Herramienta Colaborativa para la Realización de Auditorías a Sistemas Gestores de Base de Datos (RASGBD) V2.0

Autores: Claudia Vazquez Figueroa

Enrique Muschett Cortina

Tutores: Ing. Lianet Salazar Labrada

Ing. Luis Eduardo Gallardo Concepción

Junio 2015

Declaración de autoría

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 2 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ___ días del mes de _____ del año _____.

Claudia Vazquez Figueroa

_____

Firma del autor

Enrique Muschett Cortina

_____

Firma del autor

Ing. Lianet Salazar Labrada

_____

Firma del tutor

Ing. Luis Eduardo Gallardo

_____

Firma del tutor

Datos del contacto

DATOS DEL CONTACTO

Datos del autor

Claudia Vazquez Figueroa

Universidad de las Ciencias Informáticas

Email: cvfigueroa@estudiantes.uci.cu

Datos del autor

Enrique Muschett Cortina

Universidad de las Ciencias Informáticas

Email: emuschett@estudiantes.uci.cu

Datos del Tutor

Ing. Lianet Salazar Labrada, graduado de Ingeniero en Ciencias Informáticas en el año 2012.

Pertenece al área de TLM, Dpto. Telecomunicaciones.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: lslabrada@uci.cu

Datos del Tutor

Ing. Luis E Gallardo Concepción, graduado de Ingeniero en Ciencias Informáticas en el año 2014. Pertenece al área de TLM, Dpto. Telecomunicaciones.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: legallardo@uci.cu

DEDICATORIA

Enrique

A mi mamá Diana, a mi papá Kikito, a mi abuela Bella y a mi abuela Orfelina. Sé que el título de ingeniero los va a hacer a ellos más felices que a mí.

Claudia

Dedico mi tesis a mi mamá y mi papá por ser el centro de mi vida y estar siempre pendientes de mí, por acompañarme en las buenas y las malas, por traerme al mundo y cuidarme tan bien. A mi familia y amigos que me han hecho ser una mejor persona. A todo aquel que aportó un granito de arena para que yo pudiera terminar mi carrera.

AGRADECIMIENTOS

Enrique

A mis padres, los principales responsables de que todo esto haya ocurrido.

A toda mi familia que es el motor que me impulsa cuando me faltan ganas para continuar.

A mis compañeros de grupo que estuvieron conmigo en una travesía que duró 5 años en las malas y en las buenas, de todos ellos me llevo la mejor impresión y un recuerdo que nunca podrá borrarse.

A mis más cercanos amigos: El Blade, Luis, Ronal, Yassel, Erisel, con ellos aprendí a valorar más la amistad.

A mi hermano Osmar que fue el primer y más grande amigo que hice en la universidad.

A todas mis amistades, tanto de la universidad como de mi tierra natal.

A mis tutores por ayudarnos cuando nos hizo falta, fundamentalmente en la recta final.

A todos muchas gracias…

Claudia

A Dios por permitirme llegar hasta donde estoy, por darme la fortaleza y por poner en mi camino personas tan maravillosas.

A mi viejita por estar siempre tan pendiente de mi aconsejándome en los momentos más difíciles.

A mi papá por estar a mi lado siempre aún sin entender nada de informática.

A mis segundos padres tía nenita y tío pepe por quererme y cuidarme tanto.

Al oriental payaso Yunior por sus consejos siempre oportunos.

A perfu por quererme como una hija.

A mis hermanos, mis primos, abuela Nena por la fuerza que me dieron y los buenos consejos.

A mis tíos, a chicha por quererme como su hija, tía Magda por confiar siempre en mí, tía Yaya, tía Maruja, tía Anaivis, tía Nany.

A mi Tuti pequeñito y su familia por tenerme siempre presente y ayudarme tanto en estos 5 años.

A mis amistades de la Fiscalía Municipal de Bahía Honda por los consejos y el cariño que me brindaron, especialmente los perros Yaidel, Leidys y Ricardito.

A mis compañeritos de aula en especial: a los mipos, mupi, mipi, felix, ronital, cary, rolo, luis, las borrachitas, muchessito, yoel, exxon, yisel, en fin a todos por hacerme reir tanto.

A mis tutores, especialmente a Lianet por tenerme tanta paciencia. A Fernando y Osbelito.

A mis amiguis Arislenys y Yoli por sus llamadas de aliento y alegría. Al Nani y Edel por poder contar siempre con su ayuda y apoyo.

A los guaraperos, a Angelinito y el Jeen por los cafecitos con espumita.

RESUMEN

La auditoría a bases de datos es un proceso que permite monitorear los accesos a la información almacenada, así como registrar el comportamiento de los datos usados por los usuarios. En el Centro Telemática (TLM) de la Universidad de las Ciencias Informáticas (UCI) se creó la Herramienta Colaborativa para la Realización de Auditorías a Sistemas Gestores de Bases de Datos (RASGBD). Esta herramienta portable es utilizada por la Empresa de Telecomunicaciones de Cuba S.A (ETECSA).

Actualmente RASGBD posee algunas limitaciones debido a que no ejecuta correctamente las guías de revisión y no se conecta a los gestores de bases de datos Oracle en sus versiones 8i, 9i, 10g y 11g. Además no cumple con la estandarización que se lleva a cabo en el centro TLM. Para la realización de este trabajo se utilizó Extreme Programming (XP) como metodología de desarrollo de software, Python como lenguaje de programación y el framework Django. Para la modelación de los procesos de negocio se empleó BPMN utilizando Visual Paradigm como herramienta CASE.

Como resultado del trabajo se obtuvo una nueva versión de la Herramienta Colaborativa para la Realización de Auditorías a Sistemas Gestores de Base de Datos (RASGBD) que cumple con la estandarización tecnológica del Centro TLM. Esta nueva versión permite además ejecutar correctamente las guías de revisión y conectarse a diferentes versiones de Oracle.

PALABRAS CLAVE: auditoría, estandarización, gestores de bases de datos, guías de revisión, Oracle.

Índice de contenido

ÍNDICE DE TABLAS

ÍNDICE DE IMÁGENES

INTRODUCCIÓN

La acelerada evolución tecnológica ha hecho que las Tecnologías de la Informática y las Comunicaciones (TIC) faciliten la vida cotidiana y profesional ya que permiten aprovechar las posibilidades didácticas que estas ofrecen. Los ordenadores y todo lo que los rodea han logrado convertirse en el nuevo paradigma, además de ser el centro y la base de todas las operaciones importantes de la sociedad. Asimismo, la dependencia del uso de las TIC, se hace más vulnerable a los fallos que se producen en el sistema, ya sea por mal funcionamiento de los ordenadores o por uso indebido por parte de los encargados de manejarlos. Esto sugiere la necesidad de tener mayor conciencia de los aspectos de seguridad, que consiste en asegurar que los recursos del sistema de una organización sean utilizados de la manera que se decidió. Además que el acceso a la información allí contenida, así como su modificación, sólo sea posible por las personas que se encuentren acreditadas y dentro de los límites de su autorización.

Al igual que cualquier área de la organización, los sistemas de Tecnologías de la Información (TI) deben estar sometidos a controles de calidad y auditoría informática ya que las computadoras y los centros de procesamiento de datos son blancos codiciables para el espionaje. Entre los principales objetivos del Departamento de Seguridad Informática (DSI) de la Empresa de Telecomunicaciones de Cuba S.A. (ETECSA) está mantener y garantizar la integridad de los Sistemas Operativos (SO), aplicaciones web y Sistemas Gestores de Bases de Datos (SGBD) que soportan el trabajo de las telecomunicaciones en Cuba.

Uno de los mayores desafíos del país es la necesidad de perfeccionar un modelo de gestión que vincule centros de investigación con universidades y entidades productivas. La Universidad de las Ciencias Informáticas (UCI) es una universidad productiva, cuya misión es producir aplicaciones y servicios informáticos a partir de la vinculación estudio – trabajo como modelo de formación, y servir de soporte a la industria cubana de la informática. En la misma existen varios centros orientados a diversos sectores en los cuales se desarrollan productos de software. Uno de ellos, el Centro Telemática (TLM), realiza aplicaciones que tienen que ver con la seguridad informática y los servicios telemáticos. A solicitud de la empresa ETECSA este crea un proyecto llamado AuditBD cuyo objetivo principal es la gestión de auditorías a SGBD y SO. AuditBD implementa un sistema que genera dinámicamente los script a partir de diferentes gestores de bases de datos y provee reportes con los resultados obtenidos en las auditorías realizadas.

En este proyecto se desarrolló una aplicación para la gestión de auditorías que lleva por nombre SASGBD. Además de la aplicación general que contiene dos módulos, SASGBD tiene desarrolladas dos aplicaciones externas e independientes. La primera es una herramienta para la Gestión de Auditorías a Sistemas Gestores de Bases de Datos (GASGBD) que se encarga de

la creación de los plugins que son cargados por los módulos. Permite introducir cada una de las consultas y comandos propios del gestor referente al plugin y generar cada uno los ficheros XML correspondientes, da la posibilidad de implementar la interfaz definida por el módulo de BD para que pueda ser cargado por él. La otra es una herramienta portable y multiplataforma para la realización de las auditorías. Esta última llamada RASGBD se encarga de la ejecución de guías de revisión usadas para la comprobación de buenas prácticas de seguridad que se asocian a las distintas versiones de sistemas gestores de base de datos, como son PostgreSQL 8.1, 9.1, MySQL 5.0, Microsoft SQL Server 200, 2005 y Oracle 8i, 9i, 10g y 11g.

RASGBD fue desarrollada con los requerimientos tecnológicos solicitados por el cliente inicial, usando Java como lenguaje de programación y el framework Spring. La herramienta ya fue probada, liberada y es utilizada en ETECSA. Actualmente en el centro TLM se realiza un proceso de estandarización de sus productos en el que se define a Python como lenguaje de programación y a Django como framework asociado. Dicha estandarización facilita la gestión del conocimiento de los especialistas y adiestrados, además posibilita la eficacia en el uso de los recursos humanos vinculados a los proyectos de desarrollo en relación a las tareas de implementación, pruebas y soporte. No se puede dar soporte a la herramienta existente ya que en el Centro no se cuenta con gran parte del personal que dio inicio a la solución y la capacitación recibida por los adiestrados está enfocada a la estandarización que se lleva a cabo. Además en busca de nuevas oportunidades de trabajo se pretende seguir una línea única en sus productos para poder comercializar con otros clientes. Por otra parte existen limitaciones al ejecutar los script de revisión a los gestores PostgreSQL 8.1, 9.1, MySQL 5.0, Microsoft SQL Server 200, 2005 y no se ha podido realizar la conexión para ejecutar las auditorías a los gestores de Oracle.

En base a la problemática anterior se plantea como **problema de la investigación**: ¿Cómo contribuir a la realización de las auditorías en RASGBD v2.0 de forma que se cumpla con la estandarización tecnológica del centro TLM? Donde el **objeto de estudio** es: Las Auditorías a Sistemas Gestores de Bases de Datos. Con el **objetivo general** de: Desarrollar la Herramienta Colaborativa para la Realización de Auditorías a Sistemas Gestores de Base de Datos (RASGBD) v2.0 para satisfacer la estandarización tecnológica del centro TLM. Siendo el **campo de acción:** Las Auditorías a Sistemas Gestores de Bases de Datos en RASGBD v1.0.

Idea a defender: La implementación de una nueva versión de la herramienta RASGBD contribuye a la realización de auditorías usando la estandarización tecnológica del centro Telemática.

Los objetivos específicos que se definieron son los siguientes**:**

1. Fundamentar la investigación a partir de la definición del marco conceptual alrededor del

objeto de estudio y el estudio de sistemas similares en el contexto nacional e internacional.

2. Desarrollar la versión 2.0 de la herramienta RASGBD de forma que se satisfagan los requerimientos tecnológicos definidos en el Centro de Telemática.
3. Diseñar la estrategia de pruebas para la verificación y validación de las funcionalidades definidas para el sistema.

Para dar cumplimiento al objetivo general se plantean las siguientes **tareas de la investigación:**

- Estudio y selección de las herramientas, metodología y tecnologías necesarias para el desarrollo de RASGBD v2.0.
- Estudio y revisión de la documentación referente a RASGBD v1.0 para integrarlo a la arquitectura del centro TLM.
- Estudio y caracterización del proceso de negocio que se realiza en la herramienta RASGBD v1.0 para conocer sus funcionalidades.
- Análisis e identificación de requisitos para describir las características y restricciones que debe cumplir la herramienta RASGBD v2.0.
- Estudio y análisis de patrones de diseño con el fin de elaborar un diseño robusto y flexible.
- Estudio de los diferentes tipos de pruebas para verificar el correcto funcionamiento de la herramienta desarrollada.

Para apoyar el desarrollo de las tareas se emplearon los siguientes **métodos científicos:**

Métodos teóricos:

La modelación: este método se utiliza para modelar el proceso de negocio de la herramienta RASGBD v2.0 y los artefactos necesarios para su desarrollo.

Analítico-Sintético: es utilizado para resumir y entender la documentación referente a RASGBD v1.0, proporcionando la obtención de las características significativas de esta herramienta.

Análisis documental: este método es usado en el análisis de la bibliografía consultada. Además es empleado para analizar los conceptos asociados a la investigación.

Métodos empíricos:

Entrevista: este método se utilizará para realizar una serie de entrevistas a los integrantes del proyecto AuditBD para obtener la información deseada.

Observación: con el empleo de este método se conoce en la práctica cómo es el funcionamiento de RASGBD v1.0, lo que ayudará a identificar sus principales carencias.

Este trabajo de diploma está formado por 3 capítulos, los cuales tendrán la siguiente estructura:

Capítulo 1: ´´Fundamentación teórica de la investigación´´: Se lleva a cabo un estudio de los

principales conceptos y características relacionados con el proceso de auditorías a gestores de bases de datos. Además se hace un análisis de la herramienta RASGBD v1.0 y se explican las herramientas y metodología seleccionadas para realizar la implementación de la nueva versión.

Capítulo 2: ´´Exploración y planificación de la Herramienta Colaborativa para la Realización de Auditorías a Sistemas Gestores de Bases de Datos´´: Describe las características de la solución propuesta, definidas a partir de la modelación de los procesos de negocio y la especificación de los requisitos de software. Se definen los elementos del diseño de la arquitectura del sistema y los patrones de diseño utilizados.

