Universidad de las Ciencias Informáticas

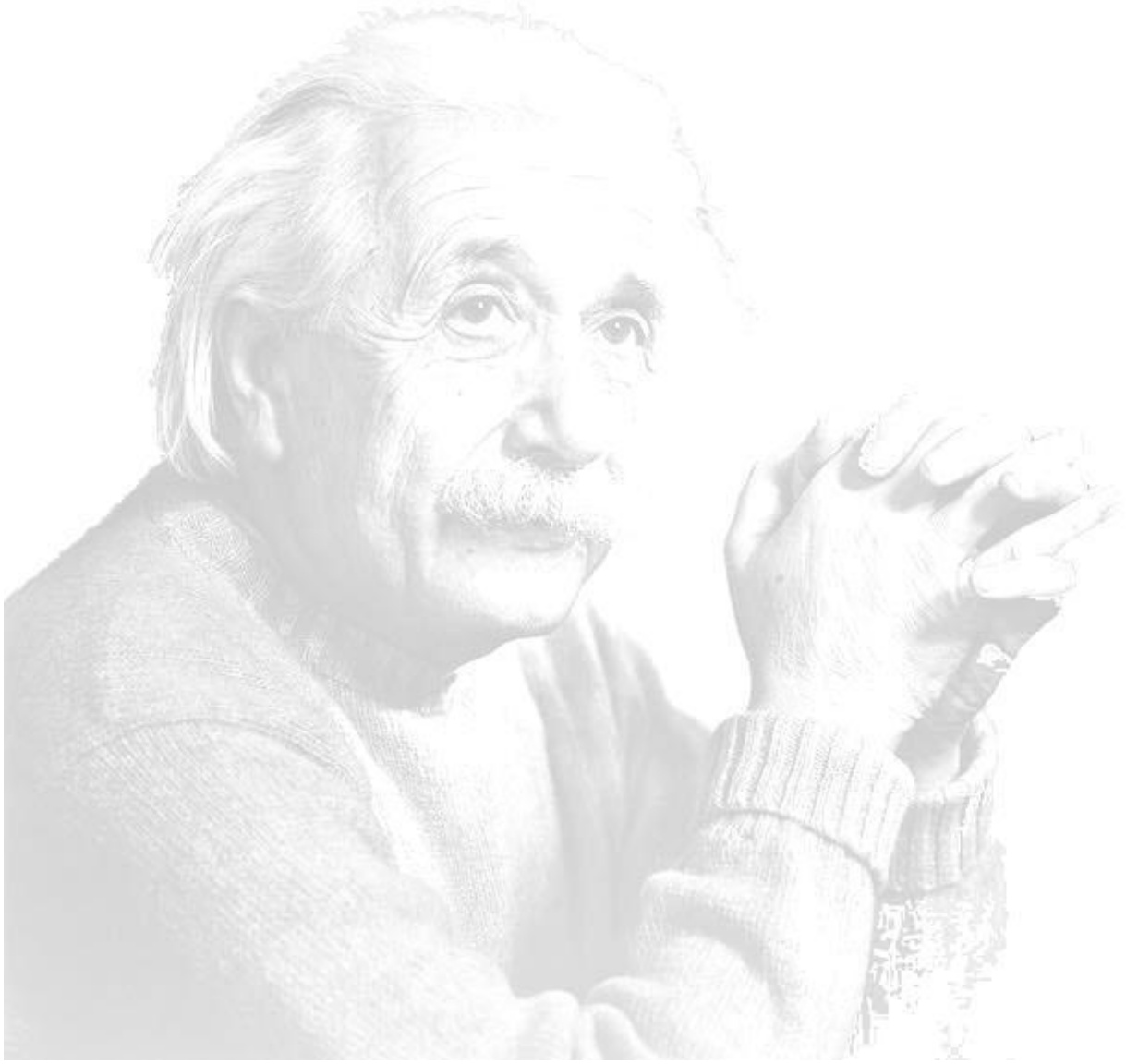
Facultad 5

Título: Desarrollo de una herramienta para el diseño gráfico de políticas de seguridad.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor: Denys López López
Tutor: Ing. Jorge Hernández Roselló

Ciudad de la Habana
"Año 53 de la Revolución"



Lo más incomprensible del mundo es que sea comprensible.

Albert Einstein

A mi familia, a mis padres, que siempre me han sabido guiar, y aguantar.

A mis amigos, a mis hermanos que siempre están conmigo.

En fin, a todas las personas que me tienen en su estima.

Agradecimientos:

A mi familia, a mis padres por darme todo el apoyo y el amor del mundo, mi abuelo, mis tíos y mis primas, porque siempre están ahí dispuesto a todo. En especial a los que hoy no pueden estar aquí, pero que no por eso están más lejos, porque están aquí en mi corazón.

A mi madre por todo su amor, su cariño y su ternura, la que solo una madre es capaz de ofrecer.
A mi padre que me inspira cada día a ser mejor y a convertirme en todo lo que quiero ser.

A mis amigos, que son mis hermanos: Pedro, Manu, Daya, Gilda y Carla.
A mi novia Zenia, por todo lo que se preocupa y por todo su cariño.

A Jorge, mi tutor, que sin él esto no echa adelante.

A todos, GRACIAS.

Declaración de autoría:

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Denys López López Ing. Jorge Hernández Roselló

Datos del contacto:

Ing.Jorge Hernández Roselló.
Ingeniero en Ciencias Informáticas, Profesor Asistente.
e-mail: jhernandez@uci.cu

Resumen

Este trabajo aborda la investigación, desarrollo e implementación de una aplicación web que consiste en una herramienta para el desarrollo de políticas de seguridad mediante paneles gráficos, permitiendo el manejo de dichas políticas de forma fácil y amena. Incluye un estado del arte acerca de algunas de las principales herramientas relacionadas con el diseño de políticas y sus características, así como las herramientas y frameworks utilizados para el diseño de la aplicación propuesta. La aplicación y sus funciones fueron desarrolladas siguiendo las estrategias propuestas por la metodología de software XP, haciendo un uso extensivo del lenguaje JAVA y frameworks como Vaadin. Además se hace mención a las estrategias de pruebas realizadas.

Palabras claves: Seguridad – Política – Política de seguridad – Servicios Web

Abstract

This paper addresses the investigation, development and implementation of a web application that consist in a tool for developing security policies using graphical panels, allowing the handling of this policies in an easy and fun way. It does include a state of the art about some of the main tools used for design policies and their characteristics, as well as the tools and frameworks to design the proposed application. The application and its main functions were developed according to the strategies proposed by the software methodology XP, making an extensive used of JAVA language and frameworks like Vaadin. Besides the test strategies made and the conclusionsare explained.

Key words: Security – Policy – Security policy – Web services

Índice

Introducción

El surgimiento de la Internet y la sucesión de importantes descubrimientos, a partir de la década de 1980, permitieron la expansión de las redes informáticas que se desarrollaron de manera vertiginosa pero sin ningún tipo de sistema de protección importante contra posibles ataques a la información, los cuales no tardaron en ocurrir de la mano de los llamados *hackers*, o sea, *aquel individuo con talento, conocimiento e inteligencia que usa sus habilidades y recursos para invadir sistemas informáticos ajenos*[1]; haciéndose comunes en casi todos los lugares.

Esta problemática se ha incrementado debido a la proliferación y el uso creciente de las tecnologías en un mundo altamente globalizado, donde una gran cantidad de datos, vitales y sensibles, se están almacenando en ordenadores con una frecuencia cada vez mayor. Entre estos datos se pueden incluir registros públicos, de individuos o de negocios, así como secretos gubernamentales y militares, además de las grandes transacciones monetarias que tienen lugar diariamente en forma de transferencias electrónicas de fondos. Este flujo de servicios de un sitio a otro hace que aparezcan vulnerabilidades que ponen en riesgo la seguridad de la infraestructura de comunicación y de toda la información contenida en sus nodos. Por lo tanto, proteger dicho flujo y los recursos tecnológicos informáticos se hace *una tarea continua y de vital importancia que debe darse en la medida en que avanza la tecnología, ya que las técnicas empleadas por aquellos que usan dichos avances para fines delictivos aumentan y como resultado los atacantes son cada vez más numerosos, mejor organizados y con mejores capacidades*[2].

El desarrollo de las comunicaciones entre sistemas informáticos y la necesidad de una estrategia de seguridad eficiente, si se entiende que esta *implica la cualidad o estado de estar seguro, es decir, la prevención de exposiciones a situaciones de peligro y la actuación para quedar a cubierto frente a contingencias adversas*[3], debe necesariamente contar con políticas, o sea, *las metas rectoras en la dirección estratégica hacia la toma de decisiones para la consecución delos objetivos de un grupo determinado*[4]. Este desarrollo en los mecanismos de comunicación ha potenciado el surgimiento de nuevos paradigmas y estándares, como la programación orientada a servicios, que han venido a moldear conceptos anteriores referentes al intercambio de información entre diferentes aplicaciones

informáticas, con la importante necesidad de desarrollar una mejor implementación de seguridad, particularmente en la protección de la comunicación que se establece entre los servicios.

Este revolucionario concepto de la programación orientada a servicios, ha potenciado, a su vez, el surgimiento de los servicios web, que se definen como *una unidad de aplicación capaz de ofrecer datos o servicios de procesamiento a otras aplicaciones informáticas*[5], es decir, *son programas que conectan con otros programas o aplicaciones, pero que no interactúan directamente con los usuarios*[6], por ejemplo, una función disponible en un servidor conectado a la Web, dígase un cálculo aritmético al que cualquier persona o servicio que tenga acceso a dicho servidor pueda utilizar; convirtiéndose el navegador de cualquier buscador en un *cliente universal*[5]. Estas ventajas hacen que este tipo de servicios gocen de una gran popularidad a nivel mundial, despertando un creciente interés por proteger la información contenida en ellos.

Este interés ha incentivado la creación de iniciativas para definir estándares de seguridad que rijan el proceso de desarrollo de las políticas que se incluyen en dichos servicios. Entre los estándares creados más importantes se encuentran las Arquitecturas Orientadas a Servicios (SOA[1], por sus siglas en inglés), una arquitectura de sistemas reconocida mundialmente y respaldada por los principales expertos y empresas tales como IBM[2] y la consultora Gartner[3] (*la cual ubica a SOA dentro de las 10 tecnologías más utilizadas en la industria de Seguros*[7]) y que se especializa en la implementación de servicios de negocio, esto es; *bloques funcionales de negocio que se pueden integrar, y compartir en distintas aplicaciones*[8]. Esta arquitectura ha sido implementada sobre protocolos y lenguajes reconocidos, de los cuales pueden mencionarse el protocolo SOAP[4] y los lenguajes WSDL[5] y XML[6].

[1]*Concepto de arquitectura de software que define el uso de servicios para dar soporte a los requisitos del negocio y provee métodos para el desarrollo de sistemas e integración.*
[2]*Empresa multinacional estadounidensede tecnología y consultoría con sede en Armonk, Nueva York, conocida como el gigante azul. IBM fabrica y comercializa hardware y software para computadoras, y ofrece servicios de infraestructura, alojamiento de Internet, y consultoría en una amplia gama de áreas relacionadas con la informática, desde computadoras centrales hasta nanotecnología.*
[3]*Es una empresa consultora y de investigación de las tecnologías de la información estadounidense con sede en Stamford, Connecticut.*
[4]*Protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.*
[5]*Acrónimo de WebServicesDescriptionLanguage el cual es un formato XML que se utiliza para describir servicios web.*
[6]*Acrónimo deeXtensibleMarkupLanguage (en español lenguaje de marcas extensible) es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C) que permite definir la gramática de lenguajes específicos.*

Las facilidades que brindan los servicios web, así como los estándares que lo implementan, entre ellos SOA, han mejorado la interoperabilidad, despliegue e integración de las aplicaciones. Sin embargo, a pesar de estas facilidades, el uso de lenguajes y protocolos como SOAP y XML, exponen a las empresas a potenciales riesgos de seguridad, de ahí la importancia de incorporar políticas que mitiguen estos problemas. Particularmente, la necesidad de proteger la gestión de la información en los negocios y los servicios partiendo de políticas de seguridad, presenta dificultades propias que de manera general yacen en el desconocimiento a la hora de implementar estas políticas, así como la falta de experiencia y de una estructura genérica que guíe el proceso de diseño e implementación de este tipo de políticas.

Partiendo de la situación antes descrita, se plantea como **problema científico** de la investigación: ¿Cómo viabilizar el diseño de políticas de seguridad dentro de soluciones orientadas a servicios, de manera que facilite la implementación de las mismas?, siendo el **objeto de estudio**, las políticas de seguridad dentro de soluciones orientadas a servicios y su **campo de acción** enmarcado en el proceso de diseño de políticas de seguridad dentro de soluciones orientadas a servicios.

El **objetivo general** que se plantea es desarrollar una herramienta que a partir del diseño gráfico de las políticas de seguridad, permita generar el código de las mismas.
De esta manera, y para dar cumplimiento al objetivo trazado, se presenta la siguiente **idea a defender**: con el desarrollo de una herramienta interactiva y de interfaz amigable, que permita la creación de políticas de seguridad por personal de negocio mediante un diseño gráfico, se viabilizará la escritura, comprensión y manejo de las mismas.

Para dar cumplimiento además al objetivo planteado se proponen los siguientes **objetivos específicos**:

- ✓ Determinar las tendencias actuales relacionadas con el desarrollo de herramientas para la escritura de las políticas de seguridad dentro de soluciones SOA.
- ✓ Diseñar la herramienta para el diseño gráfico de políticas de seguridad.
- ✓ Implementar la herramienta para el diseño gráfico de políticas de seguridad.
- ✓ Validar la herramienta desarrollada.

Las **tareas de la investigación** a realizar son las siguientes:

1. Realizar un análisis de las tendencias actuales relacionadas con el desarrollo de herramientas para la escritura de las políticas de seguridad en el marco de soluciones SOA.
2. Determinar la metodología, tecnologías y herramientas para el desarrollo de la solución.
3. Determinar los requerimientos funcionales y no funcionales para el desarrollo de la herramienta.
4. Diseñar las interfaces de la herramienta.
5. Definir la arquitectura de la herramienta.
6. Implementar la herramienta basándose en la arquitectura definida.
7. Analizar tipos de pruebas a realizar a la herramienta.
8. Realizar pruebas a la herramienta.

Para un mejor entendimiento, esta investigación ha sido estructurada de la siguiente manera:

Capítulo1. Fundamentación Teórica: Contiene la fundamentación teórica del tema tratado en la investigación. Se describen los conceptos fundamentales asociados al dominio del problema, así como el estudio del arte, antecedentes, tendencias, técnicas, tecnologías, metodologías y software[7] usados en la actualidad referente al tema a tratar.

Capítulo 2. Análisis y Diseño del Sistema: Contiene de forma detallada una breve descripción de la solución propuesta, sus reglas de negocio, requisitos funcionales y no funcionales, así como las historias de usuario, planesy descripción de las iteracionesutilizadas para darle respuesta a la problemática planteada.

Capítulo 3. Implementación y Despliegue del Sistema: Contiene los diagramas de despliegue y de componentes, así como una explicación de la arquitectura implementada en la solución propuesta, además de las pruebas y técnicas de validación que le serán aplicadas al sistema para demostrar la robustez del mismo.

[7] *Grupo de secuencias que indican al hardware que debe realizar.*

Capítulo 1: Fundamentación Teórica.

1.1 Introducción

En el presente capítulo se explica la importancia de las políticas de seguridad en los servicios web y se realiza un análisis profundo en la búsqueda de herramientas que diseñan políticas de seguridad en el mundo, donde pueden apreciarse las deficiencias encontradas en dichas herramientas que decidieron la implementación de una propia para el CDAE[8]. Además, se realiza un estudio de la metodología Programación Extrema, de la herramienta CASE Visual Paradigm y de las tecnologías y herramientas analizadas para desarrollar la aplicación, tales como los lenguajes de programación Java, el ambiente de desarrollo Eclipse, el framework Vaadin, la librería dom4j y el lenguaje de modelo UML.

