales: Describen las interacciones entre el sistema y su ambiente, en forma independiente a su implementación. El ambiente incluye al usuario y cualquier otro sistema externo con el cual interactúe el sistema. (19)

Fueron identificados un total de 39 requisitos que fueron distribuidos en 25 casos de usos y a continuación se mencionan los mismos.

CU Autenticar Usuario.

- ✓ RFU1 Autenticar Usuario

CU Editar Perfil

- ✓ RFU2 Editar Perfil de Usuario

CU Listar Usuario

- ✓ RFU3 Listar Usuarios Registrados

CU Gestionar Usuarios

- ✓ RFU4 Crear Usuario
- ✓ RFU5 Modificar Usuario
- ✓ RFU6 Buscar Usuario
- ✓ RFU7 Eliminar Usuario

CU Gestionar Grupos de Usuarios

- ✓ RFU8 Crear Grupo de Usuarios
- ✓ RFU9 Modificar Grupo de Usuarios
- ✓ RFU10 Eliminar Grupo de Usuarios

CU Listar Grupos d Usuarios

- ✓ RFU11 Listar Grupo de Usuarios

CU Gestionar Solicitud

- ✓ RFU12 Crear Solicitud
- ✓ RFU13 Modificar Solicitud
- ✓ RFU14 Eliminar Solicitud

CU Listar Solicitudes

- ✓ RFU15 Listar Solicitudes

CU Gestión de Auditorías

- ✓ RFU16 Crear Auditoría
- ✓ RFU17 Editar Auditoría

CU Gestionar Asignaciones

- ✓ RFU18 Crear asignación de auditoría
- ✓ RFU19 Editar asignación de auditoría
- ✓ RFU20 Eliminar asignación de auditoría

CU Gestionar Auditorías Web

- ✓ RFU21 Crear auditoría web
- ✓ RFU22 Editar auditoría web

CU Realizar Prueba

- ✓ RFU23 Realizar prueba de seguridad

CU Correlacionar Resultados

- ✓ RFU24 Correlacionar resultados de la prueba de seguridad

CU Gestionar Vulnerabilidades

- ✓ RFU25 Agregar vulnerabilidad
- ✓ RFU26 Crear nueva vulnerabilidad
- ✓ RFU27 Editar Vulnerabilidad

CU Listar Tareas

- ✓ RFU28 Listar Tareas

CU Realizar Prueba de Concepto

- ✓ RFU29 Prueba de concepto

CU Listar Auditorías

- ✓ RFU30 Listar auditoría

CU Gestionar Informes

- ✓ RFU31 Crear Informe General
- ✓ RFU32 Editar Informe General

CU Exportar Informe General

- ✓ RFU33 Exportar Informe General

CU Listar Asignaciones

- ✓ RFU34 Listar Asignaciones

CU Buscar Auditoría Técnica

- ✓ RFU35 Buscar Auditoría técnica

CU Exportar Detalles Auditoría

- ✓ RFU36 Exportar Detalles Auditoría Web

CU Mostrar Informe Técnico

- ✓ RFU37 Mostrar Informe Técnico

CU Mostrar Tareas

- ✓ RFU38 Mostrar Detalles de las Tareas

CU Listar Auditoría Web

- ✓ RFU39 Listar Auditoría Web

2.3 Descripción de requisitos no funcionales (RNF)

Requisitos no funcionales: Describen atributos solo del sistema o del ambiente del sistema que no están relacionados directamente con los requisitos funcionales. Los requisitos no funcionales incluyen restricciones cuantitativas, como el tiempo de respuesta o precisión, tipo de plataforma (lenguajes de programación y/o sistemas operativos, etc.) (20)

El equipo de trabajo del proyecto determinó los siguientes requisitos no funciones (RNF) a tener en cuenta para la realización del presente proyecto.

2.1.1 Especificación del RNF: Usabilidad

Tabla 1 RNF Usabilidad. Comprensibilidad.

Atributo de Calidad	Usabilidad
Objetivo	Lograr que el sistema sea entendible fácilmente para los usuarios.
Origen	El usuario
Artefacto	El sistema
Entorno	El sistema está funcionando correctamente.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
NA	NA
Medida de respuesta	
NA	

Tabla 2 RNF Usabilidad. Operabilidad.

Atributo de Calidad	Usabilidad
Objetivo	Lograr que el sistema sea capaz de realizar todas las operaciones solicitadas por el cliente.

Origen	El usuario
Artefacto	El sistema
Entorno	El sistema está funcionando correctamente.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
Medida de respuesta	
NA	

2.2.2 Especificación del RNF: Funcionalidad

Tabla 3 RNF Funcionabilidad. Precisión.

Atributo de Calidad	Funcionabilidad
Objetivo	Verificar que el sistema arroje resultados o efectos acordes a las necesidades para las cuales fue creado.
Origen	Cliente del producto final
Artefacto	El sistema
Entorno	El sistema está funcionando correctamente
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**

Medida de respuesta
NA

Tabla 4 RNF Funcionalidad. Fiabilidad

Atributo de Calidad	Funcionabilidad
Objetivo	Prevenir el acceso no autorizado, ya sea accidental o premeditado a los datos.
Origen	El usuario
Artefacto	Canal de Comunicación, Sistema
Entorno	El usuario desea obtener información sobre los datos procesados por el sistema.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a Persona no autorizada intenta acceder a los datos del sistema.	
	Bloquear el acceso al sistema.
Medida de respuesta	
1 segundo	

2. a Persona con permisos restringidos intenta acceder a información a la que no tiene acceso.	
	Bloquear el acceso a estos datos.
Medida de respuesta	
1 segundo	

2.2.3 Especificación del RNF: Eficiencia

Tabla 5 RNF Eficiencia. Utilización de Recursos.

Atributo de Calidad	Eficiencia
Objetivo	Definir la cantidad de memoria RAM, capacidad de disco duro a utilizar en los clientes, el servidor y servidor de base de datos.
Origen	Externo al sistema
Artefacto	Hardware del cliente y los servidores
Entorno	Operación normal / Modo degradado
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**

NA	NA
Medida de respuesta	
NA	

2.4 Modelo Conceptual

Para un mejor entendimiento de la solución propuesta anteriormente, se detalla un modelo conceptual. Luego se describe el objetivo fundamental de cada concepto planteado en el modelo conceptual realizado.

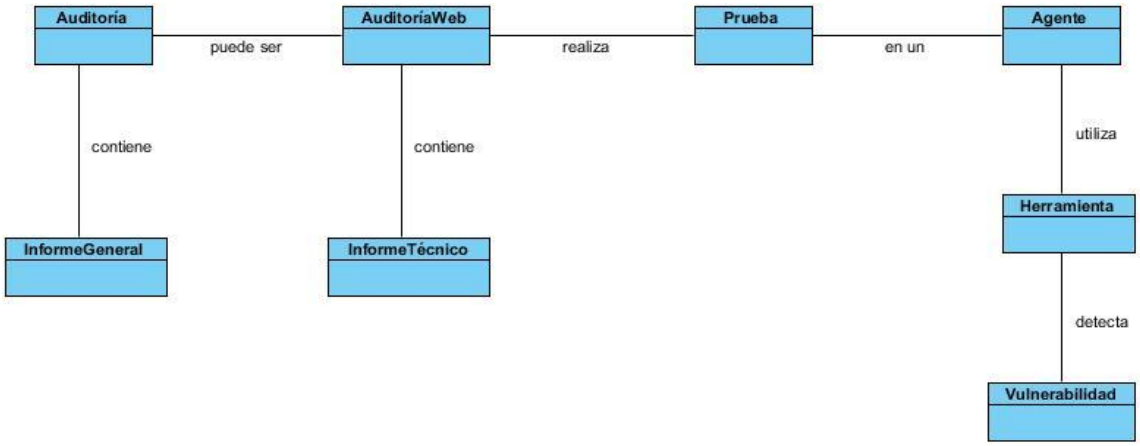


Figura 3. Modelo de conceptual.

- Auditoría: inspección o verificación de un sistema en una empresa o una entidad, realizada por un auditor con el fin de determinar y resolver las vulnerabilidades del sistema auditado.
- Auditoría Web: se refiere a que las auditorías se realizan a aplicaciones web.
- Informe General: una vez que se tienen los resultados por herramienta se conforma un reporte general con los resultados finales que son enviados al cliente, en este informe se incluyen las vulnerabilidades encontradas y las recomendaciones para eliminar las mismas.
- Informe Técnico: una vez terminada una prueba, se genera un reporte cuyo formato varía en dependencia de la herramienta empleada. Los reportes generados son revisados por el especialista que realizó la prueba, para extraer de este la información relevante y así confeccionar un informe de la misma.

- Prueba: son las pruebas de seguridad realizadas a aplicaciones web en la que se detectan vulnerabilidades.
- Agente: las pruebas se realizan utilizando el servidor distribuidor, Bambú que a través de los agentes o PC clientes ejecuta las herramientas encargadas de la búsqueda de vulnerabilidades en las aplicaciones.
- Herramienta: para realizar las pruebas se utilizan las siguientes herramientas: Nikto, Acunetix, Zap, Arachni y W3af.
- Vulnerabilidad: a partir de las pruebas de seguridad realizadas se detectan posibles brechas de seguridad y se preserva la información ante posibles ataques.

2.5 Diagrama de clases

A continuación, se muestra el modelo de clases y se explican los principales elementos que se deben tener en cuenta en cada una de ellas.

- Auditoría: esta clase es la encargada de gestionar las auditorías que se van a realizar e incluye los siguientes atributos: nombre, tipo y responsable.
- Auditoría Web: esta clase es la encargada de gestionar las auditorías web que se van a realizar e incluye los siguientes atributos: fecha de inicio, fecha de fin y especialista encargada de realizar la auditoría.
- Informe General: esta clase es la encargada de realizar el informe general e incluye los siguientes atributos: nombre y descripción.
- Informe Técnico: esta clase es la encargada de realizar el informe técnico e incluye los siguientes atributos: nombre y vulnerabilidades.
- Prueba: esta clase es la encargada de realizar las pruebas de seguridad a aplicaciones web e incluye los siguientes atributos: identificador y las herramientas que se pueden utilizar para realizar las pruebas.
- Agente: esta clase es la encargada de ejecutar las tareas a través de la utilización del servidor distribuidor, Bambú e incluye los siguientes atributos: identificador y las tareas a realizar.
- Herramienta: esta clase es la encargada de ejecutar las herramientas que se van a utilizar para realizar las pruebas e incluye los siguientes atributos: nombre y parámetros.
- Vulnerabilidad: esta clase es la encargada de gestionar las vulnerabilidades e incluye los siguientes atributos: nombre, tipo, impacto y recomendación.