Capítulo 3: ´´Diseño, implementación y pruebas´´: Se realiza el diseño e implementación de las tareas de ingeniería asociadas a las historias de usuario. Además se materializa la propuesta del sistema lo cual da cumplimiento a los objetivos trazados con la investigación y la validación del mismo a través de las pruebas realizadas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA DE LA INVESTIGACIÓN

1.1 Introducción

En este capítulo se presenta un análisis de la Herramienta Colaborativa para la Realización de Auditorías a Sistemas Gestores de Base de Datos (RASGBD) que existe actualmente. Igualmente se evidenciarán las razones por la cual se desarrollará una nueva versión. Además se especificarán las tecnologías y herramientas que se utilizarán para desarrollar la versión 2.0 de la misma. También se detallarán las principales características de la metodología que guiará el proceso del desarrollo de RASGBD v2.0.

1.2 Conceptos asociados a la investigación

1.2.1 Auditorías informáticas

La auditoría informática es la disciplina incluida en el campo de la auditoría que se refiere al análisis de las condiciones de una instalación informática por un auditor externo e independiente que realiza un dictamen sobre diferentes aspectos. Es un conjunto de procedimientos y técnicas para evaluar y controlar, total o parcialmente, un sistema informático, con el fin de proteger sus activos y recursos, verificar si sus actividades se desarrollan eficientemente y de acuerdo con la normativa informática y general existentes en cada empresa para conseguir la eficacia exigida en el marco de la organización correspondiente (1). Comprende no sólo la evaluación de los equipos de cómputo, de un sistema o procedimiento específico, sino que además evalúa los sistemas de información en general desde sus entradas, procedimientos, controles, archivos, seguridad y obtención de información (2).

1.2.2 Objetivos de la auditoría informática

Las auditorías informáticas son de vital importancia para el buen desempeño de los sistemas de información, ya que proporcionan los controles necesarios para que los sistemas sean confiables y con un buen nivel de seguridad. Además deben evaluar todo, informática, organización de centros de información, hardware y software. Tienen como objetivo fundamental mejorar la rentabilidad, la seguridad y la eficacia del sistema mecanizado de información en que se sustenta.

Las auditorías informáticas persiguen los siguientes objetivos:

-El control de la función informática.

-El análisis de la eficiencia de los sistemas informáticos.

-La verificación del cumplimiento de la normativa[1] en este ámbito.

-La revisión de la eficaz gestión de los recursos informáticos (3).

1.2.2.1Tipos de auditorías informáticas

Existen varias clasificaciones de auditorías a sistemas informáticos, el tipo que se utiliza en RASGBD v 1.0 es la auditoría a bases de datos y se va a continuar utilizando en la versión 2.0 pues ese es el tipo que se ajusta al objetivo de la herramienta que va a realizarse. A continuación se explica en qué consiste esta auditoría.

-Auditoría de Base de Datos: se encarga de monitorear, medir, asegurar y registrar los accesos a toda la información almacenada en las bases de datos. Esta auditoría de manera fundamental se basa en la seguridad de las bases de datos. Entre sus objetivos se encuentra imposibilitar el acceso interno a los usuarios no autorizados, evitar el acceso externo y autorizar el acceso solo a los usuarios autorizados. Con esta se busca monitorear y mantener un registro del uso de los datos por los usuarios autorizados y no autorizados. Además conservar las trazas de uso y permitir investigaciones, así como obtener alertas en tiempo real (4).

1.3 Estudio de la herramienta RASGBD v1.0

1.3.1 RASGBD V1.0

La Herramienta Colaborativa para la Realización de Auditorías a Sistemas Gestores de Bases de Datos (RASGBD) está destinada a apoyar el trabajo que realizan los auditores de la Empresa de Telecomunicaciones de Cuba para inspeccionar las múltiples bases de datos en busca de imperfecciones, errores o valores ilegibles en ellas. Esta herramienta se enfoca en la ejecución de los script de parámetros, con el objetivo de identificar las vulnerabilidades existentes en las distintas versiones de sistemas gestores de bases de datos PostgreSQL 8.1, 9.1, MySQL 5.0 y Oracle 8i, 9i, 10g y 11g.

La aplicación cuenta con una base de datos PostgreSQL, para el almacenamiento de las consultas, comandos, información de los usuarios con privilegios e informes de las auditorías. Es capaz de cargar las consultas y comandos a ejecutar en el servidor para auditar un gestor de bases de datos. Se conecta de manera remota al propio servidor, ejecuta cada una de las consultas y comandos, además obtiene los resultados. Luego carga esos resultados y los estructura en un fichero XML para su posterior análisis por parte del módulo de bases de datos.

---

[1] *Reglamento sobre la seguridad y protección de la información oficial y el modo en que se aplicarán las normas de seguridad establecidas, Resolución 1, de 26 de diciembre de 2000 en el Decreto Ley 199 del Ministerio de Interior.*

La herramienta colaborativa para la realización de auditorías tiene como objetivo cargar el script auditor y visualizarle al administrador de bases de datos los parámetros que el script contiene. Permite además que el administrador desactive los parámetros que no crea conveniente ejecutar en la BD, del mismo modo, facilita la conexión desde la aplicación hacia la base datos en cuestión para realizar la auditoría. Esta es una aplicación de escritorio que está desarrollada en Java como lenguaje de programación. Está diseñada de forma tal que se comporte como un asistente para la realización del diagnóstico a una base de datos determinada (5).

Durante el proceso de transferencia tecnológica a los especialistas de ETECSA fueron detectadas ciertas limitaciones en relación con la ejecución eficaz de las guías de revisión que se construyeron para realizar las auditorías a los gestores de base de datos (Microsoft SQL Server 2000 y 2005, PostgreSQL 8.4, 9.1 y MySQL 5.0). También se comprobó que la herramienta antes mencionada no cuenta con la capacidad de conectarse a los gestores Oracle en sus versiones 8i, 9i, 10g, y 11g. Luego del análisis realizado se evidenció que RASGBD v1.0 no cumple con los requerimientos de estandarización de TLM, además no tiene conexión con los gestores de Oracle. Por lo que se proponer realizar una nueva versión que permita realizar las auditorías a los gestores de bases de datos mencionados anteriormente y además que cumpla con las políticas de estandarización del Centro.

1.4 Metodología de desarrollo

La metodología hace referencia a un conjunto de procedimientos basados en principios lógicos, utilizados para alcanzar una gama de objetivos que rigen una investigación (6). Existen las metodologías Tradicionales o Pesadas y las Ágiles. Se decidió utilizar una metodología ágil ya que involucran más al equipo de desarrollo, pues al recaer gran parte del peso del análisis y el modelo del negocio sobre el equipo, los desarrolladores implicados van a tener una mayor visión de lo que se desea realizar.

1.4.1 Programación Extrema

La Programación Extrema (XP) es un enfoque de la ingeniería de software que se basa en la simplicidad, la comunicación y el reciclado continuo de código. Puede definirse como un conjunto de pasos de diversas metodologías. Tiene como principal característica su adaptación a entornos cambiantes. Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo (7). Está compuesta por cuatro fases, las cuales se describen a continuación:

Fase de planeación: Es donde se definen y desarrollan las Historias de Usuario (HU) en conjunto con el cliente. Se crea el Plan de iteraciones y Plan de entrega de esas versiones.
Fase de diseño: Se definen las tarjetas CRC (Class, Responsabilities and Collaboration) a tareas de la ingeniería asociadas a cada una de las HU.
Fase de codificación: Es donde se implementan las historias de usuario. Se deben utilizar los estándares de codificación para la implementación de la herramienta atendiendo a los estándares definidos por Python.
Fase de pruebas: Es donde se verifica el funcionamiento final de la herramienta realizando las pruebas unitarias y de aceptación.
Se decide utilizar la metodología XP pues se trata de un proyecto pequeño que no necesita la generación de tantos artefactos ni documentación. El cliente forma parte del equipo de desarrollo y el intercambio de opiniones con él juega un papel fundamental en el proceso de migración del sistema. El trabajo es desarrollado por una pareja de programadores con mediana experiencia; el propósito principal es alcanzar un producto que satisfaga las necesidades del cliente en el menor tiempo posible y con la calidad requerida. Otro punto es que se centra en los miembros del equipo, su interacción y en la entrega rápida de versiones funcionales del software. Debe ser acotado además que el cliente define que el producto sea desarrollado usando XP como metodología de desarrollo.

1.5 Herramientas y tecnologías

Las herramientas informáticas son programas, instrucciones o aplicaciones usadas para efectuar otras tareas de modo más sencillo. Es muy importante el uso de las herramientas adecuadas en cada caso ya que facilitan el trabajo en gran medida.

1.5.1 Lenguajes de programación

Un lenguaje de programación no es más que un sistema estructurado y diseñado principalmente para que las máquinas y computadoras se entiendan entre sí y con los humanos. Contienen un conjunto de acciones consecutivas que el ordenador debe ejecutar. Tienen la capacidad de especificar, de forma precisa, cuáles son los datos que debe trabajar un equipo informático, de qué modo deben ser conservados o transferidos dichos datos y qué instrucciones debe poner en marcha la computadora ante ciertas circunstancias (8). Los lenguajes de programación con los que se puede trabajar en entornos web se agrupan en dos categorías: lenguajes del lado del cliente, los cuales hacen más atractivo al usuario la presentación del producto y los lenguajes del lado del servidor, los cuales son los encargados de llevar la lógica del negocio.

1.5.2 Hyper Text Markup Language (HTML)

HTML es un lenguaje simple utilizado para crear documentos de hipertexto. Este estándar sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código, denominado código HTML, para la definición del contenido de una página web, como texto e imágenes. Básicamente su estructura se compone de etiquetas entre las cuales van insertados los elementos que componen la página como son los bloques de texto, scripts, la ruta a la ubicación de archivos externos como imágenes y otros archivos multimedia. Al navegador cargar dichos archivos representa todos los elementos en ella de forma adecuada (9). Este lenguaje juega un papel muy importante en el desarrollo del sistema ya que el objetivo que se persigue es crear una aplicación web, además un documento HTML independientemente de su organización y coherencia, comparte un mismo aspecto y una única interfaz, lo que facilita enormemente su manejo. Por otra parte HTML se encarga de desarrollar una descripción sobre los contenidos que aparecen como textos y sobre su estructura, complementando dicho texto con diversos objetos como fotografías y animaciones.

1.5.3 Python

Para la aplicación se utiliza como lenguaje de programación Python en su versión 2.7, el cual soporta la programación orientada a objeto, también es de código abierto, lo cual permite usarlo sin costo. Python es un lenguaje de scripts sencillo pero a la vez muy potente. Su filosofía hace hincapié en una sintaxis muy limpia y que favorece un código legible. Es un lenguaje de programación multiparadigma, esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional (10). Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de cadenas, números y archivos. Es multiplataforma, hay versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje, siempre y cuando exista un intérprete programado para él. Se decidió utilizar este lenguaje porque además de que posee una amplia colección de módulos estándar que se pueden usar como base de los programas o como ejemplos para aprender a programar con él, de esta manera se cumple con el proceso de estandarización llevado a cabo en el Centro de desarrollo TLM.

1.5.4 JavaScript Object Notation (JSON)

JSON en español Notación de Objetos de JavaScript es un formato ligero para el intercambios

de datos que básicamente describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos. Una de las mayores ventajas que tiene su uso es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías (11). Posee dos estructuras, la primera para representar cuatro tipos primitivos: cadenas, números, booleanos, valores nulos y la segunda dos tipos estructurados: objetos y arreglos (12).

1.5.5 Tecnología Ajax

La tecnología Ajax surgió para agilizar los desarrollos web, busca evitar las demoras propias de las peticiones y respuestas del servidor mediante la transmisión de datos en segundo plano usando un protocolo específicamente diseñado para la transmisión rápida de pequeños paquetes de datos. Con Ajax, se hace posible realizar peticiones al servidor y obtener respuesta de este en segundo plano sin necesidad de recargar la página web completa y usar esos datos para modificar los contenidos de la página creando efectos dinámicos y rápidos (13).

1.6 Framework de desarrollo

Un framework o marco de trabajo, en español, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto (14). Entre las ventajas que tiene su uso está que el programador no necesita plantearse una estructura global de la aplicación, sino que el framework le proporciona un esqueleto que hay que rellenar.

1.6.1 Django

Django1.4.7 es un framework de desarrollo web de código abierto, escrito en Python, que usa una modificación de la arquitectura Modelo Vista Controlador conocida como Model Template View. La meta fundamental de Django es facilitar la creación de sitios web complejos. Este pone énfasis en el re-uso, la conectividad, extensibilidad de componentes y el desarrollo rápido (15). Se decide utilizarlo como marco de trabajo ya que permite construir aplicaciones web más rápido y con menos código, este gestiona páginas de contenido sin necesidad de escribir controladores o vistas para esas páginas y un sistema de redirección de URLs (Localizador de Recursos Uniforme). Además agrupa aplicaciones que proporcionan un sistema de comentarios y herramientas para acoplar contenido. Asimismo gracias a que es un framework de alto nivel

escrito en Python hereda todas las características y facilidades que este brinda entre ellas escribir un código que es fácil de entender.

1.6.2 Xilema Base Web

Xilema-Base-Web es un marco de trabajo desarrollado en el Centro de TLM que está constituido por Django como framework base. Este contiene librerías de JavaScript como son Jquery y Backbone, además cuenta con las pautas de diseño de la Universidad de las Ciencias Informáticas.

1.6.3 Backbone 1.0.1

Backbone.js es un pequeño framework que permite construir aplicaciones usando Java Script siguiendo el patrón Modelo Vista Controlador. Da estructura a las aplicaciones web al ofrecer modelos con eventos personalizados y colecciones con una rica interfaz de programación de funciones .Backbone ayuda a mantener la lógica del negocio separada de la interfaz de usuario (16). Su objetivo fundamental es probar y definir un conjunto de estructuras de datos junto al manejo de la interfaz por medio de vistas y URLs útiles a la hora de construir aplicaciones Java Script. Por otro lado, Backbone permite que el comportamiento de la aplicación se pueda definir libremente con el resto de librerías o framework que se deseen incorporar (17). Se evidencia su uso en la versión 2.0 de RASGBD ya que Xilema Base Web lo utiliza para mostrar información haciendo uso de los servicios y métodos que tiene implementados.