1.2 ¿Por qué es importante la seguridad informática y sus políticas?

En la actualidad, un alto por ciento de las computadoras personales (Personal Computer - PC, por sus siglas en inglés), están infectadas con algún tipo de software malicioso, por lo que están expuestas a situaciones tales como la pérdida de datos importantes, particularmente las empresas lo cual podría provocar la detención de los negocios o incluso el quiebre de las mismas, provocando daños como la c*aída en la performance*[9] *de los equipos* [9]ylentitud en tareas que antes se realizaban con mucha mayor rapidez. Además, los entornos con alto nivel de riesgo, aquellos en los que funcionan computadoras interconectadas en un sistema de red, que poseen conexión a Internet y son muy comunes mundialmente, son los más afectados por este flagelo, ya que *los códigos maliciosos se propagan a través de la red tratando de infectar a todas las computadoras* [9]que forman parte de ella.

Por estos motivos se hace fundamental la generación de conciencia en los usuarios de lo que significa la seguridad informática. Es un hecho que los beneficios de un sistema de seguridad bien elaborado son inmediatos, ya que se trabaja sobre una plataforma confiable, lo que se refleja en el aumento de la productividad, la formación de equipos competentes y la mejora en los climas laborares para los Recursos Humanos, minimizando las pérdidas de información sensible y de dinero.

[8] *Acrónimo deCentro de Consultoría y Desarrollo de Arquitecturas Empresariales.*
[9] *Rendimiento o actuación.*

La eficiente implementación de un sistema de seguridad confiable, pasa necesariamente por el desarrollo de políticas bien definidas que lo conformen. Este tipo de políticas persiguen el objetivo de proporcionar las directrices que soporten y orienten el proceso de seguridad de la información en una organización. El documento que les da forma incluye *las reglas, normas y procedimientos que regulan la forma en que una organización previene, protege y maneja los riesgos de daño sobre:*

- *Los sistemas de soporte general y los elementos físicos asociados a estos (computadoras, cables, discos, entre otros).*
- *El software y la información almacenada en tales sistemas.*
- *Los usuarios del sistema*[10].

Además, según los principios propuestos por el *Site Security Handbook*[10], la guía de referencia para la creación de políticas de seguridad para sitios que tienen sistemas en la Internet, debe contener políticas tales como de confidencialidad, de autenticación, de acceso, de contabilidad, entre otras.

1.3 Importancia de las políticas de seguridad en los servicios web

Los avances en materia de seguridad presentan una gran ayuda al desarrollo de los servicios web, necesitados de formas eficientes que protejan los sistemas que lo conforman y manejan. Estos necesitan de un amplio espectro de mecanismos que solventen problemas como *la autenticación, el control de acceso basado en roles, la aplicación efectiva de políticas de seguridad distribuidas o la seguridad a nivel de los mensajes*[11], puesnecesitan de los mismos servicios de seguridad básicos encontrados en cualquier sistema distribuido como son la autenticación, la autorización o la confidencialidad. Desde el punto de vista más básico, el objetivo es conseguir definir un entorno en el que las transacciones de los mensajes y los procesos de negocio se puedan llevar a cabo de manera segura de extremo a extremo.

Las ventajas que ofrecen este tipo de servicios son numerosas como, por ejemplo, la interoperabilidad que aportan entre aplicaciones de software, con independencia de sus propiedades o de las plataformas sobre las que se instalen, el fomento de estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento o la habilidad de proveer

[10]*Manual de seguridad utilizado como estándar para establecer Políticas de Seguridad. Fue escrito por varios autores y publicado en Septiembre de 1997.*

servicios integrados, a partir de servicios o software de compañías ubicadas en diferentes lugares de la geografía, así como de llevar a una mejor rentabilidad los negocios, además del hecho de ser miembros de la sociedad de la tecnología, donde es casi indispensable darse a conocer por medio de un servicio de esta índole.

Cada una de estas características da crédito a la importancia de proteger los servicios de modo eficaz implementando políticas que favorezcan su defensa, haciendo de ellas una prioridad esencial. Esta necesidad impone un dilema particular para las regiones del Tercer Mundo, en la que el acceso a herramientas que diseñen políticas de seguridad se hace más difícil debido al problema económico que resulta su adquisición; de ahí el potencial implícito en el desarrollo deherramientas que traten los servicios y las políticas implementadas en ellos.

1.4 Necesidad de una herramienta de diseño gráfico para políticas de seguridad en el país

En Cuba, se han dado pasos importantes en el desarrollo de herramientas que traten el tema, con grandes perspectivas en su comercialización. La creación de la Universidad de las Ciencias Informáticas (UCI) forma parte de este proceso de iniciativas, pues devino en la creación de varios centros de investigación y elaboración de software que se benefician con el caudal intelectual atraído por tan interesante proyecto universitario, el cual vincula el proceso educativo con el productivo.

Al calor de dichos impulsos surgen centros como el CDAE, en 2007, dedicado a brindar servicios de consultoría en tecnologías informáticas y desarrollar soluciones para organizaciones que buscan optimizar sus procesos de negocio y elevar la eficiencia operacional, empleando los modelos de Arquitectura Empresarial y SOA/BPMcomo paradigmas tecnológicos de referencia. *Cuyas principales metas y funciones son lograr la independencia tecnológica en el marco SOA/BPM, promover soluciones basadas en tecnologías libres, alcanzar una posición de liderazgo en la ejecución de proyectos SOA/BPM en la región, desarrollar investigaciones asociadas a las Arquitecturas Empresariales, Interoperabilidad, Arquitecturas Orientadas a Servicios (SOA), Gestión de Procesos de Negocio (BPM), ofrecer soporte y seguimiento a clientes de productos y soluciones desarrolladas, entre otras*[12]. La implicación del Centro en soluciones enfocadas a las Arquitecturas Empresariales y SOA/BPM hace

necesaria la implementación de políticas de seguridad en los servicios, lo cual eleva la seguridad en las aplicaciones.

1.5 Sistemas informáticos para el diseño de políticas de seguridad en el mundo

Para facilitar la implementación de las políticas de seguridad se han desarrollado varias herramientas. Estas permiten a los usuarios un mejor manejo del proceso de creación de las mismas, facilitando su uso y optimizándolo. Estas, cada vez con más frecuencia, buscan hacer más sencillo su manejo, permitiendo su uso a usuarios inexpertos que desconozcan los conocimientos técnicos necesarios para su confección. En esta característica, la facilidad de manejo por usuarios inexpertos de una aplicación, así como la riqueza de opciones en ella, radican las diferencias de algunos de los principales software presentes en el mercado, como:

- **Code Access Security PolicyTool:***Herramienta directiva de seguridad que permite a los usuarios y administradores modificar la política de seguridad para el nivel de directiva de equipo, el nivel de directiva de usuario, y el nivel de la política empresarial. Define tres niveles de política: política de máquina, política de usuario y política de empresa*[13]. Esta herramienta es muy compleja en su manejo, y considerada obsoleta para las nuevas configuraciones .NET.
- **Cisco Security PolicyBuilder**[11]: *Herramienta que permite crear una política de seguridad personalizada y adaptada a las necesidades específicas de una organización. Después de completar una breve entrevista que le ayuda a navegar a través de los principales problemas de seguridad y las preocupaciones, una política de seguridad personalizada en formato Word será enviada por correo electrónico a una dirección indicada por el cliente*[14]. Solo evalúa las necesidades e identifica problemas de seguridad, no da soluciones.
- **MotOrBAC 2:***Herramienta que incorpora la política de especificación y la administración en un mismo modelo. Es capaz de simular y analizar una política de seguridad específica mediante el modelo de OrBAC*[12][15].

[11]*Constructor de políticas de seguridad Cisco.*
[12]*Control de Acceso basado en Organización (Organization-based Access Control-OrBac, por sus siglas en inglés) es un modelo de control de acceso presentado en 2003, que permite al diseñador de una política de seguridad definirla independientemente de la implementación de la misma. Su estructura es jerárquica y determina tres entidades fundamentales sujeto, acción y objeto.*

- **Samoa:***Conjunto de herramientas de seguridad para servicios web pertenecientes al proyecto Samoa que persiguen como objetivo aprovechar los últimos avances teóricos en el análisis de los protocolos de seguridad en la creación de los servicios web XML. Algunas de dichas herramientas son WSE PolicyAdvisor para WSE 2.0 y 3.0, FS2PV, TulaFale, entre otras*[16]. Es una herramienta propietaria.
- **TulaFale:***Herramienta de seguridad para servicios web que verifica automáticamente las propiedades de autenticación de los protocolos SOAP,utilizando el cálculo pi para la escritura de las colecciones de los procesadores de SOAP que se ejecutan en paralelo*[17]. Es una herramienta propietaria.
- **Oracle Web Services Manager Security Policies**[13]**:***Herramienta que instala una capa de portabilidad en la parte superior del WebLogic Server que se integra al Oracle WebServices Manager de WS-Security en el entorno de las políticas de WebLogic Server. Esta capa proporciona la portabilidad de las políticas Oracle WSM WS-Security que se pueden utilizar para proteger los servicios web y clientes de servicios web del servidor WebLogic de JAX-WS*[18]. Es una herramienta propietaria.

Cada una de las herramientas mencionadas presenta un enfoque diferente a las necesidades de seguridad presentes en la actualidad. Pero no representan una solución factible a las necesidades del CDAE en particular, pues al analizarse de manera independiente muestran deficiencias ya descritas tales como ser propietarias, obsoletas para las nuevas tecnologías, ser complejas en su manejo solo posible mediante personal capacitado o mostrar solo reportes de vulnerabilidades y no soluciones, características que las hacen inviables a la problemática enunciada y no brindan la versatilidad acorde a las exigencias expresadas del Centro de una herramienta que sea capaz de generar políticas de seguridad válidas y de fácil confección por personal no capacitado mediante interfaces gráficas amenas.

Paralelamente a este estudio se realizó además una investigación sobre la plataforma WSO2[14]Carbon de la Suite WSO2, framework sobre el cual se pretende desplegar la aplicación propuesta. Esta es una plataforma basada completamente en código abierto la cual permite a los desarrolladores organizar

[13]*Administrador de políticas de seguridad para servicios web de Oracle.*
[14]*Compañía que desarrolla aplicaciones de software abierto enfocadas en proveer una arquitectura orientada a servicios (SOA) para desarrolladores profesionales.*

rápidamente procesos empresariales, crear aplicaciones y desarrollar servicios. [19]

Sus componentes pueden contarse en más de 175, los cuales abarcan desdeentregar mensajes y datos hasta manejar negocios, la presentación, la identidad, la seguridad, la gobernanza, la vigilancia y los servicios de gestión. Las funciones básicas de su framework poseen las mismas características comunes presentes en todos los componentes de WSO2, dentro de las que pueden mencionarse un registro incorporado, el servicio y la gestión de usuarios, así como de transporte, seguridad, entre otros. Existen también numerosos productos basados en WSO2 Carbon entre los que cabe mencionar WSO2 Governance Registry[15], WSO2 Identity Server[16] y WSO2 Application Server[17], los cuales tienen su implicación en el uso de políticas de seguridad y se describen a continuación.

- WSO2 GovernanceRegistry: Este es unaherramienta que proporciona un complemento a las herramientas SOA de WSO2 en cuanto a todo el proceso de gestión y configuración de cualquier tipo de servicio, incluyendo su descripción, inserción, búsqueda y políticas.
- WSO2 Identity Server: Este es unaherramientaque se dedica por entero al manejo de las identidades, ya sea de clientes, aplicaciones o servicios, así como otorgar su autenticación o denegarla.
- WSO2 Application Server: Este es unaherramientadedicado al manejo de aplicaciones, desde su adición o administración por consola hasta el otorgamiento de políticas de seguridad a los mismos.

1.6 Tecnologías y herramientas para el desarrollo de la solución

1.6.1 Metodología de desarrollo de software. XP

La gran variedad de herramientas y metodologías capaces de brindar soluciones eficaces presentan un gran dilema a los desarrolladores de hoy en la forma de ¿cuál metodología y herramienta emplear?, ya que las mismas determinan la calidad que tendrá un producto determinado, pues definen qué papel debe ejercer cada miembro dentro de un equipo de desarrollo, las actividades que se deben de cumplir, los artefactos a generarse en cada tarea, así como el registro de cada detalle de la información a generar.

[15] *Registro de Gobierno.*
[16] *Servidor de Identidad.*
[17] *Servidor de Aplicaciones.*

Las metodologías y técnicas existentes para el desarrollo de productos de software, son analizadas y definidas en la arquitectura de cada proyecto. Se conocen dos tipos definidos de metodologías, las robustas o tradicionales y las ágiles. La primera está pensada para usarse en proyectos de gran escala y con muchos integrantes, por lo que están diseñadas para soportar la gran complejidad técnica de estos, en los que se dificulta la sincronización y comunicación entre sus miembros. Aunque las metodologías tradiciones pueden aplicarse a proyectos pequeños la gran cantidad de roles, tareas, artefactos y documentación que generan no la hacen adecuada para este tipo de empresa. En cambio las metodologías ágiles presentan diferencias substanciales con respecto a la anterior, pues se enfocan menos en la generación de artefactos y documentos y más en el código que se genera. Además ofrecen una buena solución para proyectos variables donde los requisitos no se conocen con exactitud.

Estas características de las metodologías ágiles la hacen un candidato perfecto para el desarrollo que se propone ya que se cuenta con un proyecto pequeño (1 persona) y un tiempo de desarrollo corto. Sobre estas últimas se han desarrollado varios ejemplos con características similares, de los cuales tres de los más ágiles fueron objeto de estudio con el fin de seleccionar la metodología que guiara el proceso de desarrollo de la aplicación propuesta. A continuación se muestra dicho análisis:

- **SCRUM**[18]**:**Es una metodología desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle, la cual define un marco para la gestión de proyectos y goza de gran popularidad debido a características como su utilización en proyectos con un rápido cambio de requisitos, el desarrollo de reuniones a lo largo del ciclo de vida del proyecto, así como un desarrollo basado en iteraciones, denominadas *sprints*, con una duración de 30 días cada una. Cada una de estas iteraciones se entiende como un incremento ejecutable que se muestra al cliente.
- **ASD**[19]**:** Es una metodología impulsada por JimHighsmith cuyas principales características son ser iterativa, orientada a los componentes de software y tolerante a los cambios. Tiene un ciclo de vida dividido en tres fases principales: especulación, colaboración y aprendizaje. En la primera fase se inicia elproyecto y se planifican las características del software, en la segunda fase se desarrollan lascaracterísticas y finalmente en la tercera fase se revisa su calidad, y se entrega al cliente.

[18]*Es una terminología tomada del rugby, donde se utiliza este término para representar al esfuerzo esencial de un equipo por lograr el objetivo de mover el balón hacia adelante en busca de la meta.*
[19]*Acrónimo de Adaptive Software Development.*

Larevisión de los componentes sirve para aprender de los errores y volver a iniciar elciclo de desarrollo.

- Y por último, **XP:** la cual es una disciplina de desarrollo de software creada por Kent Beck[20], centrada principalmente en la simplicidad, la comunicación y el reciclado continuo de código, preocupándose por el aprendizaje de los desarrolladores y propiciando la existencia de un buen clima de trabajo. *Sus objetivos son muy simples pues están centrados en la satisfacción del cliente y potenciar al máximo el trabajo en equipo*[20]. Además es una metodología que presenta seis fases principales en su ciclo de vida muy bien definidas, y que cuenta con una gran bibliografía, proyectos y expertos que la desarrollan.

