Universidad de las Ciencias Informáticas

Facultad 3

Personalización del módulo de seguridad para el marco de trabajo SAUXE del Sistema de Informatización de Tribunales.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Reinier Fernández Coello

Tutor: Ing. Darián González Ochoa

Co-Tutor: Ing. Dayron Abreus Ruiz

Junio 2012

"…aquí está una de las tareas de la juventud: empujar, dirigir con el ejemplo la producción del hombre de mañana. Y en esta producción, en esta dirección, está comprendida la producción de sí mismos…"

Ernesto Che Guevara

Declaración de autoría

Declaramos ser autores de este trabajo y autorizamos a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____

Reinier Fernández Coello

Autor

_____

Ing. Darián González Ochoa

Tutor

_____

Ing. Dayron Abreus Ruiz

Co-Tutor

Datos de contacto

Autor: Reinier Fernández Coello.

Correo Electrónico: rfcoello@estudiantes.uci.cu

Tutor: Ing. Darián González Ochoa.

Correo Electrónico: dochoa@uci.cu

Co-Tutor: Ing. Dayron Abreus Ruiz.

Correo Electrónico: dabreus@uci.cu

Agradecimientos

Agradecerle primeramente a toda mi familia por su apoyo y confianza en mí.

A mi mamá, que ha sido mi inspiración durante toda mi vida. Muchas gracias por ser la mejor del mundo, por apoyarme siempre en estos largos años de estudios y durante toda mi vida, por brindarme tu amor incondicional y por depositar toda tu confianza en mí.

A mi abuela Pancha, como diría yo mi otra madre, por su paciencia y entrega, por depositar en mí toda la confianza y el amor que un ser humano puede llevar consigo, gracias por existir.

A mis tías: Mumi, Eduarda, Leo, Odalys, Danae y a mi tío Ale. Gracias por brindarme su ayuda, por estar ahí siempre y por ser parte también de esto.

A mi primo Rober, mi hermano desde que tengo meses de nacido, gracias primo esto también es para ti; a mi primita Mehybis, te quiero primita.

A mi tía Daima y a mis hermanitos Neco y Yordan por simplemente existir, por inspirarme todos los días y poder ser un ejemplo ante ustedes.

A mi madrina María Luisa por su apoyo durante todos estos años.

A mi papa, a mi abuelo Compay, a mi padrastro Paquitín, mi hermanastro Gustavo, gracias por todo.

A mis vecinos que siempre han estado ahí apoyándome: Isabel, Lilian, Dania

A mis tutores por su entrega y su paciencia en la realización de este trabajo.

A los profesores Dailin y Ariel, gracias por sus consejos y su ayuda.

A todos mis compañeros y amigos que siempre voy a respetar, recordar y querer, por ayudarme y acompañarme en momentos difíciles y alegres, especialmente a Yudith, Wilber, Luis Alberto, Ania, Yoslenys, Josué, Odalis, Yuri, Denys.

A mis compañeros del antiguo 3109: Jimeno, Liván, Alexander, Sojo, Miguel Oliver, Manuel Yhon.

A mis compañeros de trabajo durante estos 3 años en el Comité Primario UJC, principalmente a Yadián.

A los profesores del proyecto: Arianna, Joan Jon y Yosvanys por su apoyo en la realización de esta tesis, muchas gracias.

A la Revolución, a Fidel y a la Universidad de las Ciencias Informáticas (UCI) por la oportunidad y permitir hacer mi sueño realidad.

A todos ustedes, muchas gracias.

Rey

Dedicatoria

Dedico especialmente este trabajo a mis madres Ana Coello Pérez y Francisca Coello Pérez, quienes por su constancia, dedicación, ternura y amor, soy, lo que he llegado a ser. Esto también es el fruto de sus quehaceres. Las quiero.

Rey

Resumen

Debido al proceso de informatización de la sociedad que se lleva a cabo en nuestro país, en la Universidad de las Ciencias Informáticas (UCI) se ha creado el Centro de Gobierno Electrónico (CEGEL) con el cual el Tribunal Supremo Popular (TSP) ha iniciado un proyecto en cooperación denominado Sistema de Informatización de los Tribunales (SIT). Su principal objetivo es la informatización de todos los procesos que se realizan en los tribunales para lograr una mayor agilidad en su tramitación. Para la creación de dicha solución informática se utiliza el marco de trabajo Sauxe, marco de trabajo orientado por componente que a su vez contiene la estructura adecuada para la realización de aplicaciones web de gestión. Sin embargo el uso de este marco de trabajo implica que no se cumplan ciertos requisitos no funcionales que atenta contra la seguridad de dicho sistema.

En la presente investigación se realiza un estudio sobre las metodologías y las herramientas existentes más utilizadas durante el proceso de desarrollo del software. Contiene la solución propuesta para la personalización del módulo de seguridad para el marco de trabajo Sauxe que garantizan el cumplimiento de los requisitos no funcionales de seguridad del SIT y finalmente la validación de la propuesta de solución mediante la realización de pruebas de caja negra y pruebas de caja blanca dirigidas a garantizar la calidad de la misma, punto principal para que el producto o software cumpla con las funcionalidades que necesita el cliente y que posea la calidad requerida, obteniéndose resultados positivos.

Palabras claves

Filtro de funcionalidades, Lista de control de acceso, marco de trabajo, pruebas de software, Sauxe.

Índice de contenidos

Índice de figuras

Índice de tablas

Introducción

Durante los últimos años las Tecnologías de la Información y las Comunicaciones (TIC) han alcanzado un alto desarrollo en la sociedad siendo aplicada a diferentes esferas de la vida. En el ámbito del Derecho el surgimiento de la Informática Jurídica es prueba de ello.

Cuba no está exenta a este proceso global, por lo que ha sido interés permanente del Estado el progreso de las TIC en todas las ramas de la sociedad, por tal motivo el Ministerio de la Informática y las Comunicaciones (MIC) lleva a cabo un proceso de informatización de la sociedad cubana. Es por ello que la Universidad de las Ciencias Informáticas (UCI) ha creado el Centro de Gobierno Electrónico (CEGEL) que tiene el objetivo de crear productos, servicios y soluciones informáticas de alta calidad. En función de esto el Tribunal Supremo de Cuba (TSP) ha iniciado un proyecto de cooperación con dicho centro con la creación del proyecto Sistema de Informatización de los Tribunales (SIT), con el objetivo de informatizar todos los procesos que se realizan en los tribunales.

Para el desarrollo de la solución informática planteada se cuenta con un marco de trabajo denominado Sauxe; el cual es orientado a componentes y provee la estructura genérica para el desarrollo de aplicaciones web de gestión, logrando una mayor integración y agilidad en el proceso de desarrollo.

Sin embargo el uso de este marco de trabajo implica que no se cumplan ciertos requisitos no funcionales que tienen un impacto en la seguridad del SIT. Ejemplo de esto es la asignación de determinadas funcionalidades definidas en el marco de trabajo Sauxe a los diferentes roles, que solo funciona de manera visual, es decir, la petición HTTP que ejecuta cada funcionalidad no es verificada contra los permisos que tiene un determinado rol sobre esa petición. Además de atender otras peticiones que no están asignadas a ningún rol, lo cual constituye una brecha de seguridad importante que puede ser aprovechada por cualquier atacante.

Por otro lado el SIT le muestra todos los expedientes a todos los roles sin tener en cuenta el estado y el trámite en que se encuentra un determinado expediente, esto implica que todo el personal que tiene acceso al sistema pueda acceder a todos los

expedientes y realizar cualquier operación sobre cada uno de ellos, lo cual va en contra de los requisitos no funcionales de seguridad planteado por el cliente.

Teniendo en cuenta la situación planteada, el **problema a resolver** queda expresado de la siguiente forma: Las deficiencias identificadas en la seguridad del marco de trabajo Sauxe impiden que se cumpla con los requisitos no funcionales de seguridad definidos para el Sistema de Informatización de Tribunales.

Teniendo como **objeto de estudio**: Marcos de trabajo para aplicaciones web de gestión, con el **objetivo** de: Personalizar la especificación e implementación de los componentes necesarios que garanticen mejoras en la seguridad para el marco de trabajo Sauxe del Sistema de Informatización de Tribunales.

Enmarcándose en el **campo de acción:** Marco de trabajo Sauxe.

Teniendo como referencia el problema a resolver y el objetivo general se plantea la siguiente **Idea a defender:** Con la personalización del módulo de seguridad para el marco de trabajo Sauxe se garantiza el cumplimiento de los requisitos no funcionales de seguridad del Sistema de Informatización de Tribunales.

Para dar cumplimiento al objetivo propuesto se han derivado un conjunto de **objetivos específicos** orientados fundamentalmente a proveer los elementos necesarios para la implementación de la solución, los cuales se listan a continuación:

- Realizar el marco teórico de la investigación.
- Realizar la implementación y la personalización de los componentes para el Sistema de Informatización de Tribunales.
- Validar los resultados obtenidos.
- Elaborar un informe sobre las modificaciones realizadas en el marco de trabajo Sauxe.

Para darle cumplimiento a los objetivos específicos se plantean las siguientes **tareas de la investigación:**

- Estudio basado en la seguridad en sistemas informáticos basado en el control de acceso.
- Estudio del proceso de desarrollo de software, de las herramientas y técnicas de implementación de software más usado.
- Realización de un estudio basado en el funcionamiento de los componentes de Zend Framework y Zend_Ext.
- Realización de un estudio referente a la implementación de las listas de control de acceso en Sauxe.
- Implementación del filtro de las peticiones que realiza cada funcionalidad.
- Implementación de las listas de control de acceso en el marco de trabajo Sauxe.
- Validación de los resultados obtenidos.
- Elaboración de un informe sobre las modificaciones realizadas.

Para dar respuesta a las tareas planteadas con anterioridad se proponen **Métodos Científicos** que se clasifican en:

Teóricos:

- **Análisis Histórico – Lógico:** para realizar un estudio de los componentes del marco de trabajo Sauxe con el objetivo de tener dominio de las características del mismo, así como un análisis de las metodologías, herramientas y lenguajes utilizados en el desarrollo de la propuesta de solución.
- **Analítico-Sintético:** para el estudio y análisis de la bibliografía, permitiendo una correcta formulación del problema y de los objetivos, la elaboración de las conclusiones y recomendaciones logrando resultados satisfactorios en la investigación.

Empíricos:

- **Entrevistas:** para obtener información relacionada con la propuesta de solución, así como toda la información referente al funcionamiento del marco de trabajo Sauxe.

El presente documento se estructura en 3 capítulos y anexos que incluyen los aspectos relacionados con la personalización del módulo de seguridad del marco de trabajo Sauxe para el SIT.

CAPÍTULO 1. Fundamentación Teórica: Se describen los aspectos y conceptos asociados al dominio del problema a resolver, siendo estos esenciales para entender el entorno del mismo. Se presenta el estado del arte de las técnicas implicadas en el objeto de estudio y el conjunto de tecnologías involucradas en el desarrollo de la propuesta y se realiza la justificación de las seleccionadas para la solución del problema.

CAPÍTULO 2. Propuesta de solución: Se hace una descripción de la propuesta de solución basada en las diferentes modificaciones que garanticen mejoras en la seguridad del marco de trabajo SAUXE de acuerdo al SIT; así como la explicación de los principales artefactos de la metodología empleada. Se exponen además los patrones empleados durante el desarrollo así como algunos rasgos esenciales en el funcionamiento del marco de trabajo.

CAPÍTULO 3. Validación de la solución: Se incluye en este capítulo un estudio de los conceptos fundamentales de las pruebas de software con sus clasificaciones. Se aplican casos de pruebas para validar la implementación del componente obtenido permitiendo validar la solución propuesta al detallar las no conformidades resultantes en cada iteración.

Capítulo 1: Fundamentación teórica

En el presente capítulo se aborda los principales conceptos necesarios para entender el desarrollo de esta investigación, como por ejemplo: seguridad en los sistemas informáticos, lista de control de acceso, framework, entre otros. De igual forma se realiza un estudio del uso de las listas de control de acceso en los diferentes marcos de trabajo. Se presenta el estado del arte de las técnicas implicadas en el objeto de estudio y el conjunto de tecnologías involucradas en el desarrollo de la propuesta y se realiza la justificación de las seleccionadas para la solución del problema.

1.1. Seguridad en sistemas informáticos

La seguridad informática es un conjunto de métodos y herramientas destinados a proteger la integridad y la privacidad de la información almacenada en un sistema informático. Un sistema informático puede ser protegido desde un punto de vista lógico (con el desarrollo de software) o físico (vinculado al mantenimiento eléctrico, por ejemplo). (Ramió Aguirre, 2006)

Por la particularidad del presente trabajo, el desarrollo del mismo está enmarcado en la protección desde el punto de vista lógico durante el desarrollo del SIT con la restricción o separación de los privilegios mediante la lista de control de acceso y el filtro de las peticiones HTTP que realiza cada funcionalidad.