1.8 Herramientas CASE

CASE es una sigla correspondiente a las iniciales Computer Aided Software Engineering cuya traducción al español significa Ingeniería de Software Asistida por Computación. Las herramientas CASE pueden definirse como la aplicación de métodos y técnicas a través de las cuales se hacen útiles a las personas comprender las capacidades de las computadoras, por medio de programas, procedimientos y su respectiva documentación (18).

1.8.1 Visual Paradigm

Visual Paradigm es una herramienta CASE que ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (19). Aunque existen diferentes herramientas que modelan el

proceso de desarrollo de un software, se decide utilizar Visual Paradigm en su versión 8.0, pues permite representar alrededor de 13 diferentes tipos de diagramas, posibilita modelar procesos de negocios y bases de datos. Brinda la opción de generar código (PHP, Java, Delphi, Python, C++, C#, Perl, Ruby) a partir de los diagramas, para las plataformas como .Net, Java y PHP, así como obtener los diagramas a partir del código. Con el uso de esta herramienta se pueden exportar los diagramas en imagen, Excel o formato XML y generar documentación en formatos HTML y PDF.

1.9 Business Process Model and Notation (BPMN)

BPMN, en español Modelo y Notación de Procesos de Negocio, es una notación gráfica estandarizada que permite el modelado de procesos de negocio, en un formato de flujo de trabajo. Su principal objetivo es proporcionar una notación que sea fácilmente legible y entendible por parte de todos los involucrados e interesados del negocio, de forma tal que se puedan representar gráficamente las diferentes etapas del proceso de negocio (20). Tiene la finalidad de servir como lenguaje común para cerrar la brecha de comunicación que frecuentemente se presenta entre el diseño de los procesos de negocio y su implementación. Se escogió BPMN en su versión 2.0 porque además de que es independiente de cualquier metodología de modelado permitirá al equipo de desarrollo modelar los procesos de una manera unificada, estandarizada y muy fácil.

1.10 Lenguajes de modelado

El lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar parte de un diseño de software orientado a objetos. Esta notación, principalmente gráfica, se utilizó para expresar el diseño que usan los métodos del sistema.

1.10.1 Unified Modeling Language (UML)

UML es un lenguaje gráfico utilizado para visualizar, especificar y documentar esquemas de sistemas de software orientados a objetos. Proporciona una forma de modelar procesos de negocio y funciones del sistema, permite escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables (21). Este es un lenguaje que ayuda a interpretar los sistemas mediante gráficos o mediante texto obteniendo modelos explícitos que ayudan a la comunicación durante el desarrollo ya que al ser estándar, los modelos pueden ser interpretados por personas que no participan en su diseño. Que sea un lenguaje de modelado significa que el vocabulario y las reglas que proporciona se utilizan para la representación conceptual y física del sistema. Ofrece la capacidad de modelar las actividades de planificación,

mediante estas se obtiene una documentación que es válida durante todo el ciclo de vida del proyecto. Este lenguaje se utilizó para modelar el diagrama de procesos de negocio del sistema.

1.11 IDE (Entorno de Desarrollo Integrado)

Un IDE, es un entorno de programación que ha sido empaquetado como un programa de aplicación que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Proveen un marco de trabajo para la mayoría de los lenguajes de programación tales como C++, PHP, Python, Java, C#, Delphi y Visual Basic (22).

1.11.1 Eclipse

Eclipse 3.7.2 es un entorno de desarrollo integrado de código abierto y multiplataforma, diseñado para la construcción de aplicaciones web y herramientas de desarrollo. Proporciona un modelo común de interfaz de usuario para trabajar con herramientas (23). Es una potente y completa plataforma de programación, desarrollo y compilación de elementos variados como sitios web, programas en los lenguajes de programación C++ o Python y aplicaciones Java (24). No es más que un IDE en el que se encuentran herramientas y funciones recogidas en una atractiva interfaz que hace el trabajo fácil y agradable de usar. Por todo lo antes mencionado es que se propone utilizar este entorno de desarrollo.

PyDev 4.0 es un plugin que permite a Eclipse ser utilizado como un IDE de Python haciendo de él un entorno de desarrollo de primera clase. Utiliza técnicas de inferencia de tipos avanzados para proporcionar características tales como finalización y análisis de código, sin dejar de ofrecer muchos otros como un depurador, consola interactiva y refactorización que no es más que tomar una pieza de código y modificarla de tal forma que haga exactamente lo mismo, pero su diseño mejore (25).

1.12 Sistemas Gestores de Bases de Datos (SGBD)

Para el desarrollo de la herramienta se emplearon los sistemas gestores de bases de datos PostgreSQL, MySQL, Microsoft SQL Server y Oracle a la hora de realizar el proceso de auditorías. Se profundizó en el estudio de este último pues la primera versión de RASGBD se conecta satisfactoriamente a todos los gestores mencionados exceptuando a Oracle en sus versiones 8i, 9i, 10g y 11g.

Oracle es un sistema gestor de base de datos relacional fabricado por Oracle Corporation. Básicamente es una potente herramienta cliente/servidor para gestionar bases de datos cuyos productos van desde bases de datos (Oracle) hasta sistemas de gestión. Cuenta además, con

herramientas propias de desarrollo para realizar potentes aplicaciones, como Oracle Designer (26). Una BD Oracle tiene una estructura física que se corresponde a los ficheros del sistema operativo y una estructura lógica que está formada por los tablespace[2] y los objetos de un esquema de bases de datos.

-**Oracle 8i:** Incluye mejoras de rendimiento y de utilización de recursos independientemente de que se necesite dar soporte a decenas de miles de usuarios y cientos de terabytes de datos, o se disponga de un sistema mucho más pequeño.

-**Oracle 9i:** Permite obtener cualquier mezcla de datos al nivel de reporte, para hacer más eficiente el negocio. Este producto está totalmente orientado a Internet. De esta manera, el usuario puede tomar las herramientas de software desarrolladas en Oracle, migrarlas a Internet y así tener acceso a su base de datos tanto de manera local, como desde cualquier otra parte del mundo.

-**Oracle 10g:** Es una versión mejorada y se compone de las funciones que le dan más control para almacenar, recuperar y procesar los datos. Puede utilizar las nuevas características de SQL a través de SQL Plus, que es la interfaz usada para la extracción y manipulación de datos. Soporta los espacios en blanco para la ruta de acceso y nombres de archivo.

-**Oracle 11g:** Permite que la infraestructura de base de datos sea mucho más resistente y fácil de administrar. Entre sus principales características está que posee copias de respaldo y recuperación ante desastres, el particionamiento de tablas y la gestión de esquemas (27).

1.13 Conclusiones parciales

En este capítulo se realizó un estudio de la herramienta RASGBD v 1.0, se analizó la necesidad de realizar una migración de la misma hacia las nuevas tecnologías definidas por el Centro de Telemática y para dar solución a los problemas que se identificaron. Se especificaron las principales herramientas, metodologías y lenguajes a utilizar para desarrollar el sistema. Primeramente se decidió utilizar la metodología de desarrollo XP, auxiliada por los lenguajes de modelado de procesos de negocio BPMN y UML con Visual Paradigm 8.0 como herramienta CASE. Para la implementación de RASGBD se seleccionó el lenguaje de programación Python 2.7 como lenguaje del lado del servidor y HTML como lenguaje del lado del cliente.

[2] *Unidad lógica de almacenamiento dentro de una base de datos Oracle.*

CAPÍTULO 2: EXPLORACIÓN Y PLANIFICACIÓN DE LA HERRAMIENTA COLABORATIVA PARA LA REALIZACIÓN DE AUDITORÍAS A SISTEMAS GESTORES DE BASES DE DATOS

2.1 Introducción

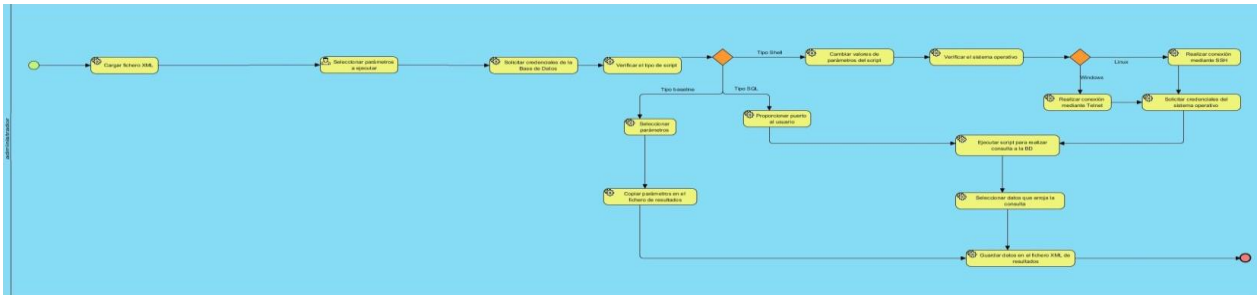
En este capítulo se presenta la propuesta y descripción del sistema Herramienta Colaborativa para la Realización de Auditorías a Sistemas Gestores de Base de Datos (RASGBD v2.0). Además, se realizan las acciones asociadas a las fases de Exploración y Planificación propias de la metodología de desarrollo utilizada, donde se confeccionan las Historias de Usuarios (HU), encargadas de la especificación de los requerimientos del sistema.

2.2 Descripción del proceso de negocio de RASGBD v2.0

El proceso de negocio de la Herramienta Colaborativa para la Realización de Autorías a Sistemas Gestores de Bases de Datos se inicia cuando se carga un fichero XML, que no es más que las guías de revisión o script para realizar el proceso de auditoría. Dicho fichero contiene el id, sistema gestor de base de datos con su respectiva versión, sistema operativo con su dirección ip, el tipo de script y la consulta a ejecutar. Seguidamente se seleccionan los parámetros que se quieren ejecutar, el sistema proporciona un puerto por el cual el usuario puede realizar la conexión y solicita las credenciales de la base de datos, las cuales son usuario, contraseña y dirección ip. Luego se verifica la conexión con el sistema operativo, si es Linux la conexión se realiza mediante SSH y si es Windows se conecta a través de Telnet. Se solicitan el usuario, la contraseña y la dirección ip del sistema operativo. Si el fichero para la ejecución de la auditoría (script) es de tipo SQL y dependiendo del gestor de bases de datos que se va a auditar, se le proporciona al usuario un puerto predeterminado que este puede o no usar y posteriormente se realiza la consulta a la base de datos. Por otra parte si el script es de tipo Shell, los valores de las variables adbuser, adbpass, adbip y adbport que están dentro del script se sustituyen internamente por los valores de las credenciales de la base de datos que se introdujeron por parámetro inicialmente. Al ejecutar el script y realizar la consulta a la base de datos, esta arroja un conjunto de datos, los cuales se recogen y son plasmados en un fichero XML que lleva por nombre fichero de resultados, que posee el resultado de la auditoría realizada a la base de datos inicialmente seleccionada. En caso de que el script sea de tipo Baseline o Nulo se seleccionan exactamente los parámetros que este contiene dentro del script y luego se copian en un fichero XML, siendo este es el resultado final que va a tener la consulta.

A continuación la imagen muestra el proceso de negocios de RASGBD v2.0.

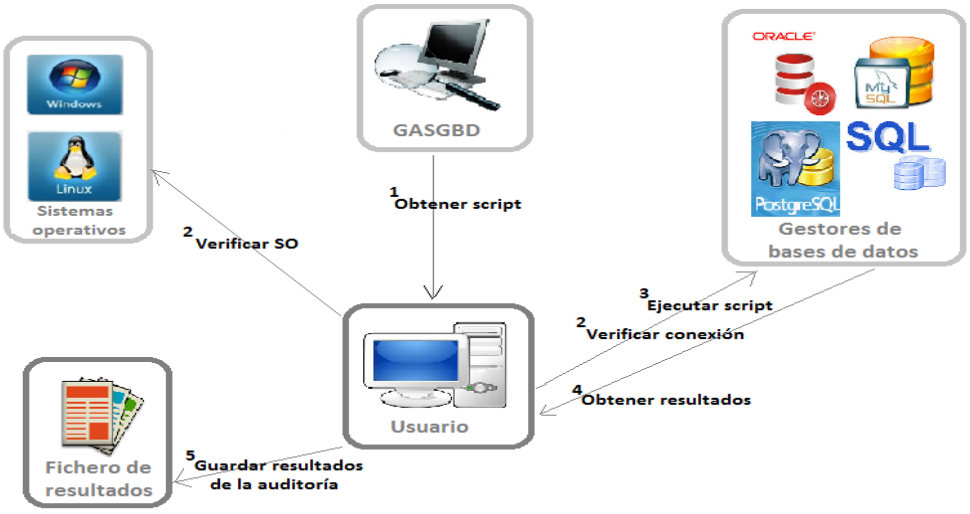
Imagen 1: Modelo de procesos del negocio de la herramienta RASGBD v2.0



2.3 Propuesta de solución

La solución informática propuesta se basa en realizar la versión 2.0 de la Herramienta Colaborativa para la Realización de Auditorías a Sistemas Gestores de Bases de Datos que cumpla con la estandarización llevada a cabo en Telemática, donde se define a Django como framework y a Python como lenguaje de programación; por lo cual se utiliza el patrón Modelo Plantilla Vista. El sistema que se pretende desarrollar es una aplicación web que permita la realización de consultas para gestores de bases de datos objeto de auditorías. Dicha solución incluye auditar los gestores de base datos MySQL 5.0, PostgreSQL 8.1, 9.1, Microsoft SQL Server 2000, 2005 y Oracle 8i, 9i, 10g, 11g. Al obtener un script enviado de la Herramienta para la Gestión de Auditorias a Sistemas Gestores de Bases de Datos (GASGBD) se debe comprobar la conexión con el sistema operativo y el sistema gestor de bases de datos. Permitirá que el usuario seleccione los parámetros y ejecute el script para realizar la auditoría. Finalmente los datos que arroja esta consulta se guardan en un fichero XML que va a contener los resultados. A continuación se muestra una imagen de la propuesta a realizar.

