UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

Plataforma de Seguridad en las Tecnologías de la Información v2.0. Módulo de gestión de auditoría web.

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

Autores: Daniela González Begué

Hector Vladimir Potete Muñoz

Tutores: Ing. Yeilenia Iris Pérez Vázquez

Ing. Lianet Salazar Labrada

"La Habana, junio, 2015"

La modernidad tecnológica implica una economía en la cual el progreso tecnológico sostenible es el principal motor del crecimiento, y de eso depende la persistencia del progreso tecnológico. Lo que se necesita es una buena teoría que dé cuenta de la clase de factores que hacen sostenible el progreso tecnológico.

Joel Mokyr

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de este trabajo y autorizamos al Centro de Telemática de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Daniela González Begué

_____

Nombre de autor

Hector Vladimir Potete Muñoz

_____

Nombre de autor

Ing. Yeilenia Iris Pérez Vázquez

_____

Nombre del tutor

Ing. Lianet Salazar Labrada

_____

Nombre del tutor

DATOS DE CONTACTO

Datos del Autor:

Daniela González Begué

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: dbegue@estudiantes.uci.cu

Datos del Autor:

Hector Vladimir Potete Muñoz

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: hvpotete@estudiantes.uci.cu

Datos del Tutor:

Ing. Yeilenia Iris Pérez Vázquez, graduado de Ingeniero en Ciencias Informáticas.

Pertenece al área de TLM, Dpto. Aplicaciones.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: yiperez@uci.cu

Datos del Tutor:

Ing. Lianet Salazar Labrada, graduado de Ingeniero en Ciencias Informáticas.

Pertenece al área de TLM, Dpto. Aplicaciones.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: lslabrada@uci.cu

DEDICATORIA

A la UCI por permitirnos llegar a este momento especial de nuestras vidas. Por los triunfos y los momento difíciles que nos han enseñado a valorarlos cada día más. A nuestros padres por su amor, trabajo y sacrificio en todos estos años. A nuestros profesores gracias por su tiempo, por su apoyo, así como por la sabiduría que nos transmitieron en el desarrollo de nuestra formación profesional. Ellos son parte de este logro, ya que ustedes lo trabajaron y espero que su esfuerzo y empeño sea reflejado en esta tesis. A nuestros amigos que gracias al equipo que formamos logramos llegar hasta el final del camino y hasta el momento seguimos siendo amigos. Sin nada más que decir esta dedicación de tesis va directamente para ustedes compañeros, muchas gracias por todo.

Daniela y Hector

RESUMEN

El ciclo de vida de un software está compuesto por diferentes etapas. La etapa de mantenimiento es la que surge luego de la entrega del producto al cliente. En esta se corrigen errores en la programación o se agregan nuevas funcionalidades, se mejora el rendimiento o se adapta la solución a un cambio en el entorno. El Centro de Telemática (TLM) de la Universidad de las Ciencias Informáticas (UCI) cuenta con un producto de software llamado Xilema – PlatSI o Plataforma de Seguridad en las Tecnologías de la Información. Esta solución se encarga de realizar pruebas a aplicaciones web en busca de vulnerabilidades, ejecutando herramientas de auditorías a aplicaciones web (Nikto y Acunetix).

La aplicación web fue desarrollada con PHP como lenguaje de programación y Symfony2 como marco de trabajo asociado a este lenguaje. Actualmente en TLM ya no se desarrolla en este lenguaje, debido a un proceso de estandarización informática que en este centro se lleva a cabo. Este proceso define Python como lenguaje de programación y Django como marco para trabajar para este lenguaje.

El objetivo que persigue este trabajo es el de adaptar el Módulo de Gestión de Auditoría Web de Xilema – PlatSI al nuevo lenguaje de programación y marco de trabajo con que se desarrolla en TLM actualmente. Para el desarrollo del mismo se analiza los tipos de mantenimientos aplicables a un software. Se utiliza los métodos de la investigación teóricos (análisis – sintético y modelación) y empíricos (entrevista y observación), además de la metodología ágil de desarrollo Programación extrema (XP).

PALABRAS CLAVES

Mantenimiento adaptativo, mantenimiento de software, Xilema – PlatSI

TABLA DE CONTENIDOS

INTRODUCCIÓN

El ciclo de vida de un software está compuesto por diferentes etapas, dentro de las cuales se encuentran la exploración, planificación de la entrega, iteraciones, producción y mantenimiento (1). La etapa de mantenimiento, surge luego de la entrega del producto al cliente. En esta etapa se corrigen errores en la programación o se agregan nuevas funcionalidades, se mejora el rendimiento o se adapta la solución a un cambio en el entorno.

Al añadirle a un sistema ya existente una nueva funcionalidad, el mantenimiento repite "en miniatura" el ciclo de vida completo de un software y añade una nueva tarea, la de comprender dicho sistema, por lo que se podría decir que el mantenimiento de un sistema es más difícil que su propio desarrollo (2). La importancia de esta etapa del desarrollo del software ha sido reconocida en los últimos años por la comunidad científica y empresarial a nivel internacional, debido a su impacto económico en la vida de las organizaciones y del país.

Cuba, a pesar del bloqueo económico, comercial y financiero impuesto por Estados Unidos, el cual le impide a la isla el acceso a su mercado, ha potenciado la industria del software. Esta hace uso de la etapa de mantenimiento en el desarrollo de sus soluciones informáticas, para evitar aplicar productos incompatibles, obsoletos e insostenibles que no permitan un desarrollo armónico e integrado de la sociedad pudiendo propiciar ineficacia en los procesos (3). La Universidad de las Ciencias Informáticas (UCI) es una de las organizaciones cubanas dedicadas al desarrollo de software y mantenimiento de los mismos. Esta cuenta con una red de centros productivos que apoyan la informatización de la sociedad cubana.

Uno de estos centros es el de Telemática (TLM), especializado en la seguridad de las redes y sistemas informáticos, en el cual se desarrolló un producto de software que valida la seguridad en las tecnologías de la información: Xilema – PlatSI o Plataforma de Seguridad en las Tecnologías de la Información. Esta solución se encarga de integrar herramientas de seguridad para detectar vulnerabilidades en aplicaciones web. Dicho sistema cuenta con un flujo de trabajo que permite la gestión de auditorías, ejecución de pruebas de seguridad, análisis, correlación de resultados y la conformación de un informe final con las evidencias encontradas y el nivel de riesgo asociados a las mismas.

La plataforma posee un estado funcional, probado y liberado por el departamento de calidad en la UCI (Ver Anexo I), además posee un módulo dedicado a la realización de auditorías a las aplicaciones web así como un conjunto de funcionalidades que gestionan la información general que es común a cualquier tipo de auditoría. Además estás compuesto por un distribuidor de tareas

(Bambú), una base de datos (PostgreSQL) y una aplicación web implementada en PHP como lenguaje de programación y Symfony2 como *framework* asociado a este lenguaje.

Actualmente del proyecto Xilema – PlatSI quedan muy pocos integrantes y los nuevos que se incorporan no tienen un vasto conocimiento y práctica en la tecnología sobre la que esta implementado el sistema, ni del funcionamiento del mismo, por lo que capacitarlos en esta requeriría mucho tiempo y recurso.

Además en el centro de Telemática se lleva a cabo un proceso de estandarización informática que define a Python como lenguaje de programación y Django como *framework* de desarrollo. Para este proceso se impartieron cursos de capacitación a estudiantes y profesores, y se creó además el *framework* Xilema – Base – Web que define la arquitectura del centro.

Debido a que Xilema - PlatSI no cumple con las pautas de diseño ha sido difícil aumentar el personal, debido a que la capacitación recibida por los nuevos graduados está enfocada a la estandarización y no al viejo marco de trabajo en que se desarrolló el producto. Además, TLM en busca de nuevas oportunidades de negocio, pretende ofertar productos que identifiquen a la UCI, así como poder dar soporte y mantenimiento a todos los productos que aquí se implementan, por lo que ha surgido la necesidad de adaptar la aplicación web a las nuevas condiciones.

Por lo expuesto anteriormente se plantea como **problema de la investigación:** ¿Cómo contribuir a la adopción de las nuevas pautas de diseño por Xilema - PlatSI en el Centro Telemática?

Este problema condujo a que el **objeto de estudio** de la presente investigación este enfocado en el: Mantenimiento de productos de software.

Se define como **campo de acción**: Módulo de Gestión de Auditoría Web de Xilema - PlatSI del Centro Telemática.

Para dar solución al problema planteado anteriormente se propone como **objetivo general:** Adaptar el Módulo de Gestión de Auditoría Web de Xilema - PlatSI de que forma cumpla con las nuevas pautas de diseño del Centro de Telemática.

Este objetivo general se desglosa en los siguientes **objetivos específicos:**

- ✓ Fundamentar la investigación a partir de la definición del marco conceptual alrededor del objeto de estudio.
- ✓ Dar mantenimiento al Módulo de Gestión de Auditoría Web de Xilema – PlatSI para adaptarlo a las nuevas pautas de diseño del centro de Telemática.

- ✓ Diseñar la estrategia de pruebas para la verificación y validación de las funcionalidades definidas para el sistema.

Para dar cumplimiento a estos objetivos se plantearon las siguientes **tareas de investigación:**

- ✓ Estudio y selección de las herramientas, metodología y tecnologías necesarias para el desarrollo del Módulo de Gestión de Auditoría web de Xilema – PlatSI v2.0.
- ✓ Análisis de los procesos que tienen lugar en el Módulo de Gestión de Auditoría Web de Xilema - PlatSI v1.0 para una correcta implementación de los mismos.
- ✓ Estudio y revisión de la documentación referente a Xilema – PlatSI v1.0 para una correcta implementación de la aplicación web.
- ✓ Descripción de las funcionalidades utilizando los artefactos definidos en la metodología de desarrollo propuesta para una correcta implementación de los requerimientos del cliente.
- ✓ Estudio de los diferentes tipos de pruebas propuesta por la metodología utilizada para verificar el correcto funcionamiento de la herramienta desarrollada.

La idea a defender es que el mantenimiento adaptativo al Módulo de Gestión de Auditoría Web de Xilema – PlatSI permitirá cumplir las nuevas pautas de diseño del Centro de Telemática.

Para desarrollar este trabajo se utilizaron los siguientes **métodos científicos de investigación:**

Dentro de los **métodos teóricos** se utilizó:

Analítico-Sintético: Se empleó para el análisis de los diferentes tipos de mantenimiento y sus características. Luego se escoge cual utilizar para solucionar el problema de la investigación.

Modelación: Se utilizó con el objetivo de realizar una reproducción simplificada de varios procesos que ocurren en el módulo de auditoría web.

Inductivo-Deductivo: Se utiliza para proponer una solución al problema planteado, a partir del estudio que se realiza, se arriban a proposiciones generales y se infieren casos particulares por un razonamiento lógico que pueden ser verificados en la práctica.

También se utilizan **métodos empíricos:**

Entrevista: Se realizan varias entrevistas al cliente para obtener información y en base a esta desarrollar los requisitos funcionales de la aplicación.

Observación: Se utilizó este método para observar el sistema en su versión anterior y analizar la documentación correspondiente al mismo.

El trabajo de diploma se divide en 4 capítulos, los cuales estarán estructurados de la siguiente forma:

Capítulo1. "**Fundamentación Teórica**": Contiene el marco conceptual en que se realiza la investigación. Hace una introducción a la metodología de desarrollo, herramientas y lenguaje de programación que apoyen la implementación de una solución.

Capítulo 2. "**Características del sistema**": Se realiza una caracterización del sistema, se plantean las historias de usuarios (HU) y se plantea una propuesta del prototipo no funcional del software.

Capítulo 3. "**Diseño e implementación del sistema**": Este capítulo contiene todo el diseño del sistema propuesto, se definen los patrones de diseño así como las clases del negocio a través de las tarjetas de Clase-Responsabilidad-Colaborador (CRC). Además se expone todo lo relacionado con la implementación del sistema.

Capítulo 4. "**Pruebas**": Se expone todo lo relacionado con las pruebas del sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

En el presente capítulo se hace referencia a los distintos tipos de mantenimiento existentes para determinar cuál utilizar en este trabajo de diploma. Se hace un análisis de los principales marcos de trabajo (*framework*), que fueron necesarios para el desarrollo del módulo de gestión de auditorías web de Xilema - PlatSI. Además se describen peculiaridades de la metodología, el lenguaje y las herramientas para el desarrollo de este módulo.

1.1 Tipos de mantenimiento de software

El mantenimiento de software es una de las actividades más comunes en la Ingeniería de Software y es el proceso de mejora y optimización del software desplegado. Dentro de los distintos tipos de actividades de mantenimiento se encuentran (4):

- ✓ **Mantenimiento Correctivo (corrección):** consiste en el cambio en el software para corregir los defectos encontrados que originan un comportamiento distinto al deseado.
- ✓ **Mantenimiento perfectivo (mejora):** consiste en añadir funciones a los requisitos funcionales originales, a fin de satisfacer las solicitudes del cliente/usuario que puede descubrir funciones adicionales que van a producir beneficios.
- ✓ **Mantenimiento preventivo (prevención):** consiste en la revisión constante del software para detectar posibles focos de problemas que puedan surgir en el futuro.
- ✓ **Mantenimiento adaptativo (adaptación):** produce modificación en el software para acomodarlo a los cambios de su entorno externo. Teniendo en cuenta el cambio en el entorno original (por ejemplo: cpu, el sistema operativo, las reglas de empresa, las características externas de productos) para el que se desarrolló el software.

De estas actividades de mantenimiento, tan sólo el 20% de los esfuerzos se dedican a la corrección de los errores. El restante 80% se emplea a la adaptación a los cambios en su entorno exterior, a las mejoras que solicitan los usuarios o al rediseño del sistema (4).

Para este trabajo de diploma se decide escoger el mantenimiento adaptativo debido a que se adapta a los objetivos de dicho trabajo. Este mantenimiento permitirá modificar el Módulo de Gestión de Auditoría Web de Xilema - PlatSI al nuevo lenguaje de programación y marco de trabajo con que se desarrolla en el centro de Telemática.