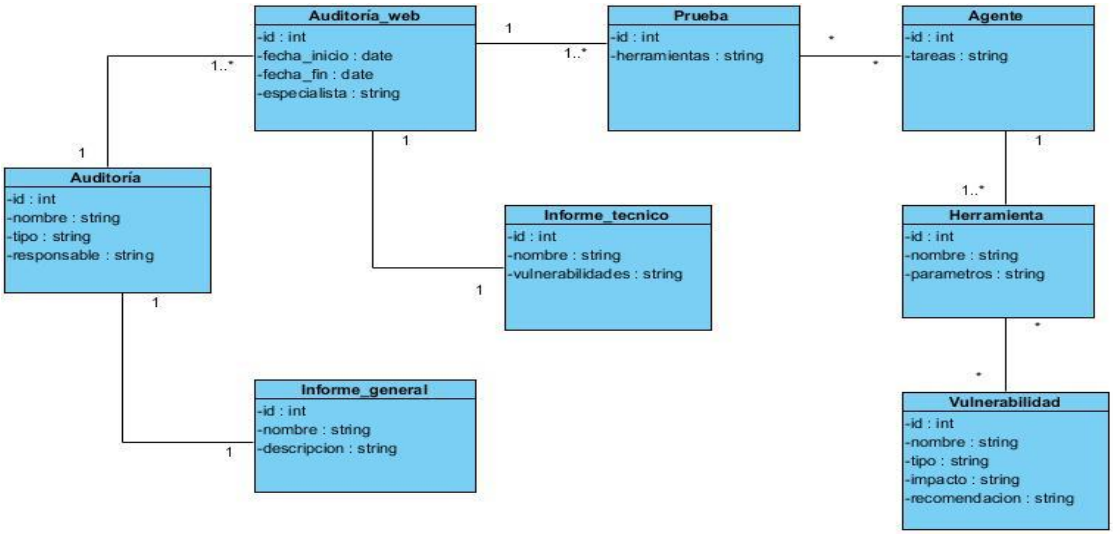


Figura 4. Diagrama de clases.

2.6 Casos de Usos del Sistema

Los casos de uso capturan el comportamiento del sistema que se está desarrollando, sin tener que especificar cómo se implementa ese comportamiento. Los casos de uso proporcionan a los desarrolladores una ruta para lograr un entendimiento común con los usuarios finales del sistema y expertos del dominio. (21)

Se encuentra identificado por cada requisito funcional un CU, mientras que un CU puede contener en él, varios requisitos funcionales. La solución está compuesta por dos módulos: Módulo de Auditoría y Módulo de Administración.

2.6.1 Descripción de Paquetes

El Módulo Administración garantiza el control de acceso a la plataforma, la gestión de usuarios y los permisos para el acceso de estos, además de la configuración de herramientas y tareas de administración. El Módulo Auditoría brinda la posibilidad de ejecutar las herramientas Acunetix, Nikto, W3AF, ZAP Y Arachni para las pruebas de seguridad de forma remota, y además conocer en tiempo real el estado de las pruebas (ejecución, detenidas, canceladas).

2.6.2 Diagrama de Paquetes

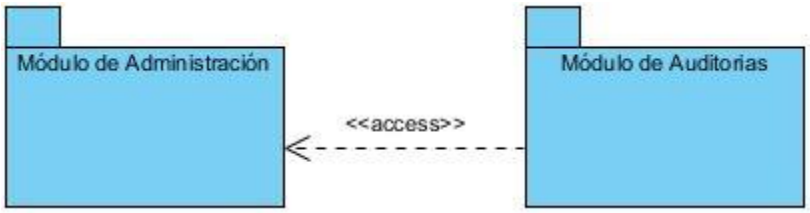


Figura 5. Diagrama de Paquete.

2.6.3 Diagrama de CU del Paquete de Administración

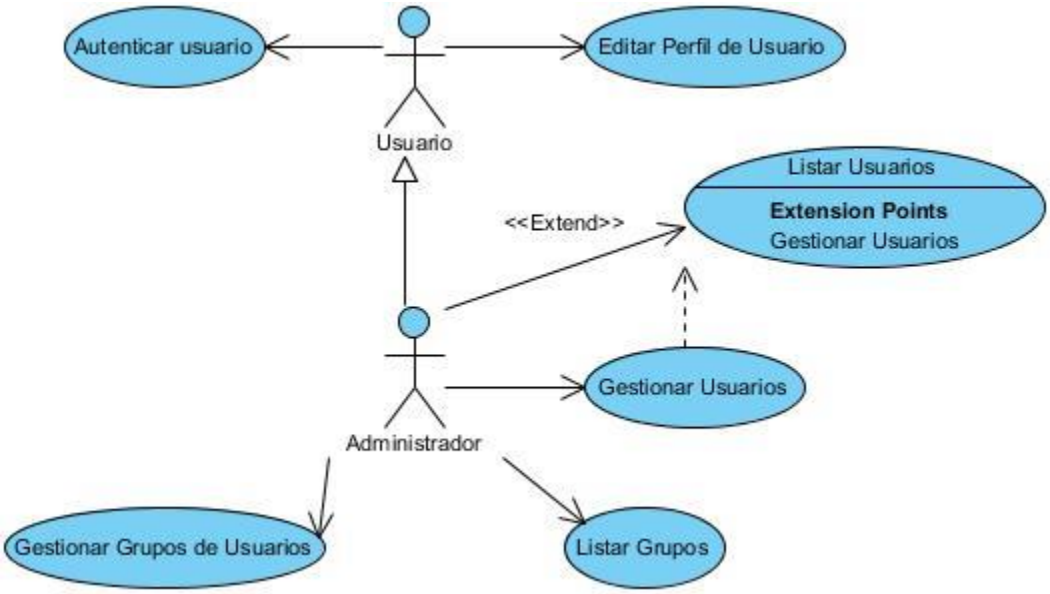


Figura 6. Diagrama de CU del paquete de Administración

2.6.4 Casos de Usos del Paquete de Administración

Tabla 6 CU Autenticar Usuario.

Objetivo	El objetivo de este CU es permitir la autenticación de los usuarios en el sistema para hacer uso del mismo.
Actores	Usuario: (Inicia) Autenticar usuario

Resumen	El caso de uso se inicia cuando un usuario desea acceder a la aplicación para hacer uso de la misma. Al autenticarse se garantiza que contenga los permisos necesarios según el grupo al que pertenece. Concluye el caso de uso cuando el usuario accede a la aplicación.
Complejidad	Alta
Prioridad	Alta
Precondiciones	N/A
Postcondiciones	N/A
Flujo de eventos	
Sección 1 “Autenticar usuario”	
Flujo básico “Autenticar usuario”	

	Actor	**Sistema**
1	El actor accede a la interfaz de inicio de la plataforma para autenticarse.	2 Muestra una ventana con los siguientes campos: Usuario y Contraseña, además muestra la opción “Entrar”.
3	Introduce los siguientes datos solicitados por el sistema: Usuario y Contraseña y selecciona la opción “Aceptar”.	4 Valida los datos introducidos. Si existen campos vacíos, ver flujo alterno 4.a. Si los datos son incorrectos ver flujo alterno 4.b

		5 El caso de uso termina cuando el usuario accede a la aplicación.
Flujos alternos		
4.a “Existen campos vacíos”		
	Actor	**Sistema**
		4.1 Muestra el mensaje:" El nombre de usuario o la contraseña son incorrectos".
		4.2 Volver a la sesión 3 del flujo básico.
4.b “Los datos son incorrectos”		
	Actor	**Sistema**
		4.1 Muestra el mensaje: “El nombre de usuario o la contraseña son incorrectos".
		4.2 Volver a la sesión 3 del flujo básico.
Relaciones	**CU incluidos**	N/A
	CU extendidos	N/A
Requisitos no funcionales	Usabilidad, Funcionalidad, Mantenibilidad	

Asuntos pendientes	N/A
Prototipo elemental de interfaz gráfica de usuario	

Tabla 7 CU Editar perfil de usuario.

Objetivo	El objetivo de este CU es permitir que el usuario edite los datos introducidos al autenticarse inicialmente.

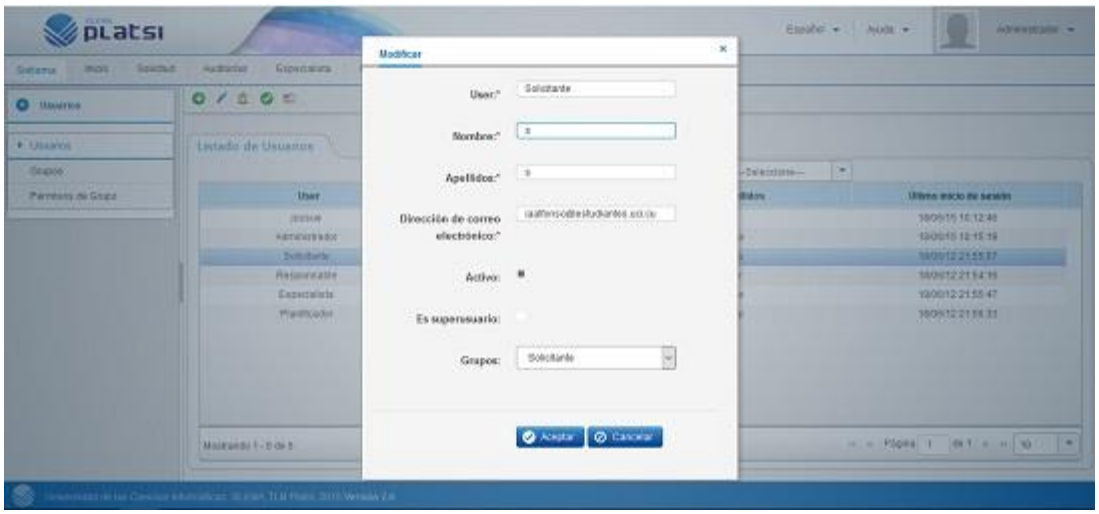
Actores	Usuario: (Inicia) Editar perfil de usuario	
Resumen	El caso de uso se inicia cuando un usuario desea editar algún o algunos de los datos introducidos inicialmente a la aplicación. Concluye el caso de uso cuando se modifican correctamente los datos insertados.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones	El usuario debe haberse registrado inicialmente en el sistema.	
Postcondiciones	Se modifican los datos deseados.	
Flujo de eventos		
Sección 1: “Editar perfil de usuario”		
Flujo básico “Editar perfil de usuario”		
	Actor	**Sistema**
1	El actor selecciona una de las opciones “Editar perfil de usuario” o “Cambiar contraseña”.	2 Muestra una interfaz con los datos de acuerdo a la opción seleccionada. Si selecciona la opción “Editar perfil de usuario” ir al paso 3 y para “Cambiar contraseña” (ver Sección 2: “Cambiar contraseña”).
3	El actor da clic en la opción “Editar perfil” en la parte superior derecha.	4 Muestra una ventana con los siguientes campos: Nombre, Apellidos, Usuario, Email y Contraseña actual.

