Universidad de las Ciencias Informáticas

Facultad 2

Herramienta Colaborativa en su Versión Portable para dispositivos móviles con Sistema Operativo Android

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autores:

Alexey Rodríguez Ávila

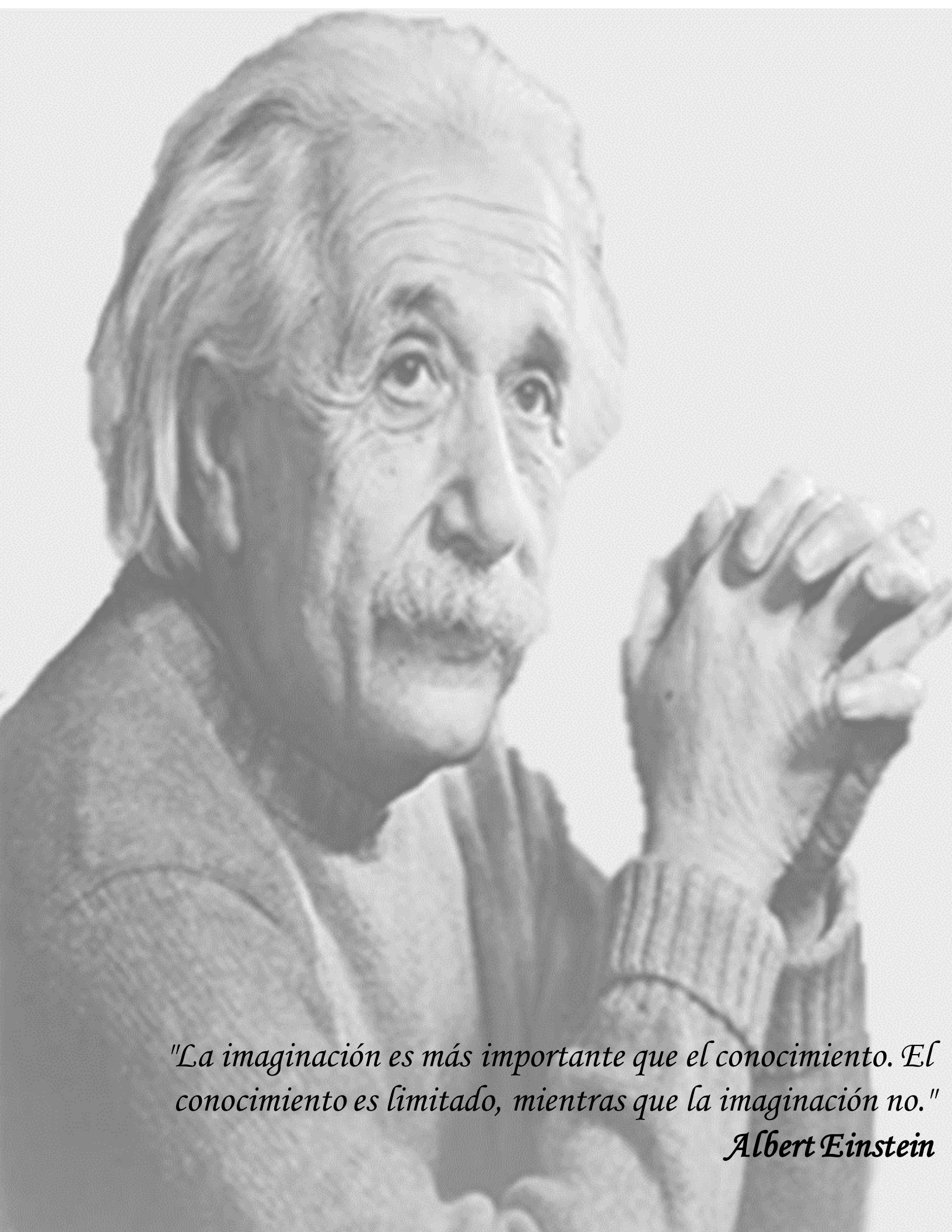
Arián Daniel Rodriguez Salgado

Tutores:

Ing. Yoanny Torres Rubio

Ing. Fernando Ricardo Romero

La Habana, junio, 2015



"La imaginación es más importante que el conocimiento. El conocimiento es limitado, mientras que la imaginación no."

Albert Einstein

Agradecimientos

Arian Daniel

Quiero agradecerles:

A mi madre, la persona más especial y más quiero en mi vida, por apoyarme siempre, por ayudarme en todo lo que estuvo a su alcance en cada momento, por su confianza y creer en mí.

A mi papá que es mi ejemplo a seguir en todos los aspectos de la vida.

A mi novia Lisy por su paciencia, por su amor, por su apoyo incondicional en todos los momentos de nuestra linda relación.

A mis hermanas, a mis tías y tío German que siempre ha sido mi segundo padre, a mis amigos de Holguín y todos los amigos que he hecho en mi vida universitaria.

A los profesores que me ayudaron en mi carrera, sin importarles mi carácter.

Dedicatoria

Arian Daniel

Quiero dedicar este trabajo diploma a toda mi familia, especialmente a mi madre que es la persona que más me ha ayudado en mi vida, sin ella este sueño no hubiese podido hacerse realidad

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo a 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 _____.

_____ _____

Alexey Rodríguez Ávila

Firma Autor

Arian Daniel Rodriguez Salgado

Firma Autor

_____ _____

Yoanny Torres Rubio

Firma Tutor

Fernando Ricardo Romero

Firma Tutor

Datos del contacto

Tutor

Nombre y Apellidos: Yoanny Torres Rubio

Sexo: M. **Institución:** UCI.

Dirección de la institución: carretera a San Antonio de los Baños, km 2 ½, Boyeros, Ciudad de La Habana.

Correo electrónico: ytrubio@uci.cu **Teléfono del trabajo:**

Área:

Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas.

Año de graduación:

Institución donde se graduó: Universidad de las Ciencias Informáticas.

Tutor

Nombre y Apellidos: Fernando Ricardo Romero

Sexo: M. **Institución:** UCI.

Dirección de la institución: carretera a San Antonio de los Baños, km 2 ½, Boyeros, Ciudad de La Habana.

Correo electrónico: fricardo@uci.cu **Teléfono del trabajo:**

Área:

Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas.

Año de graduación: 2014

Institución donde se graduó: Universidad de las Ciencias Informáticas.

Resumen

Con el continuo avance tecnológico, son cada vez más los sistemas informáticos que generan un cúmulo substancial de información, que es necesario almacenar y proteger en Base de Datos (BD); estas a su vez son gestionadas mediante los Sistemas Gestores de Bases de Datos (SGBD). El control, chequeo y supervisión que se realiza por parte de los especialistas en Seguridad Informática sobre los SGBD, se ha convertido, con el creciente desarrollo de la tecnología, en una tarea de ejecución imprescindible en todo sistema informático. El desarrollo de estos chequeos no debe ser una rutina de ejecución aislada y casual, sino que debe haber un control persistente para contribuir a la confiabilidad en los SGBD. La necesidad de auditar estos sistemas se muestra como una tarea imprescindible para garantizar la confidencialidad, integridad y disponibilidad de los datos almacenados. Teniendo en cuenta lo anteriormente expuesto, se propone el desarrollo de una aplicación que permita realizar el monitoreo de la seguridad de los servidores de BD de una empresa.

En el documento se encuentran las principales características de otros sistemas encargados de la ejecución de auditorías a SGBD, que se consultaron en la investigación. Se seleccionan las herramientas y tecnologías idóneas y se guía el proceso de desarrollo conforme a lo establecido por la metodología XP. Como resultado de la investigación se obtiene una aplicación que permite el monitoreo automatizado de la seguridad de la información, para dispositivos móviles con Sistema Operativo (SO) Android.

Palabras clave: Android, auditoría, monitoreo, seguridad.

Índice de contenidos

Índice de Tablas

Índice de Figuras

Introducción

El continuo avance tecnológico que envuelve al mundo, hace ineludible la inclusión de las Tecnologías de la Información y las Comunicaciones (TIC) en todas las esferas de la sociedad. El desarrollo en el campo de las TIC, aparejado al protagonismo que ha tomado la inserción de sistemas informáticos en los procesos empresariales, conlleva a que el volumen de información que se genera crezca cada vez más. Debido a la importancia que posee esta información para el correcto funcionamiento de cualquier empresa, se hace necesario almacenarla y protegerla, empleándose para esto las Bases de Datos (BD) y a su vez los Sistemas Gestores de Bases de Datos (SGBD).

La gran difusión de los SGBD, junto con la consagración de los datos como uno de los recursos fundamentales de las empresas, ha hecho que los temas relativos a su seguridad cobren cada día mayor interés. En este sentido, los SGBD ofrecen mecanismos para la gestión de la seguridad de las BD. Estos mecanismos, no inmunizan a los SGBD de los efectos que provocan los ataques llevados a cabo por piratas informáticos que pueden falsear, robar, manipular y tener el control total de los datos almacenados por empresas, organizaciones o gobiernos. Motivo por el cual, se ha incrementado la búsqueda de soluciones informáticas, que brinden la posibilidad de auditar la seguridad de sus servidores de BD, con el objetivo de conocer sus puntos débiles y tratar de erradicarlos. Una de las soluciones informáticas encaminadas a este propósito es el Sistema para la realización de Auditorías a Sistemas Gestores de Bases Datos (SASGBD).

El SASGBD es un software desarrollado en la Universidad de las Ciencias Informáticas (UCI), este forma parte del proyecto AuditBD, perteneciente al centro Telemática de la Facultad 2. Dicho sistema está compuesto por dos módulos: el Módulo de Gestión de Auditoría a SGBD (MGASGBD) y el módulo Herramienta Colaborativa en su Versión Portable (HCVP). El MGASGBD es el encargado de realizar el diagnóstico del riesgo de la seguridad de la información hospedada en los SGBD, para ello conforma un archivo de Lenguaje de Marcado Extensible (XML, por sus siglas en inglés) con toda la información referente al SGBD y al Sistema Operativo (SO), que se someterán a revisión, así como los aspectos que se chequearán en esta. Por otra parte, el funcionamiento del módulo HCVP está enfocado al monitoreo del SGBD y del SO, objetos de la auditoría, ya que es el encargado de importar el archivo generado por el otro módulo, ejecutar las sentencias contenidas en este archivo y guardar los resultados obtenidos para su

posterior análisis. El módulo HCVP ofrece además la opción de filtrar un Reporte de Baseline, si existen aspectos que deben medirse por Microsoft Baseline Security Analizer[1].

Durante la ejecución del proceso de auditoría, el módulo HCVP establece una conexión al SGBD y al SO sobre los que se ejecutará la auditoría. La conexión al SO puede ser por los protocolos Telnet[2] o SSH[3]. Cuando el módulo establece una conexión por SSH se abre un nuevo canal por cada sentencia a ejecutar, esto provoca que sea imposible acceder a la información generada por otras consultas ejecutadas previamente, dando lugar a la pérdida de datos durante la ejecución de las consultas planificadas, y por consiguiente, que no se realice un completo monitoreo a la seguridad del SGBD y el SO, propósitos de la auditoría.

Actualmente se está desarrollando una nueva versión del módulo MGASGBD. Una de las características que presenta esta nueva versión es una estructura diferente para el fichero de extensión XML generado. Este cambio en la estructura del fichero hace imposible que se integren la nueva versión del módulo MGASGBD, con la versión actual del módulo HCVP, ya que este último no es capaz de interpretar los aspectos que se chequearán en el proceso de auditoría contenidos en el fichero XML. Esto trae como consecuencia que no se ejecute un monitoreo a la seguridad del SGBD y el SO, objetivos de la auditoría.

La versión actual del módulo HCVP fue desarrollada, por el proyecto AuditBD, para hacer uso de los ordenadores en el proceso de auditoría, es por ello que esta versión exige disponer de acceso a un ordenador directamente conectado a la red interna de la institución donde se llevará a cabo este proceso. Esto hace que, en ambientes donde no se tiene acceso a un ordenador, sea imposible realizar un monitoreo sobre el SGBD y el SO, sobre los que se efectúa el proceso de auditoría. Por esta razón, el proyecto AuditBD, ha valorado la utilización de dispositivos móviles para efectuar el monitoreo; teniendo

[1] Microsoft Baseline Security Analyzer: es una herramienta de software lanzado por Microsoft para determinar el estado de seguridad mediante la evaluación de las actualizaciones de seguridad que faltan y las configuraciones de seguridad menos seguras dentro de Microsoft Windows y sus componentes, como Internet Explorer, IIS servidor web, productos Microsoft SQL Server y Microsoft Office.

[2] Telnet (Teletype Network): es el nombre de un protocolo de red que permite viajar a otra máquina para manejarla remotamente. A la máquina que se acceda debe tener un programa especial que reciba y gestione las conexiones. El puerto que se utiliza generalmente es el 23.

[3] SSH (Secure SHell): es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo la computadora mediante un intérprete de comandos en sistemas Unix. Permite copiar datos de forma segura (tanto archivos sueltos como simular sesiones FTP cifradas), gestionar claves RSA para no escribir claves al conectar a los dispositivos y pasar los datos de cualquier otra aplicación por un canal seguro tunelizado mediante SSH.

en cuenta, la simplicidad que brindan estos terminales a los usuarios, en la realización de tareas de baja, mediana y alta complejidad.

La utilización de los dispositivos móviles proporciona a los usuarios usabilidad, portabilidad y conectividad, cambiando radicalmente el entorno tecnológico de las empresas, y la forma de trabajar por parte de los empleados. Colaboración, productividad, movilidad y competitividad, son algunas de las ventajas que puede aportar a un especialista en Seguridad Informática, el empleo activo de la tecnología móvil en situaciones donde sea imposible acceder a un ordenador, para mantener así un completo monitoreo a la seguridad del SGBD y el SO, sobre los que se efectúa el proceso de auditoría.

Partiendo de la problemática anterior, se define como **problema a resolver**: ¿Cómo contribuir al acabado del proceso de monitoreo de la seguridad de la información del módulo HCVP, del SASGBD?

Se identifica como **objeto de estudio**: el proceso de auditoría de seguridad informática a SGBD.

Para darle solución al problema planteado se traza como **objetivo general**: desarrollar la versión 2.0 del módulo HCVP para dispositivos móviles con SO Android, contribuyendo al acabado del proceso de monitoreo de la seguridad de la información del mismo.

Centrándose en el **campo de acción**: las auditorías de seguridad informática a SGBD, empleando dispositivos móviles con SO Android.

Como tareas de la investigación se proponen las siguientes:

1. Analizar aplicaciones, a nivel nacional e internacional, enfocadas al proceso de auditoría de seguridad informática a SGBD, estableciendo similitudes con la investigación en curso.
2. Diseñar la aplicación que permita realizar un monitoreo de la seguridad de la información a SGBD desde dispositivos móviles con SO Android, cumpliendo con las exigencias del cliente.
3. Valorar las herramientas, metodologías y tecnologías, seleccionando las necesarias para el desarrollo de la aplicación.
4. Desarrollar una aplicación que permita realizar un monitoreo de la seguridad de la información a SGBD desde dispositivos móviles con SO Android, solucionando el problema planteado.
5. Realizar pruebas a la aplicación desarrollada, verificando su correcto funcionamiento.

