# Universidad de las Ciencias Informáticas

## Facultad 3

**Título:** Componente para el control de acceso en el marco de trabajo Symfony 1.3 para el proyecto productivo Sistema de Informatización de la Gestión Fiscal II.

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


**Autores:**

- Ivis Maray Amaró Moreno
- Felipe Hernández Salazar

**Tutores:**

- Ing. Maribel Silva Muñoz
- Ing. Yenier Figueroa Machado


La Havana, Junio de 2012

"Año del 54 Aniversario de la Revolución"

# ***Declaración de Autoría***

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

**Autores:**

Felipe Hernández Salazar. Ivis Maray Amaró Moreno.

**Tutores:**

Maribel Silva Muñoz. Yenier Figueroa Machado.

# ***Datos de Contacto***

**Tutor:** Ing. Maribel Silva Muñoz.

- Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el año 2008.
- Profesor de la Universidad de las Ciencias Informáticas, en la Disciplina de Programación 1.
- Correo electrónico: msmunoz@uci.cu

**Co-Tutor:** Ing. Yenier Figueroa Machado.

- Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI).
- Profesor de la Universidad de las Ciencias Informáticas, en la Disciplina de Programación.
- Jefe del Proyecto Productivo Sistema de Informatización de la Gestión Fiscal II de la Facultad 3.
- Correo electrónico: yfigueroa@uci.cu

**Autor(es):** Felipe Hernández Salazar, Ivis Maray Amaró Moreno.

- Correos electrónico:
  - fsalazar@estudiantes.uci.cu
  - imamaro@estudiantes.uci.cu

# ***Dedicatoria***

*Le dedico este sueño realizado a mis padres que han sufrido mi carrera tanto o más que yo, por tanto amor profesado sin condición, dedicación y empeño; por creer en mí, incluso cuando yo no lo hacía.*

*Gracias por ser mis padres los amo, espero que estén orgullosos de mí como yo lo estoy de ser su hija.*

*Ivis*

*Le dedico este lauro en primer lugar a mi mamá pues significa todo en mi vida ya que sin ella no hubiese sido posible la realización de este trabajo, por su amor incondicional y estar presente para mí en todo momento, también a mi hermanita querida por escucharme y estar atenta a mis necesidades y problemas, por apoyarme y aconsejarme en todo momento.*

*A mis abuelos, mis tías, mi tío político y prima también*

*le dedico este trabajo pues apoyaron a mi mamá cuando más lo necesitaba y así lo hicieron conmigo.*

*A mi novia Yamaris por soportarme estos días de estrés y comprenderme, por su apoyo, su amor y cariño pues ella constituye también una gran parte en mi vida.*

*Felipe*

# ***Agradecimientos:***

Le agradezco con todo mi corazón y todo el amor del mundo a:

A mis padres que tanto han hecho por mí sin ninguna condición, por apoyarme y darme fuerzas para seguir adelante. Por enseñarme desde pequeña cómo ser mejor persona y cómo forjarme como una buena profesional.

A mi mamá por ser lo mejor de mi vida, por quererme y ser siempre para mí; por quitarse sus cosas por tal de que no me falte nada, por tratar que siempre de lo mejor y enseñarme que todo en la vida lleva una cuota de sacrificio.

A mi papá por ser el mejor padre del mundo, por solo pensar en mí y mi bienestar, por tener paciencia y delicadeza, por querer lo mejor para mí. Por ayudarme a superarme y guiarme por el mejor camino sin presionarme ni imponerse.

A toda mi familia en especial a mi familia por parte de madre, donde cada uno de mis tíos y primos siempre fueron padres y hermanos para mí. A Mime por siempre preocuparse por mí y mi papa, por sus jaranas y siempre hacerme reír. A mi primo Aleine que aunque no te conocí desde pequeña has pasado a formar parte de mi vida por todo el amor y el cariño que siempre me has dado, gracias por tus consejos.

A Duany por estar siempre presente tanto con un abrazo, un beso, un consejo o con un buen regaño; gracias por ser mi hermano de cariño.

A mi novio David por saber sacar lo mejor de mí en tan poco tiempo, por enseñarme a amar, por quererme como lo haces; con mis virtudes y defectos.

A mis amigos de siempre, a mis amistades por compartir los momentos alegres y por siempre estar junto a mí en los momentos tristes; Yanetzi Jimeno, Reinier, Mario, Ariadna, Lianet, Yasmani (el lokol), Odalis.

A todos mis compañeros del grupo de primer año, y muchos de ellos pasaron a ser más que compañeros; a Dasieli, Yadier, Manuel, Yosvanis.

A mi compañero de tesis Felipón como le digo cariñosamente. A mi co-tutores Yenier y en especial a mi tutora Maribel porque más que una tutora se convirtió en uno de nosotros, realmente fue una muy buena guía y ejemplo.

A todos los profes que me impartieron clases. A todas las personas que de una forma u otra formaron parte de mi vida durante estos cinco años.

A la UCI por ser para mí más que una escuela, con sus cosas buenas y malas me ha hecho ser una mejor persona, por brindarme la oportunidad de formarme como profesional y cumplir unos de mis sueños.

*Ivis*

En los agradecimientos casi siempre se cometen injusticias, pues la memoria es a menudo traicionera. Pero sabiendo que mis palabras no bastan para agradecerles puedo afirmar sin lugar a dudas que este trabajo no hubiese sido posible sin la ayuda de:

Mi familia “especialmente mi mamá” la cual durante todos mis años de vida se desarrolló como madre y padre para mí estando presente en todo momento ante mis necesidades, a mi hermana Edelsys pues ella constituyó un apoyo en mis estudios un punto de referencia alguien a quien imitar por su empeño y ahínco, a mis abuelos Caridad y Ramón que representan mis segundos padres terrenales y les debo gran parte de lo que soy, a mi tías Carito y Eliete que no tienen comparación alguna, a mi prima Dianelis que me inspira con todos su años de estudios lejanos, a mi tío Moño el cual siempre se preocupa por mi dándome su apoyo y sus sabios consejos, a mi papá, a una persona que no por ser la última tiene menor importancia pues sin ella verdaderamente en estos no estuviera diciendo estos agradecimientos a mi queridísima tutora Maribel la mejor tutora del mundo a usted le agradezco con todo mi corazón y con profunda gratitud la inmensa ayuda brindada y consejos adquiridos en verdad muchísimas gracias.

También agradezco a mis compañeros que me han soportado durante todo este tiempo y han sabido sobre llevarme, a mis antiguos compañeros de aula en primer, segundo, tercero,cuarto y actual quinto año y en especial a mis compañeros de cuarto Pedro Frank, Arlyh, Pedro Enrique, Giorgy, Manuel, Jose, Yosmany y mi hermano Carlos.

*Felipe*

# ***Resumen***

En el proyecto productivo Sistema de Informatización de la Gestión Fiscal II (SIGEF II) se lleva a cabo el desarrollo de un sistema informático para la gestión de los procesos que suceden en la Fiscalía General de la República de Cuba. Sin embargo, este sistema informático no incluye un mecanismo para gestionar los privilegios apropiados de usuarios y grupos hasta el nivel de objetos del dominio. El siguiente trabajo expone los resultados de una investigación que obtiene como artefacto principal un componente para el control de acceso en el marco de trabajo Symfony 1.3 para el proyecto productivo Sistema de Informatización de la Gestión Fiscal II; se muestran las tecnologías utilizadas para el desarrollo del sistema informático que se propone; el cual facilita la configuración de la seguridad a distintos niveles mediante la utilización de interfaces cómodas y amigables.

# ***Índice***

# ***Figuras***

# ***Tablas***

# ***Introducción***

La tendencia de desarrollo de software en la actualidad va dirigida en gran medida hacia aplicaciones del tipo distribuidas, entiéndase aquellas que cuentan con distintos componentes que se ejecutan en entornos separados, normalmente en diferentes plataformas conectadas a través de una red. La UCI como parte de su estrategia de desarrollo y en correspondencia con las peticiones de los clientes cuyos contratos se encuentran vigentes en cada centro de producción se inserta dentro del desarrollo de Aplicaciones Web Distribuidas.

La Fiscalía General de la República en su afán de automatizar los procesos que tienen lugar en la institución ha contratado el servicio de la Universidad de las Ciencias Informáticas (UCI) de forma que se informaticen los procesos que allí suceden.

De acuerdo a los requerimientos y características propias de los procesos de la Fiscalía General de la República de Cuba es notable la necesidad de la introducción de mecanismos que garanticen la seguridad en la información que se gestiona, estos mecanismos deben incluir el control de acceso a los recursos sobre la base de la asignación de permisos a los roles.

Las tecnologías seleccionadas en el proyecto para la gestión de la seguridad implican el uso del marco de trabajo[1] Symfony en su versión 1.3. La arquitectura que propone el marco de trabajo así como la forma en que se gestiona la seguridad "por su interés a los efectos de la investigación" se describen a continuación:

Un proyecto desarrollado en el marco de trabajo Symfony 1.3 consta de aplicaciones compuestas por módulos y estos a su vez por acciones. El control de acceso en cada nivel se realiza mediante el archivo de configuración *security.yml* donde se especifica el rol autorizado a acceder a los recursos disponibles y con qué permiso puede hacerlo. La configuración de los archivos *security.yml* debe realizarse con la sintaxis y formato específicos lo cual requiere del conocimiento necesario para realizar procedimiento.

Los elementos enunciados anteriormente muestran la no existencia de un mecanismo para determinar los permisos de acceso a un determinado objeto del sistema informático, el funcionamiento propio del marco de trabajo requiere de usuarios con

[1]**Marco de trabajo,** estructura conceptual y tecnológica de soporte definido normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado.

conocimiento en la configuración de archivos en formato YML[2], así como que la gestión de la seguridad de los procesos que suceden actualmente en el proyecto productivo Sistema de Informatización de la Gestión Fiscal II no se encuentra informatizada hasta nivel de objeto.

Tomando como base lo antes descrito se da comienzo a una investigación con el siguiente **problema:** ¿Cómo realizar el control de acceso en el marco de trabajo Symfony 1.3 para el proyecto productivo Sistema de Informatización de la Gestión Fiscal II, de forma tal que contribuya a la gestión de los privilegios apropiados a usuarios, grupos y objetos?, teniendo como **objeto de estudio:** Proceso de Desarrollo de software para la gestión de la seguridad en el marco de trabajo Symfony 1.3, con el **objetivo general:** Desarrollar un componente para el control de acceso en el marco de trabajo Symfony 1.3 en el proyecto productivo Sistema de Informatización de la Gestión Fiscal II. Se desarrolla la investigación bajo el siguiente **campo de acción:** Proceso de desarrollo de componentes de software para el control de acceso en el marco de trabajo Symfony 1.3, tomando como **idea a defender:** El desarrollo de un componente para el control de acceso en el marco de trabajo Symfony 1.3 en el proyecto productivo Sistema de Informatización de la Gestión Fiscal II contribuirá a la gestión de los privilegios apropiados a usuarios, grupos y objetos, donde se tienen los siguientes **objetivos específicos:**

- ✓ Realizar el marco teórico de la investigación.
- ✓ Obtener el Modelo de Diseño.
- ✓ Obtener el Modelo de Implementación.
- ✓ Validar la solución propuesta.

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

- ✓ Estudio del estado del arte del proceso de desarrollo del software para la seguridad en el marco de trabajo Symfony 1.3.
- ✓ Obtención del Modelo de Diseño.
- ✓ Obtención del Modelo de Implementación.
- ✓ Diseño para la implementación de casos de pruebas.

Cumpliendo con cada **objetivo específico** que tributa al cumplimiento del **objetivo general** se tiene como **posible resultado**: Un componente para la gestión de la

[2] YAML es un formato de serialización de datos legible por humanos inspirado en lenguajes como XML, C, Python, Perl.

seguridad para el proyecto productivo Sistema de Informatización de la Gestión Fiscal II.

Para dar respuesta a las tareas planteadas con anterioridad se proponen **métodos científicos** entre los que se encuentran los siguientes:

**Teóricos:**

**Método histórico lógico:**

La seguridad en cuestiones informáticas es un ente fundamental para el funcionamiento eficiente de cualquier sistema informático, en el proyecto productivo SIGEF II desarrollado en el marco de trabajo Symfony 1.3 se requiere la implantación de políticas de seguridad que permitan definir las acciones que pueden realizar los usuarios del sistema informático sobre los objetos del dominio que se informatiza. Este cambio que requiere el proyecto productivo SIGEF II se debe a que se limita la gestión de la seguridad hasta el control del acceso sobre las acciones disponibles en cada uno de los módulos del sistema.

**Método dialéctico:**

El proyecto productivo SIGEF II no está llevando a cabo la gestión de la seguridad a nivel de objetos del dominio en el proceso de desarrollo del sistema que implementa, por lo que se hace necesario informatizar el proceso de gestión de la seguridad en el proyecto productivo Sistema de Informatización de la Gestión Fiscal II, de forma tal que permita conocer si un usuario o grupo de ellos posee los privilegios necesarios para ejecutar una determinada acción sobre un objeto del dominio.

**Analítico - sintético:**

Partiendo de un detallado análisis sobre toda la información existente de la gestión de la seguridad a nivel de objeto en el marco de trabajo Symfony 1.3, se logrará reunir la información necesaria para el analizar herramientas y mecanismos de control de acceso desarrollados para los mismos fines.

**Empíricos:**

**Método de Medición:**

El componente obtenido como resultado de la investigación será medido mediante las pruebas unitarias y funcionales garantizando el resultado factible de la investigación.

**Este trabajo ha sido organizado en tres capítulos de la siguiente manera:**

El **Capítulo 1: Fundamentación Teórica** se enmarca en la Fundamentación Teórica de la investigación. Para ello se aborda el tema de la gestión de la seguridad en el proyecto productivo SIGEF II mediante un estudio del arte de los procesos de gestión de seguridad en sistemas informáticos hasta un nivel de objeto. Además se describen en este capítulo las principales herramientas, plataformas de desarrollo y metodología usadas.

El **Capítulo 2: Planificación, Diseño e Implementación** maneja elementos relacionados con la descripción y análisis del sistema informático que se propone. Se muestran las relaciones de los distintos artefactos generados durante el proceso de implementación de la solución propuesta.

El **Capítulo 3: Validación de los Resultados** hace muestra de los resultados de las pruebas realizadas al sistema informático obtenido como resultado de la investigación. El análisis de estos resultados valida la solución propuesta por la investigación.

# 1 Capítulo 1: Fundamentación Teórica

En el presente capítulo fundamenta teóricamente la tendencia del desarrollo web actual en el mundo y particularmente en la UCI, así como software y mecanismos de control de acceso para la implementación de ACL y su aplicación en el proceso de gestión de la seguridad hasta un nivel de objeto en el marco de trabajo Symfony 1.3. Describiendo a lo largo de este las principales herramientas, plataforma de desarrollo y metodología que se sugieren para la solución informática propuesta.

En la Fiscalía General de la República resulta indispensable el tratamiento de la seguridad como parte del flujo básico del tratamiento de información relevante e incluso clasificada que se maneja en todas las entidades en cada nivel. Es por ello que garantizar el control de acceso a los objetos, vistos estos como recursos, resulta requisito indispensable de la propuesta de solución informática que se presenta como resultado de la investigación.

## 1.1 Mecanismos de Control de Acceso

La técnica de llevar a cabo la restricción hacia usuarios y grupos a determinadas partes del sistema así como a objetos vinculados del dominio es conocido como control de acceso. (1)

A continuación se relacionan los mecanismos de acceso controlado a recursos que constituyen precedente de garantía de seguridad a nivel de objeto.

### 1.1.1 DAC – Control de Acceso Discrecional

El modelo de Control de Acceso Discrecional (DAC, por sus siglas en inglés, Discretionary Access Control), también llamado modelo de seguridad limitada, es un modelo no orientado al control del flujo de información.

Los modelos DAC están basados en la idea de que el propietario de un objeto, su autor, tiene el control sobre los permisos del objeto. Es decir, el autor es autorizado a permitir u otorgar permisos para este objeto a otros usuarios.

DAC admite la copia de datos desde un objeto a otro por usuarios autorizados de manera que un usuario puede permitir el acceso para copiar datos a otro usuario no autorizado. Este riesgo puede ser extendido a todo el sistema violando un conjunto de objetos de seguridad. La principal ventaja de DAC es que el usuario se beneficia de la flexibilidad del modelo. Sin embargo es difícil con este mecanismo garantizar las reglas de integridad como “menos privilegio” o “separación de responsabilidad”

que son necesarias en los ambientes con procesos colaborativos. DAC es apropiado en ambientes donde la compartición de información es más importante que su protección. (2)

### 1.1.2 MAC – Control de Acceso Obligatorio

En el modelo de Control de Acceso Obligatorio (MAC, por sus siglas en inglés, Mandatory Access Control) todos los sujetos y objetos son clasificados basándose en niveles predefinidos de seguridad que son usados en el proceso de obtención de los permisos de acceso. Para describir estos niveles de seguridad todos los sujetos y objetos son marcados con etiquetas de seguridad que siguen el modelo de clasificación de la información militar (desde “desclasificado” hasta “alto secreto”), formando lo que se conoce como política de seguridad multinivel. Por este motivo se define MAC como un modelo “multinivel”. (3)

Este modelo puede ser implementado usando mecanismos de seguridad multinivel que usan reglas “no read-up” y “no write-down”. Estas reglas son diseñadas para asegurar que la información no fluya desde un nivel alto de sensibilidad a un nivel más bajo de sensibilidad. La regla “no read-up”, establece que los usuarios tienen la capacidad de acceder a cualquier fragmento de información que se encuentre en o por debajo de su nivel de seguridad. Por ejemplo, si un usuario tiene como nivel asignado “secreto”, este podrá acceder al nivel “básico, medio y secreto”, pero no podrá acceder al nivel “ultra secreto”. La regla “no write-down “, declara que un sujeto con un nivel de seguridad dado no debe escribir en ningún objeto etiquetado con un nivel más bajo de seguridad. Por ejemplo, si un usuario tiene como nivel asignado “secreto”, este podrá acceder a escribir cosas en el nivel “secreto” o “ultra secreto”.