1.1.1. Lista de control de acceso

Una lista de control de acceso o ACL (del inglés, Access Control List) es un concepto de seguridad informática usado para fomentar la separación de privilegios. Es una forma de determinar los permisos de acceso apropiados a un determinado objeto, dependiendo de ciertos aspectos del proceso que hace el pedido, es decir, conceder solo los permisos necesarios para la realización de un trabajo dado. (Solís, 2009)

En la presente investigación se precisa el control de acceso a diferentes objetos mediante el rol que este autenticado, para eso se realiza el estudio sobre la ACL basada en Roles (RBAC por sus siglas en ingles). RBAC difiere principalmente de los

sistemas tradicionales basados en ACL's, en que su arquitectura está pensada para asignar permisos a operaciones más que a objetos concretos (ficheros, registros). Siendo por tanto su principal ventaja el permitir alinear la infraestructura de seguridad de la información con los objetivos de negocio de una forma natural. (Practical Role-Based Access Control, 2009). La arquitectura de los sistemas basados en roles se puede representar de la siguiente forma:

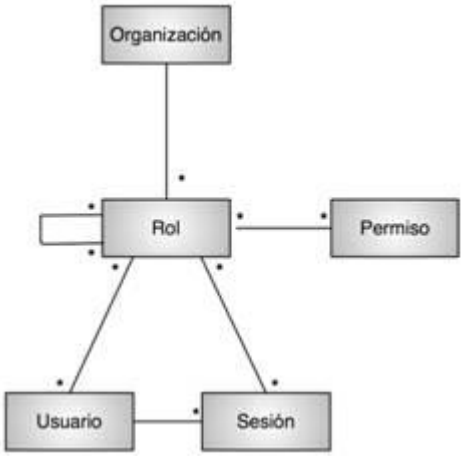


Figura 1: Arquitectura general de los sistemas basados en ACL

La idea consiste en la asignación de permisos (lectura, escritura), sobre objetos o procesos, a los distintos roles ya existentes en la organización (Revisor, Editor, Calidad, Dirección, entre otros). De forma que posteriormente se asignen estos roles a los distintos usuarios según las responsabilidades que tienen que adoptar en cada momento (sesión). (Solís, 2009)

Un ejemplo puede ser el típico flujo de trabajo que se genera en la aprobación de una factura de un proveedor en cualquier Sistema de Gestión Documental. En este caso, dicha factura pasará por distintos estados intermedios en los que distintos usuarios (Jefe de compras, Jefe de un departamento, etc.) con rol "Revisor" le darán el visto bueno hasta que finalmente sea aprobada por Dirección. Durante ese flujo dichos usuarios jugarán el Rol "Revisor" de forma consecutiva, pero una vez esté aprobada la

factura para su pago, dichos usuarios adquieren el Rol "Lector", que a diferencia del Rol anterior, solo dejará consultar la factura.

1.2. Funcionalidad

Una funcionalidad, en el contexto del marco de trabajo Sauxe, es un conjunto de operaciones básicas que gestionan la información necesaria para la realización de los flujos de dicho marco de trabajo. (Gómez Baryolo, et al., 2010)

1.3. Marcos de Trabajos (Framework)

En el desarrollo de software, un marco de trabajo o framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Normalmente, un marco de trabajo incluye soporte de programas y librerías, además de un lenguaje de programación que sirve de apoyo en el desarrollo y la integración de los diferentes componentes de una aplicación. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. En general estos se diseñan para resolver la mayor parte de la complejidad de los desarrollos, aprovechando las mejores prácticas y facilitando la construcción de nuevas aplicaciones. (Informática, 2006)

1.3.1. Características principales de los marcos de trabajo

A continuación se enuncian una serie de características que se encuentran en la mayoría de los marcos de trabajos existentes:

- **Abstracción de URLs y sesiones**: No es necesario manipular directamente las URLs ni las sesiones, el marco de trabajo ya se encarga de hacerlo.
- **Acceso a datos**: Incluyen las herramientas e interfaces necesarias para integrarse con herramientas de acceso a datos, en bases de datos, XML, etc.
- **Controladores**: La mayoría de los marcos de trabajos implementa una serie de controladores para gestionar eventos, como una introducción de datos mediante un formulario o el acceso a una página. Estos controladores suelen ser fácilmente adaptables a las necesidades de un proyecto concreto.

- **Autentificación y control de acceso**: Incluyen mecanismos para la identificación de usuarios mediante *nombre de usuario* y *contraseña* lo que permite restringir el acceso a determinadas páginas de determinados usuarios.
- **Separación entre diseño y contenido**: La mayoría de los marcos de trabajos implementan el patrón MVC (modelo - vista - controlador). Este patrón organiza la aplicación en tres modelos separados, el primero es un modelo que representa los datos de la aplicación y sus reglas de negocio, el segundo es un conjunto de vistas que representa los formularios de entrada y salida de información, el tercero es un conjunto de controladores que procesa las peticiones de los usuarios y controla el flujo de ejecución del sistema. (Sánchez, 2006)

1.3.2. Marcos de trabajos que usan el sistema de las ACL

1.3.2.1. .NET Framework

.NET Framework es un componente integral de Windows que admite la compilación y la ejecución de diversas generaciones de aplicaciones y servicios Web. Los componentes clave de .NET Framework son Common Language Runtime (CLR) y la biblioteca de clases .NET Framework, que incluye ADO.NET, ASP.NET, formularios Windows Forms y Windows Presentation Foundation (WPF). .NET Framework proporciona un entorno de ejecución administrado, un desarrollo e implementación simplificada y la integración con una gran variedad de lenguajes de programación.

En las clases del espacio de nombres System.Security.AccessControl se permite crear o modificar mediante programación las listas de control de acceso discrecional (DACL) y las listas de control de acceso al sistema (SACL) para un número de recursos protegidos como archivos, carpetas, etc. Las listas DACL le permiten controlar mediante programación el acceso a los recursos protegidos, mientras que las listas SACL le permiten controlar mediante programación las directivas de auditoría del sistema de recursos protegidos. Por ejemplo, puede utilizar las clases DACL para asegurarse de que sólo un administrador puede leer un archivo; puede utilizar las

clases SACL para asegurarse de que se han registrado todos los intentos correctos para abrir el archivo.

.NET Framework proporciona acceso a las ACL para los recursos siguientes:

- Claves criptográficas
- Directorios
- Identificadores de espera de evento
- Archivos
- Mutexes
- Claves del Registro
- Semáforos

Cada uno de estos recursos tiene varias clases que se pueden utilizar para crear y modificar las ACL. (Serrano, 2012)

1.3.2.2. Zend Framework (ZF)

ZF es un framework de código abierto para desarrollar aplicaciones web y servicios web con PHP5. ZF es una implementación que usa código 100% orientado a objetos. La estructura de los componentes de ZF es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado.

Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de ZF conforman un potente y extensible marco de trabajo de aplicaciones web al combinarse. En ZF el componente Zend_Acl provee la implementación de un sistema simple y flexible de las ACL para la administración de privilegios. (Boda, et al., 2009)

1.3.2.3. CakePHP

CakePHP es una buena opción tanto para desarrolladores PHP principiantes como para programadores más avanzados enfocados al rápido desarrollo. Su sistema de

soporte creciente, simplicidad y escalabilidad hacen que CakePHP sea una de los marcos de trabajos para PHP más populares hoy en día.

Entre sus principales características se encuentran el uso del sistema ACL. La primera implementación de ACL en Cake fue basada en archivos INI almacenados en el directorio de instalación de Cake. Si bien es útil y estable, se recomienda que se utilice la solución de ACL apoyada en la base de datos. (Cake Software Foundation, 2010)

1.3.2.4. Spring Security

Spring Security proporciona servicios de seguridad dentro de Spring. Proporciona un sistema de autenticación a través del cual los usuarios pueden autenticarse y acceder a múltiples aplicaciones a través de un único punto de entrada. Otras características avanzadas que proporciona son soporte para proxy y refrescamiento forzado del autenticado. Tiene completa integración con Spring, utiliza los propios mecanismos de configuración de Spring y asegura la seguridad a nivel de instancia de objetos del dominio. En muchas aplicaciones es deseable definir las ACL para instancias de objetos del dominio individuales. Spring Security proporciona un completo paquete ACL con características que incluyen máscaras de bits, herencia de permisos, un repositorio utilizando JDBC[1], caché y un diseño utilizando interfaces. (Alex, et al.)

1.4. Metodologías de desarrollo

Las Metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto de software.

Dichas metodologías pretenden guiar a los desarrolladores a crear un software; pero los requisitos de un software a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del software.

Se podrían clasificar en dos grandes grupos:

[1] JDBC (Java Database Connectivity) es un API para trabajar con bases de datos desde Java

- **Metodologías ligeras/ágiles**: son aquellas metodologías orientadas a la interacción con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando.
- **Metodologías pesadas/tradicionales**: son aquellas metodologías orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. (Carrillo Pérez, et al., 2008)

De una forma u otra, todas han contribuido en el desarrollo de la industria del software, sirviéndoles de guía a los desarrolladores para un mejor desempeño y cumplimiento de los requisitos de un determinado producto. Todas ellas tienen su forma particular de ser útil, pues no plantean lo mismo y no se pueden utilizar en cualquier proyecto, debido a que no existe una metodología estándar para cualquier tipo de desarrollo en la industria del software, pues todo depende del cliente, o de la empresa, de las características de la misma, entre otros factores como la cantidad de personal, el tiempo en que se debe de entregar el producto y otros agentes que influyen en el desarrollo de un software.
Para el desarrollo de la investigación en cuestión, se necesita una metodología configurable, adaptable, de poca documentación pero no nula, que responda a los intereses del cliente y el producto, que mantenga al cliente interesado, que le muestre productos funcionales en poco tiempo, y que permita que estos productos sean simples y fáciles de modificar, en aras de satisfacer los cambios recurrentes del jefe del producto. Estos requisitos determinan el uso de la corriente ágil y no la tradicional.

Dentro de las diferentes metodologías ágiles que existen se encuentran:

- XP o Programación Extrema.
- Scrum

Por estas razones, se centrará el estudio comparativo en Scrum y XP ambas metodologías ágiles, seleccionadas por ser de las más utilizadas en la universidad, aportando resultados satisfactorios, además por ser las más cercanas a los intereses que se persiguen con el desarrollo en cuestión.

1.4.1. Scrum

Scrum es una metodología para la gestión de proyectos, expuesta por Hirotaka Takeuchi e Ikujiro Nonaka, en el artículo The New Product Development Game "El precepto básico de esta metodología es que el mercado competitivo de los productos tecnológicos, además de los conceptos básicos de calidad, coste y diferenciación, exige también rapidez y flexibilidad". (Palacio, 2008)

Sin embargo, más que una metodología de desarrollo de software, es una forma de auto-gestión de los equipos de programadores. Un grupo de programadores deciden cómo hacer sus tareas y cuánto van a tardar en ello. Scrum ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro. Es una metodología ágil que no responde a ninguna moda, sino a una necesidad realmente demandada en el desarrollo del software. Scrum evita la generación documental lo cual no significa que no se deba o no se pueda documentar, si no que no se exige documentar para iniciar un proyecto. (Pressman, 2001)

La idea principal de esta metodología es comenzar a trabajar y obtener resultados desde el primer momento, de tal forma que el cliente observe los avances y muestre su satisfacción del resultado obtenido.

Características principales de Scrum:

- Es una metodología de desarrollo ágil.
- Está pensada para equipos de desarrollos pequeños.
- Se especifican pocos artefactos eliminando el "papeleo" innecesario, dedicando más tiempo a la implementación.
- Permite la entrega de un producto funcional al finalizar cada Sprint.

- Da la posibilidad de ajustar la funcionalidad en base a la necesidad de negocio del cliente.
- Permite hacer una visualización del proyecto diaria.
- Tiene un alcance acotado y viable.
- Se aplica en equipos integrados y comprometidos con el proyecto y que se auto administran.
- No es una metodología de análisis, ni de diseño (aunque puede adaptarse para que lo sea) sino de gestión del trabajo.

Los artefactos de Scrum son:

- Product Backlog (Pila de Tareas del Producto): corresponde con todas las tareas, funcionalidades o requerimientos a realizar.
- Sprint Backlog (Pila de Tareas del Sprint o Ciclo): corresponde con una o más tareas que provienen del Product Backlog de donde se obtiene una o más tareas que van a formar parte del Sprint Backlog. Estas se deben realizar (recomendado) en unas 2 ó 4 semanas. Eso debe de ser marcado antes de iniciar el Sprint. Una norma fundamental es que mientras un Sprint Backlog se inicia, éste no puede ser alterado o modificado.
- Incremento: se refiere al adelanto funcional que se tiene como resultado al finalizar cada Sprint Backlog. Se trata de un resultado completamente terminado y en condiciones de ser usado.

1.4.2. Extreme Programing (XP)

Extreme Programing, o Programación Extrema, fue desarrollada por Kent Beck. Consiste básicamente en ajustarse estrictamente a una serie de reglas que se centran en las necesidades del cliente para lograr un producto de buena calidad en poco tiempo. Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software. Promueve el trabajo en equipo, preocupándose en todo momento del aprendizaje de los desarrolladores y

estableciendo un buen clima de trabajo. Este tipo de método se basa en una retroalimentación continuada entre el cliente y el equipo de desarrollo con una comunicación fluida entre todos los participantes. Este tipo de programación es la adecuada para los proyectos con requisitos imprecisos, muy cambiantes y con un riesgo técnico excesivo.

Está pensada para un grupo pequeño y muy integrado (máximo 12 personas) dónde la comunicación sea más factible que en grupos de desarrollo grandes.