1.2 Plataforma de Seguridad en las Tecnologías de la Información v 1.0

Xilema – PlatSI es una solución tecnológica integral, acorde con las características de los procesos del departamento de Seguridad Informática de ETECSA[1] para la realización de verificaciones de seguridad en las aplicaciones web. Se integra con herramientas para la ejecución de pruebas de seguridad de forma distribuida y una plataforma de distribución de tareas (Bambú) para el procesamiento de las solicitudes de pruebas. Además posee un componente para parsear los resultados en XML[2] arrojados por cada una de las herramientas (Nikto y Acunetix)[3] y permite generar un informe final con las vulnerabilidades encontradas en la aplicación que es objeto de análisis. También implementa un módulo de administración para la gestión de usuarios y permisos en el sistema.

Este entorno web además de distribuir pruebas de seguridad de forma automática y remota, tiene la capacidad de procesar y almacenar en una base de datos común los resultados de dichas pruebas. La plataforma utiliza una arquitectura Cliente – Servidor, por lo que tiene funcionalidades tanto en el servidor como en el cliente. En el servidor se recibe de la aplicación web la solicitud de prueba y se le asigna al agente correspondiente. Además se asigna la solicitud de ejecución de herramientas de pruebas de seguridad a los hilos de ejecución, y una vez asignada la solicitud el mismo debe ser capaz de ejecutar la herramienta que corresponde a la solicitud realizada.

Esta solución se desarrolló utilizando PHP como lenguaje de programación para la aplicación web, Symfony2 como marco de trabajo, PostgreSQL como gestor de base de datos, Python como lenguaje de programación para el distribuidor de tareas (Bambú), así como el protocolo de comunicación del Middleware ICE (Internet Communication Engine) para el envío de la solicitud de prueba desde la aplicación web y SQlite para la base de datos local de los agentes. El acceso al sistema está basado en roles los cuales se definen a continuación:

Solicitante: persona encargada de realizar una solicitud de auditoría.

Planificador: persona encargada de crear una auditoría general y juega el papel de solicitante.

Especialista: persona que crea una auditoría web, edita las vulnerabilidades encontradas y agrega vulnerabilidades a la base de datos.

---

[1] Empresa de Telecomunicaciones de Cuba S.A

[2] Extensible Markup Language (Lenguaje de Marcas Extensible).

[3] Herramientas para analizar servidores web en busca de vulnerabilidades.

Responsable: persona que redacta el informe general de la auditoría web.

El Módulo de Gestión de Auditoría a Aplicaciones Web de Xilema – PlatSI tiene el siguiente flujo de trabajo: el solicitante accede al sistema para realizar una solicitud insertando los datos de esta en un formulario. Una vez creada la solicitud el planificador crea una auditoría general asignándole un especialista y un responsable. El especialista teniendo asignada una auditoría general, la acepta y realiza una auditoría web, eligiendo la herramienta a utilizar. Una vez que las herramientas terminan de hacer los test a la aplicación web, devuelven un conjunto de vulnerabilidades encontradas en dicha aplicación. Si la vulnerabilidad es real el especialista agrega la vulnerabilidad al informe o crea una vulnerabilidad que no haya sido detectada por las herramientas, edita dicha vulnerabilidad para dar recomendaciones y agregarla al informe y realiza una prueba de concepto, donde describe la vulnerabilidad y añade una foto como adjunto de donde se encuentra y termina la auditoría web cambiando el estado de la asignación de la auditoría general. Por último el responsable de la asignación obtiene los resultados de la auditoría web realizada por el especialista y redacta un informe general con las vulnerabilidades encontradas y las recomendaciones para eliminar las mismas, generando un informe en formato PDF.

1.2. Metodología de Desarrollo

Se utilizará una metodología ágil para el desarrollo del presente trabajo de diploma, específicamente XP[4], la cual está centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo (5).

A continuación se muestran las razones de elección:

- ✓ El equipo de desarrollo está compuesto por dos integrantes.
- ✓ El cliente y los programadores se comunican directamente y de forma continua, ya que el cliente forma parte del equipo de proyecto para establecer prioridades y resolver dudas en caso que existan.

[4] Extreme Programming(Programación Extrema)

- ✓ Permite la realización de ciclos muy cortos tras los cuales se muestran resultados, minimizando el tener que rehacer partes que no cumplen con los requisitos y ayudando a los programadores a centrarse en los que son más importantes.
- ✓ Permite definir en cada iteración las historias de usuario a desarrollar.
- ✓ Se enfoca más en el desarrollo que en la documentación, ya que el proyecto es pequeño y cuenta de poco tiempo para su desarrollo.

1.3. Herramientas, lenguajes y tecnologías a utilizar

1.3.1 Herramienta CASE: Visual Paradigm 8.0

Es una herramienta CASE[5] de modelado que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite realizar ingeniería tanto directa como inversa. La herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto; genera la documentación del proyecto automáticamente en varios formatos como Web o PDF[6], y permite el control de versiones. Posee como lenguaje de modelado además de UML[7], BPMN, los cuales son empleados en la modelación de bases de datos y los procesos del negocio respectivamente (7).

A continuación se muestran las ventajas de su uso en este trabajo:

- ✓ Permitió transformar los modelos de Django en un Modelo - Entidad - Relación.
- ✓ Permitió realizar el modelado del negocio a utilizar a través de su diagrama de proceso de negocio, con una notación para el modelado de negocio (BPMN).

1.3.2 Notación para el Modelado del Proceso de Negocio

Business Process Model Notation (BPMN) es una notación gráfica que describe la lógica de los pasos de un proceso de negocio. Esta notación ha sido especialmente diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades (8). BPMN está dirigido a todo aquel que necesita definir, documentar y hacer más eficientes sus procesos de negocio con el estándar más avanzado y aceptado en el mundo (9).

Ventaja de BPMN para este trabajo:

---

[5] Computer Aided Software Engineering (Ingeniería de Software Asistida por Computadora).

[6] Portable Document Format (Formato de Documento Portátil).

[7] Unified Modeling Language (Lenguaje Unificado de Modelado).

- ✓ Se creó un diagrama para la representación del proceso de negocio del Módulo de Gestión de Auditoría Web de Xilema - PlatSI.
- ✓ Se disminuyó la brecha entre el proceso de negocio y la implementación del mismo.
- ✓ Se modeló el proceso de negocio de una manera unificada permitiendo un mejor entendimiento del funcionamiento del Módulo de Gestión de Auditoría Web de Xilema – PlatSI.

1.3.3 Entorno de Desarrollo Integrado: PyCharm 4.0

PyCharm es desarrollado por la empresa JetBrains y debido a la naturaleza de sus licencias tiene dos versiones, la *Community* que es gratuita y orientada a la educación y al desarrollo puro en Python y la Professional, que incluye más características como el soporte a desarrollo web (11). En este trabajo es utilizado para desarrollar en el lenguaje de programación Python ya que proporciona análisis de código, depuración gráfica, y soporte para el desarrollo web con Django.

1.3.4 Lenguaje de Programación: Python 2.7

Python 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 (12). Es el lenguaje definido por el Centro de Telemática en su proceso de estandarización para desarrollar sus productos.

A continuación ventajas de este lenguaje:

- ✓ Permite escribir programas bastante cortos gracias a la sintaxis simple de su código.
- ✓ Ofrece un entorno interactivo que facilita la realización de pruebas y ayuda a despejar dudas acerca de ciertas características del lenguaje.
- ✓ Puede usarse como lenguaje imperativo procedimental o como lenguaje orientado a objetos, conteniendo estructuras de datos fáciles de manipular.

1.3.5 Marco de trabajo: Django 1.7

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[8] y utiliza licencia BSD[9] (13). Es el marco de trabajo asociado a Python definido por el Centro de Telemática para su proceso de estandarización.

A continuación alguna de sus ventajas:

- ✓ Permite mostrar y validar formularios de manera muy simple, fácilmente puede manipular el código del formulario adaptándolo a las necesidades de la aplicación y a una respuesta simple para el usuario.
- ✓ Convierte los datos enviados por los usuarios a través de formularios, en estructuras de datos que pueden ser manipuladas fácilmente.
- ✓ A través de plantillas ayuda a separar el contenido de la presentación, evitando tener que manipular la lógica de negocio cuando se tienen que realizar cambios de apariencia en el sitio web.

1.3.6 Marco de trabajo: Xilema – Base – Web 2.0

Es un marco de trabajo 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. 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.

1.3.7 Lenguaje de Marcas de Hipertexto: HTML 5

HTML5 es la última evolución de la norma que define HTML[10]. Se trata de una nueva versión del lenguaje, con nuevos elementos, atributos y comportamientos, y un conjunto más amplio de tecnologías que permite crear sitios Web y las aplicaciones más diversas y de gran alcance. Su uso permitió el empleo de nuevas etiquetas[11] que le dan una mejor estructura y organización al documento HTML (14).

---

[8] Don't Repeat Yourself (Una vez y sólo una)

[9] Berkeley Software Distribution (Licencia de software libre permisiva).

[10] *HyperText Markup Language* (Lenguaje de marcas de hipertexto).

[11] Código que se incluye en los archivos creados con el lenguaje HTML para estructurar, añadir significado o formato al contenido a una página web.

A continuación se describe su uso en este trabajo:

- ✓ Es el lenguaje utilizado por las plantillas de Django.
- ✓ Permitió definir texto, tablas e imágenes en el diseño de las plantillas.

1.3.8 Sistema Gestor de Base de Datos: PostgreSQL 9.1

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (15). Este es el mismo sistema gestor de base de datos utilizado en la versión actual de Xilema - PlatSI, por lo que se decide no cambiarlo, ya que el proceso de estandarización no define uno en específico. En este trabajo es utilizado para:

- ✓ Gestionar diferentes usuarios y grupos, así como los permisos asignados a cada uno de ellos.
- ✓ Almacenar los datos recogidos en los formularios.

1.3.9 Administrador de Base de Datos: PgAdmin III

Entorno de escritorio visual que permite conectarse a bases de datos PostgreSQL que estén ejecutándose en cualquier plataforma. Facilita la gestión y administración de bases de datos ya sea mediante instrucciones SQL[12] o con ayuda de un entorno gráfico. Permite acceder a todas las funcionalidades de la base de datos; consulta, manipulación y gestión de datos (16).

Conclusiones del capítulo

En este capítulo se abordó acerca de los tipos de mantenimientos de software. Se analizó Xilema – PlatSI en su versión 1.0. Además se definió la metodología de desarrollo, herramientas y tecnologías que guiarán la producción de Módulo de Gestión de Auditoría Web de Xilema – PlatSI v2.0.

[12] Structured Query Language (Lenguaje de Consulta Estructurado).

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

En el presente capítulo se elabora una propuesta del módulo a implementar, detallando las características del sistema. Cuenta con la descripción, modelación de los procesos y los principales artefactos generados, teniendo en cuenta las fases de la metodología XP. Además se describen las historias de usuarios para una mejor comprensión de los requerimientos del cliente.

2.1 Propuesta del sistema

La propuesta del sistema está enfocada en dar un mantenimiento adaptativo al módulo de auditorías web (MAAWeb) de Xilema – PlatSI v1.0, desarrollando esta nueva versión en Python como lenguaje de programación, Django como marco de trabajo y Xilema – Base – Web como arquitectura definida por el Centro de Telemática.

Una vez que el solicitante accede al sistema y realiza una solicitud, el planificador crea una auditoría general asignándole un especialista y un responsable. Luego de creada la auditoría general el especialista revisas sus asignaciones y crea una auditoría web a partir de la solicitud correspondiente con estado aceptada. El especialista teniendo en cuenta su propio conocimiento y experiencia puede crear nuevas vulnerabilidades o agregar a la auditoría alguna de las que ya estén en la Base de Datos. Como parte del flujo el mismo debe realizar la prueba de concepto que es donde se describe y adjunta una foto de la vulnerabilidad, además de editar las vulnerabilidades en cuanto al estado y el nivel de riesgo que posea la misma, es aquí donde se permite agregar o no dicha vulnerabilidad al informe. Por último termina editando la auditoria web cambiando su estado a resuelta. Una vez realizado todo este proceso el responsable se encarga de realizar el informe general a partir de las vulnerabilidades encontradas durante la auditoría web y generarlo en un archivo PDF.

Para una mejor comprensión del funcionamiento de este módulo la figura 1 muestra un diagrama del proceso de negocio.

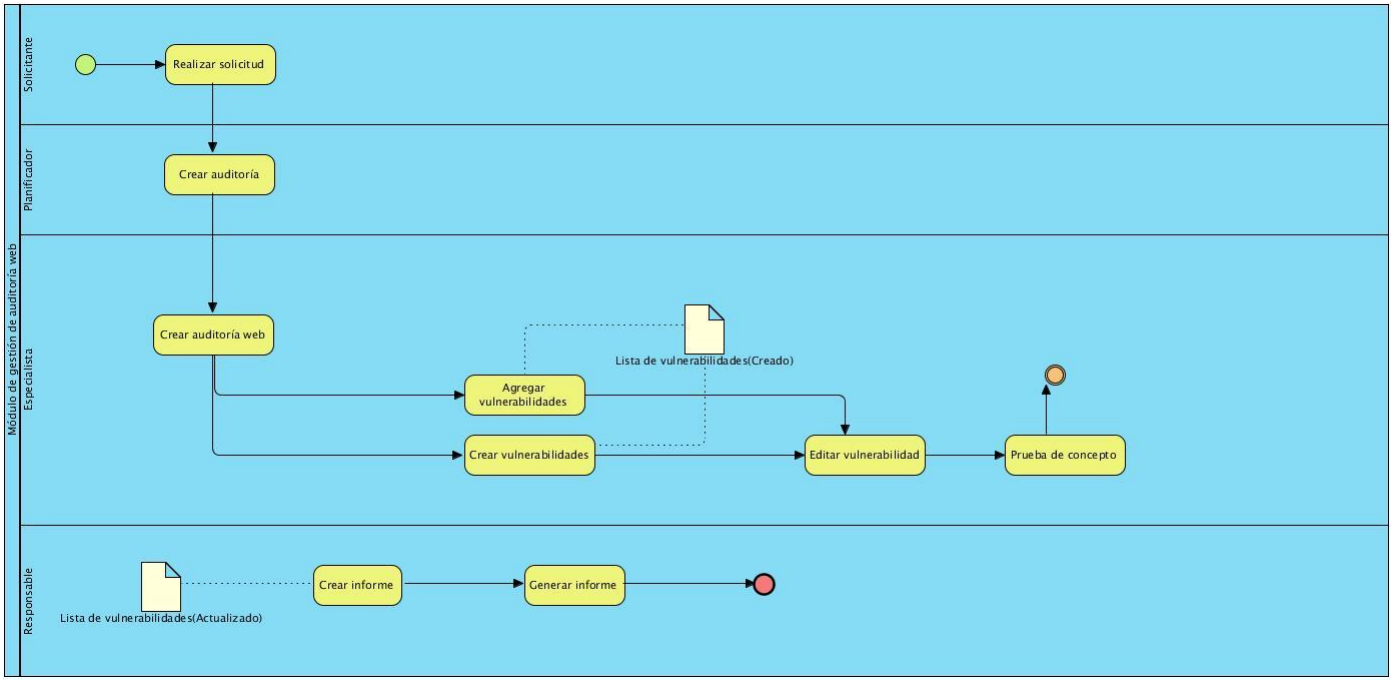


Figura 1: Diagrama de proceso de negocio

Además el sistema cumple las siguientes reglas de negocio:

- ✓ Para que una auditoría general se convierta en una auditoría web el especialista debe cambiar el estado de la misma a "Aceptada".
- ✓ Para que una vulnerabilidad detectada se convierta en una vulnerabilidad encontrada debe añadirse al informe.
- ✓ Para terminar una auditoría web la misma debe tener todas sus vulnerabilidades en estado "Terminado" y una evaluación del riesgo que representa dicha vulnerabilidad en el sistema.
- ✓ Para que el responsable pueda generar un informe técnico y general la auditoría web debe estar en estado" Cerrada".