Un importante objetivo del modelo MAC es controlar el flujo de información en orden a asegurar su confidencialidad y su integridad, objetivo no alcanzado por los modelos DAC. A diferencia de DAC, los modelos MAC proporcionan mecanismos más sólidos para la protección de datos, y tratan con requerimientos de seguridad más específicos, así como, los requerimientos derivados de las políticas de control de los flujos de información. Además, en los modelos MAC es el sistema quien protege los recursos u objetos, el administrador es el que impone las reglas de forma segura, a diferencia del DAC en el cual el dueño es quién protege los recursos. Sin embargo, asegurar las políticas MAC es a menudo una tarea difícil, particularmente en procesos colaborativos, puesto que no proporcionan soluciones factibles ya que le falta suficiente flexibilidad. (1)

### 1.1.3 RBAC – Control de Acceso Basado en Rol

El principal objetivo del modelo de Control de Acceso Basado en rol (RBAC, Role Based Access Control) es prevenir que los usuarios tengan libre acceso a la información de la organización. El modelo introduce el concepto de rol y asocia a los usuarios con los roles por los que va pasando durante la vida del sistema. Los permisos de acceso están asociados a los roles. RBAC permite modelar la seguridad desde una perspectiva empresarial puesto que podemos conectar los requerimientos de seguridad con los roles y las responsabilidades existentes en la organización. RBAC está basado en la definición de un conjunto de elementos y de relaciones entre ellos. (4) A nivel general describe un grupo de usuarios que pueden estar actuando bajo un conjunto de roles y realizando operaciones en las que utilizan un conjunto de recursos. En una organización, un rol puede ser definido como una función que describe la autoridad y responsabilidad dada a un usuario en un instante determinado. Entre estos cuatro elementos se establecen relaciones del tipo:

- Relaciones entre usuario y roles, modelando los diferentes roles que puede adoptar un usuario.
- Conjunto de operaciones que se pueden realizar sobre cada uno de los objetos. A los elementos de esta relación se les denomina permisos.
- Relaciones entre los permisos y los roles modelando cuándo un usuario, por estar en un rol determinado, tiene permiso para realizar una operación sobre un objeto.

El modelo RBAC incluye un conjunto de sesiones donde cada sesión es la relación entre un usuario y un subconjunto de roles que son activados en el momento de establecer dicha sesión. Cada sesión está asociada con un único usuario. Mientras que un usuario puede tener una o más sesiones asociadas. Los permisos disponibles para un usuario son el conjunto de permisos asignados a los roles que están activados en todas las sesiones del usuario, sin tener en cuenta las sesiones establecidas por otros usuarios en el sistema. (5)

RBAC añade la posibilidad de modelar una jerarquía de roles de forma que se puedan realizar generalizaciones y especializaciones en los controles de acceso y se facilite la modelación de la seguridad en sistemas complejos. Otro aspecto importante en el modelo RBAC es la posibilidad de especificar restricciones sobre la relación usuario/rol y sobre la activación de un conjunto de roles de usuario. Estas restricciones son un

fuerte mecanismo para establecer políticas organizacionales de alto nivel. Las restricciones pueden ser de dos tipos: estáticas o dinámicas. Las restricciones estáticas permitieron al equipo de desarrollo solucionar conflictos de intereses y reglas de cardinalidad de roles desde una perspectiva de política de seguridad. La asociación de un usuario con un rol puede estar sujeta de las siguientes restricciones:

- Un usuario es autorizado para un rol solo si el rol no es mutuamente excluyente con cualquier rol autorizado del usuario (Static Separation of Duty, SSD).
- El número de usuarios autorizados para un rol no puede exceder la cardinalidad del rol (Role Cardinality).

La Fiscalía General de la República informatiza sus procesos mediante el proyecto productivo SIGEF II, que debido a su funcionamiento interno hacia la concepción que crea el equipo de desarrollo sobre la gestión de roles y permisos en el sistema con especial atención en el flujo de información así como los privilegios que pueda tener un determinado grupo de usuarios y/o un usuario en específico sobre algún objeto del dominio se tiene el mecanismo de RBAC como solución para el control de acceso a los usuarios, grupos y objetos definidos en la entidad donde este control de acceso vendrá dado por la relación existente entre las operaciones que suceden sobre un determinado objeto y algún algoritmo que permita un chequeo estricto de las operaciones permitidas sobre un objeto así como los permisos propios de un determinado usuario que solicita dicha operación sobre un objeto del dominio.

Un algoritmo que garantiza el control de acceso a recursos a través de la definición de usuarios, grupos de usuarios y permisos lo constituye el mecanismo de ACL.

A continuación se incluyen algunos elementos que justifican su funcionamiento básico.

### 1.2 ACL-Listas de Control de Aceso

Una Lista de Control de Acceso o ACL es un concepto de seguridad informática usado para fomentar la separación de privilegios, siendo una forma de determinar los permisos de acceso apropiados a un determinado objeto, dependiendo de ciertos aspectos del proceso que hace el pedido. (6)

### 1.3 Metodologías de desarrollo

La metodología define quién debe hacer qué, cuándo y cómo debe hacerlo en un proceso de desarrollo de software. Una metodología es la guía que nos va indicando qué hacer y cómo actuar cuando se quiere obtener algún tipo de investigación.

Lamentablemente no existe una metodología universal para el desarrollo de software, sin embargo las características propias de cada equipo de desarrollo de software así como las necesidades que expongan los mismos amerita una configuración personalizada del proceso de desarrollo de software. (10)

Existen dos grandes grupos en los cuáles se dividen las metodologías: las metodologías tradicionales y los procesos ágiles con marcadas diferencias.

Actualmente persisten varias metodologías OO basadas en UML entre las que se encuentran Rational Unified Process (RUP) y XP (Extreme Programming). A continuacion se estará viendo una breve descripción de ambas metodologías de desarrollo de software.

### 1.3.1 El Proceso Unificado de Desarrollo (RUP)

Es una metodología de desarrollo de software que está basado en componentes e interfaces bien definidas, y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Es un proceso que puede especializarse para una gran variedad de sistemas de software, en diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto. RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. Es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML, y trabajo de muchas metodologías utilizadas por los clientes. La versión que se ha estandarizado vio la luz en 1998 y se conoció en sus inicios como Proceso Unificado de Rational de ahí las siglas con las que se identifica a este proceso de desarrollo. (11)

Como RUP es un proceso, en su modelación define como sus principales elementos:

- Trabajadores (“quién”): Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
- Actividades (“cómo”): Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.

- Artefactos (”qué”): Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
- Flujo de actividades (“Cuándo”): Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. Ver Figura 1.

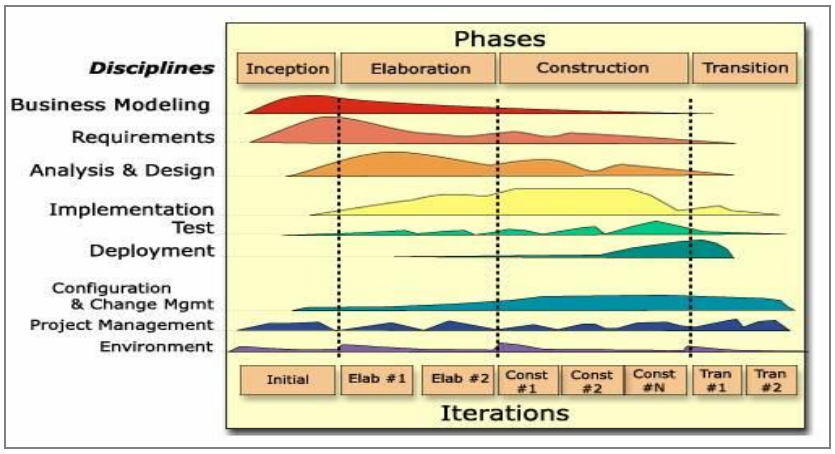



**Figura 1.** RUP en dos dimensiones.

#### 1.3.1.1 El ciclo de vida de RUP

**Características:**

- Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos.
- Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo con la que el equipo del proyecto y los usuarios deben estar de acuerdo, ya que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
- Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. (12)

La metodología RUP (Rational Unified Process) es identificada como perteneciente a los procesos de desarrollo del tipo pesados. Se le llama de esta forma por la extensión en tiempo de desarrollo, así como el gran número de especialistas necesarios para la confección del software. Los negocios de gran magnitud encajan perfectamente en esta metodología puesto que son estos los que involucran a gran cantidad de especialistas de la materia de que se trate, no siendo estos los rasgos distintivos del sistema integrado que se propone en la investigación para el proyecto productivo SIGEF II ya que el mismo no cuenta con un cronograma amplio para su desarrollo así como que el mismo no requiere de un gran número de especialistas en el tema para su desempeño, por lo que esta metodología no se adecua a la necesidades que imponen las características propias del sistema que se propone en la investigación.

### 1.3.2 XP (Programación Extrema)

La Programación Extrema es una metodología ágil de desarrollo de software que se basa en la simplicidad, la comunicación y la retroalimentación o reutilización del código desarrollado. La metodología XP tiene principalmente dos objetivos, la satisfacción del cliente tratando de darle un software que verdaderamente necesita y en el momento que lo necesita; por otro parte XP se propone potenciar al máximo el trabajo en grupo identificando a cada miembro del equipo de desarrollo con el producto entregable al cliente. (13)

La metodología XP propone cuatro actividades básicas para el desarrollo de un buen software las cuales se describen a continuación:

- **codificar:** Es la única actividad de la que no se puede prescindir ya que se necesita codificar y plasmar nuestras ideas a través del código para expresar la interpretación del problema, así se puede utilizar el código para comunicarnos.
- **hacer pruebas:** Las pruebas dan la oportunidad de saber si lo que se implementó es lo que en realidad se pensaba que se había implementado. Las pruebas indican que el trabajo funciona, cuando no se pueda pensar en ninguna prueba que pudiese originar un fallo en el sistema entonces hemos acabado por completo.
- **escuchar:** Se tiene que escuchar a los clientes cuales son los problemas de su negocio, se debe tener una escucha activa explicando lo que es fácil y difícil de obtener, y la realimentación entre ambos ayudan a todos entender los problemas.
- **diseñar:** El diseño crea una estructura que organiza la lógica del sistema, un buen diseño permite que el sistema crezca con cambios en un solo lugar. Los

diseños deben de ser sencillos, si alguna parte del sistema es de desarrollo complejo, divídela en varias. Si hay fallos en el diseño o malos diseños, estos deben de ser corregidos cuanto antes. (14)

Estas técnicas se aplican a proyectos con un equipo de desarrollo medio, para solucionar un problema no trivial. Algunas de las medidas que proponen no tienen sentido para proyectos pequeños.

El ciclo de desarrollo de XP incluye seis fases:

- **exploración:** En esta fase el cliente define el valor de negocio a implementar. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.
- **planificación de la entrega:** En esta fase, el programador estima el esfuerzo necesario para su implementación.
- **iteraciones:** Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Al final de la última iteración el sistema estará listo para entrar en producción.
- **producción:** La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación.
- **mantenimiento:** Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en desarrollo.
- **muerte del proyecto:** Es cuando el cliente no tiene más valores del negocio a ser incluidos en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se

genera la documentación final del sistema y no se realizan más cambios en la arquitectura.

En XP se comienza en pequeño y se añaden funcionalidades con la retroalimentación continua y no antes de que sean necesarias; el manejo de cambios es importante en el proceso; el costo del cambio no depende de la fase o etapa y el cliente o el usuario se convierte en miembro del equipo. (15)

La metodología XP se basa en:

- **pruebas unitarias:** pruebas realizadas a los principales procesos, de tal manera que si se quiere adelantar en algo hacia el futuro, se puedan hacer pruebas de las fallas que pudieran ocurrir. Es como adelantarse a obtener los posibles errores.
- **re fabricación:** se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- **programación en pares:** una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

La metodología seleccionada es XP debido a su adecuación para el desarrollo del sistema informático propuesto como solución en la investigación, pues dicho sistema debe ser implementado atendiendo a un cronograma de desarrollo ajustado al menor tiempo posible debido a la necesidad de garantizar en la Fiscalía General de la República de Cuba el control de acceso a los distintos objetos del dominio por parte de usuarios y grupos que pertenezcan al sistema. Por otra parte los clientes se encuentran integrados en su totalidad al proceso de desarrollo ya que el mismo realiza visitas periódicas verificando el estado del producto así como aclarando dudas respecto a los procesos que se ejecutan en la Fiscalía General de la República, los cuales requieren de una adecuada gestión del acceso. Además dentro de ellos se encuentra el personal que labora en el equipo de producción del proyecto productivo Sistema de Informatización de la Gestión Fiscal II, evidenciándose la interacción directa entre clientes y desarrolladores.

A continuación se presentan las distintas herramientas estudiadas para automatizar el desarrollo del sistema informático propuesto.

## 1.4 Herramientas CASE

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. Como es sabido, los estados en el ciclo de vida de desarrollo de un software son: Investigación Preliminar, Análisis, Diseño, Implementación e Instalación. (16)

CASE se define también como:

- Conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases.
- La sigla genérica para una serie de programas y una filosofía de desarrollo de software que ayuda a automatizar el ciclo de vida de desarrollo de los sistemas.
- Una innovación en la organización, un concepto avanzado en la evolución de tecnología con un potencial efecto profundo en la organización. Se puede ver al CASE como la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales.

Estas herramientas pueden proveer muchos beneficios en todas las etapas del proceso de desarrollo de software, algunas de ellas son:

- Verificar el uso de todos los elementos en el sistema diseñado.
- Automatizar el dibujo de diagramas.
- Ayudar en la documentación del sistema.
- Ayudar en la creación de relaciones en la Base de Datos.
- Generar estructuras de código. (17)

Actualmente existen Herramientas CASE tales como:

- Visual Paradigm for UML.
- Rational Rose.
- Enterprise Architect.
- Umbrello.
- ArgoUML.

### 1.4.1 Rational Rose

Enterprise es un producto de la familia Rational Rose incluye soporte al UML, es la mejor elección para el ambiente de modelado que soporte la generación de código a partir de modelos en Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic. Como todos los demás productos Rational Rose, proporciona un lenguaje común de modelado para el equipo que facilita la creación de software de calidad más rápidamente. (18)

**Ventajas:**

- Soporte para análisis de patrones ANSI C++, Rose J y Visual C++ basado en "Design Patterns: Elements of Reusable Object-Oriented Software".
- Característica de control por separado de componentes modelo que permite una administración más granular y el uso de modelos.
- Soporte de ingeniería Forward y/o reversa para algunos de los conceptos más comunes de Java 1.5.
- La generación de código Ada, ANSI C ++, C++, CORBA, Java y Visual Basic, con capacidad de sincronización modelo- código configurables.
- Soporte Enterprise Java Beans 2.0.
- Capacidad de análisis de calidad de código.
- El Add-In para modelado Web provee visualización, modelado y las herramientas para desarrollar aplicaciones de Web.
- Modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.
- Capacidad de crear definiciones de tipo de documento XML (DTD) para el uso en la aplicación.
- Integración con otras herramientas de desarrollo de Rational.
- Capacidad para integrarse con cualquier sistema de control de versiones SCC-compliant, incluyendo a Rational ClearCase.
- Publicación web y generación de informes para optimizar la comunicación dentro del equipo.

Rational Rose es una herramienta CASE potente que se integra bien con el Proceso de Desarrollo de Software (RUP del inglés) y encaja bien en procesos de negocio complejos y por ende proyectos grandes, sin embargo el sistema requerido como se describe en secciones anteriores no presenta procesos de negocios complejos

sumándole a esto que la presente herramienta es propietaria por lo que atenta contra la política de migración de la universidad; así como que el sistema operativo seleccionado por el proyecto productivo Sistema de Gestión Fiscal II es GNU Linux en sus distribuciones, por lo que esta herramienta no se ajusta a las necesidades actuales.

### 1.4.2 Visual Paradigm for UML

Visual Paradigm for UML es una herramienta CASE profesional que soporta la última versión de UML 2.1 así como el ciclo de vida completo del desarrollo de software, análisis y diseño orientados a objetos, construcción, pruebas y despliegue, exportación desde Rational Rose, exportación/importación XML, generación de informes y edición de figuras. Visual Paradigm tiene disponible distintas versiones: Enterprise, Professional, Standard, Modeler, Personal y Community (que es gratuita). (19)

Visual Paradigm ofrece además:

- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs[3].
- Disponibilidad en múltiples plataformas.
- Ingeniería Inversa Java, C++, Esquemas XML, XML6, NET exe/dll, CORBA DL.
- Ingeniería de ida y vuelta.

Se integra con las siguientes herramientas Java:

- Eclipse/IBM WebSphere
- JBuilder
- NETBeans IDE
- Oracle JDeveloper
- BEA Weblogic

[3] **IDEs**, abreviatura de Entorno Integrado de Desarrollo.

Visual Paradigm desde los inicios del proyecto productivo SIGEF II fue definida como herramienta CASE a utilizar en el proceso de desarrollo del proyecto debido a que primeramente es una herramienta multiplataforma que se adecua con las políticas migratorias de la universidad hacia software libre y gratuito, por otra parte la misma brinda un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores generando porciones de código automáticos en correspondencia con el lenguaje utilizado para el desarrollo del sistema, brindando la capacidad de gestionar el modelado de las entidades de la base de datos así como plasmar las relaciones entre ellas, esta herramienta se integra mejor con la plataforma Java ya que apoya los últimos estándares para las notaciones del lenguaje Java propiamente dicho.

## 1.5 Tendencias y tecnologías en el desarrollo de aplicaciones web

En el mundo de desarrollo de software se define cada vez más una tendencia al desarrollo de sistemas informáticos web. Esta tendencia ofrece grandes ventajas sobre las aplicaciones tradicionales ya que abaratan el coste de las comunicaciones, el de las implantaciones y sobre todo, el mantenimiento posterior, lográndose construir aplicaciones que cumplan con las expectativas de cualquier cliente que busque una herramienta de venta o un uso privado de la red para automatizar sus propios procesos. Sin embargo, a nivel de programación, los desarrolladores de programas en estas tecnologías no tienen las mismas herramientas ni utilizan un único leguaje. (7)

A continuación se justifican las tendencias y tecnologías actuales para el desarrollo de software enmarcado en las principales metodologías, herramientas y leguaje de modelado, plataformas de desarrollo y métricas.