Sus características principales son:

- Comunicación: Los programadores están en constante comunicación con los clientes para satisfacer sus requisitos y responder rápidamente a los cambios de los mismos. Muchos problemas que surgen en los proyectos se deben a que después de concretar los requisitos que debe cumplir el programa no hay una revisión de los mismos, pudiendo dejar olvidados puntos importantes.
- Simplicidad: Codificación y diseños simples y claros. Muchos diseños son tan complicados que cuando se quieren ampliar resulta imposible hacerlo y se tienen que desechar y partir de cero.
- Retroalimentación: Mediante la retroalimentación se ofrece al cliente la posibilidad de conseguir un sistema apto a sus necesidades ya que se le va mostrando el proyecto a tiempo para poder ser cambiado y poder retroceder a una fase anterior para rediseñarlo a su gusto.
- Coraje: Hay que tener valor para comunicarse con el cliente y enfatizar algunos puntos, a pesar de que esto pueda dar sensación de ignorancia por parte del programador, hay que tener coraje para mantener un diseño simple y no optar por el camino más fácil y por último hay que tener valor y confiar en que la retroalimentación sea efectiva. (Sommerville, 2002)

1.4.3. Fundamentación de la selección de la metodología

Como se ha podido apreciar las metodologías de desarrollo de software antes tratadas tienen su forma de guiar a los desarrolladores de software, teniendo sus métodos

particulares. Además el uso de estas depende del producto que se quiera desarrollar, del usuario final, de las características de la empresa, entre otras circunstancias que se presenten.

Para el desarrollo del presente trabajo se utilizará como metodología de desarrollo de software a Scrum, debido a que se puede gestionar las expectativas del cliente de una forma más efectiva, se realiza gran parte del producto en poco tiempo y el cliente ya puede empezar a utilizar esta funcionalidad y por último evita el estancamiento ya que las reuniones que plantea Scrum se dedican a inconvenientes recientes. Sin embargo con XP esto no se puede lograr debido a que resulta muy complicado planear el proyecto y establecer el costo y duración del mismo así como medir los avances del proyecto, pues es muy difícil el uso de una medida estándar. Es por esto y por el resto de las características explicadas anteriormente que se selecciona para el progreso del presente trabajo la metodología Scrum.

1.5. Marco de Trabajo Sauxe

A partir de la extensión de algunos componentes de Zend Framework surge Zend Ext, desarrollado por el Departamento de Tecnología y la UCID (Unidad de Compatibilización Integración y Desarrollo de Software para la Defensa), con el objetivo de crear un marco de trabajo extensible y configurable, centrando el desarrollo de las aplicaciones en la lógica del negocio, en las interfaces de usuario, alejando a los programadores de los detalles arquitectónicos, con soporte para entornos multientidad y para una arquitectura de sistema orientada a componentes. (Baryolo, 2008)

La unión de Zend Ext, Doctrine y Extjs dio lugar al Marco de Trabajo Sauxe que contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. (Baryolo, 2008)

Para el desarrollo del SIT, se decidió por parte del equipo de arquitectura la utilización de este marco de trabajo en su versión 2.0 que tiene definido para su uso una serie de herramientas y tecnologías específicas, como por ejemplo:

- PostgreSQL 8.4: como Sistema Gestor de Base Datos (SGDB)
- PHP 5.2.5: como lenguaje de programación del lado del servidor.
- Java Script 1.8: como lenguaje de programación del lado del cliente.
- ExtJS 2.2: para la realización de interfaces para los usuarios.
- Doctrine 1.2.1: para la persistencia de las clases de la base de datos.

Todas estas herramientas son libres siguiendo el paradigma de independencia tecnológica por el cual apuesta nuestro país.

1.5.1. PostgresSQL 8.4

Es un poderoso sistema de base de datos relacional de código abierto. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de confiabilidad, integridad de datos y corrección. Funciona en todos los principales sistemas operativos, incluyendo Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), y Windows. Es totalmente compatible con ACID[2], tiene soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados (en varios idiomas). También es compatible con el almacenamiento de objetos binarios, incluyendo imágenes, sonidos o vídeo. Tiene interfaces de programación nativa de C / C + +, Java, NET, Perl, Python, Ruby, Tcl, ODBC, entre otros, y la documentación en varios idiomas.

PostgreSQL cuenta con sofisticadas funciones como el Control de Concurrencia Multi-Versión (MVCC), punto en el tiempo de recuperación, replicación asincrónica, transacciones anidadas (puntos de retorno), copias de seguridad en línea, un planificador optimizador de consultas sofisticadas, y escribir por delante de registro para la tolerancia a fallos. Es compatible con conjuntos de caracteres internacionales, codificación de caracteres multi-byte, Unicode, y es consciente de la configuración regional para la clasificación, caso-sensibilidad y el formato. Es altamente escalable, tanto en la enorme cantidad de datos que puede manejar como en el número de

[2] **ACID** es un acrónimo de **A**tomicity, **C**onsistency, **I**solation and **D**urability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español.

usuarios concurrentes que puede acomodar. No hay sistemas activos de PostgreSQL en entornos de producción que manejan en exceso de 4 terabytes de datos. (PostgreSQL, 2010)

1.5.2. Doctrine 1.2.1

Doctrine es un potente y completo sistema mapeador relacional de objetos (ORM) para PHP con un capa de abstracción de bases de datos (DBAL) incorporado que permite exportar una base de datos a sus clases correspondientes y viceversa, o sea, a partir de las clases creadas y siguiendo las especificaciones de ORM, generar las tablas de la base de datos. Una de sus principales características es la opción de escribir las consultas de base de datos en un objeto con una propiedad orientada al lenguaje SQL, llamada Doctrine Query Language (DQL), inspirada en Hibernate HQL. Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código innecesaria.

Doctrine utiliza una capa de abstracción capaz de traducir a diferentes tipos de base de datos, esto permite que en caso de cambiar de proyecto de base de datos se pueda realizar sin ningún tipo de complicación. (Smith, 2009.)

1.5.3. Lenguaje del lado del cliente

Java Script 1.8

Java Script es un lenguaje de programación interpretado, surgido como resultado del estándar ECMAScript[3] y desarrollado por **Netscape**, es orientado a objetos, imperativo, y dinámico. Se utiliza esencialmente del lado del cliente, implementado como parte del navegador web permitiendo mejoras significantes en las interfaces de usuario puesto que es muy eficiente en los temas de validaciones y para el desarrollo de web dinámicas. (Flanagan, 2002)

Sus características más significativas son:

[3] **ECMAScript** es una especificación de lenguaje de programación publicada por ECMA International. El desarrollo empezó en 1996 y estuvo basado en el popular lenguaje **JavaScript** propuesto como estándar por **Netscape Communications Corporation**. Actualmente está aceptado como el estándar ISO 16262.

- Es un lenguaje orientado a eventos. Cuando un usuario interactúa con un enlace o mueve el puntero sobre algún otro elemento HTML se pueden producir diferentes eventos y a través de JavaScript se pueden desarrollar scripts que ejecuten funciones en respuesta a estos eventos.
- Es un lenguaje interpretado, o sea, no necesita ser compilado. El navegador del usuario se encarga de interpretar las sentencias JavaScript contenidas en una página HTML y ejecutarlas adecuadamente.
- Es soportado por la mayoría de los navegadores como Internet Explorer, Google Chrome, Opera, Mozilla Firefox, entre otros.

1.5.4. Lenguaje del lado del servidor

PHP 5.2.5

PHP es un acrónimo recursivo que significa Hypertext Pre-Processor (inicialmente PHP Tools, o, Personal Home Page Tools) es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor, fue escrito originalmente por Rasmus Lerdorf.

Algunas ventajas de PHP:

- **Rendimiento:** El motor de PHP 5.2.5 fue rediseñado completamente con un administrador de memoria optimizado para mejorar el rendimiento.
- **Portabilidad:** PHP está disponible para sistemas operativos como UNIX, Microsoft Windows, Mac OS y OS/2. Los programas escritos en este lenguaje son portables entre plataformas.
- **Facilidad de uso:** Su sintaxis es clara y consistente, cuenta además con documentación exhaustiva para todas las funciones de su núcleo.
- **Código Abierto:** El lenguaje es desarrollado por una comunidad de programadores voluntarios que publican su código libremente en la Web y puede ser usado sin el pago de licencias o inversiones en hardware costoso. Esto reduce el costo de la producción del software sin afectar la flexibilidad o confiabilidad.

- **Soporte comunitario:** Cuenta con amplio soporte gracias a la numerosa comunidad de programadores que lo usan en todo el mundo. (Aulbach, et al., 2002)

1.5.5. ExtJS 2.2

Es una biblioteca Java Script código abierto de alto rendimiento para la creación y desarrollo de aplicaciones web dinámicas. Su potencia radica en la rica colección de componentes para el diseño de interfaces de usuarios del lado del cliente haciendo uso extensivo de Ajax. Permite la creación de aplicaciones complejas utilizando componentes predefinidos. Es extensible para la gran mayoría de los navegadores, evitando el tedioso problema de validar el código para cada uno de estos. Entre sus principales ventajas se encuentra el balance entre Cliente-Servidor, distribuyendo la carga de procesamiento en el último, este al tener menor carga, maneja los clientes de manera más eficiente. La comunicación asíncrona permite el intercambio de información con el servidor sin necesidad de pedirle una acción al usuario, dando la libertad de cargar la información sin que este lo note. (Barrios Díaz, 2007)

1.6. Patrones de diseño

Un patrón es una solución a un problema determinado, al que se le da un nombre, y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias. Durante la etapa de diseño de un software se conciben un grupo de patrones que de acuerdo a la solución que brindan cubren una necesidad determinada en el diseño.

Existen dos grupos de patrones de diseño principales, los Patrones Generales de Software para Asignación de Responsabilidades (GRASP siglas de General Responsibility Assignment Software Patterns en inglés) y los patrones correspondientes al Grupo de los Cuatro (GOF siglas de Gang of Four en inglés) que es el nombre con el que se conoce por lo general a los autores del libro DesignPatterns. Los patrones

GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

El marco de trabajo Sauxe además de utilizar el estilo arquitectónico **Modelo-Vista-Controlador**, trae implícito el empleo de algunos patrones de diseño, como por ejemplo los patrones de **Bajo Acoplamiento** y **Controlador**, pertenecientes a los GRASP. De igual forma trae por defecto el patrón **Singleton**, correspondiente al GOF. Y por último el patrón **Inversión de control** (IoC).

1.7. Herramienta de prototipado web

Un prototipo es una pequeña muestra, de funcionalidad limitada, de cómo sería el producto final una vez terminado. La técnica del prototipado aplicado al desarrollo de productos web tiene muchas utilidades y aplicaciones, ya que permite que se pueda verificar los requerimientos del producto antes de que se inicie su desarrollo. Permite fijar presentaciones en las que se puede solicitar la aprobación del prototipo del software, así como especificar el contenido o detectar errores de diseño antes de iniciar el proceso de desarrollo.

Algunas de las herramientas de prototipado web son:

- Balsamiq Mockups
- Axure RP Pro
- Microsoft Office Visio 2007

Por parte del equipo de arquitectura del proyecto Tribunales se decidió la utilización de la herramienta Axure RP Pro en su versión 5.5, por ser diseñada explícitamente para el prototipado, posee todos los elementos necesarios para la creación de prototipos amigables que pueden ser exportados como un proyecto o como fotos de fácil uso para el equipo de desarrollo, también pueden generarse como páginas HTML dinámicas, funcionalidad que permite al usuario interactuar de forma directa con los prototipos, observando el flujo consecutivo de los eventos, permitiéndoles de esta manera tener

una idea más clara de cómo será la aplicación en un futuro y de esta forma sentirse identificados y motivados con la solución a elaborar. (Navarro, 2009)

1.7.1. Axure RP Pro 5.5

Axure RP es una aplicación ideal para crear prototipos y especificaciones muy precisas para páginas web. Se trata de una herramienta especializada en la tarea, ya que cuenta con todo lo necesario para crear prototipos de forma más eficiente. Permite componer la página web visualmente, añadiendo, quitando y modificando los elementos con suma facilidad. (Gómez, 2007)

Axure RP es una herramienta fácil de usar dónde se producen de forma instantánea modelos funcionales. Esta herramienta posee un alto grado de personalización y una versión de administración que permite la creación de modelos interactivos con anotaciones y facilita la creación de un diseño efectivo. (Gómez, 2007)

Ventajas del Axure:

- Brinda flexibilidad y sencillez en su uso.
- Crea un documento con especificaciones.
- Permite trabajar en un proyecto a varias personas a la vez. (Help, 2010)

1.8. Herramientas CASE

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, generación automática de parte del código con el diseño dado, compilación automática, documentación o detección de errores entre otras (case-tools, 2011).

Algunas herramientas CASE son:

- ER/Studio Enterprise
- DBDesigner 4

Por parte del equipo de arquitectura del proyecto se selecciona para el modelado de los datos el ER/Studio Enterprise en su versión 7.1, una herramienta fácil de usar y multinivel, para el diseño y construcción de bases de datos tanto a nivel físico como lógico, direccionando las necesidades diarias de los desarrolladores que construyen y mantienen aplicaciones de bases de datos grandes y de gran complejidad. (Navarro, 2009)

1.8.1. ER/Studio Enterprise 7.1

Dentro de la industria de software es una herramienta de modelado para el análisis, visualización y comunicación de base de datos, aplicaciones de diseños de datos y arquitectura de la información. La combinación de procesos, datos, modelado UML, y presentación de reportes en un potente entorno de diseño en multi-niveles son algunas de sus principales características. (Architect, 2009)

ER/Studio está equipado para crear y manejar diseños de bases de datos funcionales y confiables. Ofrece fuertes capacidades de diseño lógico, sincronización bidireccional de los diseños físicos y lógicos, construcción automática de bases de datos, documentación y fácil creación de reportes.