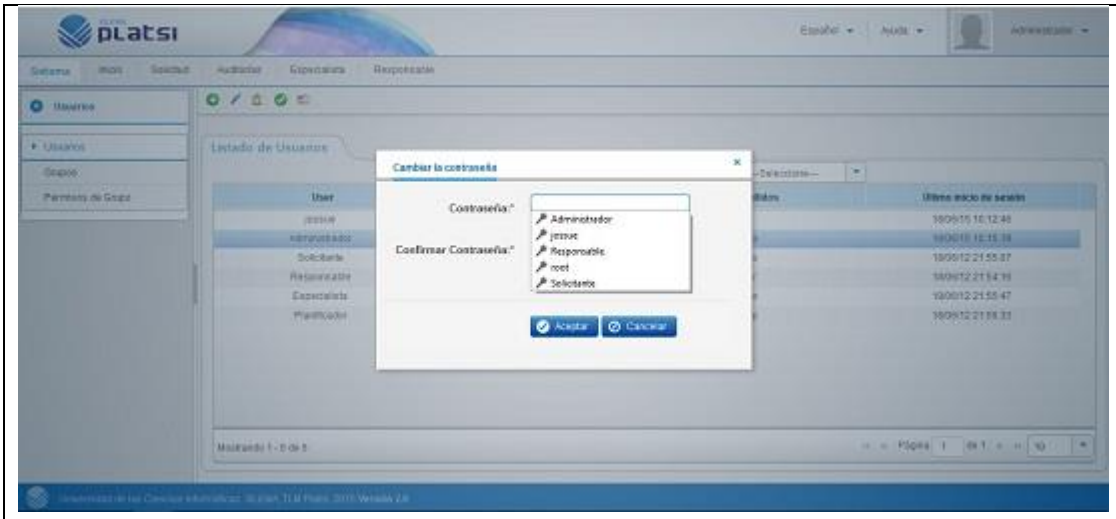
		Además, permite cambiar la contraseña. Y las opciones “Aceptar” y “Cancelar”.
5	Se modifican los siguientes datos solicitados por el sistema: Nombre, Apellidos, Usuario, Email y Contraseña actual. Selecciona la opción “Aceptar” o “Cancelar”.	6 Si selecciona la opción “Aceptar”, ver flujo básico 5. Si selecciona la opción “Cancelar”, ver flujo básico 6.
		7 Valida los datos introducidos. Si existen campos vacíos, ver flujo alterno 7.a. Si los datos son incorrectos ver flujo alterno 7.b
		8 Ver flujo alterno 8.a
		9 El caso de uso termina cuando se modifican los datos deseados.
Flujos alternos		
7.a “Existen campos vacíos”		
	Actor	**Sistema**
		7.1 Muestra el mensaje: “Existen campos vacíos.”
		7.2 Vuelve al paso 3
7.b “Los datos son incorrectos”		
	Actor	**Sistema**

		7.1 Muestra el mensaje: “Existen datos incorrectos.”
		7.2 Vuelve al paso 3
8.a Selecciona la opción “Cancelar”		
	Actor	**Sistema**
		8.1 Vuelve al paso 1 del flujo básico de eventos.
Sección 2: “Cambiar contraseña”		
2 Flujo básico “Cambiar contraseña”		
	Actor	**Sistema**
1	El actor da clic en la opción “Cambiar contraseña” en la parte inferior derecha de la ventana “Editar perfil”.	2 Se muestran los datos que pueden ser modificados: Nueva contraseña, Contraseña actual y Repetir contraseña. Además, muestra los botones “Aceptar” y “Cancelar”.
3	Se modifican los siguientes datos solicitados por el sistema: Nueva contraseña, Contraseña actual y Repetir contraseña. Selecciona la opción “Aceptar” o “Cancelar”.	4 Si selecciona la opción “Aceptar”, ver flujo básico 5. Si selecciona la opción “Cancelar”, ver flujo básico 6.
		5 Valida los datos introducidos. Si existen campos vacíos, ver flujo alterno 5.a. Si los datos son incorrectos ver flujo alterno 5.b

		6 Ver flujo alterno 6.a
		7 El caso de uso termina cuando se modifican los datos de la contraseña.
Flujos alternos		
5.a “Existen campos vacíos”		
	Actor	**Sistema**
		5.1 Muestra el mensaje: “Existen campos vacíos.”
		5.2 Vuelve al paso 3 del flujo básico.
5.b “Los datos son incorrectos”		
	Actor	**Sistema**
		5.1 Muestra el mensaje: “Las contraseñas no coinciden”.
		5.2 Vuelve al paso 3 del flujo básico.
6.a Selecciona la opción “Cancelar”		
	Actor	**Sistema**
		6.1 Vuelve al paso 1 del flujo básico de eventos.

Relaciones	**CU incluidos**	N/A
	CU extendidos	N/A
Requisitos no funcionales	Usabilidad, Funcionalidad, Mantenibilidad	
Asuntos pendientes	N/A	
Prototipo elemental de interfaz gráfica de usuario		





2.6.5 Diagrama de CU del Paquete de Auditoría

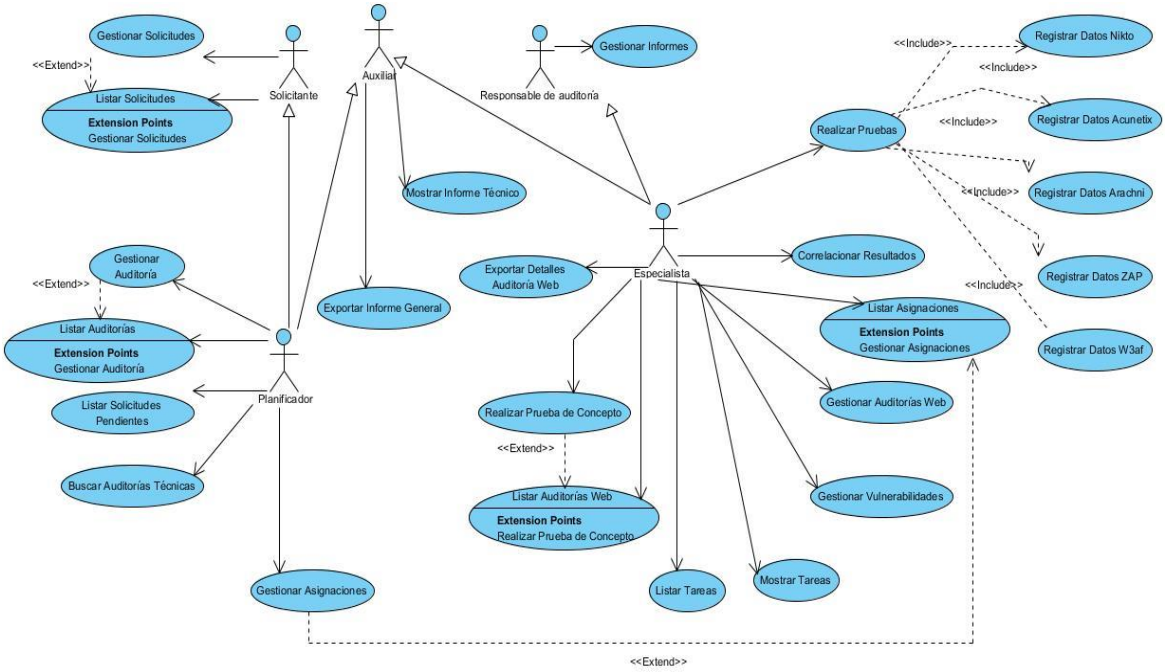


Figura 7. Diagrama de CU del Paquete de Auditoría.

2.6.6 Casos de Usos del Paquete de Auditoría

Tabla 8 CU Gestionar Solicitud

Objetivo	Entrar los principales datos de la entidad que hace la solicita.
Actores	Solicitante: (Inicia) Gestionar Solicitud
Resumen	El CU se inicia cuando una entidad solicita una auditoría a sus aplicaciones web y concluye una vez que se realizó la misma por uno de los usuarios solicitantes.
Complejidad	Alta
Prioridad	Alta
Precondiciones	N/A
Postcondiciones	N/A

Flujo de eventos		
Sección 1: “Gestionar Solicitud”		
	Actor	**Sistema**
1	El actor selecciona la opción que desea realizar.	2 Muestra una interfaz con los datos asociados a la opción seleccionada. Si desea crear una solicitud (ver Sección 2: “Crear solicitud”). Si desea modificar los datos de una solicitud (ver Sección 3: “Modificar solicitud”). Si desea ver

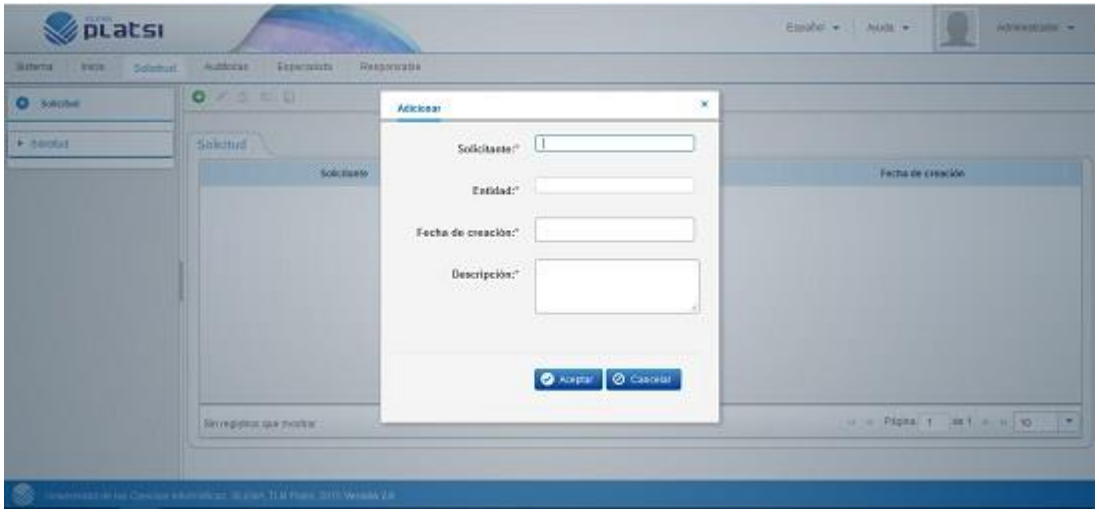
		los detalles de una solicitud (ver Sección 4: “Detalles de la solicitud”).
		3 Termina el caso de uso.
Sección 2: “Crear Solicitud”		
Flujo básico “Gestionar Solicitud”		
	Actor	**Sistema**
1	El actor accede a crear una solicitud a través de la opción “Crear solicitud”.	2 Muestra una interfaz con los siguientes campos a llenar: Solicitante, Entidad, Fecha de Creación y Descripción. Además, muestra las opciones “Aceptar” y “Cancelar”.
3	El usuario ingresa los siguientes datos: Solicitante, Entidad, Fecha de Creación y Descripción. Selecciona una de las opciones “Aceptar” o “Cancelar”.	4 Si selecciona la opción “Aceptar”, ver flujo básico 5. Si selecciona la opción “Cancelar”, ver flujo básico 6.
		5 Valida los datos introducidos. Si existen campos vacíos, ver flujo alterno 5.a. Si los datos son incorrectos ver flujo alterno 5.b
		6 Ver flujo alterno 6.a
		7 El caso de uso termina cuando se crea la solicitud correctamente.

