Universidad de las Ciencias Informáticas

Facultad 3

Desarrollo del Componente de Seguridad Onyx para el sistema Quarxo Fase 2

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

Autores

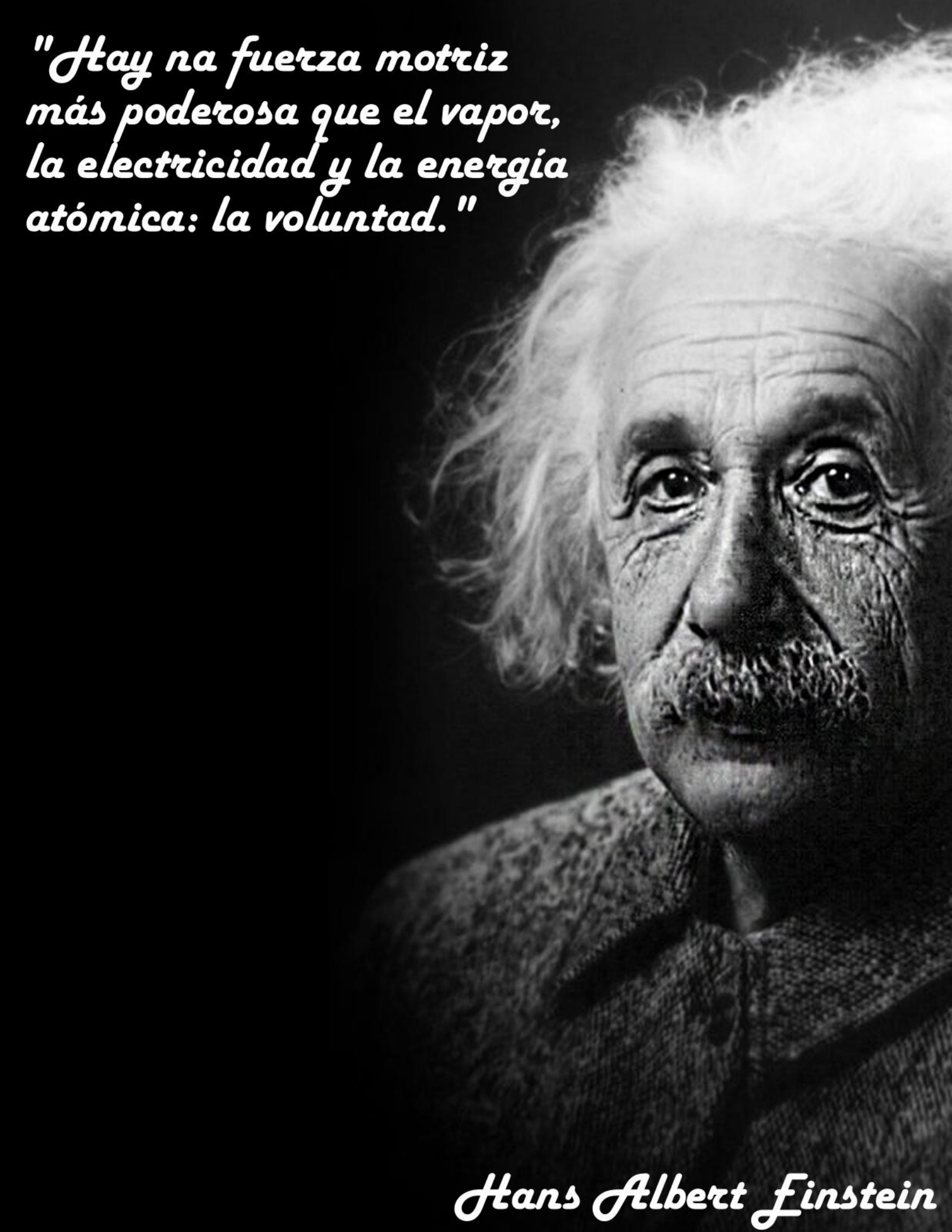
Jorge Enrique Ricardo Maceo

Jorge Fonseca Rodríguez

Tutor

Ing. Rafael Bello Lara

"Año 55 de la Revolución"


"Hay na fuerza motriz
más poderosa que el vapor,
la electricidad y la energía
atómica: la voluntad."
Hans Albert Finstein

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de este trabajo y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2013.

_____

Jorge Fonseca Rodríguez

Autor

_____

Jorge Enrique Ricardo Maceo

Autor

_____

Ing. Rafael Bello Lara

Tutor

Dedicatoria

Jorge Fonseca Rodríguez

A mis padres, por hacer de mí lo que soy hoy.

A mi abuelo, por ser mi guía desde el otro mundo.

Jorge Enrique Ricardo Maceo

A mis padres María y Jorge que me han dado su amor incondicional y me han apoyado en todas mi decisiones.

A mi abuela Ana y mi tía Ana Julia, mis hermanos Lili, Alejandro y Gabriel.

Agradecimientos

Jorge Fonseca Rodríguez

A mis padres, por no haber dudado de mí ni un instante y haberme dado todo el apoyo y afecto necesario para llegar hasta aquí.

A mis abuelos, por ser mucho más que eso y convertirse en parte de mi ser.

A mis hermanos, por el amor que compartimos.

A mi tía Xiomara, por la preocupación constante hacia mí.

Al resto de mi familia, que de una forma u otra me han apoyado y confiado en mí.

A todos mis amigos por la amistad y apoyo durante todo es te tiempo, Radamés, Reiniel, Raúl, Yanier, Wilfredo, Adrián, Luis y Karel entre muchos otros.

A Rafael Bello, por su amistad y sabia tutoría.

A Hector, Dariel, Manuel, Asdrubal, Leonardo, Efraín, Jorge L. Martín y Alejandro, compañeros del proyecto que brindaron su ayuda para la realización de este trabajo.

A mi compañero de tesis.

A todos, sinceramente, gracias….

Jorge Enrique Ricardo Maceo

A mi mamá, que es el amor más lindo que he tenido en mi vida

A mi padre que fue el que me impulso a estudiar esta carrera.

A mi hermana Lili, que ha sido madre, amiga y papá.

A mi hermano Alejandro, por haberme dado su ayuda, sin poner peros.

A mi cuñado Gabriel, que se ha convertido en hermano, gracias por cuidar a mi familia.

A mis tíos Rodríguez, Gisella, Quintero y mi primo Yandi, que fueron mis padres aquí en la habana.

A mi compañero de tesis Jorge

A mis hermanos de la UCI, los más viejos Yorlen, Roknier, Leo, Pochy, en especial a mi hermanito Felix, que esta tesis es de los dos, este título es tuyo como mío hermano.

Los no tan nuevos mis hermanos Silvio, Luis Carlos, Radamés, Uchiha, Kangri, Yenquiel, Karel, Raúl, Yanier, Reiniel, Ramón, Julio, Miguel, Diógenes, al nuevo Enrique. A mis compañeras de Aula Diana, Liset, Yadelis, Aylen, la Kiki, Yeni.

Resumen

En la actualidad, el conocimiento sobre la seguridad en aplicaciones web es sumamente necesario, pues de ello depende la integridad, confidencialidad y disponibilidad de la información almacenada en distintos equipos en la red. Continuamente se crean y perfeccionan herramientas para proveer seguridad a disímiles sistemas, siendo los componentes de seguridad uno de los más utilizados debido a su reusabilidad. En los sistemas bancarios de Cuba la gestión de la seguridad constituye un elemento clave no solo para la protección de los datos, sino para la ejecución y estabilidad del sistema; siendo de vital importancia que dicha gestión posea altos niveles de rendimiento, sea independiente de configuraciones ajenas a su negocio y permita una adecuada auditoria. El objetivo de este trabajo de diploma es desarrollar un componente de seguridad para aplicaciones web, capaz de lograr una adecuada gestión de la seguridad, para un mayor entendimiento se presentan las principales características consideradas en el desarrollo del componente, obtenidas en el estudio de sistemas de gestión de seguridad nacionales e internacionales. Además, se describen las principales tecnologías utilizadas y la metodología de desarrollo del software, así como los elementos fundamentales de la arquitectura, análisis, diseño, implementación y los resultados del proceso de pruebas.

Palabras clave: seguridad, gestión, componente, sistemas bancarios.

ÍNDICE DE CONTENIDO

Introducción

El Banco Nacional de Cuba (BNC) es el encargado de perfeccionar el sistema monetario, normalizar las relaciones financieras externas del país y apoyar las gestiones de crédito de las empresas cubanas y de los bancos que integran el sistema financiero cubano. Actividades que son desempeñadas con un alto nivel de profesionalidad por parte de sus trabajadores.

La Universidad de las Ciencias Informáticas (UCI) desde hace algunos años contribuye en la informatización del Sistema Bancario Cubano. Actualmente existe una estrecha colaboración entre dichas entidades mediante el proyecto Sistema de Gestión Bancaria. Antes de que la UCI brindara sus servicios al BNC, el mismo contaba con un sistema contable sobre MS-DOS y un sistema sobre Windows para la mensajería SWIFT, de manera que los usuarios del BNC debían trabajar sobre plataformas distintas. Estos y otros aspectos fueron erradicados con la primera entrega del sistema Quarxo que realizara la UCI al BNC a mediados del 2012.

En los sistemas bancarios de Cuba, como es el caso del BNC, la gestión de la seguridad constituye un elemento clave, pues de ello depende la confidencialidad, integridad y disponibilidad de la información almacenada en distintos equipos en la red; esta gestión no está asociada únicamente a la protección de los datos sino a la ejecución y estabilidad del sistema, siendo de vital importancia que posea determinados factores de peso como son altos niveles de rendimiento, independencia de configuraciones ajenas a su negocio y que permita una adecuada auditoria.

Las medidas claves de un sistema, como el volumen de transacciones y el uso de recursos, definen el rendimiento, de manera que un gran volumen de operaciones asociado al uso innecesario de gran cantidad de recursos, afecta la capacidad de respuesta del sistema y por lo tanto el tiempo total para llevar a cabo la operación en cuestión; en un sistema bancario esto se traduciría en una reducción de la cantidad de operaciones que se pueden realizar en el día, disminuyendo la entrada de efectivo.