El uso del ER / Studio Enterprise trae como beneficios:

- Promoción de la reutilización de datos y la colaboración en tiempo real a través de una gestión eficaz del modelo de la empresa y la capacidad de publicación de metadatos.
- Mejora de la visibilidad y la calidad de información con la ingeniería inversa y el soporte al ciclo de vida completo de base de datos.
- Comunicación efectiva de los modelos de toda la empresa con informes de metadatos en tiempo real y herramientas de publicación.

1.9. Entornos de Desarrollo Integrados (IDE)

Un Entorno de Desarrollo Integrado (IDE siglas de Integrated Development Environment en inglés), consiste básicamente en un programa informático que previamente ha sido instalado en la máquina del desarrollador y cuyo principal objetivo es la elaboración de otras aplicaciones, es decir, es un entorno de programación que ha sido empaquetado como *"Software",* que va a contener un editor de código, un compilador, un depurador y un constructor de interfaz gráfica, que van a facilitar las diferentes tareas necesarias en el desarrollo y mantenimiento de cualquier tipo de aplicación.

En la actualidad hay infinidades de entornos de desarrollos integrados, entre los cuales el equipo de arquitectura del proyecto decidió utilizar para el desarrollo del SIT el NetBeans en su versión 7.0.1 ya que es un proyecto de código abierto, soporta lenguajes dinámicos como PHP y Java Script, es multi-plataforma, tiene una interfaz muy amigable e intuitiva, tiene todas las herramientas para crear aplicaciones profesionales ya sean de escritorio, empresariales, web, móviles y aplicaciones SOA.(Navarro, 2009)

1.9.1. NetBeans 7.0.1

NetBeans IDE es un producto libre y gratuito sin restricciones de uso, de código abierto, escrito completamente en Java, teniendo el código fuente disponible para su reutilización bajo las licencias CDDL y la GPL. Es multiplataforma, presenta una interfaz muy amigable e intuitiva, soporta disímiles lenguajes, además en sus versiones más recientes se le han ido acoplando nuevas funcionalidades que resultan de gran utilidad para el desarrollo web, facilitando la creación de proyectos en PHP. NetBeans en su versión 7.0.1 presenta un sistema para examinar todos los directorios de cada proyecto, haciendo reconocimiento y carga de clases, métodos y objetos, para acelerar la programación, proponiendo un esqueleto para organizar el código fuente, el editor conjuntamente integra los lenguajes como HTML, JavaScript y CSS, permitiendo la

refactorización y búsqueda de usos para CSS y lenguajes de tipo HTML, y el completamiento de código y links para atributos de CSS. Por otra parte el editor de PHP es muy ágil y robusto, siendo eficiente en el completamiento de código, reconocimiento de sintaxis y presentando un potente depurador que facilita la programación.

NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para otros lenguajes de programación. (Netbeans)

Ventajas:

- Auto-completado y documentación de funciones PHP: Rápido acceso a la documentación de PHP, y si se necesita más información se provee el link directo a la función.
- Auto-completado de código propio: Esto es una consecuencia del punto anterior, al documentar código con el formato esperado, estos serán mostrados.
- Atajos de teclado muy útiles: Permite a través de varios comandos la ejecución de numerosas funcionalidades y existen muchas más como integración con Xdebug, soporte para Symfony, Zend Framework, Smarty, entre otros.

1.10. Pruebas de calidad

El desarrollo de un software es algo complejo y son innumerables las posibilidades de errores por lo que este ha de ir acompañado de alguna actividad que garantice la calidad; las pruebas son un elemento crítico para la garantía de calidad del software. Estas son actividades en las cuales un sistema o componente es ejecutado bajo condiciones o requerimientos especificados, los resultados son observados y registrados, además se realiza una evaluación del sistema o componente. Las pruebas y validación de los resultados se ejecutan en cada una de las etapas de desarrollo. Se realizan una y otra vez hasta que sean erradicados todos los errores que presenta el sistema y se pueda comprobar la eficiencia de las operaciones utilizadas para satisfacer las necesidades del cliente. (Pressman, 2005)

1.10.1. Pruebas de software

De acuerdo a la IEEE[4] el concepto de prueba se define como: “Una actividad en la cual un sistema o componente es ejecutado bajo condiciones especificas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente”. De aquí la importancia que tiene realizarle pruebas al software antes de ser entregado al usuario final, garantizando así que el mismo tenga la menor cantidad de errores y mejor calidad posible.

1.10.2. Objetivos de las prueba

Las pruebas de software, al contrario de lo que normalmente se considera, tienen como objetivos detectar errores no encontrados hasta el momento en la aplicación. Se puede hablar entonces del éxito de las pruebas siempre y cuando se hallen errores en el software. Con las pruebas se puede además observar hasta qué punto el software parece funcionar en concordancia con los requisitos descritos por el cliente; aunque no pueden asegurar la ausencia de defectos, sólo puede mostrar que existen. (Pressman, 2005)

Existen diferentes tipos de prueba:

- El enfoque funcional o de **caja negra**: que realiza pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar.
- El enfoque estructural o de **caja blanca**: que se basa en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.

[4] (Institute of Electrical and Electronics Engineers) en español **Instituto de Ingenieros Eléctricos y Electrónicos**

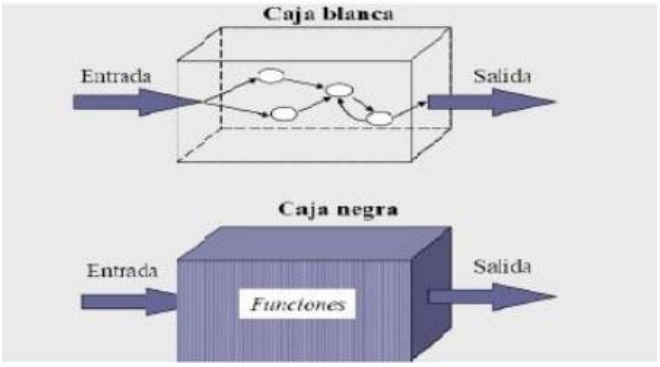


Figura 2: Representación de pruebas de Caja Blanca y Caja Negra

La figura representa la filosofía de las pruebas de caja blanca y caja negra. Como se puede observar para las pruebas de caja blanca se necesita conocer los detalles procedimentales del código, mientras que para las de caja negra únicamente se necesita saber el objetivo o funcionalidad que el código debe proporcionar.

1.10.3. Justificación de la selección de las pruebas a utilizar

Para la validación de la solución propuesta se decidió realizar pruebas de Caja Blanca con la utilización de la técnica de camino básico que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución.

Para su ejecución se utilizará la herramienta de prueba PHPUnit que es un marco de trabajo para pruebas de unidad en específico a PHP, ya que permite realizar prueba rápidamente, son fáciles de hacer y permite leer y analizar e independientes entre sí. Presenta métodos Assert los cuales realizan una prueba y retornan un "AssertionFailedError" si la prueba falló. Otros frameworks de pruebas estudiados resultaron ser de poca utilidad como el Zend test que es dependiente de la suite Zend Studio.

Para complementar estas pruebas se aplicarán además la ejecución de las pruebas de Caja Negra, ya que las mismas se centran en los requisitos funcionales que debe cumplir el software, asegurando de esa manera el correcto funcionamiento de la

aplicación mediante la realización de casos de pruebas, permitiendo que el sistema se ejecute en todas sus variantes. Se pueden introducir datos válidos o no válidos para el programa según lo que se desea: hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa se ejecute correctamente.

1.11. Conclusiones parciales

A lo largo de este capítulo se efectúo un estudio basado en la implementación de las listas de control de acceso en los diferentes marcos de trabajo así como las herramientas de desarrollo, metodologías y tecnologías a utilizar en la propuesta de solución, destacando sus características esenciales, teniendo como resultado:

- SCRUM: como metodología de desarrollo.
- Axure 5.5: como herramienta de prototipo web.
- ER/Studio: 7.1 como herramienta de modelado de datos.
- PostgreSQL 8.4: como servidor de base datos.
- Doctrine 1.2.1: como sistema mapeador relacional de objetos (ORM).
- Netbeans 7.0.1: como entorno de desarrollo.
- Java Script 1.8: como lenguaje del lado del cliente.
- PHP 5.2.5: como lenguaje del lado del servidor.
- ExtJS 2.2: para la creación de las interfaces.

También se define la estrategia a seguir para la comprobación de la calidad y funcionalidad de la propuesta de solución, en este caso:

- Pruebas de Caja Negra con la realización de los casos de pruebas.
- Pruebas de Caja Blanca mediante la técnica del camino básico.

Capítulo 2: Propuesta de solución

En el presente capítulo se hace una descripción de la solución propuesta de la presente investigación, basada en las diferentes modificaciones que garanticen mejoras en la seguridad del marco de trabajo SAUXE de acuerdo al SIT, así como la explicación de los principales artefactos de la metodología empleada. Se exponen además los patrones empleados durante el desarrollo así como la concepción de la arquitectura y algunos rasgos esenciales en el funcionamiento del marco de trabajo, con el objetivo de brindar una mayor comprensión de la propuesta de solución.

2.1. Estándares de nomenclatura y codificación utilizados

Los estándares de código resultan importantes en cualquier proyecto de desarrollo, más aún cuando muchos desarrolladores trabajan en el mismo proyecto. Los estándares de código ayudan a asegurar que el código tenga una alta calidad, menos errores y que pueda ser mantenido fácilmente. Estos incluyen pautas sobre la nomenclatura de las variables, clases y paquetes; como escribir estructuras de control e incluso las formas de poner los comentarios, en sí todo lo referente a la generación del código.

En el caso del proyecto Tribunales Populares Cubanos el equipo de arquitectura definió el estándar de codificación en sus inicios para ponerlo en práctica desde el inicio hasta el final de su desarrollo.

Nomenclatura General

- Se exceptúan el uso de las tildes y la letra ñ, la que será sustituida por nn.
- El nombre de todas las variables y métodos comenzarán con letra minúscula y si este está compuesto por varias palabras se utilizará el estilo de escritura lowerCamelCase, que dicta que para un nombre compuesto por varias palabras comenzará con minúscula pero todas las palabras internas que lo componen comienzan con mayúscula.

- En todo momento se utilizarán nombres que sean claros, concretos y libres de ambigüedades. Ejemplo: "asignarAcceso" y no solamente "asignar".

Identación

- En el contenido siempre se identará este con tabs, nunca utilizando espacios en blanco.

Clases

- El nombre de las clases controladoras estará en correspondencia con la funcionalidad que represente y seguido de la palabra Controller.
- El nombre de las clases comenzará con mayúscula y solo la letra inicial de la palabra Controller también con mayúscula. Ejemplo:

```
class TramiterolsistemaController extends ZendExt_Controller_Secure {

    }
```

Figura 3: Nombre de las clases

- Intentar mantener los nombres de las clases simples y descriptivas. Usar palabras completas, evitar acrónimos y abreviaturas (a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL o HTML).
- El número de métodos de una clase debe ser inferior a 40.

Rutinas y Métodos

- Es recomendable que los nombres de los métodos comiencen con un verbo, seguido del objeto al que afecta. Por ejemplo: cargarRol.
- Los nombres de los métodos de las clases controladoras incluirán luego del nombre de la funcionalidad la palabra “Action”. Ejemplo: function insertarPermisoAction().

Nombre de variables

- No se utilizarán nombres de variables que puedan ser ambiguos. Por ejemplo Col es un nombre ambiguo ya que puede ser columna o color.
- Se procurará evitar dar nombres sin sentido a variables temporales. Por ejemplo: temp, i, tmp.
- Las variables booleanas deben tener nombres que sugieran respuestas o contenidos, por ejemplo: trabaja, correcto, realizado.
- Los nombres de las variables booleanas deben ser positivos, por ejemplo: encontrado en lugar de noEncontrado.
- Se introducirán variables booleanas temporales cuando en una estructura de control (if, case, while) la expresión sea excesivamente compleja.

Cabecera del archivo

- Es importante que todos los archivos .php inicien con una cabecera específica que indique información de la versión y autor de los últimos cambios. Es de cada equipo decidir si se quieren o no agregar más datos. Ejemplo:

```
/*** Componente para gestionar el acceso a los expedientes
 *@versión: 1.0-5
 *@modificado: 1  de junio del 2012
 * @author Reinier Fernández Coello
 */
```

Figura 4: Cabecera de los archivos

Comentarios en las funciones

- Todas las funciones deben tener un comentario antes de su declaración, explicando que hacen. Ningún programador debería tener que analizar el código de una función para conocer su utilidad. Tanto el nombre como el comentario que acompañe a la función deben bastar para ello.