Tabla 1.1 Ranking de Agilidad entre las Metodologías escogidas

Metodologías	SCRUM	ASD	XP
Sistema como algocambiante	5	5	**5**
Colaboración	5	5	**5**
Características de la Metodología (CM)			
- Resultados	5	5	**5**
- Simplicidad	5	4	**5**
- Adaptabilidad	4	5	**3**
- Excelencia técnica	3	3	**4**
- Prácticas de colaboración	4	5	**5**
Media CM	4.2	4.4	**4.4**
Media Total	4.7	4.8	**4.8**

Cada una de estas ventajas anteriormente expuestas determinan que XP sea una metodología ágil, que genera muy pocos artefactos acelerando el proceso de culminación de un producto, además de poseer otras características como ser de desarrollo iterativo e incremental, tener simplicidad en su código, utilizar la programación en pareja y la reutilización de código. Cada una de las cuales resultan en ventajas importantes para proyectos de pequeña envergadura como son la simplicidad, la disminución de la traza de errores y no menos importante, un producto de alta calidad en el mínimo de tiempo. Por

[20]*Programador que ha trabajado en múltiples empresas y que actualmente lo hace como programador en la conocida empresa automovilística DaimlerChrysler.*

todo lo antes expuesto hacen de la Programación Extrema la metodología escogida para desarrollar el software propuesto.

XP. Fases de ciclo de vida, roles, actividades y características.

Esta metodología siendo simple solo engloba seis fases dentro de su ciclo de vida las cuales se mencionan a continuación:

I. **Exploración:** En esta fase se realiza la reunión con los clientes y en la cual son recogidas las llamadas userstories[21](HU, por sus siglas en español) que determinan los requerimientos a implementar por los programadores. Además se utiliza este período para la familiarización por parte del equipo desarrollador con las tecnologías, herramientas y prácticas a utilizarse por el proyecto.

II. **Planificación de la Entrega (*release*[22]):** En esta fase son establecidas por el cliente las prioridades para los requerimientos recogidos, realizándose una estimación del esfuerzo necesario en cada una de ellas. En este período son plasmadas las iteraciones a realizarse a través de un Plan de Entrega.

III. **Iteraciones:** Fase utilizada para desarrollar las iteraciones definidas en el Plan de Entrega, donde se escogen las arquitecturas afines a cada iteración, así como las iteraciones que fuerzan la creación de dichas arquitecturas.

IV. **Producción:** Fase utilizada para la realización de revisiones de rendimiento, pruebas y para la inclusión de nuevas características al producto, antes de ser trasladado al entorno del cliente.

V. **Mantenimiento:** Esta fase se implementa en paralelo con la fase anterior, o sea, al mismo tiempo en que se implementan nuevas iteraciones dadas las características propias de esta metodología donde la economía en el tiempo es fundamental.

VI. **Muerte del Proyecto:** En esta fase se define el final de las iteraciones, la cual requiere que se cumplan las necesidades del cliente o que el sistema no genere los beneficios requeridos por este, generándose la documentación final y más ningún cambio en la arquitectura.

[21] *Historias de usuarios.*
[22] *Producto terminado.*

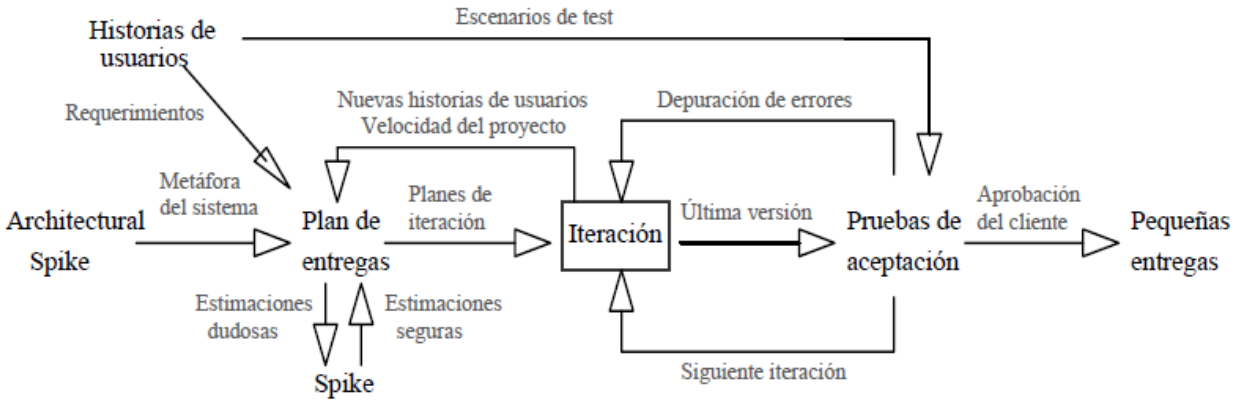


Figura 1.1 Trabajando con XP [21]

Siendo los roles a desempeñar los siguientes:

Administrador (programador): Su misión es administrar el ambiente de programación. Realiza las actividades de Configurar el Ambiente de Programación.

Programador: Su misión es ser responsable de implementar el código para dar soporte a los requisitos del cliente. Realiza las actividades de Definir el Estimado de la Tarea, Estimado del Requisito del Cliente, entre otros, generando los artefactos de XP build, XP Unit test, entre otros.

Probador: Su misión es ayudar al cliente a definir y escribir pruebas de aceptación para los requisitos. Realiza las actividades de Correr las Pruebas del cliente, Automatizar las Pruebas del cliente y Configurar el Ambiente del Probador.

Cliente: Su misión es ser responsable de definir cuál es el producto correcto a construir, determinar las características del mismo y asegurarse de que el producto realmente satisface sus requerimientos. Realiza las actividades de Ajustar las iteraciones de las actividades, Definir la Iteración, Definir la Visión del documento, Escribir los requisitos, entre otros, generando los artefactos de Requisitos, Prueba del Cliente, Plan de Iteraciones, Plan de Entrega, entre otros.

Encargado de Seguimiento: Su misión es medir y comunicar el progreso del proyecto. Realiza las actividades de Seguimiento del Progreso de la Iteración y Seguimiento del Progreso del *release* del producto.

Entrenador (coaching): Su misión es ayudar a mantener la disciplina y aprendizaje del equipo. Realiza las actividades de Explicar Proceso, Mejorar las habilidades del Equipo, Resolver Conflictos, entre otros.

Además de la metodología se procede entonces a la elección de las herramientas necesarias para apoyar el diseño de la aplicación, elección que está sujeta a las características de diseño de la herramienta sobre la que se despliega la aplicación a desarrollar.

1.6.2 Lenguaje de Programación

Es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo, el cual presenta como ventajas *que es mucho más fácil de comprender que un lenguaje máquina y que permite mayor portabilidad, es decir, que puede adaptarse fácilmente para ejecutarse en diferentes tipos de equipos*[22].

1.6.2.1 Java

Es un lenguaje de Programación Orientado a Objetos (POO) creado a principios de los ’90 y desarrollado por Sun Microsystems. Es un lenguaje de propósito general, lo cual permite crear diversos tipos de aplicaciones con él; *radicando su mayor éxito en Internet, con el uso de los applets, las aplicaciones cliente – servidor o las Java Server Pages*[23]. *Se distingue por ser capaz de gestionar la memoriaautomáticamente, no permite el uso de técnicas de programación inadecuadas; posee mecanismos de seguridad incorporados, los cuales limitan el acceso a recursos de las máquinas donde se ejecuta; incorpora herramientas de documentación; es multihilos (multithreading)*[23]. Además de constituir el lenguaje sobre el que la Suite del WSO2, utilizada por el Centro, se construyó. Todas estas características lo califican como el lenguaje de programación idóneo para utilizar.

1.6.3 Herramientas CASE. Visual Paradigm

[23] *JSP:Tecnología JAVA que permite generar contenido dinámico para Web, en forma de documentos HTML, XML o de otro tipo.*

Las herramientas CASE*representan una forma que permite modelar los procesos de negocios de las empresas y desarrollar los sistemas de información gerenciales*[24].

Sin importar su arquitectura, tales herramientas deben poseer propiedadescomo las siguientes:

- Una interfaz gráfica y textual, que le permita al usuario manejar los objetos de diseño.
- Contar con un diccionario de datos, a fin de rastrear y controlar los objetos diseñados.
- Disponer de un conjunto de herramientas que permitan chequear las reglas del diseño y analizarsu lógica.

La herramienta escogida tras una amplia labor investigativa fue Visual Paradigm, ya que es una de las herramientas CASE más completas que existe en el mercado actualmente, además de que la universidad posee la licencia para su utilización.

Visual Paradigm

Visual Paradigm es una herramienta UML profesional con características graficas muy cómodasque *soporta el ciclo de vida completo del desarrollo de software*[25]: análisis y diseño orientado a objetos (OO), construcción, pruebas y despliegue. Contiene un conjunto de ayudas para el desarrollo de software, así como que permite la elaboración de todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación de diferentes extensiones (.pdf, .doc, entre otros). Se caracteriza por ser software libre, ser multiplataforma (Windows y Linux), ser de diseño centrado en casos de uso y enfocado al negocio lo que genera un software de mayor calidad, ser fácil de instalar y actualizar, así como que es de licencia gratuita y comercial.Debido a todo lo antes mencionado, se puede concluir que Visual Paradigm se escogió por su integración con varios IDE's de desarrollo como Eclipse y NetBeans, además de su robustez, usabilidad, accesibilidad, sencillez y portabilidad.

1.6.4 Ambiente de desarrollo. Eclipse IDE

Es una plataforma universal o Entorno Integrado de Desarrollo (IntegratedDevelopmentEnvironment–IDE, por sus siglas en inglés)de código abierto multiplataforma desarrollado inicialmente por IBM y ahora por la Fundación Eclipse, *organización independiente, sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios*[26]. Presenta una arquitectura de plug-insque permite integrar diversos lenguajes sobre un mismo IDE e introducir otras aplicaciones accesorias.Eclipse es soportado por los principales sistemas operativos

tales como Linux o Windows. Dispone de un Editor de Texto con resaltado de sintaxis, compilación en tiempo real, y además contiene pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, asistentes (wizards) para creación de proyectos, clases, pruebas, entre otros, y refactorización.

Entre sus principales características como IDE de Java se encuentran:

- Editor visual con sintaxis coloreada.
- Compilación incremental de código.
- Modifica e inspecciona valores de variables.
- Avisa de los errores cometidos mediante una ventana secundaria.
- Depura código que resida en una máquina remota.

Teniendo en cuenta las características expuestas, se selecciona Eclipse (Helios) como IDE de desarrollo debido a la flexibilidad de su editor de código, las características de integración con servidores web quefacilitan el desarrollo de aplicaciones de este tipo, además de la experiencia conel uso de este IDE con la que cuenta el equipo de desarrollo fueron otros factores que influyeron en latoma de esta decisión.

1.6.5 Frameworks

Un framework es una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, es una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.

1.6.5.1 Vaadin

Framework que utiliza las librerías Java dedicadas al diseño y mantenimiento de interfaces de usuarios para aplicaciones web fáciles y de alta calidad. *La idea principal* en el modelo de diseño de esta herramienta *es permitir el trabajo de la programación web mediante interfaces de usuarios semejantes a cualquier aplicación desktop*[24] *de Java*, o sea, *sin utilizar lenguajes convencionales de la*

[24] *Escritorio.*

programación web como HTML o Javascript, además de permitir la utilización de paquetes como AWT, Swing o SWT[27].

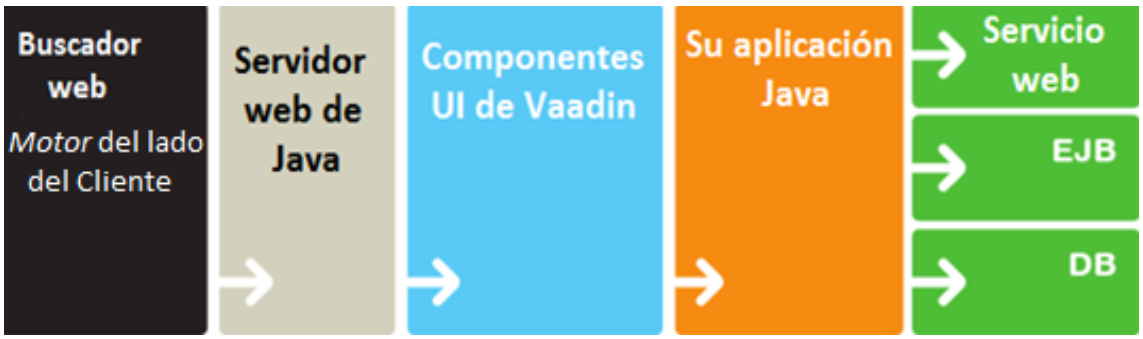


Figura 1.2 Arquitectura general de Vaadin [27]

Vaadin fue diseñado por la empresa finlandesa IT Mill junto a una comunidad que colabora en su desarrollo, mediante una licencia Apache 2.0, lo que permite un modelo de código abierto fácilmente adquirible. Este framework presenta varias ventajas significativas como son:

- Programación en JAVA.
- No necesita programación en JavaScript.
- Permite la creación de aplicaciones dinámicas de Internet que son tan sensibles e interactivascomo aplicaciones de escritorio.
- Integración con Spring, Hibernate, High Charts, entre otros.
- Permite el desarrollo de aplicaciones web vistosas e interactivas sin requerir para ello plugins en el servidor web.
- Permite a los desarrolladores enfocarse principalmente en la lógica y dejar el HTML,CSS, JavaScript y XML al framework.
- Puede integrarse al IDE Eclipse y ofrece un editor visual (en Beta) y funcionalidades para crear Componentes, Wizards y ayuda integrada.

Es por esto que se decidió adoptar esta herramienta como uno de los frameworks de desarrollo.

1.6.6 Librerías

Es un conjunto de subprogramas utilizados para desarrollar software, o sea, un conjunto de funciones pero sin una función *main()*. Estas contienen código y datosque proporcionan servicios a programas independientes, por lo que la mayoría de las aplicaciones y sistemas operativos modernos las utilizan.

1.6.6.1 Dom4j

Es una librería de código abierto para JAVA utilizada en el trabajo con XML, XPath[25]y XSLT[26] que se distribuye bajo licencia BSD[27]. Además es compatible con los estándares de DOM[28], SAX[29] y JAXP[30]. Por lo tanto se considera idónea su utilización en la herramienta.

1.6.7 Lenguaje de Modelado. UML

Utilizar un lenguaje de modelado nos permite verificar efectivamente el funcionamiento adecuado que el modelo debe poseer, pues existen menos medios de verificación, lo cual hace al modelo más sencillo que la propia programación.El Lenguaje Unificado de Modelado (*UnifiedModelingLanguage – UML*[28], por sus siglas en inglés) es un lenguaje estándar que permite *especificar, visualizar, construir y documentar*[28] todos los elementos que forman un *sistema de software*[28] orientado a objetos. Por lo que su propósito consiste en elaborar los artefactos de un sistema a través de las distintas etapas de su ciclo de vida, principalmente durante el análisis y el diseño del mismo.