La dependencia a configuraciones y operaciones ajenas al proceso de negocio que se está ejecutando no solo afecta o entorpece al mismo, sino que dificulta en mayor o menor medida el entendimiento y mantenimiento del sistema, dejándolo obsoleto ante futuros cambios.

El control de contenido y supervisión de los datos mediante una adecuada auditoría tributa a sistemas más compactos y entendibles, proporcionando rentabilidad a la organización, eficiencia y seguridad en el

procesamiento de la información además de una gestión más segura de la información que facilita la toma de decisiones.

Actualmente la gestión de la seguridad está en constante perfeccionamiento, creando herramientas que proveen seguridad a disímiles sistemas, siendo los componentes de seguridad uno de los más utilizados debido a su reusabilidad.

Dada la problemática planteada anteriormente se identificó el siguiente **problema a resolver**: Cómo garantizar la gestión de la seguridad en Quarxo con alto rendimiento, independencia de las configuraciones y garantizando una adecuada auditoría.

Teniendo como **objetivo general**: Desarrollar un componente de seguridad con alto rendimiento, independencia de las configuraciones y adecuada auditoria, que garantice una mejor gestión de la seguridad para las nuevas fases del sistema del Banco Nacional de Cuba.

Determinando como **objeto de estudio**: La gestión de la seguridad en aplicaciones web orientada a sistemas bancarios.

Y estableciendo como **campo de acción**: La gestión de la seguridad en Quarxo.

Por lo que, para darle solución al problema planteado se tiene como **idea a defender**: Con la implementación del nuevo componente de seguridad del sistema Quarxo se asegura mayor rendimiento, independencia en las configuraciones y una adecuada auditoría en la gestión de la seguridad.

Fijando de esta forma, los siguientes **objetivos específicos**:

1. Elaborar el Marco Teórico para la gestión de la seguridad en aplicaciones web orientado a sistemas bancarios.
2. Realizar la modelación de los requisitos de seguridad de la fase 2 de Quarxo.
3. Diseñar el componente de seguridad para la fase 2 de Quarxo bajo la arquitectura propuesta.
4. Implementar el componente de seguridad para la fase 2 de Quarxo bajo la arquitectura propuesta.
5. Validar el correcto funcionamiento de la herramienta y que la solución propuesta de cumplimiento al objetivo general de la investigación mediante la evaluación de los indicadores: rendimiento, navegabilidad, usabilidad, independencia y control de contenido.

El contenido de este trabajo consta de 3 capítulos definidos de la siguiente manera:

Capítulo No. 1

Se presentan los elementos teóricos que sirven de base a la investigación del problema planteado. Se realiza un estudio del estado del arte de las herramientas que se utilizan para la gestión de la seguridad. De igual manera se detallan las tecnologías a considerar, el lenguaje de programación que se utiliza para la solución, las metodologías de desarrollo y las herramientas de desarrollo.

Capítulo No. 2

Se realizan las primeras disciplinas de trabajo propuestas por el modelo de desarrollo de CEIGE para el desarrollo de la solución propuesta: Requisitos, Análisis y Diseño. Se explica la utilización de los distintos patrones empleados.

Capítulo No. 3

Se encauza en la implementación y validación del componente de seguridad: Se explican los aspectos fundamentales de la implementación, modelo de componentes, nomenclatura, y se describe la arquitectura. Además, se realizan al software, las pruebas de caja blanca y pruebas de caja negra.

Capítulo 1: Fundamentación Teórica

1. Introducción

Ninguna investigación o aplicación práctica puede y debe tomarse en serio sin una exploración y posterior documentación de la información necesaria para llevarla a cabo. De ahí que la meta de este capítulo radique en establecer la base teórica que soporta el trabajo y aplicación práctica en sí, indagando para ello en fuentes bibliográficas confiables y no restringiéndose a límites geográficos, abarcando en mayor medida todo lo relacionado con la seguridad en sistemas informáticos.

Todo esto viene amparado de un estudio de las herramientas necesarias para el análisis, diseño e implementación de estos sistemas, así como de las metodologías y lenguajes de programación de las que se hizo uso y determinaron la correcta formación del producto. El conocimiento teórico plasmado en este capítulo será una guía para el resto de la investigación, posibilitando una correcta comprensión de los diversos temas discutidos.

1.1. Conceptos fundamentales

En este apartado se abordaran los principales conceptos con los que se estará trabajando en el resto de la investigación siendo los pilares básicos que la soportan.

1.1.1. Seguridad

El término seguridad posee múltiples usos. A grandes rasgos, puede afirmarse que este concepto hace foco en la característica de seguro, es decir, realza la propiedad de algo donde no se registran peligros, daños ni riesgos. Una cosa segura es algo firme, cierto e indubitable. La seguridad, por lo tanto, puede considerarse como una certeza. (1)

Una de las acepciones del término es el que se utiliza en informática, un concepto moderno pero sumamente importante para conservar los ordenadores y equipos relacionados en buen estado. (1)

1.1.2. Seguridad informática

La seguridad informática es una disciplina que se encarga de proteger la integridad y la privacidad de la información almacenada en un sistema informático. Aunque, no existe ninguna técnica que permita asegurar la inviolabilidad de un sistema. (2)

Un sistema informático puede ser protegido desde un punto de vista ***lógico*** (con el desarrollo de software) o ***físico*** (ejemplo: vinculado al mantenimiento eléctrico). Por otra parte, las amenazas pueden proceder desde programas dañinos que se instalan en la ***computadora*** del usuario (como un ***virus***) o llegar por vía remota (cibernautas que mediante la conexión a ***Internet*** ingresan a distintos sistemas). (2)

Entre las herramientas más usuales de la seguridad informática, se encuentran los ***programas antivirus***, los cortafuegos o firewalls, la encriptación de la información y el uso de contraseñas. (2)

1.1.3. Administración de la seguridad

La necesidad de mantener la integridad de la información y proteger los activos de las tecnologías de la información, requiere de un proceso de administración de la seguridad. Este proceso incluye el establecimiento y mantenimiento de ***roles*** y ***responsabilidades*** de seguridad, políticas, estándares y procedimientos. La administración de seguridad también incluye realizar ***monitoreos*** de seguridad y pruebas periódicas así como realizar acciones correctivas sobre las debilidades o incidentes de seguridad identificados. Una efectiva administración de la seguridad protege todos los activos de las tecnologías de la información para minimizar el impacto causado por vulnerabilidades o incidentes de seguridad. (3)

1.1.4. Control de acceso

El control de acceso de los diferentes usuarios a las aplicaciones constituye una poderosa herramienta para proteger la entrada a un sistema completo o sólo a ciertos directorios concretos e incluso a ficheros o programas individuales; este control consta generalmente de tres pasos:

- **Identificación:** La identificación no es más que la acción por parte de un usuario de presentar su identidad a un sistema, usualmente se usa un identificador de usuario. Establece además que el usuario es responsable de las acciones que lleve a cabo en el sistema, lo que se relaciona con los

registros de auditoría que permiten guardar las acciones realizadas dentro del sistema y rastrearlas hasta el usuario autenticado. (4)

- **Autenticación:** Es el proceso de verificación de la identidad digital de un remitente de una comunicación que hace una petición para conectarse a un sistema. El remitente puede ser una persona que usa un ordenador u otro medio electrónico, un ordenador por sí mismo o un programa. Es un modo de asegurar que los usuarios son realmente quienes dicen ser. (4)

- **Autorización:** Es el proceso por el cual se autoriza al usuario identificado a acceder a determinados recursos del sistema, es decir, se comprueba que los usuarios con identidad válida solo tengan acceso aquellos recursos sobre los cuales tengan privilegios. (4)

1.1.5. Técnicas de identificación y autenticación

Técnicas para realizar autenticación de identidad de usuarios:

- ✓ Algo que solamente el individuo conoce: por ejemplo una contraseña.
- ✓ Algo que la persona posee: por ejemplo una tarjeta magnética.
- ✓ Algo que el individuo es y que lo identifica unívocamente: por ejemplo las huellas digitales.
- ✓ Algo que solo el individuo es capaz de hacer: por ejemplo los patrones de escritura. (4)

Estas técnicas pueden ser utilizadas individualmente o combinadas que es la autenticación con varios factores.

La fortaleza de la autenticación es mayor mientras más factores se adicionen, generalmente solo se utilizan hasta 3 factores:

- **1 factor** = contraseña
- **2 factores** = contraseña + token[1]
- **3 factores** = contraseña + token + biometría
- **4 factores** = contraseña + token + biometría + localización geográfica (GPS[2]) (4)

[1] Un token de seguridad es un pequeño dispositivo de hardware que los usuarios cargan consigo para autorizar el acceso a un servicio de red. El dispositivo puede ser en forma de una tarjeta inteligente o puede estar incorporado en un objeto utilizado comúnmente, como un llavero.

1.1.6. Seguridad en aplicaciones web

La Seguridad en Aplicaciones Web, se encuentra relacionada pura y exclusivamente con: la lógica, la escritura de código y el contenido de una aplicación web. Toda aplicación web, requiere de un entorno conformado por elementos externos, tales como sistemas operativos, servidor web y servicios.

La seguridad en la Web está formada por tres etapas fundamentales:

- ✓ **Seguridad de la computadora del usuario:** Los usuarios deben contar con navegadores y plataformas seguras, libres de virus y vulnerabilidades. También debe garantizarse la privacidad de los datos del usuario. (6)
- ✓ **Seguridad del servidor Web y de los datos almacenados:** Se debe garantizar la operación continua del servidor, que los datos no sean modificados sin autorización (*integridad*) y que la información solo sea distribuida a las personas autorizadas (*control de acceso*). (6)
- ✓ **Seguridad de la información que viaja entre el servidor Web y el usuario:** Garantizar que la información en tránsito no sea leída (*confidencialidad*), modificada o destruida por terceros. También es importante asegurar que el enlace entre cliente y servidor no pueda interrumpirse fácilmente (*disponibilidad*). (6)

1.1.7. Ataques sobre aplicaciones web

Las cinco acciones más usuales para explotar aplicaciones Web son:

1. **Ejecución de código remota:** Como su nombre lo indica, esta vulnerabilidad permite al atacante ejecutar código en el servidor vulnerable y obtener información almacenada en él. (6)
2. **Inyección SQL:** Es una vulnerabilidad de las Web, que afectan directamente a las bases de datos de una aplicación, el problema radica al filtrar erróneamente las variables utilizadas en parte de la página con código SQL, consiste en insertar o inyectar código SQL malicioso dentro de código SQL, para alterar el funcionamiento normal y hacer que se ejecute el código “invasor” dentro del sistema. (6)

[2] Es un sistema global de navegación por satélite que permite determinar en todo el mundo la posición de un objeto, una persona o un vehículo con una precisión hasta de centímetros.