Imagen 2: Propuesta del sistema RASGBD v2.0



Las funcionalidades identificadas para la versión 2.0 de RASGBD son las siguientes:

1. Cargar el script XML.
2. Mostrar los parámetros que contiene el script.
3. Seleccionar los parámetros que se desee con la posibilidad de desactivar alguno que no se desee escoger.
4. Conectar a la base de datos.
5. Conectar con el sistema operativo según el tipo de script que es cargado.
6. Ejecutar el script XML para realizar la consulta a la base de datos.
7. Cambiar valores de parámetros.
8. Elaborar fichero XML con el resultado que arrojó la auditoría.
9. Guardar el fichero de resultados.
10. Mostrar un registro del proceso de auditorías realizadas.

2.4 Listas de reservas del producto

La lista de reserva del producto no es más que los requerimientos no funcionales de una aplicación, fundamentales en el éxito del producto y normalmente vinculados a requerimientos funcionales. Estas son un conjunto de características de calidad, que es necesario tener en cuenta al diseñar e implementar el software. Deben establecer restricciones en el producto que está siendo desarrollado y en el proceso de desarrollo (28).

Usabilidad: Para interactuar con el sistema es necesario contar con una preparación previa y tener un nivel medio o alto sobre computación. Luego de finalizado el proceso de implementación de la herramienta se debe realizar una capacitación a los usuarios involucrados con el sistema para que tengan un mejor entendimiento de este.

Restricciones de diseño e implementación: El lenguaje de programación que se utilizará es Python 2.7 y como marco de trabajo Django 1.4.7 Se utilizará Eclipse v3.7.2 como IDE de desarrollo. Visual Paradigm 8.0 se usará como herramienta CASE para visualizar, especificar y documentar cada una de las fases de XP, utilizando los lenguajes de modelado UML y BPMN.

Hardware: Para la instalación de la aplicación se debe disponer de una computadora que tenga un micro de 1.6 GHZ o mayor y la memoria RAM de 1 GB o superior.

Software: Las computadoras de los clientes que utilicen el sistema deben tener instalado el navegador Mozilla Firefox en su versión 17.0 o superior. Por parte del servidor deben tener el intérprete Python v2.7, el framework Django, los gestores de base de datos MySQL 5.0, PostgreSQL 8.1, 9.1, Microsoft SQL Server 2000, 2005, Oracle 8i, 9i, 10g, 11g y las librerías psycopg2, paramiko, telnetlib, MySQLdb, pymsql y cx_Oracle .

2.5 Fase de exploración

En esta primera fase se hace una recopilación de todos los requerimientos del proyecto, se interactúa con el usuario, se planifica bien entre los desarrolladores del proyecto que es lo que se quiere con el proyecto para así lograr los objetivos finales y definir el alcance general del proyecto (29). El cliente es el encargado de realizar las HU donde define las funcionalidades que el sistema debe tener. A partir de las HU definidas los programadores deben estimar los tiempos de desarrollo de cada una y comienzan a familiarizarse con las herramientas, tecnologías y procesos que van a usar para la elaboración del sistema.

2.5.1 Historias de Usuario (HU)

Las historias de usuario son técnicas que utiliza XP para especificar y administrar los requisitos de software. Se trata de una lista de características que deben existir en el producto final. Estas son escritas por el cliente como descripciones cortas de lo que el sistema debe realizar. Tienen el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Están compuestas por las siguientes secciones:

Número: Número de la historia de usuario incremental.

Usuario: Involucrados en el desarrollo de la HU.

Nombre de historia: El nombre de la historia de usuario.

Puntos estimados: Tiempo estimado para el desarrollo de la HU.

Iteración asignada: Número de la iteración.

Programadores responsables: Nombres de los programadores responsables.

Descripción: Breve descripción de la HU.

Observaciones: Observaciones del módulo.

Interfaz: Prototipo de interfaz.

Clasificación de las Historias de Usuario en cuanto a la prioridad en el negocio y riesgo en el desarrollo:

- **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.

- **Riesgo en el desarrollo:**

Alta: cuando en la implementación de las HU se considera la posible existencia de errores que lleven 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.

Para la implementación de la herramienta, según las funcionalidades que el cliente plantea, se realizan las historias de usuarios que se muestran a continuación:

Tabla 1: HU Cargar script

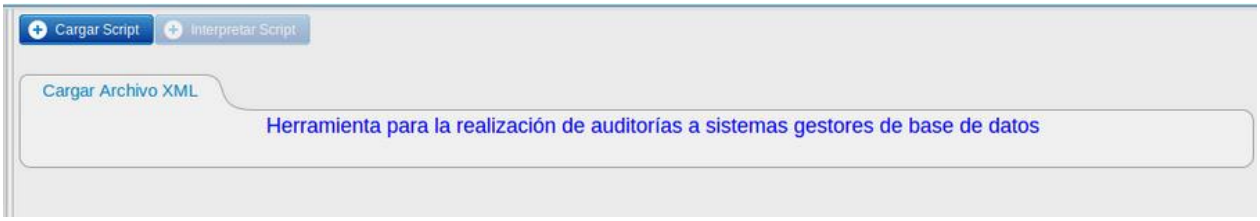
Historia de Usuario	
Número: 1	**Usuario:** Administrador de BD
Nombre de historia: Cargar script	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:**1
Programador(es) responsable(es): Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: Se debe cargar un script con los parámetros e indicadores para proceder a la realizar la auditoría a la base de datos.	
Observaciones: Debe conocerse la dirección del directorio donde está almacenado el script.	
Interfaz: 	

Tabla 2: HU Mostrar parámetros

Historia de Usuario	
Número: 2	**Usuario:** Administrador de BD
Nombre de historia: Mostrar parámetros	

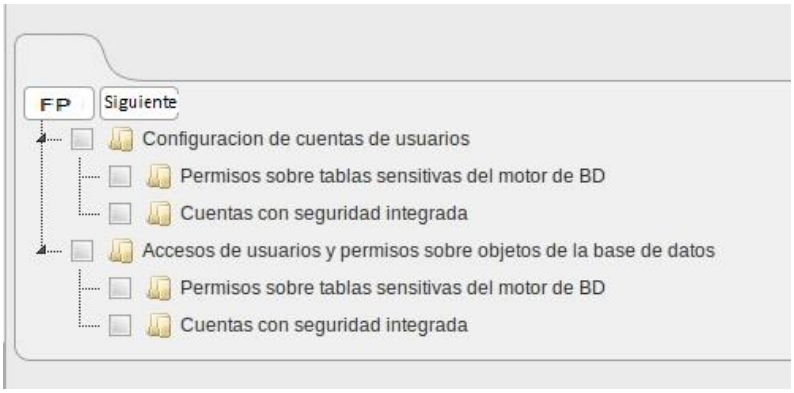
Prioridad en negocio: Media	**Prioridad en negocio:** Media
Puntos estimados:1	**Iteración asignada:** 1
Programador(es) responsable(es): Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: Deben mostrarse los parámetros con sus respectivos indicadores que vienen incluidos en el script obtenido.	
Observaciones: Se genera una vista previa de los parámetros del script una vez realizada esta acción.	
Interfaz: 	

Tabla 3: HU Seleccionar parámetros

Historia de Usuario	
Número: 3	**Usuario:** Administrador de BD
Nombre de historia: Seleccionar parámetros	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 1
Programador(es) responsable(es): Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: Brinda la posibilidad de seleccionar todos o algunos de los parámetros que se muestren. El usuario tiene la posibilidad de desactivar cualquier parámetro anteriormente seleccionado.	
Observaciones:	
Interfaz:	

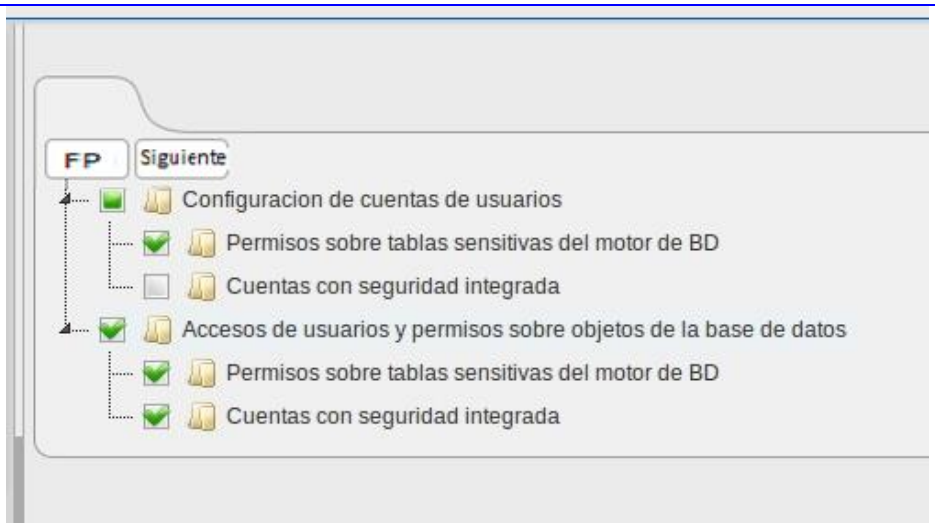


Tabla 4: HU Conectar a la BD

Historia de Usuario	
Número:4	**Usuario:** Administrador de BD
Nombre de historia: Conectar a la BD	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 2
Programador(es) responsable(es): Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: La aplicación de forma remota debe conectarse a la base de datos y además comprobar que existe conexión con ella.	
Observaciones: Deben conocerse las credenciales de la base de datos: usuario, contraseña, dirección ip y puerto por el que se va a realizar la conexión.	
Interfaz:	

Las tablas de las Historias de Usuario de la 5 a la 10 se encuentran en el **ANEXO I**.

2.6 Fase de planificación

En esta fase de la metodología XP, luego de tener definidas las HU se determina el esfuerzo que costará implementarlas. Los desarrolladores y clientes establecen los tiempos de implementación ideales de las historias de usuario, la prioridad con la que serán implementadas y las historias que serán implementadas en cada versión del programa (29). La medida que se usa en cada estimación es el punto, donde un punto equivale a una semana de trabajo sin ningún tipo de interrupción o sea 5 días. Además en esta fase se toman acuerdos sobre el contenido de las entregas que se realizarán. En esta fase se logrará crear parte del proyecto la parte física, o sea la interfaz que tendrá el cliente con el proyecto.

2.6.1 Estimación de esfuerzo por Historias de Usuario

En la tabla se muestra la estimación del esfuerzo por cada una de las historias de usuario para el desarrollo del sistema. Las estimaciones son definidas por los programadores usando como medida el punto, donde un punto equivale a una semana ideal (5 días).

Tabla 5: Estimación de esfuerzo por historias de usuario

Historias de Usuario	Puntos de estimación
1. Cargar script.	1
2. Mostrar parámetros.	1
3. Seleccionar parámetros.	1
4. Conectar a la BD.	1

5. Conectar con SO según el tipo de script.	1
6. Ejecutar script XML.	1
7. Cambiar valores de parámetros.	1
8. Elaborar fichero de resultado.	1
9. Guardar fichero de resultado.	0.4
10. Mostrar datos de la auditoría.	0.6

2.5.2 Plan de iteraciones

Después de estimar el esfuerzo de la implementación de las HU definidas, se decide realizar tres iteraciones en las cuales se agrupan las HU de acuerdo a su prioridad en el negocio, riesgo en el desarrollo y la relación que existe entre ellas.

Iteración #1

En la primera iteración del sistema RASGBD v 2.0 se implementarán las Historias de Usuario 1, 2 y 3 que están relacionadas con cargar el script para ejecutar la consulta a la base de datos, mostrar los parámetros que este contiene y seleccionarlos.

Iteración #2

En la segunda iteración de RASGBD v 2.0 se implementarán las HU 4, 5 y 6 que son las encargadas de la ejecución del script de revisión, la verificación de las conexiones a las bases de datos y la conexión a los sistemas operativos objeto de auditoría.

Iteración #3

En la última iteración del sistema se implementarán las HU 7, 8, 9 y 10 que están relacionadas con el cambio interno que se realiza en los parámetros del script, la elaboración y almacenamiento del fichero de resultados y el registro de los datos referentes a las auditorías realizadas.

2.5.3 Plan de duración de las iteraciones

El plan de duración de las iteraciones muestra el tiempo que demora realizar las iteraciones de la herramienta RASGBD v2.0. La duración total de cada una de las iteraciones se determina sumando los puntos estimados de las HU que se van a desarrollar en esa iteración.

Tabla 6: Plan de duración de las iteraciones

Iteración	Orden de las HU	Duración total
1	Cargar script Mostrar parámetros Seleccionar parámetros	3 semanas

2	Conectar a la BD Conectar con SO según el tipo de script Ejecutar script XML	3 semanas
3	Cambiar valores de parámetros Elaborar fichero de resultado Guardar fichero de resultado Mostrar datos de la auditoría	3 semanas

2.5.4 Plan de entrega

En el plan de entrega se determinan las fechas de culminación de cada una de las iteraciones de la Herramienta de Realización de Auditorías a Sistemas Gestores de Base de Datos v2.0.

Tabla 7: Plan de entrega de la herramienta

Módulo	Final de la iteración 1	Final de la iteración 2	Final de la iteración 3
Herramienta de Realización de Auditorías a Sistemas Gestores de Base de Datos v 2.0.	10 de abril de 2015	15 de mayo de 2015	22 de mayo 2015

2.6 Conclusiones parciales

Con el desarrollo de este capítulo se contribuyó a una mejor comprensión del sistema que se va a desarrollar. Se definieron las historias de usuario que se implementarán posteriormente, las cuales fueron definidas por el cliente, como parte de la fase de exploración que lleva a cabo la metodología XP, además fue definida la lista de reservas del producto. También quedó plasmada en la fase de planificación la cantidad de iteraciones que tendrá el desarrollo del sistema, así como una planificación del esfuerzo a realizar en el desarrollo de cada una de las historias de usuario.