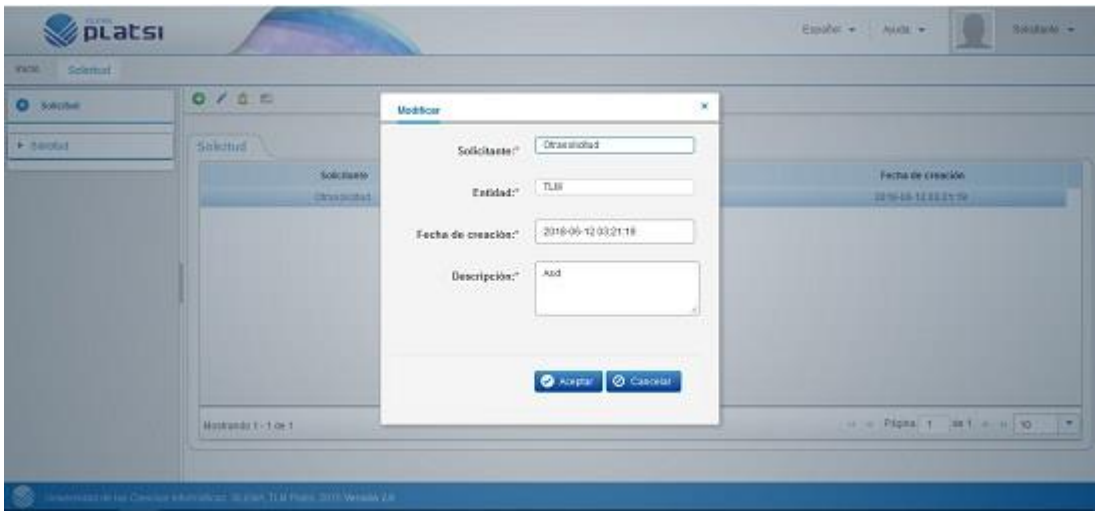
Flujos alternos		
5.a “Existen campos vacíos”		
	Actor	**Sistema**
		5.1 Muestra el mensaje “Rellene este campo”.
		5.2 Volver a la sección 3 del flujo básico.
5.b “Los datos son incorrectos”		
	Actor	**Sistema**
		5.1 Muestra el mensaje “Solo puede contener letras y comenzar con mayúscula”.
		5.2 Muestra el mensaje “Solo puede contener letras y números”.
		5.3. Muestra el mensaje “Este valor no es válido.”.
		5.4. Muestra el mensaje “Este valor es demasiado largo. Debería tener 60 caracteres o menos.”.
		5.5. Volver a la sección 3 del flujo básico.
6.a Se selecciona el botón “Cancelar”		

	Actor	**Sistema**
		6.1 Se muestra una nueva vista con la Lista de solicitudes pendientes.
Sección 3: “Modificar solicitud”		
Flujo básico “Modificar solicitud”		
	Actor	**Sistema**
1	El actor accede a modificar los datos de la solicitud a través del botón “Modificar”.	2 Muestra una interfaz con los siguientes campos: Solicitante, Entidad, Fecha de Creación y Descripción. Además se muestran los botones “Aceptar” y “Cancelar”.
3	Introduce los siguientes datos: Solicitante, Entidad, Fecha de Creación y Descripción. Y selecciona la opción “Aceptar” o “Cancelar”.	4 Si selecciona la opción “Aceptar”, ver flujo básico 5. Si selecciona la opción “Cancelar”, ver flujo básico 6.
		5 Valida los datos introducidos. Si existen campos vacíos, ver flujo alterno 5.a. Si los datos son incorrectos ver flujo alterno 5.b
		6 Ver flujo alterno 6.a
		7 Termina el caso de uso cuando se modifican los datos del usuario.
Flujos alternos		

5.a “Existen campos vacíos”		
	Actor	**Sistema**
		5.1 Muestra el mensaje: “Existen campos vacíos.”
		5.2 Vuelve al paso 2 del flujo básico.
5.b “Los datos son incorrectos”		
	Actor	**Sistema**
		3.1. Muestra el mensaje “Solo puede contener letras y comenzar con mayúscula”.
		3.2. Muestra el mensaje “Solo puede contener letras y números”.
		3.3. Muestra el mensaje “Este valor no es válido.”.
		3.4. Muestra el mensaje “Este valor es demasiado largo. Debería tener 60 caracteres o menos.”.
		3.5. Volver a la sección 2 del flujo básico.
6.a Selecciona la opción “Cancelar”		
	Actor	**Sistema**

		6.1 Vuelve al paso 1 del flujo básico de eventos.
Sección 3: “Detalles de la solicitud”		
Flujo básico “Detalles de la solicitud”		
1	El actor accede a ver los detalles de la solicitud mediante la opción “Detalles”.	2 Muestra los datos de la solicitud.
		3 Termina el caso de uso cuando se han visualizado los detalles de la solicitud.
Relaciones	**CU incluidos**	N/A
	CU extendidos	CU Listar solicitudes
Requisitos no funcionales	Usabilidad, Funcionalidad, Mantenibilidad	
Asuntos pendientes	N/A	
Prototipo elemental de interfaz gráfica de usuario		


Adicionar
Solicitante:*
Entidad:*
Fecha de creación:*
Descripción:*
Aceptar
Cancelar


Modificar
Solicitante:*
Otrasolicitud
Entidad:*
TLM
Fecha de creación:*
2018-06-12 03:21:19
Descripción:*
Aad
Aceptar
Cancelar

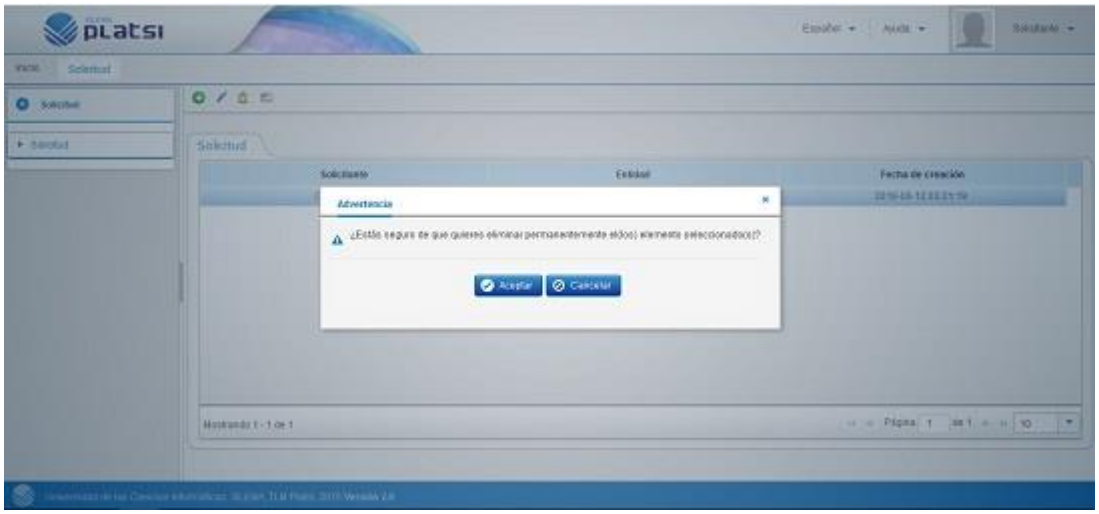

Advertencia
¿Estás seguro de que quieres eliminar permanentemente el(los) elemento seleccionado(s)?
Aceptar
Cancelar

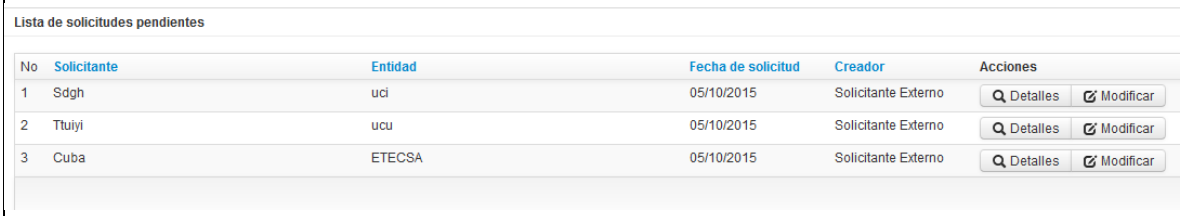
Figura 13 Prototipo de interfaz de usuario. CU: Eliminar Solicitud.

Tabla 9 CU Listar Solicitud.

Objetivo	Mostrar las solicitudes ingresadas en el sistema
Actores	Solicitante: (Inicia) Listar Solicitud
Resumen	El CU se inicia cuando un usuario desea ver las solicitudes del sistema.
Complejidad	Alta
Prioridad	Alta
Precondiciones	N/A
Postcondiciones	N/A
Flujo de eventos	
Flujo básico "Listar Solicitud"	

	Actor	**Sistema**
1	El actor accede a ver el listado de las solicitudes mediante la opción "Lista de solicitudes".	2 Muestra una interfaz con el listado de solicitudes pendientes.

		3 Termina el caso de uso cuando se visualizó la lista de solicitudes.
Relaciones	**CU incluidos**	N/A
	CU extendidos	N/A
Requisitos no funcionales	Usabilidad, Funcionalidad, Mantenibilidad	
Asuntos pendientes	N/A	
Prototipo elemental de interfaz gráfica de usuario		



2.7 Conclusiones del capítulo

En el presente capítulo se confeccionó una propuesta de solución elaborando para ello un modelo conceptual. Además se definieron los requisitos funcionales y no funcionales que propiciaron las características que debe seguir el sistema, así como los casos de usos y sus respectivos diagramas. Lo anteriormente planteado permitió de forma considerable que se obtuviera un mejor funcionamiento del negocio.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

En el presente capítulo se describe la arquitectura que rige el desarrollo de la aplicación. También se hace alusión a los artefactos que se generan en el trascurso del capítulo y se exponen los patrones a utilizar en el diseño del sistema y se procede al diseño de la base de datos. Además se reflejan las pruebas realizadas al sistema que es una de las fases más importantes dentro de la implementación del mismo porque mide la calidad de las funcionalidades desarrolladas. Se aplicarán pruebas específicas para la validación de los artefactos, así como pruebas de caja negra y de caja blanca.

3.1 Arquitectura Cliente-Servidor

La arquitectura a utilizar es una arquitectura de computación en la que se consigue un procesamiento cooperativo de la información por medio de un conjunto de procesadores, de tal forma que uno o varios clientes, distribuidos geográficamente o no, solicitan servicios de computación a uno o más servidores. De esta forma, y gracias a esta arquitectura, la totalidad de los procesadores, clientes y servidores, trabajan de forma cooperativa para realizar un determinado tratamiento de la información. (22) Esta arquitectura posee varias ventajas entre las que se encuentran:

- Centralización del control de los recursos, datos y accesos
- Facilidad de mantenimiento y actualización del lado del servidor.
- Toda la información es almacenada en el lado del servidor, que suele tener mayor seguridad que los clientes. (23)

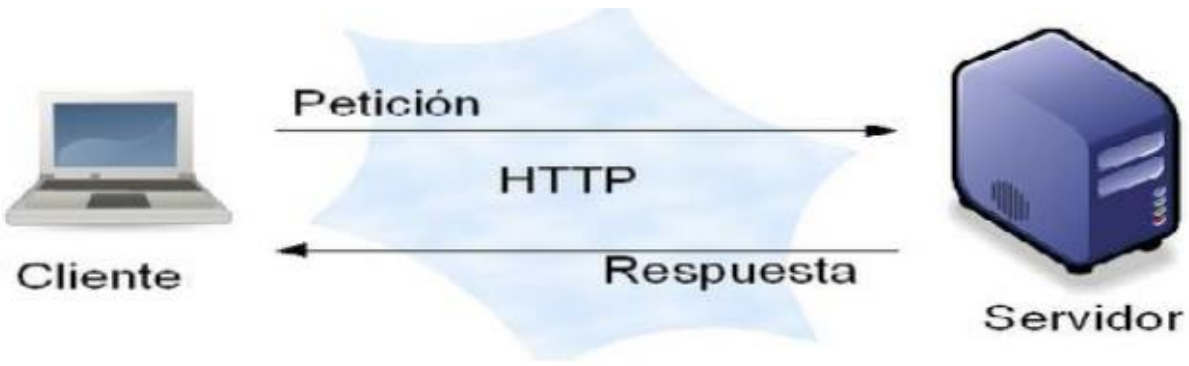


Figura 8. Arquitectura Cliente-Servidor

3.2 Patrón de Arquitectura.

Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software. (24) Por su parte el marco de trabajo empleado para el desarrollo de la aplicación Django utiliza el patrón de arquitectura Modelo-Plantilla-Vista por lo que se adoptó por el equipo de desarrolladores como patrón arquitectónico.