3. **Vulnerabilidades asociadas a cadenas:** Estos alteran el flujo de una aplicación utilizando las capacidades proporcionadas por las librerías de formato de cadenas para acceder a otro espacio de memoria. (6)

4. **Cross-Site Scripting:** Es una técnica de ataque que fuerza a un sitio Web a repetir un código ejecutable facilitado por el atacante, y que se cargará en el navegador del usuario. (6)

5. **Enumeración de usuarios:** Tipo de ataque donde los mensajes de validaciones de autenticación le dicen al atacante si el nombre de usuario proporcionado es correcto o no. Explotando esta vulnerabilidad el atacante puede experimentar con diferentes nombres de usuarios y determinar cuáles son válidos. (6)

1.2. Estudiando las estrategias de seguridad en aplicaciones

En los últimos tiempos son considerables los cambios alrededor de las aplicaciones y servicios web. Casi todos los fabricantes de software están orientando sus plataformas de desarrollo de aplicaciones para que se integren y utilicen las posibilidades de la web. (7)

Entendiendo por aplicación web a todo aquel software que interactúa con el usuario utilizando el protocolo HTTP. Por otra parte, los servicios web son un conjunto de funciones empaquetadas dentro de una entidad única y publicadas dentro de la red para que puedan ser utilizadas por las aplicaciones web. (7)

En el presente apartado se realizará un estudio de diversas estrategias y formas de aplicar seguridad a diferentes aplicaciones web, que si bien no concuerdan completamente con el negocio y marco de trabajo para con el sistema Quarxo instalado en el BNC, posibilita un acercamiento al entendimiento de cómo se aplica, funciona y se maneja la seguridad en los sistemas web en general, ofreciendo una visión general del problema en cuestión.

1.2.1. OWASP (Open Web Application Security Project)

El proyecto OWASP (Open Web Application Security Project) tiene como objetivo ofrecer una metodología, de libre acceso y utilización, que pueda ser utilizada como material de referencia por parte de los arquitectos de software, desarrolladores, fabricantes y profesionales de la seguridad involucrados en el diseño, desarrollo, despliegue y verificación de la seguridad de las aplicaciones y servicios web. (7)

Se propone la siguiente guía que empieza estableciendo el principio básico de seguridad que cualquier aplicación o servicio web debe cumplir:

- ✓ **Validación de la entrada y salida de información:** La entrada y salida de información es el principal mecanismo que dispone un atacante para enviar o recibir código malicioso contra el sistema. Por tanto, siempre debe verificarse que cualquier dato entrante o saliente es apropiado y en el formato que se espera. Las características de estos datos deben estar predefinidas y debe verificarse en todas las ocasiones. (7)

- ✓ **Diseños simples:** Los mecanismos de seguridad deben diseñarse para que sean los más sencillos posibles, huyendo de sofisticaciones que compliquen excesivamente la vida a los usuarios. Si los pasos necesarios para proteger de forma adecuada una función o modulo son muy complejos, la probabilidad de que estos pasos no se ejecuten de forma adecuada es muy elevada. (7)

- ✓ **Utilización y reutilización de componentes de confianza:** Debe evitarse reinventar la rueda constantemente. Por tanto, cuando exista un componente que resuelva un problema de forma correcta, lo más inteligente es utilizarlo. (7)

- ✓ **Defensa en profundidad:** Nunca confiar en que un componente realizará su función de forma permanente y ante cualquier situación. Ha de disponerse de los mecanismos de seguridad suficientes para que cuando un componente del sistema falle ante un determinado evento, otros sean capaces de detectarlo. (7)

- ✓ **Tan seguros como el eslabón más débil:** La frase "se garantiza la seguridad, ya que se utiliza SSL" es realmente muy popular, pero también es muy inexacta. La utilización de SSL garantiza que el tráfico en tránsito entre el servidor y el cliente se encuentre cifrado, pero no garantiza nada acerca de los mecanismos de seguridad existentes. (7)

- ✓ **La "seguridad gracias al desconocimiento" no funciona:** El simple hecho de ocultar algo no impide que, a medio o largo plazo, llegue a ser descubierto. Tampoco es ninguna garantía de que tampoco será descubierto a corto plazo. (7)

- ✓ **Verificación de privilegios:** Los sistemas deben diseñarse para que funcionen con la menor cantidad de privilegios posibles. Igualmente, es importante que los procesos únicamente dispongan de los privilegios necesarios para desarrollar su función, de forma que queden compartimentados. (7)

- ✓ **Ofrecer la mínima información:** Ante una situación de error o una validación negativa, los mecanismos de seguridad deben diseñarse para que faciliten la mínima información posible. De la misma forma, estos mecanismos deben estar diseñados para que una vez denegada una operación, cualquier operación posterior sea igualmente denegada. (7)

1.2.2. NETteller

El producto insignia de NETinfo para el mercado Bancario y Financiero es la plataforma NETteller, es un Sistema Bancario universal multi-canal, desarrollado con herramientas líderes en el mercado y adaptado a las prácticas bancarias internacionales. NETteller es una plataforma, capaz de administrar y manejar todos los canales de los clientes de forma centralizada. (8)

El desarrollo especializado del producto le permite operar en múltiples sistemas operativos, es completamente personalizable, escalable y fácil de integrar con los sistemas existentes. Permite a las instituciones financieras ofrecer a sus clientes una herramienta fácil de usar y potente para sus necesidades bancarias. Su eficacia y la fiabilidad son los factores claves para una rápida y fácil implementación. (8)

NETteller Seguridad: Gestiona todos los nombres de usuario y contraseña, validación de cliente, credenciales de usuario, encriptación de datos, registros transaccionales y de comportamiento de los usuarios, funcionalidad avanzada de auditoría y vigilancia. (8)

1.2.3. Websense ThreatSeeke

Websense, Inc., es actualmente la líder mundial en software de seguridad web y filtrado de contenido, esta compañía propone una nueva tecnología para la seguridad que identifica las amenazas en Internet antes de que los criminales puedan lanzar ataques a la seguridad web. (9)

La tecnología Websense ThreatSeeker de patente pendiente, ofrece protección preventiva contra las amenazas a la seguridad basadas en Web, amenazas que normalmente no pueden detectarse o es muy costoso prevenir usando tecnologías de seguridad como antivirus y sistemas de prevención de intrusos. A diferencia de estos métodos tradicionales, Websense busca amenazas en Internet antes de que los clientes sean comprometidos y los protege antes de que se creen los parches y firmas, sin los altos costos o cargas administrativas y sin tener que adivinar cómo lucirán los ataques en el futuro. En consecuencia, las organizaciones se protegen automáticamente contra las amenazas más recientes en cuestión de minutos. (9)

1.2.4. Soluciones para la seguridad realizadas en la UCI

En la facultad 10 de la Universidad de las Ciencias Informáticas se llevó a cabo el desarrollo de un software de autenticación y control centralizado para la corporación de PDVSA[3] capaz de cotejar las credenciales de los usuarios hacia todas las aplicaciones de dicho sistema, proporcionando una infraestructura para simular una autenticación única del usuario corporativo en una plataforma tecnológica compleja, en la que estos usuarios puedan acceder a diferentes aplicaciones con solo un conjunto de credenciales. (10)

Está aplicación trajo como resultado principal la importancia de mapeos entre diferentes entornos que manejan diferentes mecanismos de seguridad. (10)

Solución de ACAXIA

El componente de autenticación de ACAXIA utiliza el estándar SAML y brinda importantes funcionalidades, como son la inclusión de la firma y el certificado digital así como la implementación de una arquitectura de Single Sign-On a través del Proveedor de Servicios y el Proveedor de Identidad. La solución está realizada sobre Zend Framework. Permite la autenticación contra LDAP y OpenLDAP. (11)

La calidad y profundidad del registro de las identidades de los usuarios influye en la robustez y seguridad del control de acceso en general. Estos atributos pueden variar en función de las necesidades del entorno de despliegue, por esta razón es necesario que la solución de configuración permita el registro de

[3] Petróleos de Venezuela s.a

atributos y valores de forma dinámica. Debe gestionar además la fortaleza de las claves y emplear métodos criptográficos seguros para el envío, validación y almacenamiento de información sensible. (11)

Los sistemas antes mencionados, proporcionan múltiples ideas para la solución que se requiere, pero no se adecuan a las necesidades del producto, en la *tabla 1* se establecen una serie de indicadores que acotan el alcance de cada una de estas soluciones, determinando de esta forma la reusabilidad de esas ideas en la implementación de la solución definitiva.

Herramientas	Solución web Bancaria	Lenguaje Java	Integración Arquitectura Quarxo	Software Libre
OWASP	NO	NO	NO	SI
NETteller	SI	SI	NO	NO
SACC	NO	NO	NO	NO
ACAXIA	NO	NO	NO	SI
Websense ThreatSeeke	NO	NO	NO	NO

Tabla 1: Indicadores de las herramientas analizadas

Indicadores. Leyenda:

- **Solución Web Bancaria** indica si la herramienta está destinada específicamente a soluciones bancarias.
- **Lenguaje** indica si se corresponde con el lenguaje de implementación propuesto para la solución.
- **Integración Arquitectura Quarxo** indica si la arquitectura de la herramienta en cuestión es compatible con la arquitectura del sistema Quarxo.
- **Software Libre** indica si la herramienta está bajo la política de Software Libre.

De las herramientas para la seguridad en aplicaciones web estudiadas, se considera que ninguna de ellas están acordes a la solución que se requiere, ya que es necesario que cumplan con el cien por ciento de los indicadores expuestos anteriormente y como se muestra ninguna de ellas se integra con la arquitectura del sistema Quarxo, debido a que algunas ellas son privativas o de patente pendiente y no permiten tener acceso a los datos de su arquitectura.