El modelado mediante este lenguaje permite establecer un conjunto de requerimientos y estructuras necesarias para plasmar un sistema de software previo a la escritura del código. El tiempo invertido en el desarrollo de la arquitectura se minimiza, la trazabilidad y documentación del proyecto se realiza de una forma ordenada y guiada por los casos de uso, pero lo más importante es la notable efectividad y

[25]*Acrónimo deXML PathLanguage (en español ruta para lenguaje XML) es un lenguaje que permite construir expresiones que recorren y procesan un documento XML. La idea es parecida a las expresiones regulares para seleccionar partes de un texto sin atributos.*

[26]*Acrónimo de Extensible StylesheetLanguageTransformations (en español Transformaciones para las Hojas de Estilo XLS) es un estándar de la organización W3C que presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML.*

[27]*Acrónimo de Berkeley Software Distribution(en españolDistribución de Software Berkeley) es un sistema operativo derivado del sistema Unix nacido a partir de los aportes realizados a ese sistema por la Universidad de California en Berkeley.*

[28]*Acrónimo de DocumentObjectModel(en español Modelo de Objetos del Documento o Modelo en Objetos para la Representación de Documentos) es, esencialmente, una interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML.*

[29]*Acrónimo deSimple API for XML (en español API simple para XML), originalmente, una API únicamente para el lenguaje de programación Java, que después se convirtió en la API estándar de facto para usar XML en JAVA.*

[30]*Acrónimo de Java Api for XML Processing(en español API Java para el procesamiento de XML). API Java definido por Sun Microsystems sirve para la manipulación y el tratamiento de archivos XML.*

productividad que se consigue en labores de diseño arquitectónico y mantenimiento, haciendo uso de UML frente a la realización de las mismas tareas, en ausencia de modelos. Por lo que se considera beneficiosa su utilización en la herramienta.

1.7 Conclusiones Parciales del Capítulo

Luego de haber realizado un estudio de los sistemas informáticos para el diseño de políticas de seguridad existentes hasta el momento en el mundo, analizando las ventajas y desventajas de cada uno y dadas las inconveniencias que estos presentan, proponemos la realización de un sistema informático para el diseño de políticas de seguridad mediante el uso de paneles gráficos para el Centro de Consultoría y Desarrollo de Arquitecturas Empresariales de la Universidad de las Ciencias Informáticas.

Para el cual se usará una metodología XP y la herramienta CASE Visual Paradigm para el análisis, diseño y documentación del sistema, usando UML como lenguaje de modelado visual, además de los lenguajes de programación Java y frameworks como Vaadin, así como las librerías dom4j dentro del IDE de programación Eclipse.

Capítulo 2: Análisis y diseño del sistema

2.1 Introducción

En este capítulo, se realiza un análisis de las necesidades planteadas por el Centro con respecto a la confección de una herramienta gráfica para la elaboración de políticas de seguridad, y así lograr una mayor comprensión de la misma y sus principales características. El desarrollo de la herramienta se basa en la metodología de Programación Extrema por lo que se propone seguir los pasos definidos por dicha metodología para el proceso de desarrollo, haciendo uso del Lenguaje Unificado de Modelado para la modelación de los artefactos que se generen. Ha constituido una gran ayuda el empleo de la herramienta CASEVisual Paradigm, que asiste al desarrollo de software para una mayor calidad de este.

Se presenta el estado actual del negocio y sus reglas, en las cuales se define la problemática a la cual la aplicación propuesta dará solución. Además se describen las historias de usuarios las cuales describen las características con las que se debe dotar al sistema, que permitirán enumerarlos requisitos funcionales y no funcionales con los que deberá contar el sistema. Por otra parte, en el capítulo se brinda una concepción global del sistema propuesto.

2.2 Estado actual del Negocio

Actualmente en el Centro CDAE existe personal instruido en el diseño e implementación de políticas de seguridad, pero su cantidad es exigua siendo entonces el principal objetivo facilitar el trabajo con ellas lo mejor posible haciendo su uso más factible a personal menos capacitado. Entonces, se hace necesario diseñar una nueva herramienta para el manejo de las políticas de seguridad ya que los 16 escenarios contenidos en la Suite WSO2, a pesar de que contienen la descripción de una política, ya sea de transporte o de mensaje, solo pueden modificarsede forma manual, haciendo entonces esta función difícil para personal no calificado. Este hecho en sí, plantea una disyuntiva ya que constituye una limitante para el caso de crear una política propia, con unos requisitos de diseño ajenos a los ya establecidos por las políticas ubicadas en la Suite.

Luego, esto plantea otro problema relacionado con el desconocimiento y falta de experiencia del personal del centro para el manejo de dichas políticas, las cuales presentan un lenguaje construido sobre XML y una estructura para su modelado de difícil comprensión, además de que las herramientas existentes que las modelan y diseñan son imposibles de obtener, ya sea por ser herramientas caras, propietarias, complejas o por no satisfacer los requerimientos propuestos.

2.3 Reglas del Negocio

- La política deberá guardarse en una ubicación física.
- Permitir la creación de diferentes escenarios de políticas de seguridad.
- La política creada deberá ser validada, a través de un fichero de validación .xsd, antes de guardarse.
- La aplicación permitirá la creación de políticas de seguridad mediante el uso de interfaces gráficas amenas.

2.4 Exploración

La fase de exploración es la fase en la que se define el alcance del proyecto, además de realizar la entrevista con los clientes, definiéndose las llamadas Historias de Usuarios (HU) y con ellas la determinación de los requisitos planteados por el cliente, así como la que el equipo de desarrollo se familiariza con lasherramientas, tecnologías y prácticas que se utilizarán en el proyecto. Esta es una fase en la cual su duración oscila de pocas semanas a pocos meses, dependiendo del conocimiento que tenga el equipo desarrollador.

Historias de Usuarios

La definición de dichas historias permite especificar los requerimientos del software. En ellas se describen brevemente las características con las que el sistema debe contar, o sea, los requisitos planteados por el cliente, ya sean funcionales o no. Su utilización debe ser flexible y lo más comprensible posible, permitiendo su implementación por los programadores en poco tiempo.

La respuesta a dichas historias debe estar provista de metas o tareas de programación para el programador, además de que cada historia terminada debe contar con una serie de pruebas de unidad que aseguren el correcto funcionamiento de los componentes. Una vez terminada cada historia, el sistema debe ser presentado al cliente para la realización de pruebas de aceptación de la funcionalidad

implementada. Además al reunirse un número suficiente de funcionalidades válidas de la aplicación se produce una liberación del producto, la cual es una versión funcional de la aplicación que aporta valor al negocio y que debe ser mantenida a la par que se desarrollan las funcionalidades siguientes.

Su estructura queda determinada de la siguiente forma:

- **Nombre:** Nombre descriptivo de la HU.
- **Prioridad:** Grado de prioridad que le asigna el cliente a la HU en dependencia de sus necesidades. Los valores que puede tomar son Alta, Media o Baja.
- **Complejidad:** Grado de complejidad que le asigna el equipo de desarrollo a la HUluego de analizarla. Los valores que puede tomar son Alta, Media o Baja.
- **Estimación:** Unidades de tiempo estimadas por el equipo de desarrollo para darle cumplimiento a la HU.
- **Iteración:** Número de la iteración en la cual será implementada la HU.
- **Descripción:** Descripción simple que brinda el cliente sobre lo que debe hacer la funcionalidad en cuestión.

Durante este proceso se identificaron varias historias de usuarios, de las cuales se mostraran las más importantes:

Tabla 2.2 HU1 Generar Política de tipo Transporte

Historia de Usuario	
Número: HU1	Nombre: Generar Política de tipo Transporte
Usuario: Denys López	
Prioridad en Negocio: Alta	Riesgo en Desarrollo: Alta
Iteración Asignada: 1	
Descripción: La presente historia de usuario tiene como objetivo: generar el código que permita crear políticas de transporte.	
Observaciones: Hace referencia a los requisitos funcionales 4, 9 y 10.	

Tabla 2.3 HU2 Generar Política de tipo Mensaje

Historia de Usuario	
Número: HU2	Nombre: Generar Política de tipo Mensaje

Usuario: Denys López	
Prioridad en Negocio: Alta	Riesgo en Desarrollo: Alta
Iteración Asignada: 1	
Descripción: La presente historia de usuario tiene como objetivo: generar el código que permita crear políticas de mensaje.	
Observaciones: Hace referencia a los requisitos funcionales 4, 9 y 10.	

Tabla 2.4 HU3 Generar interfaz de la Aplicación

Historia de Usuario	
Número: HU3	Nombre: Generar interfaz de la Aplicación
Usuario: Jorge Infante	
Prioridad en Negocio: Alta	Riesgo en Desarrollo: Medio
Iteración Asignada: 2	
Descripción: La presente historia de usuario tiene como objetivo: desarrollar la interfaz de la aplicación propuesta.	
Observaciones: Hace referencia a los requisitos funcionales 2, 3, 5, 12.	
Prototipo de Interfaz: 	

Tabla 2.4 HU4 Generar interfaz para Modelo Básico

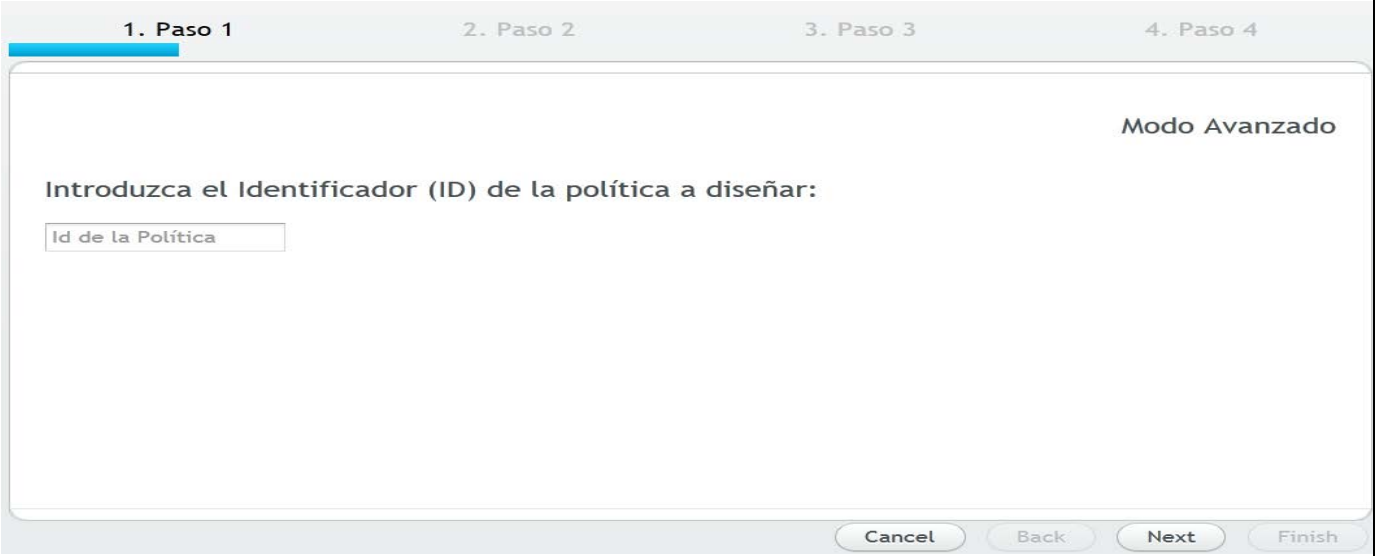
Historia de Usuario	
Número: HU4	Nombre: Generar interfaz para Modelo Básico

Usuario: Jorge Infante	
Prioridad en Negocio: Alta	Riesgo en Desarrollo: Medio
Iteración Asignada: 3	
Descripción: La presente historia de usuario tiene como objetivo: desarrollar la interfaz que permita generar políticas,a usuarios sin conocimientos amplios acerca de estas, a través de paneles gráficos. Debe permitir generar o eliminar políticas de seguridad.	
Observaciones: Hace referencia a los requisitos funcionales 1, 2, 4, 7, 8, 11 y 12.	
Prototipo de Interfaz: 	

Tabla 2.5 HU5 Generar interfaz para Modelo Avanzado

Historia de Usuario	
Número: HU5	Nombre: Generar interfazpara Modelo Avanzado
Usuario: Arturo César Áreas	
Prioridad en Negocio: Alta	Riesgo en Desarrollo: Media
Iteración Asignada: 4	
Descripción: La presente historia de usuario tiene como objetivo: desarrollar la interfaz que permita generar políticas, a usuarios más familiarizados con estas, a través de paneles. Debe permitir generar o eliminar políticas de seguridad.	
Observaciones: Hace referencia a los requisitos funcionales 1, 2, 4, 7, 8, 11 y 12.	

Prototipo de Interfaz:



2.5 Requisitos funcionales del Sistema

- RF 1: El sistema deberá permitir al usuario la opción de crear nueva política.
- RF 2: El sistema deberá mostrar al usuario los tipos de política a diseñar.
- RF 3: El sistema deberá ser capaz de mostrar paneles de selección en dependencia de los criterios seleccionados por el usuario.
- RF 4: El sistema deberá generar el XML de la política escrita por el usuario.
- RF 4.1:Generar política de tipo mensaje.
  - RF 4.1.1: El sistema deberá mostrar opciones predefinidas.
  - RF 4.1.2: El sistema deberá permitir escoger entrar las opciones predefinidas mostradas.
  - RF 4.1.3: El sistema deberá permitir agregar otras opciones, además de las predefinidas.
  - RF 4.1.4: El sistema verificará si las opciones agregadas por el cliente son válidas.
- RF 4.2: Generar política de tipo transporte.
  - RF 4.2.1: El sistema deberá mostrar opciones predefinidas.
  - RF 4.2.2: El sistema deberá permitir escoger entre las opciones predefinidas mostradas.
  - RF 4.2.3: El sistema deberá permitir agregar otras opciones, además de las predefinidas.

- RF 4.2.4: El sistema verificará si las opciones agregadas por el cliente son válidas.
- RF 5: El sistema deberá generar políticas predefinidas.
- RF 6: El sistema deberá ser capaz de realizar modificaciones mediante paneles gráficos.
- RF 7: El sistema deberá permitir salvar la política diseñada en una ubicación física seleccionada por el usuario.
- RF 8: El sistema deberá permitir al usuario cancelar el diseño de una política.
- RF 9: El sistema deberá mostrar una ayuda especializada en cada paso de la confección de la política.

2.6 Requisitos no funcionales del Sistema

2.6.1 RNF de Apariencia o Interfaz Interna.

- Las páginas no poseerán muchas imágenes.
- Todas las páginas poseerán una ayuda en cada parámetro a modificar.
- Interfaz rápida y funcional.

2.6.2 RNF de Usabilidad.

- La aplicación tendrá un ambiente sencillo y amigable, fácil de manejar para cualquier usuario, sobre todo aquel sin un conocimiento avanzado en la confección de políticas de seguridad.

2.6.3 RNF de Seguridad.

- Todos los requerimientos de seguridad de la aplicación estarán supeditados a aquellos presentes en la Suite WSO2, a la cual pertenecerá la aplicación.

2.6.4 RNF de Software.

- En las computadoras de los clientes deberán existir las mismas restricciones presentes en los sistemas operativos (SO) incluyendo un navegador asociado al SO para la visualización de las interfaces web.

2.6.5 RNF Restricciones en el Diseño y la Implementación

- Se utilizará como herramienta CASE Visual Paradigm para el modelado de los artefactos que se generen en cada fase de la metodología a utilizar.
- Se utilizará UML como lenguaje de modelado.
- Se utilizará el IDE de desarrollo Eclipse para el diseño de la aplicación.
- Se usará como lenguaje de programación JAVA y Javascript, así como los frameworks SpringMVC y Vaadin, y las librerías dom4j y JQuery.

2.7 Planificación

En esta fase se establecen las prioridades para los requerimientos recogidos, la composición de las versiones –qué debería incluir cada una de ellas– y la fecha de cada una de dichas versiones, además es estimada la duración requerida para implementar las funcionalidades deseadas por el cliente y se realiza una planificación detallada dentro de cada versión. De aquí las potencialidades de XP como metodología de desarrollo de software ya que no solo es centrada en el código sino que también es útil para la gestión de proyectos de software.