CAPÍTULO 3: DISEÑO, IMPLEMENTACIÓN Y PRUEBAS

3.1 Introducción

En este capítulo se describe la fase de diseño, implementación y pruebas como parte de la metodología XP. Se identifica la arquitectura, los patrones arquitectónicos y las clases para las funcionalidades del sistema. Serán delimitadas las tareas de la ingeniería correspondientes a cada una de las historias de usuario y las tarjetas Clase-Responsabilidad-Colaborador.

3.2 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. Permite flexibilidad y adaptabilidad en mercados cambiantes, ofrece una guía para el desarrollo favoreciendo el control de los proyectos y establece un vocabulario común entre los participantes (30).

3.3 Patrón de arquitectura Model Template View (MTV)

El patrón de arquitectura Model Template View, en español Modelo Vista Plantilla, es implementado por el framework de desarrollo Django. Los tres componentes principales de este framework tienen como función crear una comunicación entre ellos para extraer información vital de la base de datos y presentarlas en el navegador (31). En este patrón, el Modelo hace referencia al acceso a la capa de datos, la Vista se refiere a la parte del sistema que selecciona qué mostrar y cómo mostrarlo, y el Controlador implica la parte del sistema que decide qué vista usar, dependiendo de la entrada del usuario, accediendo al modelo si es necesario (32).

Modelo: tiene como objetivo mapear la base de datos de forma que crea una sincronización entre la base de datos y la aplicación, con el fin de mantener actualizada toda la información de esta.

Vista: recibe el requerimiento que es enviado por el navegador y procesa la información presentada por el usuario.

Plantilla: muestra en pantalla al usuario la respuesta que es enviada por la vista.

El patrón Modelo Vista Plantilla o Modelo Vista Template en Django se entiende de la siguiente manera, el modelo sigue siendo el modelo, la vista no es una vista, sino que más bien es un controlador que se llama vista, y el template o plantilla son las vistas, es decir, los formularios, que hacen peticiones a las vistas, van en template y las vistas obtienen datos de los modelos (33).

La relación existente en el patrón Modelo Vista Template funciona de la siguiente manera: primeramente el navegador envía una solicitud a la vista, luego la vista interactúa con el modelo

para obtener datos de la base de datos y llama a la plantilla, la cual se encarga de enviar la respuesta a la solicitud del navegador.

3.4 Capas de la arquitectura Modelo Vista Template

A continuación se explican cada una de las clases que contienen las capas del patrón MTV las cuales están representadas en la imagen# 3 a través del diagrama de componentes.

Capa vista: En esta capa se muestran las clases que controlan todo lo relacionado con la lógica del negocio así como la información enviada a la plantilla y el control al acceso a datos.

-La clase RealizarAuditoría se encarga de realizar todo el proceso de auditoría y supervisar cada una de las clases que trabajan en la lógica del negocio.

-La clase ConexionPostgreSQL se encarga de realizar la conexión con el SGBD PostgreSQL.

-La clase ConexionOracle se encarga de realizar la conexión con el SGBD Oracle.

-La clase ConexionMySQL se encarga de realizar la conexión con el SGBD MySQL.

-La clase ConexionSQLServer se encarga de realizar la conexión con el SGBD Microsoft SQL Server.

-La clase ConexionSOWindows se encarga de realizar la conexión con el Sistema Operativo Windows.

-La clase ConexionSOLinux se encarga de realizar la conexión con el Sistema Operativo Linux.

-La clase FicheroResultados se encarga de elaborar el fichero de resultados.

-La clase InterpreteScriptRevision se encarga de interpretar el script de revisión para la ejecución de la auditoría.

-El subsistema Django se refiere al framework utilizado para el desarrollo de la herramienta.

Capa modelo: Muestra las clases resultantes de mapear la base de datos de las herramientas GASGBD y RASGBD v2.0.

-La clase rasgbd_auditoria es la encargada de guardar los datos asociados a la realización de las auditorías.

-La clase adbd_nSGBD se encarga de guardar los nombres de los sistemas gestores de bases de datos.

-La clase adbd_nSO se encarga de guardar los nombres de los sistemas operativos.

Capa plantilla: Muestra las clases relacionadas con enviar al usuario las respuestas de las vistas.

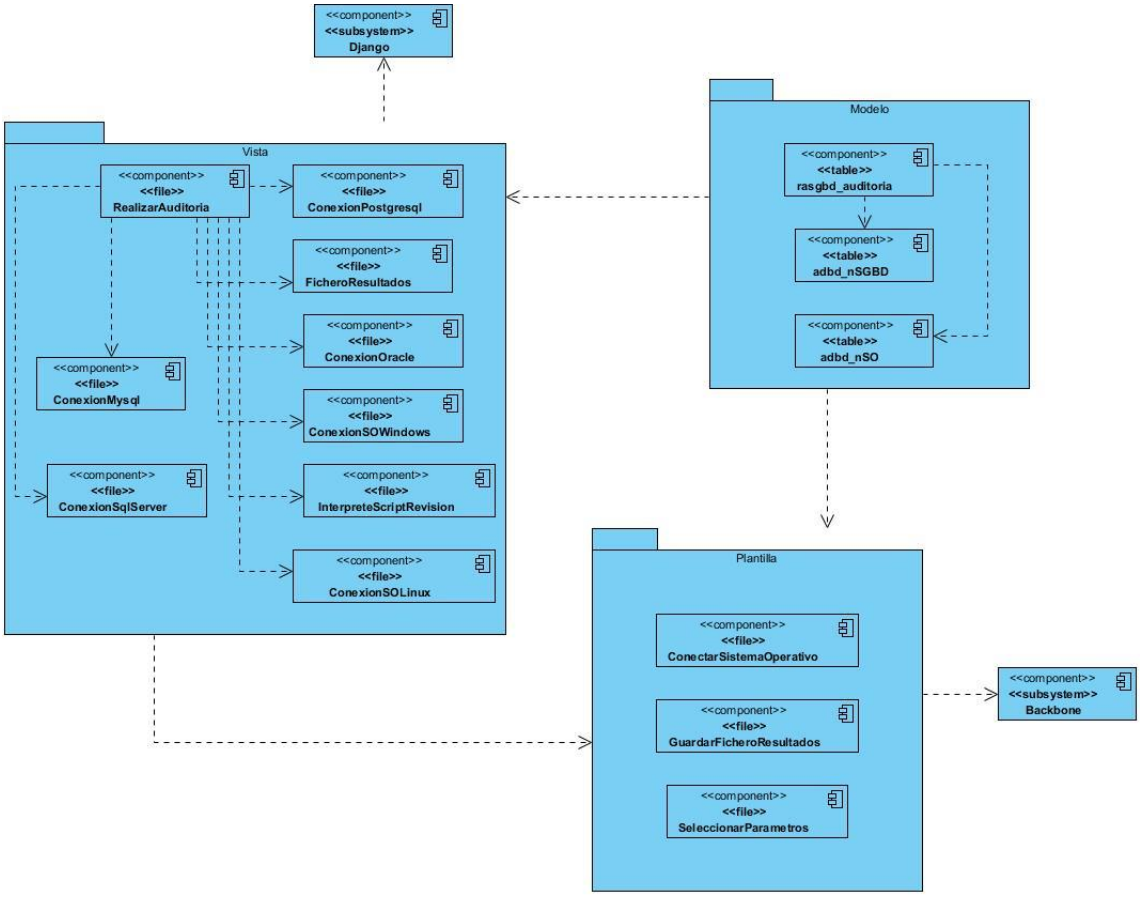
-La clase ConectarSistemaOperativo se encarga de mostrar los datos relacionados con la conexión al sistema operativo.

-La clase GuardarFicheroResultados muestra los datos relacionados con el almacenamiento del fichero de resultados.

-La clase SeleccionarParametros se encarga de mostrar los datos referentes a los parámetros que contiene el script.

-Backbone es el framework que se utilizó para el diseño de las interfaces.

Imagen 3: Arquitectura Modelo Vista Plantilla



3.5 Patrón de diseño

Los patrones de diseño son un conjunto de prácticas de óptimo diseño que se utilizan para abordar problemas recurrentes en la programación orientada a objetos. Pueden considerarse como un documento que define una estructura de clases que aborda una situación particular (34).

3.5.1 Patrones Generales de Software para la Asignación de Responsabilidades (GRASP)

Los patrones de diseño GRASP (General Responsibility Assignment Software Patterns) describen los principios fundamentales de la asignación de responsabilidades a objetos en una aplicación. Estos constituyen una serie de buenas prácticas recomendables para el diseño de software (35). En la herramienta RASGBD v2.0 se utilizaron los siguientes patrones GRASP:

Experto: Es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un

principio básico que suele utilizarse en el diseño orientado a objetos. Con la utilización de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide (35).

Las clases que contienen la información para cumplir con sus responsabilidades son ConexionWindows y ConexionPostgreSQL, las cuales se encargan de realizar todo el proceso de conexión a sus respectivas aplicaciones, así como de ejecutar las diversas tareas que impliquen el trabajo con las mismas.

Controlador: Se evidencia en una clase que sea la responsable de manejar un evento del sistema. Este patrón asigna la responsabilidad a la clase que representa al sistema completo (36). Se encuentra ejemplificado en la clase RealizarAuditoria, ya que la misma es la responsable de controlar todo el proceso del negocio.

Creador: Asigna responsabilidades relacionadas con la creación de instancia (36). Se pone de manifiesto en la clase InterpreteScriptRevision y también en gran medida en la clase RealizarAuditoria debido que la misma crea objetos de casi todas las clases que contiene la aplicación. Al escogerlo como creador, se da soporte al bajo acoplamiento.

Alta Cohesión: La cohesión es la medida en la que un componente se dedica a realizar solo la tarea para la cual fue creado, delegando las tareas complementarias a otros componentes (37). Su uso se evidencia en las clases de conexión ConexionWindows y ConexionPostgreSQL ya que ellas tienen una función bien definida en el sistema y sus propósitos están bien enfocados.

Bajo Acoplamiento: La idea de este patrón es tener las clases lo menos ligadas entre sí que se pueda, de tal forma que, en caso de producirse una modificación en alguna de ellas, tenga la mínima repercusión posible en el resto de clases. Es la medida en que los cambios de un componente tienden a necesitar cambios de otro componente (38). Todas las clases de la aplicación están bajamente acopladas, por ejemplo un cambio en la implementación en la clase FicheroResultados no afectaría a la clase controladora que utiliza sus métodos ya que se comunican a través de una interfaz bien definida.

3.6 Librerías de Python

Para realizar las conexiones a los protocolos Telnet, SSH y a los gestores de bases de datos PostgreSQL, Microsoft SQL Server, MySQL y Oracle se utilizaron las siguientes bibliotecas:

-cx_Oracle: es un módulo de extensión de Python que permite el acceso a bases de datos Oracle.

-telnetlib: se utiliza para realizar la conexión al sistema operativo Windows mediante el servicio Telnet.

-paramiko: es un módulo que se emplea en el uso del protocolo SSH para conexiones seguras a máquinas remotas en el sistema operativo Linux.

-pymssql: es una librería utilizada para realizar la conexión con Microsoft SQL Server.

-MySQLdb: es un módulo que se emplea para realizar las conexiones al gestor de base de datos MySQL.

-psycopg2: Se utiliza a la hora de realizar la conexión al gestor PostgreSQL.

3.7 Tarjetas CRC (Clase-Responsabilidad-Colaborador)

Esta técnica es usada para guiar el sistema a través de análisis guiados por la responsabilidad donde las clases se examinan, se filtran y se refinan en base a sus responsabilidades con respecto al sistema. El nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente.

Las tarjetas CRC están formadas por tres secciones:

Nombre de las clases: se especifica el nombre de la clase que se describe.

Responsabilidades: son las funciones que contiene implementada la clase.

Colaboradores: representa las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades (39).

A continuación se muestran las tarjetas Clase-Responsabilidad-Colaborador del sistema:

Tabla 8: Tarjetas CRC Clase: RealizarAuditoria

Nombre de la clase: RealizarAuditoria	
Responsabilidad:	**Colaborador:**
Es la clase controladora principal. Se encarga de realizar todo el proceso de auditoría. Supervisa a cada una de las clases que trabajan en la lógica del negocio.	ConexionPostgreSQL ConexionMySQL ConexionMSSQLServer ConexionOracle FicheroResultados InterpreteScriptRevison ConexionSOWindows ConexionSOLinux

Tabla 9: Tarjetas CRC Clase: InterpreteScriptRevison

Nombre de la clase: InterpreteScriptRevison	
Responsabilidad:	**Colaborador:**

Es la responsable de extraer los indicadores junto a sus parámetros de los script de revisión en lenguaje XML, junto a toda la información de cabecera de dicho fichero.	Indicador Parametro

Tabla 10: Tarjetas CRC Clase: FicheroResultados

Nombre de la clase: FicheroResultados	
Responsabilidad:	**Colaborador:**
Su misión consiste en elaborar el fichero de resultados con todos los datos resultantes de la ejecución de la auditoria.	Indicador Parametro

En el **ANEXO II** se muestran las demás tarjetas CRC.

3.8 Estándar de codificación

El estilo de código empleado es el que utiliza Python, el cual está definido por el estándar PEP 8. Las pautas tienen como objetivo mejorar la legibilidad del código y hacerlo consistente a través la comunidad Python (40). Seguidamente se muestran algunos de estos convenios y la evidencia de ellos en el código implementado de la herramienta RASGBD v2.0.

- El PEP 257 describe los estilos de nombramiento donde se define que no deben utilizarse caracteres con tildes. Se distinguen los siguientes estilos:

  Minúscula Mayúscula (minusculaMayuscula)

Imagen 4: Ejemplo del estándar mayúscula minúscula

```
def ejecutarScript(self,parametro):
    tipo=parametro.tipo.lower()
    if(tipo=='sql'):
        try:
            resultado=self.conexionBD.ejecutar(parametro.script)
            return resultado
        except:
            return['error']
```

- Diseño del código: Usa cuatro espacios por indentación.
- Para escribir correctamente las cadenas de documentación se usan las """ (comillas), las que finalizan la cadena multilínea debe estar solo en una línea, y preferiblemente precedido por una línea en blanco.