2.2 Fase de Exploración

La fase de Exploración es la primera fase definida por la metodología XP. Es aquí donde se define el alcance real del sistema. Los clientes definen las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y procesos que se utilizan en el proyecto.

2.2.1 Lista de funcionalidades a implementar

Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Para este sistema los requisitos funcionales identificados son:

RF 1 Gestionar usuarios y grupos

- ✓ Crear grupos
- ✓ Crear usuarios

RF 2 Gestionar solicitud

- ✓ Crear solicitud
- ✓ Mostrar lista de solicitudes
- ✓ Editar solicitud
- ✓ Ver detalles de solicitud
- ✓ Eliminar solicitud

RF 3 Gestionar auditoría general

- ✓ Crear auditoría general
- ✓ Editar auditoría general
- ✓ Mostrar lista de auditorías generales
- ✓ Ver detalles de auditoría general
- ✓ Eliminar auditoría general

RF 4 Gestionar auditoría web

- ✓ Mostrar lista de asignaciones
- ✓ Mostrar detalles de las asignaciones
- ✓ Cambiar estado de las asignaciones
- ✓ Crear auditoría web
- ✓ Editar auditoría web
- ✓ Ver detalles de la auditoría web

- ✓ Mostrar lista de auditorías web
- ✓ Mostrar datos de auditoría web
- ✓ Terminar auditoría web

RF 5 Gestionar vulnerabilidades

- ✓ Crear vulnerabilidad
- ✓ Editar vulnerabilidad
- ✓ Realizar prueba de concepto
- ✓ Ver detalles de la vulnerabilidad
- ✓ Eliminar vulnerabilidad

RF 6 Gestionar informe

- ✓ Mostrar lista de auditorías web cerradas
- ✓ Generar informe técnico
- ✓ Crear informe general
- ✓ Ver detalles del informe

2.2.2 Historias de usuarios (HU)

Las historias de usuarios definen en XP los requisitos funcionales del sistema. Son escritas por el cliente en un lenguaje no técnico, sin hacer mucho hincapié en los detalles y especificando el tiempo estimado de duración de la funcionalidad que describe. También se utilizan en la fase de pruebas, para comprobar si el sistema cumple con lo establecido en las historias de usuarios. Además el cliente define la prioridad en el negocio y el riesgo en el desarrollo que tiene cada una de estas HU. Estos parámetros se definen a continuación:

La prioridad en el negocio:

- ✓ Alta: Se le otorga a las HU que resultan funcionalidades fundamentales en el desarrollo del sistema, a las que el cliente define como principales para el control integral del sistema.
- ✓ Media: Se le otorga a las HU que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.

- ✓ Baja: Se le otorga a las HU que constituyen funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo, a la estructura y no tienen nada que ver con el sistema en desarrollo.

El riesgo en el desarrollo:

- ✓ Alta: Cuando en la implementación de las HU se considera la posible existencia de errores que lleven a la inoperatividad del código.
- ✓ Media: Cuando pueden aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión.
- ✓ Baja: Cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto.

El cliente y el equipo de desarrollo trabajan en conjunto para definir como agrupar las HU para su lanzamiento. A continuación se muestran las Historias de usuarios definidas:

Tabla 1: Maqueta de una Historia de Usuario

Historia de usuario	
Número: Número de la HU incremental en el tiempo.	**Usuarios:** Involucrados en el desarrollo de la HU.
Nombre de la Historia de usuario: El nombre de la HU es para identificarla mejor entre los desarrolladores y el cliente.	
Prioridad en el negocio: (Alta / Media / Baja)	**Riesgo en Desarrollo:** (Alta / Media / Baja)
Puntos estimados: (El tiempo estimado que se demorará el desarrollo de la HU en semanas(5 días))	**Iteración Asignada:** (Número de la iteración)
Programador responsable: (Los programadores encargados de realizar la aplicación)	
Descripción: (Breve descripción de la HU)	
Observaciones: Señalamiento o advertencia del sistema.	

Tabla 2: Gestionar solicitud

Historia de usuario	
Número: 2	**Usuarios:** Solicitante
Nombre de la Historia de usuario: Gestionar solicitud	
Prioridad en el negocio: Alta	**Riesgo en Desarrollo:** Bajo
Puntos estimados: 1	**Iteración Asignada:** 1
Programadores responsables**:** Daniela González Begué y Hector V. Potete Muñoz	
Descripción: El sistema debe permitir la gestión de las solicitudes, lo cual se refiere a la creación, modificación y eliminación de las mismas, así como listarlas y ver los detalles de cada una de ellas. Los campos asociados a dichas solicitudes son: Solicitante, entidad, fecha de creación y creador.	
Observaciones: Para acceder al sistema el usuario debe estar previamente autenticado.	

Tabla 3: Gestionar auditoría general

Historia de usuario	
Número: 3	**Usuarios:** Planificador
Nombre de la Historia de usuario: Gestionar auditoría general	
Prioridad en el negocio: Alta	**Riesgo en Desarrollo:** Media
Puntos estimados: 1	**Iteración Asignada:** 2
Programadores responsables: Daniela González Begué y Hector V. Potete Muñoz	
Descripción: El sistema debe permitir la gestión de las auditorías generales, lo cual se refiere a la creación, modificación y eliminación de las mismas, así como listarlas y ver los detalles de cada una de ellas. Los campos asociados a dichas auditorías generales son: Auditoría, Solicitante, Entidad, Fecha de inicio, Fecha de fin, Especialista y Responsable.	
Observaciones: Para acceder al sistema el usuario debe estar previamente autenticado.	

Tabla 4: Gestionar auditoría web

Historia de usuario	
Número: 4	**Usuarios:** Especialista
Nombre de la Historia de usuario: Gestionar auditoría web	
Prioridad en el negocio: Alta	**Riesgo en Desarrollo:** Media
Puntos estimados: 1.4	**Iteración Asignada:** 2
Programadores responsables: Daniela González Begué y Hector V. Potete Muñoz	
Descripción: El sistema debe permitir la gestión de las auditorías web, lo cual se refiere a la creación y modificación de las mismas, así como listarlas, ver los detalles de cada una de ellas, terminarlas o generar un informe a partir de sus datos. Los campos asociados a dichas auditorías web son: Nombre del producto, Host, Estado, Evaluación de riesgo, Fecha de inicio y Fecha de fin.	
Observaciones: Para acceder al sistema el usuario debe estar previamente autenticado. Las auditorías web no podrán ser eliminadas.	

Las demás HU pueden verse en el Anexo II.

2.3 Lista de reserva del producto

Los requerimientos no funcionales de una aplicación son características que debe poseer todo software. Para el desarrollo de la aplicación propuesta se ha definido la siguiente lista de reserva del producto:

2.3.1 Usabilidad

- ✓ El diseño de la aplicación web mostrará al usuario el estado de las acciones que se realizarán, a través de los mensajes de confirmación y de error.
- ✓ Es necesaria una preparación previa para operar con la Plataforma.

2.3.2 Software

- ✓ Para la ejecución de la aplicación web en el cliente se usará el navegador Mozilla Firefox 28.0 o superior.

- ✓ Para el despliegue de la aplicación en el servidor se utilizará el Intérprete Python 2.7 y como gestor de base de datos PostgreSQL 9.1.

2.3.3 Hardware

Para las estaciones cliente

- ✓ Memoria RAM de 512 MB
- ✓ Microprocesador 2.6 GHz

Para los servidores

- ✓ Memoria RAM de 512 MB
- ✓ Microprocesador 2.6 GHz
- ✓ Disco duro 10 GB

2.4 Fase de Planificación

En la fase de planificación de la metodología XP se realiza la estimación del esfuerzo que costará la implementación de cada historia de usuario, en esta metodología ágil las métricas son libres, por lo que puede utilizarse cualquier criterio para medir el desempeño del proyecto en cuestión, aunque la más usada es la medida de puntos; en la que cada punto en esta métrica es considerado como una semana de trabajo, donde los miembros de los equipos de desarrollo trabajan sin interrupciones.

2.4.1 Estimación de esfuerzo por historias de usuarios

Para el desarrollo de la aplicación propuesta se realizó una estimación de esfuerzo según el método juicio de experto basado en la experiencia por cada una de las historias de usuario identificadas, resultados que se muestran a continuación:

Tabla 5: Estimación de esfuerzo por HU

Historias de usuario	**Puntos de estimación**
Gestionar usuarios y grupos	0.4
Gestionar solicitud	1
Gestionar auditoría general	1

Gestionar auditoría web	1.8
Gestionar vulnerabilidades	1
Gestionar informe	0.8

2.4.2 Plan de Iteraciones

Una vez identificadas las historias de usuario del sistema y estimado el esfuerzo dedicado a la realización de cada una de estas se procede a la planificación de la etapa de implementación de la aplicación. Se considera realizar la implementación del módulo en 3 iteraciones, las cuales se detallan a continuación:

Iteración 1

El objetivo de esta iteración es implementar las historias de usuario 1 y 2 por su menor complejidad y prioridad en el negocio. Durante la misma se trabajará de acuerdo a la arquitectura propuesta para el sistema, finalizando con una primera versión del producto final.

Iteración 2

En esta iteración se desarrollaran las historias de usuarios 3 y 4 pertenecientes a la gestión de auditorías generales y web.

Iteración 3

En esta iteración se desarrollarán las historias de usuario 5 y 6 correspondientes a la gestión de vulnerabilidades y creación del informe general.

2.4.3 Plan de duración de las iteraciones

El plan de duración de las iteraciones es el encargado de mostrar la duración estimada en semanas y días y el orden las historias de usuario que serán implementadas en cada iteración.

Tabla 6: Plan de duración de las iteraciones

Iteración	**Orden de la HU a implementar**	**Duración total**
1	Gestionar usuarios y grupos Gestionar solicitud	1 semana y 2 días
2	Gestionar auditoría general	2 semanas y 4 días

	Gestionar auditoría web	
3	Gestionar vulnerabilidad Gestionar informe	1 semana y 4 días

2.4.4 Plan de entregas

Se presenta el plan de entregas de cada iteración, detallando la fecha de inicio y fin de cada una de estas iteraciones.

Tabla 7: Plan de entregas

Historia de usuario	Módulo	Iteración 1 Inicio:15/04/2015 Fin: 23/04/2015	Iteración 2 Inicio: 24/04/2015 Fin: 14/05/2015	Iteración 3 Inicio: 18/05/2015 Fin: 28/05/2015
Gestionar usuarios y grupos Gestionar solicitud	MAAWeb v2.0	1ra entrega	Finalizada	Finalizada
Gestionar auditoría general Gestionar auditoría web	MAAWeb v2.0		2da entrega	Finalizada
Gestionar vulnerabilidades Gestionar informe	MAAWeb v2.0			3ra entrega

Conclusiones del capítulo

En este capítulo se describieron las principales características del sistema, con el objetivo de realizar una aplicación que se adapte a las necesidades del cliente. Se expuso la planificación propuesta el equipo de desarrollo, compuesta por iteraciones donde de forma incremental se

implementará el módulo de auditorías web. Además se propuso la lista de reservas del producto para el correcto funcionamiento del sistema.

CAPÍTULO 3: DISEÑO E IMPLEMENTACION DEL SISTEMA

En este capítulo se hace alusión a la fase de diseño de la metodología XP utilizada para la implementación del sistema que se propone. También se exponen los diferentes artefactos que se generan en el transcurso del capítulo y se procede al diseño de la base datos.

3.1 Arquitectura.

La arquitectura no es más que una vista estructural de alto nivel que define los estilos o grupos de estilos adecuados para cumplir con las características no funcionales de un software. Esta es importante como disciplina debido a que los sistemas de software crecen de forma tal, que resulta muy complicado que sean diseñados especificados y entendidos por un solo individuo (17).

Para el desarrollo de esta aplicación se propone una arquitectura cliente-servidor.

3.1.1 Arquitectura Cliente - Servidor

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Normalmente el servidor es una máquina bastante potente que actúa de depósito de datos y los clientes son estaciones de trabajo que solicitan varios servicios al servidor. Además será necesario que ambas partes estén conectadas entre sí mediante una red.

Dentro de las ventajas de la arquitectura cliente-servidor se encuentran:

- ✓ Centralización del control de los recursos, datos y accesos.
- ✓ Facilidad de mantenimiento y actualización del lado del servidor: Esto es porque el lado del servidor se puede mantener o actualizar fácilmente.
- ✓ Toda la información es almacenada en el lado del servidor, que suele tener mayor seguridad que los clientes (18).