XP no tiene métricas predeterminadas, sino que son libres, lo que permite utilizar cualquier criterio para medir el alcance del proyecto. La métrica a utilizar en este caso es la que toma como medida el punto, el cual se considera como una semana ideal de trabajo, donde se trabaja el tiempo planeado sin ningún tipo de interrupción.La estimación incluye todo el esfuerzo asociado a la implementación de la historia de usuario.

Para el buen desarrollo del sistema propuesto, se realizó una estimación de esfuerzo para cada una de las historias de usuario anteriormente descritas.

Tabla 2.6 Esfuerzo estimado para HU

No.	Historias de Usuarios	Esfuerzo estimado
HU1	Generar políticas de tipo Transporte	8
HU2	Generar políticas de tipo Mensaje	8
HU3	Generar interfaz de la Aplicación	3
HU4	Generar interfaz para Modelo Básico	3
HU5	Generar interfaz para Modelo Avanzado	4
	Total	26

2.7.1 Plan de Duración de las Iteraciones

A continuación se presenta el plan de duración de iteraciones. Este plan tiene como meta el mostrar la duración de cada iteración, así como el orden en que serán implementadas las historias de usuario en cada una de las mismas.

Tabla 2.7 Plan de duración de las iteraciones

Iteración	Orden de las historias de usuario a implementar	Duración total de la Iteración
1	Generar política de tipo Transporte Generar política de tipo Mensaje	16
2	Generar interfaz para la Aplicación	3
3	Generar interfaz para Modelo Básico	3
4	Generar interfaz para Modelo Avanzado	4
	Total de semanas	26

2.7.2 Plan de Entregas

En el plan de entregas se establecen los plazos de entregas de las liberaciones del producto, el cual contribuye a la organización del trabajo.

Tabla 2.8 Plan de Entrega de las Iteraciones

Número de Iteraciones	1ra Iteración – 1ra semana de enero	2da Iteración – 2da semana de febrero	3ra Iteración – 1ra semana de abril	4ta Iteración – 2da Semana de mayo
Versiones	0.1	1.0	1.1	2.0

2.8 Iteraciones

Esta es la fase utilizada para desarrollar las iteraciones definidas en el Plan de Entrega, donde se escogen las arquitecturas afines a cada iteración, así como las iteraciones que fuerzan la creación de dichas arquitecturas y donde se genera todo el código necesario para satisfacer los requerimientos recogidos en las Historias de Usuarios. Luego del proceso de identificación y descripción de las

historias de usuarios y de la estimación del esfuerzo para realizar cada una de dichas historias se procede a especificar qué historias serán implementadas por cada iteración del sistema.

Iteración 1

En esta iteración se implementarán las historias de usuario de mayor prioridad para el cliente, y por tanto, en el proyecto. Al finalizar esta iteración se contará con la implementación necesaria para el desarrollo de las funcionalidades descritas en las historias de usuario 3 y 4, referentes a los modelos Básico y Avanzado. Esta iteración tiene como principal objetivo mostrarle al cliente el desarrollo del código de la aplicación, para comprobar su funcionamiento.

Tareas de Programación para la Iteración 1:

Tabla 2.9 Tarea de Programación #1 Iteración 1

HU Generar políticas de tipo Transporte	
Número de tarea: 1	**Número de HU:** 1
Nombre de la tarea: Desarrollar código para políticas de tipo Transporte.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 8
Fecha inicio: 20 de noviembre del 2011	**Fecha fin:**
Descripción: El desarrollador implementa el código para políticas de tipo Transporte.	

Tabla 2.10 Tarea de Programación #2 Iteración 1

HU Generar políticas de tipo Mensaje	
Número de tarea: 2	**Número de HU:** 2
Nombre de la tarea: Desarrollar código para políticas de tipo Mensaje.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 8
Fecha inicio:	**Fecha fin:**
Descripción: El desarrollador implementa el código para políticas de tipo Mensaje.	

Iteración 2

El objetivo de esta iteración es el desarrollo de la interfaz principal de la aplicación. Al término de la misma se tendrán implementadas las funcionalidades descritas en la historia de usuario 3, referente al diseño de la interfaz de inicio para el sistema propuesto. Esta iteración tiene como principal objetivo mostrarle al cliente cómo va quedando la aplicación, para comprobar el grado de aceptación que tiene el producto con respecto al cliente.

Tareas de Programación para la Iteración 2:

Tabla 2.11 Tarea de Programación #1 Iteración 2

HU Generar interfaz para la Aplicación	
Número de tarea: 1	**Número de HU:** 3
Nombre de la tarea: Diseñar interfaz de inicio para la aplicación.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio:	**Fecha fin:**
Descripción: El diseñador modela la interfaz de inicio de la aplicación.	

Tabla 2.12 Tarea de Programación #2 Iteración 2

HU Generar interfaz para la Aplicación	
Número de tarea: 2	**Número de HU:** 3
Nombre de la tarea: Programar los elementos de la interfaz de inicio de la aplicación.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio:	**Fecha fin:**
Descripción: El desarrollador implementa los elementos de la interfaz de inicio de la aplicación.	

Iteración 3

El objetivo de esta iteración es el desarrollo de una de las interfaces principales de la aplicación, el Modelo Básico, que permitirá su despliegue final. Al término de la misma se tendrán implementadas las funcionalidades descritas en la historia de usuario 4, referente al Modelo Básico. Esta iteración tiene como principal objetivo mostrarle al cliente cómo va quedando la aplicación, para comprobar el grado de aceptación que tiene el producto con respecto al cliente.Para un mejor entendimiento observar en los Anexos Diagrama de Secuencias para Modelo Básico pág. 57.

Tareas de Programación para la Iteración 3:

Tabla 2.13 Tarea de Programación #1 Iteración 3

HU Generar interfaz para modelo Básico	
Número de tarea: 1	**Número de HU:** 4
Nombre de la tarea: Diseñar interfaz para el modelo Básico.	

Tipo de tarea: Desarrollo	**Puntos estimados: 0.5**
Fecha inicio:	**Fecha fin:**
Descripción: El diseñador modela la interfaz del modelo Básico.	

Tabla 2.14 Tarea de Programación #2 Iteración 3

HU Generar interfaz para modelo Básico	
Número de tarea: 2	**Número de HU:** 4
Nombre de la tarea: Programar los elementos del modelo Básico.	
Tipo de tarea: Desarrollo	**Puntos estimados: 2.5**
Fecha inicio:	**Fecha fin:**
Descripción: El desarrollador implementa los elementos que modelan la política para el modelo Básico.	

Iteración 4

El objetivo de esta iteración es el desarrollo de una de las interfaces principales de la aplicación, el Modelo Avanzado, que permitirán su despliegue final. Al término de la misma se tendrán implementadas las funcionalidades descritas en la historia de usuario 5, referente al Modelo Avanzado. Esta iteración tiene como principal objetivo comprobar el grado de aceptación que tiene el producto con respecto al cliente.Para un mejor entendimiento observar en los Anexos Diagrama de Secuencias para Modelo Avanzado pág. 57.

Tareas de Programación para la Iteración 4:

Tabla 2.15 Tarea de Programación #1 Iteración 4

HU Generar interfaz para modelo Avanzado	
Número de tarea: 1	**Número de HU:** 5
Nombre de la tarea: Diseñar interfaz para el modelo Avanzado.	
Tipo de tarea: Desarrollo	**Puntos estimados: 0.5**
Fecha inicio:	**Fecha fin:**
Descripción: El diseñador modela la interfaz de inicio del modelo Avanzado.	

Tabla 16 Tarea de Programación #2 Iteración 4

HU Generar interfaz para modelo Avanzado	
Número de tarea: 2	**Número de HU:** 5
Nombre de la tarea: Programar los elementos del modelo Avanzado.	
Tipo de tarea: Desarrollo	**Puntos estimados: 2.5**
Fecha inicio:	**Fecha fin:**
Descripción: El desarrollador implementa los elementos que modelan la política para el modelo Avanzado.	

2.9 Definiciones del Diseño

La mejor solución para el diseño de un software es la que se basa en cumplir con las expectativas del cliente de manera exitosa, presentando la menor cantidad de clases y métodos posibles garantizando su usabilidad. Siguiendo los principios de la metodología escogida se utiliza la técnica para el modelamiento de clases representada por la Tarjetas CRC[31], las cuales permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica. El uso de esta técnica permite guiar el diseño de la aplicación a través de las clases que se examinan y filtran en base a sus responsabilidades para con el sistema, así como las clases con las que se relacionan para completar dichas responsabilidades. Esta representación de las clases y sus relaciones también puede ser observada a través de los Anexos Diagrama de Clases con estereotipos web pág. 56.

Las partes que componen una tarjeta CRC son las siguientes:

- **Clase:** Nombre de la clase con que se está modelando.
- **Súper Clase:** Nombre de la clase padre en la herencia.
- **Sub Clase(s):** Nombre de las clases hijas en la herencia.
- **Responsabilidades:** Descripción breve del propósito de la clase.
- **Colaboraciones:** Indica las relaciones entre clases para cumplir con la responsabilidad.

[31] *Acrónimo para Class, Responsabilities and Collaboration, CRC por sus siglas en ingles, o Clases, Responsabilidades y Colaboraciones, por sus siglas en español.*

Tabla 2.17 Tarjeta CRC de la Clase Aplicación

Tarjeta CRC	
Clase: Aplicación. **Súper clase:** **Subclase(s):**	
Responsabilidades	Colaboraciones
➢ Crear Política Tipo Avanzado ➢ Crear Política Tipo Básico	➢ Wizard ➢ Básico

Tabla 2.18 Tarjeta CRC de la Clase Wizard

Tarjeta CRC	
Clase: Wizard **Súper clase:** **Subclase(s):**	
Responsabilidades	Colaboraciones
➢ Crear Política Tipo Avanzado	➢ Aplicación

Tabla 2.19 Tarjeta CRC de la Clase Basico

Tarjeta CRC	
Clase: Básico **Súper clase:** **Subclase(s):**	
Responsabilidades	Colaboraciones
➢ Crear Política Tipo Básico	➢ Aplicación

2.9 Patrones de Diseño

Los patrones de diseño son una ayuda innegable al desarrollo de cualquier software ya que son una solución genérica ya probada y estudiada a un problema determinado. Se dividen en tres grupos importantes:

- **Creacionales:** Inicialización y configuración de objetos.
- **Estructurales:** Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
- **De Comportamiento:** Describen la comunicación entre objetos o clases.

Controlador Frontal (Front Controller): Es un patrón de tipo estructural queestipula que, haciendo uso de la arquitectura MVC, es conveniente utilizar un único punto para gestionar todas las peticiones, incluyendo invocaciones a servicios de seguridad, gestión de excepciones, selección de la siguiente vista, entre otras, lo que hace mucho más robusta la aplicación aislando cada uno de estos aspectos del resto de los componentes.

Singleton: Es un tipo de patrón creacional que se suele utilizar cuando una clase controla el acceso a un recurso físico único o cuando hay datos que deben estar disponibles para todos los objetos de la aplicación.

Ejemplo:

```
public class PoliticaBasico {
    static private PoliticaBasico politica = new PoliticaBasico();
      private PoliticaBasico(){}
        public static PoliticaBasico getInstance(){
            if ( politica == null )
                politica = new PoliticaBasico();
             return politica;
        }
}
```

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

2.10 Conclusiones Parciales del Capítulo

El desarrollo de este capítulo ha permitido comprender la estructura, diseño y dinámica del sistema que se pretende implantar. Hemos comprendido los problemas que posee el Centro en materia de seguridad en servicios web, así como las restricciones que tendrá que cumplir el sistema a desarrollar una vez confeccionado el levantamiento de requisitos a partir de las necesidades del mismo.

Como resultado, se han obtenido los requisitos funcionales y no funcionales del sistema a través de las historias de usuarios, describiéndose y desarrollándose a través de las iteraciones, las funcionalidades de la aplicación. Se introdujeron además, para apoyar el proceso de implementación, las tarjetas CRC

para la representación de las clases, así como los patrones de diseño para aplicaciones web que se utilizarán. A partir de los resultados obtenidos en el análisis y diseño del sistema se puede pasar entonces a la implementación y despliegue, etapa final en el proceso de construcción de la solución de software propuesta, el cual se presenta en el próximo capítulo.

Capítulo 3: Implementación y Despliegue del Sistema.

3.1 Introducción

En este capítulo, siguiendo las fases finales de desarrollo del software propuestas por XP, se realiza una descripción detallada de la arquitectura sobre la cual está implementada la solución, la cual permite la independencia de los sistemas que componen la herramienta, así como la modificación del sistema en períodos de tiempo reducido, ahorrar espacios y documentar más fácilmente el código. Así como también se da una argumentación sobre las pruebas y técnicas de validación utilizadas para el sistema.

3.2 Despliegue del sistema

Definir la arquitectura de un sistema es, sin duda, el elemento más importante durante el proceso de construcción de un software, y de ella depende en gran medida el éxito del desarrollo. *La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución*.[29]
Este es un proceso durante el cual la toma de decisiones de alto nivel que comienzan incluso antes de la selección de un lenguaje o plataforma de desarrollo.

Luego del proceso de descripción de las reglas del negocio y las historias de usuarios se determinan las características arquitectónicas con las que se dotará al sistema propuesto, las cuales se muestran a continuación:

- Se deben guardar los archivos de políticas como archivos físicos dentro del disco duro
- No se utilizará sistema gestor de bases de datos (SGBD).

Otras decisiones relevantes de orden arquitectónico como la selección de la plataforma de desarrollo, lenguaje, frameworks y herramientas de modelado, no se discuten aquí porque fueron abordadas suficientemente en el capítulo anterior. Acontinuación se abordan las decisiones tomadas aquí de forma más detallada.

Se deben guardar los archivos de políticas como archivos físicos dentro del disco duro

Debido a la necesidad de crear, modificar o eliminar políticas de seguridad, estas deben ser guardadas en una ubicación física para su ulterior manejo.

No se utilizará sistema gestor de bases de datos (SGBD)

Las políticas de seguridad constituyen los datos que la aplicación maneja, y como se explica anteriormente, pueden ser almacenados en una ubicación física o el Registro de la herramienta WSO2. Se toma en consideración entonces que utilizar un SGBD para manejar tan pocos datos aumentaría innecesariamente la complejidad tanto del desarrollo como del despliegue de la aplicación implicando una gran pérdida de rendimiento durante el diseño de políticas que solo se guardaran de forma temporal, lo cual no sería tan beneficioso como su utilización.

3.2.1 Arquitectura MVC del Sistema

Para la implementación del sistema, se utiliza el patrón de arquitectura MVC (Modelo Vista Controlador), el cual fue descrito por primera vez en 1979 y separa la lógica de negocio de la interfaz de usuario permitiendo la evolución por separado de ambos aspectos, dando como resultado el incremento notable de la flexibilidad y la reutilización. Además de que ha sido utilizado y documentado en numerosos frameworks para aplicaciones web como JAVA Swing, Java Enterprise Edition (J2EE), Apache Struts, Vaadin, entre otros.

A continuación son definidos cada uno de sus componentes:

- ✓ **Modelo:** se trata de las estructuras de datos. Está compuesto por varias clases, las cuales representan los parámetros a recoger necesarios para la estructuración de una política, ya sea de mensaje o de transporte:
  - clase Política, para tipo Transporte (ya sea modo Básico o Avanzado).
  - clase Política, para tipo Mensaje (ya sea modo Básico o Avanzado).
- ✓ **Vista:** es la interfaz con la que interactúa el usuario. Está compuesto por varias interfaces de usuarios:
  - la interfaz Principal de la Aplicación.
  - las interfaces de construcción de política Modelo Avanzado y Modelo Básico.

  Este componente se encuentra relacionado con la selección de las interfaces de construcción de políticas, así como el diseño de las mismas.