Imagen 5: Ejemplo del estándar de identación y diseño de código

```
def sustituirValoresScriptRev(self,script):
    script1=script.replace('adbdUser',"'"+self.user+"'")
    script2=script1.replace('adbdPass',"'"+self.password+"'")
    script3=script2.replace('adbdHost',"'"+self.host+"'")
    script4=script3.replace('adbdPort',"'"+self.puerto+"'")
    return script4
```

- Palabras Con Mayúscula (PalabrasMayusculas)

Imagen 6: Ejemplo del estándar palabras mayúsculas

```
class RealizarAuditoria():

    def interpretarXml(self,ruta):
        op= open(ruta)
        self.inter= Interprete(op)
        self.indicadores= self.inter.indicadores()
```

3.8 Diseño de la base de datos

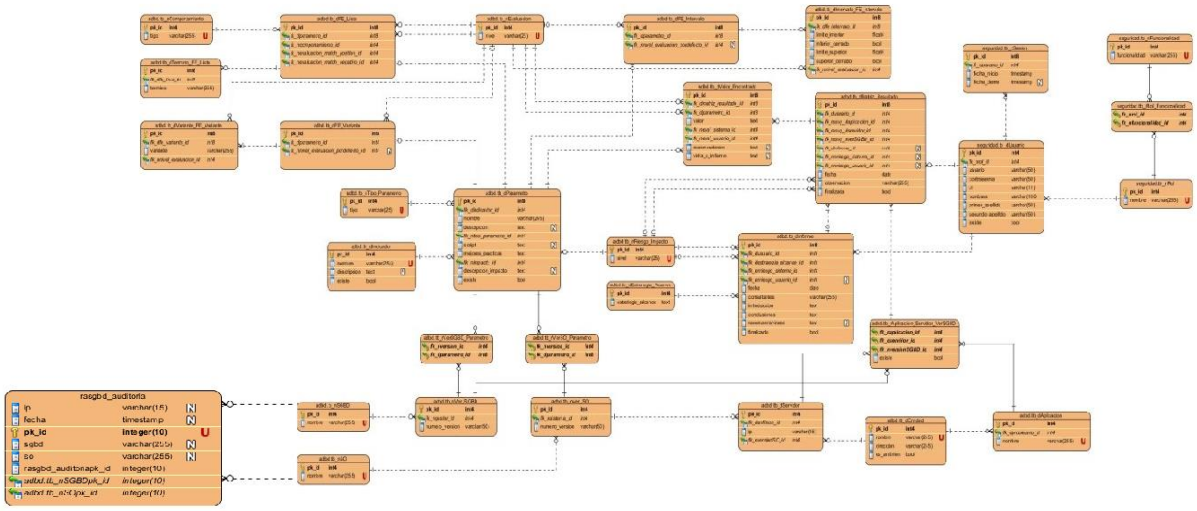
En la imagen# 7 se muestra el diagrama de la base de datos de GASGBD la cual almacena la información utilizada por la herramienta. Este diseño es de vital importancia para el funcionamiento de RASGBD v2.0 ya que para la correcta ejecución de las auditorías es imprescindible el script enviado por dicha herramienta. Las entidades que se utilizan son las que se muestran a continuación.

-La tabla adbd_nSGBD guarda los nombres de los sistemas gestores de bases de datos a los que se les realiza auditorías.

-La tabla adbd_nSO se guarda los nombres de los sistemas operativos que son objeto de auditorías.

-rasgbd_auditoria es una tabla de la herramienta RASGBD v2.0 que almacena los datos asociados a la realización de las auditorías dentro de los que se encuentran sistema operativo, sistema gestor de bases de datos, ip, fecha en que se realizó.

Imagen 7: Modelo de datos



3.9 Tareas de la ingeniería

XP plantea que la implementación de un software se hace iterativamente, obteniendo al culminar cada iteración un producto funcional, que debe ser probado y mostrado al cliente. Durante el transcurso de las iteraciones, se realiza la implementación de las HU (41). Como parte de este plan, se descomponen estas HU en tareas de la ingeniería las cuales son asignadas a los programadores para ser implementadas durante la iteración correspondiente.

Las tareas de la ingeniería están conformadas por:

Número tarea: los números deben ser consecutivos.

Número HU: número de la historia de usuario a la que pertenece la tarea.

Nombre tarea: nombre que identifica a la tarea.

Tipo de tarea: la tarea puede ser de tipo: Desarrollo, Corrección, Mejora u Otro.

Puntos estimados: tiempo estimado en semanas que se le asignará a su desarrollo, este no puede ser mayor que tres semanas.

Fecha inicio: fecha en que inicia el desarrollo de la tarea.

Fecha fin: fecha en que finaliza el desarrollo de la tarea.

Programador responsable: nombre y apellidos del(los) programador(es).

Descripción: brinda una breve descripción de la tarea.

A continuación se muestran algunas de las tareas de ingeniería del sistema:

Tabla 11: Tareas de ingeniería #1 Cargar script

Tareas de ingeniería

Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre de Tarea: Cargar script.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:**1
Fecha inicio: 23/03/2015	**Fecha fin:** 27/03/2015
Programador Responsable: Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: El administrador de la BD carga un fichero XML que no es más que las llamadas guías de revisión.	

Tabla 12: Tareas de ingeniería #2 Mostrar parámetros

Tareas de ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 2
Nombre de Tarea: Mostrar parámetros.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 30/03/2015	**Fecha fin:** 03/04/2015
Programador Responsable: Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: El sistema muestra una vista previa de los parámetros que contiene el script de acuerdo con el sistema gestor de bases datos y la versión que este posee entre sus parámetros.	

Tabla 13: Tareas de ingeniería #3 Seleccionar parámetros

Tareas de ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 3
Nombre de Tarea: Seleccionar parámetros.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 06/04/2015	**Fecha fin:** 10/04/2015
Programador Responsable: Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: El administrador de la BD selecciona el gestor de BD y su versión para luego seleccionar el (los) parámetro (s) que desea ejecutar. Si lo desea puede seleccionar todos los parámetros que se muestra o desactivar alguno que no considere de interés.	

Las tareas de la ingeniería que corresponden a las historias de usuario de la 4 a la 10 se

encuentran en el **ANEXO III**.

3.10 Pruebas

Uno de los pilares de la metodología XP es el uso de pruebas para comprobar el funcionamiento de los códigos implementados. XP divide las pruebas en dos grupos: las pruebas unitarias y las pruebas de aceptación.

Las pruebas unitarias suelen denominarse pruebas de caja blanca, ya que permiten determinar si un módulo del programa está listo y correctamente terminado. El objetivo de estas pruebas es asegurar el correcto funcionamiento de las interfaces o flujo de datos entre componentes (42). Las pruebas de aceptación se conocen también por el nombre de pruebas de caja negra y son creadas a partir de las historias de usuario. Verifican que el sistema funciona y lo hace de acuerdo con las especificaciones descritas por el cliente.

3.11 Pruebas unitarias

Las pruebas unitarias son efectuadas a fragmentos de código de la herramienta, mediante ellas se verifica el correcto funcionamiento de las unidades lógicas en las que se divide el software. Para la ejecución de las pruebas unitarias se utilizó la herramienta unittest, también conocida como PyUnit, la cual está contenida en el lenguaje de programación Python. A continuación se muestran los resultados obtenidos en las 3 iteraciones realizadas al método *ejecutarScript* con el objetivo de comprobar su funcionamiento.

3.11.1 Iteración 1

Imagen 8: Pruebas unitarias. Iteración 1

En esta iteración se seleccionaron 6 fragmentos de código en los cuales no se detectaron errores.

```
----------------------------------------------------------------------
Ran 6 tests in 0.042s

OK
Destroying test database for alias 'default'...
```

3.11.2 Iteración 2

Imagen 9: Pruebas unitarias. Iteración 2

Durante esta iteración se seleccionaron 9 fragmentos de código en los cuales no se detectaron errores.

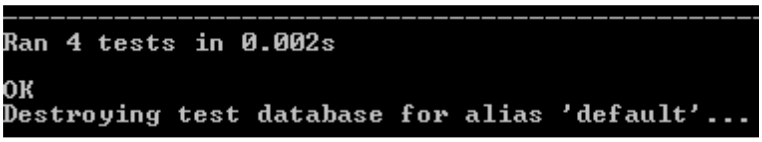
```
----------------------------------------------------------------------
Ran 9 tests in 0.060s

OK
Destroying test database for alias 'default'...
```

3.11.3 Iteración 3

Imagen 10: Pruebas unitarias. Iteración 3

En esta iteración se seleccionaron 4 fragmentos de código en los cuales no se detectaron errores.



3.12 Pruebas de aceptación

A continuación se muestran algunos casos de pruebas de aceptación realizadas a las interfaces de la herramienta RASGBD v2.0:

Tabla 14: Prueba de Aceptación #1: HU 1: Cargar script

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba
El usuario accede al módulo (RASGBD), y selecciona la opción (Cargar script), luego busca en el directorio donde tiene almacenado el script, lo selecciona y presiona el botón (Aceptar).		El sistema carga satisfactoriamente el script que es obtenido de la herramienta GASGBD.	Satisfactorio.
	El usuario accede al módulo (RASGBD), y selecciona la opción (Cargar script de revisión), luego busca en el directorio donde tiene almacenado el script, no lo selecciona y presiona el botón (Aceptar).	El sistema muestra un error “Por favor cargue nuevamente el script de revisión”.	
	El usuario accede al módulo (RASGBD), y	El sistema muestra un error “El archivo no se	

	selecciona la opción (Cargar script de revisión), luego busca en el directorio donde tiene almacenado el script, selecciona un fichero que no es de extensión XML y presiona el botón (Aceptar).	corresponde con un script de revisión, por favor inténtelo de nuevo".	

Las pruebas de aceptación correspondientes a las Historias de Usuario 2, 3, 4, 5 y 9 se encuentran en el **ANEXO IV.**

En las pruebas de aceptación se realizaron 3 iteraciones donde fueron detectadas 17 no conformidades. Durante la primera iteración se detectaron 9 no conformidades, donde 7 fueron resueltas y quedaron pendientes 2. En la segunda iteración fueron detectadas 5 no conformidades de las cuales 4 fueron resueltas y quedó 1 pendiente. En la tercera y última iteración se detectaron 3 no conformidades las cuales fueron resueltas y ninguna quedó pendiente. Estos resultados se pueden observar en la siguiente imagen:

Imagen 11: Resultados de las pruebas de aceptación

3.13 Conclusiones parciales

En este capítulo se realizó un análisis de las distintas arquitecturas y los patrones arquitectónicos que se utilizaron durante el desarrollo del sistema RASGBD v 2.0. Se representaron las tareas de la ingeniería las cuales forman parte de las historias de usuario detalladas anteriormente en el desarrollo del capítulo 2, así como las tarjetas CRC las cuales organizan las clases necesarias para implementar el sistema y los estándares que son utilizados para la codificación de la herramienta. Además se redactaron las tareas de ingeniería y se le realizaron pruebas unitarias y de aceptación a la herramienta.

CONCLUSIONES GENERALES

Mediante la presente investigación se abordó acerca del proceso de desarrollo de la Herramienta Colaborativa para la Realización de Auditorías Sistemas Gestores de Bases de Datos en su versión 2.0. Una vez finalizado el desarrollo del sistema se dio cumplimiento, de forma satisfactoria, a los objetivos trazados para el desarrollo de este trabajo evidenciándose las siguientes conclusiones:

- El análisis realizado a la primera versión de RASGBD v1.0 permitió determinar la necesidad de desarrollar una versión 2.0 de esta herramienta.
- El estudio de las distintas herramientas, tecnologías y metodologías de software definidas por el centro permitieron un uso correcto de las mismas en la implementación de RASGBD v2.0.
- La aplicación web implementada genera una segunda versión para la realización de auditorías, que cumple con los requerimientos tecnológicos del Centro Telemática y todas las condiciones mínimas exigidas.

Se puede afirmar luego del análisis anterior que el presente trabajo logró los objetivos propuestos.

Recomendaciones

Se recomienda probar las conexiones a los sistemas gestores de bases de datos Oracle y Microsoft SQL Server con la implementación realizada en la herramienta.

REFERENCIAS

1. **mastermagazine.info. [En línea] http://www.mastermagazine.info/termino/3958.php.**

2. Piattini, Mario. *Auditoria Informática, un enfoque práctico.* 2011.

3. fceia.unr.edu.a. [En línea] 2011. http://www.fceia.unr.edu.ar/asist/intro-aa-t.pdf.

4. slideshare.net. *Auditoría Informática.* [En línea] http://es.slideshare.net/wilvin77/tipos-de-auditoria-informatica?related=1.

5. Informáticas, Centro de telemática de La Universidad de las Ciencias. *Herramienta Colaborativa para la Realización de Auditorías a Sistemas gestores de bases de Datos.* La Habana, Cuba : s.n., 2012.

6. Valdéz, José Luis Cendejas. Enciclopedia vitual. *Modelos y metodologías para el desarrollo de software.* [En línea] 2014. http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm.

7. programacionextrema.com. *Fases de la programación extrema.* [En línea] http://programacionextrema.tripod.com/fases.htm.

8. Definición de lenguaje de programación. [En línea] 2014. Definición de lenguaje de programación - Qué es, Significado y Concepto http://definicion.de/lenguaje-de-programacion/#ixzz3YavSO0FH.

9. HTML.com. [En línea] 2008. http://www.html.com.

10. hipertextual.com. [En línea] 2014. http://hipertextual.com/archivo/pycharm-ide-python.

11. geekytheory. [En línea] 13 de octubre de 2013. https://geekytheory.com/json-i-que-es-y-para-que-sirve-json/.

12. http://frontendlabs.io/. [En línea] 26 de julio de 2014. http://frontendlabs.io/1490--json-que-es-json-parse-json-stringify.

13. aprenderaprogramar.com. [En línea] 2006-2015. http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=882:ique-es-y-para-que-sirve-ajax-ventajas-e-inconvenientes-javascript-asincrono-xml-y-json-cu01193e&catid=78:tutorial-basico-programador-web-javascript-desde-&Itemid=206.

14. Diccionario de Informática y Tecnología. [En línea] 2014. http://www.alegsa.com.ar/Dic/framework.php.

15. Pagina Oficial Django. [En línea] http://www.django.es.

16. backbone.js. [En línea] http://backbonejs.org/.