3.2.1 Patrón Arquitectónico Modelo-Plantilla-Vista (MTV).

El patrón arquitectónico Modelo-Plantilla-Vista es una modificación del patrón Modelo-Vista-Controlador y es utilizado por Django para convertirse en un framework más funcional. EL Modelo (M) manipula los datos de la aplicación, la Plantilla (T) decide cómo se van a mostrar los datos en el navegador y la Vista (V) decide cuáles datos va a mostrar el template. El mapeo de objeto relacional (ORM) de Django permite escribir código Python en lugar de SQL para hacer las consultas que necesita la vista. Para complementar, la plantilla: es básicamente una página HTML con algunas etiquetas extras propias de Django, conteniendo estructuras de datos necesarias para la presentación lógica de los datos y manteniendo la lógica del sistema en la vista. También permite crear contenido XML, CSS, JavaScript y CSV. (25)

Para una mejor comprensión de esta arquitectura observar la siguiente figura:

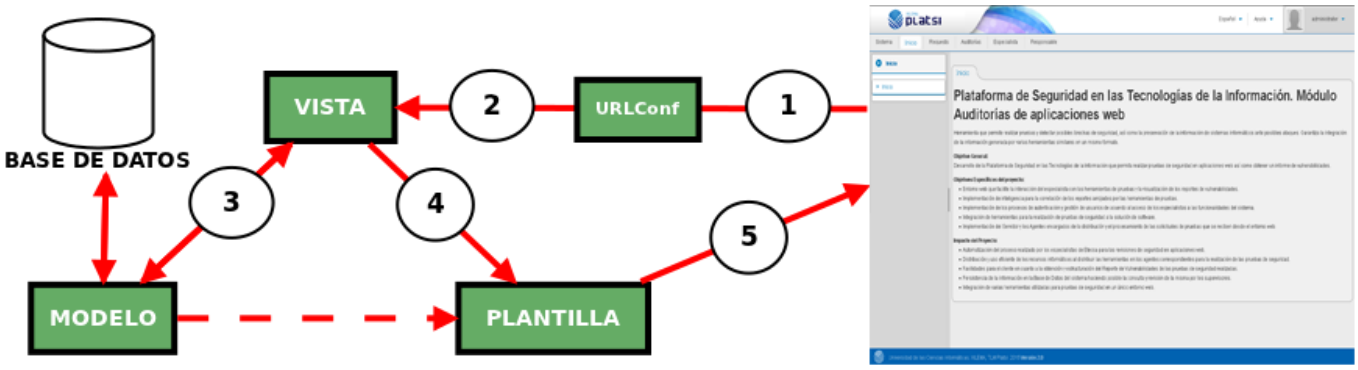


Figura 9. Patrón Arquitectónico Modelo-Plantilla-Vista

3.3 Patrones de Diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Se debe tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del

problema) y las consecuencias (costos y beneficios). Los patrones de diseño facilitan la reutilización de arquitecturas y diseños de software exitosos. (26)

3.3.1 Patrones para Asignar Responsabilidades (GRASP)

Los patrones GRASP representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es el acrónimo para *General Responsibility Assignment Software Patterns* (Patrones Generales de Software para Asignar Responsabilidades). A continuación, se presentan los patrones básicos de asignación de responsabilidades con una respectiva figura que evidencia la presencia de cada uno en el código del sistema.

Patrones básicos de asignación de responsabilidades:
Experto: Se encarga de asignar una responsabilidad al experto en información, o sea, aquella clase que cuenta con la información necesaria para cumplir la responsabilidad.

```
class Auditoria(models.Model):
    Nombre = models.CharField(max_length=150)
    FechaInicio = models.DateTimeField()
    FechaFin = models.DateTimeField()
    Responsable = models.ForeignKey(User)
    Especialista = models.CharField(max_length=150, null=True)
    DescripcionE = models.TextField(max_length=1000)
    Estado = models.CharField(max_length=50, default='Nueva')
    Descripcion = models.TextField(max_length=1000)
    AuditoriaWeb = models.ForeignKey(AuditoriaWeb, null=True)
    def __unicode__(self):
        return self.Nombre
|
```

Figura 10. Ejemplo del Patrón de diseño Experto

Creador: Este patrón es el responsable de asignarle a la clase B la responsabilidad de crear una instancia de clase A. B es un creador de los objetos A.

```
class EliminarAsignacionRestView(generics.RetrieveUpdateDestroyAPIView):

    def delete(self, request, *args, **kwargs):
        iD=str(kwargs['pk'])
        tem = Solicitud.objects.get(id=iD)
        auditoria = Auditoria.objects.get(id=tem.AuditoriaFK.id)

        auditoria.Especialista= None
        auditoria.DescripcionE= ''
        auditoria.save()
        return Response(request.DATA, status=status.HTTP_201_CREATED)
```

Figura 11. Ejemplo del Patrón de diseño Creador

Alta Cohesión: Asigna una responsabilidad de forma tal que la cohesión siga siendo alta.

```
class PlanificadorUpdateDeleteSerializer(serializers.ModelSerializer):
    Auditoria = serializers.Field(source='Auditoria.Auditoria')
    FechaInicio = serializers.Field(source='Auditoria.FechaInicio')
    FechaFin = serializers.Field(source='Auditoria.FechaFin')
    Responsable = serializers.Field(source='Auditoria.Responsable.username')
    Especialista = serializers.Field(source='Auditoria.Especialista')
    DescripcionAuditoria = serializers.Field(source='Auditoria.Descripcion')

    class Meta:
        model = Solicitud
        fields = ('id', 'Auditoria', 'FechaInicio', 'FechaFin', 'Responsable', 'Especialista', 'DescripcionAuditoria')


    def validate_Auditoria(self, attrs, source):
        value = (attrs[source])
        regex = r'[0-9A-Za-z ÑñáéíóúÁÉÍÓÚ]+'
        value = value.encode('utf8')
        if re.findall(regex, value) != [value]:
            raise serializers.ValidationError(_("Solo se admiten numeros y letras"))
        return attrs

    def validate_FechaFin(self, attrs, source):
        if attrs['FechaFin'] < attrs['FechaInicio'] :
            raise serializers.ValidationError(_("Debe ser mayor que la fecha de inicio"))
        return attrs
```

Figura 12. Ejemplo del Patrón de diseño Alta Cohesión

Bajo Acoplamiento: Este patrón es el encargado de asignar una responsabilidad para conservar bajo acoplamiento.

```
def delete(self, request, *args, **kwargs):
    iD=str(kwargs['pk'])
    tem = Solicitud.objects.get(id=iD)
    auditoria = Auditoria.objects.get(id=tem.AuditoriaFK.id)

    tem.AuditoriaFK=None
    tem.save()
    auditoria.delete()
    return Response(request.DATA, status=status.HTTP_201_CREATED)
```

Figura 13. Ejemplo del Patrón de diseño Bajo Acoplamiento

3.4 Diseño de la base de datos empleada

El modelo físico de la base de datos empleada se generó a partir de las clases modelos de Django y contiene las clases persistentes que implementan las entidades del problema.

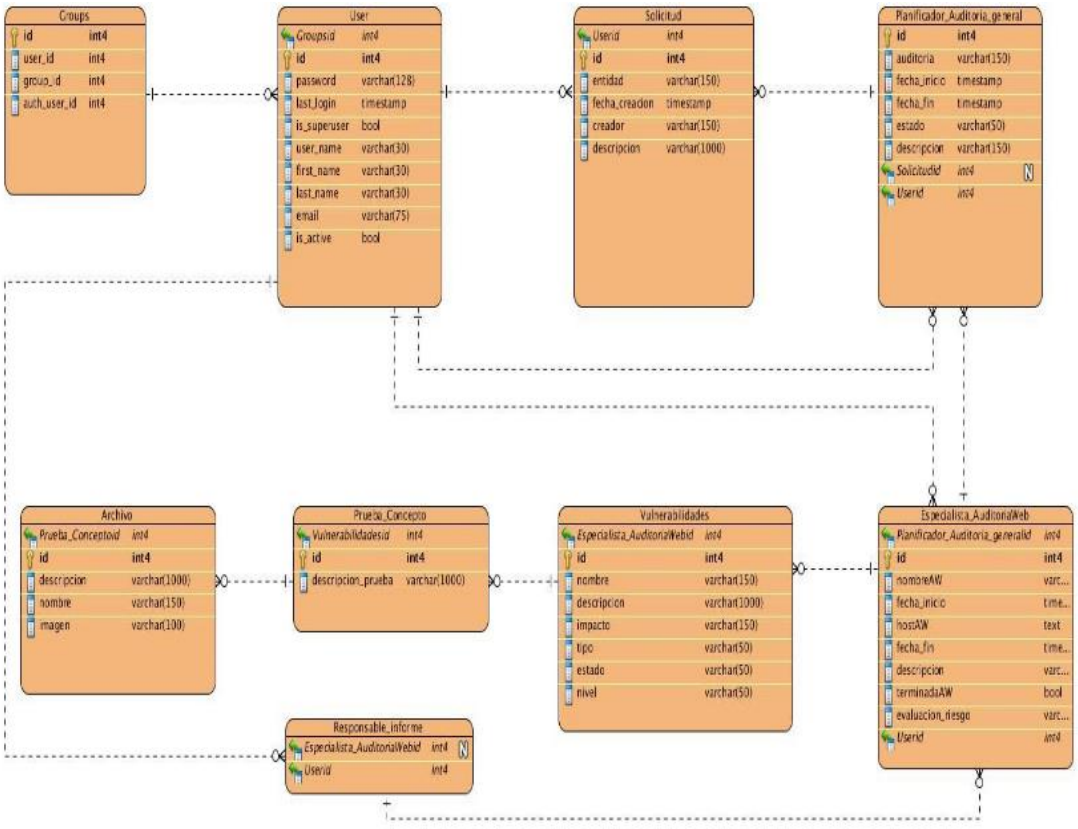


Figura 14. Tabla de diseño de la base de datos

Estas tablas están estructuradas de la siguiente manera:

Tabla 42: DB_Groups

Tabla de la base de datos		
Nombre: Groups		
Descripción: Guarda los grupos existentes en la base de datos.		
Atributo	**Tipo de dato**	**Descripción**
id	integer	Identificador del grupo (llave primaria)
user_id	integer	Identificador de la tabla user (llave foránea)
auth_user_id	integer	Identificador de la tabla auth_user (llave foránea)

Tabla 43: DB_User

Tabla de la base de datos		
Nombre: User		
Descripción: Guarda los usuarios existentes en la base de datos.		
Atributo	**Tipo de dato**	**Descripción**
id	integer	Identificador del usuario (llave primaria)
password	integer	Contraseña del usuario
last_login	integer	Fecha de la última vez que el usuario entró al sistema
Is_superuser	bool	Indica si el usuario es administrador o no
User_name	varchar	Usuario
First_name	varchar	Nombre del usuario
Last_name	varchar	Apellidos del usuario
email	varchar	Correo electrónico del usuario
Is_active	bool	Indica si el usuario estará activo o no

3.5 Diagrama de Despliegue

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos. (27)

A continuación, en la siguiente figura, se muestra el diagrama de despliegue donde se visualiza las relaciones entre los componentes del software en los nodos físicos.