### 1.5.1 UML-Lenguaje Unificado de Modelado

El entendimiento de un sistema a veces resulta difícil así como su administración, por lo que se hace más fácil comprenderlo si tenemos una forma de dividirlo en partes o pequeños fragmentos pudiendo representarse las mismas como modelos que de una forma sencilla abstraigan los aspectos esenciales del sistema que se analiza siendo precisamente un paso fundamental para la creación de un software la creación de modelos que organicen y expresen los aspectos más importantes de la vida real con que se relaciona y del sistema en cuestión. Los modelos se componen de otros modelos, diagramas y documentos que describen cosas, siendo preciso mencionar al Lenguaje Unificado de Modelado (UML). (8)

UML es un lenguaje, que proporciona un vocabulario y reglas para permitir una comunicación. En este caso, este lenguaje se centra en la representación gráfica de un sistema. Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar, clasificando sus elementos en estructurales (cases, interfaces. colaboraciones, casos de uso, clases activas, componentes y nodos), de comportamiento (interacciones y máquinas de estado), de agrupación (paquetes) y de anotación (notas). A su vez, hay cuatro tipos de relaciones: Dependencia, de asociación, de agrupación y de realización. Para construir un plano de software que tenga sentido, lo que se hace es combinar los elementos estructurales con sus respectivas relaciones, según sea el caso, obteniendo como resultado uno de los nueve diagramas que existen en UML, a saber: de clases, de objetos, de casos de uso, de secuencia, de colaboración, de estados, de actividades, de componentes y de despliegue. (9)

**Funciones de UML:**

- ✓ Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- ✓ Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- ✓ Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- ✓ Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

## 1.6 Programación Orientada a Objetos (POO)

La Programación Orientada a Objetos o (POO) es un paradigma de programación que define objetos y sus interacciones, para diseñar aplicaciones y programas informáticos. (20)

**Entre los conceptos fundamentales de este paradigma se encuentran:**

- ➢ Objetos.
- ➢ Clases.
- ➢ Polimorfismo.
- ➢ Encapsulamiento.
- ➢ Envío de mensajes.

- Asociaciones.
- Herencia.
- Agregación & Composición.
- Método.

**clase:** definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas.

**objeto:** entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos) los mismos que consecuentemente reaccionan a eventos. Se corresponde con los objetos reales del mundo que nos rodea, o a objetos internos del sistema (del programa). Es una instancia a una clase.

**herencia:** Herencia es la capacidad de un objeto (clase) para utilizar las estructuras y los métodos existentes en antepasados o ascendientes. Es la reutilización de código desarrollado anteriormente. Cuando usamos herencia decimos que hacemos programación por herencia: Definición de nuevos tipos a partir de otros con los que comparten algún tipo de característica.

**método:** Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema. (21)

**polimorfismo:** El término de polimórfico viene del griego y significa "muchas formas" (poly = muchas, morphos = forma). En ocasiones una operación tiene el mismo nombre en diferentes clases. Por ejemplo podría abrir una ventana, una puerta, un regalo, en cada uno de estos casos, realizará una operación diferente. En POO, cada clase sabe cómo realizar tal operación. Esto es polimorfismo.

**encapsulamiento:** La esencia del encapsulamiento es que cuando un objeto trae consigo su funcionalidad, esta última se oculte. Por ejemplo la mayoría de las personas que ven televisor no se preocupan de la complejidad electrónica que hay detrás de la pantalla, ni de todas las operaciones que tienen que ocurrir para mostrar una imagen en la pantalla. Esto es de gran importancia en el mundo del software porque permite reducir el potencial de errores que pueden ocurrir. En un sistema que consta de objetos, éstos dependen unos de otros en diversas formas. Si uno falla y

están ocultas las operaciones de otros objetos sólo es necesario modificar el que falló y no el resto. Un objeto tiene que presentar un rostro al mundo exterior para poder iniciar sus operaciones.

### 1.7 Marco de Trabajo

El marco de trabajo seleccionado por la dirección del proyecto es Symfony 1.3 debido a que el mismo está diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características las cuales se exhiben a continuación: separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web, proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja y automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

El marco de trabajo Symfony 1.3 está basado en un patrón del diseño web conocido como arquitectura MVC (Modelo-Vista-Controlador), que está formado por tres capas:

- ✓ El Modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- ✓ La Vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- ✓ El Controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura MVC separa la lógica de negocio (el Modelo) y la presentación (la Vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El Controlador se encarga de aislar al modelo y la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, entre otras.). El Modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación. (22)

**Estructura del proyecto: Aplicaciones, Módulos y Acciones del marco de trabajo Symfony 1.3.**

Un proyecto en el marco de trabajo de Symfony 1.3 cuenta con una jerarquía de directorios bien definida ya que se organiza de forma lógica en aplicaciones las cuales se ejecutan en modo independiente respecto a otras aplicaciones del mismo proyecto, cada aplicación está compuesta por uno o más módulos que almacenan las acciones que representan cada una de las operaciones que se pueden realizar en un módulo determinado.

Symfony 1.3 está desarrollado completamente en PHP 5.3. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

Symfony 1.3 incluye como novedades destacadas:

- **rendimiento:** Mejora en el rendimiento de todos sus componentes. Las mejoras más notables se producen en el sistema de enrutamiento y en una nueva tarea llamada project:optimize que optimiza el rendimiento de los proyectos en producción.
- **emails:** Cuenta con un mecanismo sencillo, potente e integrado para el envío de emails a través de la librería SwiftMailer 4, proyecto recientemente incorporado a la gran familia de Symfony.
- **formularios:** Sus formularios son tan fáciles de extender e intuitivos de utilizar ya que ahora los formularios incluyen eventos para redefinir fácilmente su comportamiento y se han añadido varios métodos para facilitar su creación y uso.
- **depuración:** la barra de depuración web incluye dos nuevos paneles dedicados a la vista y al envío de emails. El panel de la vista muestra información sobre las plantillas y elementos parciales utilizados para crear la página que visualiza el navegador y también incluye el listado completo de variables disponibles en la parte de la vista.

- **eficiencia:** gracias al nuevo cargador automático de clases, ya no será necesario limpiar la caché con el comando symfony cc cada vez que añades una nueva clase. Las pruebas unitarias y funcionales también son más eficientes porque permiten volver a ejecutar solamente las pruebas que produjeron un error la última vez.

Ventajas del marco de trabajo Symfony 1.3:

- Permite la internacionalización para la traducción del texto de la interfaz, los datos y el contenido de localización.
- La presentación usa templates y layouts[4] que pueden ser construidos por diseñadores de HTML[5] que no posean conocimientos del marco de trabajo.
- Los formularios soportan la validación automática, lo cual asegura mejor calidad de los datos en las base de datos y una mejor experiencia para el usuario.
- El manejo de cache reduce el uso de banda ancha y la carga del servidor.
- La facilidad de soportar autenticación y credenciales facilita la creación de áreas restringidas y manejo de seguridad de los usuarios.
- El enrutamiento y las URLs[6] inteligentes hacen amigable las direcciones de las páginas de la aplicación.
- Las listas son más amigables, ya que permite la paginación, clasificación y filtraje automáticos.
- Los plugins[7] proveen un alto nivel de extensibilidad.
- La interacción con AJAX[8] es mucho más sencilla.

El marco de trabajo Symfony 1.3 hará aportes significativos al sistema informático propuesto pues permitirá usar de forma rápida las funciones de acceso Ajax para evitar recargar la página, así como importar código reutilizable a través del uso de los plugins[9] para el manejo de funcionalidades vinculadas al sistema permitiendo manipular la autenticación de los usuarios y credenciales gestionando el acceso que posee el usuario a la información; constituyendo herramienta para el desarrollo del sistema llevado a cabo por dicho proyecto ya que viene a fin con las prestaciones solicitadas y el tipo de sistema a desarrollar.

---

[4] L**ayout,** utilizado para nombrar al esquema de distribución de los elementos dentro un **diseño**.

[5] **HTML**, siglas de ***HyperText Markup Language*** («lenguaje de marcado de hipertexto»)

[6] **URL Uniform Resource Locator**, es decir, localizador uniforme de recurso y se refiere a la dirección única que identifica a una página web.

[7] Es un complemento (o plug-in en inglés) de una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica.

[8] Ajax, acrónimo de Asynchronous JavaScript And XML (Javascript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas.

[9] Plugin, aaplicación que se relaciona con otra para aportarle una nueva función .

## 1.8 Entorno Integrado de Desarrollo (IDE)

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica, los mismos proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, PHP, Python, Java y C# entre otros. (23)

El Entorno Integrado de desarrollo seleccionado por la dirección del proyecto es Netbeans 6.9.1.

**Creación de Proyectos PHP:** Netbeans 6.9.1 provee de una estructura para los proyectos que se puedan crear junto a este IDE, propone un esqueleto para organizar el código fuente, el editor conjuntamente integra los lenguajes como HTML, JavaScript[10] y CSS[11]. Además NetBeans posee un sistema para examinar todo los directorios de cada proyecto, haciendo reconocimiento y carga de clases, métodos y objetos, para acelerar la programación. (24)

**Integración con Symfony 1.3:** Netbeans 6.9.1 se integra de forma idónea con el marco de trabajo Symfony 1.3 dejando de lado la consola de comandos de Symfony y permitiendo centrar al desarrollador en la implementación sobre el IDE, además se encuentra cargadas todas las clases, ayuda en línea.

**Editor de Código Fuente:** Netbeans 6.9.1 mejora su editor, sobre todo en el editor de PHP, es mucho más ágil y a la vez robusto, contiene más ayuda en línea, reconocimiento de sintaxis y todo lo que provee la última versión de PHP, la 5.3

**Integración PHP –Prueba de Unidad:** Es posible crear pruebas con PHPUnit, para diferentes funciones, luego realizar la comprobación y ver todos los resultados. En las propiedades PHPUnit puede definir una configuración personalizada de archivos XML, un archivo de arranque para las opciones de línea de comandos, o una serie de pruebas a medida, o puede que el IDE genera el código esqueleto.

**Depuración de PHP:** NetBeans integra muy bien la utilización Xdebug, gracias a esto se puede inspeccionar y examinar cada variable local, establecer puntos de interrupción y evaluar el código en nuestra lógica. El IDE de NetBeans para PHP también ofrece la línea de comandos de depuración: La salida del programa PHP

---

10JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript, basado en prototipos, imperativo, débilmente tipado y dinámico.

11 CSS (Cascading Style Sheets, u Hojas de Estilo en Cascada) es la tecnología desarrollada por el World Wide Web Consortium (W3C) con el fin de separar la estructura de la presentación.

aparece en una pantalla de línea de comandos en el IDE de sí mismo y se puede inspeccionar el código HTML generado sin tener que cambiar a un navegador. (24)

**Integración con Sistemas de Control de Versiones:** Esta es una de las condiciones necesarias para los proyectos y es la posibilidad de contar con la integración de sistemas de control de versiones, tales como SVN, CVS, Mercurial y Git.

El Entorno Integrado de Desarrollo integra bien con el lenguaje php 5.0 que implementa el propio marco de trabajo seleccionado, así como la vinculación de forma directa que existe entre el marco de trabajo de Symfony 1.3 con el IDE antes visto ,brindado un conjunto de comodidades en términos de implementación.

## 1.9 Sistema de Gestión de Base de Datos

En la actualidad los Sistemas de Gestión de Base de Datos (SGBD) sirven de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan, teniendo como propósito manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización.

Seguidamente se exponen las características del Sistema de Gestión de Base de Datos seleccionado por el equipo de desarrollo así como los argumentos de su selección.

### 1.9.1 PostgreSQL

PostgreSQL es un servidor de base de datos objeto relacional libre, liberado bajo la licencia BSD. Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo, dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group). Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos explicadas en epígrafes anteriores. (25)

#### 1.9.1.1 Características de PostgreSQL:

- Implementación del estándar SQL92/SQL99.
- Soporta distintos tipos de datos, además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, etc. También permite la creación de tipos propios.

- Incorpora una estructura de datos array[12].
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

PostGreSQL es un gestor de bases de datos disponible en casi cualquier Unix (34 plataformas en la última versión estable), y una versión nativa de Windows, usando una estrategia de almacenamiento de filas llamada MVCC[13] para conseguir una mejor respuesta en ambientes de grandes volúmenes, teniendo un mantenimiento y ajuste mucho menor que los productos de los proveedores comerciales. Conservando todas las características, estabilidad y rendimiento dando al traste con las necesidades que plantea el proyecto productivo Sistema de Gestión Fiscal II, dado por las características potenciales del gestor y su gratitud, además que los distintos procesos de negocio requieren el almacenamiento y procesamiento de grandes volúmenes de información así como de estabilidad en dichas operaciones.

### 1.10 ORM-Mapeo de Objeto Relacional

Propel es un grupo de Mapeo de Objeto Relacional (ORM) para PHP5. Le permite acceder a su base de datos utilizando un conjunto de objetos, que proporciona un interfaz sencilla para almacenar y recuperar datos dándole al desarrollador de aplicaciones web, las herramientas para trabajar con bases de datos de la misma manera se trabaja con otras clases y objetos en PHP, ya que le da a su base de datos una API bien definida utilizando los estándares de PHP5 OO - Excepciones, carga automática e iteradores. (26)

Propel utiliza PDO[14] como capa de abstracción, y la generación de código para eliminar la carga de la introspección en tiempo de ejecución. Por lo tanto Propel es

[12]Tipo de dato el cual almacena una secuencia de objetos definida por el desarrollador.
[13]Multi-Versión para el control de concurrencia.
[14] PDO, solo funciona con php5 y permite cadenas de conexión de drivers específicos.

rápido, implementa todos los conceptos clave de las capas ORM maduras: el patrón ActiveRecord[15], validadores, los comportamientos, herencia de tablas, la ingeniería inversa a una base de datos existente, conjuntos anidados, las transacciones anidadas y la carga diferida. Propel se ha creado para los desarrolladores que necesitan mantener el control de su código.

Propel facilita cambiar su sistema de gestión de base de datos en el curso del proyecto, ya que soporta los siguientes gestores de base de datos: MySQL, PostgreSQL, SQLite, MSSQL y Oracle.

### 1.11 Capa de Presentación o Vista

El manejo de la capa de presentación de un sistema informático constituye paso fundamental para la obtención de un producto eficiente y a la vez agradable para el usuario final es por ello que luego de hacer un análisis sobre las distintas herramientas dedicadas al manejo de la vista se selecciona Ajax con JQuery debido al buen manejo de las funciones JavaScript y html dinámico, permitie además la confección de vistas dinámicas que permitan intercambiar información con el servidor y la base de datos sin necesidad de recargar la página completa y nos facilitará el diseño de páginas haciendo uso de su vinculación con AJAX así como su facilidad de uso y su cualidad de ser liviana además de su portabilidad. (27)

### 1.12 Control de versiones

Contar con un respaldo del sistema informático propuesto en la investigación tanto como para recuperar información perdida así como para tener un control de las versiones del producto, constituye prioridad para el equipo de desarrollo. Luego de hacer un análisis de distintas herramientas dedicadas al control de versiones se selecciona Subversion[16] debido a las ventajas que brinda:

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- La creación de ramas y etiquetas es una operación más eficiente; Tiene costo de complejidad constante (O(1)) y no lineal (O(n)). (28)
- Se envían sólo las diferencias en ambas direcciones.

[15] ActiveRecord, facilita el entendimiento del código asociado a base de datos y encapsula la lógica especifica haciéndola más fácil de usar para el programador.

[16] Software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual posee varias deficiencias. Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como svn por ser ese el nombre de la herramienta utilizada en la línea de órdenes.

- Puede ser servido mediante Apache permitiendo que cualquier tipo de cliente utilice Subversion en forma transparente.
- Maneja eficientemente archivos binarios. (29)
- Permite selectivamente el bloqueo de archivos. Se usa en archivos binarios que, al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez.

Subversion es conocido en la comunidad de software libre y se utiliza en muchos proyectos, incluyendo la fundación del software de Apache constituyendo un precedente que contribuirá al equipo de desarrollo realizar un chequeo del sistema informático llevando el control de las versiones obtenidas así como obtener una versión del sistema en caso de pérdidas.

### 1.13 Prototipado

Los prototipos de interfaz no funcional proporcionan una vista cercana a cómo quedará implementada una determinada vista del sistema por lo que se requiere de una herramienta capaz de facilitar la obtención de prototipos meramente confiables. Haciendo un estudio se encuentra como herramienta a fin a las necesidades del equipo de desarrollo a **axure 5.5** ya que es una simple extensión de Firefox contando con una potencia y flexibilidad considerables. Permite crear proyectos con varias pantallas, en las cuales es posible añadir cualquier elemento de una librería típica (botones, tablas, cajas de activación y deslizadores), dando una idea exacta del resultado final, es sencillo y cómodo. (30)

### 1.14 Pruebas

Las pruebas son un paso importante en el proceso de culminación de un sistema informático ya que permiten aumentar la calidad de los mismos reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección.

La metodología XP divide las pruebas del sistema en dos grupos (Beck 1999) las pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final. (31)

#### 1.14.1 Pruebas Funcionales o de Aceptación

Las pruebas funcionales o de aceptación son la mejor forma de probar que se tiene un sistema informático con las características que desea el cliente, desde la petición realizada por el navegador hasta la respuesta enviada por el servidor. Las pruebas

funcionales prueban todas las capas del sistema informático dígase el sistema de enrutamiento, el modelo, las acciones y las plantillas. (31)

Las pruebas de aceptación son una parte integral del desarrollo incremental según lo practicado por la metodología XP. Todas las historias de usuario son apoyadas por las pruebas de aceptación, las cuales son definidas por el cliente en el sistema. Estas pruebas se realizan frente a los temores de que el negocio ha sido mal entendido por parte del equipo de desarrollo. Las pruebas de aceptación significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente. Las pruebas de aceptación exigen al cliente a profundizar en su conocimiento del dominio y, precisamente, afirmar lo que el sistema informático debe hacer en circunstancias específicas, por esto, el cliente es la persona adecuada para diseñar las pruebas de aceptación.

En la presente investigación se propone el desarrollo de las pruebas funcionales o de aceptación como otra alternativa de validación del sistema informático. El cliente hará un diseño de sus propias pruebas las cuales estarán acorde con las historias de usuarios propuestas a implementar al inicio del desarrollo del sistema informático, describiéndose las entradas y salidas para el sistema informático.