1.3. Metodología de desarrollo de software

La informatización de los principales sectores de la sociedad cubana es hoy una de las mayores prioridades, de ahí que el uso de las herramientas informáticas ya es remarcable en estos sectores.

La UCI no ha quedado exenta de esta revolución informática, de hecho desempeña un rol protagónico desde su surgimiento mediante la producción de soluciones y servicios informáticos. El Centro de Informatización de la Gestión de Entidades (CEIGE) dirige sus resultados hacia la esfera de la gestión de empresas y entidades, la producción se concentra en el desarrollo de proyectos generalmente de gran magnitud, por lo que se hace necesario contar con un modelo estandarizado, que establezca las distintas fases por las que se debe transitar y el conjunto de artefactos a generar en cada una de ellas. (12)

Teniendo como precedente dicha necesidad y en colaboración con las distintas líneas de desarrollo donde se ejecutan cada uno de estos proyectos, se propone el ***Modelo de desarrollo de software para el CEIGE***. Se concreta como un proceso orientado a componentes, es decir, un conjunto de actividades para transformar los requisitos de un sistema. No se buscan pautas de todas las posibles acciones a realizar, sino un acumulado de metodologías flexibles al contexto y necesidades de cada organización. (12)

La producción en el centro se concentra en el desarrollo de proyectos generalmente de gran magnitud, por lo que se hace necesario contar con un modelo estandarizado, que establezca las distintas fases por las que se debe transitar y el conjunto de artefactos a generar en cada una de ellas. (12)

La flexibilidad y robustez son elementos que enaltecen la calidad del modelo propuesto, incluyendo además artefactos, siendo los productos tangibles del proceso y roles que caracterizan y definen las funcionalidades que desempeñaran los usuarios. (12)

Para el ciclo de vida de los proyectos del CEIGE y posterior especificación se tienen en cuenta las disciplinas y actividades por áreas de procesos que plantea el nivel dos de CMMI[4] establecido en la UCI. (12)

[4] Capability Maturity Model Integration.

Disciplinas de las que dispone el Modelo de Desarrollo de Software:

1. **Estudio preliminar:** La producción se concentra en el desarrollo de proyectos generalmente de gran magnitud, por lo que se hace necesario contar con un modelo estandarizado, que establezca las distintas fases por las que se debe transitar y el conjunto de artefactos a generar en cada una de ellas. (12)

2. **Modelado del negocio:** Es la fase destinada a comprender los procesos de negocio de la organización. Se comprende cómo funciona el negocio que se desea automatizar para tener garantías de que el software desarrollado va a cumplir su propósito. (12)

3. **Requisitos:** El esfuerzo principal en la fase de requisitos es desarrollar un modelo del sistema que se va a construir. Incluye un conjunto de artefactos que describen todas las interacciones que tendrán los usuarios con el software y que responden a los requisitos funcionales del sistema. Se especifican los requisitos funcionales y no funcionales. (12)

4. **Análisis y diseño:** Durante esta fase es modelado el sistema para que soporte todos los requisitos. Esto contribuye a una arquitectura sólida y estable que se convierte en un plano para la próxima fase. Los artefactos generados en esta etapa son más formales y específicos de una implementación. En caso de llevarse a cabo la reutilización de componentes de software ya desarrollados, durante esta fase se ajusta el modelado existente a los requisitos actuales. (12)

5. **Implementación:** A partir de los resultados del análisis y diseño se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ejecutables y similares. Al reutilizar componentes software ya implementados se lleva a cabo el desarrollo necesario para ajustar a los requisitos actuales y posteriormente realizar la integración de los componentes. (12)

6. **Pruebas internas:** Durante esta fase se desarrollan las pruebas del grupo de calidad del centro verificando el resultado de la implementación. Permite identificar posibles errores en la documentación y el software, es decir requisitos que el producto debería cumplir y que aún no los cumple. (12)

7. **Pruebas de liberación:** Se aplican pruebas diseñadas e implementadas por el Laboratorio Industrial de Pruebas de Software a todos los entregables de los proyectos antes de ser proporcionados al cliente para su aceptación. (12)

1.4. Patrones arquitectónicos

Un patrón arquitectónico es un patrón de alto nivel que determina la arquitectura global de una aplicación, proporcionando un esquema genérico probado para solucionar un problema en específico.

Proveen un vocabulario común y comprensible, facilitan la construcción de software con propiedades particulares y ayudan a construir arquitecturas heterogéneas y complejas. (13)

Con este objetivo se utiliza para la definición de la arquitectura de la aplicación Quarxo el patrón MVC.

MVC (Modelo Vista Controlador)

Este patrón propone dividir la aplicación en tres capas: el Modelo, la Vista y el Controlador; el modelo es la representación de los datos del sistema; la vista se encarga de todo lo relacionado con presentar la interfaz al usuario y el controlador es el encargado de escuchar los cambios en la vista y enviarlos al modelo, el cual remite a su vez los datos a la vista como un ciclo. El controlador es el que decide la vista y la información que se debe presentar. Al aplicarse este modelo se desacopla el modelo de las vistas y se hace a los sistemas flexibles y adaptables. (13)

1.5. Patrones de diseño

Posterior a la definición de la arquitectura de la aplicación, el diseño hace uso de varios patrones que constituyen la base para la búsqueda de soluciones a problemas que se presentan en situaciones comunes del diseño. Proponen una solución efectiva y probada para un problema general dentro del marco del diseño de software.

A continuación se describen un conjunto de patrones de diseño que son empleados en la aplicación, contribuyendo a su implementación organizada y eficiente.

1.5.1. Patrón de Acceso a Datos (DAO)

Engloba todo el acceso a datos en una capa independiente, desacoplando la lógica de negocio de la lógica de acceso a datos de manera que se pueda cambiar la fuente de datos fácilmente. Reduce la complejidad de los objetos de negocio al abstraerlos de la implementación real de la comunicación con la fuente de datos. (13)

1.5.2. Patrones de asignación de responsabilidades (GRASP)

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. (14)

Patrón Experto

Se asigna una responsabilidad al experto en la información, o sea, se definen las clases y sus funcionalidades atendiendo a la información concreta que deben manejar. De este modo se obtiene un diseño con mayor cohesión y la información se mantiene encapsulada (disminución del acoplamiento). (13)

Patrón Creador

Guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. (13)

Patrón Controller

Se asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas que tengan una posición intermediaria entre la interfaz de usuario y las clases donde reside la lógica de la aplicación, facilitando la centralización de actividades. (13)

Patrón Bajo Acoplamiento

Propone mantener pocas dependencias entre las clases, logrando que el código sea más fácil de entender, mantener y reutilizar. (13)

Patrón Alta Cohesión

Propone que cada elemento del diseño debe realizar una labor única dentro del sistema, logrando un alto grado de funcionalidad combinada con una reducida cantidad de operaciones, trayendo como ventaja que se simplifique el mantenimiento y que aumente la capacidad de reutilización. (13)

1.5.3. Patrones estructurales (GOF)

Por sus siglas en inglés *Gang of Four* (Banda de los Cuatro). Llamados así debido a que fueron cuatro autores los que escribieron el libro *Design Patterns* (Diseño de Patrones) que ilustra sus funciones. (15)

Facade (Fachada)

El objetivo de este patrón es proveer un intermediario entre dos grupos de objetos, disminuyendo el número de objetos con los que trabaja un cliente y simplificando el acceso de éste a un conjunto de objetos relacionados. (13)

Composite View (Vistas Compuestas)

Un objeto vista que está compuesto de otros objetos vista. Por ejemplo una página JSP que incluye otras páginas JSP y HTML usando la directiva include. La aplicación de este patrón hace que la representación de vistas sea más manejable gestionando los diferentes elementos de una página por medio de una plantilla. (13)

1.6. Ambiente de desarrollo

Para lograr reducir al mínimo el tiempo de desarrollo y lograr un producto con la calidad requerida, se hace necesario seleccionar correctamente las herramientas y tecnologías a utilizar teniendo en cuenta el contexto donde se aplicará el software.

Para el desarrollo del sistema Onyx se definen un conjunto de tecnologías, herramientas, lenguajes y patrones a utilizar, en base a la disponibilidad e idoneidad de los mismos considerando las características del proyecto y el entorno de despliegue. A continuación se expone una breve caracterización de cada uno de estos elementos.

1.6.1. Lenguajes y Herramientas

A continuación se mencionaran los lenguajes y herramientas que se usaran para el desarrollo de la solución, las mismas se corresponden al desarrollo tecnológico del sistema Quarxo.

UML

Se utiliza Unified Modeling Language (UML) en la solución para el modelado de los procesos de negocio y funciones del sistema. Incluye las clases, esquemas de base de datos y componentes reutilizables de software en un lenguaje determinado y soporta el paradigma orientado a objetos. (13)

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. (15)

Java

Java es un lenguaje de programación orientado a objetos, robusto y fácil de aprender, permite la codificación de la propuesta de solución de una manera rápida y flexible, al integrarse con los diferentes frameworks de desarrollo, permite programar páginas web dinámicas con accesos a bases de datos utilizando XML con cualquier tipo de conexión de red entre cualquier sistema.