Para dar cumplimiento al objetivo propuesto, se utilizan diferentes métodos teóricos y empíricos. Estos métodos permiten la modelación del objeto de la investigación, el examen de la aplicación en su fundamentación y el arribo a consideraciones teóricas planteadas en el transcurso del proceso investigativo.

Métodos teóricos:

- Histórico Lógico: se utiliza para realizar un estudio del estado del arte de los principales sistemas informáticos auditores de SGBD, así como las tendencias y tecnologías en el desarrollo de este tipo de sistemas.
- Analítico-sintético: se utiliza para procesar la información del negocio de la organización y arribar a las conclusiones de la investigación. Además se analiza y se comprende la teoría y documentación relacionada con las auditorías a SGBD, permitiendo así, extraer los elementos coherentes e importantes. Por último, se emplea análisis de características positivas y negativas del módulo HCVP, con el fin de lograr reunir el conjunto de cualidades realmente útiles que este posee.

Métodos Empíricos:

- Entrevista: este método brinda apoyo a la incorporación de conocimientos mediante las entrevistas no estructuradas efectuadas a los especialistas de áreas de TLM, responsables del desarrollo de la HCVP.

El presente trabajo consta de cuatro capítulos, los cuales estarán estructurados de la siguiente forma:

Capítulo 1: Fundamentación Teórica, se exponen de forma general los aspectos teóricos de la investigación. Se analizan las herramientas, metodología de desarrollo y lenguajes de programación favorables para la implementación de la solución. Además se realiza un estudio del estado del arte y se vincula con la aplicación.

Capítulo 2: Características de la Aplicación, se describen las características de la aplicación a desarrollar, se inician las fases de Exploración y Planificación, y se definen las restricciones y funcionalidades, especificadas por el cliente, que tendrá la aplicación.

Capítulo 3: Análisis y diseño de la Aplicación, se define la arquitectura de la solución y los patrones de diseño. Se describen las tarjetas Clase-Responsabilidad-Colaboradores para garantizar una adecuada organización.

Capítulo 4: Implementación y Prueba, se exponen las principales características del proceso de implementación. Se desarrollan las tareas de ingenierías correspondientes a las Historias de Usuario, además se realizan las pruebas que validan que el producto final cumpla con requerimientos definidos previamente.

Capítulo 1: Fundamentación Teórica

Introducción

En este capítulo se analizan conceptos asociados a la investigación, tecnologías y metodologías utilizadas para la implementación de la aplicación, así como el estado del arte, haciendo referencia a las tendencias actuales de estos sistemas.

1.1 Base de Datos

Conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. Puede considerarse una colección de datos variables en el tiempo. (1)

1.2 Sistema gestor de Bases de Datos

El software que permite manejar los datos almacenados en una o varias bases de datos, por uno o varios usuarios, desde diferentes puntos de vista y a la vez. Tiene como objetivo fundamental suministrar al usuario las herramientas que le permitan manipular, en términos abstractos, los datos (información). (1)

1.3 Auditorías Informáticas

La auditoría informática es un proceso llevado a cabo por profesionales especialmente capacitados para el efecto, y que consiste en recoger, agrupar y evaluar evidencias para determinar si un sistema de información salvaguarda el activo empresarial, mantiene la integridad de los datos, lleva a cabo eficazmente los fines de la organización, utiliza eficientemente los recursos, y cumple con las leyes y regulaciones establecidas. Permite detectar de forma sistemática el uso de los recursos y los flujos de información dentro de una organización y determinar qué información es crítica para el cumplimiento de su misión y objetivos, identificando necesidades, duplicidades, costes, valor y barreras, que obstaculizan flujos de información eficientes. (2)

1.4 Tipos de Auditorias (3)

- ✓ **Auditoría de la gestión**: la contratación de bienes y servicios, documentación de los programas, etc.
- ✓ **Auditoría de los datos**: clasificación de los datos, estudio de las aplicaciones y análisis de los flujogramas.
- ✓ **Auditoría de las bases de datos**: controles de acceso, de actualización, de integridad y calidad de los datos.

- ✓ **Auditoría de la seguridad**: referidos a datos e información, verificando disponibilidad, integridad, confidencialidad, autenticación y no repudio.
- ✓ **Auditoría de la seguridad física**: referido a la ubicación de la organización, evitando ubicaciones de riesgo, y en algunos casos no revelando la situación física de esta. También está referida a las protecciones externas (arcos de seguridad, CCTV, vigilantes, etc.) y protecciones del entorno.
- ✓ **Auditoría de la seguridad lógica**: comprende los métodos de autenticación de los sistemas de información.
- ✓ **Auditoría de las comunicaciones**: se refiere a la auditoría de los procesos de autenticación en los sistemas de comunicación.
- ✓ **Auditoría de la seguridad en producción**: frente a errores, accidentes y fraudes.

1.5 Auditorías de Seguridad Informáticas a SGBD

Una auditoría de seguridad informática a SGBD, se puede definir como el estudio que comprende el análisis y gestión de sistemas para identificar y corregir las vulnerabilidades y fallas de seguridad que pudieran presentarse en una revisión exhaustiva de las estaciones de trabajo, los servidores y las redes de comunicaciones a servidores de BD. Durante una auditoría de seguridad informática se realizan las auditorías de la seguridad lógica y auditoría de las comunicaciones. (4)

1.6 Estado del Arte

Existen actualmente varias herramientas que dan soporte al proceso de auditoría de seguridad informática sobre SGBD. Hacer un correcto uso de estas y un estudio de su funcionamiento, permite mantener un estricto control sobre posibles vulnerabilidades y brechas de seguridad que pudieran presentarse en SGBD. A continuación se presentan algunas de estas herramientas.

DB Audit Expert

DB Audit Expert, es una herramienta de seguridad y de auditoría para los SGBD Oracle, SyBase, DB2, MySQL y Microsoft SQL Server. Esta herramienta permite a los administradores de seguridad o auditores, realizar un seguimiento y análisis de las BD. Entre sus principales características se encuentra la gestión de los informes de los resultados de auditorías, reduciendo en gran medida la cantidad de datos, identificando mejor las violaciones de seguridad y proporcionando mayores detalles que en los sistemas nativos. (5)

Entre las deficiencias que posee DB Audit Expert está la carencia de soporte para PostgreSQL y para otros Sistemas Operativos (SO) distintos de Windows.

SQL Server Audit

SQL Server Audit se encarga de recopilar una única instancia de acciones y grupos de acciones a nivel de servidor o de base de datos para su supervisión. Los resultados de una auditoría se envían a un destino, que puede ser un archivo, el registro de eventos de seguridad de Windows o el registro de eventos de aplicación Windows. (6)

Entre las principales desventajas se encuentran que solo se puede encontrar gratis versiones de prueba, además solo esta implementado para sistemas operativos Windows y el SGBD SQL Server.

Oracle® Audit Vault

Es una solución completa para la auditoría de base de datos y control de actividades, que ofrece características de creación de informes y alertas. Automatiza el proceso de auditoría de la base de datos y cuenta con características (7):

- Programación de informes, notificación y autenticación, que pueden ayudar a las empresas a bajar el costo que implica cumplir con las obligaciones de privacidad y protección de datos internos y externos.
- Informes de permisos y privilegios con copias actualizadas de los usuarios, privilegios y perfiles de Oracle Database, que permiten a los auditores controlar los cambios al acceso de la base de datos.
- Limpieza automática de los datos del proceso de auditoría desde bases de datos Oracle y de otros proveedores, una vez que los datos de auditorías se han consolidado en forma segura en el repositorio Oracle Audit Vault. Esto ayuda a reducir los costos operativos de auditar la base de datos.

Oracle® Audit Vault es una herramienta privativa que, aunque es de las mejores en el mercado, no es posible su utilización debido a que está enfocada solomente a SGBD Oracle.

AppDetectivePro

Herramienta de análisis de vulnerabilidades en BD. Es un escáner de red y evaluador de vulnerabilidades. Detecta las aplicaciones de BD existentes en la infraestructura de las empresas y evalúa sus

características de seguridad. Con el respaldo de una probada metodología y un amplio conocimiento de las vulnerabilidades a nivel de aplicación. AppDetectivePro localiza, examina, reporta y repara brechas de seguridad y configuraciones erróneas. Como consecuencia, las empresas pueden mejorar activamente sus aplicaciones de bases de datos, fortalecer sus operaciones de seguridad y demostrar el cumplimiento de las normas regulatorias. Soporta Bases de Datos: MySQL, Oracle, Sybase, IBM DB2, en Mainframe, Microsoft SQL Server, Oracle Application Server y Lotus Notes/Domino. (8)

No consta con soporte para PostgreSQL y solamente está disponible para la plataforma Microsoft Windows.

Sistema para la realización de Auditorías a Sistemas Gestores de Bases Datos (SASGBD).

El SASGBD es un software desarrollado en UCI, este forma parte del proyecto AuditBD, perteneciente al centro Telemática de la Facultad 2 y tiene como objetivo contribuir a la realización de las auditorías a los SGBD. Unos de los módulos del SASGBD en el HCVP. Este módulo está enfocado al monitoreo de seguridad de la información sobre SGBD.

Entre las características favorables que presenta este sistema, es su implementación en el lenguaje de programación Java, pues mediante la reutilización del código se aprovecha el trabajo anterior y se economiza tiempo. Las principales limitantes que presenta el módulo HCVP es que no completa el proceso de monitoreo y que está desarrollado para Computadoras Personales (PC, por sus siglas en inglés).

Resultado del análisis

De manera general, las herramientas antes mencionadas, no satisfacen las necesidades del cliente, pues son herramientas privativas que no dan soporte para los SGBD PostgresSQL y no dan soporte para la plataforma Linux. Además no existe ninguna aplicación disponible para la plataforma Android que se encargue de este tipo de tareas.

1.7 Tecnologías a utilizar

A medida que se desarrolla la humanidad, junto a ella también de desarrolla la tecnología, por ello para implementar un sistema informático es necesario contar con todas las tecnologías que permitan llegar a una solución del problema de manera que se garanticen, la calidad, confidencialidad y seguridad de la solución.

1.7.1 Comunicaciones Inalámbricas

La comunicación inalámbrica o sin cables es aquella en la que la comunicación (emisor/receptor) no se encuentra unida por un medio de propagación físico, sino que se utiliza la modulación de ondas electromagnéticas a través del espacio. En este sentido, los dispositivos físicos solo están presentes en los emisores y receptores de la señal, entre los cuales se encuentran: antenas, computadoras portátiles, asistente personal digital (PDA), teléfonos móviles, etc. La comunicación inalámbrica, que se realiza a través de ondas de radiofrecuencia, facilita la operación en lugares donde la computadora no se encuentra en una ubicación fija (almacenes, oficinas de varios pisos, etc.). Actualmente se utiliza de una manera general y accesible para todo público. Actualmente las redes cableadas presentan ventaja en cuanto a transmisión de datos sobre las inalámbricas. Mientras que las cableadas proporcionan altas velocidades de transmisión, las inalámbricas no llegan a superarlas. (9)

Wi-Fi

Wi-Fi es una marca de la Wi-Fi Alliance, la organización comercial que adopta, prueba y certifica que los equipos cumplen los estándares 802.11 relacionados a redes inalámbricas de área local. La norma IEEE 802.11 fue diseñada para sustituir el equivalente a las capas físicas y control de acceso al medio (MAC por sus siglas en inglés) de la norma 802.3 (Ethernet). Esto significa que en lo único que se diferencia una red Wi-Fi de una red Ethernet es en cómo se transmiten las tramas o paquetes de datos y por consiguiente la velocidad de transferencia; el resto es idéntico. Por tanto, una red local inalámbrica 802.11 es completamente compatible con todos los servicios de las redes de área local (LAN, por sus siglas en inglés) 802.3 (Ethernet). (10)

Ventajas y desventajas de las redes Wi-Fi

Las redes Wi-Fi poseen una serie de ventajas, entre las cuales se pueden destacar (10):

- Al ser redes inalámbricas, la comodidad que ofrecen es superior a las redes cableadas, porque cualquiera que tenga acceso a la red puede conectarse desde distintos puntos dentro de un rango suficientemente amplio de espacio.
- Una vez configuradas, las redes Wi-Fi, permiten el acceso de múltiples ordenadores sin ningún problema ni gasto en infraestructura, no así en la tecnología por cable.

- La Wi-Fi Alliance asegura que la compatibilidad entre dispositivos con la marca Wi-Fi es total, con lo que en cualquier parte del mundo se puede utilizar la tecnología Wi-Fi con una compatibilidad total.

Pero como red inalámbrica, la tecnología Wi-Fi presenta desventajas como (10):

- Menor velocidad en comparación a una conexión con cables, debido a las interferencias y pérdidas de señal que el ambiente puede acarrear.
- La desventaja fundamental de estas redes existe en el campo de la seguridad. Existen algunos programas capaces de capturar paquetes, trabajando con su tarjeta Wi-Fi en modo promiscuo, de forma que puedan calcular la contraseña de la red, y de esta forma acceder a ella.

1.8 Sistema operativo

Es el programa o conjunto de programas que efectúan la gestión de los procesos básicos de un sistema informático y permite la normal ejecución del resto de las operaciones. Un sistema operativo es un conjunto de programas o software, destinado a permitir la comunicación entre el usuario y la máquina de forma cómoda y eficiente; se encarga de gestionar los recursos del ordenador, esto incluye la gestión del *hardware* desde los niveles más básicos. (11)

Hoy en día existen varios sistemas operativos para dispositivos móviles, entre ellos se puede mencionar Windows Phone, desarrollado por la Microsoft, Symbian OS que es un producto de Nokia. También como SO para dispositivos móviles se encuentra iOS, desarrollado por Apple Inc. que solo puede ser utilizado en dispositivos de este fabricante y BlackBerry OS lanzado al mercado por BlackBerry para sus dispositivos. Además existe Android. (11)

1.9 Android

Android es un sistema operativo orientado a dispositivos móviles. Lo que lo hace diferente es que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma. El sistema permite programar aplicaciones en una variación de Java llamada Dalvik. Este sistema operativo proporciona todas las interfaces necesarias para desarrollar aplicaciones que accedan a las funciones del teléfono (como el GPS, Wi-Fi, Bluetooth) de una forma muy sencilla en un lenguaje de programación muy conocido como es Java, haciendo uso del SDK de Android. (12)

Principales elementos (13)**:**

- Framework de aplicaciones: permite el reemplazo y la reutilización de los componentes. Muchos de estos componentes integrados de forma nativa al SO.
- Máquina virtual Dalvik: máquina virtual especializada, diseñada específicamente para Android y optimizada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados.
- Bluetooth, EDGE, 3g y Wi-Fi: orientado a la conexión, Android tiene soporte para tecnologías de conectividad dependiendo del terminal.
- Pantalla Táctil: las aplicaciones están diseñadas para el uso sobre dispositivos con pantallas táctiles.

Arquitectura:

- Aplicaciones: las aplicaciones base, incluyen un cliente de correo electrónico, programa de mensajería, calendario, mapas, navegador, contactos y otros. La mayoría de estas aplicaciones están escritas en el lenguaje de programación Java.
- Marco de trabajo de aplicaciones: los desarrolladores tienen acceso completo a las mismas API del *framework*, usadas por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.
- Bibliotecas: Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del *framework* de aplicaciones de Android; algunas son: System C Library (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos y SQLite, entre otras.
- Runtime de Android: incluye un *set* de bibliotecas base, que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android ejecuta su propio proceso, con su propia instancia de la máquina virtual Dalvik. Esta ha sido escrita de forma que un dispositivo puede ejecutar múltiples máquinas virtuales de forma eficiente. Dalvik ejecuta archivos en el formato Dalvik Ejecutable (.dex), el cual está optimizado para memoria

mínima. La máquina virtual está basada en registros y ejecuta clases compiladas por el compilador de Java que han sido transformadas al formato .dex por la herramienta incluida "dx".