2.2. Arquitectura del Sistema de Informatización de Tribunales

La arquitectura de software es el conjunto de técnicas metodológicas desarrolladas con el fin de facilitar la programación. Se refiere a un grupo de abstracciones y patrones que nos brindan un esquema de referencia útil para guiarse en el desarrollo de software dentro de un sistema informático. Establece todos los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema. (Fernández Lanvin, 2009)

Como se menciona en el capítulo anterior, para el desarrollo del SIT se utiliza el marco de trabajo Sauxe el cual responde a un estilo arquitectural en capas, basándose en una distribución jerárquica de las responsabilidades para proporcionar una división efectiva de los problemas a resolver, de forma tal que cada capa contiene la funcionalidad relacionada solo con las tareas de esa capa, las capas inferiores no tienen dependencias de las capas superiores y la comunicación entre ellas está basada en una abstracción que les proporciona un bajo acoplamiento. Esta arquitectura está compuesta básicamente por cinco capas:

- **Capa de Presentación:** Es la capa superior, contiene los componentes con los que va a interactuar el usuario. Desde esta capa se pueden alcanzar las funcionalidades que brinda el negocio y mostrar o capturar la información a través de los diferentes elementos que comprende. En esta capa se emplea las facilidades que brinda el marco de trabajo ExtJS para la construcción de interfaces amigables a la vista de los usuarios. (Baryolo, 2008)
- **Capa de Control o Negocio:** Su diseño depende directamente del negocio específico al que se refiera cada subsistema. Esta capa recibe una petición del nivel superior, gestiona o procesa la misma, de ser necesario solicitando la capa de Acceso a Datos y finalmente envía una respuesta continuando el proceso en el punto donde se inició dicha petición. (Baryolo, 2008)
- **Capa de Acceso a Datos:** En esta capa estará presente el ORM (Object Relational Mapping) Doctrine, como marco de trabajo para la persistencia y la

comunicación con el servidor de datos mediante el protocolo PDO, también estará un Persistidor de Configuración que es el encargado de comunicarse vía XML con los Ficheros de Configuración del sistema denominado FastResponse. (Baryolo, 2008)

- **Capa de Datos:** En esta capa estará ubicado como servidor de base de datos PostgreSQL y un conjunto de Ficheros de Configuración de la arquitectura tecnológica. (Baryolo, 2008)
- **Capa de Servicio:** En esta última capa se encuentran todos los subsistemas que prestan y consumen servicios entre sí. (Baryolo, 2008)

2.3. Patrones utilizados por el marco de trabajo Sauxe

2.3.1. Modelo Vista Controlador

La estructura que define el marco de trabajo Sauxe para la separación de la vista del modelo deja bien estructurado y delimitado donde quedan las clases del modelo, las de la vista y las controladoras. Las clases que se encuentran en el paquete "TPC/apps/seguridad/model", van a ser las que responden al modelo, como se muestra en la figura 5, que son las que representan la información con la que trabaja la aplicación, es decir, la lógica del negocio. Dentro del paquete "bussines" (negocio) se encuentran las clases "Model" que son las encargadas de insertar, modificar o eliminar objetos de las clases entidades del modelo de datos. Dentro del paquete "domain" (dominio) se encuentran las clases del dominio que heredan de sus respectivas clases Base[5] que se encuentran en el paquete "generated" que también pertenece al paquete "domain".

[5] Son las clases que genera el ORM Doctrine, que contienen los mismos atributos y relaciones que existen en el modelo de datos.

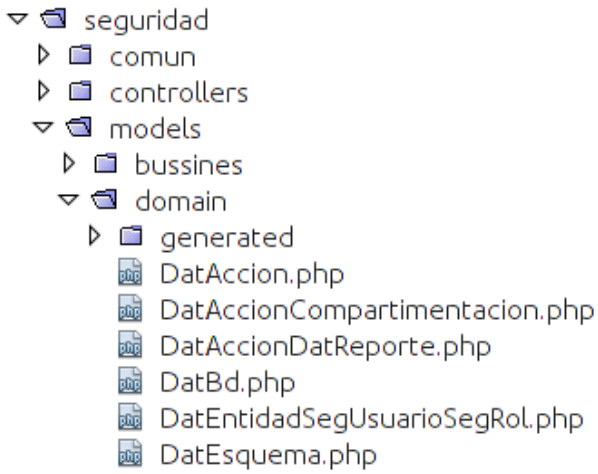


Figura 5: Capa del Modelo

Las clases que se encuentran dentro del paquete "TPC/apps/seguridad/views", en conjunto con las que importan que se encuentran en "TPC/web/seguridad/views" son las que responden a la capa de la vista como se aprecia en las figuras 6 y 7 respectivamente, que son las que transforman el modelo en una página web que le permite al usuario interactuar con ella, para esto se apoya en la utilización de la librería ExtJS, que facilita construcción de interfaces con un buen acabado y agradables a la vista del usuario.

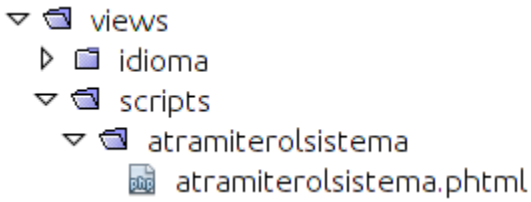


Figura 6: Capa de la Vista en el paquete apps.

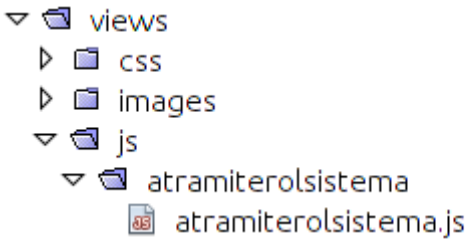


Figura 7: Capa de la Vista en el paquete web.

Las controladoras son las que se encuentran dentro del paquete "TPC/apps/seguridad/controllers", estas son las encargadas de procesar las interacciones del usuario, recibir peticiones de la vista y gestionar y procesar la vista, además si es necesario solicita acciones del modelo y realizan cambios apropiados en la capa del modelo o la vista según sea el caso.

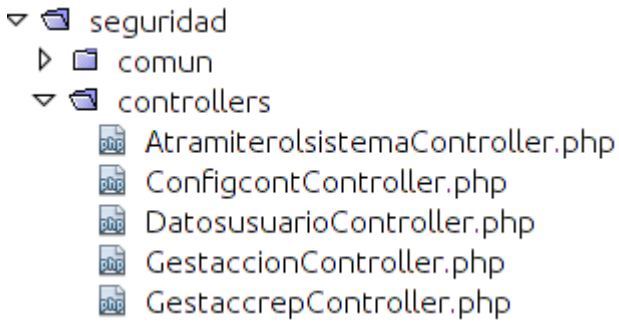


Figura 8: Capa del Controlador

2.3.2. Patrón Bajo Acoplamiento

Las clases que implementan la lógica de negocio y de acceso a datos se encuentran en el Modelo, estas clases no tienen asociaciones con las de la Vista o el Controlador por lo que la dependencia en este caso es baja, demostrándose de esta manera el uso de este patrón como se muestra en la figura 9.

```
<?php
class SegRol extends BaseSegRol
{
    public function setUp(){...}
    static public function cantrol($filtroDominio = null) {...}
    static public function canttotalrol() {...}
    static public function obtenerrol($filtroDominio,$limit, $start){...}
    static public function obtenerCantRolesUsuario($idusuario){...}
    static public function obtenerrolesasociados($limit, $start, $idusuario, $arrayRolesDominio)
    static public function cantrolesDominio($arrayRolesDominio) {...}
    static public function obtenerRolesBuscado($rolbuscado,$limit,$start,$idusuario) {...}
    static public function cargarcomborol(){...}
    static public function obtenerrolesusuario($idusuario){...}
    static public function obtenerrolesusuarioentidad($idusuario,$identidad){...}
    static public function comprobarrol($denominacion,$abreviatura){...}
    static public function obtenerrolBuscado($filtroDominio, $denominacion, $limit, $start){...}
    static public function cantrolBuscados($filtroDominio, $denominacion){...}
    static public function eliminarRoles($idrol){...}
}
```

Figura 9: Clase del modelo que solo depende de su clase Base.

2.3.3. Patrón Inversión de Control (IoC)

La Inversión de Control es un patrón de diseño pensado para permitir un menor acoplamiento entre componentes de una aplicación y fomentar así la reutilización de los mismos. Este patrón es utilizado por el marco de trabajo para crear y brindar servicios. Cada módulo del proyecto cuenta con un esquema propio en la base de datos y cuando un módulo necesita acceder al esquema de otro, lo que se establece es brindar un servicio para que aquel módulo que lo necesite lo consuma.

El marco de trabajo a través del patrón IoC, define la creación de clases, para de ellas brindar los servicios necesarios. En la siguiente figura se muestra la clase SegUsuarioService ubicada en el paquete apps/seguridad/services donde se crean las funciones que ofrecerá como servicio. (Figura 9)

```
<?php

class SegUsuarioService extends ZendExt_Model {

    var $usuario;
    var $dat_entidad;//para ver a q estructuras tiene permiso un usuario
    function __construct(){...}
    function UsuariosAG(){...}
    function BuscarNombreUsuario($idpersona){...}
    function DevolverTribunalesRolDadoIdusuario($idusuario) {...}
    function VerSiRolTienePermisoAEsaEntidad($idUsuario, $padre) {...}
    function BuscaTribunalesPuedeAcceder($idUsuario) {...}
    function Busca_Padre($id){...}
    function BuscaRol(){...}
    function UsuarioActivo($idusuario){...}
    function PermisoUsuarioFuncionalidad($idusuario, $peticion) {...}

}
?>
```

Figura 10: Representación de la clase SegUsuarioService

Cada módulo cuenta con un fichero ioc-general.xml en el cual se brindan los servicios deseados, para que otros lo consuman. La siguiente figura muestra el fichero ioc-general.xml que se encuentra en el paquete apps/seguridad/común/recursos/xml.

```
<seguridad src="seguridad">
    <UsuariosAG>
    <UsuarioActivo>
    <DevolverTribunalesRolDadoIdusuario>
<VerificarFuncionalidadUsuario reference="">
        <inyector clase="SegUsuarioService" metodo="PermisoUsuarioFuncionalidad" />
        <prototipo>
            <parametro nombre="idusuario" tipo="integer" />
            <parametro nombre="peticion" tipo="string"/>
           <resultado tipo="integer" />
        </prototipo>
</VerificarFuncionalidadUsuario>
<BuscarNombreUsuario>
</seguridad>
```

Figura 11: Fichero ioc-general.xml

La siguiente figura muestra la función insertarPermisoAction en la clase TramitrerolsistemaController.php del módulo de Seguridad, la cual consume el servicio insertarTRS brindado por cada uno de los módulos según sea el caso.

```
function insertarPermisoAction(){
        $id_rol = $this->_request->getPost('id_rol');
        $idnactoprocesal = $this->_request->getPost('idnactoprocesal_seleccionado');
        $idsistema = $this->_request->getPost('idsistema');
        $integrator = ZendExt_IoC::getInstance();
        if($idsistema == 30003)
        {
            $ntramites = $integrator->penal->listarNtramite($idnactoprocesal);
            $insertar = $integrator->penal->insertarTRS($id_rol,$ntramites);
            die();
        }
        if($idsistema == 30006)
        {
            $ntramites = $integrator->economico->listarNtramite($idnactoprocesal);
            $insertar = $integrator->economico->insertarTRS($id_rol,$ntramites);
            die();
        }
        if($idsistema == 30014)
        {
            $ntramites = $integrator->laboral->listarNtramite($idnactoprocesal);
            $insertar = $integrator->laboral->insertarTRS($id_rol,$ntramites);
            die();
        }

        return $insertar;
    }
```

Figura 12: Procedimiento insertarPermisoAction()

2.3.4. Patrón Controlador

El patrón Controlador es un patrón que sirve de intermediario entre una interfaz específica y el algoritmo que la implementa, de tal manera que es quien recibe los

datos del usuario y los envía a las distintas clases según el método ejecutado. Este patrón propone que la lógica de negocios debe estar separada de la capa de presentación, con el objetivo de aumentar la reutilización de código y a la vez tener un mayor control. En la estructura que propone Sauxe este patrón se evidencia dentro de la carpeta "controllers" que se encuentra en "TPC/app/Nombre_Módulo/controllers", dentro de esta carpeta se encuentran todas las clases controladoras de cada uno de los módulos y dichas clases son las responsables de implementar todas las funcionalidades pertenecientes a una interfaz de un componente determinado, además son las que reciben los datos del usuario y los utilizan de acuerdo a la acción solicitada. A continuación se muestra clase TramitrerolsistemaController.php del módulo de seguridad que es la encargada de implementar las funcionalidades del componente Configurar acceso a los expedientes.

```
class TramiterolsistemaController extends ZendExt_Controller_Secure {

    function init() {
        parent::init();
    }

    function tramiterolsistemaAction() {
        $this->render();
    }

function cargarSistemaAction() {...}
function cargarActoprocesalesAction() {...}
function cargarRolAction(){...}
function verificarPermisoAction(){...}
function insertarPermisoAction(){...}
function eliminarPermisoAction(){...}

}
```

Figura 13: Clase TramiterolsistemaController.php

2.3.5. Patrón Singleton

El patrón de diseño singleton (que significa instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su propósito consiste en garantizar que una clase sólo tenga una instancia y proporcionar un único punto de acceso global a ella. Dicho patrón provee una única instancia global debido a que si se solicita una instancia de la clase:

- La propia clase es la responsable de crear la única instancia, la cual debe garantizar que no se pueda crear ninguna otra (interceptando todas las peticiones para crear nuevos objetos) y proporcionando un solo punto de acceso a ella.
- Si no se ha creado anteriormente (o sea, es la primera vez que se utiliza esta clase) se crea la instancia.
- Si existe ya anteriormente una instancia (es la segunda o más veces que se utiliza), se retorna la existente todas las veces que se solicite.
- El constructor de la clase debe declararse como privado para que la clase no pueda ser instanciada directamente.
- Además en PHP5 no se permite que la clase sea clonada con el método __clone().