Es un lenguaje multiplataforma de código abierto, distribuido, que proporciona un conjunto de clases para su uso en aplicaciones de red, permitiendo abrir sockets y establecer conexiones con servidores o clientes remotos. Además de ser bastante confiable manejando excepciones, Java fue diseñado para crear software altamente fiable. (15)

JavaScript

Es un lenguaje que no requiere compilación y se utiliza comúnmente para la construcción de páginas web en combinación con el XHTML. Se caracteriza por ser un lenguaje que responde a eventos generados por el usuario o por el navegador, es independiente de la plataforma necesitando solo de un navegador para ejecutar el código, permite un desarrollo rápido y es relativamente fácil de aprender. Es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Mozilla Firefox y Google Chrome. (13)

Plataforma J2EE

Como plataforma de desarrollo se decide utilizar Java Enterprise Edition (JEE). La misma define un estándar para el desarrollo de aplicaciones empresariales basándolas en componentes modulares y estandarizados, proporcionando un conjunto completo de servicios a estos componentes, y manejando muchas de las funciones de la aplicación de forma automática sin necesidad de una programación compleja. (13)

Contenedor Web (Tomcat 6.0)

Como servidor de aplicaciones se determina utilizar el Apache Tomcat en su versión 6.0, el cual es un contenedor de servlets que implementa las especificaciones de JavaServer Page (JSP) y funciona como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. (13)

Entorno de Desarrollo Integrado (Eclipse 3.3)

Se selecciona como entorno de desarrollo integrado Eclipse en su versión 3.3. Esta herramienta es multiplataforma, de código abierto y presenta una arquitectura de plugins que lo hace ser bastante flexible y configurable. (12)

Visual Paradigm 8.0

La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Dicha herramienta ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta de software libre de probada utilidad para el analista. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque orientado a objetos. (14)

Erwin 7.5

Para el modelado de la base de datos se utiliza el Erwin en su versión 7.5 que permite diseñar, generar y mantener aplicaciones de bases de datos, desde un modelo lógico hasta un modelo físico, todo esto con un gran nivel de facilidad. (12)

Control de versiones (SubVersion 1.6.6)

Es un software de sistema de control de versiones. Una característica importante de SubVersion es que, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo. SubVersion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer por que la calidad del mismo vaya a verse afectada, si se ha hecho un cambio incorrecto a los datos, simplemente deshaga ese cambio. (15)

Base de Datos (SQL)

SQL Server es un servidor de base de datos basado en el modelo relacional. Se caracteriza por brindar soporte para transacciones y procedimientos almacenados y posee como lenguajes de consulta al SQL y al T-SQL (en inglés: Transact-SQL). Requiere para su funcionamiento del sistema operativo Microsoft Windows, atentando contra la premisa de desarrollar las aplicaciones cubanas sobre software libre. Su uso está basado en una decisión del BNC y dado que la solución de seguridad planteada será para el sistema Quarxo de dicha institución se hizo conveniente su aplicación. (12)

Aunque debido a que la solución de seguridad es independiente a Quarxo y cuenta con muy pocas tablas, además utiliza el framework Hibernate se hace muy sencillo la migración a otra base de datos, aunque no es recomendable ya que quedarían implantados en el mismo servidor 2 gestores de base de datos sin ninguna necesidad real.

Lenguajes adicionados

Los lenguajes que se muestran a continuación fueron adicionados al proceso de desarrollo de la solución actual, tratando de aprovechar las ventajas que brindan los mismos para lograr no solo darle cumplimiento al problema en cuestión sino para enriquecer la solución en cuanto a calidad visual y al mismo tiempo lograr un mayor grado de compatibilidad y ajuste a las versiones de los navegadores actuales.

JQuery 1.8.2

jQuery es una biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código. Con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio. (16)

HTML5

HTML5 (HyperText Markup Language, versión 5) es la quinta revisión importante del lenguaje básico de la World Wide Web. HTML5 especifica dos variantes de sintaxis para HTML: un “clásico” HTML (text/html), la variante conocida como HTML5 y una variante XHTML conocida como sintaxis XHTML5 que deberá ser tratada como XML (XHTML) (application/xhtml+xml). Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo. (17)

HTML5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Algunos de ellos son técnicamente similares a las etiquetas, pero tienen un significado semántico. Otros elementos proporcionan nuevas funcionalidades a través de una interfaz estandarizada. (18)

CSS3

CSS3 está dividida en varios documentos separados, llamados "módulos". Cada módulo añade nuevas funcionalidades a las definidas en CSS2, de manera que se preservan las anteriores para mantener la compatibilidad. Permite el control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo. Además optimiza el ancho de banda de la conexión, pudiendo definirse los mismos estilos para muchos elementos con un sólo selector; o un mismo archivo CSS puede servir para una multitud de documentos. (19)

1.6.2. Framework

A continuación se explicaran los *frameworks* o marcos de trabajo seleccionados para el desarrollo de la solución de seguridad.

Spring 3.1.0

Se emplea para el desarrollo de la aplicación el Spring Framework en su versión 3.1.0, el cual contiene un conjunto de librerías de clases que brindan una estructura conceptual y tecnológica para el desarrollo de aplicaciones de la plataforma J2EE. Permite crear soluciones bien documentadas, seguras, robustas y ofrece gran libertad a los desarrolladores en java. Además en su implementación tiene en cuenta las tres capas arquitectónicas: acceso a datos, negocio y presentación. Este framework es actualizado con frecuencia, solucionando errores son detectados por el equipo de desarrollo o notificados por los usuarios, mejorando de esta forma la solución.(12)

Spring MVC 3.1.0

Dentro del Spring Framework se encuentra el Spring MVC, utilizado para atender las peticiones web simples con navegación lineal a través de clases "Controller". Debido a la versión utilizada para la implementación, esta incluye un conjunto de anotaciones que facilitan la gestión de los controladores y la unificación de los mismos bajo una misma anotación, en este caso "@Controller", evitando de esta forma el uso de los disímiles tipos de controladores que poseía Spring en versiones anteriores.

Spring DAO 3.1.0

Spring DAO es un módulo perteneciente al framework Spring. Posee algunas librerías de clases para trabajar con base de datos a través del API[5] de Java JDBC[6] y brinda soporte para invocar procedimientos y funciones almacenadas. Ofrece además una capa de excepciones para la gestión de los errores emitidos por los servidores de base de datos. Ayuda a mantener el código simple y evita que sea muy repetitivo, además minimiza los errores al intentar cerrar la conexión con algunas bases de datos. (12)

[5] Application Programming Interface.
[6] Java Data Base Connectivity

Spring ORM 3.1.0

Spring ORM (Object Relation Mapping) es un módulo de Spring que permite su integración con los frameworks ORM más populares en el mercado. Dicha integración proporciona mayor seguridad, facilidad y eficiencia en la gestión de sesiones, recursos, transacciones ORM, excepciones y pruebas. Es utilizada en el proyecto para lograr la integración de Spring con el framework ORM Hibernate. (12)

Spring Security 3.1.0

Spring Security es un sub-proyecto de Spring que permite gestionar completamente la seguridad de aplicaciones Java. Ofrece servicios de identificación de usuarios y acceso a los recursos sin necesidad de añadir código; proporciona asignación de permisos a usuarios o grupos de usuarios; y facilita, al separar claramente la seguridad de un sistema de su lógica de negocio, la modificación o adición de seguridad a aplicaciones existentes. Además posee varias formas de configuración, permitiendo ir desde lo más general hasta lo más específico y reemplazar, modificar o ajustar a un comportamiento determinado las funcionalidades deseadas. (12)

Hibernate 3.5

Para la gestión del acceso a datos se decidió usar el Hibernate framework. Una de sus características principales es que abstrae el trabajo con la base de datos, soportando la manipulación de los datos persistentes objetualmente a través de ficheros que mapean clases Java contra tablas. Permite transacciones, asociaciones, polimorfismo, herencia, carga mediante referencia (carga lazy, consiste en cargar sólo la referencia de donde se encuentran los datos), persistencia transitiva y estrategias de fetching. Permite la ejecución de consultas SQL (Standard Query Language) y además posee un potente lenguaje de consulta de datos llamado HQL (Hibernate Query Language), que permite realizar consultas basándose en los objetos del negocio y no en las tablas de la base de datos. (12)

Reemplazo del framework de presentación

Se decidió reemplazar el framework Dojo por Bootstrap, ya que simplifica el proceso de creación de diseños web combinando CSS y JavaScript siendo mucho más ligero, debido en gran medida a la cantidad de librerías y archivos dependientes de Dojo siendo aproximadamente 18 mil, no así en el caso de Bootstrap, que cuenta con no más de 10, ayudando a la portabilidad de la solución.

Bootstrap 2.3.1

Twitter Bootstrap es una colección de herramientas de software libre para la creación de sitios y aplicaciones web. Contiene plantillas de diseño basadas en HTML5 y CSS3 con tipografías, formularios, botones, gráficos, barras de navegación y demás componentes de interfaz, así como extensiones opcionales de JavaScript. Además ofrece herramientas de andamiaje (scaffolding) para estructurar el LAYOUT de manera estática o líquida y componentes JavaScript reutilizables, todos realizados adoptando las mejores prácticas y permitiendo a los programadores extenderlos. (20)

1.7. Conclusiones del capítulo

Se realizó un estudio de los principales conceptos de seguridad, haciendo énfasis en los relacionados con las aplicaciones web debido a las características de la solución de seguridad que se propone, con el objetivo de crear la base de la investigación.

Después de realizar una investigación de las diferentes estrategias de seguridad aplicadas a diversos sistemas web, se anuló la utilización de las aplicaciones informáticas que no se ajustaban al negocio, por no corresponderse con la solución que se necesita. Se decidió utilizar Spring Security como el framework fundamental para garantizar la seguridad de la aplicación, teniendo en cuenta las principales funcionalidades del mismo.

Las tecnologías seleccionadas para el desarrollo de la solución están acordes a los requerimientos del cliente, a las políticas del centro y a los estándares internacionales.

Capítulo 2: Modelamiento del Negocio, Requisitos, Análisis y Diseño

2. Introducción

En este capítulo se realizan las primeras disciplinas propuestas por el modelo de desarrollo de CEIGE para el desarrollo de la solución propuesta: Requisitos, Análisis y Diseño. En la etapa de Requisitos se identifican y describen los requisitos funcionales y no funcionales. En el Análisis y el Diseño, es modelado el sistema para que soporte todos los requisitos. Se describe la arquitectura base para el desarrollo de este tipo de aplicación.

2.1. Modelado del negocio

El modelado del negocio es una disciplina de la fase Desarrollo del Modelo del CEIGE utilizado para desarrollar la solución de seguridad para Quarxo.

Es una disciplina destinada a entender los procesos de negocio y a través de ella se deben identificar y analizar los procesos que se llevan a cabo en el negocio que se desea automatizar, con el objetivo de organizar y documentar todas las acciones a tener en cuenta en el análisis para el desarrollo del software.

Los artefactos que se tienen en cuenta durante el desempeño de esta disciplina son la descripción de los procesos de negocio, el modelo conceptual y las reglas del negocio, los mismos son generados por cada proceso identificado: A partir del modelo de negocio que se obtenga se pueden derivar los requisitos del sistema.

Durante el desarrollo de esta disciplina se identificaron 5 procesos de negocio (PN)

- PN1: Gestionar Usuario
- PN2: Gestionar Rol.
- PN3: Buscar Logs.
- PN4: Consultar Estadísticas.
- PN5: Generar Reportes.