- Núcleo Linux: Android depende de Linux para los servicios base del sistema como: seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el *hardware* y el resto de la pila de software.

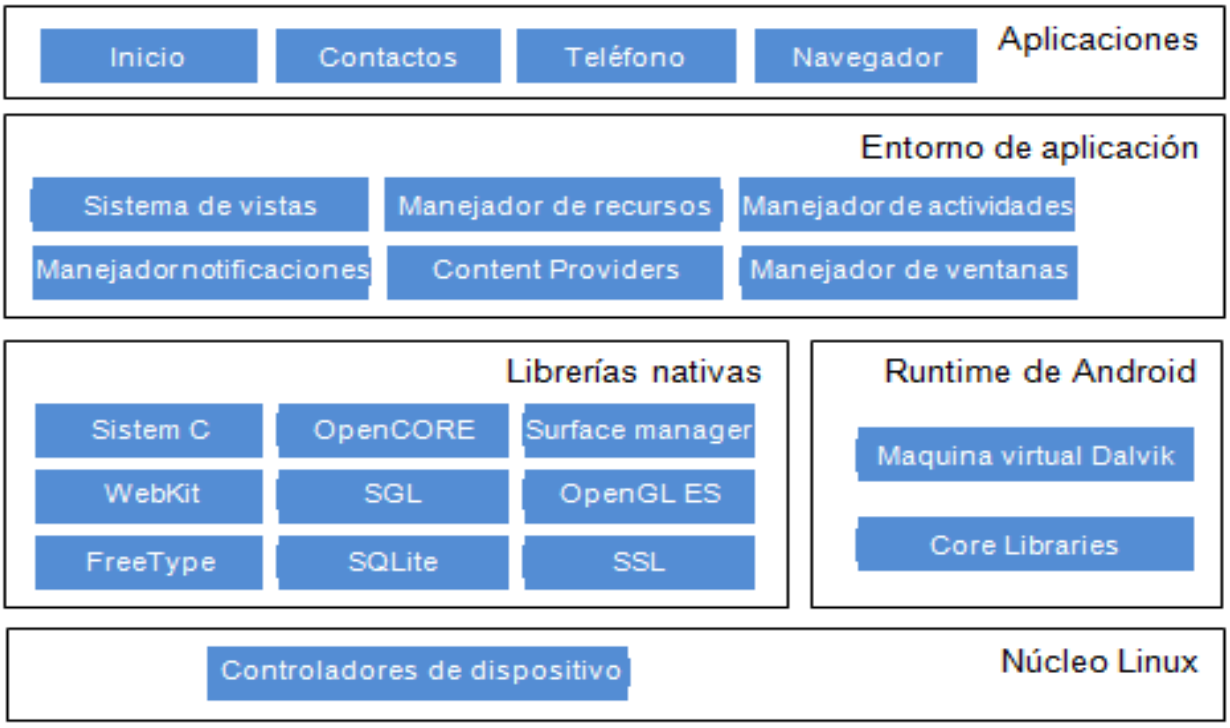


Figura 1. Arquitectura de Android (14)

Descripción de las capas (14)

1. Núcleo Linux:

Android utiliza el núcleo de Linux como una capa de abstracción para el *hardware*, disponible en los dispositivos móviles. Esta capa contiene los *drivers* necesarios para que cualquier componente *hardware* pueda ser utilizado mediante las llamadas correspondientes. Siempre que un fabricante incluye un nuevo elemento de *hardware*, lo primero que se debe realizar para que pueda ser utilizado

desde Android, es crear las librerías de control o *drivers* necesarios dentro de este kernel de Linux, embebido en el propio Android.

2. Entorno de ejecución (*Runtime*) de Android:

Está basado en el concepto de máquina virtual utilizado en Java. Dadas las limitaciones de los dispositivos donde ha de ejecutarse Android (poca memoria y procesador limitado), no fue posible utilizar una máquina virtual de Java estándar. Google tomó la decisión de crear una nueva, la máquina virtual Dalvik, que respondiera mejor a estas limitaciones. A partir de Android 5.0 se reemplaza Dalvik por ART. Esta nueva máquina virtual consigue reducir el tiempo de ejecución del código Java hasta en un 33%.

3. Librerías Nativas:

Incluye un conjunto de librerías en C/C++ usadas en varios componentes de Android. Están compiladas en código nativo del procesador. Muchas de las librerías utilizan proyectos de código abierto. Algunas de estas librerías son:

- System C library: una derivación de la librería BSD de C estándar (libc), adaptada para dispositivos embebidos basados en Linux.
- Media Framework: librería basada en PacketVideo's OpenCORE; soporta *codecs* de reproducción y grabación de multitud de formatos de audio, vídeo e imágenes.
- Surface Manager: maneja el acceso al subsistema de representación gráfica en 2D y 3D.
- WebKit: soporta un moderno navegador web utilizado en el navegador Android y en la vista webview. Se trata de la misma librería que utiliza Google Chrome y Safari.
- SGL: motor de gráficos 2D.
- Librerías 3D: implementación basada en OpenGL ES 1.0 API. Las librerías utilizan el acelerador *hardware* 3D si está disponible, o el software altamente optimizado de proyección 3D.
- FreeType: fuentes en *bitmap* y renderizado vectorial.
- SQLite: potente y ligero motor de bases de datos relacionales disponible para todas las aplicaciones.

- SSL: proporciona servicios de encriptación Secure Socket Layer (capa de conexión segura).

4. Entorno de Aplicación:

Proporciona una plataforma de desarrollo libre para aplicaciones con gran riqueza e innovaciones (sensores, localización, servicios, barra de notificaciones, etc.) Esta capa ha sido diseñada para simplificar la reutilización de componentes. Las aplicaciones pueden publicar sus capacidades y otras pueden hacer uso de ellas (sujetas a las restricciones de seguridad). Este mismo mecanismo permite a los usuarios reemplazar componentes.

5. Aplicaciones:

Este nivel está formado por el conjunto de aplicaciones instaladas en una máquina Android. Todas las aplicaciones han de correr en la máquina virtual Dalvik para garantizar la seguridad del sistema. Normalmente las aplicaciones Android están escritas en Java. Para desarrollar aplicaciones en Java se puede utilizar el Android SDK. Existe otra opción consistente en desarrollar las aplicaciones utilizando C/C++. Para esta opción se puede utilizar el Android NDK (*Native Development Kit*).

1.10 Herramientas y metodologías

El desarrollo del módulo HCVP está orientado a dispositivos móviles con SO Android. Con este fin, se utilizarán las herramientas y metodologías expuestas a continuación:

1.10.1 Entorno de Desarrollo Integrado Android Studio 1.2

Android Studio es el Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) oficial de Android; el cual brinda acceso a un conjunto de funciones que permiten el flujo de trabajo de desarrollo. A continuación se enuncian algunas características (15):

- Experiencia de Inicio:
  - ✓ Asistente de configuración para la primera ejecución: con la instalación inicial se instala el SDK de Android, se establece la configuración del IDE y se crea un emulador optimizado para probar la aplicación. Además, se incluye un conjunto de plantillas de código para iniciar el proceso de desarrollo.

- ✓ Importación de muestras de código y plantillas: Android Studio incluye asistentes que le permiten comenzar con nuevas plantillas de proyecto o ejemplos de código de importación de GitHub.

- Código y edición de recursos, diseño de interfaz de usuario:
  - ✓ Edición de código: Android Estudio aprovecha todas las capacidades de edición de código inteligente de IntelliJ IDEA[4] como el autocompletado avanzado de código, refactorización, y análisis de código.
  - ✓ Edición de Internacionalización: administra traducciones de cadenas de texto de las aplicaciones en Android Studio.
  - ✓ Diseño de la interfaz de usuario: permite editar y pre-visualizar los diseños de Android a través de múltiples tamaños de pantalla, idiomas, e incluso versiones de API.
- Análisis de rendimiento:
  - ✓ Monitor de memoria: permite monitorear el uso de memoria de la aplicación en tiempo real para ayudar a encontrar maneras de mejorar el rendimiento de la aplicación.
- Sistema de construcción unificada:
  - ✓ Android Studio utiliza un sistema de construcción basado en Gradle que proporciona una gran flexibilidad y capacidad de ampliación, así como la posibilidad de construir desde dentro y fuera del IDE. Este sistema de construcción unificada desacopla la construcción del propio Android Studio, lo que significa que las actualizaciones del IDE nunca influyen en la salida de su construcción.

1.10.2 Gradle 2.2.1

Gradle es un sistema de construcción avanzada, así como un conjunto de herramientas que permiten crear una lógica de generación personalizada a través de *plugins*. (16)

Algunas características de Gradle son (16):

- Dominio especifico del idioma (DSL, por sus siglas en inglés) para describir y manipular la lógica de construcción.

[4] IntelliJ IDEA es un IDE para el desarrollo de programas informáticos desarrollado por JetBrains.

- Construcción de archivos que permiten la mezcla de elementos declarativos a través del DSL y el uso de código para proporcionar una lógica personalizada.
- Muy flexible (permite el uso de las mejores prácticas, sin imponer su propia manera de hacer las cosas).
- Los *plugins* pueden exponer su propio DSL y su propia API para construir archivos para su uso.

1.10.3 Lenguaje de programación Java 1.8

Java es un lenguaje orientado a objetos, eso implica que su concepción es muy próxima a la forma de pensar humana. También posee otras características importantes (17):

- Es un lenguaje compilado, que genera ficheros de clases, las que son en realidad interpretadas por la máquina virtual de Java. Siendo esta la que mantiene el control sobre las clases que se estén ejecutando.
- Es un lenguaje multiplataforma: el mismo código escrito en Java que funciona en un SO, funcionará en cualquier otro que tenga instalada la máquina virtual de Java.
- Es un lenguaje seguro: la máquina virtual, al ejecutar el código escrito en Java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros.
- Gracias al API de java se puede ampliar el lenguaje para que sea capaz de, por ejemplo, comunicarse con equipos mediante red, acceder a bases de datos, crear páginas HTML dinámicas, crear aplicaciones visuales.

Teniendo en cuenta las características antes expuestas se decidió utilizar el lenguaje de programación Java, siendo este, el lenguaje recomendado para el desarrollo de aplicación en Android e integrado perfectamente al IDE Android Studio.

1.10.4 Herramientas de Desarrollo de Software

Se utilizará Android SDK en su versión 24.0.1. Esta incluye un conjunto de herramientas de desarrollo y comprende un depurador de código, bibliotecas, un simulador de dispositivos, documentación, ejemplos de código y tutoriales. (18)

Fue elegido el Android SDK como herramienta para el desarrollo de la aplicación por su perfecta integración con el IDE Android Studio, por su soporte extendido por parte de los desarrolladores del SO

Android y por disponer de todas las librerías necesarias para desarrollar aplicaciones en la plataforma Android. Además, es libre, gratuito y de código abierto, disponible en http://developers.android.com/.

1.11 Extreme Programming (XP)

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. (19)

La metodología XP tiene como objetivos establecer las mejores prácticas de Ingeniería de Software en el desarrollo de proyectos, mejorar la productividad de estos y garantizar la calidad del *software* desarrollado, haciendo que este supere las expectativas del cliente. Esta metodología se aplica en un contexto bien definido por, un cliente, requisitos cambiantes, grupo pequeño y muy integrado, y un equipo con formación elevada y capacidad de aprender. XP propone valores como la simplicidad, la comunicación, la retroalimentación y el coraje. (19)

Teniendo en consideración estas características, se selecciona XP como la metodología a emplear, porque combina las que han demostrado ser las mejores prácticas para desarrollar software, y las lleva al extremo. Esta metodología se adapta perfectamente a los que desarrollaran la herramienta, ya que es un equipo de dos integrantes bien preparados, donde la comunicación con el cliente constituye la clave para el éxito. Además de reducir el costo del cambio al que puede estar sometido el proyecto en todas las etapas del ciclo de vida del sistema.

Conclusiones del capítulo

En este capítulo se realizó un estudio acerca de las tecnologías sobre las cuales se desarrollará la aplicación. Apoyándose en dicho estudio se logró organizar y guiar el trabajo hacia el objetivo trazado en el inicio de la investigación. Se profundizó el estudio en los conceptos fundamentales para el desarrollo de una aplicación, que permita la realización de auditorías a SGBD a través de dispositivos móviles con SO Android, donde se obtuvo como resultado que no existe actualmente en el mercado ninguna herramienta que realice este tipo de tareas. También se pudieron definir, después de un análisis exhaustivo, las herramientas y metodología que serán utilizadas durante todo el proceso de construcción, definiendo que se utilizará la metodología XP para guiar el trabajo.

Capítulo 2: Características de la Aplicación

Introducción

En el presente capítulo se realiza el análisis y diseño de la propuesta del sistema. La investigación se centra en las características de la aplicación, dirigida por la metodología de desarrollo empleada en sus fases de Exploración y Planeación. Además se especifican los requisitos funcionales y no funcionales que debe cumplir la aplicación, para satisfacer las necesidades del cliente.

2.1 Propuesta de Solución

Para dar solución a las deficiencias identificadas y cumplir con las demandas del cliente de retirar la funcionalidad de filtrado de reporte Baseline, se propone desarrollar una aplicación, para dispositivos móviles con SO Android, que corrija estas insuficiencias.

La aplicación HCVP parte de un fichero XML, con la información referente al SGBD y al SO a auditar. Este fichero será cargado y leído por la aplicación mediante el método XMLPull. Luego se muestran y se seleccionan las sentencias que se desean ejecutar, se solicitan las credenciales para la conexión al SGBD y al SO, la conexión será por el protocolo SSH para las plataformas Linux y Microsoft Windows; una vez insertadas las credenciales se realizará una conexión de prueba para validar los datos introducidos. Finalmente se realiza el proceso de monitoreo de seguridad de la información y se generan los resultados obtenidos, que serán almacenados en la tarjeta de memoria externa del dispositivo para su posterior análisis. Para establecer la conexión al SGBD y al SO, la aplicación HCVP emplea una red Wi-Fi. La aplicación funcionará de acuerdo a la figura 1.