En la figura 13 se muestra un ejemplo de la implementación del patrón Singleton que se emplea en la clase ZendExt_Event:

```
class ZendExt_Event {
        /**...
        private $_xml;

        /**...
        private $_ioc;

        /**...
        private static $_instance;

        /**...
        private function __construct() {
                $this->_xml = ZendExt_FastResponse::getXML('events');
                $this->_ioc = ZendExt_IoC::getInstance();
        }

        /**
         * Retorna una instancia de ZendExt_Event para el Singleton
         * @return ZendExt_Event
         */
        static function getInstance () {
                if (self :: $_instance == null)
                        self :: $_instance = new self ();

                return self :: $_instance;
        }
```

Figura 14: Implementación del patrón Singleton

2.4. Descripción de la solución

Para el cumplimiento del objetivo propuesto al inicio de este trabajo las vías de solución están basadas las siguientes actividades:

- Control de las peticiones que realiza una funcionalidad y los permisos que tiene un rol sobre esa funcionalidad.
- Configuración de las listas de control de acceso en el marco de trabajo Sauxe para el acceso a los expedientes según su estado.

Para darle solución a estas dos actividades se hace necesario el estudio del funcionamiento del marco de trabajo Sauxe para logar el cumplimiento de las actividades descritas anteriormente, para ello se relaciona a continuación los artefactos de la metodología empleada así como las modificaciones realizadas en el módulo de seguridad de dicho marco de trabajo.

2.4.1. Product Backlog / Pila de Tareas del Producto

Las actividades a realizar para darle cumplimiento al objetivo de este trabajo se enumeran en la Pila de Tareas del Producto (PTP). El administrador del producto es responsable del contenido, priorización, y disponibilidad de este: nunca se acaba y es usado en la planificación del proyecto, es simplemente una estimación inicial de los requisitos. La PTP se desarrolla paralelamente a medida que el producto y el ambiente en el cual se trabaja evoluciona, es dinámico; maneja constantemente los cambios para identificar que necesita el producto para ser: apropiado, competitivo y útil. Mientras exista un producto, la PTP también existe.

Pila de Tareas del Producto

Número	Nombre	Descripción
1	Filtrar peticiones.	El sistema debe ser capaz de filtrar las peticiones que realiza cada funcionalidad

		verificando el rol que esta autenticado y las funcionalidades a las que ese rol tiene acceso.
2	Configurar acceso a los expedientes.	El sistema debe asignarle a un rol acceso a los diferentes expedientes, de acuerdo al estado del mismo. Una vez que se haya asignado debe darle la opción al usuario de modificar dicha operación.
3	Elaborar informe de modificaciones al módulo de seguridad.	Este informe recogerá todas las modificaciones desarrolladas en el módulo de seguridad del marco de trabajo Sauxe, logrando así obtener una información detalla de la personalización realizada.

Tabla 1: Pila de Tareas del Producto

2.4.2. Sprint Backlog / Pila de Tareas del Sprint

La Pila de Tareas del Sprint (PTS) define el trabajo, o las tareas, que el equipo desarrollará para poder convertir la tarea seleccionada en la PTP para ese Sprint en un incremento potencialmente funcional del producto. El equipo crea una lista inicial de estas tareas en la reunión de planeación del Sprint, las cuales deben ser divididas de modo que cada una demore entre 4 a 16 días finalizarlas.

A continuación se muestran las planificaciones de cada uno de los Sprint creados para la solución planteada conformada por:

- **Tarea**: se refiere a las tareas que se deben realizar.
- **Estado**: expresa el porciento realizado de dicha tarea en el momento que se realiza la reunión del Sprint.
- **Estimado**: se refiere a la cantidad de días que se ha estimado para realizar la tarea.
- **Real**: refleja los días reales que se tomó realizar la tarea.

Sprint I: Filtrar peticiones

Tarea	Estado	Estimado	Real
Realizar un estudio sobre la integración de los diferentes componentes del marco de trabajo SAUXE.	100%	12	15
Realizar las consultas necesarias en la base de datos de tal forma que reciban por parámetro el identificador de la petición y del rol que está aspirando acceder a una funcionalidad del sistema.	100%	14	14
Realizar las modificaciones en el componente Zend_Ext de tal forma que hagan uso de las consultas realizadas y que finalmente realice el filtro de las peticiones.	100%	11	13

Tabla 2: Pila del Sprint I

Anteriormente el marco de trabajo SAUXE solo verificaba el acceso a las diferentes funcionalidades de forma visual, por lo que si un usuario realizaba la petición HTTP de una funcionalidad a la que no tiene acceso el sistema atendía dicha petición, por lo que se hizo necesario realizar las modificaciones antes mencionadas en la Tabla del Sprint I para lograr una mayor seguridad en dicho marco de trabajo. Para ver las clases modificadas en este Sprint consultar el **Anexo 1**.

Con la terminación del Sprint I se tiene logrado el 35% de la solución total puesto que sus funciones son vitales para la seguridad del sistema.

Sprint II: Configurar acceso a los expedientes

Inicialmente para ello se necesita tener almacenado en base de datos que rol tendrá acceso a realizar un trámite[6], lo cual está muy vinculado al acto procesal[7] destino de dicho trámite. A continuación se muestra la tabla del Sprint II para darle solución a esta tarea:

Tarea	Estado	Estimado	Real
Realizar el análisis del control de acceso a los	100%	14	14

[6] Cada uno de los pasos que se realizan de manera sucesiva para solucionar un asunto.
[7] La actividad encaminada a lograr la finalidad que se propone el asunto.

expedientes de acuerdo al trámite y el estado en el que se encuentran dichos expedientes.			
Realizar el prototipo de interfaz del control de acceso.	100%	6	7
Realizar las modificaciones en el modelo de datos de cada unos de los módulos del SIT.	100%	14	15
Implementar la asignación de rol a los diferentes trámites por los cuales puede pasar un expediente en el módulo Penal.	100%	5	6
Implementar la modificación de la asignación de rol a los diferentes trámites por los cuales puede pasar un expediente en el módulo Penal.	100%	4	4
Implementar la asignación de rol a los diferentes trámites por los cuales puede pasar un expediente en el módulo Económico.	100%	4	5
Implementar la modificación de la asignación de rol a los diferentes trámites por los cuales puede pasar un expediente en el módulo Económico.	100%	4	4
Implementar la asignación de rol a los diferentes trámites por los cuales puede pasar un expediente en el módulo Laboral.	100%	4	4
Implementar la modificación de la asignación de rol a los diferentes trámites por los cuales puede pasar un expediente en el módulo Laboral.	100%	5	5
Implementar la asignación de rol a los diferentes trámites por los cuales puede pasar un expediente en el módulo Civil.	100%	4	4
Implementar la modificación de la asignación de rol a los diferentes trámites por los cuales puede pasar un expediente en el módulo Civil.	100%	4	4

Implementar la asignación de rol a los diferentes trámites por los cuales puede pasar un expediente en el módulo Administrativo.	100%	4	4
Implementar la modificación de la asignación de rol a los diferentes trámites por los cuales puede pasar un expediente en el módulo Administrativo.	100%	4	4

Tabla 3: Pila del Sprint II

Para poder almacenar la lista de los trámites a los que un rol tiene acceso se hizo necesario la creación de una tabla en el modelo de datos de los módulos antes mencionados que constaría con tres atributos:

- El identificador de la tabla
- El identificador del trámite
- El identificador del rol que tiene acceso a realizar ese trámite.

Esta tabla tendrá una relación con la tabla nTramite del módulo propio y con la tabla seg_rol del módulo de seguridad del SIT. La siguiente imagen pertenece al módulo Penal donde se muestra como quedaría finalmente dicha tabla.

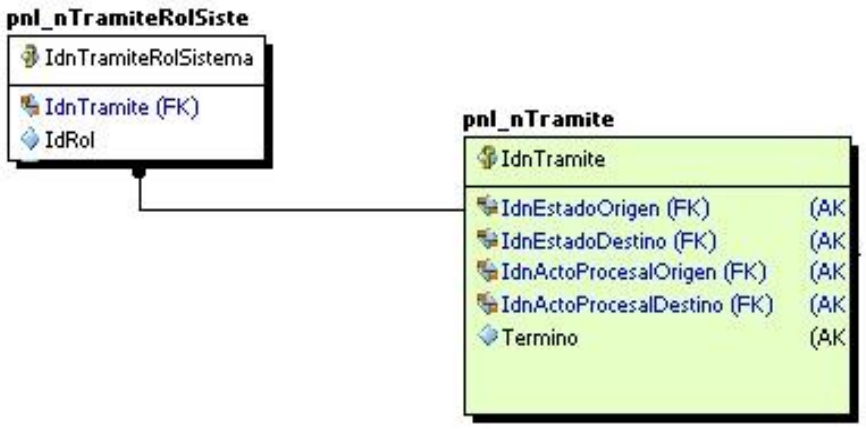


Figura 15: Tabla nTramiteRolSistema

Anteriormente en el SIT no poseía un control de los roles que podían acceder a un determinado expediente, es decir, todos los roles del sistema accedían a todos los expedientes sin tener en cuenta el estado que tenía un expediente determinado. Debido a que un rol solo puede realizar un acto procesal y por ende un trámite solo puede ser realizado por un rol, se realizó el Sprint II dejando completamente funcional la lista de control de acceso en el SIT. Con este se puede lograr recoger toda la información necesaria para tener una mayor seguridad en dicho sistema. Para ver el componente como resultado de este Sprint consultar el **Anexo 1**.

Con la culminación del presente Sprint se tiene logrado el 95 % de la solución.

Sprint II: Elaborar informe de modificaciones al módulo de seguridad

Tarea	Estado	Estimado	Real
Elaborar el documento “Descripción de las modificaciones realizadas en el módulo de seguridad del marco de trabajo Sauxe para el SIT”.	100%	4	3

Tabla 4: Pila del Sprint III

Este documento contiene todas las modificaciones realizadas en las diferentes clases en el módulo de seguridad, logrando la personalización de dicho módulo del marco de trabajo Sauxe para el SIT. Para ver este documento consultar el **Anexo 1**

Con la culminación del presente Sprint se garantiza el cumplimiento del 100% de la propuesta de solución.

2.5. Conclusiones parciales

La elaboración del documento “Descripción de las modificaciones realizadas en el módulo de seguridad del marco de trabajo Sauxe para el SIT” permite tener un mayor conocimiento basado en la personalización desarrollada durante la presente investigación en las diferentes clases del módulo de seguridad.

La descripción de la propuesta de solución y las modificaciones realizadas al marco de trabajo Sauxe de acuerdo al SIT, teniendo en cuenta sus componentes y la interacción

del mismo, es de vital importancia para la seguridad del sistema ya que posibilitan tener un mayor control sobre el acceso de un determinado rol en dicho sistema. En particular:

- El control de las peticiones que realiza una funcionalidad una vez realizado permite filtrar las peticiones de cada una de las funcionalidades así como los permisos que tiene un rol sobre esa funcionalidad que anteriormente solo existía de forma visual.

- La configuración de las listas de control de acceso en el marco de trabajo Sauxe para el acceso a los expedientes según su estado permite asignarle un rol a cada acto procesal y a su vez a cada trámite.

Capítulo 3: Validación de la solución

En el presente capítulo se realiza la validación a la solución propuesta, a través de pruebas de software que respondan al correcto funcionamiento y totalidad de las funcionalidades definidas por el cliente, a través de las pruebas de caja blanca y pruebas de caja negra. Además se hace una valoración de las mismas según los resultados obtenidos.

3.1. Validación de la propuesta de solución

Las **pruebas de caja blanca** permiten examinar la estructura interna del programa realizando un seguimiento del código fuente según va ejecutando los casos de prueba, de manera que se determinan concretamente las instrucciones o bloques en los que existen errores. Para la realización de las pruebas de unidad se diseñan casos de prueba que se encargan de examinar la lógica del sistema. Los casos de prueba garantizan que se ejecuten todos los caminos independientes de cada módulo o clase y todas las decisiones lógicas así como todos los bucles y las estructuras de datos internas. (Rojas, et al., 2007)

Las **pruebas de caja negra** comprueban que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta. Permiten detectar funcionamiento incorrecto o incompleto, errores de interfaz, errores de acceso a estructuras de datos externas, problemas de rendimiento y/o errores de inicio y terminación. (Rojas, et al., 2007)

3.2. Pruebas de Caja Negra

Estas pruebas son llevadas a cabo sobre la interfaz del software, actuando sobre ella como una caja negra, proporcionando entradas y estudiando las salidas para ver si son o no las esperadas. Conociendo la función para la que fue diseñado, se hacen pruebas que demuestren que cada función es operativa y al mismo tiempo se buscan errores en cada una. Las llamadas pruebas de caja negra se basan en la especificación del

programa o componente a ser probado, para elaborar los casos de prueba. También son conocidas como pruebas de comportamiento o pruebas inducidas por los datos. Permite obtener un conjunto de condiciones de entrada que ejecutan todos los requisitos funcionales de un programa.