A continuación se describe el PN1: Gestionar Usuario.

Descripción del proceso de negocio PN1: Gestionar Usuario.

La *ilustración 1* representa el diagrama de proceso de negocio *Gestionar Usuario*. El objetivo de este proceso es la gestión de la información y estado de los usuarios del sistema.

En este proceso el administrador del sistema tiene acceso al registro y búsqueda de usuarios, en el último caso si no se encuentra ningún usuario el proceso termina en caso contrario le da acceso a otras 3 operaciones: *desbloquear usuario*, *asignar contraseña* y *actualizar usuario*, las cuales una vez finalizadas dan fin al proceso.

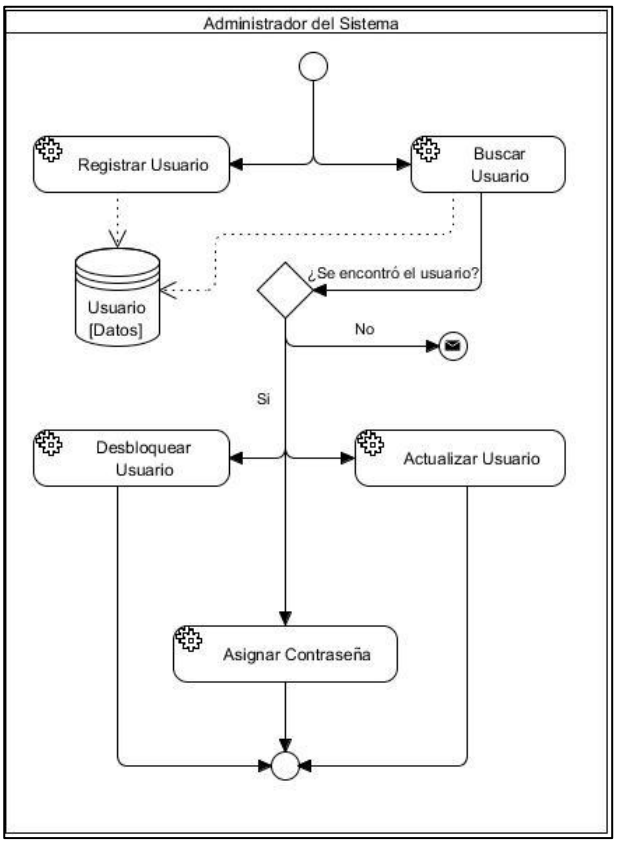


Ilustración 1: Diagrama de Proceso "Gestionar Usuarios"

Los restantes procesos de negocios pueden ser consultados en el documento de tesis en formato digital en los anexos (**Ver Anexo 14**).

Aunque debido a la significatividad del proceso de negocio *Gestionar Usuario*, se incluyeron los artefactos del mismo en el los anexos. **(Ver Anexo 19)** Todos estos artefactos cuentan con la estructura que se muestra a continuación.

- ✓ **Gestionar Usuario**
  - CIG-QF2-N-SEG-i1201 Descripción de proceso de negocio
  - CIG-QF2-N-SEG-i1301 Modelo conceptual
  - CIG-QF2-N-SEG-i1701 Reglas de negocio
- ✓ **Gestionar Rol**
  - CIG-QF2-N-SEG-i1202 Descripción de proceso de negocio
  - CIG-QF2-N-SEG-i1302 Modelo conceptual
  - CIG-QF2-N-SEG-i1702 Reglas de negocio
- ✓ **Buscar Logs**
  - CIG-QF2-N-SEG-i1203 Descripción de proceso de negocio
  - CIG-QF2-N-SEG-i1303 Modelo conceptual
  - CIG-QF2-N-SEG-i1703 Reglas de negocio
- ✓ **Consultar Estadísticas**
  - CIG-QF2-N-SEG-i1204 Descripción de proceso de negocio
  - CIG-QF2-N-SEG-i1304 Modelo conceptual
  - CIG-QF2-N-SEG-i1704 Reglas de negocio
- ✓ **Generar Reportes**
  - CIG-QF2-N-SEG-i1205 Descripción de proceso de negocio
  - CIG-QF2-N-SEG-i1305 Modelo conceptual
  - CIG-QF2-N-SEG-i1705 Reglas de negocio

2.2. Requisitos

Esta disciplina tiene como propósito fundamental guiar el desarrollo hacia un sistema correcto. Esto se consigue mediante una descripción de los requisitos del sistema suficientemente buena como para que pueda llegarse a un acuerdo entre el cliente y los desarrolladores sobre qué debe y qué no debe hacer el sistema. (24)

Un requisito es una condición o capacidad que tiene que ser alcanzada por un sistema o componente de un sistema para satisfacer un contrato, estándar u otro documento impuesto formalmente. (25)

2.2.1. Requisitos funcionales (RF)

Los requisitos funcionales (RF) son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer. (25)

A continuación se listan los requisitos funcionales asociados con la propuesta de solución obtenidos a partir del análisis de los 5 procesos de negocio en la disciplina Modelamiento de negocio:

Gestionar Usuario

Los usuarios son los que rigen el sistema administrativo y la aplicación a proteger, tendrán acceso a una serie de operaciones que le permitirán gestionar los usuarios, así como la información que lo conforma.

- **RF 01** Registrar Usuario. (Prioridad Alta) (**Ver Anexo 3**)
- **RF 02** Actualizar Usuario. (Prioridad Alta) (**Ver Anexo 4**)
- **RF 03** Asignar Contraseña. (Prioridad Alta) (**Ver Anexo 5**)
- **RF 04** Desbloquear Usuario. (Prioridad Alta) (**Ver Anexo 6**)
- **RF 05** Buscar Usuario. (Prioridad Alta) (**Ver Anexo 7**)
    - El sistema permitirá buscar, filtrar y ordenar usuarios a partir de los siguientes criterios:
        - Usuario del sistema
        - Nombre del usuario
        - IP
        - Habilitado
        - Bloqueado
        - Fecha del último cambio de la contraseña.

Gestionar Rol

Cada usuario tiene uno o varios roles, estos son los que definen los permisos y el nivel de accesibilidad de los usuarios al sistema, restringiéndolo de esta forma a las funcionalidades que le son necesarias para su desempeño en el mismo.

- **RF 06** Registrar Rol. (Prioridad Alta) (**Ver Anexo 8**)
- **RF 07** Actualizar Rol. (Prioridad Alta) (**Ver Anexo 9**)
- **RF 08** Buscar Rol. (Prioridad Alta) (**Ver Anexo 10**)
    - El sistema permitirá buscar, filtrar y ordenar roles a partir de los siguientes criterios:
        - Nombre del Rol
        - Descripción del Rol.

Buscar Logs

El listado de logs nos permite revisar el flujo de operaciones en el sistema, posibilitando un mayor control sobre las acciones de los usuarios en el sistema.

- **RF 09** Buscar Logs. (Prioridad Alta) (**Ver Anexo 11**)
    - El sistema permitirá buscar, filtrar y ordenar logs a partir de los siguientes criterios:
        - Usuario del sistema
        - IP
        - Tipo de log
        - Descripción
        - Fecha

Consultar Estadísticas

Las estadísticas brindan una visión del comportamiento y tendencia de los datos, permitiendo llegar a conclusiones que de otra forma seria muy complejo lograr.

- **RF 10** Consultar Estadísticas. (Prioridad Baja) (**Ver Anexo 12**)

- Muestra la cantidad de funcionalidades por roles existentes así como la cantidad de usuarios por roles existentes en el sistema en por ciento.

Generar Reportes

Los reportes son informes que organizan y exhiben los datos contenidos en una base de datos, con un diseño atractivo y fácil de interpretar por los usuarios. Esto contribuye en gran medida a la toma de mejores decisiones y permite a su vez conocer cuán cerca o lejos se está de alcanzar las metas u objetivos trazados.

- **RF 11** Generar reportes. (Prioridad Media) (**Ver Anexo 13**)
  - Genera un reporte en formato PDF.

2.2.2. Requisitos no funcionales

Los requisitos no funcionales (RNF) son restricciones que afectan a los servicios o funciones del sistema, tales como restricciones de tiempo, sobre el proceso de desarrollo y estándares. Definen propiedades y restricciones del sistema (25). De acuerdo con las características sobre las cuales se desarrolla la solución se toman en cuenta los siguientes RNF. (*Ver tabla 2*)

Tipo de requisito	PCs clientes	PCs servidores	
		Servidor 1	**Servidor 2**
Software	Máquina virtual de Java 6.0u20 o superior. Mozilla Firefox 12 o superior.	Windows Server 2003. Apache Tomcat 6.0 o superior. Máquina virtual de Java 6.0u20 o superior.	Windows Server 2003. Microsoft SQL Server 2005 o superior
Hardware	Procesador Pentium IV o superior 2.0 GHZ. RAM: 256 MB(recomendado 512) Una tarjeta de red.	Procesador: Core 2 Duo 2.0 GHZ o superior RAM: 4 GB Disco duro: 160 GB UPS: 1 Lector de CD: 1	

Tabla 2: Requisitos no funcionales

Funcionalidad

1. El sistema mostrará los errores en forma de mensajes.
   - Todos los mensajes de error del sistema deberán incluir una descripción textual del error.

Usabilidad

1. Los formularios serán estandarizados, por tanto:
    - Los campos de texto tendrán un tamaño estándar, de acuerdo con el espacio que se tenga en el área de la página y en la medida que se llene esa área primaria agregar la barra de desplazamiento vertical.
    - No se utilizarán textos extensos para las etiquetas de la interfaz de usuario.
2. El menú de navegación estará disponible en todas las páginas.
3. En caso de que los resultados de las consultas tengan más de 10 coincidencias, estos se mostrarán de forma paginada en una tabla.
    - Se mostrará en la parte inferior de la tabla el total de elementos encontrados, enlaces de navegación: ir hacia delante, hacia atrás o ir al inicio de los resultados mostrados.

Fiabilidad

1. El sistema estará disponible durante toda la jornada laboral del BNC.

Seguridad

1. El sistema permitirá la visualización de la información según el usuario autenticado.
2. El sistema implementará el uso de campos obligatorios y validaciones para garantizar la integridad de la información que se introduce por el usuario.