### 1.14.2 Pruebas unitarias

Las pruebas unitarias son las formas de probar el correcto funcionamiento en código de un sistema informático, facilitando que cada uno de los módulos que componen dicho sistema funcione correctamente por separado.

En la presente investigación se propone ir añadiendo pruebas unitarias para que se encuentren y solucionen errores en el sistema informático propuesto trayendo consigo que después de distintas iteraciones el código no sólo sea adecuado, sino que cada vez sea mayor el porcentaje del sistema que este cubierto por pruebas.

El principal problema de las librerías para crear pruebas es que son bastante difíciles de aprender a manejar. Por este motivo el marco de trabajo Symfony 1.3 incluye su propio marco de trabajo para pruebas llamado Lime ya que proporciona el soporte para las pruebas unitarias, es más eficiente que otros marcos de trabajo. (22)

El marco de trabajo Lime está orientado al desarrollo de pruebas PHP y tiene las siguientes ventajas:

- Ejecuta los archivos de prueba en un entorno independiente para evitar interferencias entre las diferentes pruebas. No todos los marcos de trabajo orientados a la realización pruebas garantizan un entorno de ejecución “limpio” para cada prueba.

- Las pruebas de Lime son fáciles de leer y sus resultados también lo son. En los sistemas operativos que lo soportan, los resultados de Lime utilizan diferentes colores para mostrar de forma clara la información más importante.
- Symfony 1.3 utiliza Lime para sus propias pruebas, por lo que el código fuente de Symfony 1.3 incluye muchos ejemplos reales de pruebas unitarias.
- El núcleo de Lime se valida mediante pruebas unitarias.

La solución informática incluirá estas pruebas haciendo uso de las siguientes métricas: **complejidad cliclomática:** Mide la complejidad estructural del código calculando el número de rutas de acceso del código diferente del flujo del programa ya que el programa que tenga un flujo de control complejo requerirá más pruebas para lograr una buena cobertura de código y será más difícil de mantener.

**tamaño operacional de la clase(TOC):** Está dada por el número de funcionalidades asignadas a cada una de las clases del sistema informático propuesto en la investigación evaluando los siguientes atributos de calidad:

- **responsabilidad:** Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
- **complejidad de implementación:** Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
- Reutilización: Un aumento del TOC implica una disminución del grado de reutilización de la clase.

## 1.15 Conclusiones Parciales

El estudio de las tendencias del desarrollo web actual, así como la justificación de las tecnologías que se adecuan a las necesidades del proyecto productivo SIGEF II ha sido el objetivo de este capítulo; de este estudio resultaron seleccionadas las siguientes herramientas y tecnologías:

- Metodología de desarrollo ágil XP.
- Herramienta CASE: Visual Paradigm for UML.
- Lenguaje de Modelado UML.
- Sistema Gestor de Base de Datos: Postgresql 8.4.
- Herramienta de prototipado: AxurePro 5.5.
- Herramienta para el control de versiones: Subversion 1.2
- ORM: Propel 1.6.

# 2 Capítulo 2: Planificación, diseño e implementación

El siguiente capítulo justifica la obtención de un conjunto de artefactos de la metodología XP en su aplicación al desarrollo del sistema informático para las fases: Planificación, Diseño y Desarrollo, donde se documentan los elementos imprescindibles para que se obtenga como resultado un producto con la calidad requerida por el usuario final, haciéndose muestra de estos en la Figura 2.

**Fases y artefactos generados en la metodología XP**

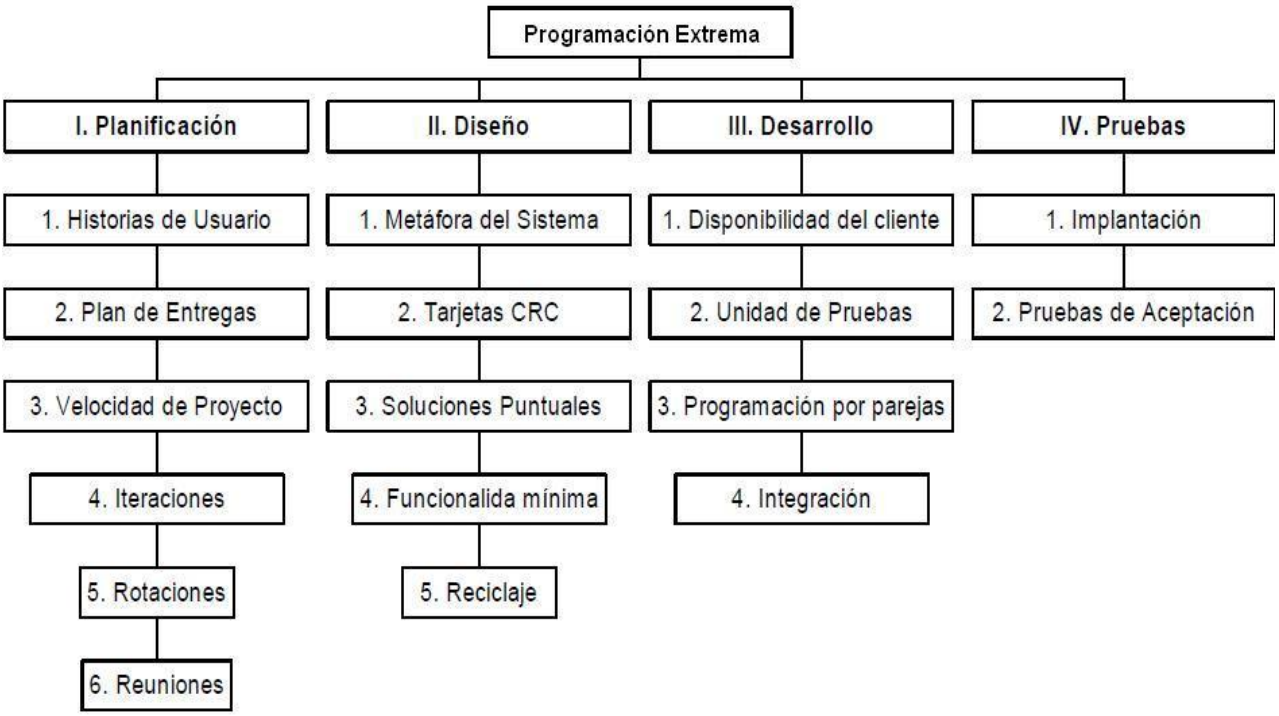



**Figura 2.** Fases de la metodología XP.

## 2.1 Planificación

La metodología de desarrollo de software XP muestra la planificación como un permanente diálogo entre el usuario final y el equipo de desarrollo; definiéndose lo que el software tiene que resolver para que genere algún valor, la parte del software que debe hacerse primero, lo que se necesita hacer para saber si el negocio requiere o no de la informatización a través de un software, fechas de entrega de las primeras

versiones del software, la organización del trabajo y el equipo de desarrollo y por último se detalla dentro de una versión del producto los problemas que deben de ser resueltos con carácter urgente. (31)

### 2.1.1 Historias de Usuario

Las historias de usuario son descritas por los propios clientes, tal y como ven ellos las necesidades del sistema. Por tanto serán descripciones cortas y escritas en el lenguaje del usuario, sin terminología técnica, las mismas conducen el proceso de creación de las pruebas de aceptación las cuales se utilizarán para verificar que las historias de usuario han sido implementadas correctamente, la principal diferencia es el nivel de detalle. El nivel de detalle de las historias de usuario debe ser el mínimo posible que permita hacerse una ligera idea de cuánto costará implementar el sistema, siendo los desarrolladores los que hacen una estimación de cuánto tiempo, les llevará implementar cada historia de usuario (31). Las historias de usuario proporcionaran los detalles sobre la estimación del riesgo y cuánto tiempo conllevará la implementación de la misma.

**Historias de Usuario detectadas:**

- Configuración básica de acceso
- Gestión de recurso
- Configuración avanzada de acceso
- Gestión de usuarios

Historia de usuario correspondiente a la “configuración básica de acceso” en cargada de gestionar el acceso hasta un nivel de acciones a ser ejecutadas en el sistema informático:

<table>
<tr><th colspan="3">Historia de Usuario</th></tr>
<tr><td>Número: 1</td><td colspan="2">Nombre Historias de usuario: Configuración básica de acceso.</td></tr>
<tr><td colspan="3">Modificación de Historia de Usuario Número:</td></tr>
<tr><td colspan="3">Referencia:</td></tr>
<tr><td colspan="2">Programador: Felipe Hernández Salazar</td><td>Iteración Asignada: 1</td></tr>
<tr><td colspan="2">Prioridad en Negocio: Alta<br>(Alta/Media/Baja)</td><td>Puntos Estimados(días): 15</td></tr>
<tr><td colspan="2">Riesgo en Desarrollo: Bajo<br>(Alto/Medio/Bajo)</td><td>Puntos Reales(días): 24</td></tr>
<tr><td colspan="3">Descripción:<br>Inicia cuando el Administrador del sistema desde el menú Herramientas pulsa la opción Configurar el acceso básico donde podrá establecer un nivel de acceso personalizado a cada parte del sistema en cuestión estableciendo a quien le será accesible una determinada aplicación así como a las acciones perteneciente a</td></tr>
</table>

| |
|---|
| cada uno de los módulos, se especificará si se debe de estar autenticado en el sistema para acceder a las partes que lo componen. |
| **Observaciones:** |

**Tabla 1.** Historia de usuario Configuración básica de acceso.

Historia de usuario correspondiente a la “configuración avanzada de acceso” encargada de gestionar el acceso hasta un nivel de objeto del domino:

| **Historia de Usuario** | |
|---|---|
| **Número:** 3 | **Nombre Historias de usuario:** Configuración avanzada de acceso. |
| **Modificación de Historia de Usuario Número:** | |
| **Referencia:** | |
| **Programador:** Felipe Hernández Salazar | **Iteración Asignada:** 3 |
| **Prioridad en Negocio:** Alta **(Alta/Media/Baja)** | **Puntos Estimados(días):** 21 |
| **Riesgo en Desarrollo:** Bajo **(Alto/Medio/Bajo)** | **Puntos Reales(días):** 31 |
| **Descripción:** Inicia cuando el Administrador del sistema desde el menú **Herramientas** pulsa la opción **Configurar el acceso avanzado** donde podrá especificar que privilegios poseen los usuarios del sistema sobre determinados objetos que el sistema hace persistente, lográndose separar de sus privilegios sobre un determinado objeto a los usuarios del sistema especificando el objeto persistente en cuestión para ambos casos. | |
| **Observaciones:** | |

**Tabla 2.** Historia de Usuario Configuración avanzada de acceso.

### 2.1.2 Requisitos

Una vez realizadas por el cliente las distintas historias de usuarios se procede a documentar las necesidades del mismo capturando los requisitos identificados a partir de las historias de usuario. Los requisitos son la base para un desarrollo exitoso así como para una plena conformidad con el entregable final. A continuación se muestra el listado de requisitos definidos para el sistema informático propuesto por la investigación.

#### 2.1.2.1 Requisitos funcionales

Los requisitos funcionales muestran las características requeridas por el sistema informático propuesto, que expresan una capacidad de acción del mismo, una funcionalidad o comportamiento interno; generalmente expresada en una declaración en forma verbal. (32)

A continuación se listan los requisitos funcionales detectados para el sistema

- Listar las aplicaciones del sistema.
- Listar módulos de una aplicación seleccionada.
- Listar dado un módulo sus acciones.
- Listar los recursos del sistema.
- Mostrar estado del recurso seleccionado.
- Registrar nuevo recurso.
- Guardar configuración básica.
- Listar identificadores dado un recurso.
- Listar usuarios que posean algún privilegio sobre el recurso seleccionado.
- Listar privilegios inferiores al que posee el usuario seleccionado que tiene algún privilegio sobre el recurso seleccionado.
- Listar usuarios que no posean ningún privilegio sobre el recurso seleccionado.
- Listar conjunto de privilegios a ser otorgados sobre el recurso seleccionado.
- Guardar configuración avanzada.

#### 2.1.2.2 Requisitos no funcionales

Los requisitos no funcionales definen propiedades del sistema informático que se desea como producto final. Estos requisitos son normalmente a los que debe apuntar la arquitectura y si estos no son cumplidos, el sistema informático propuesto puede no funcionar o el cliente simplemente no aceptar el producto. (32)

**apariencia o interfaz externa:**

- El sistema tiene que ofrecer una interfaz amigable, fácil de operar.
- El sistema tiene que mantener la línea de diseño establecida para la institución que mantiene la uniformidad y representatividad de la misma.
- Las interfaces tienen que ostentar un diseño sencillo, con pocas entradas, permitiendo un balance adecuado entre funcionalidad y simplicidad de tal manera que no se haga difícil para los usuarios la utilización del sistema.

**usabilidad:**

- El software tendrá siempre la posibilidad de ayuda disponible para cualquier tipo de usuario, lo que le permitirá un avance considerable en la explotación de la aplicación en todas sus funcionalidades.
- Existirán servidores locales con capacidad necesaria para el procesamiento de las solicitudes del conjunto de aplicaciones de las diferentes oficinas.
- Las aplicaciones siempre solicitarán los datos a través del servidor local.

- Desde cada servidor local se establecerá la conexión con servidores centrales para mantener la actualización de los datos en ambos sentidos.
- El tiempo de entrenamiento requerido para que usuarios normales y avanzados sean productivos operando el sistema es de 15 días.
- Tiene poseer una interfaz agradable para el cliente.

**soporte:**

- Tiene que brindar soporte para grandes volúmenes de datos y velocidad de procesamiento.
- El sistema tiene que ser multiplataforma.

**seguridad:**

- Tiene que brindar una protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.
- El sistema tiene mantener en todo momento la seguridad de la información asegurando la autenticidad de la misma.
- La seguridad se establecerá por roles que se le asignarán a los usuarios que interactúen con el sistema.
- El software brindará solamente aquellas funcionalidades que competen a la unidad ejecutora donde esté implantado.

**hardware:**

- Memoria RAM 256 MB o superior.
- Disco duro de 20 GB o superior.
- Procesador Intel® a 1 GHz. de velocidad de procesamiento o superior.
- Tarjeta de red Ethernet.
- El sistema tiene que interactuar con dispositivos de impresión (IMPRESORA HP 1018 LASERJET).

**software:**

- Debe tener el sistema operativo con entorno de escritorio gráfico, preferentemente GNU/Linux en cualquiera de sus versiones.

- Debe poseer un navegador web para acceder al sistema informático, preferentemente el Morzilla Firefox 3.6 o versión superior.

### 2.1.3 Plan de Iteración

Tomando como base cada una de las historias de usuarios y el esfuerzo que se requiere para el desarrollo de estas, se procede a fragmentar el trabajo en distintas iteraciones; obteniendo un trabajo incremental, donde la piedra angular es la comunicación entre el equipo de desarrollo y el cliente. El equipo de desarrollo precede a dividir la confección del sistema informático en 4 iteraciones; las cuales equilibrarán las distintas secuencias de trabajo.

A continuación se describen cada una de las iteraciones propuestas donde la duración total de iteraciones en días se obtiene a partir del esfuerzo en días estimado por el desarrollador para implementar cada historia de usuario:

| Iteración | Historias de Usuario | Duración total iteraciones (días) |
|---|---|---|
| Iteración 1 | Configuración básica de acceso. | 15 |
| Iteración 2 | Gestión de recurso. | 7 |
| Iteración 3 | Configuración avanzada de acceso. | 21 |
| Iteración 4 | Gestión de usuarios. | 15 |

**Tabla 3.** Plan de Iteración

### 2.1.4 Plan de Entrega

Ya elaboradas por el cliente las distintas historias de usuarios y confeccionado por los desarrolladores el plan de iteración se procede a la confección del plan de entrega con la intención de que los mismos obtengan una estimación real en cuanto a nivel de detalle se refiere, fijándose un periodo de tiempo sobre el cual se debe implementar cada historia de usuario definiéndose el grado de dificultad de la misma. El Plan de Entrega posibilita la obtención de una clasificación teniendo en cuenta el riesgo que se corre a la hora de implementar la historia. Estos datos se almacenan en campos que permanecen vacíos en la historia de usuario, el responsable de llenar estos datos es únicamente el programador una vez que haya hecho el análisis requerido de los mismos.

A continuación se presenta el plan de entrega elaborado por el equipo de desarrollo donde se reflejan las fechas de entregas para las primeras versiones de las historias de usuario:

| Artefacto | Hito | Entrega |
|---|---|---|
| Configuración básica de acceso. | Final de la primera iteración. | 2/02/2012 |
| Gestión de recurso. | Final de la segunda iteración. | 16/02/2012 |
| Configuración avanzada de acceso. | Final de la tercera iteración. | 20/03/2012 |
| Gestión de usuarios. | Final de la cuarta iteración. | 12/04/2012 |

**Tabla 4.** Plan de entrega.

## 2.2 Diseño

En la metodología de desarrollo de software XP se aprecia una simplicidad en el diseño, pues esta permite elegir una "metáfora" de sistema, un conjunto de nombres ilustrados de la realidad, al usar tarjetas CRC. Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. La complejidad innecesaria y el código extra debe ser removido inmediatamente. (31)

### 2.2.1 Patrones

Los patrones son una disciplina de resolución de problemas reciente en la ingeniería del software que ha emergido en mayor medida de la comunidad de orientación a objetos, aunque pueden ser aplicados en cualquier ámbito de la informática y las ciencias en general. Los patrones tienen raíces en muchas áreas y constituyen un modelo que se puede seguir para realizar algo, surgen de la experiencia de seres humanos al tratar de lograr ciertos objetivos. Los patrones capturan la experiencia existente y probada para promover buenas prácticas.

A continuación se muestran los distintos patrones empleados en el desarrollo del sistema informático propuesto.

#### 2.2.1.1 Patrón Aquitectónico

Un patrón arquitectónico de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución.

A continuación se presenta el patrón arquitectónico utilizado para la realización del sistema informático.

El patrón arquitectónico modelo vista controlador (MVC) que incluye el marco de trabajo seleccionado para el desarrollo del sistema informático propuesto en la investigación, separa la lógica de negocio y la presentación está formado por tres capas:

- El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio, el marco de trabajo Symfony 1.3 almacena todos los ficheros y clases vinculadas al modelo de datos en el directorio model**/** el cual se encuentra dentro del directorio lib/ del sistema informático.
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella. En Symfony 1.3 las plantillas php se alojan en el directorio template**/** de cada módulo y aplicación.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. Cada petición es procesada por un controlador frontal los cuales demandan dichas labores a las acciones las cuales se encuentran agrupadas en módulos.