- ✓ **Controlador:** se encarga de procesar los eventos. Está compuesto por varias clases y métodos:
  - las clases encargadas de la construcción de las políticas, a partir de los datos recogidos en el modelo,
  - y los métodos de validación.

  Este componente se encuentra relacionado con todo el desarrollo y confección de las políticas.

Figura 3.1 Representación del flujo de eventos dentro del patrón de arquitectura MVC

Aplicación del patrón:

El modelo: las clases destinadas a recoger los parámetros necesarios en la confección de las políticas. Ejemplo Clase PoliticaTransporteBasico para crear políticas de tipo Transporte utilizando el Modelo Básico.

```
public class PoliticaTransporteBasico {

    private String id_politica;
    private String tipo_algoritmo;

    static private PoliticaBasico politica = new PoliticaBasico();
    private PoliticaBasico(){}
      public static PoliticaBasico getInstance(){
          if ( politica == null )
              politica = new PoliticaBasico();
          return politica;
      }
    public String getId_politica() {
      return id_politica;
    }
    public void setId_politica(String id_politica) {
      this.id_politica = id_politica;
    }

    public String getTipo_algoritmo() {
      return tipo_algoritmo;
    }

    public void setTipo_algoritmo(String tipo_algoritmo) {
      this.tipo_algoritmo = tipo_algoritmo;
    }
}
```

Figura 3.2 Vista de la clase PolíticaTransporteBasico

La vista: las clases destinadas a representar las interfaces de usuario.

Ejemplo la clase AcordioTransporteBasico para mostrar la vista del Modelo Básico.

```
public class AcordionTransporteBasico extends CustomComponent implements SelectedTabChangeListener, ClickListener {
private Accordion a;
private Label label;
private Button terminar;
private VerticalLayout content;

//Paso1
private TextField idPolitica;
private ComboBox tipo_algoritmo;

public Component getContent() {
if(content == null)
{
    content = new VerticalLayout();
    content.setSpacing(true);

    a = new Accordion();
    // Crea la subventana
    subwindow = new Window("Elija donde guardar la política creada");
    subwindow.setWidth("400px");
    subwindow.setHeight("250px");
    subwindow.setClosable(false);

    label = new Label("Modo Básico");
    label.addStyleName("posicion");
    textIdPol = new Label("Introduzca el Identificador (ID) de la política a diseñar:");
    content.addComponent(textIdPol);
    idPolitica = new TextField();
```

Figura 3.3 Vista de la clase AcordionTransporteBasico (parte 1)

```
    idPolitica.setInputPrompt("Escriba un id para su política");
    idPolitica.setDescription("Crea el nombre de su política");
    idPolitica.setComponentError(null);
    content.addComponent(idPolitica);
    for (int i = 0; i < 2; i++)
    {
/*-----------------------------------PASO 1-------------------------------------------*/
      if( i == 0) //PASO 1
       {
          VerticalLayout layout = new VerticalLayout();
          a.addComponent(layout);

          layout.setCaption("Paso 1 : Definiciones Básicas");
          Label Tipo_alg = new Label("Seleccione el algoritmo con el que se realizarán las operaciones criptográficas (cifrado) para establecer la
          seguridad de los mensajes:");
          Tipo_alg.addStyleName("flowbox");
          layout.addComponent(Tipo_alg);
          tipo_algoritmo = new ComboBox();
          //tipo_algoritmo.setInputPrompt("Seleccione..");
          tipo_algoritmo.setComponentError(null);
          for (int i2 = 0; i2 < type_algor.length; i2++)
           {tipo_algoritmo.addItem(type_algor[i2]);}
          tipo_algoritmo.setValue("Basic256");
          layout.addComponent(tipo_algoritmo);

          layout.setHeight("150px");
       }
}
return content
```

Figura 3.4 Vista de la clase AcordionTransporteBasico (parte 2)

Vista de Interfaz de Inicio de la Aplicación:

Figura 3.5 Vista de la Interfaz Principal de la Aplicación

Vista de Interfaz de Selección de modelos de la Aplicación:

Figura 3.6 Vista de la Interfaz de selección de modelos de la Aplicación

Vista del Modelo Básico para tipo Transporte:

Figura 3.7 Vista de la Interfaz Modelo Básico para tipo Transporte

Vista del Modelo Avanzado para tipo Transporte:

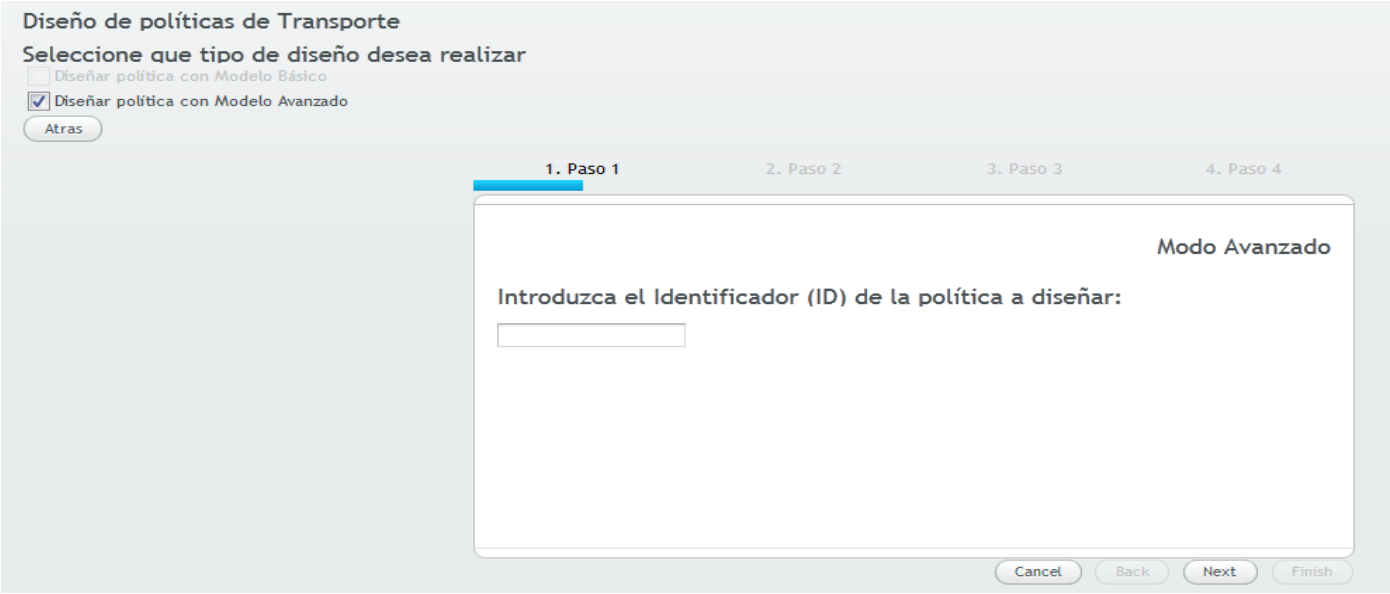


Figura 3.8 Vista de la Interfaz Modelo Avanzado para tipo Transporte

Vista del Modelo Básico para tipo Mensaje:

Figura 3.9 Vista de la Interfaz Modelo Básico para tipo Mensaje

Vista del Modelo Avanzado para Tipo Mensaje:

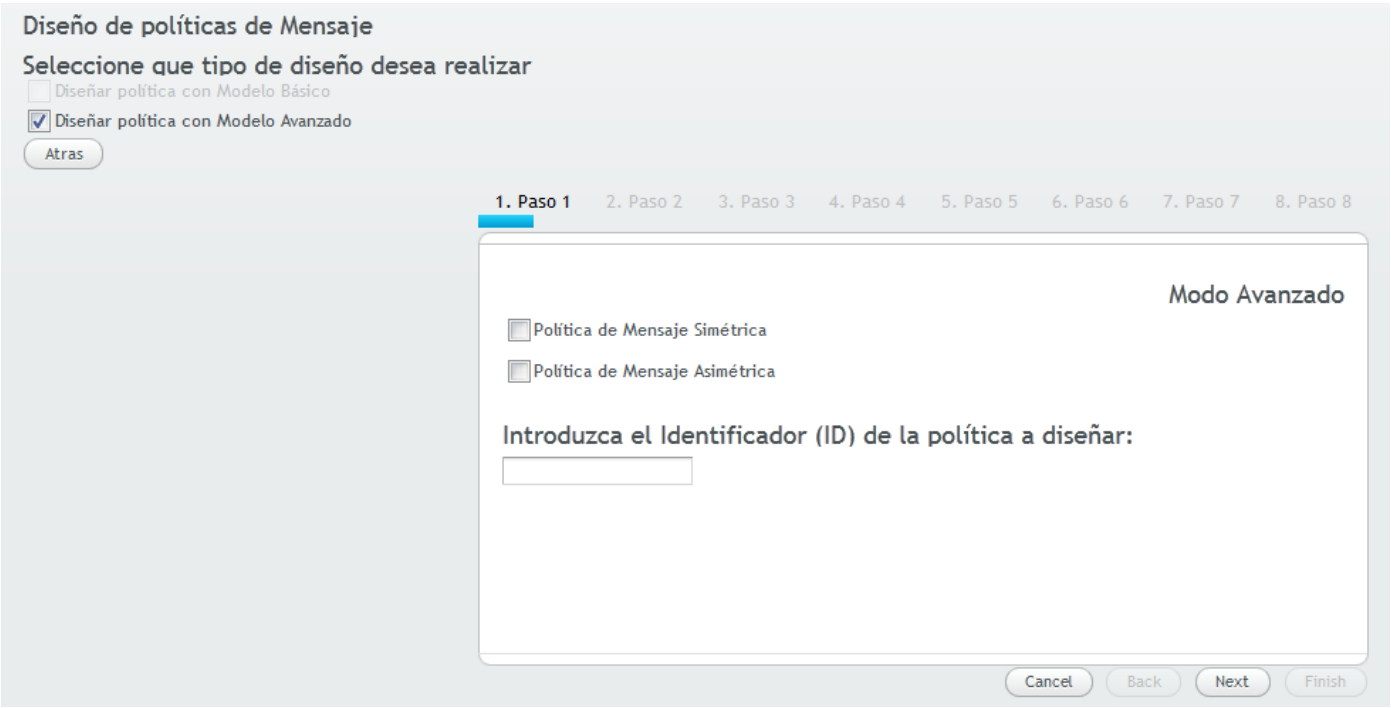


Figura 3.10 Vista de la Interfaz Modelo Avanzado para tipo Mensaje

El controlador: las clases y métodos destinados a validar y modelar los datos recogidos con el objetivo de crear el archivo de política.

Ejemplo la clase TransporteBasico que contiene la llamada a la vista AcordionTransporteBasico, y crea el archivo .xml de la política para tipo Transporte.

```
public class TransporteBasico extends Application {
    private VerticalLayout mainLayout;
    private Acordion acordion;

    public Component getContent(){
     if(mainLayout == null)
      {
        mainLayout = new VerticalLayout();
        mainLayout.setSizeFull();
        mainLayout.setMargin(true);
        Window mainWindow = new Window("Herramienta para el Diseño Gráfico de Políticas de Seguridad");
        mainWindow.addComponent(mainLayout);
        setMainWindow(mainWindow);

        acordion = new Acordion();

        mainLayout.addComponent(acordion.getContent());
        mainLayout.setComponentAlignment(acordion.getContent(), Alignment.MIDDLE_CENTER);
        setTheme("Prueba");
      }
     return mainLayout;
    }
}
```

Figura 3.11 Vista de la clase TransporteBasico

Ejemplo de ejecución de la acción buttonClick del botón Guardar contenido en el método que crea el archivo .xml de la política de tipo Transporte con Modelo Básico:

```
Button close = new Button("Guardar", new Button.ClickListener() {
        // inline click-listener
        public void buttonClick(ClickEvent event) {

            Document doc = DocumentHelper.createDocument();
            Element Policy = doc.addElement("wsp:Policy");

            Policy.addAttribute("wsu:Id", PoliticaBasico.getInstance().getId_politica());
            Policy.addNamespace("wsu", "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd");
            Policy.addNamespace("wsp", "http://schemas.xmlsoap.org/ws/2004/09/policy");
            Element ExactlyOne = Policy.addElement("wsp:ExactlyOne");
            Element All = ExactlyOne.addElement("wsp:All");

            Element TransportBinding = All.addElement("sp:TransportBinding");
            TransportBinding.addNamespace("sp", "http://schemas.xmlsoap.org/ws/2005/07/securitypolicy");

            XMLWriter salida;
            try {
                salida = new XMLWriter(new FileWriter( new File(DirPol.getValue().toString())));
                salida.write(doc);
                salida.close();
               } catch (IOException e) {
                    e.printStackTrace();
               }
}});
```

Figura 3.12 Vista del evento buttonClick

3.3 Producción

Esta es la fase donde se realizan las pruebas de rendimiento al sistema propuesto antes de presentarse al cliente, además de tomarse decisiones sobre si incluir o no nuevas funcionalidades a la versión a presentar por cambios ocurridos durante la fase anterior. Todas las ideas y sugerencias han de ser documentadas para su posterior implementación.

3.3.1 Pruebas del Sistema

El proceso de pruebas es un elemento importante y uno de los pilares de la metodología XP, el cual estimula a los desarrolladores a probar constantemente tanto como sea posible. Mediante esta filosofía se reduce el número de errores no detectados así como el tiempo entre la introducción de este en el sistema y su detección.

Dicha metodología divide las pruebas en varios grupos:

- **Pruebas Unitarias:** Son las pruebas implementadas por los desarrolladores, encargadas de verificar el código.
- **Pruebas de Integración:**Son las pruebas destinadas a verificar el sistema una vez añadida una nueva funcionalidad.
- **Pruebas de Aceptación(o pruebas funcionales):**Son las pruebas destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente.

XP es una metodología de desarrollo que hace énfasis tanto en las pruebas unitarias como en las de integración y aceptación. Un programador que desarrolle su aplicación bajo las pautas de esta metodología debe siempre probar su código de forma unitaria y una vez probado el mismo debe integrarlo a la aplicación y hacer las pruebas de integración correspondientes. Al sistema propuesto se le aplicaron las pruebas de aceptación como XP propone.

3.3.1.1 Pruebas de aceptación

Las pruebas de aceptación son pruebas de caja negra ejecutadas por el cliente o el equipo de desarrollo, con el objetivo de comprobar que la solución implementada cumple con las funcionalidades descritas y descartar los posibles errores. Durante la fase de desarrollo de las iteraciones las HU que fueron seleccionadas serán transformadas en pruebas de aceptación, especificándose los escenarios que aprobados por el cliente se utilizaran para probar las funcionalidades desarrolladas.

Estas pruebas, en especial, son de mucha más importancia que las relatadas anteriormente ya que miden el nivel de satisfacción del cliente con cada iteración concluida, además de que marcan el final de esta y el comienzo de la próxima. Por lo que a continuación se muestran una serie de casos de prueba, los cuales servirán como muestra visual del proceso de pruebas realizadas a la aplicación propuesta.

Dichos casos de pruebas se describirán en tablas que contendrán los siguientes campos:

- **Código:** Identificador de la prueba realizada, a su vez será sugerente al nombre de la prueba a la que hace referencia.
- **Número de UH:** Nombre de la historia de usuario a la que hace referencia la prueba a realizar.
- **Condiciones de Ejecución:** Muestra las condiciones que deben cumplirse para poder llevar a cabo el caso de prueba, estas condiciones deben ser satisfechas antes de la ejecución del caso de prueba para que se puedan obtener los resultados esperados.

- **Entradas / Pasos de Ejecución:** Descripción de cada uno de los pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las entradas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
- **Resultados de la prueba:** Descripción breve del resultado que se espera obtener con la prueba realizada.