Restricciones de diseño

1. El sistema se implementará usando la plataforma JEE.
2. El sistema estará basado en un estilo arquitectónico en capas y arquitectura basada en componentes.

Interfaz de usuario

1. Todos los textos y mensajes en pantalla aparecerán en idioma español. Los errores serán visibles al usuario y en lo posible incluirán sugerencias de las posibles soluciones.
2. El sistema presentará los términos capitalizados, es decir, tendrán su primera letra en mayúsculas.

2.2.3. Validación de requisitos mediante prototipos y casos de pruebas

Se presentaron los prototipos (**Ver Anexo #15**) elaborados durante la especificación al analista principal y al jefe de línea del proyecto, para corroborar que responden a las necesidades y aspiraciones del cliente. Para ello, se desarrollaron varios escenarios posibles con el apoyo de juegos de datos, de forma tal que se visualizaron las diferentes funcionalidades que tendría el componente. Las no conformidades fueron documentadas y corregidas.

Revisión técnica formal de artefactos

Se realizaron revisiones de los artefactos por parte del Analista principal del proyecto y el grupo de calidad del centro, corrigiéndose las no conformidades identificadas.

2.3. Análisis y Diseño

Durante esta fase es modelado el sistema para que soporte todos los requisitos. Esto contribuye a una arquitectura sólida y estable que se convierte en una guía para la próxima fase. Los artefactos generados en esta etapa son más formales y específicos de una implementación. (11)

2.3.1. Arquitectura base de Onyx

La arquitectura es el resultado de acoplar elementos arquitectónicos de forma apropiada para satisfacer los requerimientos funcionales y no funcionales de un sistema. Se decidió utilizar el estilo arquitectónico “Llamada y retorno”, sub estilo “arquitectura en capas”. Aunque al ver a cada módulo como un componente, la relación entre ellos indica el uso de algunos rasgos de estilos basados en componentes, definiendo entonces la arquitectura como un hibrido de estos estilos, donde en un primer nivel de abstracción se encuentra la clásica arquitectura Cliente-Servidor, en un segundo nivel los componentes y sus relaciones, y en un 3er nivel se hallan 3 +1 capas (*ver Ilustración* 2). (26)

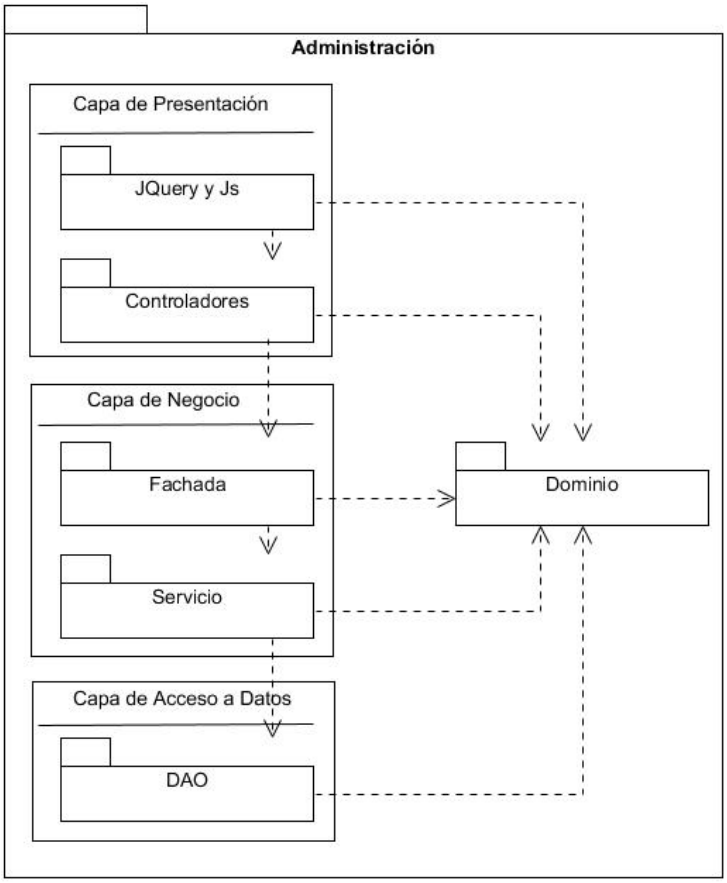


Ilustración 2: Tercer nivel de abstracción de la arquitectura de Onyx

Con el uso de la arquitectura en capas se logra alto nivel de abstracción ya que para implementar los niveles superiores no se requiere conocer el entorno subyacente, basta con conocer las interfaces que proporcionan los niveles inferiores, el mantenimiento del sistema se logra fácilmente teniendo en cuenta que al realizar algún cambio en alguna de las capas normalmente no afectan a sus capas superiores e inferiores. (26)

Componentes de la solución de seguridad

La aplicación está dividida en dos componentes:

Componente Seguridad: El mismo está compuesto por un conjunto de filtros interrelacionados entre sí y brinda una serie de servicios asociados fundamentalmente a dichos filtros. Este componente es el núcleo

donde se re-implementaron y usaron las funcionalidades brindadas por Spring Security basados en la idea de empaquetarlo como un *.jar* el mismo no cuenta con interfaces visuales.

Componente Administración: Este se encarga de la gestión de usuarios, roles, sistema de trazas así como la asignación de permisos entre los roles y las funcionalidades correspondientes.

Teniendo en cuenta lo antes descrito se muestra la relación entre los mismos. (*Ver Ilustración* 3)

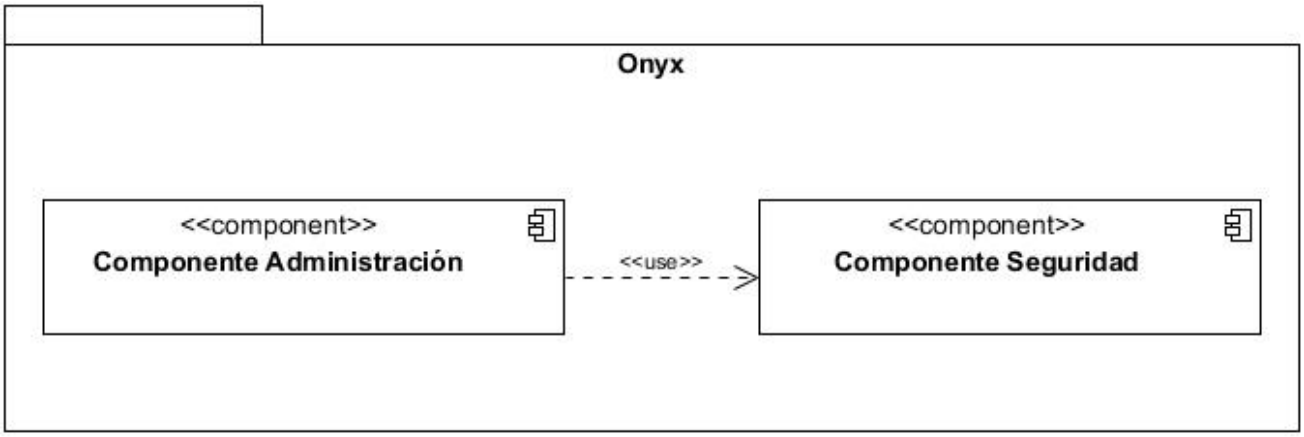


Ilustración 3: Relación entre los componentes de Onyx

2.3.2. Capa de acceso a datos

Esta capa contiene todas las operaciones relacionadas con la persistencia y acceso a información de la base de datos, a través de llamadas a funciones y de instrucciones básicas utilizando los frameworks Spring con Hibernate, cuya integración es posible mediante el uso de Spring ORM.

En esta capa se puede hacer uso de una propiedad denominada persistencia de objetos, que permite vincular objetos de bases de datos relacionales a objetos de lenguajes de programación como Java, para aumentar el nivel de abstracción y facilitar el acceso a los datos desde la capa de negocio. Existen varias implementaciones tecnológicas sobre persistencia que deberán emplearse atendiendo a las necesidades de cada aplicación. (27)

2.3.3. Capa de negocio

Esta capa está divida en dos subcapas principales sin dejar de incluir otras que se necesiten y que estén relacionadas con el negocio. En la Fachada se expondrán todas las funcionalidades que la capa de presentación necesitará. Esta capa invocará métodos de la subcapa de Desarrollo del negocio. En la capa de Desarrollo del negocio se implementará el negocio de los módulos en cuestión, y de aquí se accederá de ser necesario a la Capa de Acceso a Datos, a otras Capas de Negocios y/o a la Capa de Dominio.

2.3.4. Capa de presentación

En esta capa se desarrolla la lógica de presentación, se reciben y controlan los pedidos de la interfaz de usuario y se envían las respuestas procesadas a dichas peticiones, a través de la utilización de las funcionalidades brindadas por la capa de servicios. Para representar y controlar los flujos de presentación complejos se utilizan los controladores de Spring MVC para responder a peticiones simples y lineales. En la parte del cliente se hace uso de las librerías Bootstrap y jQuery para generar y diseñar los componentes visuales con los que interactúa el usuario.