Para una mejor comprensión de esta arquitectura observar la figura 2.

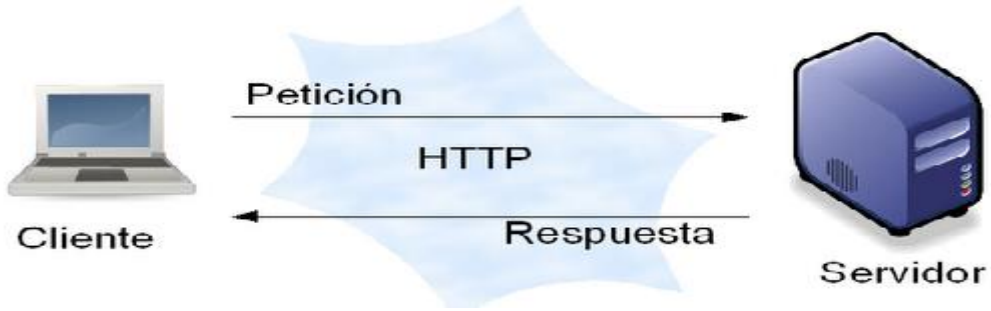


Figura 2: Arquitectura Cliente - Servidor

3.2 Patrón de arquitectura

Un patrón de arquitectura expresa la estructura fundamental para el sistema de software a desarrollar. El marco de trabajo Django, utilizado para el desarrollo de la aplicación web utiliza el patrón de arquitectura MTV[13], por tal motivo, este es el adoptado por el equipo de desarrollo.

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

MTV es una modificación del conocido Modelo - Vista – Controlador, utilizado por Django para convertirse en un *framework* más funcional. El modelo define los datos almacenados, que se encuentran en forma de clases de Python. La vista se presenta en forma de funciones de Python y determina que datos serán visualizados, también se encarga de la validación de datos a través de formularios. 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 (13).

En la figura 3 se observa el uso del patrón en el marco de trabajo.

---

[13] Model-Template-Views(Modelo-Plantilla-Vista)

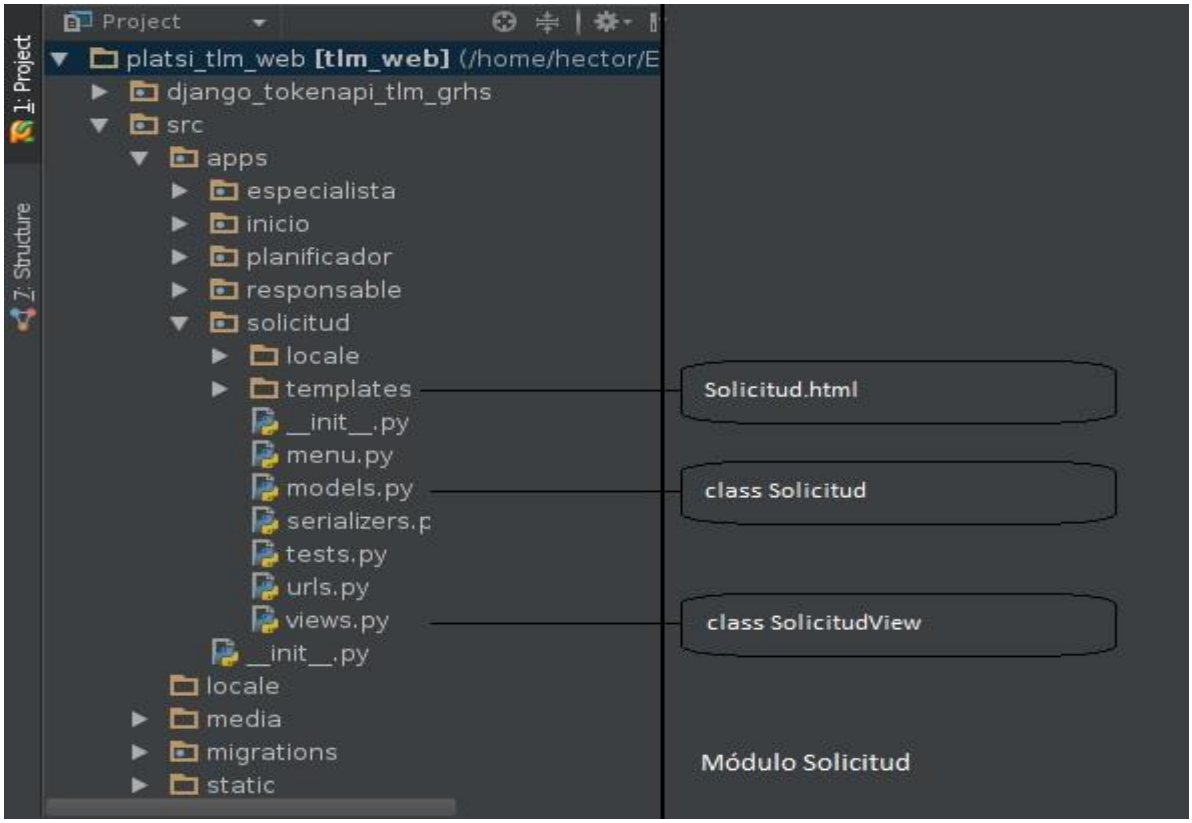


Figura 3: Patrón MTV en Django

3.4 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Representan una descripción de las clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. El uso de patrones posibilita estandarizar el modo en que se realiza el diseño y proporciona reusabilidad, extensibilidad y mantenimiento del código (19).

3.4.1 Patrones para Asignar Responsabilidades (GRASP)

Los Patrones GRASP[14] describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. A continuación se explican los patrones de asignación de responsabilidades que se implementan en el software (19):

[14]General Responsibility Assignment Software Patterns (Patrones Generales de Software para Asignación de Responsabilidades)

Experto: es el principio básico de asignación de responsabilidades, expresa que los objetos realizarán tareas en correspondencia con la información que poseen.

Problema: ¿de qué forma se puede lograr que cada clase cumpla con la responsabilidad que le corresponde?

- ✓ **Solución:** asignar una responsabilidad al experto en información, la clase que tiene la información necesaria para cumplir con la responsabilidad.

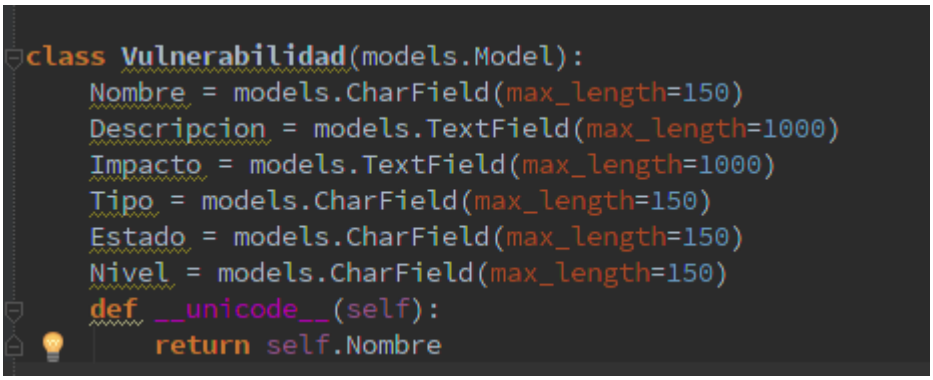


Figura 4: Fragmento de código 1

- ✓ **Ejemplo:** la clase **Vulnerabilidad** cuenta con la información necesaria para cumplir con las responsabilidades que les corresponden.

Creador: el patrón creador ayuda a identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases.

- ✓ **Problema:** ¿quién es el responsable de crear una nueva instancia de una clase?
- ✓ **Solución:** asignarle a una clase la responsabilidad de crear una instancia necesaria de otra clase.

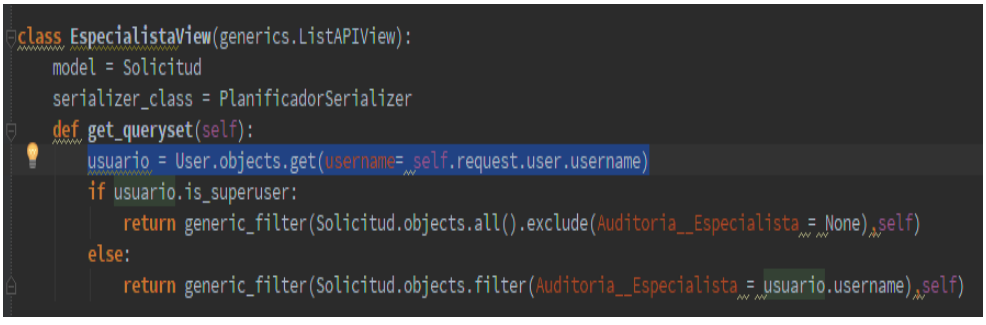


Figura 5: Fragmento de código 2

- ✓ **Ejemplo:** la clase **EspecialistaView** es la encargada de crear una nueva instancia de la clase Solicitud para ser visualizada.

Alta cohesión: la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase.

- ✓ **Problema:** ¿cómo mantener la complejidad manejable?
- ✓ **Solución:** asignar una responsabilidad de manera que la cohesión permanezca alta.

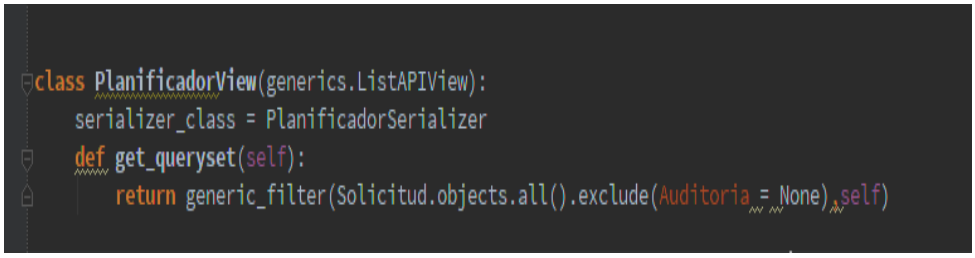


Figura 6: Fragmento de código 3

- ✓ **Ejemplo:** a la clase **PlanificadorView** se le asignan responsabilidades propias y que no tengan mucha complejidad.

Bajo acoplamiento: es un patrón que define las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

- ✓ **Problema:** ¿cómo mantener un bajo acoplamiento para lograr, entre otras cosas, alta reutilización?
- ✓ **Solución:** asignar responsabilidades a las clases manteniendo bajo el acoplamiento. Tratando que las clases estén lo menos ligadas posible, ya que de ser necesario realizar alguna modificación en una de ellas, la afectación en las demás sea mínima.

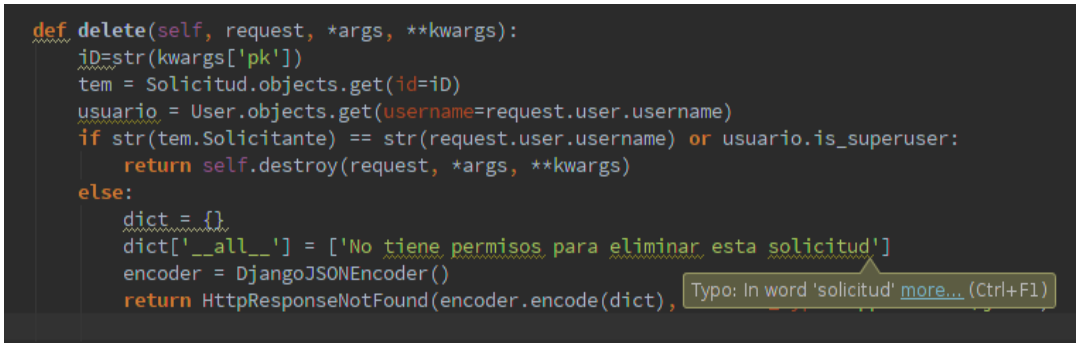


Figura 7: Fragmento de código 4

- ✓ **Ejemplo:** al método **delete** se le asigna la responsabilidad de eliminar de forma tal que solo él es el encargado de hacerlo.

3.5 Diseño de la base de datos empleada

El modelo físico de la base de datos se generó a partir de las clases modelos de Django. Contiene las clases persistentes, aquellas cuya durabilidad persiste más allá de la ejecución del software. En la figura 8 se muestra como está definido dicho modelo.