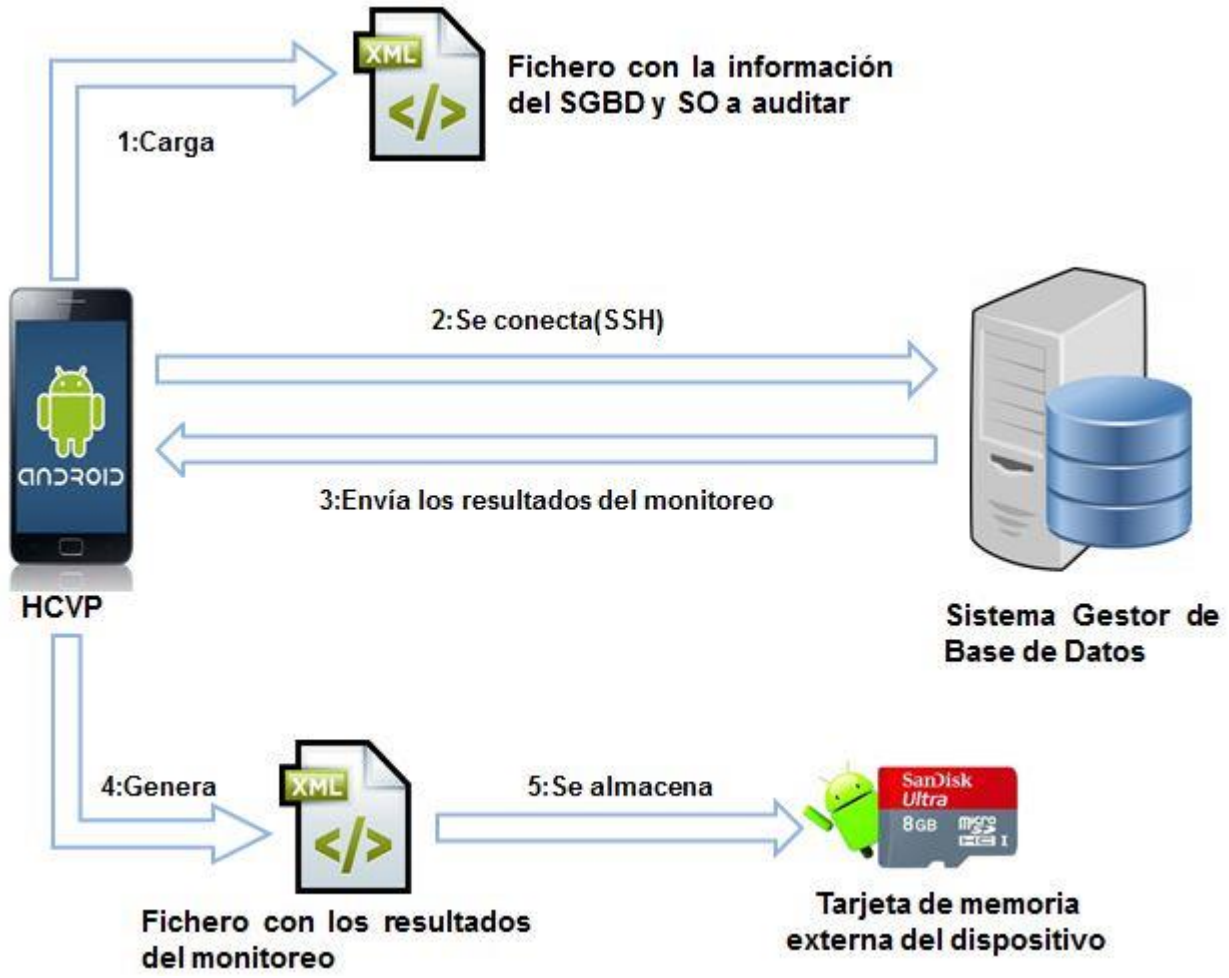


Figura 2. Propuesta de Solución

2.2 Requerimientos del sistema

Los requerimientos de un software son las propiedades o restricciones, determinadas con precisión, que un producto software debe poseer. Los mismos se clasifican en funcionales y no funcionales. Los requisitos funcionales son las condiciones que debe cumplir el sistema, mientras que los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Estas propiedades son las que hacen al producto atractivo, usable, rápido o confiable. (20)

2.2.1 Requisitos no funcionales

Son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema (21). Para la solución propuesta se definen los siguientes requisitos no funcionales:

Restricciones de Diseño e Implementación

- ✓ Plataforma Windows o Linux.
- ✓ El sistema deberá ser implementado en el lenguaje de programación Java 1.8.
- ✓ Se utilizará el marco de trabajo de desarrollo Android en su versión 5.0.1.
- ✓ Se empleará la herramienta de desarrollo Android Studio 1.2.

Software

Requisitos mínimos para el dispositivo móvil:

- ✓ Soporte para conexiones WIFI.
- ✓ Sistema operativo Android 2.2.

Hardware

Requisitos mínimos del dispositivo móvil:

- ✓ Capacidad disponible de 11.34 Mb en la memoria del dispositivo para la instalación y ejecución de la herramienta.
- ✓ Memoria RAM libre del dispositivo 32 Mb.
- ✓ Soporte para conexiones Wi-Fi

Seguridad

- ✓ Instalación previa de un sistema de *firewall*.

Interfaz de Usuario

- ✓ El sistema debe presentar un diseño responsivo que adapte las dimensiones del contenido y muestre los elementos de una forma ordenada y optimizada independientemente de la resolución del dispositivo.

2.3 Personas relacionadas con el Sistema

La auditoría solo puede ser ejecutada por las personas autorizadas y de la forma autorizada, lo que significa que solo los especialistas de seguridad informática podrán acceder al sistema. La interacción directa con el sistema la realizarán dichos especialistas.

Tabla 1. Personas relacionadas con el Sistema

Persona Encargada	Descripción
Especialista	Especialista de seguridad informática encargado de realizar auditorías informáticas

2.4 Exploración

El ciclo de vida de un proyecto realizado con la metodología XP se inicia con la Exploración. El cliente plantea a grandes rasgos las historias de usuarios. Al finalizar, el equipo cuenta con suficiente material de trabajo para producir una primera entrega. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas y tecnologías que serán utilizadas en el proyecto. (20)

2.5 Historias de usuario (HU)

Las historias de usuario corresponden a la técnica utilizada para especificar los requisitos del software. Se trata de formatos en los cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. (20)

Las Historias de Usuario se clasifican según:

Prioridad del negocio:

Alta: se le otorga a las HU que resultan funcionalidades fundamentales en el desarrollo del sistema, a las que el cliente define como principales para el control integral del sistema.

Media: se le otorga a las HU que resultan para el cliente como funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.

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

El riesgo de desarrollo:

Alto: cuando en la implementación de las HU se consideran la posible existencia de errores que conlleven a la inoperatividad del código.

Medio: cuando pueden aparecer errores en la implementación de la HU que puedan retrasar la entrega de la versión.

Bajo: cuando pueden aparecer errores que serán tratados con relativa facilidad sin que traigan perjuicios para el desarrollo del proyecto.

Las Historias de Usuario son representadas en tablas divididas por secciones donde:

- ✓ Número: número de la historia de usuario incremental en el tiempo.
- ✓ Nombre de Historia de Usuario: facilita la identificación para los programadores.
- ✓ Prioridad del negocio: Alta, Media, Baja.
- ✓ Riesgo de desarrollo: Alto, Medio, Bajo.
- ✓ Iteración asignada: número de la iteración.
- ✓ Puntos estimados: tiempo estimado en semanas que demorará el desarrollo de la HU.
- ✓ Descripción: breve descripción de la HU.
- ✓ Observaciones: señalamientos o advertencias de la aplicación.
- ✓ Prototipo Interfaz: prototipo Interfaz si aplica a la HU.

A continuación se expone un ejemplo de HU con prioridad alta, el resto se describen en el Anexo 1 de este documento:

Tabla 2. HU #3 Mostrar datos generales del fichero revisión

Historia de Usuario	
Número: 3	**Usuario:** Administrador
Nombre de Historia de Usuario: Mostrar datos generales del fichero de revisión	
Prioridad en negocio: Alta	**Riesgo de Desarrollo:** Media
Puntos estimados: 0.5	**Iteración asignada:** 1
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez	
Descripción: se muestran los datos generales del fichero, dígase, sistema operativo, versión del sistema operativo, tipo de SGBD y versión del SGBD.	
Observaciones: se obtiene un listado de datos generales.	
Prototipo de interfaz: 	

2.6 Planeación

El cliente establece la prioridad que tendrá cada HU según sus necesidades más inmediatas, luego los programadores realizan una estimación del esfuerzo que se necesita para cada una de ellas. Además se toman acuerdos sobre el contenido de las entregas. (20)

2.6.1 Estimación de Esfuerzo por Historia de Usuario

Las estimaciones de esfuerzo asociado a la implementación de las historias de usuario las establecen los programadores, para ello utilizan puntos. Un punto, equivale a una semana ideal de programación. A continuación se muestra la estimación de esfuerzo para cada HU:

Tabla 3. Estimación de esfuerzo por HU

No	Historia de Usuario	Puntos de Estimación
1	Obtener ruta del fichero de revisión	0.5
2	Leer el fichero de revisión	0.5
3	Mostrar datos generales del fichero de revisión	0.5
4	Mostrar las sentencias del fichero de revisión	0.5
5	Seleccionar las sentencias a ejecutar	0.5
6	Insertar credenciales para la conexión al SGBD	0.2
7	Insertar credenciales para la conexión al Sistema Operativo	0.3
8	Conectar al Sistema Operativo	1
9	Conectar al Sistema Gestor de Bases de Datos	1
10	Ejecutar las sentencias de auditoría sobre el SGBD	0.5
11	Generar el fichero de resultados del proceso de auditoría	1
12	Guardar el fichero de resultados	1

2.7 Plan de Iteraciones

Luego de ser identificadas y descritas las HU, el próximo paso será la planeación de la fase de implementación, donde se realizaran tres iteraciones, las cuales se describen a continuación:

Iteración 1

Serán desarrolladas las HU 1, 2, 3, 4, 5 que son las de prioridad Alta. El cliente podrá probar las funcionalidades ya implementadas.

Iteración 2

Serán desarrolladas las HU 6, 7, 8, 9, 10 que son las de prioridad Media, una vez implementadas estas funcionalidades, facilitarán un funcionamiento más completo de la aplicación.

Iteración 3

Serán desarrolladas las HU 11, 12 que son las de prioridad Baja, implementadas estas funcionalidades, el cliente podrá ver una versión final de la aplicación.

Tabla 4. Plan de Iteraciones

Iteración	Orden de las HU a implementar	Duración Total
1	Obtener ruta del fichero de revisión	2.5 semanas
	Leer el fichero de revisión	
	Mostrar datos generales del fichero de revisión	
	Mostrar las sentencias del fichero de revisión	
	Seleccionar las sentencias a ejecutar	
2	Insertar credenciales para la conexión al SGBD	3 semanas
	Insertar credenciales para la conexión al Sistema Operativo	
	Conectar al Sistema Operativo	
	Conectar al Sistema Gestor de Bases de Datos	
	Ejecutar las sentencias de auditoría sobre el SGBD	
3	Generar el fichero de resultados del proceso de auditoría	2 semanas
	Guardar el fichero de resultados	

2.8 Plan de entregas

El plan de entrega da un aproximado de las versiones, tomando como fecha de inicio de la implementación el 15 de febrero del 2015:

Tabla 5. Plan de Entregas

Iteración	Fecha de Entrega
1	1 de marzo del 2015
2	25 de marzo del 2015
3	15 de abril del 2015

Conclusiones del capítulo

Con la realización de este capítulo se identificaron las funcionalidades y los requisitos que la aplicación debe cumplir. Se realiza una estimación de esfuerzo dedicado al desarrollo para cada funcionalidad, identificando el orden en que serán desarrolladas, facilitando así una mejor organización del trabajo. Además se crea un plan de entregas que permite establecer fechas de culminación para cada iteración.

Capítulo 3: Análisis y Diseño de la Aplicación

Introducción

Después de haber desarrollado las fases de Exploración y Planificación en el capítulo anterior, se da paso a la fase de Diseño de la Aplicación, propia de la metodología XP. En este capítulo se define el patrón de arquitectura a seguir, además de los patrones de diseños que serán utilizados en la implementación de la aplicación. Además serán definidas las tarjetas CRC como técnicas de diseño.

3.1 Arquitectura

Arquitectura de software se refiere a las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos. (22)

3.1.1 Estilo Arquitectónico: Arquitectura en Capas

Tras haber analizado los diferentes estilos arquitectónicos se decide utilizar el estilo: Arquitectura en Capas, teniendo en cuenta las características y especificaciones de la aplicación a desarrollar, debido a que se enfoca a la distribución de responsabilidades y roles.

Los principales beneficios del estilo de arquitectura basado en capas son (23):

- Aislamiento: el estilo de arquitectura de capas permite aislar los cambios en tecnologías a ciertas capas para reducir el impacto en el sistema total.
- Rendimiento: distribuir las capas entre múltiples sistemas (físicos) puede incrementar la escalabilidad, la tolerancia a fallos y el rendimiento.
- Mejoras en Pruebas: beneficio de tener interfaces bien definidas para cada capa, así como la habilidad para cambiar a diferentes implementaciones de las interfaces de cada capa.

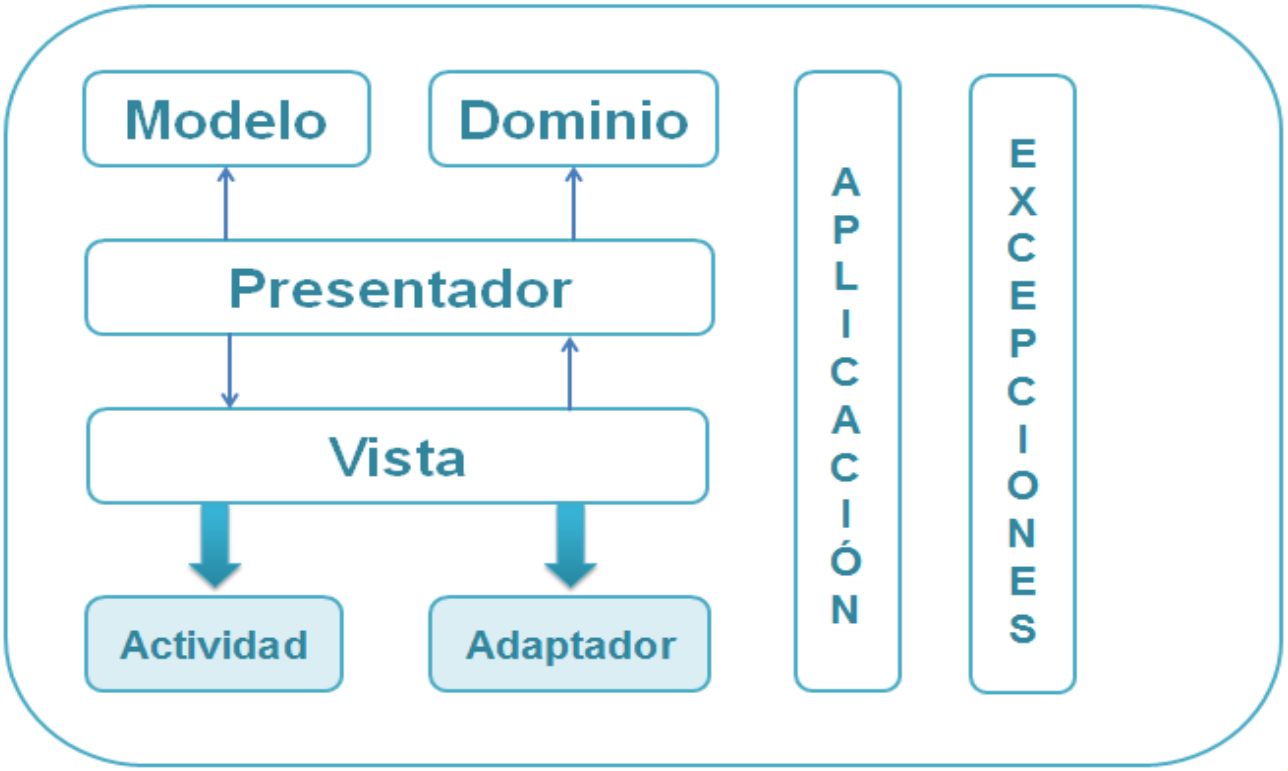


Figura 3. Arquitectura en capas

A continuación se realiza una breve descripción de cada una de las capas por las que está compuesta la aplicación de acuerdo a la figura anterior.