### 2.2.2 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Identifican: Clases, Instancias, Roles, Colaboraciones y la distribución de responsabilidades.

Los patrones de diseño utilizados para el desarrollo del sistema informático son los que se presentan a continuación.

#### 2.2.2.1 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. (33)

**creador**: La aplicación que maneja la seguridad del sistema informático (SEG**)** está formada por un módulo que contiene una clase *actions.class.php* donde se alojan las distintas acciones que permite dicho subsistema, siendo posible en dichas acciones la creación de los objetos de las clases persistentes que se almacenan en la base de datos siendo la clase *actions.class.php* la creadora de dichos objetos.

**experto**: Se usa el marco de trabajo para el mapeo [17]de objetos relacionales propel haciéndose notable este patrón el cual encapsula toda la lógica de los datos generando clases funcionales análogas a la entidades de la base de datos. Siendo las

[17] El mapeo objeto-relacional (más conocido por su nombre en inglés, Object-Relational mapping, o sus siglas O/RM, ORM, y O/R mapping) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia

mismas expertas en manejar la información pertinente a la entidad a la cual representan.

**controlador**: Los pedidos que se realizan al subsistema de seguridad SEG son manejados por su controlador frontal SEG.php constituyendo un único punto de entrada a la aplicación pues cada pedido es chequeado con el sistema de enrutamiento del subsistema construyendo una url confiable.

**alta Cohesión**: el marco de trabajo Symfony 1.3 asigna las responsabilidades posibilitando una alta cohesión pues cada clase *actions.class.php* define las acciones sobre cada una de las plantillas php y tributa a otras creando objetos y realizando operaciones.Luego sus funcionalidades se encuentran estrechamente relacionadas lo cual garantiza la flexibilidad del sistema informático ante cambios eminentes.

**bajo acoplamiento**: En el sistema informático propuesto solo existe un nivel de herencia en la jerarquía propuesta así como la herencia propuesta por el marco de trabajo utilizado es limitada haciendo uso de un bajo acoplamiento pues cada clase *actions.class.php* hereda solamente de *sfActions* logrando un bajo acoplamiento entre clases.

#### 2.2.2.2 Patrones GoF

Los patrones de diseño GoF se caracterizan por indicar resoluciones técnicas basadas en Programación Orientada a Objetos (POO). En ocasiones tienen más utilidad con algunos lenguajes de programación y en otras son aplicables a cualquier lenguaje, además favorecen la reutilización de código pues ayudan a construir un software basado en la reutilización.(34)

**creacionales**: Instancia Única (en el inglés Singleton): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia, siendo así con el objeto usuario el cual se accede desde el método getUser() desde el controlador, no siendo así en las platillas en donde se obtiene a partir del objeto sf_user siendo la salida de ambos el mismo objeto del usuario garantizándose que una vez que el usuario inicie sección se acceda a sus datos desde una única instancia.

**estructurales**: Decorador (en el inglés Decorator): Añade funcionalidad a una clase, dinámicamente. El archivo *layout.php*, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el Layout decorando la misma.

### 2.2.3 Tarjetas CRC (Cargo o Clase, Responsabilidad y Colaboración)

Las tarjetas CRC permiten desprenderse del método basado en procedimientos y trabajar con una metodología basada en objetos, así el programador se centra y comienza a apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación clásica. Las tarjetas CRC representan objetos; la clase a la que pertenece el objeto se escribe en la parte de arriba de la tarjeta, a modo de título, en una columna a la izquierda se escriben las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad. (31)

A continuación se muestran las clases de diseño identificadas por el equipo de desarrollo así como las tarjetas CRC correspondientes a cada una de ellas.

Clases:

- ✓ Tbsegacl.
- ✓ Tbsegidentidadobjeto.
- ✓ Tbpersona.
- ✓ Nsegrecurso.
- ✓ Nsegpermisoobjeto.
- ✓ Initseg.
- ✓ SegfiltroPDC.
- ✓ SegfiltroCLEP.
- ✓ Parseoacciones.

A continuación se muestra la información correspondiente a la tarjeta CRC correspondiente a la clase Tbsegidentidadobjeto que por su significación desde el punto de vista del negocio se decide mostrar estando conformada por dos funcionalidades relacionadas con la clase Nsegrecurso:

**Clase: Tbsegidentidadobjeto**

| Responsabilidad | Clases relacionadas |
|---|---|
| Registrar una identidad para cada recurso del sistema. | • Nsegrecurso |
| Obtener una identidad de un objeto dado. | • Nsegrecurso |

**Tabla 5.** Tarjeta CRC: Tbsegidentidadobjeto.

A continuación se muestra la información correspondiente a la tarjeta CRC correspondiente a la clase Tbsegacl que por su significación desde el punto de vista

del negocio se decide mostrar estando conformada por tres funcionalidades relacionadas, las cuales se relacionan con las clases Tbseidentidadobjeto,Tbpersona y Nsegpermisoobjeto:

**Clase: Tbsegacl**

| Responsabilidad | Clases relacionadas |
|---|---|
| Registrar privilegios que posee un usuario sobre un determinado recurso del sistema. | • Tbsegidentidadobjeto.<br>• Nsegpermisoobjeto.<br>• Tbpersona. |
| Obtener privilegios que posee un usuario sobre un determinado recurso del sistema. | • Tbsegidentidadobjeto.<br>• Nsegpermisoobjeto.<br>• Tbpersona. |
| Separar privilegios que posee un usuario sobre un recurso del sistema. | • Tbsegidentidadobjeto.<br>• Nsegpermisoobjeto.<br>• Tbpersona. |

**Tabla 6.** Tarjeta CRC: Tbsegacl.

A continuación se muestra la información correspondiente a la tarjeta CRC correspondiente a la clase SegfiltroPDC que por su significación desde el punto de vista del negocio se decide mostrar estando conformada por tres futuras funcionalidades relacionadas con la clase Tbsegacl:

**Clase: SegfiltroPDC**

| Responsabilidad | Clases relacionadas |
|---|---|
| Comprobar si el usuario autenticado en el sistema posee privilegios necesarios para ver detalles del rollo laborar solicitado | • Tbsegacl |
| Comprobar si el usuario autenticado en el sistema posee privilegios necesarios para actualizar el rollo laborar solicitado. | • Tbsegacl |
| Hacer propietario al usuario autenticado en el sistema que adicione un nuevo rollo laboral. | • Tbsegacl |

**Tabla 7.** Tarjeta CRC: SegfiltroPDC

A continuación se muestra la información correspondiente a la tarjeta CRC correspondiente a la clase Initsecurity que por su significación desde el punto de vista del negocio se decide mostrar encargada de informatizar el acceso hasta un nivel de acciones conformada por seis funcionalidades relacionadas las cuales no cuentan con dependencias de clases externas:

**Clase: InitSecurity**

| Responsabilidad | Clases relacionadas |
|---|---|
| Listar aplicaciones del sistema. | - |
| Crear fichero security.yml en el directorio config de la aplicación seleccionada. | - |
| Listar módulos dada una aplicación. | - |
| Crear fichero security.yml en el directorio config de un módulo dado. | - |
| Crear si no existe el directorio config dentro del módulo seleccionado. | - |
| Escribir en el fichero security.yml la configuración seleccionada. | - |

**Tabla 8.** Tarjeta CRC: InitSecurity

A continuación se muestra la información correspondiente a la tarjeta CRC correspondiente a la clase parseoacciones que por su significación desde el punto de vista del negocio se decide mostrar, la misma es encargada de obtener las acciones correspondientes a un módulo dado conformada por una funcionalidad la cual no tiene ninguna dependencia con clase externa:

**Clase: Parseoacciones**

| | Clases relacionadas |
|---|---|
| Dado un módulo listar el conjunto de acciones que posee | - |

**Tabla 9.** Tarjeta CRC: Parseoacciones

A continuación se muestra la información correspondiente a la tarjeta CRC correspondiente a la clase SegfiltroCLEP que por su significación desde el punto de vista del negocio se decide mostrar, la cual es encargada de filtrar las peticiones del

cliente controlando así el acceso al subsistema CLEP, se encuentra conformada por tres futuras funcionalidades relacionadas con la clase Tbsegacl:

**Clase: SegfiltroCLEP**

| Responsabilidad | Clases relacionadas |
|---|---|
| Comprobar si el usuario autenticado en el sistema posee privilegios necesarios para ver detalles del dictamen solicitado | • Tbsegacl |
| Comprobar si el usuario autenticado en el sistema posee privilegios necesarios para actualizar el dictamen solicitado. | • Tbsegacl |
| Hacer propietario al usuario autenticado en el sistema que adicione un nuevo dictamen. | • Tbsegacl |

**Tabla 10.** Tarjeta CRC: SegfiltroCLEP

A continuación se muestra la información correspondiente a la tarjeta CRC correspondiente a la clase Nsegpermisoobjeto que por su significación desde el punto de vista del negocio se decide mostrar, la cual es conformada por tres funcionalidades que no poseen dependencias de clases externas:

**Clase: Nsegpermisoobjeto**

| Responsabilidad | Clases relacionadas |
|---|---|
| Obtener un permiso dado su identificador. | **-** |
| Obtener un permiso dado su nombre. | **-** |
| Obtener todos los permisos disponibles en el sistema. | **-** |

**Tabla 11.** Tarjeta CRC: Nsegpermisoobjeto

Tarjeta CRC correspondiente a la clase Nsegrecurso conformada por tres futuras funcionalidades que no poseen dependencias de clases externas:

**Clase: Nsegrecurso**

| Responsabilidad | Clases relacionadas |
|---|---|
| Obtener un recurso dado su identificador. | **-** |
| Obtener todos los recursos del sistema. | **-** |
| Obtener recurso seleccionado. | **-** |

**Tabla 12.** Tarjeta CRC: Nsegrecurso

A continuación se muestra la información correspondiente a la tarjeta CRC correspondiente a la clase Tbpersona que por su significación desde el punto de vista del negocio se decide mostrar estando conformada por tres funcionalidades que no poseen dependencias de clases externas:

**Clase: Tbpersona**

| Responsabilidad | Clases relacionadas |
|---|---|
| Obtener una persona dado su identificador. | - |
| Obtener todas las personas del sistema. | - |
| Obtener recurso seleccionado. | - |

**Tabla 13.** Tarjeta CRC: Tbpersona

### 2.2.4 Diseño de base de datos

La creación del modelo de datos constituye prioridad en la abstracción para la creación de la base de datos siendo posible esto a través de un grupo de herramientas las cuales describen los datos y las relaciones que existen entre ellos. Haciendo uso de diagrama entidad relación de la herramienta CASE Visual Paradigm for UML se realizó el siguiente modelo de datos:

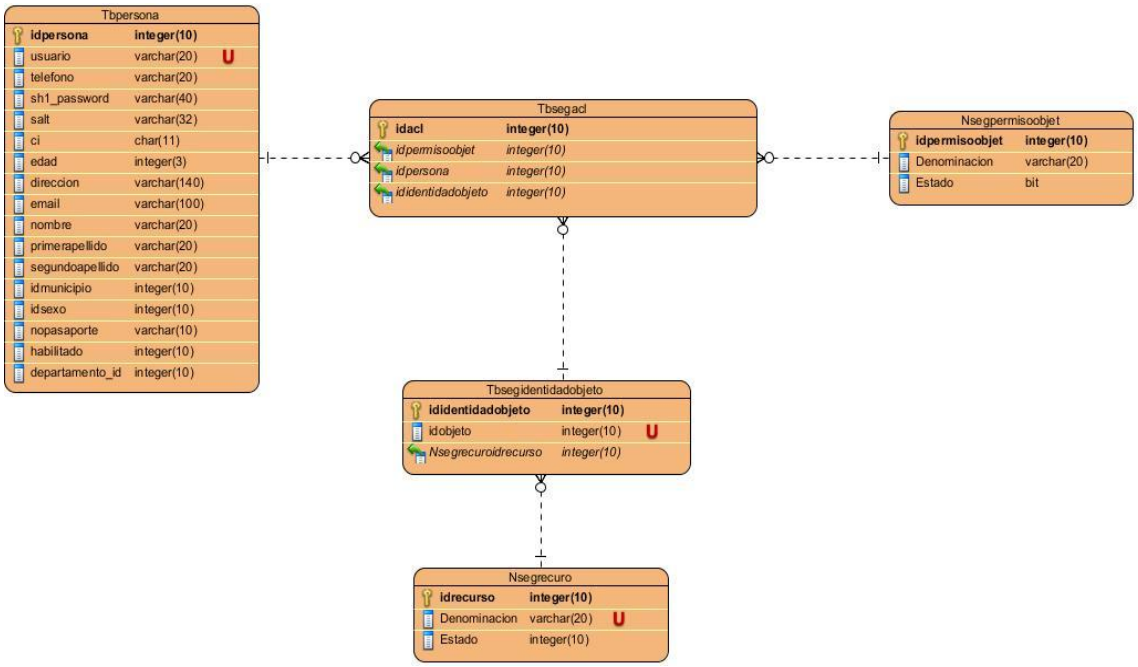



**Figura 3.** Diagrama de la base de datos.

El modelo de datos se encuentra compuesto por un conjunto de tablas que facilitarán conocer los permisos exactos que posee un usuario sobre un determinado objeto del sistema definiéndose en primer lugar los nomencladores[18] nombrados Nsegpermisos donde se registrarán los posibles permisos que puede tener un usuario del sistema el cual incluye un conjunto de operaciones a ser ejecutadas. La tabla Nsegrecurso facilita el registro del nombre de los recursos del sistema, mientra que la tabla Tbsegidentidadobjeto es encargada de llevar el control de las identidades de los objetos que están siendo asegurados. La tabla Tbsegacl es encargada de registrar los permisos que posee un usuario determinado sobre un objeto del sistema. Por último se tiene a la tabla Tbpersona la cual contiene toda la información referente a los usuarios del sistema.

## 2.3 Desarrollo

El proceso de desarrollo de software constituye la fase fundamental de la metodología XP pues se obtiene una primera versión del producto deseado por el cliente, realizándose bajo el siguiente ciclo de vida:

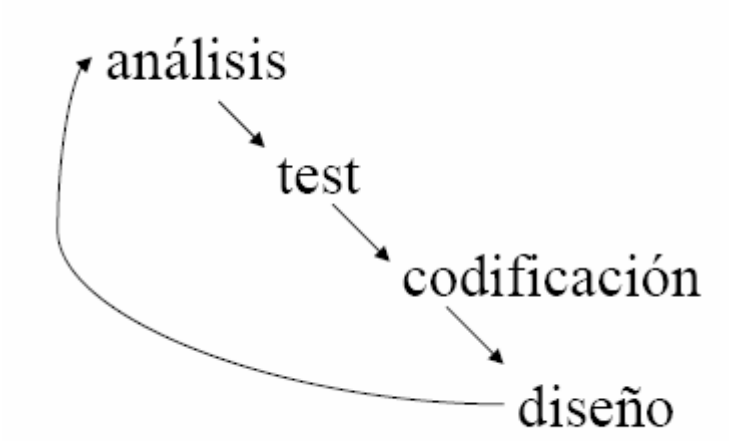



**Figura 4.** Ciclo de vida de XP para la fase de desarrollo

### 2.3.1 Tareas de ingeniería

Las historias de usuarios están divididas en distintas tareas, las cuales serán implementadas por los desarrolladores, dentro del equipo de desarrollo; aplicando la práctica de la programación en parejas, cada tarea es detallada especificando la

[18] Forma de clasificar o agrupar diversas prácticas.

historia de usuario relacionada, programador asignado, tiempo estimado, una breve descripción así como la fase en que se encuentra.

A continuación se muestran las tareas asignadas por cada historia de usuario. Por su significación a los efectos del negocio se detallan aquellas tareas asociadas a las historias de usuario “configuración básica de acceso” y “configuración avanzada de acceso”.