La evaluación de la pruebarealizada se hará según el resultado de la misma, la tendrá uno de los tres resultados que a continuación se describen:

Bien: Cuando el resultado de la prueba es exactamente el esperado por el usuario.

Parcialmente bien: Cuando el resultado no es completamente el esperado por el cliente o usuario de la aplicación y muestra resultados erróneos o fuera de contexto.

Mal: Cuando el resultado de la prueba realizada genera un error de codificación en la aplicación o muestra como resultado elementos no deseados o fuera de contexto, trayendo como consecuencia que la funcionalidad requerida por el cliente no tenga resultado, lo que invalida también la UH.

Tabla 3.1 Caso de Prueba de Aceptación #1

Caso de prueba de aceptación	
Número de HU: 3	**Código:***HU3-CP1*
Condiciones de ejecución: El usuario debe haber seleccionado el tipo de diseño que desea realizar.	
Entrada/Pasos de ejecución: El usuario selecciona la opción que desea, el sistema verifica la opción seleccionada, si es Básico entonces muestra la interfaz de escoger política.	
Resultados esperados de la prueba: Si la opción seleccionada por el usuario es Básico entonces el sistema debe mostrar la interfaz de realizar Política con Modelo Básico o Modelo Avanzado.	
Resultados reales de la prueba: El sistema muestra la interfaz de realizar Política con Modelo Básico o Avanzado.	

Tabla 3.2 Caso de Prueba de Aceptación #2

Caso de prueba de aceptación	
Número de HU: 3	**Código:***HU3-CP2*
Condiciones de ejecución: El usuario debe haber seleccionado el tipo de diseño que desea realizar.	
Entrada/Pasos de ejecución: El usuario selecciona la opción que desea, el sistema verifica la opción seleccionada, si es Básico entonces muestra la interfaz de Modelo Básico.	

Resultados esperados de la prueba: : Si la opción seleccionada por el usuario es Básico entonces el sistema debe mostrar la interfaz de Modelo Básico.
Resultados reales de la prueba: El sistema muestra la opción de Modelo Básico.

Tabla 3.3 Caso de Prueba de Aceptación #3

Caso de prueba de aceptación	
Número de HU: 3	**Código:***HU3-CP3*
Condiciones de ejecución: El usuario debe haber seleccionado el tipo de diseño que desea realizar.	
Entrada/Pasos de ejecución: El usuario selecciona la opción que desea, el sistema verifica la opción seleccionada, si es Avanzado entonces muestra la interfaz de Modelo Avanzado.	
Resultados esperados de la prueba: Si la opción seleccionada por el usuario es Avanzado entonces el sistema debe mostrar la interfaz de Modelo Avanzado.	
Resultados reales de la prueba: El sistema muestra la interfaz de Modelo Avanzado	

Tabla 3.4 Caso de Prueba de Aceptación #4

Caso de prueba de aceptación	
Número de HU: 4	**Código:***HU4-CP1*
Condiciones de ejecución: El usuario debe haber seleccionado el tipo de política a realizar Básico y alguna opción de política mostrada.	
Entrada/Pasos de ejecución: El usuario selecciona las opciones que desee y la opción de Terminar, con lo que el sistema debe verificar cada una de estas opciones y confeccionar la política.	
Resultados esperados de la prueba: Si el usuario selecciona la opción de Terminar, el sistema debe confeccionar una política y mostrar un mensaje satisfactorio.	
Resultados reales de la prueba: El sistema muestra un mensaje satisfactorio y la opción de Terminar. Confecciona la política.	

Tabla 3.5 Caso de Prueba de Aceptación #5

Caso de prueba de aceptación	
Número de HU: 4	**Código:***HU4-CP2*
Condiciones de ejecución: El usuario debe haber seleccionado el tipo de política a realizar Básico y alguna opción de política mostrada, y luego seleccionado las opciones de atributos de política con la opción de Terminar.	

Entrada/Pasos de ejecución: El usuario selecciona la opción de Terminar, el sistema confecciona la política y muestra la opción de donde guardar dicha política.
Resultados esperadas de la prueba: Si el usuario selecciona la opción de Terminar, el sistema muestra la opción de donde guardar la política.
Resultados reales de la prueba: El sistema muestra la opción de escoger donde guardar la política.

Tabla 3.6 Caso de Prueba de Aceptación #6

Caso de prueba de aceptación	
Número de HU: 5	**Código:***HU5-CP1*
Condiciones de ejecución: El usuario debe haber seleccionado el tipo de política a realizar Avanzado.	
Entrada/Pasos de ejecución: El usuario selecciona las opciones de atributos de política que desee y la opción de Finish, con lo que el sistema debe verificar cada una de estas opciones y mostrar la opción de Generar Política.	
Resultados esperados de la prueba: Si el usuario selecciona la opción de Finish, el sistema debe mostrar la opción de Generar Política.	
Resultados reales de la prueba: El sistema muestra la opción de Generar Política.	

Tabla 3.7 Caso de Prueba de Aceptación #7

Caso de prueba de aceptación	
Número de HU: 5	**Código:***HU5-CP2*
Condiciones de ejecución: El usuario debe haber seleccionado el tipo de política a realizar Avanzado, y luego de seleccionar las opciones de atributos de política deseados selecciona la opción Finish y la opción de Generar Política.	
Entrada/Pasos de ejecución: El usuario selecciona la opción de Finish, con lo que el sistema debe mostrar la opción de Generar Política, el usuario selecciona la opción de Generar Política, con lo que el sistema debe verificar cada opción de atributo de política anteriormente seleccionada y confeccionar una política.	
Resultados esperados de la prueba: Si el usuario selecciona la opción de Generar Política, el sistema debe confeccionar una política y mostrar un mensaje satisfactorio.	
Resultados reales de la prueba: El sistema muestra un mensaje de política generada, además la opción de Generar Política. Se genera la política al ejecutar dicha opción.	

Tabla 3.8 Caso de Prueba de Aceptación #8

Caso de prueba de aceptación	
Número de HU: 5	**Código:***HU5-CP3*
Condiciones de ejecución: El usuario debe haber seleccionado el tipo de política a realizar Avanzado y seleccionado la opción de Cancel.	
Entrada/Pasos de ejecución: El usuario selecciona la opción de Cancel, con lo que el sistema debe mostrar la opción de Cancelar Política, si el usuario selecciona dicha opción el usuario deberá cancelar la política en curso y mostrar un mensaje satisfactorio.	
Resultados esperados de la prueba: Si el usuario selecciona la opción de Cancel, el sistema debe mostrar la opción de Cancelar Política, y si el usuario la selecciona entonces el sistema deberá cancelar la política en curso y mostrar un mensaje satisfactorio.	
Resultados reales de la prueba: El sistema muestra un mensaje de política cancelada, además la opción de Cancelar Política. Se cancela la política al ejecutar dicha opción.	

3.3.1.2Pruebas unitarias

Las pruebas unitarias son un tipo de pruebas de caja blanca que revisa a cada clase en aislamiento como elemento unitario. El objetivo fundamental de estas pruebas es asegurar el correcto funcionamiento de las interfaces y el flujo de datos entre componentes.

Con este fin se implementaron métodos de validación para cada modelo de construcción de política (Básico y Avanzado), garantizando que los datos recogidos tuvieran la estructura correcta para conformarlas.

- Para el Modelo Básico los códigos de validación se implementan al ejecutar evento buttonClick() del botón Generar Política:

```
public void buttonClick(ClickEvent event) {

    boolean ok = true;
    if(idPolitica.getValue().toString().equals(""))
      {    ok = false;
           idPolitica.setComponentError(new UserError("Debe escribir un Identificador (ID) para la política" +
                " que desea diseñar"));
           content.getApplication().getMainWindow().showNotification("Debe escribir un " +
                "Identificador (ID) para la política que desea diseñar", Notification.TYPE_TRAY_NOTIFICATION);
      }
    else
    {
       idPolitica.setComponentError(null);
       PoliticaBasico.getInstance().setId_politica(idPolitica.getValue().toString());
    }

    //VALIDACION PASO 1
    boolean mioP1 = true;
    if(mec_autenticacion.getValue() == null)
     {
        mec_autenticacion.setComponentError(new UserError("Debe seleccionar si desea o no exigir " +
                "un mecanismo de autenticación entre cliente y servidor"));
        content.getApplication().getMainWindow().showNotification("Debe seleccionar si desea o no" +
                " exigir un mecanismo de autenticación entre cliente y servidor", Notification.TYPE_TRAY_NOTIFICATION);
        ok = false;
     }
```

Figura 3.13 Vista del evento buttonClick

- Para el Modelo Avanzado los códigos de validación se implementan en cada paso de construcción al ejecutar el método onAdvance() del botón Advance:

```
public boolean onAdvance() {
      boolean avance = true;
      boolean mio = true;
      if(mec_autenticacion.getValue() == null)
       {
          avance = false;
          mec_autenticacion.setComponentError(new UserError("Debe seleccionar si desea o no exigir un mecanismo de autenticación entre cliente y servidor"));
          content.getApplication().getMainWindow().showNotification((String) notif.getValue(), Notification.TYPE_TRAY_NOTIFICATION);
          mio = false;
       }
      else if(mec_autenticacion.getValue() != null)
       {
          mec_autenticacion.setComponentError(null);
          Politica.getInstance().setUso_TransportToken("RequireClientCertificate="+ "\""+ mec_autenticacion.getValue().toString()+"\"");
       }
      if(tipo_algoritmo.getValue() == null && mio )
       {
          avance = false;
          tipo_algoritmo.setComponentError(new UserError("Debe seleccionar algun tipo de algoritmo"));
          content.getApplication().getMainWindow().showNotification((String) notif1.getValue(), Notification.TYPE_TRAY_NOTIFICATION);
       }
      else
       {
          tipo_algoritmo.setComponentError(null);
          String tipoAlgoritmo =  (String) tipo_algoritmo.getValue();
          Politica.getInstance().setTipo_algoritmo(tipoAlgoritmo);
       }
       return avance;
   }
```

Figura 3.14 Vista del evento onAdvance

3.3.1.3 Validación de políticas

La creación de políticas de seguridad válidas debe pasar necesariamente por un proceso de validación de las mismas, que en este sentido no es más que la comprobación de los archivos .xml con los que se formulan dichas políticas. Para ello se impone la necesidad de verificar que estos tengan una sintaxis correcta, o sea, validar la corrección, integridad e interpretación de los datos, dependiendo para ellos de factores como la calidad de dichos datos, quién los manipula, los crea o de dónde se originan.

Se disponen de varios métodos de validación, de los cuales los más usados son:

- **DTD[32](de XML versión 1.0):** Es el más antiguo de todos los métodos de validación mostrados y utiliza una sintaxis no-XML para definir la estructura del contenido de los archivos .xml. En él se definen todos los elementos y sus relaciones, siendo el más sencillo de los métodos usados para validar. A pesar de esto presenta desventajas como no soportar las nuevas versiones de XML.
- **Schematron(de Academia Sinica Computing Centre):** Este método utiliza expresiones de acceso en lugar de lasgramaticales, lo que aporta una gran flexibilidad en la descripción de estructuras relacionales pero lo limita a la hora de especificar la estructura básica del documento.
- **XSD[33](de W3C[34]):**Es una evolución del DTD explicado anteriormente, pero mucho más complejo y potente que utiliza una sintaxis XML que le permite especificar de forma más detallada un extenso sistema de tipos de datos, soportando la extensión de los documentos sin problemas a diferencia de los DTD's.En él se definen los nombres de los elementos y sus atributos, los tipos de datos, así como sus relaciones y estructuras. Aunque su utilización supone una pérdida de recursos y tiempo dada la complejidad de su sintaxis, después de validar el archivo .XML es posible transformar el documento en una jerarquía de objetos accesibles a través de un lenguaje orientado a objetos dada la posibilidad de expresar su estructura y contenido en términos del modelo de datos usado por el esquema de validación.

Este último sería entonces el más idóneo para nuestra aplicación dada su capacidad descriptiva e integración con los lenguajes de programación orientado a objetos como JAVA, por lo que se decidió su utilización en la misma.

[32] *Acrónimo de DocumentTypeDefinition, el cual es una descripción de estructura y sintaxis de un documento XML o SGML.*
[33] *Acrónimo de XML SchemaDefinition.*
[34] *Es un consorcio internacional que produce recomendaciones para la World Wide Web.*

3.4 Conclusiones Parciales del Capítulo

En el presente capitulo se han desarrollado las últimas fases en la implementación de la aplicación propuesta, siguiendo las estrategias propuestas por la metodología XP. En él se ha detallado y elaborado el diseño estructural del sistema así como las pruebas de aceptación y unitarias necesarias para terminar con el ciclo de producción de la aplicación y obtener la salida del producto, siguiendo las orientaciones del cliente.

Conclusiones Generales

Como resultado de este trabajo se realizó un estudio de los sistemas informáticos para el diseño de políticas de seguridad existentes, así como las ventajas y desventajas de cada uno de ellos. Se realizóademás un estudio de las herramientas y tecnologías adecuadas para el desarrollo de la aplicación propuesta disponibles para el desarrollo, según diversos criterios.

Proponiéndose entonces un diseño estructural de una aplicación, siguiendo los requerimientos expresados, que permita construir una herramienta capaz de confeccionar políticas de seguridad para servicios web y que ayude a limar las limitantes presentes en este sentido.

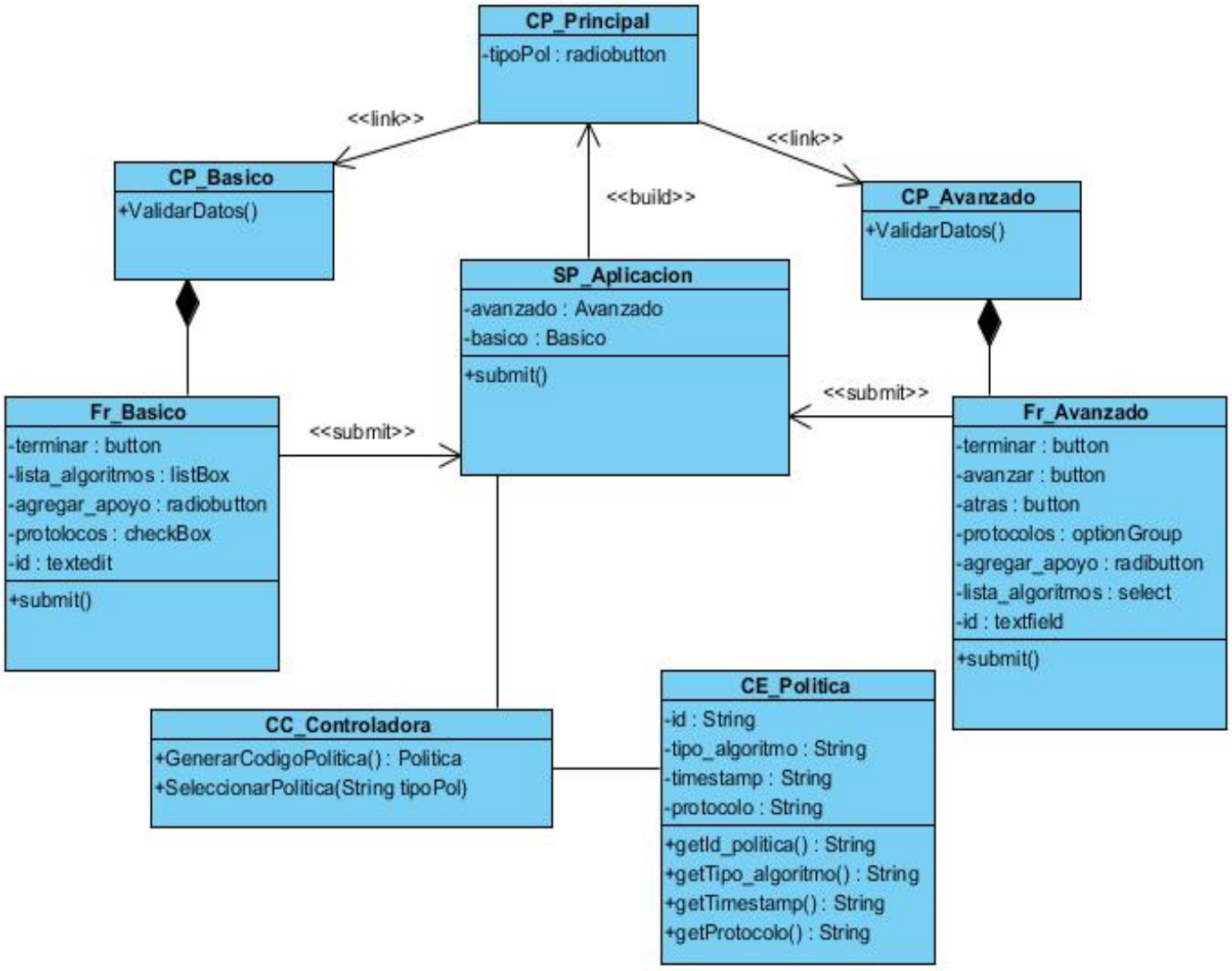
Esta herramienta puede constituir también un apoyo significativo para centros que gestionen los servicios web y su seguridad, brindando una alternativa con respecto a herramientas privativas o de difícil manejo. También es importante mencionar que la aplicación fue validada de forma satisfactoria, siguiendo técnicas de validación y pruebas de caja negra y blanca.