La capa **Modelo**: será la encargada de proveer la información, esta no conocerá nada del dominio, tampoco de la presentación.

La capa **Presentador**: será la encargada, como su nombre indica, de gestionar y proveer datos a la vista.

La capa **Vista**: contiene una referencia al presentador. Esta capa será la encargada de modificar las interfaces gráficas y de capturar los eventos sobre ella.

Dominio: es totalmente independiente de la capa de presentación, en ella residirá la lógica de negocio de la aplicación.

Excepciones: permite el tratamiento de excepciones a través de las diferentes capas de la aplicación.

Aplicación: permite gestionar datos globales de la aplicación.

3.2 Patrones de Diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Se deben tener presente los siguientes elementos de un patrón: su nombre, el problema, la solución y las consecuencias. (24)

Un patrón de diseño identifica clases, instancias, roles, colaboraciones y la distribución de responsabilidades, además de que ayuda a construir clases y a estructurar sistemas de clases. Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de *software* y otros ámbitos referentes al diseño de interacción o interfaces. (24)

Los patrones de diseño no se basan en la creación de estructuras de datos concretas que se pueden implementar en una clase, si no que se centran en una solución a un problema concreto en el ámbito de la programación orientada a objetos, proponiendo una solución genérica de clases y relaciones para resolver dicho problema.

3.2.1 Patrones para asignar responsabilidades (GRASP)

Los patrones GRASP[5] describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. (25)

- Experto: este patrón es el encargado de asignar una responsabilidad al experto en información, es decir, la clase que posee la información necesaria y suficiente para ejecutar la responsabilidad asignada. En la aplicación, este patrón se evidencia en la clase EjecutarAuditoriaPresenter.java.

- Creador: la creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. Este patrón es el encargado de asignar a las clases la responsabilidad de instanciar otra clase, si esta asignación se realiza correctamente se puede garantizar que el diseño pueda soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización. Este patrón se refleja

[5] General Responsibility Assignment Software Patterns o Responsabilidad de la asignación general Patrones de Software.

en las clases CargarXMLPresenter.java, FileExplorerPresenter.java, MostrarXMLPresenter.java, ConectarBDPresenter.java y ConectarSOPresenter.java.

A continuación de se muestra un fragmento de código donde este patrón es utilizado:

```
public class FileExplorerPresenter {

    private FileExplorerActivity _fileExplorer;
    private static final String _raiz = "/";
    private List<String> _item;
    private List<String> _ruta;

    public FileExplorerPresenter(FileExplorerActivity fileExplorer) {...}

    public void getDir(String dirPath) {...}

    public void onListItemClicked(int position) {...}
}
```

Figura 4. Fragmento de código de la utilización de patrón GRASP Creador

- Bajo Acoplamiento: este patrón consiste en mantener las clases lo menos relacionadas posible. De forma tal que, al producirse un cambio en alguna clase, se tenga el mínimo de repercusión en las otras. En la asignación de responsabilidad se debe tener en cuenta este aspecto, pero sin afectar la funcionalidad. En la aplicación, la utilización de este patrón se evidencia en la clase XMLParser.java porque tiene la menor dependencia posible de otras clases. Además se encuentra en las clases ConexionOracle.java, ConexionPostgres.java, ConexionMySQL.java y ConexionSQLServer.java, porque no tienen dependencia de ninguna otra clase.
- Alta cohesión: cada elemento del diseño debe realizar una labor única dentro del sistema, lo cual expresa que la información que almacena una clase debe ser coherente y estar en la mayor medida de lo posible relacionada con ella. Este patrón es el encargado de asignar responsabilidades de manera que la información que se almacena en una clase, sea la necesaria y esté bien delimitada. Este patrón se refleja en las clases Indicador.java, Parametro.java, XMLDatos.java y ConexionDatos.java.

3.2.2 Patrones GOF

Los patrones de diseño GOF[6] se clasifican en tres categorías basadas en su propósito: creacionales, estructurales y de comportamiento, desglosados en 23 patrones. (24)

- Singlenton (Creacional)

El objetivo de este patrón es asegurarse de que, de una clase solo existe una instancia y que esta es accesible. La utilización de este patrón aporta beneficios como (24):

- ✓ Acceso controlado a la única instancia.
- ✓ Reduce el espacio de nombres, ya que evita contaminarlo con variables globales.
- ✓ Permite refinar operaciones y la representación a través de la creación de subclases.
- ✓ Permite controlar fácilmente y sin apenas cambios, el número de instancias que se crea.

A continuación se muestra un fragmento de código donde este patrón es utilizado:

```
public class PortableApplication extends Application {

    private static PortableApplication application;
    private XMLManager xmlManager;
    private ConexionManager conexionManager;

    @Override
    public void onCreate() {
        super.onCreate();
        application = this;
        application.initInstance();
    }

    protected void initInstance() {
        xmlManager = new XMLManager();
        conexionManager = new ConexionManager();
    }

    public static PortableApplication getApplication() { return application; }

    public XMLManager getXmlManager() { return xmlManager; }

    public ConexionManager getConexionManager() { return conexionManager; }


}
```

Figura 5. Fragmento de código de la utilización del patrón GoF Singlenton

[6] Gang of Four (Grupo de los Cuatro): Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides.

3.2.3 Modelo-Vista-Presentador (MVP)

MVP es un patrón de diseño que tiene como objetivo, separar la interfaz de usuario de la lógica de las aplicaciones.

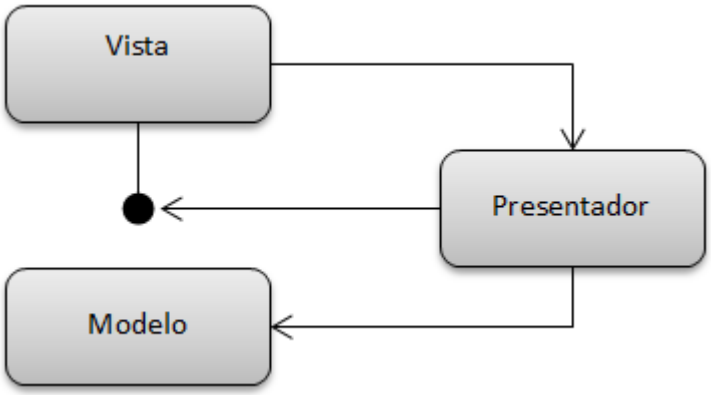


Figura 6. Patrón Modelo – Vista – Presentador (26)

En concepto de este patrón consta de tres componentes. El primero de ellos, la Vista, es el que se encarga de mostrar la información al usuario e interactuar con él, para realizar ciertas operaciones. El segundo es el Modelo que, ajeno a cómo es mostrada la información al usuario, realiza toda la lógica de las aplicaciones, usando las entidades del dominio. Y por último se tiene al Presentador, que es el que relaciona a ambos componentes sin que haya ningún tipo de dependencia entre ellos.

A continuación se muestra como queda estructurado el proyecto haciendo uso del patrón MVP:

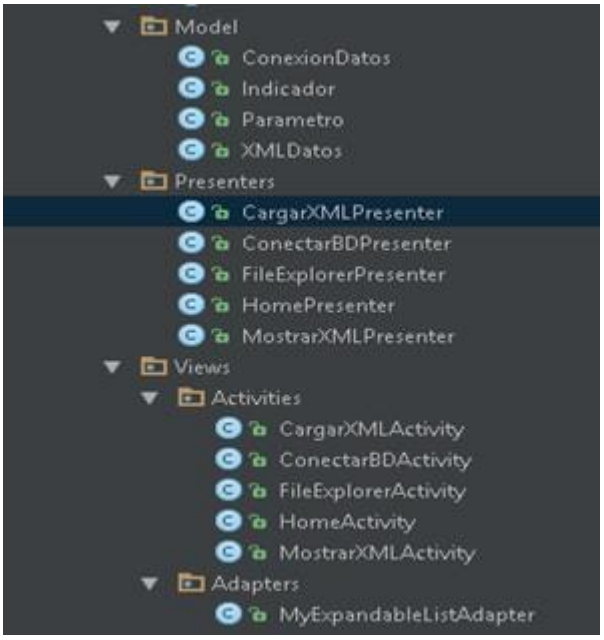


Figura 7. Estructura del proyecto utilizando el patrón MVP

3.3 Tarjetas Clases – Responsabilidad – Colaborador (CRC)

La técnica CRC propone una forma de trabajo en equipo, para encontrar los objetos del dominio de la aplicación, sus responsabilidades y su colaboración con otros para realizar tareas. Las tarjetas CRC registran el nombre de las clases, sus responsabilidades y las clases con la que colaboran. Las principales características de las tarjetas CRC son: (27)

- ✓ Identificación de clases y asociaciones que participan del diseño del sistema.
- ✓ Obtención de las responsabilidades que debe cumplir cada clase.
- ✓ Establecimiento de cómo una clase colabora con otras clases para cumplir con sus responsabilidades.

Para una primera versión de la aplicación se identificaron 41 tarjetas CRC. A continuación se muestran algunas de estas tarjetas CRC y el resto se especifican en el Anexo 3 del presente documento.

Tabla 6. Tarjeta CRC CargarXMLPresenter

CargarXMLPresenter	
Responsabilidad:	Colaborador:
Capturar los eventos de la vista CargarXMLActivity, ejecutar las funcionalidades de la clase XMLParser y devolver los resultados a la vista.	CargarXMLActivity XMLParser

Tabla 7. Tarjeta CRC XMLParser

XMLParser	
Responsabilidad:	Colaborador:
Leer el fichero XML y guardar los datos en XMLDatos.	XMLDatos

Conclusiones

En este capítulo se realizó el análisis y diseño de la aplicación, en el mismo fueron definidos los patrones arquitectónicos y de diseño, usados con el fin de lograr una mejor organización de los elementos que dan forma a la aplicación y a su vez fueron confeccionadas las Tarjetas CRC que son fundamentales para la implementación de la aplicación.

Capítulo 4 Implementación y Prueba

Introducción

En el presente capítulo, según dicta la metodología de desarrollo XP, se documentan y desarrollan las fases de Implementación y Prueba. Las Historias de Usuarios se desglosan en Tareas de Ingeniería con el objetivo de facilitar el trabajo de los desarrolladores. Se muestran los resultados de las pruebas de aceptación realizadas a la aplicación.

4.1 Implementación

Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración. Como las historias de usuario no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas de ingeniería, para facilitar la comprensión del análisis, recabando con el cliente, todos los datos que sean necesarios. El cliente, por lo tanto, también debe participar activamente durante esta fase del ciclo. (28)

4.1.1 Tareas de Ingeniería

Ajustándose a la planificación realizada, se llevaron a cabo tres iteraciones de desarrollo de la aplicación, obteniéndose al final un producto funcional y listo para ser utilizado. Para llevar a cabo una correcta implementación se deben definir las tareas de ingeniería (TI) que serán desarrolladas en cada una de las iteraciones. Las TI permitirán a los desarrolladores obtener con mejor detalle, la funcionalidad que se desea ejecutar con la HU. A continuación se muestran algunas de las TI definidas, el resto se describen en el Anexo 3 de este documento:

Tabla 8. Tarjeta CRC XMLParser

Tarea de Ingeniería	
Número de la tarea : 5	**Historia de Usuario:** 5
Nombre de la Tarea: Seleccionar las sentencias a ejecutar.	
Tipo: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: 28/2/15.	**Fecha fin:** 6/3/15.
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez.	

Descripción: de las sentencias SQL mostradas, el especialista podrá seleccionar las que quiere ejecutar en ese momento, una vez que son seleccionadas, se crea el listado final de las sentencias SQL a ejecutar en la revisión.

Tabla 9. Tarea de Ingeniería: Insertar credenciales para la conexión al SGBD

Tarea de Ingeniería	
Número de la tarea : 6	**Historia de Usuario:** 6
Nombre de la Tarea: Insertar credenciales para la conexión al SGBD.	
Tipo: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: 7/3/15.	**Fecha fin:** 13/3/15.
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez.	
Descripción: se registra en el sistema los datos necesarios para la conexión al SGBD, el especialista se debe haber autenticado (usuario, contraseña para el SGBD) e ingresado las credenciales de conexión (dirección y puerto) para acceder al SGBD.	

Tabla 10. Tarea de Ingeniería: Guardar el fichero de resultados

Tarea de Ingeniería	
Número de la tarea : 12	**Historia de Usuario:** 12
Nombre de la Tarea: Guardar el fichero de resultados.	
Tipo: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: 7/5/15.	**Fecha fin:** 13/5/15.
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez.	
Descripción: la aplicación debe brindar la posibilidad de exportar un fichero con el resultado obtenido en el proceso de auditoría, este fichero será almacenado en la memoria interna del dispositivo móvil para su posterior análisis.	

4.2 Estándares de nomenclatura y codificación utilizados

Nomenclatura General

- ✓ El idioma por defecto a la hora de dar sentido funcional al nombre de clases, variables, constantes, etc., será una mezcla entre la nomenclatura tradicional en inglés y la nomenclatura funcional adoptada.
- ✓ El nombre de todas las variables comenzarán con letra minúscula y si este está compuesto por varias palabras se utilizará el estilo de escritura lowerCamelCase, que dicta que un nombre compuesto por varias palabras comenzará con minúscula pero todas las palabras internas que lo componen comienzan con mayúscula.

Paquetes:

- ✓ Por defecto todos los nombres de paquetes se escribirán en minúsculas y sin utilizar caracteres especiales. El paquete base queda definido como com.tesis.android, en este paquete no se definirá ninguna clase.

Identación:

- ✓ En el contenido siempre se identará con tabulaciones, nunca utilizando espacios en blanco.

Clases:

- ✓ Los nombres de clases deben ser mezclas de mayúsculas y minúsculas, con la primera letra de cada palabra interna en mayúsculas (CamelCase).
- ✓ Se debe intentar mantener los nombres de clases, simples y descriptivos.
- ✓ Se deben usar palabras completas y evitar acrónimos y abreviaturas.

Métodos:

- ✓ Los métodos deberán ser verbos (en infinitivo), en mayúsculas y minúsculas con la primera letra del nombre en mayúscula, y con la primera letra de cada palabra interna en mayúsculas también (UpperCamelCase).
- ✓ No se permiten caracteres especiales.
- ✓ El nombre ha de ser lo suficientemente descriptivo, no importando a priori la longitud del mismo.

Variables:

- ✓ El nombre de las variables debe empezar con letra minúsculas y de existir un salto de palabra comenzaría con mayúscula, en el caso de los atributos de las clases, estos comenzarán con guión bajo.

Constantes:

- ✓ Los nombres de constantes deben escribirse todo en mayúsculas con las palabras separadas por guión bajo. Todas serán declaradas como public static final.

Estilos de codificación

Comentarios:

Reglas generales a la hora de escribir comentarios de documentación.

- ✓ Siempre se escribe en tercera persona.
- ✓ Las descripciones siempre deberían empezar por un verbo.