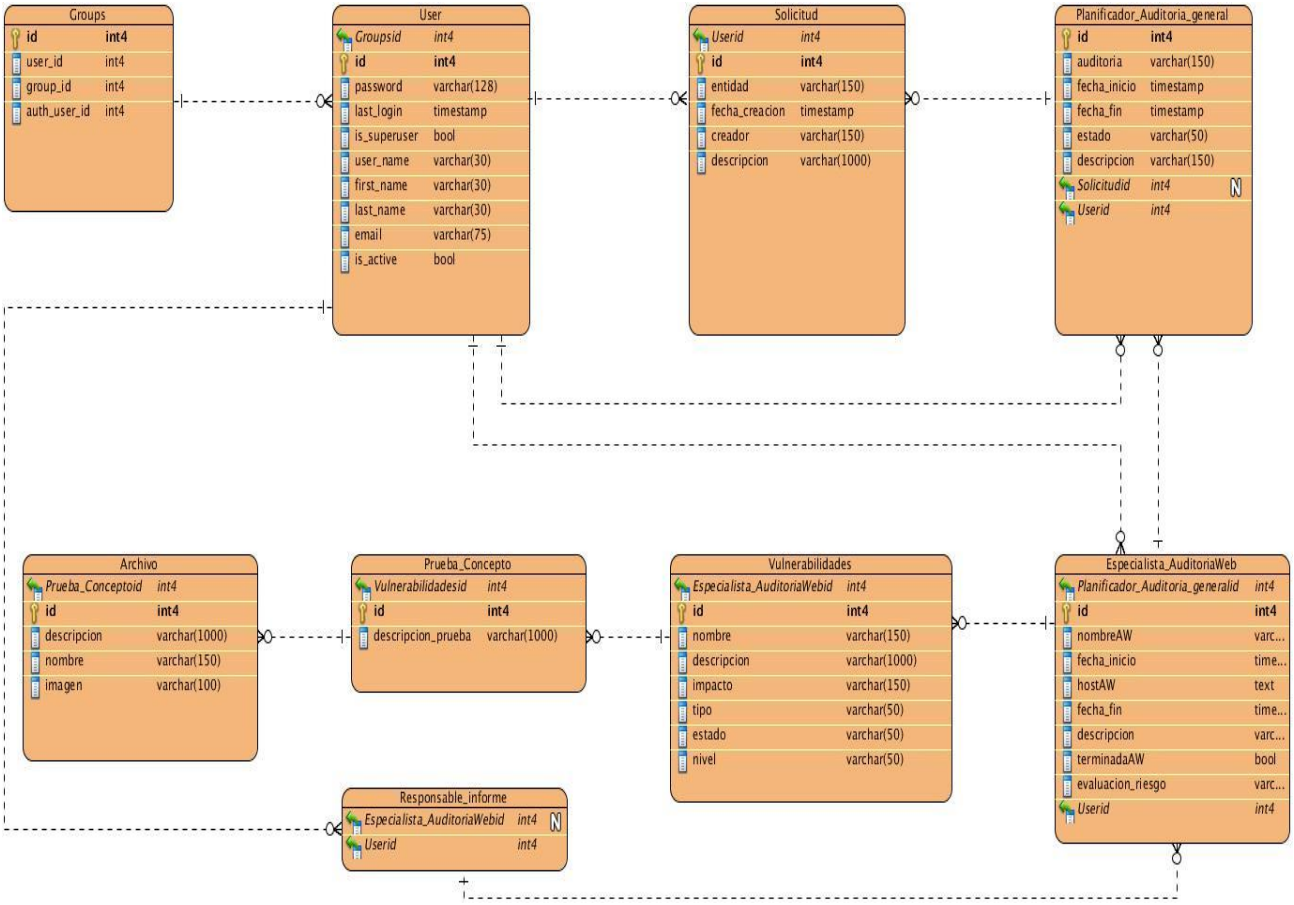


Figura 8: Modelo de la base de datos

Estas tablas tienen la siguiente estructura:

Tabla 8: 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 9: BD_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

Tabla 10: BD_Solisitud

Tabla de la base de datos		
Nombre: Solicitud		
Descripción: Guarda las solicitudes realizadas en el sistema.		
Atributo	**Tipo de dato**	**Descripción**
id	integer	Identificador de la solicitud.

		(llave primaria)
user_id	integer	Identificador de la tabla user.(llave foránea)
entidad	varchar	Entidad que realiza la solicitud
fecha_creacion	timestamp	Fecha de creación de la solicitud
creador	varchar	Indica si el creador es interno o externo
descripcion	varchar	Descripción de la solicitud

Tabla 11: BD_Planificador_Auditoría_general

Tabla de la base de datos		
Nombre: Planificador_Auditoria_general		
Descripción: Guarda las auditorias generales creadas por el planificador.		
Atributo	**Tipo de dato**	**Descripción**
id	integer	Identificador de la auditoría general (llave primaria)
Solicitud_id	integer	Identificador de la tabla Solicitud (llave foránea)
User_id	integer	Identificador de la tabla user (llave foránea)
auditoria	varchar	Nombre de la auditoría
fecha_inicio	timestamp	Fecha en que se iniciará la auditoría
fecha_fin	timestamp	Fecha en que concluirá la auditoría
estado	varchar	Estado en que se encuentra la auditoría

descripcion	varchar	Descripción de la auditoría

Tabla 12: BD_Especialista_Auditoría_web

Tabla de la base de datos		
Nombre: Especialista_Auditoria_web		
Descripción: Guarda las auditorias web creadas por el especialista.		
Atributo	**Tipo de dato**	**Descripción**
id	integer	Identificador de la auditoría web. (llave primaria)
nombreAW	varchar	Nombre de la auditoría web
hostAW	text	Host al que se le va a realizar la auditoría web
terminadaAW	bool	Indica si está terminada o no
fecha_inicio	timestamp	Fecha en que se iniciará la auditoría web
fecha_fin	timestamp	Fecha en que concluirá la auditoría web
evaluación_riesgo	varchar	Evaluación del riesgo que puede tener la auditoría
descripcion	varchar	Descripción de la auditoría
User_id	integer	Identificador de la tabla user (llave foránea)
Planificador_Auditoria_general_id	integer	Identificador de la tabla Planificador_auditoria_general (llave foránea)

Tabla 13: BD_Vulnerabilidades

Tabla de la base de datos

Nombre: Vulnerabilidades		
Descripción: Guarda las vulnerabilidades creadas por el especialista.		
Atributo	**Tipo de dato**	**Descripción**
id	integer	Identificador de la vulnerabilidad (llave primaria)
nombre	varchar	Nombre de la vulnerabilidad
descripcion	varchar	Descripción de la vulnerabilidad
estado	varchar	Estado en que se encuentra la vulnerabilidad
tipo	varchar	Tipo de vulnerabilidad
nivel	varchar	Nivel de la vulnerabilidad
Especialista_AuditoríaWeb_id	integer	Identificador de la tabla Especialista_AuditoriaWeb (llave foránea)

Tabla 14:BD_Prueba_Concepto

Tabla de la base de datos		
Nombre: Prueba_Concepto		
Descripción: Guarda las pruebas de concepto correspondientes a las vulnerabilidades.		
Atributo	**Tipo de dato**	**Descripción**
id	integer	Identificador de la prueba de concepto (llave primaria)
Vulnerabilidades_id	integer	Identificador de la tabla Vulnerabilidades (llave foránea)
descripción	varchar	Descripción del archivo.
imagen	varchar	Dirección de la imagen subida

Tabla 15: BD_Archivo

Tabla de la base de datos		
Nombre: Archivo		
Descripción: Guarda los datos correspondientes a los archivos subidos en las pruebas de concepto		
Atributo	**Tipo de dato**	**Descripción**
id	integer	Identificador del archivo. (llave primaria)
Prueba_Concepto_id	integer	Identificador de la tabla Prueba_Concepto (llave foránea)
descripción_prueba	varchar	Descripción de las pruebas de concepto

Tabla 16: BD_Pesponsable_Informe

Tabla de la base de datos		
Nombre: Responsable_Informe		
Descripción: Carga los datos de la auditoría web a la que se le realizará el informe.		
Atributo	**Tipo de dato**	**Descripción**
Especialista_AuditoriaWeb_id	integer	Identificador de la tabla Especialista_AuditoriaWeb (llave foránea)
user_id	integer	Identificador de la tabla user (llave foránea)
descripción_prueba	varchar	Descripción de las pruebas de concepto

3.6 Tarjetas Clase – Responsabilidad - Colaborador

XP como metodología de desarrollo propone el modelado de Clase - Responsabilidad - Colaborador (CRC), lo que constituye un modelo simple de organizar las clases más relevantes

para las funcionalidades de la plataforma. Un modelo CRC es una colección de tarjetas índices estándar que representan clases. Las tarjetas se dividen en tres secciones. A lo largo del borde superior de la tarjeta se escribe el nombre de la clase. En el cuerpo de la tarjeta a la izquierda se listan las responsabilidades de la clase que no es más que lo que la clase sabe o hace y a la derecha los colaboradores que son aquellas clases que se requieren para que una clase reciba la información necesaria para completar una responsabilidad (4).

Tabla 17: CRC Vista_Solicitud

Clase: Vista_Solicitud	
Responsabilidad	**Colaborador**
Se encarga mostrar los datos de una solicitud.	Usuario

Tabla 18: CRC Vista_Planificador

Clase: Vista_Planificador	
Responsabilidad	**Colaborador**
Se encarga de mostrar las solicitudes al planificador, para hacer una auditoría general.	Solicitud Usuario

Tabla 19: CRC Vista_Especialista

Clase: Vista_Especialista	
Responsabilidad	**Colaborador**
Se encarga de mostrar a los especialista, las auditorías que les han sido asignadas, para realizar la auditoría web.	Solicitud Usuario

El resto de las tarjetas CRC se encuentran en el ANEXO III.

3.7 Tareas de ingeniería

Las tareas de ingeniería son escritas por el equipo de desarrollo a partir de las historias de usuarios. Cada una de estas tareas describe a las historias de usuarios de una forma más detallada para su implementación, estimando un tiempo más aproximado a la realidad para el

desarrollo de cada una de ellas. A continuación se muestran las tareas de ingeniería correspondientes a la Historia de usuario 3, las restantes se exponen en el ANEXO III.

Tabla 20: Muestra de Tarea de Ingeniería

Tarea de Ingeniería	
Número tarea: (Los números deben ser consecutivos)	**Número Historia de Usuario:** (Número de la historia de usuario a la que pertenece la tarea)
Nombre Tarea: (Nombre que identifica la tarea)	
Tipo de tarea: (Las tareas pueden ser de: desarrollo, correlación, mejora, otra (especificar))	**Puntos estimados:** (Tiempo en semanas que se le asignará (estimado))
Fecha de inicio: (fecha en que inicia la tarea)	**Fecha de fin:** (fecha en que termina la tarea)
Programador responsable: (Los programadores encargados)	
Descripción: (Breve descripción de la tarea)	

Tabla 21: Tarea de ingeniería: Crear auditoría general

Tarea de Ingeniería	
Número tarea: 8	**Número Historia de Usuario:** 3
Nombre Tarea: Crear auditoría general.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 24/04/2015	**Fecha de fin:** 24/04/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para adicionar una auditoría general. Una vez que el planificador selecciona la opción Crear auditoría general, aparecerá un formulario con los campos necesarios para la creación de la misma.	

Tabla 22: Tarea de ingeniería: Mostrar lista de auditoría general

Tarea de Ingeniería

Número tarea: 9	**Número Historia de Usuario:** 3
Nombre Tarea: Mostrar lista de auditoría general.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 27/04/2015	**Fecha de fin:** 27/04/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para mostrar una lista de las auditorías generales existentes. Una vez que el planificador selecciona el menú Auditorías, podrá ver una tabla que refleja todas las auditorías generales y los datos correspondientes a cada una de ellas.	

Tabla 23: Tarea de ingeniería: Editar auditoría general

Tarea de Ingeniería	
Número tarea: 10	**Número Historia de Usuario:** 3
Nombre Tarea: Editar auditoría general.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 28/04/2015	**Fecha de fin:** 28/04/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para editar una auditoría general. Una vez que el planificador selecciona la opción Editar, aparecerá un formulario mediante el cual podrá cambiar los valores de los datos de la auditoría general en cuestión.	

Tabla 24: Tarea de ingeniería: Mostrar detalles de auditoría general

Tarea de Ingeniería	
Número tarea: 11	**Número Historia de Usuario:** 3
Nombre Tarea: Mostrar detalles de auditoría general.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 29/04/2015	**Fecha de fin:** 29/04/2015

Programador responsable: Daniela González Begué - Hector V. Potete
Descripción: Esta tarea facilita la creación de una interfaz para mostrar los detalles de la auditoría general. Una vez que el planificador selecciona la opción Ver detalles, aparecerá un formulario con los detalles de la auditoría general en cuestión.

Tabla 25: Tarea de ingeniería: Eliminar auditoría general

Tarea de Ingeniería	
Número tarea: 12	**Número Historia de Usuario:** 3
Nombre Tarea: Eliminar auditoría general.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 30/04/2015	**Fecha de fin:** 30/04/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la posibilidad de eliminar una auditoría general. Una vez que el planificador selecciona la opción Eliminar, se mostrará un mensaje de confirmación de eliminación.	

3.8 Estándares de codificación

Un estándar de codificación agrupa los aspectos relacionados con la generación del código. Los programadores lo realizan para trabajar de forma coordinada dentro del proyecto, de acuerdo con el tipo de lenguaje que utilizan y las normas que propone el mismo, tratando de lograr un código claro y legible tanto para él como para otro programador (21).

A continuación se muestran algunos estándares de codificación que se tuvieron en cuenta para la implementación del Módulo de Gestión de Auditoría Web de Xilema – PlatSI v2.0 haciendo uso del lenguaje Python:

- ✓ Se utilizó líneas separadas para las importaciones, además de colocarse al comienzo del archivo, luego de cualquier comentario o documentación del módulo, y antes de globales y constantes. Siempre siguiendo el orden:

1. Importaciones de la librería estándar
2. Importaciones terceras relacionadas
3. Importaciones locales de la aplicación / librería. Por ejemplo:

```
from django.core.validators import URLValidator
from django.utils.translation import ugettext as _
from rest_framework import serializers
from src.apps.especialista.models import Vulnerabilidad, AuditoriaWeb, PruebaConcepto, VulnerabilidadBaseConocimiento
from src.apps.solicitud.models import Solicitud
```

- ✓ Se utilizó un # (numeral) seguido de un espacio para hacer comentarios en línea. Por ejemplo:

```
def validate_NombreAW(self, attrs, source): # Para validar que el nombre de la Auditoria Web sólo admita números y letras
    value = (attrs[source])
    regex = r'[0-9A-Za-z ÑñáéíóúÁÉÍÓÚ]+'
```

- ✓ Se utilizó la convención “CapWords” (palabras que comienzan con mayúsculas) para los nombres de las clases. Por ejemplo:

```
class AuditoriaWebSerializer(serializers.ModelSerializer):
    Estado = serializers.SerializerMethodField('get_Estado')
```

- ✓ Se utilizó la convención “CapWords”, más el sufijo “Error” en los nombres de las clases excepciones. Por ejemplo:

```
raise serializers.ValidationError(_("Solo se admiten números y letras"))
```

- ✓ Se utilizó un nombramiento en minúscula, con las palabras separadas por un guión bajo para las funciones. Por ejemplo:

```
def validate_Host_AW(self, attrs, source):
```

- ✓ Se utilizó “self”, para el primer argumento de los métodos de instancia. Por ejemplo:

```
def validate_Host_AW(self, attrs, source):
```

3.9 Submódulos para la implementación

Para la implementación se dividió el Modulo de Gestión de Auditoría Web v2.0 en varios submódulos de Django. La figura 9 muestra como quedó estucturado:

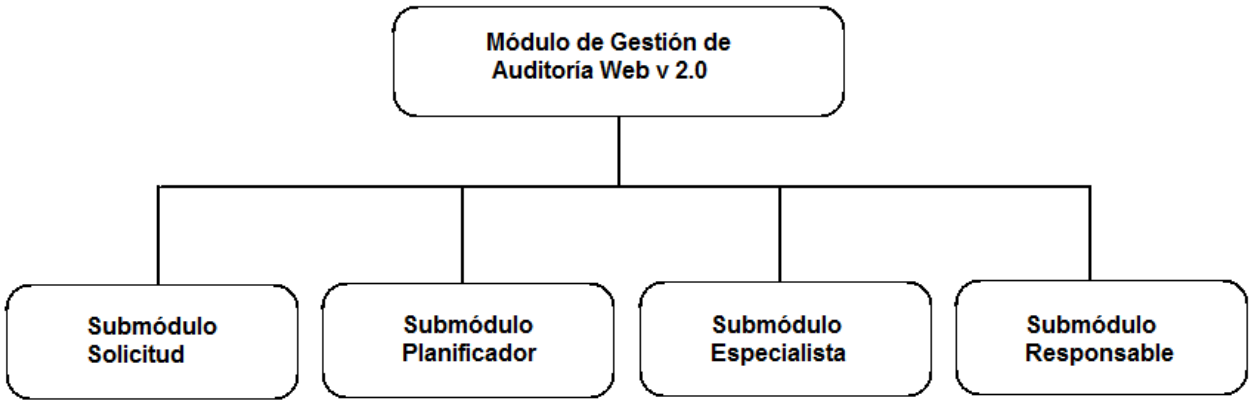


Figura 9: Estructura del Módulo de Gestión de Auditoría Web v2.0

Conclusiones del capítulo

En el presente capítulo se profundizó en el patrón arquitectónico utilizado por Django. Se detalló el modelo físico de la base de datos para obtener una mejor visión de la organización de los nuevos datos de Xilema - PlatSI, así como se identificaron y organizaron las clases relevantes para las funcionalidades de la aplicación. Además se trataron las tareas de ingeniería, las cuales favorecieron la descripción e implementación de las funcionalidades del sistema.