Las pruebas de caja negra no son alternativas a las técnicas de prueba de caja blanca. Es un enfoque complementario. Intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes; errores de interfaz, en estructuras de datos o en acceso a bases de datos externas; errores de rendimiento, de inicialización y de terminación.

A continuación se muestran resultados obtenidos durante la realización de las pruebas de caja negra. Estas arrojaron resultados satisfactorios a los efectos de la implementación de la propuesta de solución.

Caso de prueba de Caja Negra "Configurar acceso a los expedientes"

Descripción General

La funcionalidad comienza cuando el Administrador del sistema desea configurar el acceso a los expedientes, mediante la selección de una de las materias registradas en la aplicación, luego selecciona el acto procesal al cual le asignará o modificará el rol previamente asignado introduciendo los datos correspondientes. La funcionalidad termina cuando el sistema guarda definitivamente la operación correspondiente.

Condiciones de Ejecución

El Administrador del sistema debe estar autenticado correctamente en el sistema.

Secciones a probar en el Caso de Prueba

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad

SC 1: Asignar Rol	EC 1.1: Asignar rol satisfactoriamente.	Asigna los datos del acto procesal y del rol seleccionado exitosamente.
	EC 1.2:Insertar campos vacíos.	Muestra un error al insertarse datos vacíos.
	EC 1.3: Operación cancelada.	Cancela la Operación y vuelve a la interfaz previa.
SC 2: Modificar Rol	EC 2.1: Modificar rol satisfactoriamente.	Se le muestra el rol asignado previamente, modifica los datos del acto procesal y del rol seleccionado.
	EC 2.3: Operación cancelada.	Cancela la Operación y vuelve a la interfaz previa.

Tabla 5: Casos de pruebas de Caja Negra

Descripción de variable

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Rol	Campo de selección	No	Introducir el rol asignado al acto .

Tabla 6: Descripción de las variables

Sección 1 “Asignar Rol”

Escenario	Descripción	Motivo	Fecha	Respuesta del Sistema	Flujo Central
EC 1.1: Asignar rol satisfactoriamente.	Asigna los datos del acto procesal y del rol seleccionado	V Extemporánea	V 12/2/2012	Debe guardar los datos correctamente. El sistema	1-Materia 2-Administración y Gobierno 3-Configurar

	exitosamente.			muestra un mensaje indicando que se ha guardado exitosamente la operación y vuelve a la interfaz previa.	acceso a los expedientes
EC 1.2: Insertar campos vacíos.	Al insertar datos pertenecientes al acto procesal, no ha seleccionado el rol al que le asignara permiso.	I	V 12/2/2012	Señala que no se seleccionó el rol y muestra un mensaje indicando que debe seleccionar el rol. El sistema mantiene la ventana abierta.	1-Materia 2-Administración y Gobierno 3-Configurar acceso a los expedientes
EC 1.3: Operación cancelada.	Se termina la operación.	NA	NA	Debe cancelarse la operación exitosamente.	1-Materia 2-Administración y Gobierno 3-Configurar acceso a los expedientes

Tabla 7: Sección 1 ¨Asignar Rol¨

Sección 2 “Modificar Rol”

Escenario	Descripción	Motivo	Fecha	Respuesta del Sistema	Flujo Central
EC 2.1: Modificar rol satisfactoriamente.	Se le muestra el rol asignado previamente, modifica los datos del acto procesal y del rol seleccionado exitosamente.	V Extemporánea		Debe guardar los datos correctamente. El sistema muestra un mensaje indicando que se ha modificado exitosamente la operación y vuelve a la interfaz previa.	1-Materia 2-Administración y Gobierno 3-Configurar acceso a los expedientes
EC 2.2: Operación cancelada.	Se termina la operación.	NA	NA	Debe cancelarse la operación exitosamente.	1-Materia 2-Administración y Gobierno 3-Configurar acceso a los expedientes

Tabla 8: Sección 2 ¨Modificar rol¨

Resultados de las pruebas de caja negra

Para la realización de las pruebas de caja negra a la solución fueron analizadas todas las actividades referentes a la configuración de acceso a los expedientes. Se hizo una descripción de cada escenario de prueba. Se realizaron un total de tres iteraciones para poder alcanzar los resultados satisfactorios desde el punto de vista funcional del componente, atendiendo al correcto comportamiento del mismo ante diferentes situaciones (entradas válidas y no válidas).

A continuación se realiza una descripción de las no conformidades detectadas durante la ejecución de las pruebas de caja negra. En las mismas se encontraron mayormente errores de faltas ortográficas, de validaciones, y errores en las interfaces.

En la primera iteración se encontraron un total de 17 no conformidades (NC):

- Faltas ortográficas 3 NC
- Errores en las interfaces 12 NC
- Errores de validación 2 NC

En la segunda iteración se encontraron un total de 8 no conformidades (NC):

- Faltas ortográficas 0 NC
- Errores en las interfaces 7 NC
- Errores de validación 1 NC

En la tercera iteración no se encontraron no conformidades (NC):

- Faltas ortográficas 0 NC
- Errores en las interfaces 0 NC
- Errores de validación 0 NC

La gráfica que se muestra a continuación representa el resultado de estas pruebas según las NC detectadas durante su ejecución:

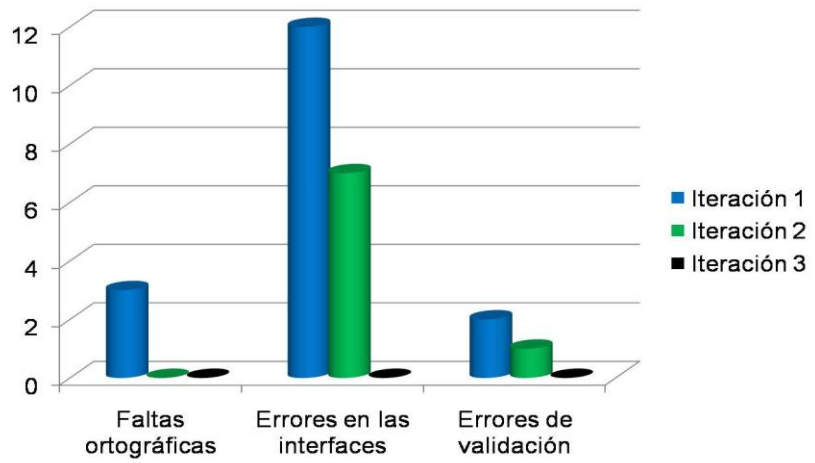


Figura 16: Resultados de las iteraciones

Luego de ser corregidos los errores encontrados en las pruebas, se pudo comprobar que el flujo de trabajo de las interfaces estaba correcto, ya que cumplía con las condiciones necesarias que se habían planteado.

3.3. Pruebas de Caja Blanca

La prueba de la caja blanca del software se encarga de comprobar los caminos lógicos del software proponiendo casos de prueba para que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado. (D'Onofrio, 2006)

Como se señaló en el capítulo 1 de la presente investigación, para llevar a cabo las pruebas de caja blanca se utilizará la técnica de camino básico. A continuación se muestran los elementos utilizados alrededor de dicha técnica:

- **Grafo de flujo:** representa el flujo de control lógico de un programa y se utiliza para trazar más fácilmente los caminos de éste. Cada nodo representa una o más sentencias procedimentales y cada arista representa el flujo de control. (Caja Blanca, 2008)

- **Complejidad ciclomática:** es una métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto de las pruebas, el cálculo de la complejidad ciclomática representa el número de caminos independientes del conjunto básico de un programa. (Caja Blanca, 2008)
- **Camino independiente:** cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una nueva condición. (Caja Blanca, 2008)

Esta prueba permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto de caminos de ejecución. Además también garantiza que durante la realización de los casos de prueba obtenidos a través del camino básico se ejecute cada sentencia del programa por lo menos una vez.

Para aplicar la técnica del camino básico se debe introducir la notación para la representación del flujo de control, este puede representarse por un Grafo de Flujo en el cual:

- Cada nodo del grafo corresponde a una o más sentencias de código fuente.
- Todo segmento de código de cualquier programa se puede traducir a un Grafo de Flujo.
- Se calcula la complejidad ciclomática del grafo.

Para construir el grafo se debe tener en cuenta la notación para las instrucciones:

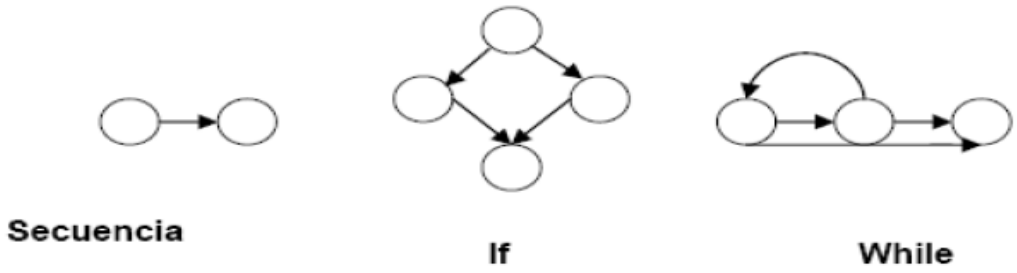


Figura 17: Notación de grafos de flujo (Secuencia, If, While)

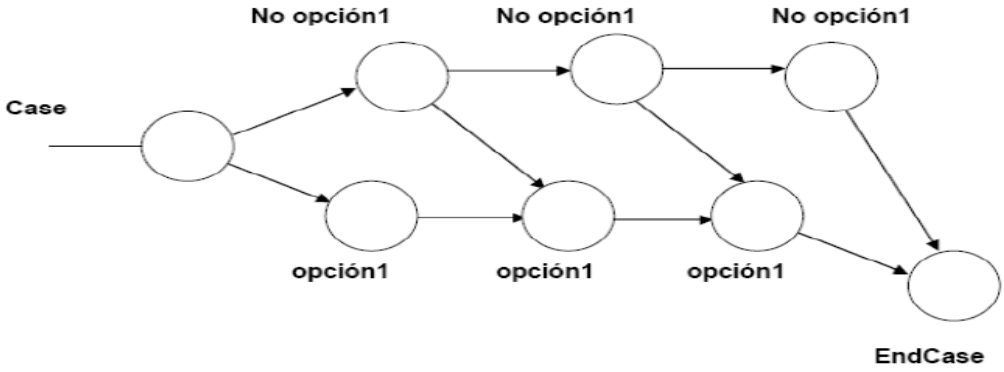


Figura 18: Notación de grafos de flujo (Case)

Un grafo de flujo está formado por 3 componentes fundamentales que ayudan a su elaboración y comprensión, estos brindan información para confirmar que el trabajo se está haciendo adecuadamente.

Componentes del grafo de flujo:

- **Nodo:** son los círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.

- **Aristas:** son constituidas por las flechas del grafo, son iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aún cuando el nodo no representa la sentencia de un procedimiento.
- **Regiones:** son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo, como una región más. Las regiones se enumeran siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.

Para realizar la técnica del camino básico, es necesario calcular la complejidad ciclomática del algoritmo o fragmento de código a analizar, pues esta proporciona una medición cuantitativa de la complejidad lógica de un programa. A continuación se enumeran las sentencias de código del procedimiento realizado al método insertarTramiteRolSistema ($id_rol, $idntramite) el cual tiene como función insertar los permisos según el sistema seleccionado por el usuario.

```
static function insertarTramiteRolSistema($id_rol,$idntramite)
{
    for ($i = 0; $i < count($idntramite); $i++) (1)
    {
    $id_ntramite = $idntramite[$i]['idntramite']; (2)
    $pnl_ntramiterolsistema = new PnlNtramiterolsistema; (3)
    $pnl_ntramiterolsistema->idrol = $id_rol; (4)
    $pnl_ntramiterolsistema->idntramite = $id_ntramite; (5)
    $consulta = PnlNtramiterolsistemaModel::insertarpnl_ntramiterolsistema($pnl_ntramiterolsistema); (6)
    }
    return $consulta; (7)

}
```

Figura 19: Procedimiento insertarTramiteRolSistema

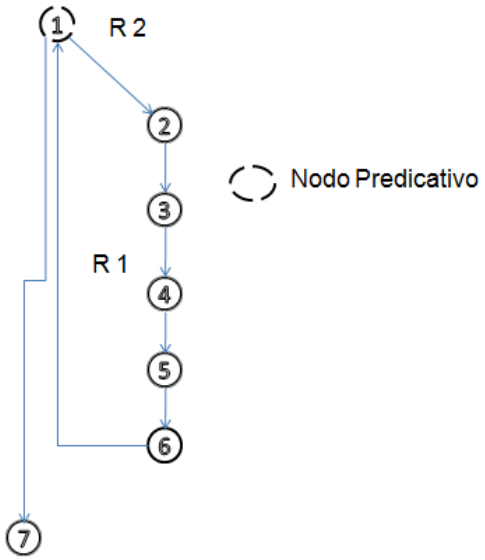


Figura 20: Grafo de flujo asociado al algoritmo

Cálculo de la complejidad ciclomática a partir de un segmento de código

El valor calculado como complejidad ciclomática define el número de caminos independientes, del conjunto básico de un programa y define un límite superior para el número de pruebas que se deben realizar, para asegurar que se ejecute cada sentencia al menos una vez. Para efectuar el cálculo de la complejidad ciclomática del código es necesario tener varios parámetros como son, la cantidad total de aristas del grafo (**A**), la cantidad total de nodos (**N**), la cantidad de nodos predicativos (**P**): se refiere a los nodos de los cuales parten más de una arista, y por último la cantidad de regiones (**R**), que se emplean en las siguientes fórmulas:

V (G) = (A - N) + 2	**V (G) = P + 1**	**V (G) = R**
V (G) = (7 - 7) + 2	V (G) = 1 + 1	V G) = 2
V (G) = 2	V (G) = 2	

Para cada formula “V (G)” representa el valor del cálculo.