17. Backbone.js 1.0. [En línea] 23 de marzo de 2013. http://www.genbetadev.com/desarrollo-web/backbone-js-el-framework-para-construir-

aplicaciones-usando-javascript-siguiendo-el-patron-mvc-alcanza-la-version-1-0.

18. Herramientas Case. [En línea] http://fds-herramientascase.blogspot.com.

19. Sitio Oficial Visual paradigm. [En línea] 2014. http://www.visual-paradigm.com/.

20. Business Process Modeling Notation. [En línea] Enero de 2008. http://www.omg.org/bpmn/Documents/BPMN_1-1_Specification.pdf.

21. The Unified Modeling Language. [En línea] 2014. http://www.uml-diagrams.org/.

22. fergarciac.wordpress.com. [En línea] 2014. https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.

23. Eclipse. [En línea] 2015. http://www.eclipse.org/ide/.

24. Redinertho. [En línea] https://redinertho.wordpress.com/2012/06/22/ventajas-en-la-utilizacion-de-eclipse/.

25. Eclipse marketplace. [En línea] 2014. http://marketplace.eclipse.org/content/pydev-python-ide-eclipse#sthash.8nBr0i4y.dpuf.

26. Pagina Oficial Oracle. [En línea] http://www.oracle.com.

27. iessanvicente.com. [En línea] https://iessanvicente.com/colaboraciones/oracle.pdf.

28. Carlos Cortez, Vanessa Molina, Liseth Paternina, Oscar Vargas. Metodologias Ágiles. [En línea] 2015. http://es.slideshare.net/LisPater1/metodologias-agiles-xp.

29. Metodologías Ágiles de desarrollo de software (XP) Fases . [En línea] 28 de junio de 2008. http://boards5.melodysoft.com/UBV_INGS/metodologias-agiles-de-desarrollo-43.html.

30. Facultad de Ingeniería Uruguay. [En línea] Centro de Posgrados y Actualización Profesional en Informática. https://www.fing.edu.uy/node/4656.

31. revistatelematica.cujae.edu.cu. [En línea] 2012. revistatelematica.cujae.edu.cu/index.php/tele/article.

32. LIBROSWEB. [En línea] 2015. https://librosweb.es/libro/django_1_0/capitulo_5/el_patron_de_diseno_mtv.html.

33. Unodepiera. Tutoriales sobre desarrollo web y móvil. [En línea] 2014. http://uno-de-piera.com/mvt-en-django-un-poco-de-teoria/.

34. kioskea.net. [En línea] http://es.kioskea.net/contents/224-patrones-de-diseno.

35. Astudillo, Marcello Visconti y Hernán. *Fundamentos de Ingeniería de Software.*

36. Practicas de software. [En línea] 21 de 03 de 2011. http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

37. lsi.us.es. [En línea] http://www.lsi.us.es/docencia/get.php?id=906.

38. juan-garcia-carmona.blogspot.com. [En línea] 07 de 09 de 2012. http://juan-garcia-

carmona.blogspot.com/2012/09/grasp-alta-cohesion-y-bajo-acoplamiento.html.

39. www.davidvalverde.com. [En línea] 2014. http://www.davidvalverde.com/blog/introduccion-a-la-programacion-extrema-xp/.

40. Por Guido van Rossum, Barry Warsaw. Guía de estilo del código Python. [En línea] 10 de agosto de 2007. http://www.python.org/doc/essays/styleguide.html.

41. Emilio A. Sánchez, Patricio Letelier, José H. Canós. *Mejorando la gestión de historias de usuario en XP.* Valencia España : Departamento de Sistemas Informáticos y Computación, 2014.

42. B., Ing. Alexander Oré. CalidadySoftware.com. [En línea] 2009. http://www.calidadysoftware.com/testing/pruebas_unitarias2.php.

BIBLIOGRAFÍA

1. **mastermagazine.info. [En línea] http://www.mastermagazine.info/termino/3958.php.**

2. Piattini, Mario. *Auditoria Informática, un enfoque práctico.* 2011.

3. fceia.unr.edu.a. [En línea] 2011. http://www.fceia.unr.edu.ar/asist/intro-aa-t.pdf.

4. slideshare.net. *Auditoría Informática.* [En línea] http://es.slideshare.net/wilvin77/tipos-de-auditoria-informatica?related=1.

5. Informáticas, Centro de telemática de La Universidad de las Ciencias. *Herramienta Colaborativa para la Realización de Auditorías a Sistemas gestores de bases de Datos.* La Habana, Cuba : s.n., 2012.

6. Valdéz, José Luis Cendejas. Enciclopedia vitual. *Modelos y metodologías para el desarrollo de software.* [En línea] 2014. http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm.

7. programacionextrema.com. *Fases de la programación extrema.* [En línea] http://programacionextrema.tripod.com/fases.htm.

8. Definición de lenguaje de programación. [En línea] 2014. Definición de lenguaje de programación - Qué es, Significado y Concepto http://definicion.de/lenguaje-de-programacion/#ixzz3YavSO0FH.

9. HTML.com. [En línea] 2008. http://www.html.com.

10. hipertextual.com. [En línea] 2014. http://hipertextual.com/archivo/pycharm-ide-python.

11. geekytheory. [En línea] 13 de octubre de 2013. https://geekytheory.com/json-i-que-es-y-para-que-sirve-json/.

12. http://frontendlabs.io/. [En línea] 26 de julio de 2014. http://frontendlabs.io/1490--json-que-es-json-parse-json-stringify.

13. aprenderaprogramar.com. [En línea] 2006-2015. http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=882:ique-es-y-para-que-sirve-ajax-ventajas-e-inconvenientes-javascript-asincrono-xml-y-json-cu01193e&catid=78:tutorial-basico-programador-web-javascript-desde-&Itemid=206.

14. Diccionario de Informática y Tecnología. [En línea] 2014. http://www.alegsa.com.ar/Dic/framework.php.

15. Pagina Oficial Django. [En línea] http://www.django.es.

16. backbone.js. [En línea] http://backbonejs.org/.

17. Backbone.js 1.0. [En línea] 23 de marzo de 2013. http://www.genbetadev.com/desarrollo-web/backbone-js-el-framework-para-construir-

aplicaciones-usando-javascript-siguiendo-el-patron-mvc-alcanza-la-version-1-0.

18. Herramientas Case. [En línea] http://fds-herramientascase.blogspot.com.

19. Sitio Oficial Visual paradigm. [En línea] 2014. http://www.visual-paradigm.com/.

20. Business Process Modeling Notation. [En línea] Enero de 2008. http://www.omg.org/bpmn/Documents/BPMN_1-1_Specification.pdf.

21. The Unified Modeling Language. [En línea] 2014. http://www.uml-diagrams.org/.

22. fergarciac.wordpress.com. [En línea] 2014. https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.

23. Eclipse. [En línea] 2015. http://www.eclipse.org/ide/.

24. Redinertho. [En línea] https://redinertho.wordpress.com/2012/06/22/ventajas-en-la-utilizacion-de-eclipse/.

25. Eclipse marketplace. [En línea] 2014. http://marketplace.eclipse.org/content/pydev-python-ide-eclipse#sthash.8nBr0i4y.dpuf.

26. Pagina Oficial Oracle. [En línea] http://www.oracle.com.

27. iessanvicente.com. [En línea] https://iessanvicente.com/colaboraciones/oracle.pdf.

28. Carlos Cortez, Vanessa Molina, Liseth Paternina, Oscar Vargas. Metodologias Ágiles. [En línea] 2015. http://es.slideshare.net/LisPater1/metodologias-agiles-xp.

29. Metodologías Ágiles de desarrollo de software (XP) Fases . [En línea] 28 de junio de 2008. http://boards5.melodysoft.com/UBV_INGS/metodologias-agiles-de-desarrollo-43.html.

30. Facultad de Ingeniería Uruguay. [En línea] Centro de Posgrados y Actualización Profesional en Informática. https://www.fing.edu.uy/node/4656.

31. revistatelematica.cujae.edu.cu. [En línea] 2012. revistatelematica.cujae.edu.cu/index.php/tele/article.

32. LIBROSWEB. [En línea] 2015. https://librosweb.es/libro/django_1_0/capitulo_5/el_patron_de_diseno_mtv.html.

33. Unodepiera. Tutoriales sobre desarrollo web y móvil. [En línea] 2014. http://uno-de-piera.com/mvt-en-django-un-poco-de-teoria/.

34. kioskea.net. [En línea] http://es.kioskea.net/contents/224-patrones-de-diseno.

35. Astudillo, Marcello Visconti y Hernán. *Fundamentos de Ingeniería de Software.*

36. Practicas de software. [En línea] 21 de 03 de 2011. http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

37. lsi.us.es. [En línea] http://www.lsi.us.es/docencia/get.php?id=906.

38. juan-garcia-carmona.blogspot.com. [En línea] 07 de 09 de 2012. http://juan-garcia-

carmona.blogspot.com/2012/09/grasp-alta-cohesion-y-bajo-acoplamiento.html.

39. www.davidvalverde.com. [En línea] 2014. http://www.davidvalverde.com/blog/introduccion-a-la-programacion-extrema-xp/.

40. Por Guido van Rossum, Barry Warsaw. Guía de estilo del código Python. [En línea] 10 de agosto de 2007. http://www.python.org/doc/essays/styleguide.html.

41. Emilio A. Sánchez, Patricio Letelier, José H. Canós. *Mejorando la gestión de historias de usuario en XP.* Valencia España : Departamento de Sistemas Informáticos y Computación, 2014.

42. B., Ing. Alexander Oré. CalidadySoftware.com. [En línea] 2009. http://www.calidadysoftware.com/testing/pruebas_unitarias2.php.

ANEXO I

Tabla 15: HU Conectar con Sistema Operativo (SO) según el tipo de script

Historia de Usuario	
Número: 5	**Usuario:** Administrador de BD
Nombre de historia: Conectar con Sistema Operativo (SO) según el tipo de script	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 2
Programador(es) responsable(es): Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: Se debe comprobar que exista conexión con el SO, además de especificar los datos donde se encuentra el servidor de BD que son usuario y contraseña, además se podrá comprobar si existe conexión con él.	
Observaciones: Según el tipo de script posteriormente se debe realizar una acción determinada.	
Interfaz: 	

Tabla 16: HU Ejecutar script XML

Historia de Usuario	
Número: 6	**Usuario:** Administrador de BD
Nombre de historia: Ejecutar script XML	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 2
Programador(es) responsable(es): Enrique Muschett Cortina y Claudia Vazquez Figueroa.	

Descripción: Se ejecutará el fichero XML para de esta forma realizar la consulta a la base de datos.
Observaciones: Debe de existir conexión a la BD que se auditará.
Interfaz: No aplica.

Tabla 17: HU Cambiar valores de parámetros

Historia de Usuario	
Número: 7	**Usuario:** Administrador de BD
Nombre de historia: Cambiar valores de parámetros	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 3
Programador(es) responsable(es): Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: Deben cambiarse los valores que tienen dentro del script las variables adbuser, adbpass, adbip y adbport por los de las credenciales de la base de datos que se introdujeron inicialmente.	
Observaciones: Solo se hará en caso de que el script sea de tipo Shell.	
Interfaz: No aplica.	

Tabla 18: HU Elaborar fichero de resultados

Historia de Usuario	
Número: 8	**Usuario:** Administrador de BD
Nombre de historia: Elaborar fichero de resultado	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 3
Programador(es) responsable(es): Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: Se debe elaborar un fichero en formato XML donde se almacene el resultado de la ejecución de la auditoría.	
Observaciones: El fichero de resultado debe contener los datos de la ejecución del script luego de haberse realizado la consulta a la(s) base(s) de datos.	
Interfaz: No aplica.	

Tabla 19: HU Guardar fichero de resultados

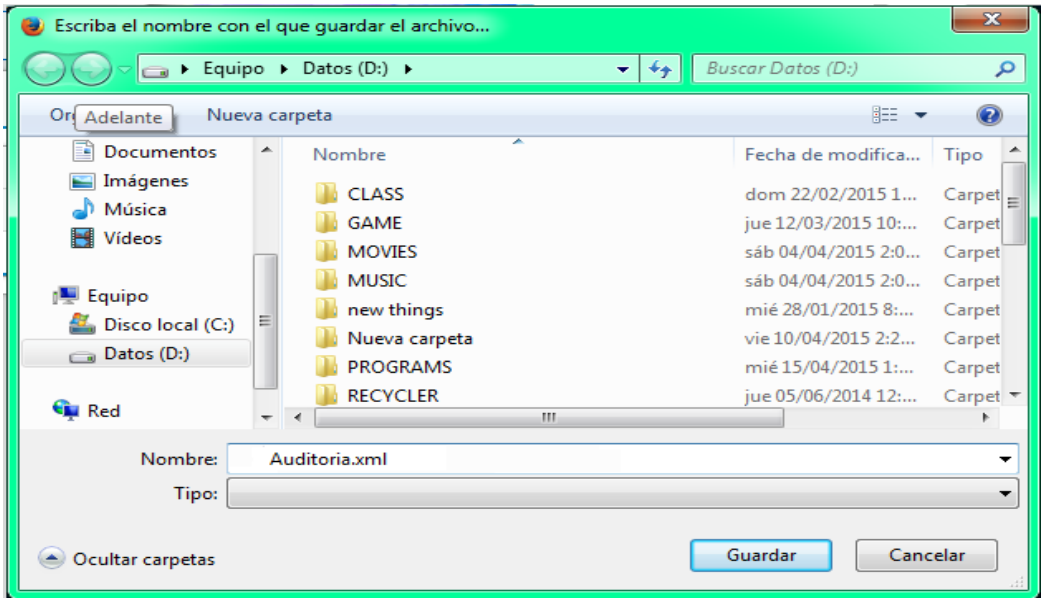
Historia de Usuario	
Número: 9	**Usuario:** Administrador de BD
Nombre de historia: Guardar fichero de resultado	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 3
Programador(es) responsable(es): Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: El fichero de resultados se guardará en la dirección que el usuario determine.	
Observaciones: Debe haberse creado anteriormente el fichero de resultados.	
Interfaz: 	

Tabla 20: HU Mostrar datos de la auditoría

Historia de Usuario	
Número: 10	**Usuario:** Administrador de BD
Nombre de historia: Mostrar datos de la auditoría	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 3
Programador(es) responsable(es): Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: Se mostrarán los datos de las auditorías previamente realizadas, como son: fecha,	

usuario, dirección ip, gestor de bases de datos, sistema operativo.
Observaciones: Debe haberse realizado el proceso de auditoría.
Interfaz:

ANEXO II

Tabla 21: Tarjetas CRC Clase: ConexionPostgreSQL

Nombre de la clase: ConexionPostgreSQL	
Responsabilidad:	**Colaborador:**
Es la encargada de realizar la conexión con el SGBD PostgreSQL, así como de realizar las consultas SQL contenidas en los parámetros del script de revisión a dicho gestor.	