**Distribución de tareas por cada historia de usuario.**

| Historia de Usuario | Tareas |
|---|---|
| Configuración básica de acceso | ✓ Crear el diseño del prototipo.<br>✓ Crear interfaz de configuración básica.<br>✓ Garantizar que seleccionada una aplicación se muestre los módulos correspondientes a la misma.<br>✓ Garantizar que seleccionado un módulo se visualicen todas las acciones pertinentes al mismo.<br>✓ Guardarla configuración seleccionada.<br>✓ Validar que la entrada que se somete a guardar sea correcta. |
| Gestionar recursos | ✓ Crear el diseño del prototipo.<br>✓ Crear interfaz para la gestión de los recursos del sistema.<br>✓ Garantizar que seleccionado un recurso se muestre su estado actual.<br>✓ Crear campos dinámicos para insertar nuevos recursos al pulsar adicionar.<br>✓ Guardar los recursos introducidos.<br>✓ Validar que la entrada que se somete a guardar sea correcta. |
| Configuración avanzada de acceso | ✓ Crear el diseño del prototipo.<br>✓ Crear interfaz para la configuración avanzada de acceso al sistema.<br>✓ Garantizar que seleccionado un recurso del sistema se visualicen |

| | |
|---|---|
| | todos los identificadores de dicho recurso.<br>✓ Garantizar que seleccionado un recurso y un identificador se visualicen todos los usuarios que poseen acceso al mismo<br>✓ Garantizar que seleccionado un usuario con algún privilegio sobre el recurso seleccionado se listen todos los permisos inferiores al que posee.<br>✓ Garantizar que se muestren todos los usuarios que no posean ningún privilegio sobre el recurso seleccionado.<br>✓ Guardar la configuración establecida.<br>✓ Validar que la entrada que se somete a guardar sea correcta. |
| Gestión de usuarios. | ✓ Crear el diseño del prototipo.<br>✓ Crear interfaz para la gestión de usuario.<br>✓ Mostrar por cada recurso sobre el cual el usuario autenticado en el sistema es propietario, el nombre completo del usuario, el nombre del recurso, el id del recurso y el permiso que posee dicho usuario sobre este recurso.<br>✓ Dado un recurso en específico mostrar datos pertinentes tales como: el nombre completo del usuario, el nombre del recurso, el id del recurso y el permiso que posee dicho usuario sobre este recurso.<br>✓ Permitir exportar en un documento en formato pdf.<br>✓ Permitir imprimir mediante una vista previa el listado obtenido. |

**Tabla 14.** Distribución de tareas por cada historia de usuario.

### 2.3.2 Tareas detalladas

**Configuración básica de acceso**

| Tarea de ingeniería | |
|---|---|
| **Número de tarea:** 1 | **Nombre Historias de usuario:** Configuración básica de acceso. |
| **Nombre de la tarea:** Crear el diseño del prototipo | |
| **Tipo de tarea:** Desarrollo **(Desarrollo/Corrección/Mejora)** | **Puntos Estimados(días):** 1 |
| **Fecha inicio:** 9/01/2012 | **Fecha fin:** 10/01/2012 |
| **Programador responsable:** Felipe Hernández Salazar. | |
| **Descripción:** Esta tarea da una visión exacta de cómo será representada la interfaz de configuración básica de acceso facilitando la comprensión a la pareja de desarrolladores. | |

**Tabla 15.** Descripción de la tarea de ingeniería #1.

| Tarea de ingeniería | |
|---|---|
| **Número de tarea:** 2 | **Nombre Historias de usuario:** Configuración básica de acceso. |
| **Nombre de la tarea:** Crear interfaz de configuración básica. | |
| **Tipo de tarea:** Desarrollo **(Desarrollo/Corrección/Mejora)** | **Puntos Estimados(días):** 2 |
| **Fecha inicio:** 11/01/2012 | **Fecha fin:** 13/01/2012 |
| **Programador responsable:** Felipe Hernández Salazar. | |
| **Descripción:** Se plasma mediante el código los elementos necesarios para que se realice la selecciona de la configuración propuesta por el Administrador del sistema. | |

**Tabla 16.** Descripción de la tarea de ingeniería #2.

| Tarea de ingeniería | |
|---|---|
| **Número de tarea:** 3 | **Nombre Historias de usuario:** Configuración básica de acceso. |
| **Nombre de la tarea:** Garantizar que seleccionada una aplicación se muestre los módulos correspondientes a la misma. | |
| **Tipo de tarea:** Desarrollo **(Desarrollo/Corrección/Mejora)** | **Puntos Estimados(días):** 3 |
| **Fecha inicio:** 16/01/2012 | **Fecha fin:** 19/01/2012 |
| **Programador responsable:** Felipe Hernández Salazar. | |
| **Descripción:** Una vez que el administrador del sistema seleccione una aplicación aparecerán el listado de todos los módulos asociados a la misma previendo que la configuración sea correcta. | |

**Tabla 17.** Descripción de la tarea de ingeniería #3.

| Tarea de ingeniería |
|---|

| **Número de tarea:** 4 | **Nombre Historias de usuario:** Configuración básica de acceso. |
|---|---|
| **Nombre de la tarea:** Garantizar que seleccionado un módulo se visualicen todas las acciones pertinentes al mismo. | |
| **Tipo de tarea:** Desarrollo **(Desarrollo/Corrección/Mejora)** | **Puntos Estimados(días):** 3 |
| **Fecha inicio:** 20/01/2012 | **Fecha fin:** 25/01/2012 |
| **Programador responsable:** Felipe Hernández Salazar. | |
| **Descripción:** Una vez que el administrador del sistema seleccione un módulo aparecerán el listado de todos las acciones asociados a la misma previendo que la configuración sea correcta. | |

**Tabla 18.** Descripción de la tarea de ingeniería #4.

| **Tarea de ingeniería** | |
|---|---|
| **Número de tarea:** 5 | **Nombre Historias de usuario:** Configuración básica de acceso. |
| **Nombre de la tarea:** Validar que la entrada que se somete a guardar sea correcta. | |
| **Tipo de tarea:** Desarrollo **(Desarrollo/Corrección/Mejora)** | **Puntos Estimados(días):** 3 |
| **Fecha inicio:** 26/01/2012 | **Fecha fin:** 28/01/2012 |
| **Programador responsable:** Felipe Hernández Salazar. | |
| **Descripción:** Una vez que el administrador del sistema selecciona su configuración los datos seleccionados se chequean que estén correctos para ser sometidos. | |

**Tabla 19.** Descripción de la tarea de ingeniería #5.

| **Tarea de ingeniería** | |
|---|---|
| **Número de tarea:** 6 | **Nombre Historias de usuario:** Configuración básica de acceso. |
| **Nombre de la tarea:** Guardarla configuración seleccionada. | |
| **Tipo de tarea:** Desarrollo **(Desarrollo/Corrección/Mejora)** | **Puntos Estimados(días):** 3 |
| **Fecha inicio:** 30/01/2012 | **Fecha fin:** 2/02/2012 |
| **Programador responsable:** Felipe Hernández Salazar. | |
| **Descripción:** Los datos de configuración enviados son guardados por el sistema. | |

**Tabla 20.** Descripción de la tarea de ingeniería #6.

**Configuración avanzada de acceso**

| **Tarea de ingeniería** |
|---|

| **Número de tarea:** 1 | **Nombre Historias de usuario:** Configuración avanzada de acceso |
|---|---|
| **Nombre de la tarea:** Crear el diseño del prototipo. | |
| **Tipo de tarea:** Desarrollo **(Desarrollo/Corrección/Mejora)** | **Puntos Estimados(días):** 1 |
| **Fecha inicio:** 17/02/2012 | **Fecha fin:** 18/02/2012 |
| **Programador responsable:** Felipe Hernández Salazar. | |
| **Descripción:** Esta tarea da una visión exacta de cómo será representada la interfaz de configuración avanzada de acceso facilitando la comprensión a la pareja de desarrolladores. | |

**Tabla 21.** Descripción de la tarea de ingeniería #1.

| **Tarea de ingeniería** | |
|---|---|
| **Número de tarea:** 2 | **Nombre Historias de usuario:** Configuración avanzada de acceso |
| **Nombre de la tarea:** Crear interfaz para la configuración avanzada de acceso al sistema. | |
| **Tipo de tarea:** Desarrollo **(Desarrollo/Corrección/Mejora)** | **Puntos Estimados(días):** 2 |
| **Fecha inicio:** 20/02/2012 | **Fecha fin:** 22/02/2012 |
| **Programador responsable:** Felipe Hernández Salazar. | |
| **Descripción:** Se plasma mediante el código los elementos necesarios para que se realice la configuración avanzada emitida por el Administrador del sistema. | |

**Tabla 22.** Descripción de la tarea de ingeniería #2.

| **Tarea de ingeniería** | |
|---|---|
| **Número de tarea:** 3 | **Nombre Historias de usuario:** Configuración avanzada de acceso |
| **Nombre de la tarea:** Garantizar que seleccionado un recurso del sistema se visualicen todos los identificadores de dicho recurso | |
| **Tipo de tarea:** Desarrollo **(Desarrollo/Corrección/Mejora)** | **Puntos Estimados(días):** 2 |
| **Fecha inicio:** 23/02/2012 | **Fecha fin:** 25/02/2012 |
| **Programador responsable:** Felipe Hernández Salazar. | |
| **Descripción:** Una vez seleccionado un recurso se muestra un listado de todos los identificadores asociados al recurso seleccionado. | |

**Tabla 23.** Descripción de la tarea de ingeniería #3.

| **Tarea de ingeniería** | |
|---|---|
| **Número de tarea:** 4 | **Nombre Historias de usuario:** Configuración avanzada de acceso |

| | |
|---|---|
| **Nombre de la tarea:** Garantizar que seleccionado un recurso y un identificador se visualicen todos los usuarios que poseen acceso al mismo. | |
| **Tipo de tarea:** Desarrollo **(Desarrollo/Corrección/Mejora)** | **Puntos Estimados(días):** 5 |
| **Fecha inicio:** 27/02/2012 | **Fecha fin:** 2/02/2012 |
| **Programador responsable:** Felipe Hernández Salazar. | |
| **Descripción:** Dada la selección de un recurso y el identificador pertinente del mismo se visualiza un listado de usuarios que poseen algún privilegio sobre dicho recurso. | |

**Tabla 24.** Descripción de la tarea de ingeniería #4.

| **Tarea de ingeniería** | |
|---|---|
| **Número de tarea:** 5 | **Nombre Historias de usuario:** Configuración avanzada de acceso |
| **Nombre de la tarea:** Garantizar que seleccionado un usuario con algún privilegio sobre el recurso seleccionado se listen todos los permisos inferiores al que posee. | |
| **Tipo de tarea:** Desarrollo **(Desarrollo/Corrección/Mejora)** | **Puntos Estimados(días):** 5 |
| **Fecha inicio:** 5/03/2012 | **Fecha fin:** 10/03/2012 |
| **Programador responsable:** Felipe Hernández Salazar. | |
| **Descripción:** Una vez seleccionado un usuario el cual posee un privilegio sobre el recurso seleccionado se lista los privilegios inferiores al que posee dicho usuario sobre dicho recurso con el fin de separar privilegios. | |

**Tabla 25.** Descripción de la tarea de ingeniería #5.

| **Tarea de ingeniería** | |
|---|---|
| **Número de tarea:** 6 | **Nombre Historias de usuario:** Configuración avanzada de acceso |
| **Nombre de la tarea:** Garantizar que se muestren todos los usuarios que no posean ningún privilegio sobre el recurso seleccionado. | |
| **Tipo de tarea:** Desarrollo **(Desarrollo/Corrección/Mejora)** | **Puntos Estimados(días):** 2 |
| **Fecha inicio:** 12/03/2012 | **Fecha fin:** 14/03/2012 |
| **Programador responsable:** Felipe Hernández Salazar. | |
| **Descripción:** Una vez seleccionado un recurso con su identificador se muestra un listado de todos los usuarios que no poseen ningún privilegio sobre el recurso seleccionado con el fin de conceder privilegio. | |

**Tabla 26.** Descripción de la tarea de ingeniería #6.

| **Tarea de ingeniería** | |
|---|---|
| **Número de tarea:** 7 | **Nombre Historias de usuario:** Configuración avanzada de acceso |
| **Nombre de la tarea:** Validar que la entrada que se somete a guardar sea correcta. | |

| | |
|---|---|
| **Tipo de tarea:** Desarrollo **(Desarrollo/Corrección/Mejora)** | **Puntos Estimados(días):** 2 |
| **Fecha inicio:** 15/03/2012 | **Fecha fin:** 17/03/2012 |
| **Programador responsable:** Felipe Hernández Salazar. | |
| **Descripción:** Se comprueba que los datos de entradas se encuentren correctos para ser sometidos. | |

**Tabla 27.** Descripción de la tarea de ingeniería #7.

| **Tarea de ingeniería** | |
|---|---|
| **Número de tarea:** 8 | **Nombre Historias de usuario:** Configuración avanzada de acceso |
| **Nombre de la tarea:** Guardar configuración establecida. | |
| **Tipo de tarea:** Desarrollo **(Desarrollo/Corrección/Mejora)** | **Puntos Estimados(días):** 1 |
| **Fecha inicio:** 19/03/2012 | **Fecha fin:** 20/03/2012 |
| **Programador responsable:** Felipe Hernández Salazar. | |
| **Descripción:** Los datos de configuración enviados son guardados por el sistema. | |

**Tabla 28.** Descripción de la tarea de ingeniería #8.

### 2.3.3 Desarrollo en pareja

En la metodología XP se propone un desarrollo en pareja. Esta estrategia o estilo de programación tiene muchas ventajas: muchos errores son detectados conforme son introducidos en el código; por lo que la cantidad de errores del sistema informático final suelen ser menos, los diseños son mejores y el tamaño del código menor, los problemas de la implementación se resuelven en menos tiempo, se posibilita la transferencia de conocimientos entre los miembros del equipo; pues cada uno entiende las diferentes partes del sistema y por último los programadores o desarrolladores disfrutan más su trabajo. (31)

## 2.4 Conclusiones Parciales

En el presente capítulo se abordaron los elementos fundamentales correspondientes a las fases de: planificación, diseño e implementación del sistema informático que se propone, basándose en lo que propone la metodología XP, aplicando las técnicas propuestas por esta metodología para cada una de las fases, se obtuvieron un grupo de artefactos necesarios para la implementación del sistema informático propuesto siendo los siguientes:

- Historias de usuario.

- Plan de iteración.
- Plan de entrega.
- Tarjetas CRC.
- Modelo de datos.
- Tareas de ingeniería.

# 3 Capítulo 3: Validación de los resultados

Uno de los pilares de la Programación Extrema es el proceso de pruebas. XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones dividiendo dichas pruebas en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final. (31)

## 3.1 Pruebas Funcionales o de Aceptación

A continuación se muestran los casos de pruebas diseñados a las historias de usuario "Configuración básica de acceso y Configuración avanzada de acceso" debido a que las mismas incluyen el conjunto de funcionalidades críticas para el desarrollo exitoso del sistema informático propuesto en la investigación.

Caso de prueba correspondiente a la funcionalidad "configuración del acceso a la aplicación" encargada de especificar el rol necesario para acceder a la aplicación así como si se tiene que estar autenticado para realizar dicho acceso:

| **Caso de prueba de aceptación** | |
|---|---|
| **Código:** HU1_P1 | **Historia de Usuario:** 1 |
| **Nombre:** Configurar el acceso a la aplicación. | |
| **Descripción:** Prueba para la funcionalidad de configurar el acceso a la aplicación. | |
| **Condiciones de Ejecución:**<br>• El usuario debe seleccionar una aplicación del sistema.<br>• El usuario debe seleccionar una credencial para acceder a dicha aplicación.<br>• De forma opcional señala si es seguro o no el acceso a dicha aplicación. | |
| **Resultados esperados:** El sistema guarda la configuración seleccionada y muestra | |

| |
|---|
| un mensaje informando que la operación se ha realizado con éxito, indicando además la dirección del archivo editado. |
| **Evaluación de la prueba:** Prueba satisfactoria. |

**Tabla 29.** Configurar el acceso a la aplicación.

Caso de prueba correspondiente a la funcionalidad “configuración del acceso a la acción” encargada de especificar el rol necesario para acceder a la misma así como si se tiene que estar autenticado para realizar dicho acceso:

| **Caso de prueba de aceptación** | |
|---|---|
| **Código:** HU1_P2 | **Historia de Usuario:** 1 |
| **Nombre:** Configurar el acceso a la acción. | |
| **Descripción:** Prueba para la funcionalidad de configurar el acceso a la acción. | |
| **Condiciones de Ejecución:**<br>• El usuario debe seleccionar una aplicación del sistema.<br>• El usuario debe seleccionar una credencial para acceder a dicha aplicación.<br>• De forma opcional señala si es seguro o no el acceso a dicha aplicación.<br>• El usuario debe seleccionar un módulo el cual pertenece a la aplicación seleccionada.<br>• El usuario debe seleccionar una acción correspondiente al módulo señalado.<br>• El usuario debe seleccionar una credencial para acceder a dicha acción.<br>• De forma opcional señala si es seguro o no el acceso a dicha acción. | |
| **Resultados esperados:** El sistema guarda de forma satisfactoria la configuración seleccionada y muestra dos mensajes en ambos casos informando que se ha realizado la operación con éxito, indicando la dirección del archivo editado para la aplicación y acción seleccionada. | |
| **Evaluación de la prueba:** Prueba satisfactoria. | |

**Tabla 30.** Configurar el acceso a la acción.

Caso de prueba correspondiente a la funcionalidad “separar privilegios sobre un recurso” encargada de retirar el privilegio que posee un usuario sobre un recurso del sistema:

| **Caso de prueba de aceptación** | |
|---|---|
| **Código:** HU3_P1 | **Historia de Usuario:** 3 |
| **Nombre:** Separar privilegios sobre un recurso. | |
| **Descripción:** Prueba para la funcionalidad de separar privilegios sobre un recurso. | |
| **Condiciones de Ejecución:**<br>• El usuario debe especificar la entidad de un recurso.<br>• El usuario selecciona un usuario que posea algún privilegio sobre el recurso especificado.<br>• El usuario selecciona un privilegio inferior al que posee el usuario seleccionado. | |
| **Resultados esperados:** El sistema registra el cambio de privilegios sobre el recurso seleccionado y muestra un mensaje informando que se ha separado el privilegio de forma exitosa. | |
| **Evaluación de la prueba:** Prueba satisfactoria. | |

**Tabla 31.** Separar privilegios sobre un recurso.

Caso de prueba correspondiente a la funcionalidad “conceder privilegios sobre un recurso” encargada de otorgar privilegios a un usuario sobre un recurso del sistema:

| **Caso de prueba de aceptación** | |
|---|---|
| **Código:** HU3_P2 | **Historia de Usuario:** 3 |
| **Nombre:** Conceder privilegios sobre un recurso. | |
| **Descripción:** Prueba para la funcionalidad de conceder privilegios sobre un recurso. | |

| |
|---|
| **Condiciones de Ejecución:**<br>• El usuario debe especificar la entidad de un recurso.<br>• El usuario selecciona un usuario que no posea privilegio alguno sobre el recurso especificado.<br>• El usuario selecciona un privilegio a ser concedido al usuario seleccionado. |
| **Resultados esperados:** El sistema registra el cambio de privilegios sobre el recurso seleccionado y muestra un mensaje informando que se ha concedido el privilegio de forma exitosa. |
| **Evaluación de la prueba:** Prueba satisfactoria. |

**Tabla 32.** Conceder privilegios sobre un recurso.

**Análisis de los resultados:**

Para validar que la salida emitida por el sistema informático concordara con el resultado esperado por el cliente se diseñaron 11 pruebas funcionales en conjunto cliente-desarrolladores de las cuales 5 salidas coincidieron con los resultados esperados representando un 45%.Fue notable el poco tiempo empleado por el sistema en la devolución del resultado, por otra parte 6 pruebas resultaron fallidas representando un 55%,en una segunda iteración de 11 pruebas funcionales realizadas 8 resultaron satisfactorias constituyendo un 72% y 3 implicaron fallos lo que significó un 18%, mientras que en una tercera iteración para las pruebas se arrojó resultados satisfactorios pues para el desarrollo de 11 pruebas funcionales se obtuvieron un total de 11 pruebas exitosas para un 100% de pruebas satisfactorias, mientras que 0 pruebas resultaron fallidas representando un 0% de las pruebas realizadas.