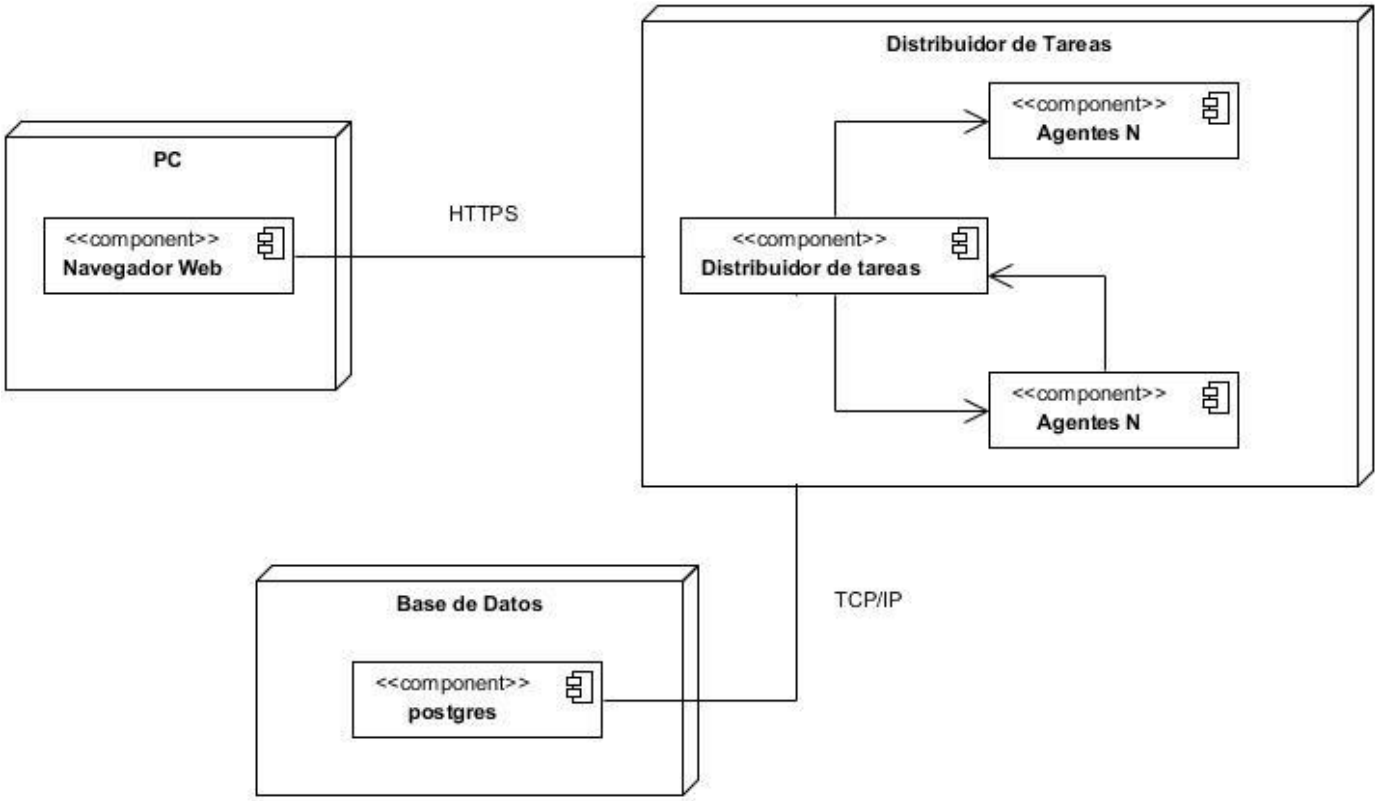


Figura 15. Diagrama de despliegue

3.6 Pruebas de software

Las pruebas son un elemento crítico para la calidad del software. La importancia de los costos asociados a los errores, promueve la definición y aplicación de un proceso de pruebas minuciosas y bien planificadas. Las pruebas permiten validar y verificar el software, entendiendo como validación del software el proceso, externo al equipo de desarrollo, que determina si el software satisface los requisitos, y verificación como el

proceso interno que determina si los productos de una fase satisfacen las condiciones de dicha fase. Este sistema, como cualquier otro en ingeniería, puede probarse de dos formas: a) conociendo la función específica para la que fue diseñado; y b) conociendo el funcionamiento del producto. El primer enfoque se centra en las llamadas pruebas de caja negra y el segundo en las pruebas de caja blanca. (28)

Elementos a tener en cuenta para que una prueba tenga éxito (29):

- Estrategia de prueba.
- Niveles de Prueba.
- Tipo de prueba.
- Método de prueba.
- Caso de prueba.

3.6.1 Estrategia de prueba

Las estrategias de pruebas de software proporcionan una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requerirán. Por tanto, cualquier estrategia de prueba debe incorporar la planificación, diseño, ejecución, recolección y evaluación de los casos de prueba (30).

Para realizar una correcta estrategia de pruebas se deben tener en cuenta algunos criterios de importancia (31):

- Describe el enfoque y los objetivos generales de las actividades de prueba.
- Incluye los niveles de prueba a ser diseccionados, el tipo de prueba a ser ejecutada y los casos de prueba diseñados para lograr los objetivos.
- Define:
  - ✓ Técnicas de pruebas (manual o automática) y herramientas a ser usadas.
  - ✓ Criterios de éxitos y culminación de las pruebas.
- ✓ Consideraciones especiales relacionadas con los recursos necesarios para realizar las pruebas.

Niveles de pruebas

La prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se distinguen los siguientes niveles de pruebas (31):

- Prueba de desarrollador.

- Prueba independiente.
- Prueba de unidad.
- Prueba de integración.
- Prueba de regresión.
- Prueba de sistema.
- Prueba de aceptación.

Se trabajará en los siguientes niveles:

- En el nivel de desarrollador haciendo uso del método de caja blanca para realizar pruebas al código implementado haciendo uso de la técnica de pruebas unitarias.
- Aplicando las pruebas de regresión para verificar que una vez que termine una iteración y se corrijan no conformidades los nuevos cambios no afecten la calidad de lo que se había hecho antes previniendo que se cometan nuevos errores.

Es válido aclarar que La metodología AUP-UCI propone 3 disciplinas de pruebas para aplicarle a los proyectos: pruebas internas, pruebas de liberación y pruebas de aceptación.

1- Pruebas internas: Se realizan en el centro y se verifica el resultado de la implementación probando cada construcción. Se desarrollan los casos de pruebas. Se realizaron pruebas de caja negra que permiten obtener conjuntos de condiciones de entrada que ejerciten todos los requisitos funcionales para encontrar varios tipos de errores (interfaz, rendimiento, estructura de datos).Se utilizó la técnica de la partición de equivalencia permite examinar los valores válidos y no válidos de las entradas existentes en el software.

2- Pruebas de liberación: Pruebas diseñadas y ejecutadas por una entidad certificadora externa antes de ser entregados al cliente para su aceptación. Se realizaron pruebas de caja negra que verifican que el sistema funcione apropiadamente y sin errores.

3- Pruebas de aceptación: Se verifica que el software está listo y que puede ser usado por usuarios finales. Se hizo entrega del módulo al cliente, y este realizo un acta de aceptación planteando haberse cumplido el objetivo del mismo.

3.6.2 Tipos de pruebas

Un grupo de actividades de pruebas pueden tener por objetivo verificar el sistema de software (o parte el sistema) en base a un motivo u objetivo específico para probar. Un

tipo de prueba se centra en un objetivo de prueba en particular, que puede ser cualquiera de los siguientes (32):

- Una función a realizar por el software
- Una característica de calidad no funcional, tales como la fiabilidad o la usabilidad
- La estructura o arquitectura del software o sistema
- Cambios asociados, es decir, confirmar que se han solucionado los defectos (pruebas de confirmación) y localizar cambios no intencionados (pruebas de regresión)

Puede desarrollarse y/o utilizarse un modelo del software en las pruebas estructurales (por ejemplo, un modelo de flujo de control o un modelo de estructura de menús), en las pruebas no funcionales (por ejemplo, un modelo de rendimiento, un modelo de amenaza de seguridad y un modelo de usabilidad), y en las pruebas funcionales (por ejemplo, un modelo de flujo de procesos, un modelo de transición de estados o una mera especificación de lenguaje).

✓ **Pruebas funcionales**

Las funciones que un sistema, subsistema o componente deben llevar a cabo pueden describirse en productos de trabajo tales como una especificación de requisitos, casos de uso o una especificación funcional, o incluso pueden no estar documentadas. Las funciones son “lo que” el sistema hace.

Las pruebas funcionales se basan en funciones y prestaciones (descritas en documentos o entendidas por los probadores) y en su interoperabilidad con sistemas específicos, y pueden llevarse a cabo en todos los niveles de pruebas.

✓ **Pruebas no funcionales**

Las pruebas no funcionales incluyen, pero sin limitarse a ello, pruebas de rendimiento, pruebas de carga, pruebas de estrés, pruebas de usabilidad, pruebas de mantenibilidad, pruebas de fiabilidad y pruebas de portabilidad. Estas pruebas se refieren a “cómo” funciona el sistema. Las pruebas no funcionales pueden ejecutarse a todos los niveles de prueba. El término pruebas no funcionales hace referencia a las pruebas necesarias para medir las características de los sistemas y software que pueden cuantificarse según una escala variable, tales como los tiempos de respuesta en el caso de las pruebas de rendimiento.

✓ **Pruebas estructurales**

Las pruebas estructurales (caja blanca) pueden realizarse en todos los niveles de prueba. Las técnicas estructurales son más idóneas de utilizar después de las técnicas basadas

en la especificación, para ayudar a medir la exhaustividad de las pruebas mediante una evaluación de la cobertura de un tipo de estructura. La cobertura es la medida en que un juego de pruebas ha probado una estructura, expresada como porcentaje de los elementos cubiertos. Si la cobertura no es del 100%, entonces podrán diseñarse más pruebas para probar los elementos que faltan para aumentar la cobertura.

- ✓ **Pruebas de regresión**

  Una vez detectado y corregido un defecto, el software debe volver a probarse para confirmar que el defecto original ha sido corregido con éxito. A esto se le denomina confirmación. La depuración (localizar y corregir defectos) es una actividad de desarrollo, no una actividad de pruebas. Las pruebas de regresión son pruebas reiteradas de un programa ya probado, después de haber sido modificado, con vistas a localizar defectos surgidos o no descubiertos como resultado del cambio o de los cambios. Estos defectos pueden estar en el software objeto de las pruebas, o en cualquier otro componente de software asociado o no asociado. Se realizan cuando el software, o su entorno, sufren modificaciones. El alcance de las pruebas de regresión depende del riesgo de no encontrar defectos en el software que antes funcionaba.

  El tipo de prueba a aplicar es funcional. Estas pruebas fijan su atención en la validación de las funciones, métodos, servicios y casos de uso. Permiten comprobar el correcto funcionamiento de los requisitos funcionales del módulo.

3.6.3 Métodos de prueba

Los métodos de prueba definen la estrategia a seguir en función de la verificación y validación del sistema diseñado para descubrir fallos. Los métodos estudiados fueron las pruebas de caja blanca y las pruebas de caja negra.