4.3 Validación de la Propuesta

Uno de los pilares de la metodología XP es el proceso de pruebas. XP anima a realizar pruebas constantemente, tanto como sea posible. Esto permite aumentar la calidad de los sistemas, reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones. (29)

XP divide las pruebas del sistema en dos grupos: pruebas unitarias (encargadas de verificar el código y diseñada por los programadores) y pruebas de aceptación o pruebas funcionales (destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final). (29)

4.3.1 Pruebas unitarias

Uno de los métodos utilizados para realizar pruebas de *software* en la metodología XP son las pruebas unitarias. La base de este método es hacer pruebas en pequeños fragmentos del código de la aplicación, estos fragmentos deben ser unidades estructurales del programa, encargados de una tarea específica. En programación orientada a objetos se puede afirmar que estas unidades son los métodos o las funciones

que se tienen definidos. El objetivo de estas pruebas es el aislamiento de partes del código y la demostración de que no contienen errores. Estas no generan artefactos y no son directamente palpables para el cliente. (29)

Resultados de las pruebas unitarias

Las pruebas unitarias fueron ejecutadas sistemáticamente, cada vez que se terminaba de implementar cada una de las iteraciones. Dichas pruebas fueron desarrolladas utilizando la herramienta Android JUnit. Esta herramienta permite probar componentes específicos mediante clases de casos de prueba. Estas clases proporcionan métodos auxiliares para la creación de objetos de imitación y métodos que ayudan a controlar el ciclo de vida de una aplicación. A continuación se muestran los resultados de las pruebas realizadas a las funcionalidades en cada una de las iteraciones:

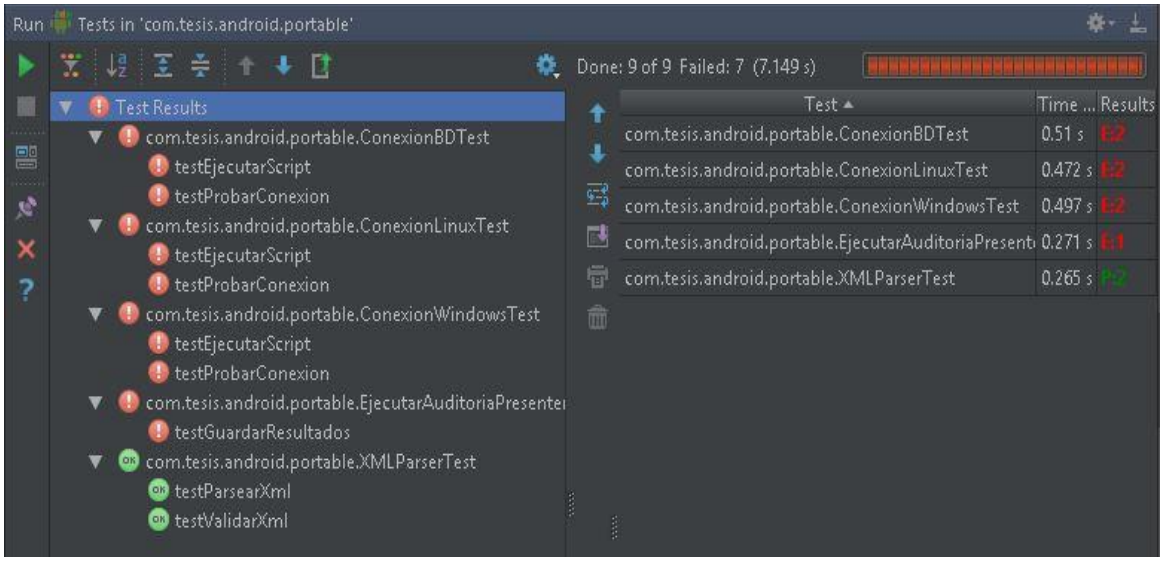


Figura 8. Resultados de las pruebas unitarias en la iteración 1

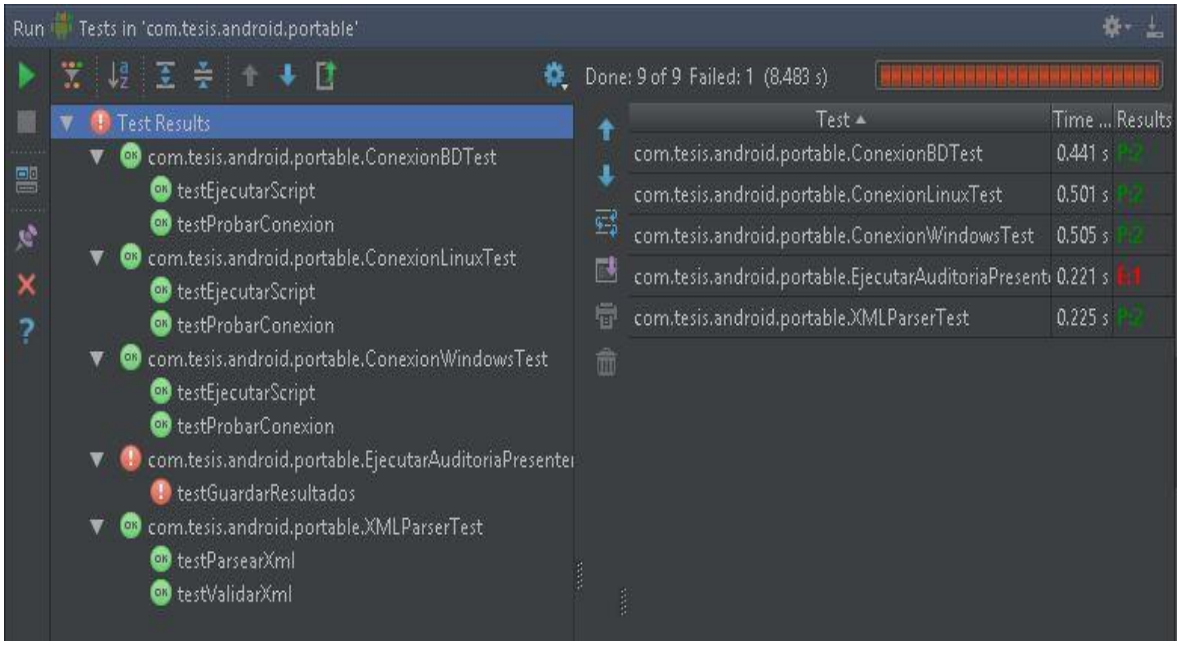


Figura 9. Resultados de las pruebas unitarias en la iteración 2

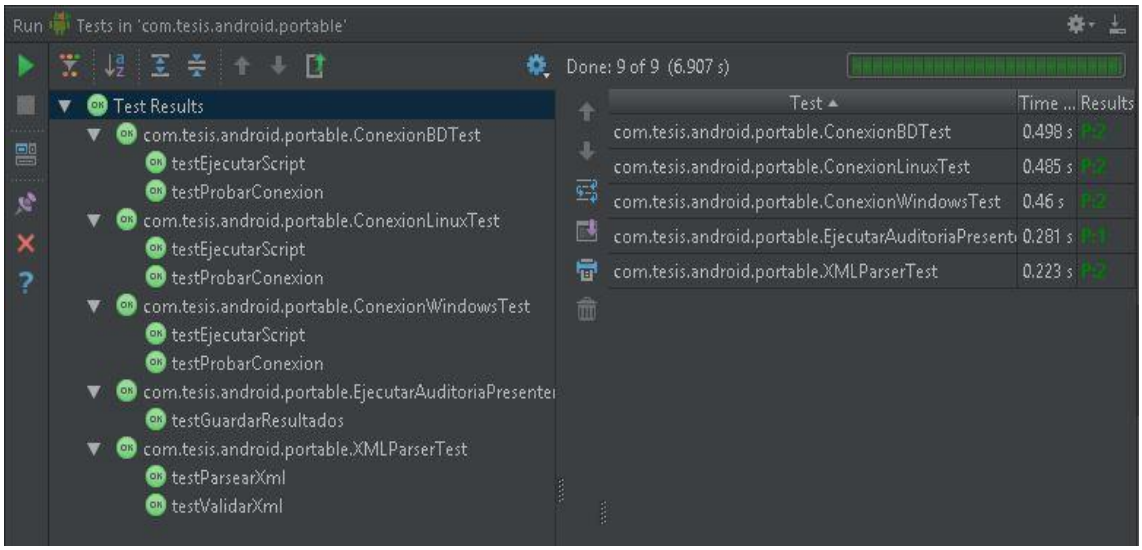


Figura 10. Resultados de las pruebas unitarias en la iteración 3

4.3.2 Pruebas de aceptación

El objetivo de estas pruebas es verificar los requisitos, por este motivo, los propios requisitos del sistema son la principal fuente de información a la hora de construir las pruebas de aceptación.

Las pruebas de aceptación son creadas a partir de las historias de usuario. Durante una iteración la historia de usuario seleccionada en la planificación de iteraciones, se convertirá en una prueba de aceptación. El cliente o usuario especifica los aspectos a probar, cuando una historia de usuario ha sido correctamente implementada. (29)

Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección de las pruebas de aceptación y tomar decisiones acerca de las mismas. (29)

A continuación se muestran tres de los casos de prueba de aceptación correspondientes a una historia de usuario de cada iteración, el resto se encuentran expuestas en el Anexo 4 del presente documento.

Tabla 11. PA-1: Mostrar datos generales del fichero de revisión

Caso de prueba de aceptación	
Código : PA-1	**Historia de Usuario:** 3
Nombre: Mostrar datos generales del fichero de revisión.	
Descripción: prueba la funcionalidad de mostrar los datos generales de fichero de revisión a utilizar.	
Condiciones de ejecución: El usuario debe seleccionar el fichero de revisión.	
Entrada/ Pasos de ejecución: el usuario selecciona el fichero de revisión, la aplicación selecciona de la cabecera del fichero los datos necesarios y los muestra el resultado al usuario.	
Resultado esperado: una interfaz con los siguientes datos: • Tipo y versión del Sistema Operativo • Tipo y versión del SGBD	

Evaluación: Satisfactoria.

Tabla 12. PA-3: Conectar al Sistema Gestor de Bases de Datos Postgres SQL

Caso de prueba de aceptación	
Código : PA-3	**Historia de Usuario:** 9
Nombre: Conectar al Sistema Gestor de Bases de Datos Postgres SQL.	
Descripción: prueba para la funcionalidad Conectar al Sistema Gestor de Bases de Datos Postgres SQL.	
Condiciones de ejecución: • El usuario deberá cargar el fichero de revisión correspondiente a Postgres SQL. • El usuario deberá introducir las credenciales para la conexión para el SGBD. • El usuario deberá seleccionar la opción: Probar Conexión.	
Entrada/ Pasos de ejecución: el usuario introduce las credenciales de conexión y escoge la opción “Probar Conexión”. La aplicación construye la url y conforma una solicitud al servidor del SGBD. Recoge la respuesta y muestra al usuario el resultado.	
Resultado esperado: mensaje de respuesta.	
Evaluación: satisfactoria.	

Tabla 13. PA-6: Guardar el fichero de resultados

Caso de prueba de aceptación	
Código : PA-6	**Historia de Usuario:** 12
Nombre: Guardar el fichero de resultados.	
Descripción: prueba para la funcionalidad Guardar el fichero de resultados.	
Condiciones de ejecución:	

• El usuario deberá haber ejecutado las sentencias de revisión. • El usuario deberá seleccionar la opción: Guardar resultados. • El usuario especifica la ruta donde desea guardar los resultados. • El usuario deberá seleccionar la opción: Guardar.
Entrada/ Pasos de ejecución: el usuario selecciona la opción Guardar resultados. La aplicación muestra un gestor de ficheros para que el usuario seleccione la ruta donde desea guardar los resultados. El usuario selecciona guardar.
Resultado esperado: mensaje de respuesta.
Evaluación: satisfactoria.

Resultados de las pruebas aceptación

En la etapa de pruebas se realizaron seis pruebas de aceptación las cuales arrojaron resultados satisfactorios con un número de cinco no conformidades, las cuales fueron resueltas con éxito (ver Figura 12).

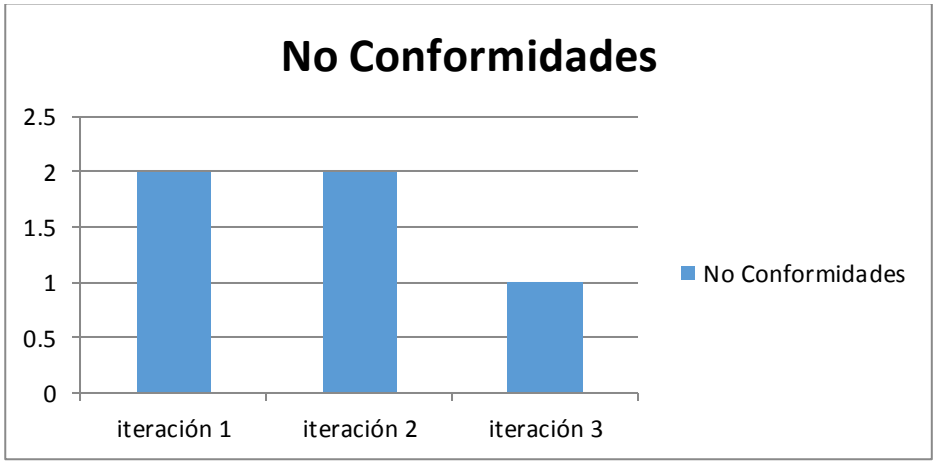


Figura 11. Resultados de las pruebas de aceptación en cada iteración

4.3.3 Pruebas de rendimiento

De acuerdo a las características de la aplicación se decide realizar pruebas de rendimiento para comprobar el desempeño de la solución bajo distintos entornos de trabajo. Para llevar a cabo estas pruebas se utilizaron los siguientes dispositivos móviles:

Samsung Galaxy S4 i337 version de Android: 4.4.2 CPU: 4 cores @1.9 GHz RAM: 2GB.

Alcatel One Touch 4030A version de Android: 4.1.1 CPU: 1 core @1.0GHz RAM: 512MB.

En la comparación se tuvo en cuenta el consumo de los siguientes recursos: tiempo de ejecución, batería, microprocesador, memoria RAM.

Resultados de las pruebas de rendimiento

En esta etapa de las pruebas se pudo analizar el comportamiento de los dispositivos móviles en la ejecución de la aplicación. A continuación de muestran los resultados obtenidos:

Tabla 14. Resultados obtenidos en las pruebas de rendimiento.

	Samsung Galaxy S4 i337	**Alcatel One Touch 4030A**
Tiempo de ejecución (segundos)	2,3	3,8
Batería (%)	1	3
Microprocesador (%)	4,59	12,66
Memoria RAM (MB)	27,46	28,23

Conclusiones

En este capítulo fueron generados y desarrollados los artefactos propuestos por la metodología XP en sus fases de Implementación y Prueba. Se implementaron las tareas de ingeniería correspondientes a las HU, logrando así una mayor organización en el desarrollo de la aplicación. Además el producto final se validó satisfactoriamente mediante las pruebas unitarias y las pruebas de aceptación.

Conclusiones Generales

Con el propósito de dar cumplimiento al objetivo general y a la problemática planteada en el presente trabajo, fueron resueltas cada una de las tareas trazadas al inicio de la investigación de forma satisfactoria. Se profundizó en el conocimiento de los sistemas auditores a SGBD, así como lo mecanismos necesarios para su implementación.