CAPÍTULO 4: PRUEBAS

En el presente capítulo se muestran las pruebas realizadas al conjunto de artefactos que se obtuvieron del trabajo realizado en la planificación y el diseño por parte del equipo de desarrollo. Se aplicarán métricas específicas para la validación de estos artefactos además de pruebas unitarias y de aceptación a la aplicación desarrollada tal como lo define la metodología utilizada.

4.1 Pruebas

Las pruebas miden el nivel de calidad de un software, determinando hasta donde este cumple las funcionalidades establecidas por el cliente. La metodología XP propone la Fase de Prueba durante el desarrollo del software, estableciendo pruebas tantas como sean posible con el objetivo de reducir el número errores no detectados durante la implementación, disminuir el tiempo entre la introducción de estos en el sistema y su detección; aumentar la seguridad y evitar efectos colaterales no deseados a la hora de realizar modificaciones en la aplicación. En este trabajo se implementaron las pruebas al final de cada iteración

4.1.1 Pruebas unitarias

Las pruebas unitarias son pruebas de caja blanca, realizadas constantemente por los programadores al concluir una parte de la implementación, el objetivo principal es verificar que las funcionalidades exigidas por el cliente se implementaron correctamente. Las pruebas unitarias se realizaron al concluir la implementación de cada funcionalidad utilizando la librería de Python (unit testing). Se realizaron 3 iteraciones con los siguientes resultados:

En la primera iteración, se analizaron las funcionalidades correspondientes al submódulo Solicitud, comprobando el correcto funcionamiento de cada una de estas.

Para la segunda iteración se analizaron las funcionalidades correspondientes al submódulo Planificador y Especialista (gestionar auditoría web) donde no se detectaron errores.

En la tercera iteración se comprobaron las funcionalidades correspondientes al submódulo Especialista (gestionar vulnerabilidades) y Responsable sin encontrar errores.

Una vez terminada cada iteración las no conformidades detectadas quedaron resueltas.

A continuación la figura 10 muestra los resultados obtenidos de las pruebas unitarias aplicadas a tres funcionalidades, mediante el uso del IDE Pycharm:

Test: se refiere a la ubicación donde se encuentra la prueba.
Time elapsed: se refiere al tiempo en que se demora ejecutar la prueba.

Results: se refiere al resultado de la prueba, devuelve uno si el código no tiene ningún error y cero en caso contrario.

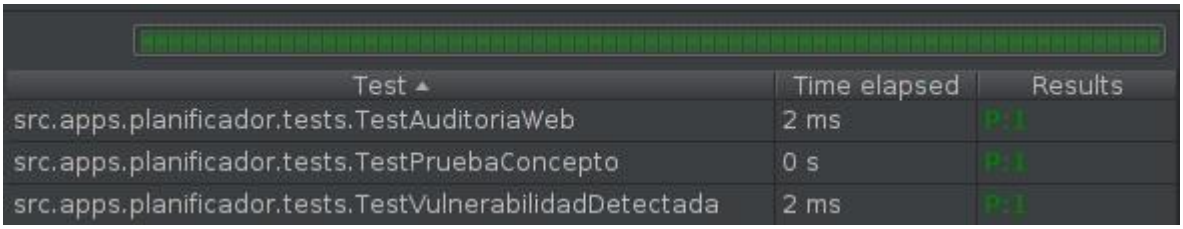


Figura 10: Resultados de las pruebas unitarias obtenidas mediante el IDE Pycharm

4.1.2 Pruebas de aceptación

Las pruebas de aceptación son pruebas de caja negra, especificadas por el cliente y se centran en las características y funcionalidades generales del sistema. Se derivan de las HU que se han implementado como parte de la liberación del software (20). La representación de las pruebas de aceptación correspondientes a cada una de las funcionalidades del Módulo de Gestión de Auditoría Web se hará mediante tablas con la siguiente estructura:

Tabla 26: Ejemplo de caso de prueba

	Clases Válidas	**Clases Inválidas**
Datos	(Descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las entradas válidas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado)	(Descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las posibles entradas inválidas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado)
Resultado esperado	(Descripción del resultado que se espera para entradas válidas)	(Descripción del resultado que se espera para entradas inválidas)
Resultado de la prueba	(Descripción del resultado que se obtiene)	(Descripción del resultado que se obtiene)
Observaciones	(Descripción de cualquier señalamiento o advertencia que sea necesario hacerle a la sección que se está probando)	

A continuación se muestran algunas de las pruebas de aceptación realizadas al Módulo de Gestión de Auditoría Web, las restantes se encuentran en el ANEXO IV:

Tabla 27: Caso de Prueba Gestionar solicitud

	Clases Válidas	**Clases Inválidas**
Datos	**Paso 1:** El solicitante selecciona la opción de Adicionar solicitud. **Paso 2:** Introduce los datos de la solicitud. Ejemplo: **Entidad:** Etecsa **Fecha de creación:** 2015-05-30 17:35:27 **Creador:** Solicitante externo **Descripción:** Prueba al sistema **Paso 3:** Pulsa el botón de aceptar.	**Paso 1:** El solicitante selecciona la opción de Adicionar solicitud. **Paso 2:** Introduce los datos de la solicitud. Ejemplo: **Entidad:** Etecsa% **Fecha de creación**: 2015-05-30 17:35:27 **Creador:** Solicitante externo **Descripción:** Prueba al sistema **Paso 3:** Pulsa el botón de aceptar.
Resultado esperado	El sistema adiciona una solicitud a la base de datos mostrándola al solicitante en una tabla.	El sistema muestra un mensaje indicando que el campo Entidad sólo puede contener letras y números.
Resultado de la prueba	Satisfactorio	Satisfactorio
Observaciones		

Tabla 28: Caso de prueba Gestionar auditoría general

	Clases Válidas	**Clases Inválidas**
Datos	**Paso 1:**	**Paso 1:**

	El planificador selecciona la solicitud a la cual desea crearle una auditoría. **Paso 2:** El planificador selecciona la opción crear auditoría. **Paso 3:** Introduce los datos de la auditoría. Ejemplo: **Auditoría:** AuditoríaX. **Fecha de inicio:** 2015-05-30 17:35:27 **Fecha de fin:** 2015-05-30 18:35:27 **Especialista:** Daniela **Responsable:** Pepe **Descripción:** Auditoría de prueba **Paso 4:** Pulsa el botón de aceptar.	El planificador selecciona la solicitud a la cual desea crearle una auditoría. **Paso 2:** El planificador selecciona la opción crear auditoría. **Paso 3:** Introduce los datos de la auditoría. Ejemplo: **Auditoría:** $asdasd*. **Fecha de inicio:** 2015-05-30 17:35:27 **Fecha de fin:** 2015-05-29 18:35:27 **Especialista:** Daniela **Responsable:** Pepe **Descripción:** Auditoría de prueba **Paso 4:** Pulsa el botón de aceptar.
Resultado esperado	El sistema adiciona una auditoría a la base de datos mostrándola al planificador en una tabla.	El sistema muestra un mensaje indicando que el campo Auditoría sólo puede contener letras y números y otro mensaje indicando que la fecha de inicio no puede ser posterior a la fecha de fin.
Resultado de la	Satisfactorio	Satisfactorio

prueba		
Observaciones		

Al terminar las pruebas de aceptación se obtuvo un total de 10 no conformidades en la primera iteración, de las cuales 8 fueron resueltas y 2 no procedían. En la segunda iteración arrojó un total de 6 no conformidades, de las cuales 4 fueron resueltas y 2 no procedían. En la tercera iteración se obtuvieron 6 no conformidades quedando las 6 resueltas; durante las 2 iteraciones no quedó ninguna no conformidad pendiente. La figura 11 muestra los resultados obtenidos durante las pruebas de aceptación.

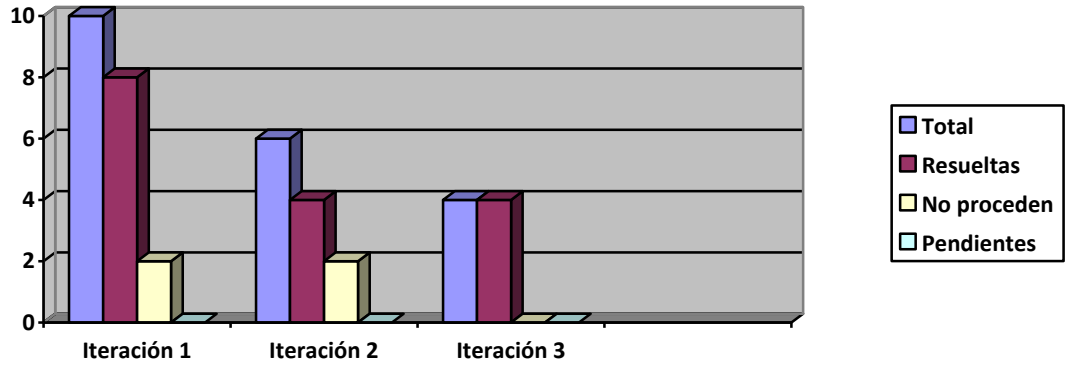


Figura 11: Resultados de las pruebas unitarias

Conclusiones del capítulo

En este capítulo se realizaron las pruebas unitarias y de aceptación propuestas por XP para detectar y eliminar los posibles errores en la implementación del sistema. Se eliminaron los errores durante la implementación, así como las no conformidades del cliente

CONCLUSIONES

- ✓ Se realizó un mantenimiento adaptativo para modificar el Módulo de Gestión de Auditoría Web a las nuevas pautas de diseño del Centro de Telemática.
- ✓ El estudio de las diferentes herramientas, tecnologías y metodología de desarrollo de software seleccionadas permitieron el correcto uso de las mismas en la implementación del módulo.
- ✓ Se realizó un estudio de los procesos relacionados con el Módulo de Gestión de Auditoría Web de Xilema – PlatSI v1.0 para una mejor comprensión de los requerimientos del cliente.
- ✓ Se llevaron a cabo las pruebas del sistema en el transcurso de la implementación del mismo para corregir los posibles errores y fallas.
- ✓ Se cumplió el objetivo general planteado en la investigación pues se adaptó el Módulo de Gestión de Auditoría Web de Xilema – PlatSI a las nuevas pautas de diseño de TLM.

RECOMENDACIONES

Una vez concluido el Módulo de Gestión de Auditoría Web de Xilema - PlatSI y haber cumplido con los objetivos propuestos al inicio de la investigación se recomienda:

- ✓ Agregar al sistema la capacidad de integrarse con las herramientas de detección de vulnerabilidades Nikto y Acunetix.
- ✓ Agregar al sistema la capacidad de conectarse con el distribuidor de tareas (Bambú).

REFERENCIAS

1. **Torres, Ramón.** *Maestría en Educación, Mención Informática y Diseño Instruccional, Metodologías Ágiles para el Desarrollo de Software eXtreme Programming (XP).* Universidad de los Andes, Facultad de Humanidades y Educación, Méridas, 2009.
2. **Glass, Robert L.** *Facts and Fallacies of Software Engineering.* 2003. ISBN: 0321117425.
3. **Hernández, Vismar Santos.** *La industria de software. Estudio a nivel global y América Latina.* La Habana, Cuba, 2004.
4. **Pressman, Roger S.** Capítulo 8 Modelado de análisis. *Ingeniería de software. Un enfoque práctico.* 2005.
5. **Ingeniería de software**. [En línea] Universidad Unión Boliviana, 13 de Febrero de 2015. http://Ingenieriadesoftware.mex.tt/52753_XP---Extreme-Programing.html.
6. **Navegapolis.** [En línea] 1 de Diciembre de 2014. http:/www./navegapolis.net/content/view/807/62.
7. **Buzz, S. Arquitectura de software**. [En línea] 2009. http://sq.com.mx/content/view/922.
8. **Cdlibre.** [En línea] http://www.cdlibre.org/boletin/boletines/boletin 410.html.
9. **Jeffries, R., Anderson, A. y Hendrickson, C.** ***C****. Extreme Programming Installed.* 2001.
10. **Sommerville, Ian.** *Ingeniería de software.* 2005. ISBN: 8478290745.
11. **Hipertextual.** [En línea] 9 de Diciembre de 2014. http://hipertextual.com/archivo/2014/06/pycharm-ide-python/.
12. **Gracia, Isabel y Marzal, Andrés.** *Introducción a la programación con python.* Universidad Jaime I. Castellón de la Plana, España, 2003.
13. **Infante, Sergio Montero.** Maestros del web. [En línea] 13 de Febrero de 2015. www.maestrosdelweb.com/curso-django-introduccion/.
14. **Mozilla Fundation. Mozilla.** [En línea] 7 de Febrero de 2015. https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5.
15. **PostgresSQL-es. Portal en español sobre PostgresSQL**. [En línea] [Citado el: 13 de Febrero de 2015.] http://www.postgresql.org.es/sobre postgresql.