3.6.3.1 Pruebas de caja blanca

Las pruebas de caja blanca del software se basan en el examen cercano de los detalles de procedimiento. Las rutas lógicas a través del software y las colaboraciones entre componentes se ponen a prueba al revisar conjuntos específicos de condiciones y/o bucles. (33)Estas pruebas van dirigidas explícitamente a la estructura lógica interna del software. El tipo de prueba de caja blanca a ejecutar son las pruebas unitarias.

.

Pruebas Unitarias

Este tipo de pruebas son ejecutadas normalmente por el equipo de desarrollo, básicamente consisten en la ejecución de actividades que le permitan verificar al desarrollador que los componentes unitarios están codificados bajo condiciones de robustez, esto es, soportando el ingreso de datos erróneos o inesperados y demostrando así la capacidad de tratar errores de manera controlada. Adicionalmente, Las pruebas sobre componentes unitarios, suelen denominarse pruebas de módulos o pruebas de clases, siendo la convención definida por el lenguaje de programación la que influye en el término a utilizar. (34)Se realizaron pruebas en pyunit, que es parte de la biblioteca estándar de Python a través de su módulo unitest. La función de estos marcos de trabajo es buscar y ejecutar las pruebas definidas en el fichero que se le pasan como argumento. Finalmente devolverá un informe sobre el estado de las pruebas y los errores que se ha ido encontrando.

Para realizar estas pruebas a la aplicación confeccionada se realizaron 3 iteraciones para 19 fragmentos de códigos donde se detectaron 10 errores que posteriormente son corregidos de manera satisfactoria. A continuación se muestran los resultados obtenidos de dichas iteraciones.

Iteración 1: Se seleccionaron 4 fragmentos de código en los cuales se detectaron 3 errores de lógica algorítmica que fueron corregidos de manera satisfactoria.

Figura 16. Prueba unitaria de la iteración 1

Iteración 2: Se seleccionaron 6 fragmentos de código en los cuales se detectaron 4 errores de lógica algorítmica que fueron corregidos de manera satisfactoria.

Figura 17. Prueba unitaria de la iteración 2

Iteración 3: Se seleccionaron 9 fragmentos de código en los cuales se detectaron 3 errores de lógica algorítmica que fueron corregidos de manera satisfactoria.

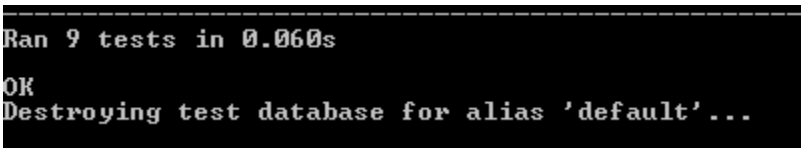


Figura 18. Prueba unitaria de la iteración 3

3.6.3.2 Pruebas de caja negra

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software. Se trata de demostrar que las funciones del software son operativas, que las entradas se manejan de forma adecuada y que se produce el resultado esperado. (35)Las pruebas de caja negras son pruebas dirigidas en particular al cliente, en las cuales se mide su nivel de satisfacción. El tipo de prueba de caja negra ejecutar son las pruebas de aceptación.
Pruebas de Aceptación

Independientemente de que se haya tercerizado el proceso de pruebas y así la firma responsable de estas actividades haya emitido un certificado de calidad sobre el sistema objeto de prueba, es indispensable, que el cliente designe a personal que haga parte de los procesos de negocio para la ejecución de pruebas de aceptación, es incluso recomendable, que los usuarios finales que participen en este proceso, sean independientes al personal que apoyó el proceso de desarrollo. Cuando las pruebas de aceptación son ejecutadas en instalaciones o ambientes proporcionados por la firma desarrolladora se les denominan pruebas Alpha, cuando son ejecutadas desde la infraestructura del cliente se les denomina pruebas Beta. (36)

A continuación se muestra cómo se precisaron las pruebas de aceptación, las cuáles se encuentran separadas en casos de pruebas:

Tabla 51 .Caso de Prueba Crear Grupo de Usuario.

Escenario	Descripción	Nombre	Permisos	Respuesta del sistema	Flujo central
		V	V		

EC 1. Crear Grupo de usuario.	Se crea un nuevo grupo de usuarios.	Solicitante	Planificador	Se muestra un mensaje "El grupo ha sido creado"	1. El usuario se encuentra en la página principal y selecciona la opción "Administración". 2. Desplegar la opción "Grupo" en el menú principal de la página de administración. 3. Seleccionar la opción "Nuevo grupo". 4. Se introducen los datos del grupo y se presiona el botón "Aceptar".
EC 1.1 Campos vacíos.	Se valida que no queden campos vacíos.	I	N/A	Se muestra un mensaje "Rellene este campo".	
EC 1.2 Campos incorrectos.	Se validan los campos incorrectos y nombres repetidos.	I 123	V Solicitante	Se muestra un mensaje "Solo se aceptan letras y espacios en blanco".	
		V Solicitante	V Planificador	Se muestra un mensaje "Este valor ya se ha utilizado."	

Tabla 52 .Caso de Prueba Modificar Grupo de Usuario

Escenario	**Descripción**	**Nombre**	**Permisos**	**Respuesta del sistema**	**Flujo central**
EC 1. Modificar grupo de usuarios.	Se modifica los datos del grupo de usuarios deseado.	V Solicitante	V Solicitante	Se muestra un mensaje "El grupo ha sido actualizado".	1. El usuario se encuentra en la página principal y selecciona la opción "Administración". 2. En "Grupos" dar doble clic en "Listar grupos".
EC 1.1 Campos vacíos.	Se valida que no queden campos vacíos.	I	N/A	Se muestra el siguiente mensaje en el campo vacío "Rellene este campo"	
		I	V		

EC 1.2 Campos incorrectos.	Se validan los campos incorrectos.	123	Administrador	Se muestra el mensaje ''Solo se aceptan letras y espacios en blanco''.	3. Seleccionar la opción "Modificar". 4. Se modifican los datos y se presione el botón ''Aceptar''. ---------- **Nota**: El usuario puede dar aceptar sin haber realizado ningún cambio.
	Se validan los campos incorrectos.	I a	V Administrador	Se muestra el mensaje ''Ajústese al formato solicitado''.	

Los restantes casos de prueba se encuentran en el Anexo II.

A realizar las pruebas al sistema a partir de los casos de pruebas se realizaron 4 iteraciones para 11 pruebas de aceptación donde se obtuvieron un total de 5 no conformidades que posteriormente fueron resueltas en su totalidad. Para la primera iteración se realizaron 4 pruebas de aceptación y se detectaron 2 no conformidades. En la segunda iteración se realizaron 3 pruebas de aceptación y fueron detectadas 2 no conformidades. En la tercera iteración se realizaron 2 pruebas de aceptación y fueron detectadas 1 no conformidades y en la cuarta iteración se realizaron 2 pruebas de aceptación y no se encontraron no conformidades.

A continuación en la figura 20 se muestra los resultados obtenidos de cada iteración.

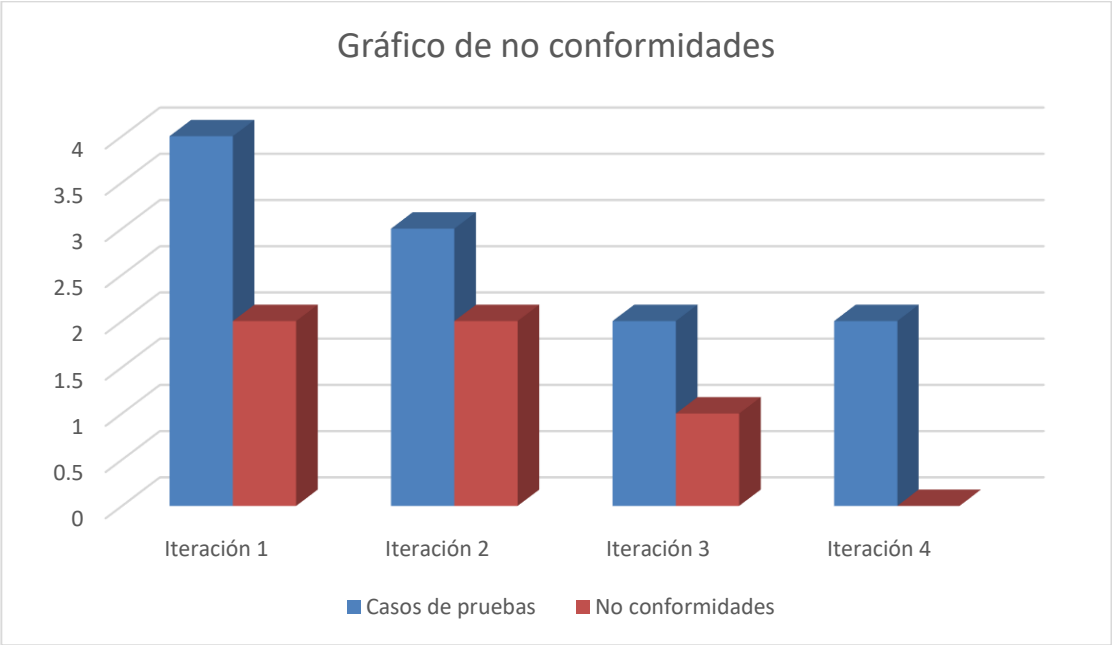


Figura 19. Gráfica de comportamiento de no conformidades

3.7 Conclusiones del capítulo.

En este capítulo se definieron la arquitectura, los patrones de diseño y arquitectónico a utilizar, lo cual contribuyó a facilitar el adecuado desarrollo del software. Se generó el modelo físico de la base de datos para identificar las clases relevantes lo que propició una mejor organización para la realización del sistema. Además se detalló el diagrama de despliegue que permitió un mejor entendimiento de las relaciones físicas entre los componentes software y hardware. Luego de realizar las pruebas de caja blanca y de caja negra al sistema, se llegó a la conclusión de que se detectaron un total de no conformidades a las cuales luego de varias iteraciones se le dieron solución. De esta forma se evaluaron las funcionalidades del sistema y se concluye que el mismo cumple satisfactoriamente con los estándares requeridos.

CONCLUSIONES

Durante el desarrollo del presente trabajo se realizó la migración de la Plataforma de Seguridad de las Tecnologías y la Información (PlatSI) a Xilema-Base-Web cumpliéndose de esta manera con el objetivo planteado en esta investigación. Las herramientas con las que se conformó PlatSI no regían dentro de los estándares que sigue el centro de Telemática (TLM) por lo que la migración realizada constituye un paso positivo para dicho centro. La nueva plataforma mantiene las mismas funcionalidades que la aplicación anterior permitiendo de esta forma la fácil interacción de especialistas y miembros del centro con dicha plataforma.

RECOMENDACIONES

Partiendo de los conocimientos adquiridos durante la investigación del trabajo y con el objetivo de contribuir a un mayor desarrollo del mismo se propone la siguiente recomendación:

- Que se realicen nuevas investigaciones de las herramientas de auditoría web integradas a la aplicación en pos de acuatizarlas a versiones más recientes de las mismas.

REFERENCIAS BIBLIOGRÁFICAS

1. [En línea] 3 de Febrero de 2012. [Citado el: 13 de Noviembre de 2017.] http://www.cubahora.cu/articulos-de-opinion/40-razones-para-desmentir-que-cuba-sea-enemiga-de-internet.