A continuación se muestra como quedan reflejados los resultados por cada una de las iteraciones de pruebas funcionales realizadas el sistema:

| Iteración | Satisfactoria | Insatisfactoria | Total |
|---|---|---|---|
| 1 | 5 | 6 | 11 |
| 2 | 8 | 3 | 11 |
| 2 | 11 | 0 | 11 |

**Tabla 33.** Iteraciones de pruebas funcionales.

### 3.2 Pruebas unitarias:

A continuación se muestran las pruebas unitarias aplicadas al sistema informático propuesto así como el resultado de aplicar métricas relacionadas con las mismas.

#### 3.2.1 Prueba del camino básico:

La prueba del camino básico es una técnica de prueba unitaria que permite al equipo de desarrollo obtener una medida de la complejidad lógica del código implementado como resultado del sistema propuesto en la investigación; usando dicha medida como guía para la definición de un conjunto de caminos independientes[19] de ejecución, lo que garantiza que durante la prueba se ejecuten por lo menos una vez cada sentencia del programa. (35)

##### 3.2.1.1 Complejidad Ciclomática

La complejidad ciclomática está basada en la teoría de grafos y da una métrica del software extremadamente útil, calculándose de tres formas:

1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
2. La complejidad ciclomática, V (G), de un grafo de flujo G se define como V (G)=A-N+2, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.
3. La complejidad ciclomática, V (G), de un grafo de flujo G también se define como V (G) P+ 1, donde P es el número de nodos predicado contenidos en el grafo de flujo G.

En la Figura 5 se puede apreciar el código correspondiente al método CrearFicheroM perteneciente a la clase InitSecurity que debido a la relevancia que posee en el sistema informático propuesto se le ha aplicado la métrica de complejidad ciclomática, donde se procede a:

- ✓ Confeccionar el grafo de flujo correspondiente.
- ✓ Calcular la complejidad ciclomática asociada.
- ✓ Extracción de los caminos independientes según el valor de la complejidad ciclomática.
- ✓ Realizar los casos de pruebas.
- ✓ Analizar los resultados obtenidos en las pruebas.

[19] Camino independiente se entiende aquel que introduce un nuevo conjunto de sentencias o una nueva condición. En términos del grafo, por una arista que no haya sido recorrida antes.

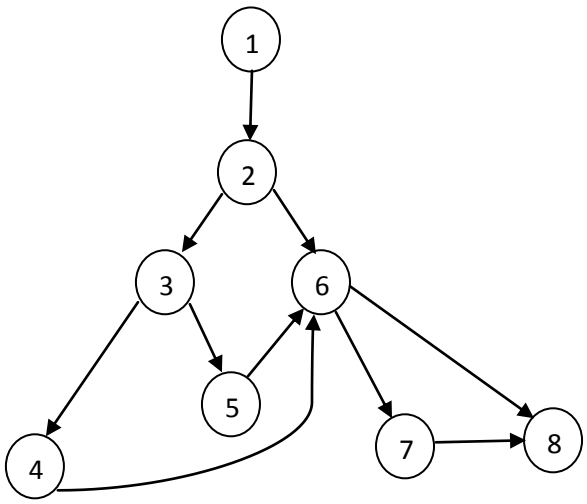
A continuación se muestra el código fuente que genera el fichero de configuración *security.yml*, ubicado en el directorio config del módulo indicado como parámetro, perteneciente a la aplicación especificada debido a que el mismo representa una funcionalidad fundamental para el control de acceso que posee el sistema informático.

```
public function CrearFicheroM($app,$mod)
{            //1
    $bandera=TRUE;
    $apps = sfFinder::type('dir')->maxdepth(0)->follow_link()->relative()->in(sfConfig::get('sf_apps_dir'));
    $modulos= sfFinder::type('dir')->maxdepth(0)->follow_link()->relative()->in(sfConfig::get('sf_apps_dir').'/'

              //2
        if (!is_dir(sfConfig::get('sf_apps_dir').'/'.$app.'/modules/'.$mod.'/config'))
            if(in_array($app, $apps) && in_array($mod, $modulos))//3
             {
                //4
                 mkdir(sfConfig::get('sf_apps_dir').'/'.$app.'/modules/'.$mod.'/config');
                 chmod(sfConfig::get('sf_apps_dir').'/'.$app.'/modules/'.$mod.'/config', 0777);
             }
             else
             {
                 //5
                $bandera=FALSE;
             }
         }
              //6
     if($bandera && !file_exists(sfConfig::get('sf_apps_dir').'/'.$app.'/modules/'.$mod.'/config/security.yml'))
     {
              //7
         fopen(sfConfig::get('sf_apps_dir').'/'.$app.'/modules/'.$mod.'/config/security.yml','w+');
         chmod(sfConfig::get('sf_apps_dir').'/'.$app.'/modules/'.$mod.'/config/security.yml', 0777);
     }
              //8
   return $bandera;
}
```

**Figura 5.** Código fuente que genera el fichero de configuración security.yml.

### Grafo de flujo



**Cálculo de la Complejidad Ciclomática:**

1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.

   El número de regiones del grafo del flujo definido es 4 lo que deviene que se obtenga el siguiente resultado:
   V (G)= 4

2. V (G)=A-N+2=10-8+2=4
3. V(G)=P+1=3+1=4

Una vez calculada la complejidad ciclomática por cada una de sus variantes se define como límite superior cuatro pruebas a realizar para que todo el código se encuentre probado, no siendo complejo el código analizado por el equipo de desarrollo debido que el resultado arrojado por la métrica pertenece al intervalo entre 1-10 indicando que código perteneciente al sistema informático propuesto en la investigación es simple y sin mucho riesgo.

**Caminos independientes establecidos:**

Camino1: 1-2-6-8

Camino2: 1-2-6-7-8

Camino3: 1-2-3-5-6

Camino4: 1-2-3-4-6-7-8

### 3.2.2 Casos de pruebas

Cada camino es un caso de prueba, de forma tal que los datos introducidos provoquen que se visiten las sentencias vinculadas a cada nodo del camino como se refleja en las Figuras 6, 7, 8 y 9 respectivamente.