Para guiar el proceso de desarrollo de la aplicación se aplicó la metodología XP. Mediante esta metodología de desarrollo fueron generados los artefactos necesarios para la implementación y validación de la solución. Por tanto, se puede concluir:

- ✓ Se eligieron las tecnologías adecuadas para desarrollar la aplicación, cumpliendo con los requisitos planteados por el cliente.
- ✓ Al evaluarse los procesos de negocio asociados al monitoreo de la seguridad de la información se obtuvo un modelo guía para la implementación del sistema mediante la generación de los artefactos correspondientes a los flujos de trabajo propuestos por la metodología XP.
- ✓ La implementación de las funcionalidades de la aplicación, acorde a las pautas de diseño, garantizó el cumplimiento de lo establecido en el objetivo general de la investigación.
- ✓ Se realizaron las pruebas necesarias, comprobando el correcto funcionamiento de la aplicación.

Recomendaciones

Luego de la experiencia adquirida durante la realización de la investigación, se recomienda:

- ✓ Incorporar a la aplicación la opción de subir los resultados a un repositorio externo accesible por el módulo MGASGBD, garantizando así, agilizar el análisis de estos resultados, obtenido el diagnóstico de la seguridad del SGBDO y del SO de una manera más ágil.
- ✓ Cifrar los resultados del proceso de monitoreo, asegurando la seguridad de estos, ya que es información confidencial del SGBD y del SO objetos de la auditoría.

Referencias Bibliográficas:

1. Garcia, LIC. Rosa Maria Mato. *Diseño de Bases de Datos.*

2. Hoy. *Que es auditoría informática.* [En línea] 8 de noviembre de 2014. http://hoy.com.do/tecnologiaque-es-auditoria-informatica/.

3. Rodriguez, German Ramirez. *SOBRE LA AUDITORÍA INFORM ÁTICA Y LOPD DESDE LA EXPERIENCIA PERSONAL Y PROFESIONAL.* Madrid : Personal, 2009.

4. Ruiz, Alberto Jiménez. *myEchelon: Un sistema de Auditoría de Seguridad Informática Avanzado bajo GNU/Linux.*

5. Soft Tree Tecnologies. *DB Audit Home Page .* [En línea] [Citado el: 10 de noviembre de 2014.] www.softtreetech.com/dbaudit/.

6. Microsoft Developer Network. *SQL Server Audit.* [En línea] [Citado el: 10 de noviembre de 2014.] msdn.microsoft.com/es-es/library/cc280386.aspx.

7. CIO America Latina. [En línea] [Citado el: 10 de noviembre de 2014.] http://www.cioal.com/2010/01/18/oracle-audit-vault/.

8. Data Security Solutions. *Monitoreo de Actividad en Base de Datos y Prevención de Intrusiones.* [En línea] [Citado el: 10 de noviembre de 2014.] www.datasecuritys.com/dbProtect.html.

9. Gralla, Preston. *Cómo funcionan las redes inalámbricas.* s.l. : Anaya Multimedia, 2007.

10. Ecured. *Wi-Fi.* [En línea] 30 de noviembre de 2014. http://www.ecured.cu/index.php/Wifi.

11. EcuRed. *Sistema Operativo.* [En línea] 10 de diciembre de 2014. http://www.ecured.cu/index.php/Sistema_operativo.

12. ¿Qué es Android? [En línea] [Citado el: 3 de noviembre de 2014.] http://www.xatakandroid.com/sistema-operativo/que-es-android.

13. Configurar Equipos. *Que es Android: Características y Aplicaciones.* [En línea] 12 de diciembre de 2014. http://www.configurarequipos.com/doc1107.html.

14. Curso de Android. *Vision general y Entorno de Desarrollo.* [En línea] 14 de diciembre de 2014. http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/99-arquitectura-de-android.

15. Developer Android. *Andriod Studio Overview.* [En línea] 2015 de febrero de 2. http://developer.android.com/tools/studio/index.html.

16. Android Tools Project Site. *Gradle Plugin User Guide.* [En línea] [Citado el: 15 de febrero de 2015.] http://tools.android.com/tech-docs/new-build-system/user-guide.

17. Área de Programación y Desarrollo. Curso de Introducción a Java. [En línea] [Citado el: 8 de diciembre de 2014.] http://www.mundojava.net/caracteristicas-del-lenguaje.html?Pg=java_inicial_4_1.html.

18. Android SDK | Android Developers. [En línea] [Citado el: 24 de noviembre de 2014.] http://developer.android.com/intl/es/sdk/index.html.

19. XP – Extreme Programming Ingeniería de Software. [En línea] [Citado el: 15 de diciembre de 2014.] http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.

20. Gimson, Loraine. *Metodologías ágiles y desarrollo basado en conocimientos.* 2012.

21. Sommerville, Ian. *Ingeniería de Software. 7ma Edición.* Madrid : Pearson Educación, 2005.

22. L. Bass, P. Clements, R. Kazman. *Software Architecture in Practice 2nd Edition.* s.l. : Addison Wesley, 2003.

23. C. de la Torre, U.Zorrilla, M.A. Ramos,J Calvarro. *Guía de Arquitectura N-Capas orientada al Dominio con .Net 4.0.* s.l. : Krasis Consulting , 2010.

24. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Patrones de Diseño. Elementos de software orientados a objetos reutilizable.* Madrid : Perason Educación, 2003.

25. Practicas de Software. *Patrones GRASP.* [En línea] 5 de marzo de 2015. http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

26. Lime Creative Labs. [En línea] 20 de marzo de 2015. http://www.limecreativelabs.com/mvp-android/.

27. Extreme Programming. [En línea] [Citado el: 2015 de febreo de 3.] http://www.extremeprogramming.org/rules/crccards.html.

28. Joskowicz, Ing. José. *Reglas y Prácticas en eXtreme Progamming.* 2008.

29. Maite Rodríguez Corbea, Meylin Ordóñez Pérez. *La Metodología XP Aplicable al Desarrollo del Software.* Universidad de las Ciencias Informáticas, Cuba : s.n., 2007.

Bibliografía

Garcia, LIC. Rosa Maria Mato. *Diseño de Bases de Datos.*

Rodriguez, German Ramirez. *SOBRE LA AUDITORÍA INFORM ÁTICA Y LOPD DESDE LA EXPERIENCIA PERSONAL Y PROFESIONAL.* Madrid : Personal, 2009.

Ruiz, Alberto Jiménez. *myEchelon: Un sistema de Auditoría de Seguridad Informática Avanzado bajo GNU/Linux.*

Gralla, Preston. *Cómo funcionan las redes inalámbricas.* s.l. : Anaya Multimedia, 2007.

Gimson, Loraine. *Metodologías ágiles y desarrollo basado en conocimientos.* 2012.

Sommerville, Ian. *Ingeniería de Software. 7ma Edición.* Madrid : Pearson Educación, 2005.

L. Bass, P. Clements, R. Kazman. *Software Architecture in Practice 2nd Edition.* s.l. : Addison Wesley, 2003.

C. de la Torre, U.Zorrilla, M.A. Ramos,J Calvarro. *Guía de Arquitectura N-Capas orientada al Dominio con .Net 4.0.* s.l. : Krasis Consulting , 2010.

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. *Patrones de Diseño. Elementos de software orientados a objetos reutilizable.* Madrid : Perason Educación, 2003.

Joskowicz, Ing. José. *Reglas y Prácticas en eXtreme Progamming.* 2008.

Maite Rodríguez Corbea, Meylin Ordóñez Pérez. *La Metodología XP Aplicable al Desarrollo del Software.* Universidad de las Ciencias Informáticas, Cuba : s.n., 2007.

Anexos

Anexo 1: Historias de Usuarios

Tabla 15. HU1 - Obtener ruta del fichero de revisión.

Historia de Usuario	
Número: 1	**Usuario:** Administrador
Nombre de Historia de Usuario: Obtener ruta del fichero de revisión	
Prioridad en negocio: Alta	**Riesgo de Desarrollo:** Media
Puntos estimados: 0.5	**Iteración asignada:** 1
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez	
Descripción: Se obtiene la dirección donde se encuentra el fichero a ejecutar	
Observaciones: Se almacena la dirección de fichero a ejecutar	
Prototipo de interfaz:	

Tabla 16. HU 2 - Leer el fichero de revisión.

Historia de Usuario	
Número: 2	**Usuario:** Administrador
Nombre de Historia de Usuario: Leer el fichero de revisión	
Prioridad en negocio: Alta	**Riesgo de Desarrollo:** Media
Puntos estimados: 0.5	**Iteración asignada:** 1
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez	
Descripción: Después de obtener la dirección del fichero, se abre para ver el contenido	
Observaciones: Se almacena el fichero de revisión	
Prototipo de interfaz:	

Tabla 17. HU 4 - Mostrar las sentencias del fichero de revisión.

Historia de Usuario	
Número: 4	**Usuario:** Administrador
Nombre de Historia de Usuario: Mostrar las sentencias del fichero de revisión	
Prioridad en negocio: Alta	**Riesgo de Desarrollo:** Media
Puntos estimados: 0.5	**Iteración asignada:** 1
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez	
Descripción: Se muestran las posibles sentencias SQL a ejecutar en la revisión	
Observaciones: Se obtiene un listado de las posibles sentencias SQL a ejecutar	
Prototipo de interfaz:	

Tabla 18. HU 5 - Seleccionar las sentencias a ejecutar.

Historia de Usuario	
Número: 5	**Usuario:** Administrador
Nombre de Historia de Usuario: Seleccionar las sentencias a ejecutar	
Prioridad en negocio: Alta	**Riesgo de Desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 1
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez	
Descripción: De las sentencias SQL mostradas el especialista podrá seleccionar las que quiere ejecutar en ese momento	
Observaciones: Se obtiene el listado final de las sentencias SQL a ejecutar en la revisión	
Prototipo de interfaz:	

Tabla 19. HU 5 - Insertar credenciales para la conexión al SGBD.

Historia de Usuario	
Número: 6	**Usuario:** Administrador
Nombre de Historia de Usuario: Insertar credenciales para la conexión al SGBD	
Prioridad en negocio: Media	**Riesgo de Desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 2
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez	
Descripción: Se registra en el sistema los datos necesarios para la conexión al SBBD	
Observaciones: Se debe haber autenticado(usuario, contraseña) e ingresado las credenciales de conexión (dirección y puerto) para acceder al gestor	
Prototipo de interfaz:	

Tabla 20. HU 7 - Insertar credenciales para la conexión al Sistema Operativo.

Historia de Usuario	
Número: 7	**Usuario:** Administrador
Nombre de Historia de Usuario: Insertar credenciales para la conexión al Sistema Operativo	
Prioridad en negocio: Media	**Riesgo de Desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 2
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez	
Descripción: Se registra en el sistema los datos necesarios para la conexión al sistema operativo	
Observaciones: Se debe haber autenticado(usuario, contraseña) e ingresado la dirección física de la PC a la cual se quiere conectar	
Prototipo de interfaz:	

Tabla 21. HU 8 - Conectar al Sistema Operativo.

Historia de Usuario	
Número: 8	**Usuario:** Administrador
Nombre de Historia de Usuario: Conectar al Sistema Operativo	
Prioridad en negocio: Media	**Riesgo de Desarrollo:** Media
Puntos estimados: 2	**Iteración asignada:** 2
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez	
Descripción: Se establece la conexión de la aplicación con el Sistema Operativo.	
Observaciones:	
Prototipo de interfaz:	

Tabla 22. HU 9 - Conectar al Sistema Gestor de Bases de Datos.

Historia de Usuario	
Número: 9	**Usuario:** Administrador
Nombre de Historia de Usuario: Conectar al Sistema Gestor de Bases de Datos	
Prioridad en negocio: Media	**Riesgo de Desarrollo:** Media
Puntos estimados: 2	**Iteración asignada:** 2
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez	
Descripción: Una vez conectada la aplicación con el Sistema Operativo, se conecta con el Sistema Gestor de Bases de Datos.	
Observaciones:	
Prototipo de interfaz:	

Tabla 23. HU 10 - Ejecutar las sentencias de auditoría sobre el SGBD.

Historia de Usuario	
Número: 10	**Usuario:** Administrador
Nombre de Historia de Usuario: Ejecutar las sentencias de auditoría sobre el SGBD	
Prioridad en negocio: Media	**Riesgo de Desarrollo:** Media
Puntos estimados: 2	**Iteración asignada:** 2
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez	
Descripción: Después de conectar el sistema al SO y al SGBD se ejecuta el listado final de las sentencias SQL para la revisión y se almacenan los resultados	
Observaciones:	
Prototipo de interfaz:	

Tabla 24. HU 11 - Generar el fichero de resultados del proceso de auditoría.

Historia de Usuario	
Número: 11	**Usuario:** Administrador
Nombre de Historia de Usuario: Generar el fichero de resultados del proceso de auditoría	
Prioridad en negocio: Baja	**Riesgo de Desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 3
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez	
Descripción: Una vez terminada la ejecución de las sentencias SQL se genera un fichero con los resultados obtenidos en el proceso de auditoría	
Observaciones: El fichero generado será un XML	
Prototipo de interfaz:	

Tabla 25. HU 12 - Guardar el fichero de resultados.

Historia de Usuario	
Número: 12	**Usuario:** Administrador
Nombre de Historia de Usuario: Guardar el fichero de resultados	
Prioridad en negocio: Baja	**Riesgo de Desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 3
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez	
Descripción: La aplicación debe brindar la posibilidad de exportar un fichero con el resultado obtenido en el proceso de auditoría.	
Observaciones: Se almacena el fichero XML con los resultados de la ejecución del proceso de auditoría en la memoria interna del dispositivo.	
Prototipo de interfaz:	

Anexo 2: Tarjetas CRC.