El resultado del cálculo de la complejidad ciclomática del código fue 2, lo que significa que existen dos posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.
Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo.

Número	Camino
1	1-2-3-4-5-6-1-7
2	1-7

Tabla 9: Caminos básicos

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico. Para realizarlos es necesario cumplir con las siguientes exigencias:

- **Descripción:** Se hace la entrada de datos necesarios, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.
- **Condición de ejecución:** Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
- **Entrada:** Se muestran los parámetros que entran al procedimiento.
- **Resultados Esperados:** Se expone resultado que se espera que devuelva el procedimiento.

Casos de Pruebas

Número	Descripción	Condición de ejecución	Entrada	Resultado esperado

1	El parámetro $idntramite no está vacío, $id_rol contiene el identificador del rol seleccionado y $idntramite los datos de los tramites donde está incluido el acto procesal previamente seleccionado.	El acto procesal por el usuario posee una cierta cantidad de trámites asociados a él.	$id_rol = 90000023 $idntramite [] = 7	Muestra un mensaje informando al usuario que se ha insertado correctamente la configuración.
2	El acto procesal seleccionado anteriormente no posee ningún trámite que dependa de él.	El parámetro $idntramite viene vacío.		Muestra un mensaje informando al usuario que el acto procesal no está vinculado a ningún trámite en el sistema.

Tabla 10: Casos de pruebas de Caja Blanca

La prueba de la caja blanca demostró que el estado real del software coincide con el esperado (tabla 11), comprobándose a través de los casos de prueba desarrollados con la herramienta PHPUnit donde la ejecución de las condiciones tuvieron resultados satisfactorios, como se muestra en el **Anexo 2** correspondiente al Caso de Prueba 1 y en el **Anexo 3** correspondiente al Caso de Prueba 2.

Número	$número	$arreglo	Resultado esperado	Resultado obtenido
1	90000023	array(“9,7,13”)	true	true

2	90000023	Array()	false	false

Tabla 11: Resultados de los casos de pruebas realizados con PHPUnit

3.4. Conclusiones parciales

Durante este capítulo se describieron las pruebas a las cuales estuvo sometida la propuesta de solución con el objetivo de garantizar la calidad y funcionalidad de la misma. En particular:

- Se realizaron pruebas de caja negra donde se probaron las interfaces que se utilizaran para la configuración del acceso a los expedientes en el marco de trabajo SAUXE del SIT. Estas pruebas se llevaron a cabo en 3 iteraciones, corrigiendo los errores encontrados en cada una de ellas y asegurando una propuesta de solución totalmente funcional.

- Se desarrollaron pruebas de caja blanca que permitieron conocer la complejidad ciclomática de cada procedimiento, en este caso del procedimiento insertarTramiteRolSistema ($id_rol,$idntramite) teniendo como resultado que la funcionalidad revisada mostró estar correctamente construida.

Conclusiones generales

Al finalizar el presente trabajo investigativo se arribó a las siguientes conclusiones:

- El estudio realizado sobre las diferentes tecnologías de desarrollo permitió seleccionar las herramientas y metodologías más adecuadas a utilizar.

- El control de las peticiones que realiza una funcionalidad una vez desarrollado posibilitó filtrar las peticiones de cada una de las funcionalidades así como los permisos que tiene un rol sobre dichas funcionalidades, que anteriormente solo existía de forma visual.

- La configuración de las listas de control de acceso a los expedientes en el marco de trabajo Sauxe permitió asignarle un rol a cada trámite según el acto procesal.

- La aplicación de las pruebas tanto al código como a las interfaces, permitió corregir las no conformidades de manera satisfactoria arrojando los resultados esperados por el equipo de desarrollo.

Recomendaciones

Una vez concluido el presente trabajo se recomienda:

- Continuar el estudio del tema con el objetivo de incluir nuevas funcionalidades en versiones posteriores del sistema.

- Continuar el proceso de pruebas de software a la propuesta de solución para aumentar la eficiencia y confiabilidad de la misma.

Bibliografía

Alex, Ben and Taylor, Luke. Spring Security. [Online] http://static.springsource.org/spring-security/site/docs/3.1.x/reference/springsecurity-single.html.

Alvarez, Miguel Angel. 2007. Aptana Studio. [Online] 2007. http://www.desarrolloweb.com/articulos/aptana-studio.html.

Architect, Enterprise. 2009. *Información General de Enterprise Architect.* 2009.

Aulbach, Alexander, et al. 2002. *Manual de PHP.* 2002.

Barrios Díaz, Ricardo J. 2007. Sencha. [Online] Marzo 2007. http://www.sencha.com/learn/legacy/Tutorial:Introduction_to_Ext_%28Spanish%29.

Baryolo, Gómez, Tenrero, Cabrera, Marianela y Silega, Martínez, Nemuris. 2008. *Plantilla Registro de la Propiedad intelectual(Sauxe).* La Habana : s.n., 2008.

Boda, Pedro, et al. 2009. *Zend Framework. Manual en español.* s.l. : Zend Technologies Inc, 2009.

Carrillo Pérez, Isaías, Pérez González, Rodrigo and Rodríguez Martín, Aureliano David. 2008. *Metodologías de desarrollo de software.* 2008.

Cake Software Foundation, Inc. 2010. CakePHP. [Online] 2010. http://book.cakephp.org/1.3/es/view/1242/Listas-de-Control-de-Acceso.

Fernández Lanvin, Daniel. 2009. *Definición de una arquitectura software para el diseño de aplicaciones web.* Oviedo : s.n., 2009.

Flanagan, David. 2002. *The Definitive Guide.* 2002.

Gamma, Erich. 2011. *Patrones del "Gang of Four".* Madrid : s.n., 2011.

Gómez Baryolo, Oiner and Tenrero, M. 2010. *Libro de Ayuda de la Arquitectura Tecnológica del ERP-Cuba, en la versión 2.0 del Marco de Trabajo.* 2010.

Gómez, Julián. 2007. softonic. [Online] INTERSHARE, S.L, 2007. [Cited: Abril 4, 2012.] http://axure-rp.softonic.com/.

Help, Axure RP 5. 2009. INTRODUCING AXURE RP PRO. 2009.

IBM. 2008. *Rational Rose Enterprise.* 2008.

Inc, Scribd. 2012. Scribd. [Online] Scribd Inc, 2012. http://es.scribd.com/doc/37352693/Dbdesigner-4-Mini-Manual.

Informática, Centro Técnico de. 2006. *El Framework de desarrollo del Consejo Superior de Investigaciones Científicas.* Sevilla : s.n., 2006.

Jacobson, Ivar y Booch, Grady. 2001. *El Proceso Unificado de Desarrollo de Software.* s.l. : Pearson Addisson-Wesley, 2001.

López Barrio, Carlos. 2005. *Metodología de desarrollo: Programación Extrema: Programa de Doctorado Ingeniería de Sistemas Electrónicos para Entornos Inteligentes.* 2005.

Microsoft. 2009. Centro de Información de Productos de Microsoft. [Online] 2009. http://www.microsoft.com/products/info/product.aspx?view=44&pcid=0a05620b-d256-487f-88d7-ceaa334cf95a&type=ovr.

Navarro, Maikel. 2009. *Documento de Arquitectura de Software.* La Habana : s.n., 2009.

Netbeans. symfony.es. [Online] http://www.symfony.es/2009/10/05/netbeans-ya-incluye-soporte-para-symfony.

Palacio, Juan. 2008. *ScrumManager: Gestión de proyectos.* s.l. : Safe Creative, 2008.

PostgreSQL, Global Development Group. 2010. *PostgreSQL. Documentation.* 2010.

Practical Role-Based Access Control. **Galante, Victoria. 2009.** 2009, Information Security Journal: A Global Perspective.

Pressman, R. S. 2001. *Ingeniería de Software. Un enfoque práctico.* 2001.

Pressman, Roger S. 2005. *Ingeniería de Software, un enfoque práctico.* Quinta edición. 2005. pp. 281-303.

Ramió Aguirre, Jorge. 2006. *Libro electrónico de Seguridad Informática y Criptografía.* Madrid : s.n., 2006. 84-86451-69-8.

Rojas, Johanna and Barrios, Emilio. 2007. Grupo ARQUISOFT. [Online] 2007. http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node26.html.

Sánchez, Jordi. 2006. jordisan.net. [Online] Septiembre 29, 2006. http://jordisan.net/blog/2006/que-es-un-framework.

Serrano, Jorge. 2012. msdn. [Online] 2012. http://msdn.microsoft.com/es-es/netframework/aa496123

Software and Systems, Popkin. *Modelado de Sistemas com UML.*

Solís, Jonathan. 2009. [Online] Enero 29, 2009. http://www.equipom45.es/m45blog/25-seguridad/135-control-acceso-basado-roles.html.

Sommerville, I. 2002. Ingeniería de Software. [Online] Pearson Educación, 2002.

2004. *Visual Paradigm for UML.* 2004.

Smith, L. 2009.. *Introduction to the Doctrine Object Relational Mapper.* 2009.

Visio, Microsoft Office. 2007. 2007.

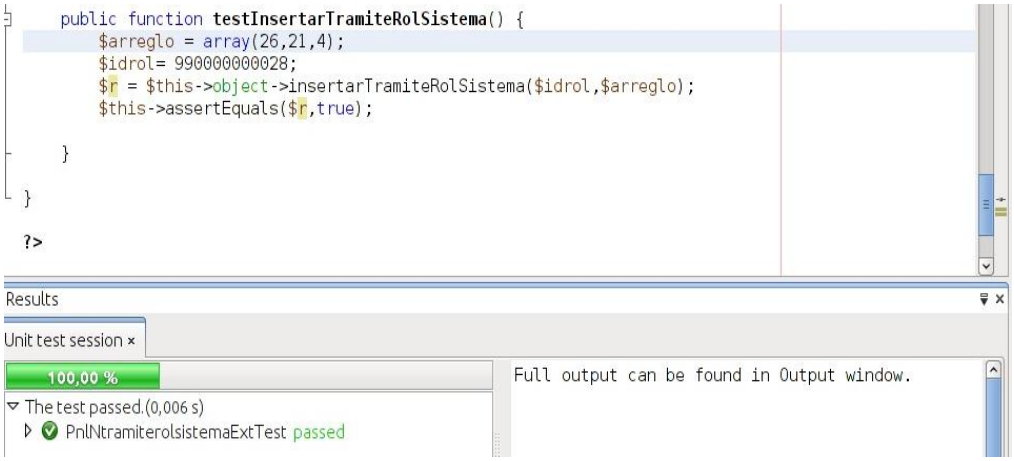
Walsh, Bob. 2012. balsamiq. [Online] Balsamiq Studios, LLC, 2012. [Cited: Abril 4, 2012.] http://www.balsamiq.com/products/mockups.

Wesley, Addison. 2000. *El Proceso Unificado de Desarrollo de Software.* 2000.

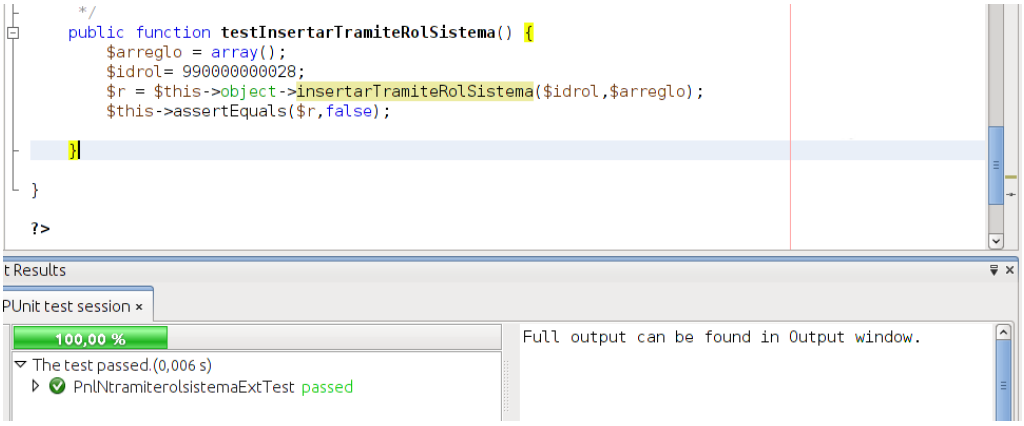
Anexos

(Consultar documento con el nombre "Descripción de las modificaciones en el módulo de seguridad del marco de trabajo Sauxe para el SIT 1.0.0.1.doc")

Anexo 1: Documento - Descripción de las modificaciones en el módulo de seguridad del marco de trabajo Sauxe para el SIT.



Anexo 2: Prueba de Caja Blanca (camino 1)



Anexo 3 Prueba de Caja Blanca (camino 2)