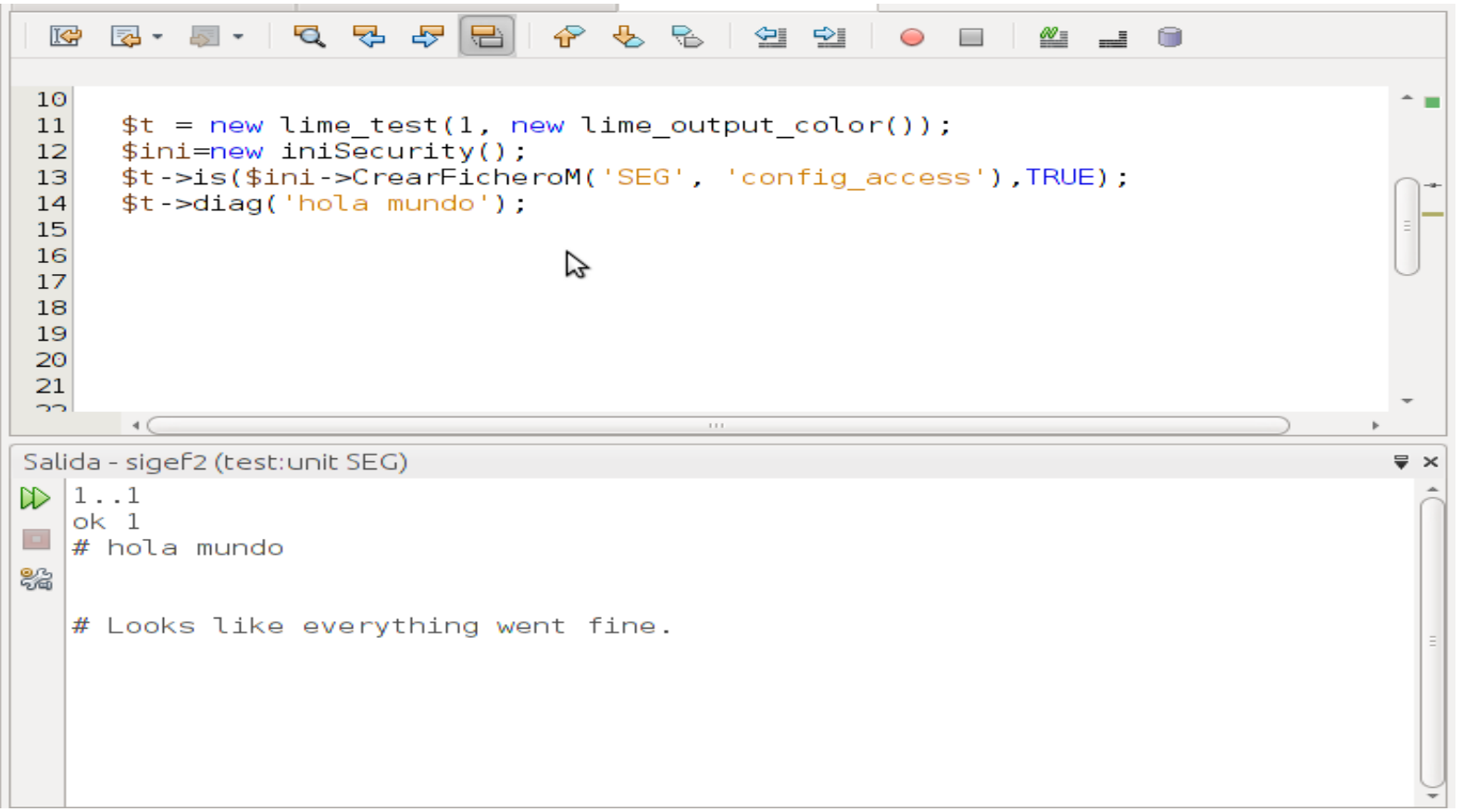



**Figura 6.** Caso de prueba para el camino 1.

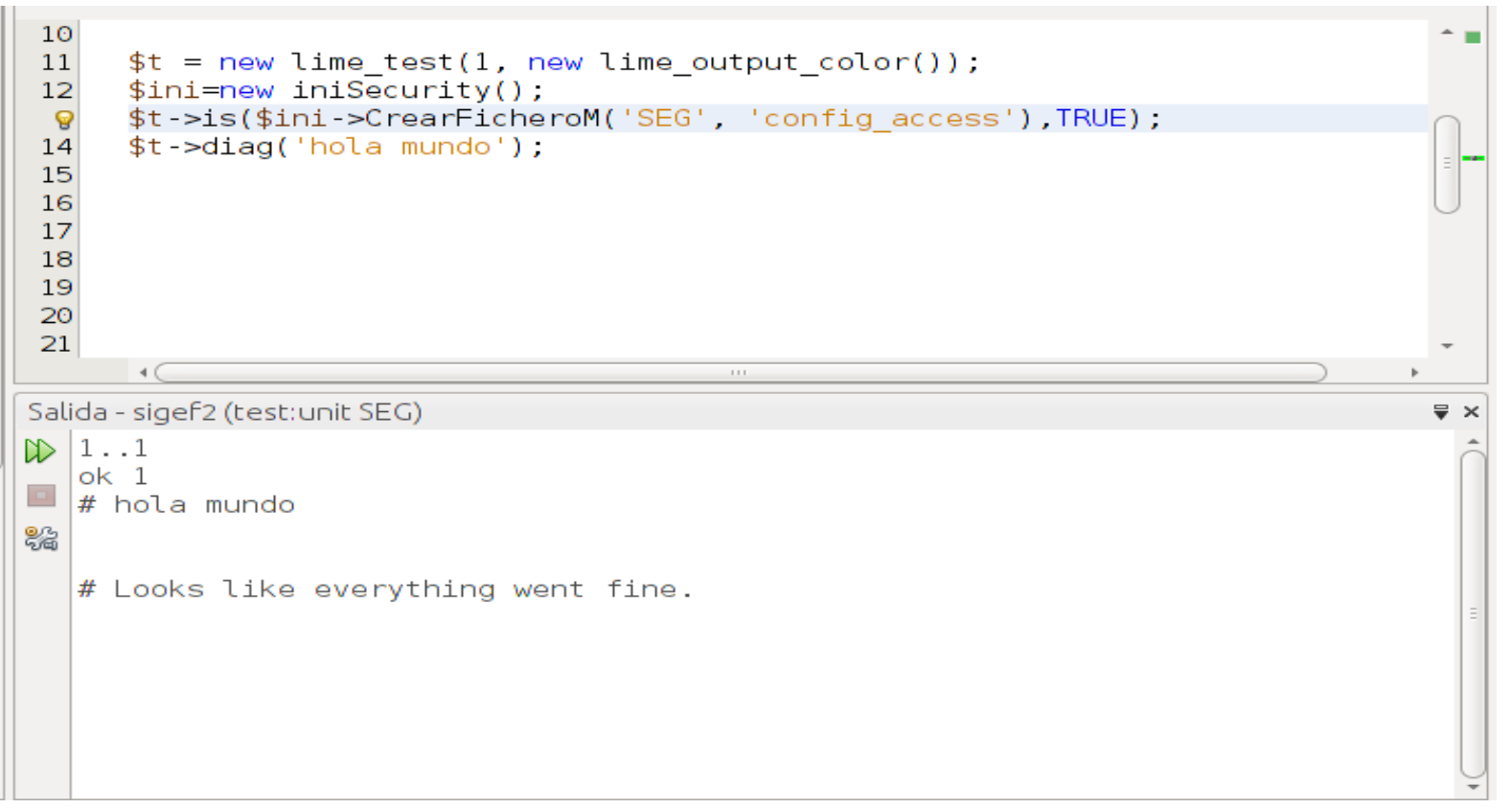



**Figura 7.** Caso de prueba para el camino 2.

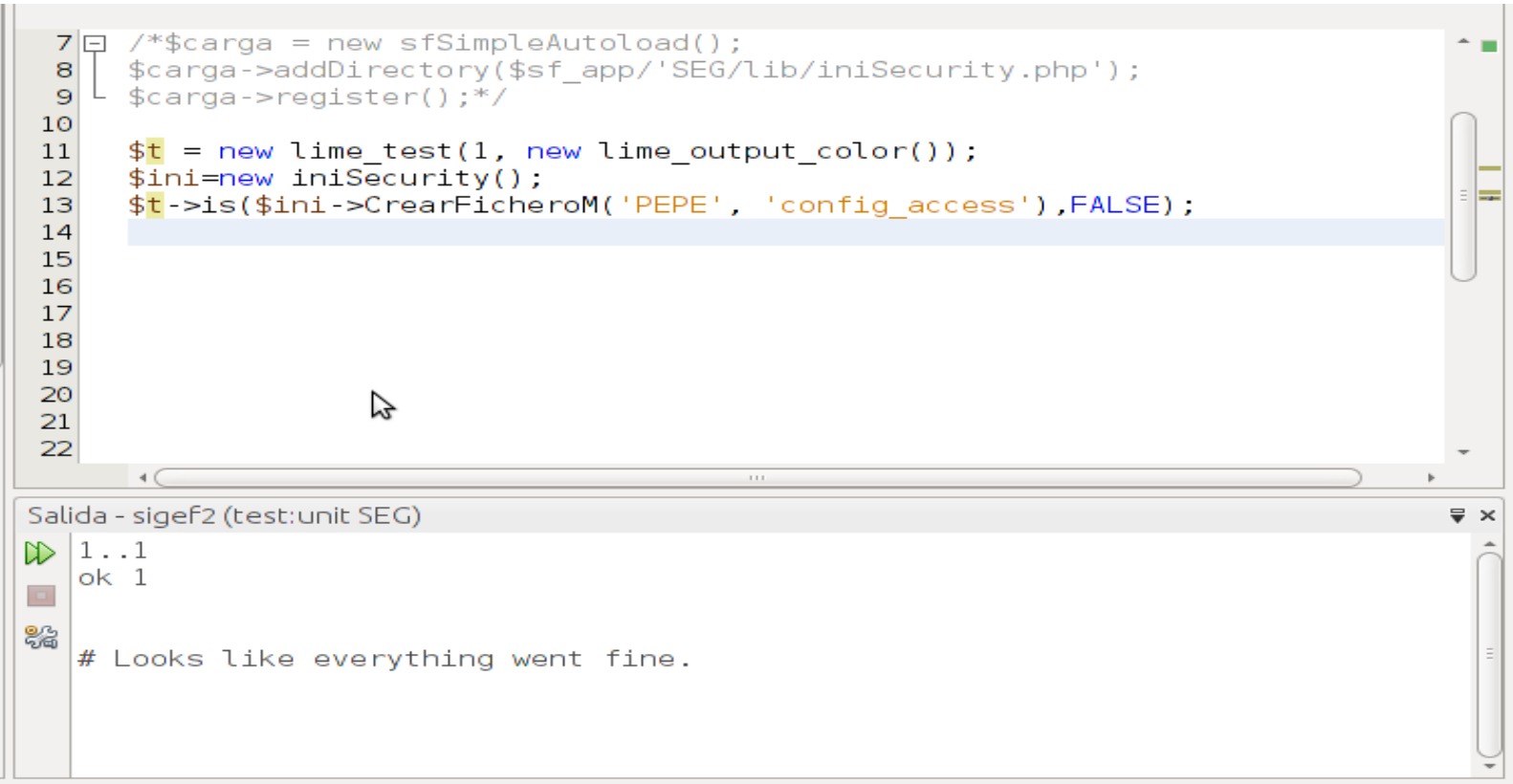



**Figura 8.** Caso de prueba para el camino 3.

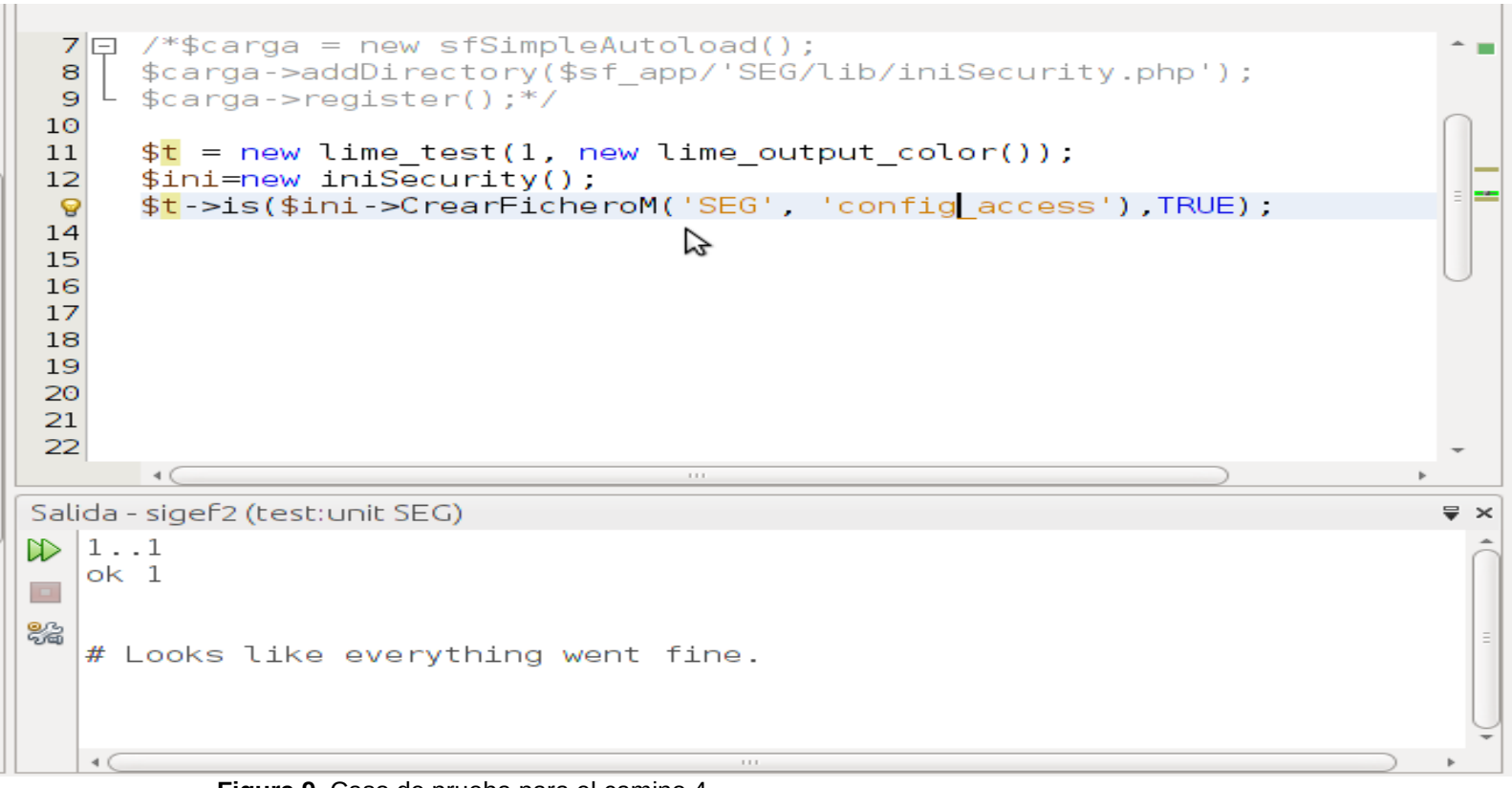



**Figura 9.** Caso de prueba para el camino 4.

#### 3.2.2.1 Análisis de los resultados

Para validar que el código implementado, se realizaron veinte pruebas a las funcionalidades del sistema informático propuesto como parte de la investigación.De las pruebas unitarias realizadas cuatro estuvieron vinculadas a la funcionalidad CrearFicheroM descrita con anterioridad, donde de los 20 casos de pruebas realizados 18 resultaron satisfactorios representando un 90%; verificando la estabilidad de la lógica aplicada en el código probado, perteneciente al sistema informático propuesto en la investigación, mientras que las 2 restantes resultaron fallidas representando un 10%.En una segunda iteración de pruebas al sistema informático de un total de 20 pruebas se obtuvo como resultado 20 pruebas satisfactorias representando un 100% y 0 pruebas fallidas lo que significa un 0% quedando probado más del 80% del código del sistema informático propuesto en la investigación.

A continuación se hace muestra de los resultados obtenidos para el desarrollo de las iteraciones correspondientes a las pruebas unitarias:

| Iteración | Satisfactoria | Insatisfactoria | Total |
|---|---|---|---|
| 1 | 18 | 2 | 20 |
| 2 | 20 | 0 | 20 |

**Tabla 34.** Iteraciones de pruebas unitarias.

El grado de calidad y fiabilidad del diseño alcanzado por el sistema se midió a través de métricas basadas en las clases; las cuales miden categorías como tamaño operacional de clase y relaciones entre clases, permitiendo el resultado de ambas métricas validar el diseño propuesto en la investigación.

A continuación se muestra como se aplicó la métrica Tamaño Oeracional de Clase (TOC) al sistema informático.

### 3.2.3 Tamaño operacional de clase (TOC)

El tamaño operacional de una clase se puede determinar empleando medidas para saber el número total de operaciones. (35)

Las clases que juegan un papel importante en los procesos del sistema informático propuesto en la investigación se encuentran relacionadas en el controlador y en el modelo, siendo aplicada a estas clases la métrica TOC.

A continuación se muestran las clases que juegan un papel fundamental en los procesos principales del sistema informático:

| Módulo | Clase | Cantidad de Procedimientos |
|---|---|---|
| config_access | Tbsegacl | 8 |
| config_access | Tbsegidenditadobjeto | 4 |
| config_access | Nsegpermisoobjeto | 4 |
| config_access | Nsegrecurso | 4 |
| config_access | Tbpersona | 12 |
| config_access | InitSecurity | 6 |
| config_access | SegFiltroPDC | 1 |
| config_access | SegFiltroCLEP | 1 |
| config_access | ParseoAcciones | 3 |
| config_access | Config_AccessActions | 20 |

**Tabla 35.** Clases fundamentales.

#### 3.2.3.1 Resultados obtenidos

Los umbrales que referencian a los atributos de calidad: responsabilidad de las clases, complejidad al implementar las mismas, así como sus niveles de reutilización los cuales se le aplican a las 10 clases del sistema informático con un promedio de 6.3 operaciones, quedan especificados en la siguiente tabla:

| | Categoría | Criterio |
|---|---|---|
| **Responsabilidad** | Baja | < =6.3 |
| | Media | Entre 6.3 y 12.6 |
| | Alta | > 12.6 |
| | | |
| **Complejidad de implementación** | Baja | < =6.3 |
| | Media | Entre 6.3 y 12.6 |
| | Alta | > 12.6 |
| | | |
| **Reutilización** | Baja | > 12.6 |
| | Media | Entre 6.3 y 12.6 |

| | | |
|---|---|---|
| | Alta | <= 6.3 |

**Tabla 36.** Umbrales para la responsabilidad, complejidad y reutilización.

En la Tabla 32 y Figuras 10,11, y 12 se muestran como quedan reflejados los atributos de calidad antes mencionados en las clases sometidas a la métrica de diseño propuesta:

| Módulo | Clase | Cantidad de Procedimientos | Responsabilidad | Complejidad | Reutilización |
|---|---|---|---|---|---|
| config_access | Tbsegacl | 8 | Alta | Alta | Baja |
| config_access | Tbsegidenditadobjeto | 4 | Media | Media | Media |
| config_access | Nsegpermisoobjeto | 4 | Media | Media | Media |
| config_access | Nsegrecurso | 4 | Media | Media | Media |
| config_access | Tbpersona | 12 | Alta | Alta | Baja |
| config_access | InitSecurity | 6 | Media | Media | Media |
| config_access | SegFiltroPDC | 1 | Baja | Baja | Alta |
| config_access | SegFiltroCLEP | 1 | Baja | Baja | Alta |
| config_access | ParseoAcciones | 3 | Baja | Baja | Alta |
| config_access | Config_AccessActions | 20 | Alta | Alta | Baja |

**Tabla 37.** Clases analizadas con resultados obtenidos.

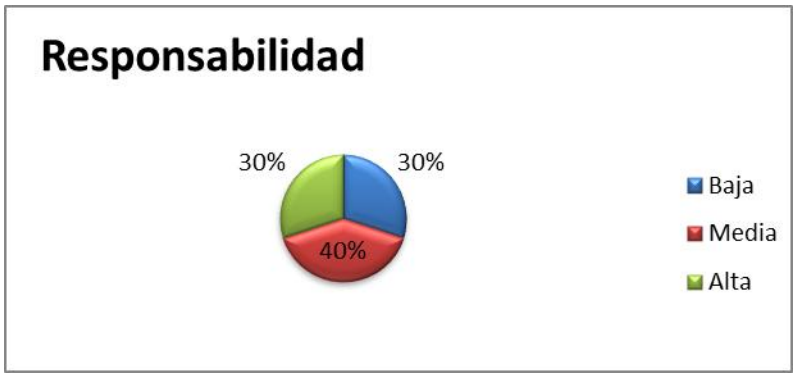



**Figura 10.** Atributo de calidad que refleja la responsabilidad.

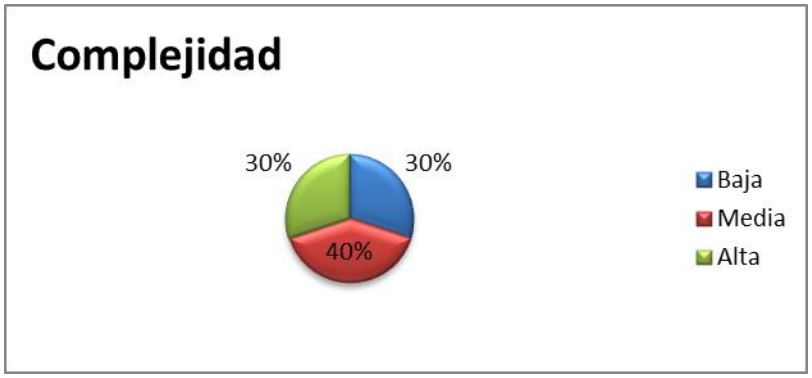



**Figura 11.** Atributo de calidad que refleja la complejidad.

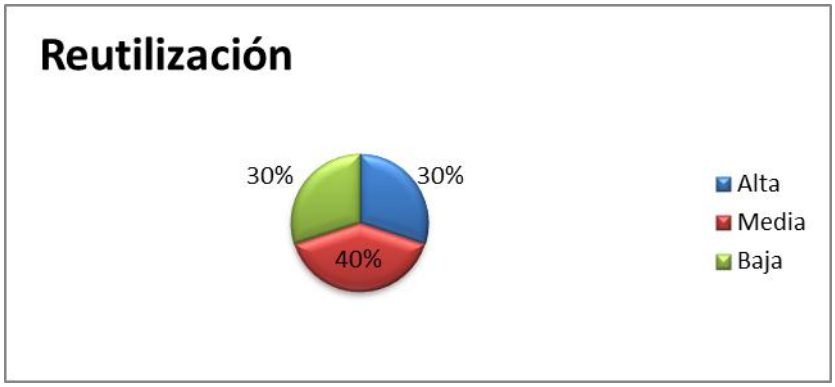



**Figura 12.** Atributo de calidad que refleja la reutilización.

Tomando como referencia la información referida por las Figuras 10, 11 y 12 se muestra con un 40% al umbral medio como el porciento sobresaliente a los atributos de calidad responsabilidad, reutilización y complejidad. Esto trae consigo que el nivel de responsabilidad sea medio así como la capacidad de ser reutilizada y complejidad de las clases del sistema informático propuesto.

### 3.3 Conclusiones Parciales

El capítulo presentó las pruebas realizadas al software. En los valores obtenidos en las 3 iteraciones de pruebas funcionales se comprobó que las funcionalidades diseñadas por el cliente se encontraban correctas, mientras que en las 2 iteraciones de pruebas unitarias se hace uso de la herramienta Lime verificando que el código diseñado por el equipo de desarrollo es válido. Por otra parte al aplicar la métrica TOC se obtuvieron valores medios para los atributos de calidad responsabilidad, complejidad y reutilización.

# ***Conclusiones Generales***

Con la culminación del presente trabajo devienen favorables resultados los cuales pueden concluirse de la siguiente forma:

- El estudio realizado permitió la selección de un mecanismo de control del acceso así como las herramientas, tecnologías y metodología adecuadas para conducir el desarrollo del sistema informático.
- Se diseñó el sistema propuesto sobre la base de la identificación de las historias de usuario.
- Se implementó el sistema propuesto sobre la base de la identificación de las funcionalidades identificadas por el usuario en conjunto con el equipo de desarrollo.
- Se validó la solución informática a través de las iteraciones previstas por el equipo de desarrollo adecuándose el resultado obtenido a un alto grado de calidad.

# ***Recomendaciones***

- Reutilizar el componente desarrollado en otros proyectos de la Universidad con similares características.

- Informatizar configuraciones de archivos con extensión yml de uso en el marco de desarrollo y que tributan a la seguridad de la aplicación.

- Incluir funcionalidades que permitan verificar los privilegios apropiados de un usuario sobre un determinado objeto o recurso del sistema teniendo en cuenta el uso de más de un usuario físico en la base de datos.

# ***Bibliografía***


1. **Sánchez, Miguel, Jiménez, Beatriz y Gutiérrez, Francisco L.** *Estudio del Control de Acceso en Sistemas Colaborativos.* Zaragoza : s.n., 2007 .

2. **Ribagorda, Antonio.** *Glosario de Términos de Seguridad de las T.I.* s.l. : Ediciones CODA, 1997.

3. Oracle. [En línea] 1997. [Citado el: 18 de marzo de 2012.] http://docs.oracle.com/cd/E24842_01/html/E22521/ugintro-14.html.

4. **Dominguez Martin, Grabiel A.** *Control de Acceso Basado en Roles.* 1999.

5. **Caloyannides, Michael A. y Ferraiolo, David F. .** *Role-Based Access Control.* 2007. ISBN 13: 978-1-59693-113-8.

6. EcuRed. [En línea] 15 de diciembre de 2011. [Citado el: 23 de enero de 2012.] http://www.ecured.cu/index.php/Sistemas_de_control_de_acceso.

7. Maestros de la web. [En línea] 2003. [Citado el: 20 de marzo de 2012.] http://www.maestrosdelweb.com/editorial/phpintro/.

8. DocIRS. [En línea] 2002. [Citado el: 18 de Abril de 2012.] http://www.docirs.cl/uml.htm.

9. The Unified Modeling Language. [En línea] 1997. [Citado el: 20 de febrero de 2011.] http://www.uml.org/.

10. EcuRed. [En línea] 2011. [Citado el: 12 de marzo de 2012.] http://www.ecured.cu/index.php/Metodolog%C3%ADas_de_desarrollo_de_software.

11. **JACOBSON, Ivar y BOOCH, Grady.** *El Proceso Unificado de Desarrollo de Software. Pearson Addisson-Wesley.* 2000.

12. **Martinez, Alejandro.** *Guia a rational Unified Process.* Castilla de laMancha : s.n., 1997.

13. Programación Extrema. [En línea] 14 de noviembre de 2002. [Citado el: 16 de febrero de 2011.] http://www.programacionextrema.org/.

14. Programación Extrema. [En línea] 2002. [Citado el: 14 de marzo de 2012.] http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-XP.pdf.

15. La nueva Metodología. [En línea] marzo de 2003. [Citado el: 15 de Marzo de 2012.] http://www.programacionextrema.org/articulos/newMethodology.es.html.

16. EcuRed. [En línea] 15 de diciembre de 2011. [Citado el: 20 de febrero de 2012.] http://www.ecured.cu/index.php/Herramienta_CASE.

17. Buenas tareas. [En línea] 2003. [Citado el: 20 de marzo de 2012.] http://www.buenastareas.com/materias/ventajas-de-las-herramientas-case/0.

18. Rational software. [En línea] 2003. [Citado el: 25 de febrero de 2012.] http://www-01.ibm.com/software/rational/.

19. Visual Paradigm for Uml. [En línea] 2004. [Citado el: 10 de marzo de 2012.] http://www.visual-paradigm.com/product/vpuml/.

20. POO (Programación Orientada a Objetos) con php. [En línea] [Citado el: 15 de marzo de 2012.] http://www.phpya.com.ar/poo/.

21. slishare. [En línea] [Citado el: 25 de marzo de 2012.] http://www.slideshare.net/dersteppenwolf/introduccin-a-programacin-orientada-a-objetos-oop-clases-y-objetos.

22. **Potencier, Fabien y Zaninotto, François.** *Symfony Guía Definitiva.* 2000.

23. EcuRed. [En línea] 2011. [Citado el: 18 de marzo de 2012.] http://www.ecured.cu/index.php/Eclipse,_entorno_de_desarrollo_integrado.

24. Entorno Integrado de Desarrollo Netbeans. [En línea] [Citado el: 20 de marzo de 2012.] www.netbeans.org.

25. PostgreSQL. [En línea] 1996. [Citado el: 22 de marzo de 2012.] http://www.postgresql.org/.

26. Propel ORM. [En línea] [Citado el: 25 de marzo de 2012.] http://www.propelorm.org .

27. EcuRed. [En línea] 2011. [Citado el: 30 de Marzo de 2012.] http://www.ecured.cu/index.php/JQuery.

28. Subversion. [En línea] 1997. [Citado el: 13 de abril de 2012.] subversion.tigris.org.

29. EcuRed. [En línea] 2011. [Citado el: 12 de abril de 2012.] http://www.ecured.cu/index.php/Subversion.

30. Axure. [En línea] [Citado el: 22 de marzo de 2012.] www.axure.com/.

31. **Noble, Angela Martin y de Robert Biddle, James.** *The XP Customer Role in Practice: Three Studies Agile.* 2004.

32. **Bikha, Avinash .** *Requirements Management – Defining the Project Scope and Developing Use Cases.* 2008.

33. **Larman, Craig .** *UML y Patrones.*

34. EcuRed. [En línea] 2011. [Citado el: 15 de abril de 2012.] http://www.ecured.cu/index.php/Patrones_Gof.

35. **Pressman, Roger S.** *Ingenieria de Software.* Madrid : s.n., 1997.

36. Subversion. [En línea] 2001. [Citado el: 22 de marzo de 2012.] http://subversion.tigris.org/.

37. EcuRed. [En línea] [Citado el: 13 de marzo de 2012.] http://www.ecured.cu/index.php/Sistemas_de_control_de_versiones.