Presentadores

Tabla 26. Tarjeta CRC HomePresenter

HomePresenter	
Responsabilidad:	Colaborador:
Captura los eventos de HomeActivity	HomeActivity

Tabla 27. Tarjeta CRC ConectarBDPresenter

ConectarBDPresenter	
Responsabilidad:	Colaborador:
Capturar los eventos de la vista ConectarBDActivity, ejecutar las funcionalidades de la clase AsyncTaskConnect y devolver los resultados a la vista. Además la conexión y los datos son validados.	ConectarPresenter AsyncTaskConnect

Tabla 28. Tarjeta CRC FileExplorerPresenter

FileExplorerPresenter	
Responsabilidad:	Colaborador:
Capturar los eventos de la vista FileExplorerActivity, procesarlos y devolver los resultados a la vista.	FileExplorerActivity

Tabla 29. Tarjeta CRC MostrarXMLPresenter

MostrarXMLPresenter	
Responsabilidad:	Colaborador:
Capturar los eventos de la vista MostrarXMLActivity, ejecutar las funcionalidades de la clase MyExpandableListAdapter, crea un adapter de MyExpandableListAdapter y devolver los resultados a la vista.	MostrarXMLActivity MyExpandableListAdapter

Tabla 30. Tarjeta CRC ConectarPresenter

ConectarPresenter	
Responsabilidad:	Colaborador:
Proporcionar una abstracción de conexión al SO y al SGBD.	ConectarActivity Conexion

Tabla 31. Tarjeta CRC ConectarSOPresenter

ConectarSOPresenter	
Responsabilidad:	Colaborador:
Capturar los eventos de la vista ConectarSOActivity, ejecutar las funcionalidades de la clase AsyncTaskConnect y devolver los resultados a la vista. Además la conexión y los datos son validados.	ConectarPresenter AsyncTaskConnect

Tabla 32. Tarjeta CRC EjecutarAuditoriaPresenter

EjecutarAuditoriaPresenter	
Responsabilidad:	Colaborador:
Capturar los eventos de la vista EjecutarAuditoriaActivity, ejecutar las funcionalidades de la clase AsyncTaskAuditoria, crea un adapter de ListAdapterResult y devolver los resultados a la vista.	EjecutarAuditoriaActivity ListAdapterResult AsyncTaskAuditoria

Tabla 33. Tarjeta CRC FileSavePresenter

FileSavePresenter	
Responsabilidad:	Colaborador:
Capturar los eventos de la vista FileSaveActivity, procesarlos y devolver los resultados a la vista.	FileSaveActivity

Vistas

Activities

Tabla 34. Tarjeta CRC HomeActivity

HomeActivity	
Responsabilidad:	Colaborador:
Mostrar una interfaz de inicio para la aplicación.	HomePresenter

Tabla 35. Tarjeta CRC CargarXMLActivity

CargarXMLActivity	
Responsabilidad:	Colaborador:
Mostrar una interfaz solicitando la ruta del fichero XML. Además muestra el fichero seleccionado con los atributos de dicho fichero.	CargarXMLPresenter

Tabla 36. Tarjeta CRC ConectarActivity

ConectarActivity	
Responsabilidad:	Colaborador:
Proporcionar una abstracción de solicitud de credenciales para la conexión a un SGBD o un SO.	ConectarPresenter

Tabla 37. Tarjeta CRC ConectarBDActivity

ConectarBDActivity	
Responsabilidad:	Colaborador:
Mostrar una interfaz solicitando las credenciales para la conexión a un SGBD. Además se ofrece opción de probar la conexión.	ConectarActivity

Tabla 38. Tarjeta CRC ConectarSOActivity

ConectarSOActivity

Responsabilidad:	Colaborador:
Mostrar una interfaz solicitando las credenciales para la conexión a un SO. Además se ofrece opción de probar la conexión.	ConectarActivity

Tabla 39. Tarjeta CRC EjecutarAuditoriaActivity

EjecutarAuditoriaActivity	
Responsabilidad:	Colaborador:
Mostrar una interfaz con la opción de ejecutar la auditoría. Además muestra una lista con una vista previa de los resultados del proceso de auditoría.	EjecutarAuditoriaPresenter

Tabla 40. Tarjeta CRC FileExplorerActivity

FileExplorerActivity	
Responsabilidad:	Colaborador:
Mostrar una interfaz con un explorador de ficharos.	FileExplorerPresenter

Tabla 41. Tarjeta CRC MostrarXMLActivity

MostrarXMLActivity	
Responsabilidad:	Colaborador:
Mostrar una interfaz con los datos del fichero XML. Además muestra la opción de seleccionar los scripts que serán ejecutados.	MostrarXMLPresenter

Tabla 42. Tarjeta CRC FileSaveActivity

FileSaveActivity	
Responsabilidad:	Colaborador:
Mostrar una interfaz con un explorador de ficharos, un cuadro de texto y un botón guardar.	FileSavePresenter

Adapter

Tabla 43. Tarjeta CRC ListAdapterResult

ListAdapterResult	
Responsabilidad:	Colaborador:
Crear un adapter para la lista de resultados del proceso de auditoría. Además captura los eventos de selección sobre esta lista.	

Tabla 44. Tarjeta CRC MyExpandableListAdapter

MyExpandableListAdapter	
Responsabilidad:	Colaborador:
Crear un adapter para la lista expandible de datos del fichero XML. Además captura los eventos de selección sobre esta lista.	XMLManager

Modelos

Tabla 45. Tarjeta CRC Indicador

Indicador	
Responsabilidad:	Colaborador:
Almacenar los datos de un indicador.	

Tabla 46. Tarjeta CRC Parametro

Parametro	
Responsabilidad:	Colaborador:
Almacenar los datos de un parámetro.	

Tabla 47. Tarjeta CRC XMLDatos

XMLDatos	
Responsabilidad:	Colaborador:

Almacenar los datos del fichero XML.	

Dominio

Tabla 48. Tarjeta CRC AsyncTaskAuditoria

AsyncTaskAuditoria	
Responsabilidad:	Colaborador:
Crea una tarea asíncrona para ejecutar el proceso de auditoría.	EjecutarAuditoriaPresenter ConexionBD ConexionSO

Tabla 49. Tarjeta CRC AsyncTaskConnect

AsyncTaskConnect	
Responsabilidad:	Colaborador:
Crear una tarea asíncrona para probar la conexión a un SGBD o un SO.	ConectarPresenter Conexion

Tabla 50. Tarjeta CRC Conexion

Conexion	
Responsabilidad:	Colaborador:
Proporcionar una interfaz del proceso de conexión, prueba y ejecución de script sobre un SGBD o un SO.	

Tabla 51. Tarjeta CRC ConexiónBD

ConexionBD	
Responsabilidad:	Colaborador:
Proporcionar una abstracción del proceso de conexión, prueba y ejecución de script sobre un SGBD.	

Tabla 52. Tarjeta CRC ConexionLinux

ConexionLinux	
Responsabilidad:	Colaborador:
Abrir, probar y cerrar la conexión a una terminal Linux. Además ejecuta los scripts de revisión sobre este SO.	ConexionSO

Tabla 53. Tarjeta CRC ConexionMySQL

ConexionMySQL	
Responsabilidad:	Colaborador:
Abrir, probar y cerrar la conexión al SGBD MySQL. Además ejecuta los scripts de revisión sobre este Gestor.	ConexionBD

Tabla 54. Tarjeta CRC ConexionOracle

ConexionOracle	
Responsabilidad:	Colaborador:
Abrir, probar y cerrar la conexión al SGBD Oracle. Además ejecuta los scripts de revisión sobre este Gestor.	ConexionBD

Tabla 55. Tarjeta CRC ConexionPostgres

ConexionPostgres	
Responsabilidad:	Colaborador:
Abrir, probar y cerrar la conexión al SGBD Postgres SQL. Además ejecuta los scripts de revisión sobre este Gestor.	ConexionBD

Tabla 56. Tarjeta CRC ConexionSO

ConexionSO	
Responsabilidad:	Colaborador:

Proporcionar una abstracción del proceso de conexión, prueba y ejecución de script sobre un SO.	Conexion

Tabla 57. Tarjeta CRC ConexionSQLServer

ConexionSQLServer	
Responsabilidad:	Colaborador:
Abrir, probar y cerrar la conexión al SGBD SQL Server. Además ejecuta los scripts de revisión sobre este Gestor.	ConexionBD

Tabla 58. Tarjeta CRC ConexionWindows

ConexionWindows	
Responsabilidad:	Colaborador:
Abrir, probar y cerrar la conexión a una terminal Windows. Además ejecuta los scripts de revisión sobre este SO.	ConexionSO

Application

Tabla 59. Tarjeta CRC ConexionManager

ConexionManager	
Responsabilidad:	Colaborador:
Inicializar una instancia a las clases ConexionBD y ConexionSO	ConexionBD ConexionSO

Tabla 60. Tarjeta CRC PortableApplication

PortableApplication	
Responsabilidad:	Colaborador:
Brindar una instancia global a las clases XMLManager y ConexionManager	XMLManager ConexionManager

Tabla 61. Tarjeta CRC XMLManager

XMLManager	
Responsabilidad:	Colaborador:
Almacenar los datos del XML cargado, una lista de los Indicadores y los parámetros, además almacena el tipo de auditoria a ejecutar	XMLDatos

Anexo 3: Tareas de Ingeniería.

Tabla 62. Tarea #1 Obtener ruta del fichero de revisión.

Tarea de Ingeniería	
Número de la tarea : 1.	**Historia de Usuario:** 1.
Nombre de la Tarea: Obtener ruta del fichero de revisión.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.5
Fecha Inicio: 14/2/15.	**Fecha fin:** 17/2/15.
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez.	
Descripción: Se obtiene la dirección física del fichero XML que se va a cargar para ejecutar la auditoría.	

Tabla 63. Tarea #2 Leer el fichero de revisión.

Tarea de Ingeniería	
Número de la tarea : 2.	**Historia de Usuario:** 2.
Nombre de la Tarea: Leer el fichero de revisión.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.5
Fecha Inicio: 18/2/15.	**Fecha fin:** 20/2/15.
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez.	
Descripción: Permite al especialista cargar y leer el fichero que será utilizado en la ejecución de la auditoría.	

Tabla 64. Tarea #3 Mostrar datos generales del fichero de revisión.

Tarea de Ingeniería	
Número de la tarea : 3.	**Historia de Usuario:** 3.
Nombre de la Tarea: Mostrar datos generales del fichero de revisión.	
Tipo: Desarrollo.	**Puntos Estimados:** 0.5
Fecha Inicio: 21/2/15.	**Fecha fin:** 24/2/15.

Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez.
Descripción: Se muestran los datos generales del fichero, dígase, sistema operativo, versión del sistema operativo, tipo de SGBD y versión del SGBD, así el especialista puede verificar que el archivo cargado es el que realmente desea utilizar.

Tabla 65. Tarea #4 Mostrar las sentencias del fichero de revisión.

Tarea de Ingeniería	
Número de la tarea : 4	**Historia de Usuario:** 4
Nombre de la Tarea: Mostrar las sentencias del fichero de revisión	
Tipo: Desarrollo.	**Puntos Estimados:** 0.5
Fecha Inicio: 25/2/15.	**Fecha fin:** 27/2/15.
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez.	
Descripción: Se muestra un listado de las posibles sentencias SQL a ejecutar en la revisión, obtenidas del fichero XML cargado.	

Tabla 66. Tarea# 7 Insertar credenciales para la conexión al Sistema Operativo.

Tarea de Ingeniería	
Número de la tarea : 7	**Historia de Usuario:** 7
Nombre de la Tarea: Insertar credenciales para la conexión al Sistema Operativo.	
Tipo: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: 14/3/15.	**Fecha fin:** 20/3/15.
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez.	
Descripción: Se registra en el sistema los datos necesarios para la conexión al sistema operativo, el especialista se debe haber autenticado (usuario, contraseña de la PC donde se encuentra el SGB) e ingresado la dirección física de la PC a la cual se quiere conectar.	

Tabla 68. Tarea #8 Conectar al Sistema Operativo.

Tarea de Ingeniería

Número de la tarea : 8	**Historia de Usuario:** 8
Nombre de la Tarea: Conectar al Sistema Operativo.	
Tipo: Desarrollo.	**Puntos Estimados:** 2
Fecha Inicio: 21/3/15.	**Fecha fin:** ¼/15.
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez.	
Descripción: Se validan los datos introducidos por el especialista y se establece la conexión de la aplicación con el Sistema Operativo.	

Tabla 68. Tarea #9 Conectar al Sistema Gestor de Bases de Datos.

Tarea de Ingeniería	
Número de la tarea : 9	**Historia de Usuario:** 9
Nombre de la Tarea: Conectar al Sistema Gestor de Bases de Datos.	
Tipo: Desarrollo.	**Puntos Estimados:** 2
Fecha Inicio: 2/4/15.	**Fecha fin:** 15/4/15.
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez.	
Descripción: Una vez conectada la aplicación con el Sistema Operativo y validados los datos de conexión al SGBD introducidos por el especialista, se conecta con el Sistema Gestor de Bases de Datos.	

Tabla 69. Tarea #10 Ejecutar las sentencias de auditoria sobre el SGBD.

Tarea de Ingeniería	
Número de la tarea : 10	**Historia de Usuario:** 10
Nombre de la Tarea: Ejecutar las sentencias de auditoría sobre el SGBD.	
Tipo: Desarrollo.	**Puntos Estimados:** 2
Fecha Inicio: 16/4/15.	**Fecha fin:** 30/4/15.
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez.	
Descripción: Se ejecuta el listado final de las sentencias SQL para la revisión y se almacenan los resultados.	

Tabla 70. Tarea #11 Generar el fichero de resultados del proceso de auditoría.

Tarea de Ingeniería	
Número de la tarea : 11	**Historia de Usuario:** 11
Nombre de la Tarea: Generar el fichero de resultados del proceso de auditoría.	
Tipo: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: 1/5/15.	**Fecha fin:** 6/5/15.
Programador(es) responsable(s): Arian Daniel Rodríguez, Alexey Rodríguez.	
Descripción: Una vez terminada la ejecución de las sentencias SQL al SGBD se genera un fichero con los resultados obtenidos en el proceso de auditoría.	

Anexo 4: Casos de prueba de aceptación.

Tabla 71. PA-2 Mostrar las sentencias del fichero de revisión.

Caso de prueba de aceptación	
Código : PA-2	**Historia de Usuario:** 4
Nombre: Mostrar las sentencias del fichero de revisión.	
Descripción: Prueba para la funcionalidad mostrar las sentencias del fichero de revisión.	
Condiciones de ejecución: El usuario deberá seleccionar la opción: Mostrar.	
Entrada/ Pasos de ejecución: Al usuario seleccionar la opción ´´Mostrar´´ la aplicación obtiene del fichero las sentencias que serán ejecutadas en la revisión.	
Resultado esperado: Una interfaz con listado de las sentencias a ejecutar en la revisión, cada sentencia podrá ser marcada o desmarcada para la ejecución según desee el usuario.	
Evaluación: Satisfactoria.	

Tabla 72. PA-4 Conectar al Sistema Operativo Linux.

Caso de prueba de aceptación	
Código : PA-4	**Historia de Usuario:** 8
Nombre: Conectar al Sistema Operativo Linux.	
Descripción: Prueba para la funcionalidad Conectar al Sistema Operativo Linux.	
Condiciones de ejecución: • El usuario deberá cargar un fichero de revisión correspondiente al SO Linux. • El usuario deberá introducir las credenciales para la conexión para el SGBD. • El usuario deberá seleccionar la opción: Probar Conexión.	
Entrada/ Pasos de ejecución: El usuario introduce las credenciales de conexión y escoge la opción ´´Probar Conexión´´. La aplicación construye la url y conforma una solicitud al servidor.	

Recoge la respuesta y muestra al usuario la respuesta.
Resultado esperado: Mensaje de respuesta.
Evaluación: Satisfactoria.

Tabla 73. PA-5 Ejecutar las sentencias de auditoría sobre el SGBD.

Caso de prueba de aceptación	
Código : PA-5	**Historia de Usuario:** 10
Nombre: Ejecutar las sentencias de auditoría sobre el SGBD.	
Descripción: Prueba para la funcionalidad ejecutar las sentencias de auditoria sobre el SGBD Postgres SQL.	
Condiciones de ejecución: • El usuario deberá cargar un fichero de revisión correspondiente al SGBD Postgres SQL. • El usuario deberá seleccionar al menos una sentencia a para ejecutar. • El usuario deberá seleccionar la opción: Ejecutar auditoría.	
Entrada/ Pasos de ejecución: El usuario selecciona la opción Ejecutar auditoría. La aplicación se conecta al SGBD Postgres SQL, ejecuta secuencialmente las sentencias y guarda los resultados.	
Resultado esperado: Listado de resultados.	
Evaluación: Satisfactoria.	