16. **PgAdmin: PostgreSQL Tools.** [En línea] [Citado el: 24 de Abril de 2015.] http://www.pgadmin.org/.
17. **Concepto: Software architecture.** [En línea] [Citado el: 8 de Abril de 2015.] http://epf.eclipse.org.
18. **ALEGSA.** [En línea] [Citado el: 25 de Abril de 2015.] http://www.alegsa.com.ar/Respuesta/Ventajas y desventajas del modelo clienteservidor.html.
19. **Largman, Craig.** ***UML y Patrones****. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 2009.
20. **Ecured. Estrategia de pruebas.** [En línea] [Citado el: 25 de Mayo de 2015.] http://www.ecured.cu/index.php/Estrategia_de_Pruebas.
21. **Revisiones de código y estándares de codificación.** ***Microsoft Developer Network****.* [En línea] [Citado el: 8 de Junio de 2015.] http://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx.

BIBLIOGRAFÍA

1. **Hernández, Vlamar Santos.** *La industria de software. Estudio a nivel global y América Latina.* La Habana, Cuba, 2004.

2. **Glass, Robert L.** *Facts and Fallacies of Software Engineering.* 2003. ISBN: 0321117425.

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

4. **Navegapolis.** [En línea] 1 de Diciembre de 2014. http:/www./navegapolis.net/content/view/807/62.

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

6. Cdlibre. [En línea] http://www.cdlibre.org/boletin/boletines/boletin 410.html.

7. Jeffries, R., Anderson, A. y Hendrickson, C. ***C****. Extreme Programming Installed.* 2001.

8. Sommerville, Ian. *Ingeniería de software.* 2005. ISBN: 8478290745.

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

10. Gracia, Isabel y Marzal, Andrés. *Introducción a la programación con python.* Universidad Jaime I. Castellón de la Plana, España, 2003.

11. Torres, Ramón. *Maestría en Educación, Mención Informática y Diseño Instruccional, Metodologías Ágiles para el Desarrollo de Software eXtreme Programming (XP).* Universidad de los Andes, Facultad de Humanidades y Educación, Méridas, 2009.

12. Infante, Sergio Montero. Maestros del web. [En línea] 13 de Febrero de 2015. www.maestrosdelweb.com/curso-django-introduccion/.

13. Mozilla Fundation. Mozilla. [En línea] 7 de Febrero de 2015. https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5.

14. PostgresSQL-es. Portal en español sobre PostgresSQL. [En línea] [Citado el: 13 de Febrero de 2015.] http://www.postgresql.org.es/sobre postgresql.

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

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

17. Concepto: Software architecture. [En línea] [Citado el: 8 de Abril de 2015.] http://epf.eclipse.org.

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

19. Largman, Craig. ***UML y Patrones****. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 2009.

20. Ecured. Estrategia de pruebas. [En línea] [Citado el: 25 de Mayo de 2015.] http://www.ecured.cu/index.php/Estrategia_de_Pruebas.

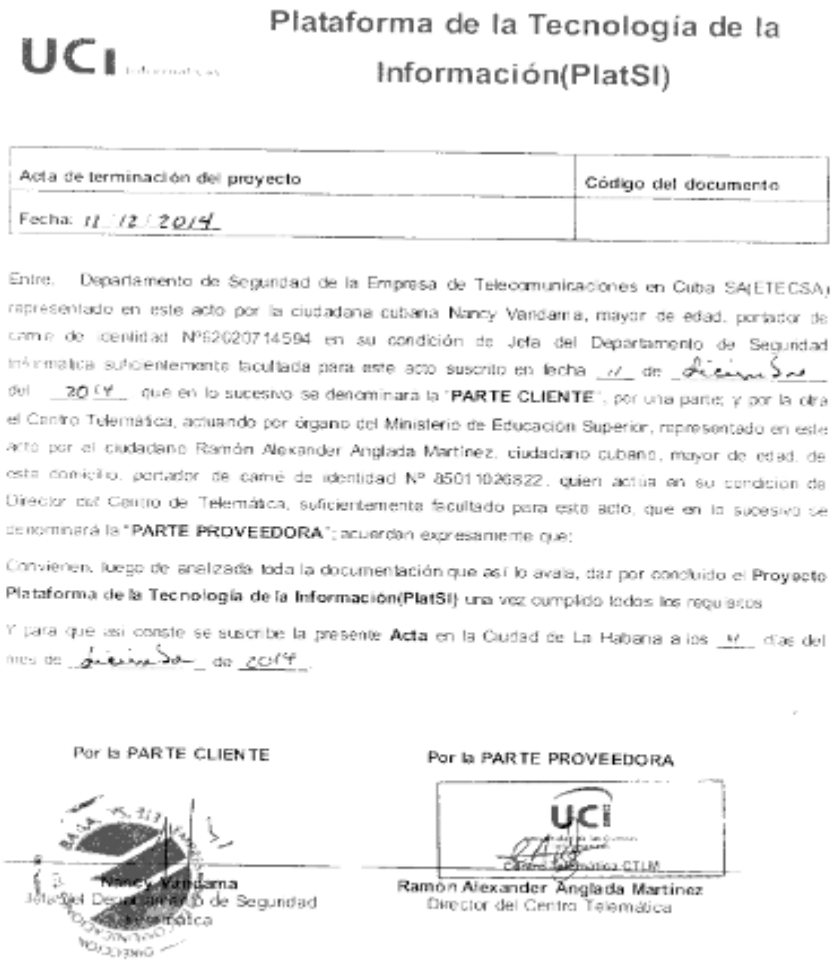
21. Desarrolloweb. [En línea] [Citado el: 10 de Diciembre de 2014.] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

22. Hernández, León, Coello, Rolando Alfredo y González, Zayda. *El proceso de investigación científica.* La Habana : Editorial Universitaria, 2011. ISBN: 978-959-16-1307-3.

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

ANEXOS

ANEXO I



UCI

Plataforma de la Tecnología de la Información(PlatSI)

Acta de terminación del proyecto	Código del documento
Fecha: 11/12/2014	

Entre: Departamento de Seguridad de la Empresa de Telecomunicaciones en Cuba SA(ETECSA) representado en este acto por la ciudadana cubana Nancy Vandama, mayor de edad, portador de carné de identidad Nº82020714594 en su condición de Jefa del Departamento de Seguridad Informática suficientemente facultada para este acto suscrito en fecha 11 de diciembre del 2014 que en lo sucesivo se denominara la "**PARTE CLIENTE**", por una parte; y por la otra el Centro Telemática, actuando por órgano del Ministerio de Educación Superior, representado en este acto por el ciudadano Ramón Alexander Anglada Martínez, ciudadano cubano, mayor de edad, de este domicilio, portador de carné de identidad Nº 85011926822, quien actúa en su condición de Director del Centro de Telemática, suficientemente facultado para este acto, que en lo sucesivo se denominará la "**PARTE PROVEEDORA**"; acuerdan expresamente que:

Convienen, luego de analizada toda la documentación que así lo avala, dar por concluido el **Proyecto Plataforma de la Tecnología de la Información(PlatSI)** una vez cumplido todos los requisitos

Y para que así conste se suscribe la presente **Acta** en la Ciudad de La Habana a los 11 días del mes de diciembre de 2014

Por la PARTE CLIENTE

Nancy Vandama
Jefa del Departamento de Seguridad Informática

Por la PARTE PROVEEDORA

UCI
Centro Telemática CTLM

Ramón Alexander Anglada Martínez
Director del Centro Telemática

Figura 10: Acta de terminación del software aprobado por Calidad

ANEXO II

Tabla 29: HU Gestionar vulnerabilidades

Historia de usuario	
Número: 5	**Usuarios:** Especialista.
Nombre de la Historia de usuario: Gestionar vulnerabilidades.	

Prioridad en el negocio: Alta	**Riesgo en Desarrollo:** Alta
Puntos estimados: 1	**Iteración Asignada:** 3
Programador responsable: Daniela González Begué y Hector V. Potete Muñoz	
Descripción: Permite agregar vulnerabilidades encontradas y crear nuevas vulnerabilidades. El sistema debe permitir la gestión de las vulnerabilidades, lo cual se refiere a la creación, asignación, modificación y eliminación de las mismas, así como listarlas, ver los detalles de cada una de ellas, realizarle una prueba de concepto y añadirlas al informe. Los campos asociados a dichas auditorías generales son: Nombre, Impacto, Tipo, Estado, Nivel y Detalles de la prueba de concepto.	
Observaciones: Para acceder al sistema se debe previamente autenticar, proporcionando usuario y contraseña.	

Tabla 30: HU Gestionar informe

Historia de usuario	
Número: 6	**Usuarios:** Responsable.
Nombre de la Historia de usuario: Gestionar informe.	
Prioridad en el negocio: Alta	**Riesgo en Desarrollo:** Bajo
Puntos estimados: 0.8	**Iteración Asignada:** 3
Programador responsable: Daniela González Begué y Hector V. Potete Muñoz	
Descripción: El sistema debe permitir la gestión de los informes, lo cual se refiere a la creación de informes técnicos o generales, así como listar las auditorías web a partir de las cuales se generan dichos informes. Los campos asociados a dichas auditorías web son: Nombre del producto, Host, Fecha de inicio y Fecha de fin.	
Observaciones: Para acceder al sistema se debe previamente autenticar, proporcionando usuario y contraseña. Para poder listar las auditorías web y generar informes a partir de ellas, estas deben haber sido cerradas por el especialista.	

Tabla 31: HU Gestionar usuarios y grupos

Historia de usuario	
Número: 1	**Usuarios:** Administrador.
Nombre de la Historia de usuario: Gestionar usuarios y grupos.	
Prioridad en el negocio: Alta	**Riesgo en Desarrollo:** Bajo
Puntos estimados: 0.4	**Iteración Asignada:** 1
Programador responsable: Daniela González Begué y Hector V. Potete Muñoz	
Descripción: El sistema debe permitir la gestión de usuarios y grupos, lo cual se refiere a la creación de creación, modificación y eliminación de usuarios y grupos, así como listarlos y cambiar la contraseña de los usuarios. También debe permitirse asignar permisos a los grupos existentes.	
Observaciones: Este módulo ya lo implementa Xilema – Base - Web.	

ANEXO III

Tabla 32: CRC Vista_Vulnerabilidades

Clase: Vista_Vulnerabilidades	
Responsabilidad	**Colaborador**
Permite mostrar las vulnerabilidades correspondientes a una auditoría web.	Solicitud Auditoría

Tabla 33: CRC Vista_Responsable

Clase: Vista_Responsable	
Responsabilidad	**Colaborador**
Permite mostrar los informes por auditorías.	Solicitud Auditoría Vulnerabilidades

Tabla 34: CRC Serializadora_Vulnerabilidad

Clase: VulnerabilidadSerializer	
Responsabilidad	**Colaborador**
Permite cargar datos de las vulnerabilidades para mostrarlos en una tabla posteriormente.	AuditoriaWeb

Tabla 35: CRC Serializadora_AuditoriaWebCRUD

Clase: Serializadora_AuditoriaWebCRUD	
Responsabilidad	**Colaborador**
Permite validar los campos mediante los cuales se crearán las auditorías web.	AuditoriaWeb

Tabla 36: CRC Serializadora_DetalleAuditoria

Clase: Serializadora_DetalleAuditoria	
Responsabilidad	**Colaborador**
Permite cargar todos los datos de una auditoría web (lo cual incluye los datos de su solicitud, su auditoría, auditoría web y las vulnerabilidades asociadas a ella).	Solicitud Auditoria AuditoriaWeb Vulnerabilidades

Tabla 37: Tarea de ingeniería: Crear usuario

Tarea de Ingeniería	
Número tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Crear usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 15/04/2015	**Fecha de fin:** 15/04/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para adicionar un usuario. Una vez	

que el administrador selecciona la opción Crear usuario, aparecerá un formulario con los campos necesarios para la creación del mismo.

Tabla 38: Tarea de ingeniería: Crear grupos

Tarea de Ingeniería	
Número tarea: 2	**Número Historia de Usuario:** 1
Nombre Tarea: Crear grupos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 16/04/2015	**Fecha de fin:** 16/04/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para adicionar un grupo. Una vez que el administrador selecciona la opción Crear grupo, aparecerá un formulario con los campos necesarios para la creación del mismo.	

Tabla 39: Tarea de ingeniería: Crear solicitud

Tarea de Ingeniería	
Número tarea: 3	**Número Historia de Usuario:** 2
Nombre Tarea: Crear solicitud.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 17/04/2015	**Fecha de fin:** 17/04/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para la adición de una solicitud. Una vez que el solicitante selecciona la opción Crear Solicitud, se mostrará un formulario mostrando los campos a llenar, correspondientes a la solicitud.	

Tabla 40: Tarea de ingeniería: Mostrar lista de solicitudes

Tarea de Ingeniería

Número tarea: 4	**Número Historia de Usuario:** 2
Nombre Tarea: Mostrar lista de solicitudes.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 20/04/2015	**Fecha de fin:** 20/04/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para mostrar una lista de las solicitudes existentes. Una vez que el solicitante selecciona el menú Solicitud, podrá ver una tabla que refleja todas las solicitudes y los datos correspondientes a cada una de ellas.	

Tabla 41: Tarea de ingeniería: Editar solicitud

Tarea de Ingeniería	
Número tarea: 5	**Número Historia de Usuario:** 2
Nombre Tarea: Editar solicitud.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 21/04/2015	**Fecha de fin:** 21/04/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para editar una solicitud. Una vez que el solicitante selecciona la opción Editar, aparecerá un formulario mediante el cual podrá cambiar los valores de los datos de la solicitud en cuestión.	