Recomendaciones

- Guardar las políticas creadas en el Registro de Gobierno de la Suite WSO2.
- Realizar versiones posteriores de la aplicación propuesta que diseñen políticas de mayor complejidad.
- Desplegar la aplicación propuesta en proyectos actuales del CDAE.
- Desarrollar la aplicación propuesta como un plugin de la Suite WSO2.

Anexos

Diagrama de Clases con Estereotipos Web



Diagramas de Secuencias del Diseño

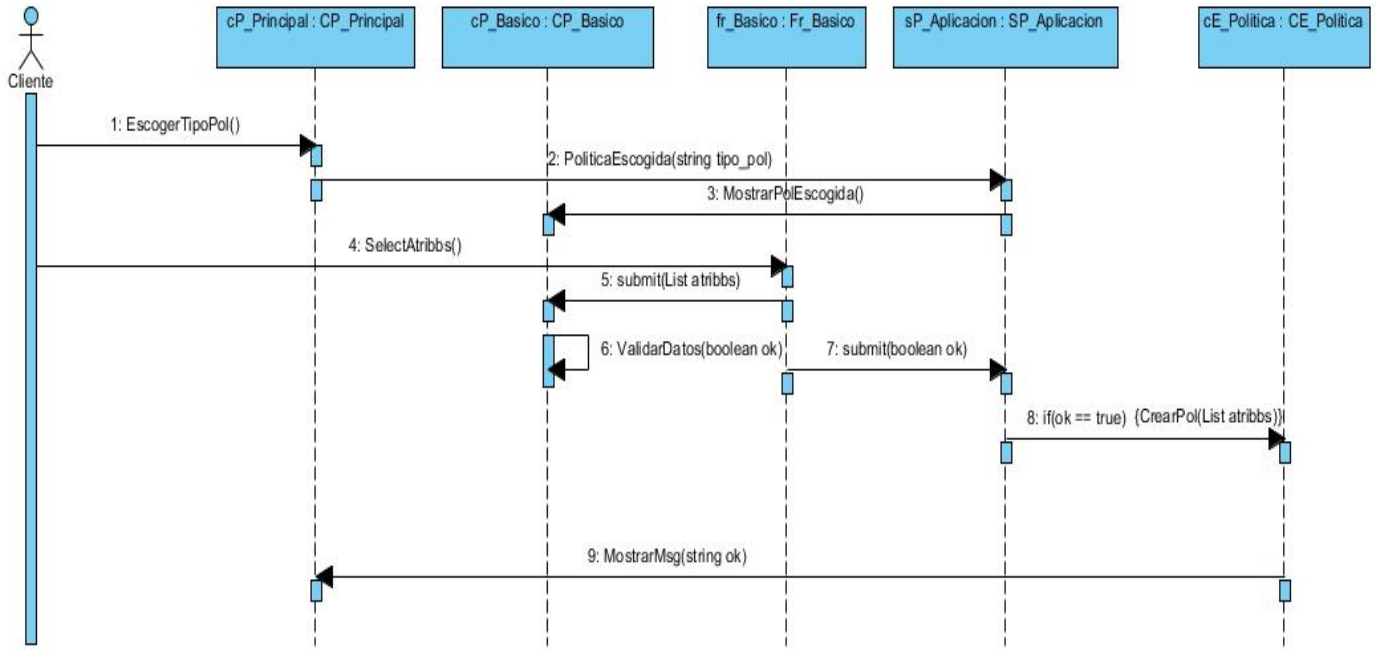


Diagrama de secuencias básico

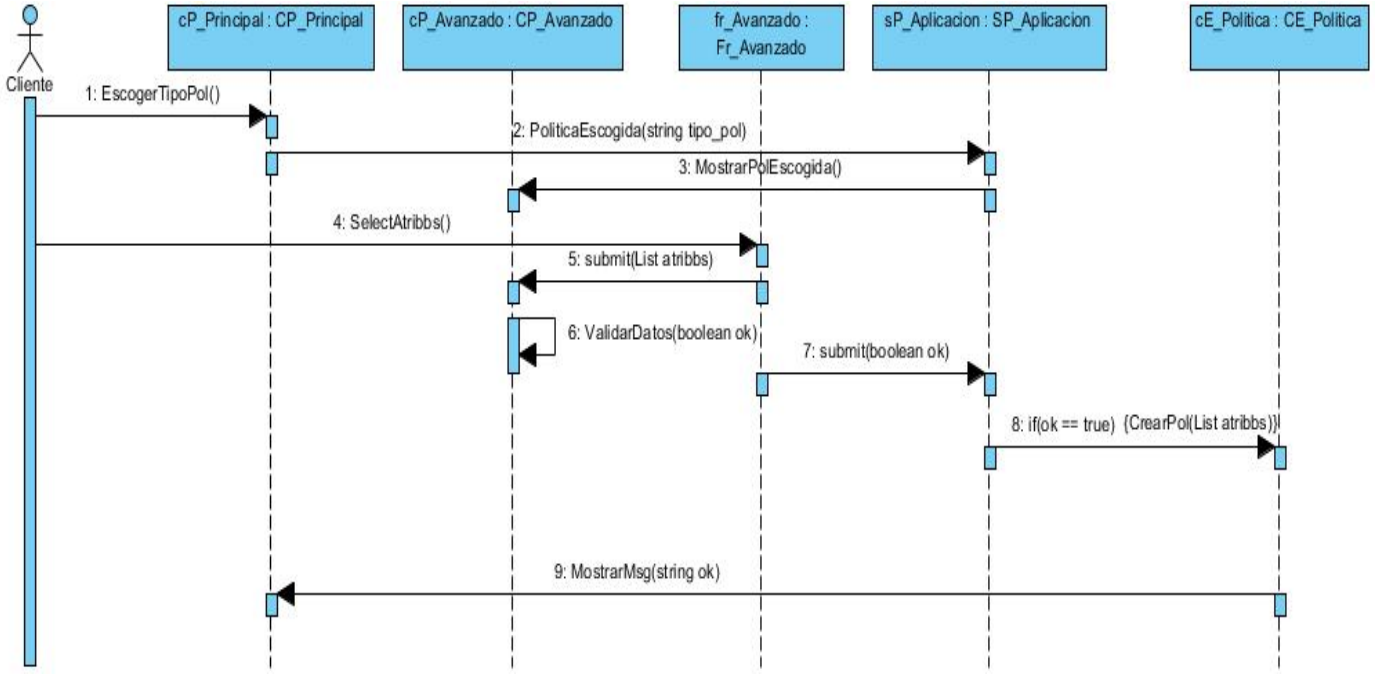


Diagrama de secuencias avanzado

Modelo de despliegue del sistema

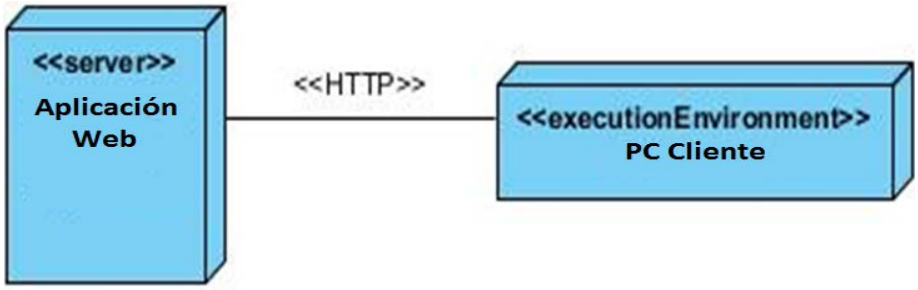
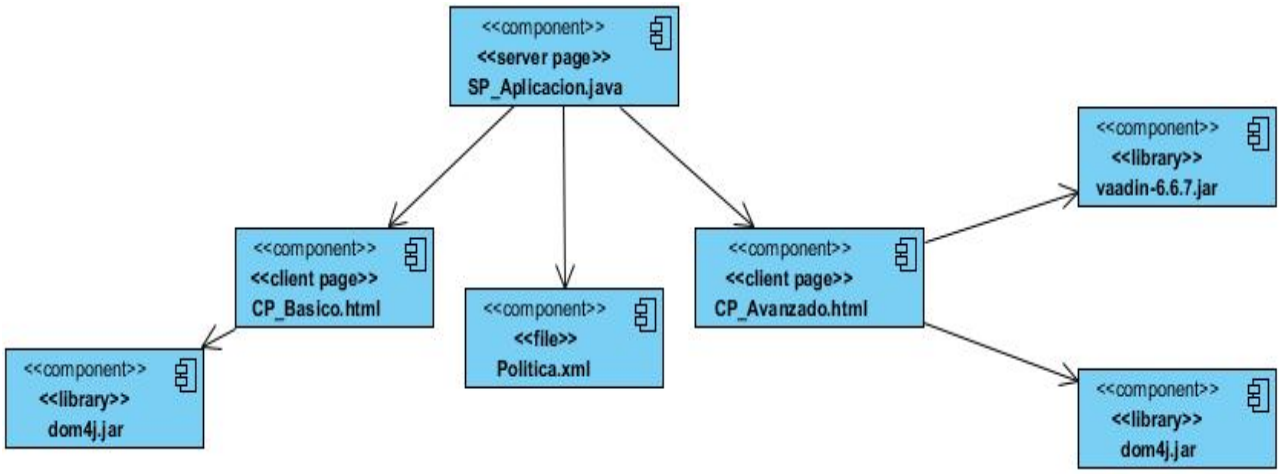


Diagrama de componentes



Glosario de Términos

Pruebas de caja negra: Son una estrategia de pruebas que se enfocan directamente en el exterior delmódulo, sin detenerse en analizar el código fuente. El objetivo es demostrar que las funciones delsoftware son operativas, que las entradas se aceptan de forma adecuada y se produce un resultadocorrecto, y la integridad de la información externa se mantiene. Se centran en los requisitos funcionalesdel software.

Pruebas de caja blanca: Son una estrategia de pruebas que se enfoca en comprobar los caminoslógicos del software. Se puede examinar el estado del programa en varios puntos para determinar si elestado real coincide con el esperado. Requieren de conocimiento de la estructura interna del programa yson derivadas a partir de las especificaciones internas de diseño o el código.

Referencias Bibliográficas

1. [cited; Availablefrom: seguridadPC.net.
2. Aguirre, J.V., Fundamentos para el desarrollo de un sistema de gestión de seguridad de la información. 2005, Universidad del Valle: Santiago de Cali.
3. Española, R.A., Diccionario de la Lengua Española - Vigésima Segunda Edición. 2001.
4. Navarro, S.H., Politicaspublicas y Politicas de Vejez. Aspectos Teoricos-Conceptuales.
5. Brun, R.E. XML y Servicios Web. 2002 [cited.
6. Saffirio, M. (2006) ¿Qué son los Web Services. Volume,
7. Ferrante, H., "GartnerOutlines Top 10 Technologies toImpactProperty and CasualtyInsurance". 2010.
8. Agenda, T.S. Arquitectura Orientada a Servicios. [cited; Availablefrom: http://soaagenda.com/journal/articulos/category/soa/.
9. Eduardo, J.
10. Sanabria, J.L.C., Políticas de Seguridad. 2004: Centro Nacional de Cálculo Cíentifico.
11. C. Gutiérrez, E.F.-M., M. Piattini, Seguridad en Servicios Web.
12. Proyectos, L.d.G.d. CDAE. 2012 [cited; Availablefrom: http://gespro.cdae.prod.uci.cu/.
13. Microsoft. [cited; Availablefrom: http://msdn.microsoft.com/en-us/library/cb6t8dtz%28v=vs.80%29.aspx.
14. Cisco Systems, I. [cited; Availablefrom: http://www.ciscowebtools.com/spb/.
15. FabienAutrel, F.C., Nora Cuppens, Céline Coma, MotOrBAC 2: a securitypolicytool.
16. Research, M. [cited; Availablefrom: http://research.microsoft.com/en-us/projects/samoa/.
17. KarthikeyanBhargavan, C.e.F. and R.P. Andrew D. Gordon, TulaFale: A Security Toolfor Web Services.
18. Security, O.F.M. [cited; Availablefrom: http://docs.oracle.com.
19. WSO2. WSO2 Carbon. 2012 [cited; Availablefrom: https://www.wso2.com.
20. Solís, M.C., V Encuentro usuarios xBase 2003: Madrid.
21. Escribano, G.F., Introducción a Extreme Programming. 2002.
22. Lenguajes de Programación. 2008 [cited; Availablefrom: http://es.kioskea.net.
23. Nuño, I.C., Guía de iniciación al lenguaje Java. 1999.
24. Périssé, M.C., Una Metodología Simplificada. 2001.

25. EcuRed, P.d. Visual Paradigm. [cited; Availablefrom: http://www.ecured.cu/index.php/Visual_Paradigm.

26. Foundation, T.E. [cited; Availablefrom: www.eclipse.org/.

27. Grönroos, M., Book of Vaadin:Vaadin 6.4. 2010.

28. Group, O.M. UML® Resource Page. 2012 [cited; Availablefrom: http://www.uml.org/.

29. Reynoso, B., ArchitectAcademyWebcast #1: Seminario de Arquitectura de Software. 2005.

Bibliografía

1. C. Gutiérrez, E.F.-M., M. Piattini, Seguridad en Servicios Web.
2. Nuño, I.C., Guía de iniciación al lenguaje Java. 1999.
3. Périssé, M.C., Una Metodología Simplificada. 2001.
4. Jonhson, R., ExpertOne-on-One Java EE Design and Development 2002: WroxPress.
5. Grönroos, M., Book of Vaadin:Vaadin 6.4. 2010.
6. Aguirre, J.V., Fundamentos para el desarrollo de un sistema de gestión de seguridad de la información. 2005, Universidad del Valle: Santiago de Cali.
7. Reynoso, B., ArchitectAcademyWebcast #1: Seminario de Arquitectura de Software. 2005.
8. Patricio Letelier, M.C.P., Métodologías ágiles para el desarrollo de software: eXtremeProgramming(XP), Valencia.
9. Schwaber K., B.M., Martin R.C., Agile Software Developmentwith SCRUM. 2001.
10. Highsmith J., O.K., Adaptive Software Development: A CollaborativeApproachtoManagingComplexSystems. 2000.