Tabla 22: Tarjetas CRC Clase: ConexionMySQL

Nombre de la clase: ConexionMySQL	
Responsabilidad:	**Colaborador:**
Es la encargada de realizar la conexión con el SGBD MySQL, así como de realizar las consultas SQL contenidas en los parámetros del script de revisión a dicho gestor.	

Tabla 23: Tarjetas CRC Clase: ConexionSQLServer

Nombre de la clase: ConexionMSSQLServer	
Responsabilidad:	**Colaborador:**
Es la encargada de realizar la conexión con el SGBD Microsoft SQL Server, así como de realizar las consultas SQL contenidas en los parámetros del script de revisión a dicho gestor.	

Tabla 24: Tarjetas CRC Clase: ConexionOracle

Nombre de la clase: ConexionOracle	
Responsabilidad:	**Colaborador:**

Es la encargada de realizar la conexión con el SGBD Oracle, así como de realizar las consultas SQL contenidas en los parámetros del script de revisión a dicho gestor.	

Tabla 25: Tarjetas CRC Clase: ConexionSOWindows

Nombre de la clase: ConexionSOWindows	
Responsabilidad:	**Colaborador:**
Es la encargada de realizar la conexión con el SO Windows, así como de ejecutar los comandos de tipo Shell contenidos en los parámetros del script de revisión en dicho sistema.	

Tabla 26: Tarjetas CRC Clase: ConexioSOLninux

Nombre de la clase: ConexionSOLinux	
Responsabilidad:	**Colaborador:**
Es la encargada de realizar la conexión con el SO Linux, así como de ejecutar los comandos de tipo Shell contenidos en los parámetros del script de revisión en dicho sistema.	

Tabla 27: Tarjetas CRC Clase: Indicador

Nombre de la clase: Indicador	
Responsabilidad:	**Colaborador:**
Contiene uno de los indicadores que se le extraen al script de revisión, uno de sus atributos es la lista de parámetros que posee en el script.	Parametro

Tabla 28: Tarjetas CRC Clase: Parametro

Nombre de la clase: Parametro	
Responsabilidad:	**Colaborador:**

Representa a un parámetro del script de revisión.	

ANEXO III

Tabla 29: Tareas de ingeniería #4 Conectar a la BD.

Tareas de ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 4
Nombre de Tarea: Conectar a la BD.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 13/04/2015	**Fecha fin:** 17/04/2015
Programador Responsable: Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: El administrador puede introducir un puerto o aceptar uno que el sistema le proporciona mediante el cual puede realizar la conexión a la base de datos. El sistema pide las credenciales, que son usuario, contraseña y dirección ip.	

Tabla 30: Tareas de ingeniería #5 Conectar con SO si el script es tipo Shell

Tareas de ingeniería	
Número Tarea: 5	**Número Historia de Usuario:** 5
Nombre de Tarea: Conectar con SO si el script es tipo Shell.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.4
Fecha inicio: 20/04/2015	**Fecha fin:** 21/04/2015
Programador Responsable: Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: El sistema, luego de cambiar los valores de los parámetros correspondientes a las credenciales de la base de datos, verifica el sistema operativo que posee el script dentro de sus parámetros. Si el SO es Linux el sistema realiza la conexión mediante SSH y si el SO el Windows la realiza mediante Telnet. Luego el sistema pide las credenciales del sistema operativo, o sea usuario y contraseña para finalmente realizar la conexión con el SO.	

Tabla 31: Tareas de ingeniería #6 Conectar con SO si el script es de tipo SQL

Tareas de ingeniería	
Número Tarea: 6	**Número Historia de Usuario:** 5
Nombre de Tarea: Conectar con SO si el script es de tipo SQL.	

Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.4
Fecha inicio: 22/04/2015	**Fecha fin:** 23/04/2015
Programador Responsable: Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: Según el gestor de bases de datos que va a ser objeto de la auditoría el sistema le proporciona al usuario un puerto mediante el cual puede realizar la conexión con el SO.	

Tabla 32: Tareas de ingeniería #7 Conectar con SO si el script es de tipo Baseline o Nulo

Tareas de ingeniería	
Número Tarea: 7	**Número Historia de Usuario:** 5
Nombre de Tarea: Conectar con SO si el script es de tipo Baseline o Nulo.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha inicio: 24/04/2015	**Fecha fin:** 24/04/2015
Programador Responsable: Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: El sistema copia los parámetros que posee el script para colocarlos en el fichero de resultados, luego se realiza directamente la conexión con el SO.	

Tabla 33: Tareas de ingeniería #8 Ejecutar script XML

Tareas de ingeniería	
Número Tarea: 8	**Número Historia de Usuario:** 6
Nombre de Tarea: Ejecutar script XML.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 1
Fecha inicio: 27/04/2015	**Fecha fin:** 02/05/2015
Programador Responsable: Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: El sistema ejecuta el fichero XML para realizar de esta manera la auditoría la BD o al SO según corresponda. El usuario debe comprobar que exista conexión con la base de datos para poder realizar dicho proceso.	

Tabla 34: Tareas de ingeniería #9 Cambiar valores de parámetros

Tareas de ingeniería	
Número Tarea: 9	**Número Historia de Usuario:** 7
Nombre de Tarea: Cambiar valores de parámetros.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.4

Fecha inicio: 04/05/2015	**Fecha fin:** 08/05/2015
Programador Responsable: Enrique Muschett Cortina, Claudia Vazquez Figueroa.	
Descripción: El sistema cambia los valores que tienen en el script las variables que identifican las credenciales de la base de datos por las variables adbuser, adbpass, adbip y adbport.	

Tabla 35: Tareas de ingeniería #6 Elaborar fichero de resultado

Tareas de ingeniería	
Número Tarea: 10	**Número Historia de Usuario:** 8
Nombre de Tarea: Elaborar fichero de resultado.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha inicio: 11/05/2015	**Fecha fin:** 15/05/2015
Programador Responsable: Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: El sistema recoge los datos que arroja la consulta a la base de datos, luego el usuario crea un documento donde se van a colocar los datos obtenidos, el cual va a contener el resultado de la auditoría.	

Tabla 36: Tareas de ingeniería #7 Guardar fichero de resultado

Tareas de ingeniería	
Número Tarea: 11	**Número Historia de Usuario:** 9
Nombre de Tarea: Guardar fichero de resultado.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.4
Fecha inicio: 18/05/2015	**Fecha fin:** 19/05/2015
Programador Responsable: Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: El usuario selecciona un directorio donde desea que sea guardado el fichero XML con los resultados de la auditoría a la base de datos. El sistema debe validar que este fichero haya sido creado anteriormente.	

Tabla 37: Tareas de ingeniería #8 Mostrar datos de la auditoría

Tareas de ingeniería	
Número Tarea: 12	**Número Historia de Usuario:** 10
Nombre de Tarea: Mostrar datos de la auditoría.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.6

Fecha inicio: 20/05/2015	**Fecha fin:** 22/05/2015
Programador Responsable: Enrique Muschett Cortina y Claudia Vazquez Figueroa.	
Descripción: El sistema muestra los datos referentes a las auditorías realizadas como son la fecha en que se realizó la auditoría, el nombre de usuario del sistema operativo, la dirección ip, el gestor de bases de datos que se auditó y el sistema operativo.	

ANEXO IV

Tabla 38: Prueba de Aceptación #2: HU 2: Mostrar parámetros

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba
El usuario accede al módulo (RASGBD), y selecciona la opción (Mostrar parámetros), luego selecciona el que estime conveniente o presiona el botón (marcar todos) para seleccionar todos los parámetros y presiona el botón (Seleccionar parámetros).		El sistema muestra la interfaz (Seleccionar parámetros).	Satisfactorio.
	El usuario accede al módulo (RASGBD), y selecciona la opción (Mostrar parámetros), luego no selecciona ningún parámetro y presiona el botón	El sistema muestra un error “Debe seleccionar al menos un parámetro”.	

	(Seleccionar parámetros).		
	El usuario accede al módulo (RASGBD), y selecciona la opción (Mostrar parámetros), luego presiona el botón (Seleccionar parámetros)	El sistema muestra un mensaje de error ¨Debe seleccionar el (los) parámetro (s) ¨.	

Prueba de Aceptación #3: HU 3: Seleccionar parámetros

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba
El usuario accede al módulo (RASGBD), y selecciona la opción (Seleccionar parámetros), luego selecciona el (los) parámetro (s) que estime conveniente o presiona el botón (marcar todos) para seleccionar todos los parámetros y presiona el botón (Filtrar parámetros).		El sistema muestra un mensaje ¨Se han filtrado correctamente los parámetros¨	Satisfactorio.
	El usuario accede al módulo (RASGBD), y	El sistema muestra	

	selecciona la opción (Seleccionar parámetros), luego no selecciona ningún parámetro.	un error "Debe seleccionar al menos un parámetro".	

Tabla 39: Prueba de Aceptación #4: HU 4: Conexión con SGBD según el tipo de script

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observación
El usuario accede al módulo (RASGBD), y selecciona la opción (Conexión a SGBD), luego llena los campos usuario, contraseña, puerto, introduce correctamente los campos requeridos y presiona el botón (Conectar).		El sistema se conecta satisfactoriamente al sistema gestor de base de datos y muestra un mensaje "Se ha conectado satisfactoriamente al SGBD".	Satisfactorio	
	El usuario accede al módulo (RASGBD), y selecciona la	El sistema muestra un error "Por favor inserte un valor(es) para este campo(s)".	No se adicionan los datos de la conexión al SGBD.	

	opción (Conexión a SGBD), no introduce todos los campos y presiona el botón (Conectar).			
	El usuario accede al módulo (RASGBD), y selecciona la opción (Conexión a SGBD) e introduce datos en formato desconocido para el campo (Usuario) y presiona el botón (Conectar).	El sistema muestra un error “Por favor inserte nuevamente el nombre de usuario”.	No se adicionan los datos de la conexión al SGBD.	

	El usuario accede al módulo (RASGBD), y selecciona la opción (Conexión a SGBD) e introduce datos en formato desconocido para el campo (Contraseña) y presiona el botón (Conectar).	El sistema muestra un error “Por favor inserte nuevamente la contraseña”.	No se adicionan los datos de la conexión al SGBD.	
	El usuario accede al módulo (RASGBD), y selecciona la opción (Conexión a SGBD) e introduce datos en formato desconocido para el campo (Puerto) y presiona el botón (Conectar).	El sistema muestra un error “Por favor inserte nuevamente el puerto”.	No se adicionan los datos de la conexión al SGBD.	

	El usuario accede al módulo (RASGBD), y selecciona la opción (Conexión a SGBD) y presiona el botón (Conectar), pero el sistema no tiene conexión con el gestor de bases de datos.	El sistema no responde.	El sistema no se conecta al gestor de bases de datos.	

Tabla 40: Prueba de Aceptación #5: HU 5: Conexión a SO

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observación
El usuario accede al módulo (RASGBD), y selecciona la opción (Conexión a SO), luego llena los campos usuario, contraseña puerto e introduce correctamente		El sistema se conecta satisfactoriamente al sistema operativo y muestra un mensaje “Se ha conectado satisfactoriamente al SO” y muestra la interfaz (Realizar Auditoría).	Satisfactorio.	

los campos requeridos y presiona el botón (Conectar).				
	El usuario accede al módulo (RASGBD), y selecciona la opción (Conexión a SO), no introduce todos los campos y presiona el botón (Conectar).	El sistema muestra un error “Por favor inserte un valor(es) para este campo(s)”.	No se adicionan los datos de la conexión al SO.	
	El usuario accede al módulo (RASGBD), selecciona la opción (Conexión a SO), introduce datos en formato desconocido para el campo (Usuario) y presiona el botón (Conectar).	El sistema muestra un error “Por favor inserte nuevamente el nombre de usuario”.	No se adicionan los datos de la conexión al SO.	

	El usuario accede al módulo (RASGBD), selecciona la opción (Conexión a SO) e introduce datos en formato desconocido para el campo (Contraseña) y presiona el botón (Conectar).	El sistema muestra un error “Por favor inserte nuevamente la contraseña”.	No se adicionan los datos de la conexión al SO.	
	El usuario accede al módulo (RASGBD), selecciona la opción (Conexión a SO) e introduce datos en formato desconocido para el campo (Puerto) y presiona el botón (Conectar).	El sistema muestra un error “Por favor inserte nuevamente el puerto”.	No se adicionan los datos de la conexión al SO.	
	El usuario accede al módulo (RASGBD),	El sistema no responde.	El sistema no se conecta al sistema operativo.	

	selecciona la opción (Conexión a SO), pero el sistema no tiene conexión con el sistema operativo y presiona el botón (Conectar).			

Tabla 41: Prueba de Aceptación #6: HU 9: Guardar fichero

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba
El usuario accede al módulo (Realizar Auditoría), y selecciona la opción (Guardar Fichero), luego selecciona el directorio donde lo va a guardar y presiona el botón (aceptar).		El sistema muestra un mensaje ¨Se realizado correctamente la auditoría¨ y muestra la interfaz (Guardar Fichero).	Satisfactorio.

	El usuario accede al módulo (Realizar Auditoría), y selecciona la opción (Guardar Fichero), luego no selecciona el directorio donde lo va a guardar y presiona el botón (aceptar).	El sistema muestra un error “Seleccione la dirección donde desea guardar el fichero”.	
	El usuario accede al módulo (Realizar Auditoría) y no selecciona la opción (Guardar Fichero).	El sistema no guarda el fichero de resultados.	