Tabla 42: Tarea de ingeniería: Mostrar detalles de solicitud

Tarea de Ingeniería	
Número tarea: 6	**Número Historia de Usuario:** 2
Nombre Tarea: Mostrar detalles de solicitud.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 22/04/2015	**Fecha de fin:** 22/04/2015

Programador responsable: Daniela González Begué - Hector V. Potete
Descripción: Esta tarea facilita la creación de una interfaz para mostrar los detalles de la solicitud. Una vez que el solicitante selecciona la opción Ver detalles, aparecerá un formulario con los detalles de la solicitud en cuestión.

Tabla 43: Tarea de ingeniería: Eliminar solicitud

Tarea de Ingeniería	
Número tarea: 7	**Número Historia de Usuario:** 2
Nombre Tarea: Eliminar solicitud.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 23/04/2015	**Fecha de fin:** 23/04/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la posibilidad de eliminar una solicitud. Una vez que el solicitante selecciona la opción Eliminar, se mostrará un mensaje de confirmación de eliminación.	

Tabla 44: Tarea de ingeniería: Mostrar lista de asignaciones

Tarea de Ingeniería	
Número tarea: 13	**Número Historia de Usuario:** 4
Nombre Tarea: Mostrar lista de asignaciones.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 04/05/2015	**Fecha de fin:** 04/05/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para mostrar una lista de las asignaciones hechas al especialista autenticado. Una vez que el especialista selecciona el menú Asignaciones, podrá ver una tabla que refleja todas las auditorías generales que tiene asignadas y los datos correspondientes a cada una de ellas.	

Tabla 45: Tarea de ingeniería: Mostrar detalles de la asignación

Tarea de Ingeniería	
Número tarea: 14	**Número Historia de Usuario:** 4
Nombre Tarea: Mostrar detalles de la asignación.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 05/05/2015	**Fecha de fin:** 05/05/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para mostrar los detalles de las auditorías generales asignadas. Una vez que el especialista selecciona la opción Ver detalles, aparecerá un formulario con los detalles de la auditoría general en cuestión.	

Tabla 46: Tarea de ingeniería: Cambiar estado de las asignaciones

Tarea de Ingeniería	
Número tarea: 15	**Número Historia de Usuario:** 4
Nombre Tarea: Cambiar estado de las asignaciones.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 06/05/2015	**Fecha de fin:** 06/05/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para cambiar el estado de la auditoría general. Una vez que el especialista selecciona la opción Cambiar estado, se mostrará un formulario donde se podrán seleccionar los estados posibles para la auditoría general.	

Tabla 47: Tarea de ingeniería: Crear auditoría web

Tarea de Ingeniería	
Número tarea: 16	**Número Historia de Usuario:** 4
Nombre Tarea: Crear auditoría web.	

Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 07/05/2015	**Fecha de fin:** 07/05/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para la adición de una auditoría web. Una vez que el especialista selecciona la opción Crear Auditoría Web, se mostrará un formulario mostrando los campos a llenar, correspondientes a la auditoría web.	

Tabla 48: Tarea de ingeniería: Mostrar lista de auditorías web

Tarea de Ingeniería	
Número tarea: 17	**Número Historia de Usuario:** 4
Nombre Tarea: Mostrar lista de auditorías web.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 8/05/2015	**Fecha de fin:** 8/05/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para mostrar una lista de las auditorías web. Una vez que el especialista selecciona el menú Auditoría web, podrá ver una tabla que refleja todas las auditorías web creadas por él y los datos correspondientes a cada una de ellas.	

Tabla 49: Tarea de ingeniería: Editar auditoría web

Tarea de Ingeniería	
Número tarea: 18	**Número Historia de Usuario:** 4
Nombre Tarea: Editar auditoría web.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 11/05/2015	**Fecha de fin:** 11/05/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para editar una auditoría web. Una vez	

que el especialista selecciona la opción Editar, aparecerá un formulario mediante el cual podrá cambiar los valores de los datos de auditoría web en cuestión.

Tabla 50: Tarea de ingeniería: Mostrar detalles de auditoría web

Tarea de Ingeniería	
Número tarea: 19	**Número Historia de Usuario:** 4
Nombre Tarea: Mostrar detalles de auditoría web.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 12/05/2015	**Fecha de fin:** 12/05/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para mostrar los detalles de las auditorías web. Una vez que el especialista selecciona la opción Ver detalles, aparecerá un formulario con los detalles de la auditoría web en cuestión.	

Tabla 51: Tarea de ingeniería: Mostrar datos de la auditoría web

Tarea de Ingeniería	
Número tarea: 20	**Número Historia de Usuario:** 4
Nombre Tarea: Mostrar datos de la auditoría web.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 13/05/2015	**Fecha de fin:** 13/05/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para mostrar los datos de la auditoría web. Una vez que el especialista selecciona la opción Mostrar datos de la auditoría web, se muestra un formulario que contiene todos los datos de la auditoría, incluyendo la opción de generar un documento PDF con dichos datos.	

Tabla 52: Tarea de ingeniería: Terminar auditoría web

Tarea de Ingeniería

Número tarea: 21	**Número Historia de Usuario:** 4
Nombre Tarea: Terminar auditoría web.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 14/05/2015	**Fecha de fin:** 14/05/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para terminar una auditoría web. Una vez que el especialista selecciona la opción Terminar auditoría web, se mostrará un mensaje de confirmación especificando a su vez cuales son los requisitos para terminar una auditoría web.	

Tabla 53: Tarea de ingeniería: Crear vulnerabilidad

Tarea de Ingeniería	
Número tarea: 22	**Número Historia de Usuario:** 5
Nombre Tarea: Crear vulnerabilidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 18/05/2015	**Fecha de fin:** 18/05/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para la adición manual de una vulnerabilidad. Una vez que el especialista selecciona la opción Crear vulnerabilidad, se mostrará un formulario mostrando los campos a llenar, correspondientes a la misma.	

Tabla 54: Tarea de ingeniería: Editar vulnerabilidad

Tarea de Ingeniería	
Número tarea: 23	**Número Historia de Usuario:** 5
Nombre Tarea: Editar vulnerabilidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 19/05/2015	**Fecha de fin:** 19/05/2015

Programador responsable: Daniela González Begué - Hector V. Potete
Descripción: Esta tarea facilita la creación de una interfaz para editar una vulnerabilidad. Una vez que el especialista selecciona la opción Editar, aparecerá un formulario mediante el cual podrá cambiar los valores de los datos de vulnerabilidad en cuestión, así como cambiar su estado a terminada.

Tabla 55: Tarea de ingeniería: Realizar prueba de concepto

Tarea de Ingeniería	
Número tarea: 24	**Número Historia de Usuario:** 5
Nombre Tarea: Realizar prueba de concepto.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 20/05/2015	**Fecha de fin:** 20/05/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para realizar una prueba de concepto a una vulnerabilidad. Una vez que el especialista selecciona la opción Realizar Prueba de Concepto, se mostrará una interfaz donde deberá llenar los campos correspondientes a la misma, donde se incluye subir una imagen que describe el lugar donde se encontró dicha vulnerabilidad.	

Tabla 56: Tarea de ingeniería: Mostrar detalles de la vulnerabilidad

Tarea de Ingeniería	
Número tarea: 25	**Número Historia de Usuario:** 5
Nombre Tarea: Mostrar detalles de la vulnerabilidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 21/05/2015	**Fecha de fin:** 21/05/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para mostrar los detalles de las vulnerabilidades. Una vez que el especialista selecciona la opción Ver detalles, aparecerá un	

formulario con los detalles de la vulnerabilidad en cuestión.

Tabla 57: Tarea de ingeniería: Eliminar vulnerabilidad

Tarea de Ingeniería	
Número tarea: 26	**Número Historia de Usuario:** 5
Nombre Tarea: Eliminar vulnerabilidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 22/05/2015	**Fecha de fin:** 22/05/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la posibilidad de eliminar una vulnerabilidad. Una vez que el especialista selecciona la opción Eliminar, se mostrará un mensaje de confirmación de eliminación.	

Tabla 58: Tarea de ingeniería: Mostrar lista de auditorías web cerradas

Tarea de Ingeniería	
Número tarea: 27	**Número Historia de Usuario:** 6
Nombre Tarea: Mostrar lista de auditorías web cerradas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 25/05/2015	**Fecha de fin:** 25/05/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para mostrar las auditorías web cerradas. Una vez que el responsable selecciona el menú Informe, se mostrará una tabla con todas las auditorías web en estado de cerradas.	

Tabla 59: Tarea de ingeniería: Mostrar lista de auditorías web asignadas

Tarea de Ingeniería	
Número tarea: 28	**Número Historia de Usuario:** 6

Nombre Tarea: Mostrar lista de auditorías web asignadas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 26/05/2015	**Fecha de fin:** 26/05/2015
Programador responsable: Daniela González Begué – Hector V. Potete	
Descripción: Esta tarea facilita la creación de una interfaz para mostrar las auditorías web asignadas al responsable. Una vez que el responsable selecciona el menú Asignaciones, podrá ver una tabla con todas las auditorías web que le fueron asignadas independientemente de si las mismas están o no en estado de terminadas.	

Tabla 60: Tarea de ingeniería: Crear informe técnico

Tarea de Ingeniería	
Número tarea: 29	**Número Historia de Usuario:** 6
Nombre Tarea: Crear informe técnico.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 27/05/2015	**Fecha de fin:** 27/05/2015
Programador responsable: Daniela González Begué - Hector V. Potete	
Descripción: Esta tarea facilita la posibilidad de seleccionar la opción Generar informe técnico. Una vez que el responsable haya seleccionada dicha opción el sistema generará un documento PDF donde lista todas las auditorías web cerradas y las vulnerabilidades correspondientes a cada una de las mismas.	

Tabla 61: Tarea de ingeniería: Crear informe general

Tarea de Ingeniería	
Número tarea: 30	**Número Historia de Usuario:** 6
Nombre Tarea: Crear informe general.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 28/05/2015	**Fecha de fin:** 28/05/2015

Programador responsable: Daniela González Begué - Hector V. Potete
Descripción: Esta tarea facilita la posibilidad de seleccionar la opción Generar informe general. Una vez que el responsable haya seleccionada dicha opción, se mostrará un formulario para añadir datos correspondientes a informe y a partir de ellos el sistema generará un documento PDF donde muestra dichos datos y lista todas las auditorías web cerradas y las vulnerabilidades correspondientes a cada una de las mismas.

ANEXO IV

Tabla 62: Caso de Prueba: Gestionar auditoría web

	Clases Válidas	**Clases Inválidas**
Datos	**Paso 1:** El especialista selecciona en la tabla la auditoría general a la cual desea crear una auditoría web. **Paso 2:** Selecciona la opción Crear auditoría web. **Paso 3:** Introduce los datos de la auditoría web. **Ejemplo:** **Nombre del producto:** ProductoX **Host:** http://10.8.2.3 **Fecha de inicio:** 2015-05-30 17:35:27 **Fecha de fin:** 2015-05-30 17:35:28	**Paso 1:** El especialista selecciona en la tabla la auditoría general a la cual desea crear una auditoría web. **Paso 2:** Selecciona la opción Crear auditoría web. **Paso 3:** Introduce los datos de la auditoría web. **Ejemplo:** **Nombre del producto:** Producto*/& **Host:** http://10.8.2.3.45 **Fecha de inicio:** 2015-06-30 17:35:27 **Fecha de fin:**

	Descripción: Auditoría web para ProductoX **Paso 4:** Pulsa el botón de aceptar.	2015-05-30 17:35:28 **Descripción:** Auditoría web para ProductoX **Paso 4:** Pulsa el botón de aceptar
Resultado esperado	El sistema adiciona una auditoría web a la base de datos mostrándola al especialista en una tabla.	El sistema muestra un mensaje indicando que el campo Auditoría web sólo puede contener letras y números, otro mensaje indicando que la fecha de inicio no puede ser posterior a la fecha de fin y otro indicando que en el campo Host debe introducirse una dirección IP válida.
Resultado de la prueba	Satisfactorio	Satisfactorio
Observaciones		

Tabla 63: Caso de Prueba: Gestionar vulnerabilidades

	Clases Válidas	**Clases Inválidas**
Datos	**Paso 1:** El especialista selecciona la opción de Adicionar vulnerabilidad. **Paso 2:** Introduce los datos de la solicitud. **Ejemplo:** **Nombre:** VulnerabillidadX **Descripción:**	**Paso 1:** El especialista selecciona la opción de Adicionar vulnerabilidad. **Paso 2:** Introduce los datos de la solicitud. **Ejemplo:** **Nombre:** Vulnerabillidad%”! **Descripción:**

	Fallo de seguridad. **Impacto:** Impacto negativo **Tipo:** Configuración. **Estado:** Terminado **Nivel:** Alto **Paso 3:** Pulsa el botón de aceptar	Fallo de seguridad **Impacto:** Impacto negativo **Tipo:** Configuración. **Estado:** Terminado **Nivel:** Alto **Paso 3:** Pulsa el botón de aceptar.
Resultado esperado	El sistema adiciona una vulnerabilidad a la base de datos mostrándola al especialista en una tabla.	El sistema muestra un mensaje indicando que el campo Nombre sólo puede contener letras y números.
Resultado de la prueba	Satisfactorio	Satisfactorio
Observaciones		

Tabla 64: Caso de prueba Gestionar Informe

	Clases Válidas	**Clases Inválidas**
Datos	**Paso 1:** El responsable selecciona en la tabla una auditoría web y luego la opción de Crear informe general. **Paso 2:** Introduce los datos del informe general. **Ejemplo:** **Introducción:** Auditoría a	No existen clases inválidas.

	aplicación web x. **Estrategia:** Estrategia para solucionar vulnerabilidades. **Recomendaciones:** Se recomienda utilizar más de una herramienta. **Evaluación:** Alta. **Paso 3:** Pulsa el botón de aceptar	
Resultado esperado	El sistema genera un documento PDF, mostrando las opciones de abrirlo y guardarlo	
Resultado de la prueba	Satisfactorio	Satisfactorio
Observaciones		