2. Rodriguez, Omar. Seguridad Informática. [En línea] 9 de 2015. [Citado el: 12 de 2 de 2018.] http://www.backtrackomar.com/2015/09/auditando-con-nikto-2.html.

3. north networks. [En línea] 2003. [Citado el: 12 de 2 de 2018.] http://www.north-networks.com/acunetix/.

4. Velasco, Rubén. redeszone. [En línea] 25 de 4 de 2015. //www.redeszone.net/2015/04/25/seguridad-web-owasp-zap/.

5. Auditoria Informática y Hacking Ético. [En línea] 14 de 9 de 2017. //auditoriainformaticahackingeticoypruebasdepenetracion.wordpress.com.

6. Culoccioni, Sergio. solvetic. [En línea] 12 de 3 de 2016. //www.solvetic.com/tutoriales/article/2453-auditar-y-escanear-seguridad-web-con-arachni/.

7. Odm. [En línea] 28 de Marzo de 2016. [Citado el: 5 de Julio de 2018.] http://www.odm.es/en/noticia.asp?p=la-migracion-tecnologica-clave-para-la-productividad-de-las-empresas.

8. Jimenez, Jesus. [En línea] [Citado el: 5 de Junio de 2018.] https://es.slideshare.net/JesusJimenez123/migracion-de-sistemas-computacionales.

9. python. [En línea] 2001-2018. [Citado el: 12 de Diciembre de 2017.] https://www.python.org/doc/essays/blurb/.

10. github. [En línea] 2015. [Citado el: 24 de Diciembre de 2017.] http://github.com/saulgm/djangobook.com.

11. recursos python. [En línea] 2013. [Citado el: 12 de 2 de 2018.] recursospython.com.

12. Clemente, Celia y Fajardo, Lara. it.uc3m. [En línea] 2007. [Citado el: 25 de Diciembre de 2017.] http://www.it.uc3m.es/spickin/docencia/comsoft/presentations/spanish/doc/Python.pdf.

13. Montero, Sergio Infante. Maestros del Web. [En línea] 16 de 4 de 2012. [Citado el: 12 de 2 de 2018.] http://www.maestrosdelweb.com/curso-django-introduccion/.

14. Alegsa, Leandro. Alegsa. [En línea] 31 de 7 de 2015. [Citado el: 12 de 2 de 2018.] http://www.alegsa.com.ar/Dic/html5.php.

15. postgresqltutorial. [En línea] 2000. [Citado el: 25 de Diciembre de 2017.] http://www.postgresqltutorial.com/what.

16. jetbrains. [En línea] 2000. [Citado el: 25 de Diciembre de 2017.] https://www.jetbrains.com/pycharm/features/.

17. visualparadigm. [En línea] 16 de Agosto de 2010. [Citado el: 25 de Diciembre de 2017.] https://www.visualparadigm.com/aboutus/newsreleases/vpuml80.jsp.

18. Sommerville, Ian. *Software Engineering.* Madrid : Pearson Educación S, 2002. ISBN.

19. S, Pearson Educación. *Somerville, Ian.* Madrid : Pearson Educación S, 2005. ISBN.

20. Somerville, Ian. *Software Engineering.* Madrid : Pearson Educación S, 2005. ISBN.

21. García, Gabriel. mat.uson. [En línea] Diciembre de 2004. [Citado el: 12 de Febrero de 2018.] http://www.mat.uson.mx/mireles/Casos%20de%20usonota.htm#_Toc64166509.

22. DANILO, GONZALEZ REYES. Monografías. [En línea] [Citado el: 18 de abril de 2018.] http://www.monografias.com/docs114/telecomunicaciones-arquitectura-cliente-servidor/telecomunicaciones-arquitectura-cliente-servidor.shtml.

23. alegsa. [En línea] 12 de Noviembre de 2010. [Citado el: 18 de Abril de 2018.] http://www.alegsa.com.ar/Respuesta/ventajas_y_desventajas_del_modelo_clienteservidor.htm.

24. cgr. [En línea] 2006. [Citado el: 23 de Marzo de 2018.] https://cgrw01.cgr.go.cr/rup/RUP.es/SmallProjects/core.base_rup/guidances/termdefinitions/architectural_pattern_E2E8EB79.html.

25. Librosweb. [En línea] 2007. [Citado el: 20 de abril de 2018.] http://librosweb.es/libro/django_1_0/capitulo_5/el_patron_de_diseno_mtv.html.

26. Wordpress. [En línea] 27 de Agosto de 2012. [Citado el: 20 de abril de 2018.] https://arlethparedes.wordpress.com/2012/08/27/patrones-de-arquitectura-vs-patrones-de-diseno/.

27. sparx systems. [En línea] [Citado el: 12 de Mayo de 2018.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.

28. Pressman, Roger. *Ingeniería de Software.Un efoque práctico.* México : Mc Grawn Hill, 2005.

29. LARA, M. *Pruebas de software .* 2013. págs. 1-34.

30. PRESSMAN, R.S. *Ingeniería del software. Un enfoque práctico.* 2010.

31. RUIZ, B y VAILLANT, C. *Módulo Recomendaciones del sistema para repositorios digitales REPXOS 3.0.* Universidad de las Ciencias Informáticas. 2015.

32. ISTQB, colectivo de autores. *International Software Testing Qualification Boards Probador Certificado Programa de estudio de nivel básico.* 2011.

33. Aniel. Aniel. [En línea] 10 de 12 de 2017. http://www.aniel.es/page/5/.

34. UCI. Universidad de las Ciencias informàticas. [En línea] 2018. http://www.uci.cu.

35. Cuba, Partido Comunista de. *Partido Comunista de Cuba.* República de Cuba : s.n., 2011.

36. [En línea] [Citado el: 12 de 2 de 2018.]
http://www.sincows.com/sincows/index.php?option=com_content&view=article&id=70&Itemid=68.
37. asfsoluciones. [En línea] 2013. [Citado el: 12 de 2 de 2018.]
http://asfsoluciones.com/index.php/lineas-de-negocio/consultoria-y-outsourcing/mantenimiento-software-y-hardware.
38. Esteban, Ney Toribio. scribd. [En línea] 11 de 2017. [Citado el: 14 de 2 de 2018.]
https://es.scribd.com/document/366091694/Proyecto-Taller-II.
39. pmoinformatica. [En línea] [Citado el: 13 de Mayo de 2018.]
http://www.pmoinformatica.com/p/pruebas-de-software.html.

BIBLIOGRAFÍA

1.Cuba., Partido Comunista de. Partido Comunista de Cuba. . (República de Cuba, 2011).

2.SENA.[En línea] 2018. https://senaintro.blackboard.com/bbcswebdav/institution/semillas/228106_2_VIRTUAL-2015/contenido/oaaps/oaap10/aa1/oa_migracion/utilidades/descargable.pdf).

3.Senaintro. [En línea] 2017. https://senaintro.blackboard.com/bbcswebdav/institution/semillas/228106_2_VIRTUAL-2015/contenido/oaaps/oaap10/aa1/oa_migracion/utilidades/descargable.pdf.

4.Pressman, Roger S.Capítulo 8 Modelado de análisis. Ingeniería de software. Un enfoque práctico. 2005.

5.Buzz, S. Arquitectura de software. [En línea] 2009. http://sq.com.mx/content/view/922.

6.Sommerville, Ian. Ingeniería de software. 2005. ISBN: 8478290745.

7.Hipertextual. [En línea] 9 de Diciembre de 2014. http://hipertextual.com/archivo/2014/06/pycharm-ide-python/.

8.sincows.[En línea] http://www.sincows.com/sincows/index.php?option=com_content&view=article&id=70&Itemid=68.

9. python. [En línea] https://www.python.org/.

10.mozilla. [En línea] https://developer.mozilla.org/.

11.alegsa. [En línea] 2018. http://www.alegsa.com.ar/Dic/html5.php.

12.postgresql. [En línea] 2018. https://www.postgresql.org/about/.

13.UML Modeling -Unified Modeling Language Tool, 2010. VP Gallery[online].[Accessed 26February 2018]. Available from: https://www.visual-paradigm.com/VPGallery/diagrams/index.html.

14.Visual Paradigm 8.0 (formerly VP-UML 8.0) Released, 2010. Visual Paradigm[online]. [Accessed 15 December 2017]. Available from: https://www.visualparadigm.com/aboutus/newsreleases/vpuml80.jsp

16.SOMMERVILLE, Ian, 2005. Software Engineering. 7th edition. Madrid: Pearson Educación S.A.ISBN 84-7829-074-5.

17.TEDESCHI, Nicolás, 2018. ¿Qué es un Patrón de Diseño? [online]. 2018. [Accessed8April2018]. Available from: https://msdn.microsoft.com/es-es/library/bb972240.aspx

18.The Agile Unified Process (AUP)Home Page, [no date]. [online]. [Accessed 22 December 2017]. Available from: http://www.ambysoft.com/unifiedprocess/agileUP.html

19.What is Python? Executive Summary, 2018. Python.org[online]. [Accessed 22 December 2017]. Available from: https://www.python.org/doc/essays/blurb/

20.IAN, Sommerville. 2006. Ingeniería del software. México DF: Pearson. 2005, 2006.

21.Mediavilla, Elena. 2010. Programacion orientada a objetos. Máster de computación.Modelado de casos de uso. 2010.

22.PRESSMAN, Roger S. 2005. Ingeniería de Software. Un enfoque práctico. sexta. Mc Graw Hill,. México

23.Paradigm, Visual. 2013. Visual paradigm for uml. Visual Paradigm for UML-UML tool for software application development. 2013.

24. ALEGSA. [En línea] [Citado el: 25 de Abril de 2015.] http://www.alegsa.com.ar/Respuesta/Ventajas y desventajas del modelo clienteservidor.html.

25.Desarrolloweb. [En línea] [Citado el: 10 de Diciembre de 2014.]http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

26. Ingeniería de software. [En línea] Universidad Unión Boliviana, 13 de Febrero de 2015. http://Ingenieriadesoftware.mex.tt/.

27.PgAdmin: PostgreSQL Tools. [En línea] [Citado el: 24 de Abril de 2015.] http://www.pgadmin.org/.

28. Woods, Nick Rozanski y Eoin. Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives. s.l.: Addison-Wesley Professional, 2005

29- PostgreSQL: Advantages, [no date]. [onl

ine]. [Accessed 15 Febrero 2018]. Available from: https://www.postgresql.org/about/advantages

30.PRESSMAN, Roger S, 2010. Software Engineering. A practitioner´s approach. 7th edition.

31.SÁNCHEZ, Tamara Rodríguez, 2014. Metodología de desarrollo para la actividad productiva de la UCI. 21 November 2014.

32.SOMMERVILLE, Ian, 2005. Software Engineering. 7th edition. Madrid: Pearson Educación S.A. ISB N84-7829-074-5.

33.The Agile Unified Process (AUP)Home Page, [no date]. [online]. [Accessed 22 Febrero 2018]. Available from: http://www.ambysoft.com/unifiedprocess/agileUP.html.

34.Lamarca, María Jesú. 2013. Hipertexto. [En línea] 2013. http://www.hipertexto.info/documentos/uml.htm.

35.Maida, Esteban Gabriel y Pacienzia, Julián. 2015. Metodologías de desarrollo de software. Argentina : Biblioteca digital de la universidad católica de Argentina, 2015.