2.3.5. Diagrama de paquetes del diseño

Durante el desarrollo de software resulta muy conveniente agrupar clases y ficheros por diferentes criterios para lograr la organización y facilitar la comprensión del código de la aplicación, resultando conveniente el desarrollo de los diagramas de paquetes, los cuales muestran cómo está dividido el sistema en agrupaciones lógicas mostrando las dependencias entre las mismas. (28)

A continuación se muestra la estructura y dependencia de paquetes del componente de administración con su correspondiente explicación. (*Ver ilustración 4*)

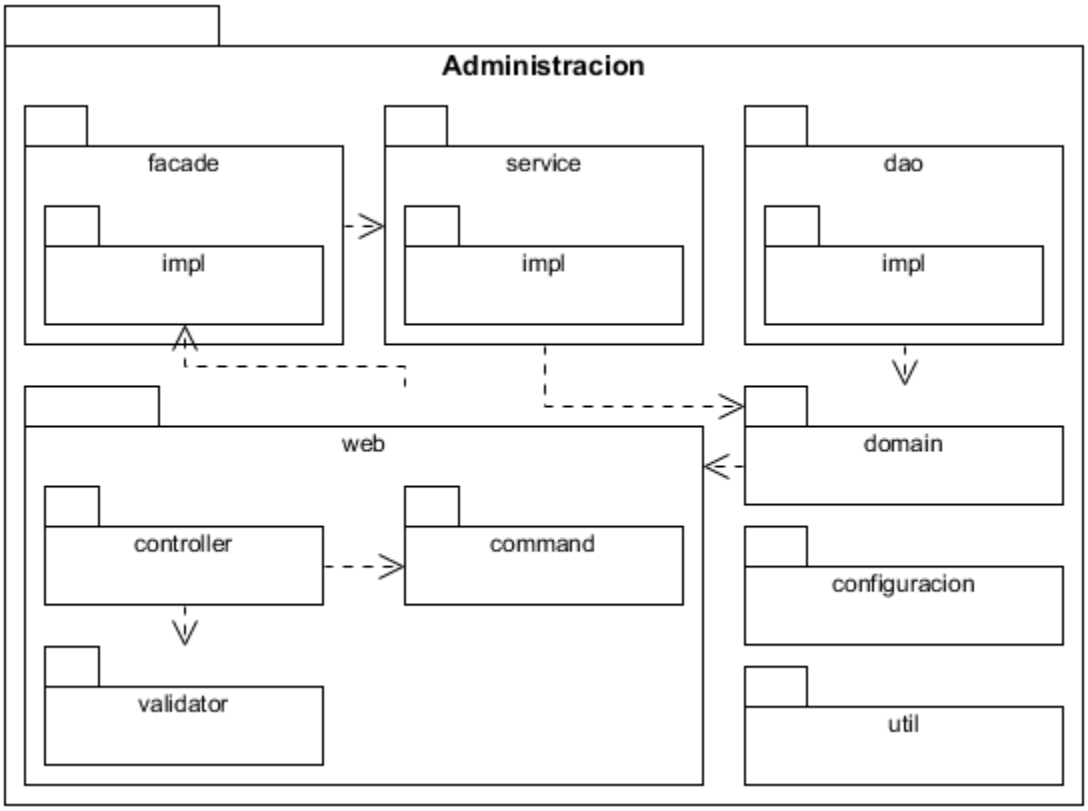


Ilustración 4: Diagrama de paquetes del diseño del Componente de Administración

Paquete *facade*: Agrupa las interfaces y sus respectivas implementaciones, encargadas de brindar las funcionalidades que serán usadas por la presentación.

Paquete *service*: En este paquete se hallan las interfaces y las implementaciones que comprenden la lógica de negocio que existe en el componente de administración, que será brindada a las capas superiores.

Paquete *dao*: Posee las interfaces y las implementaciones encargadas de brindar los servicios de comunicación con la base de datos.

Paquete *domain*: Paquete con las clases relacionadas con el dominio del módulo en cuestión.

Paquete *util*: Paquete con las clases encargadas de la búsqueda y lectura del fichero XML conformado por el cliente con las funcionalidades a proteger. Además cuenta con el fichero "*securityConfig.properties*" que almacena algunas configuraciones por defecto utilizadas por el contexto del framework de seguridad al iniciarse.

Paquete *configuracion*: En este paquete se encuentran el fichero de configuración en formato XML del contexto de Spring.

Paquete *web*: El paquete web agrupa un conjunto de clases y paquetes que son los encargados de realizar todo lo referente a la presentación en el lado del servidor, estos paquetes se describen a continuación:

- **Paquete *controller*:** Clases que extienden de los controladores propuestos por Spring MVC, encargadas de responder a las peticiones realizadas por el cliente.
- **Paquete *validator*:** Cuenta con las clases encargadas de realizar la validación de datos en el lado del servidor.
- **Paquete command:** Cuenta con las clases utilizadas en la presentación para la almacenar los datos y enviarlos al servidor.

Seguidamente se muestra la estructura y dependencia de paquetes del componente seguridad con su correspondiente explicación. (*Ver ilustración 5)*

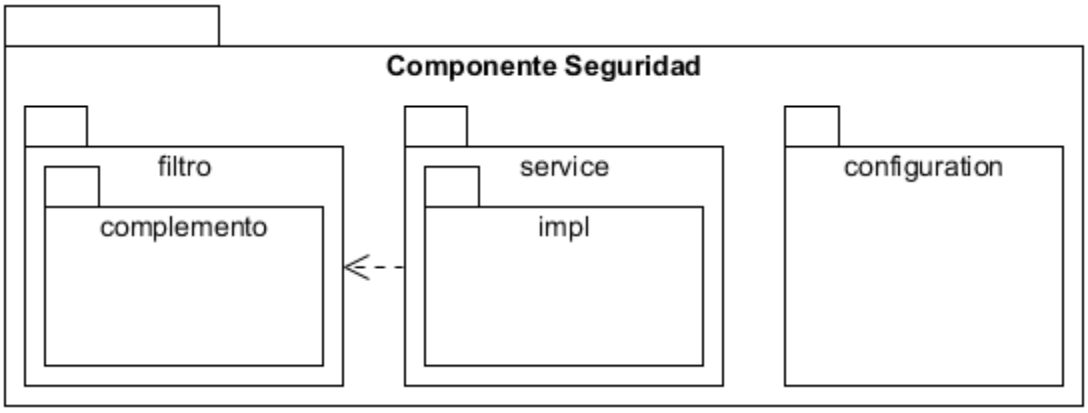


Ilustración 5: Diagrama de paquetes del diseño del Componente de Seguridad

Paquete *configuration*: En este paquete se encuentran el fichero de configuración en formato XML del contexto de Spring Security.

Paquete *filtro*: Contiene los filtros de seguridad nuevos y modificados y las clases complementarias que redefinen su uso.

Paquete *service*: En este paquete se hallan las interfaces y las implementaciones que comprenden los servicios brindados por el componente de seguridad.

2.3.6. Diagramas de clases del diseño

Un diagrama de clases del diseño describe como se realiza un requisito específico y como se ejecuta en término de clases de diseño y sus objetos.

Específicamente los diagramas de clases de diseño son muy útiles porque muestran a través de atributos y métodos la estructura de las clases que después serán escritas en algún lenguaje de programación. Además, representa la parte estática del sistema y las clases y sus relaciones. (22)

En el desarrollo de esta disciplina se obtuvo el diagrama de clases del diseño (**Ver Anexo #17**) de la solución, el diagrama de clases del diseño del requisito funcional "Registrar Usuario" es uno de los más significativos. A continuación se muestra el diagrama de clases del diseño contenido en el artefacto generado durante las disciplinas Análisis y diseño del Modelo de desarrollo de CEIGE. (*Ver ilustración 6)*

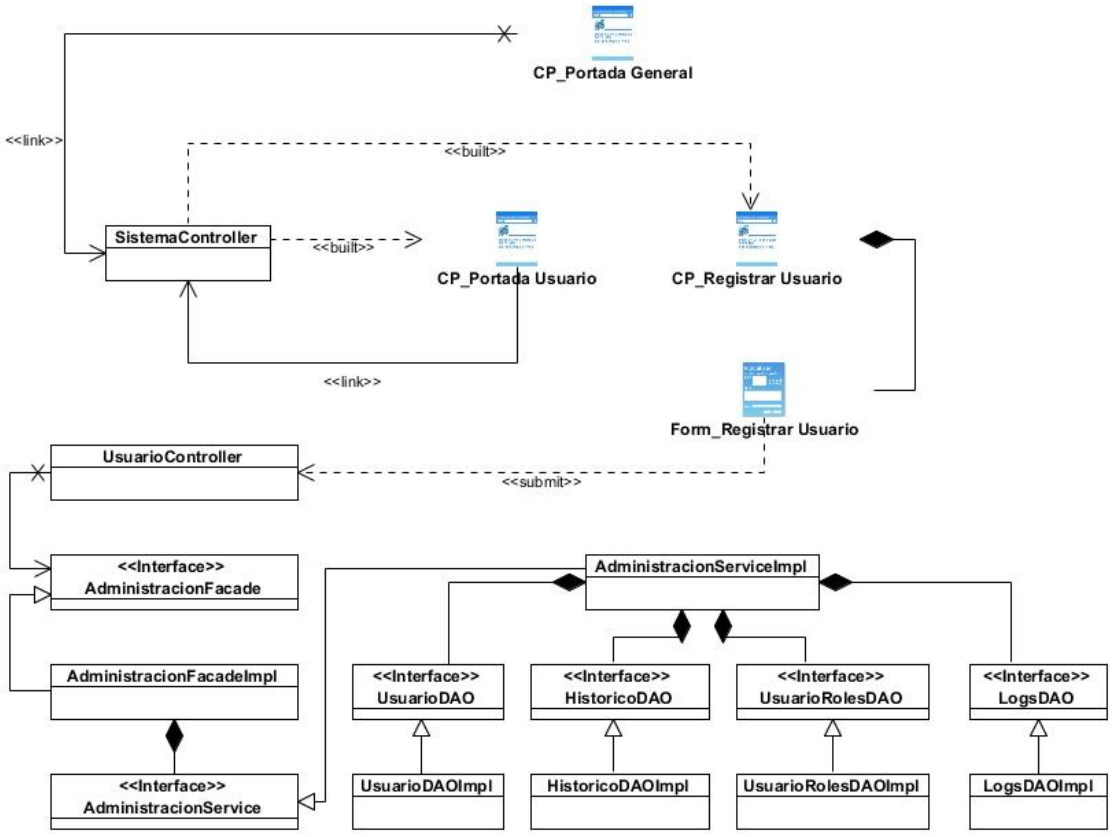


Ilustración 6: Diagrama de clases del diseño "Registrar Usuario"

2.3.7. Diagrama de secuencia

Los diagramas de interacción son utilizados para modelar los aspectos dinámicos de los sistemas, muestran la realización de un flujo o escenario concreto de un requisito en términos de interacción entre objetos del diseño. Se les atribuye una gran importancia debido a la compresión del sistema a la que puede llegarse a través de ellos. (21)

Debido a lo anterior se realizaron diagramas de secuencia (**Ver Anexo #17**), que son un ejemplo de diagramas de interacción. A continuación se presenta el diagrama de secuencia asociado a la realización del requisito funcional "Registrar Usuario" *(ver ilustración 7 y 8)*.

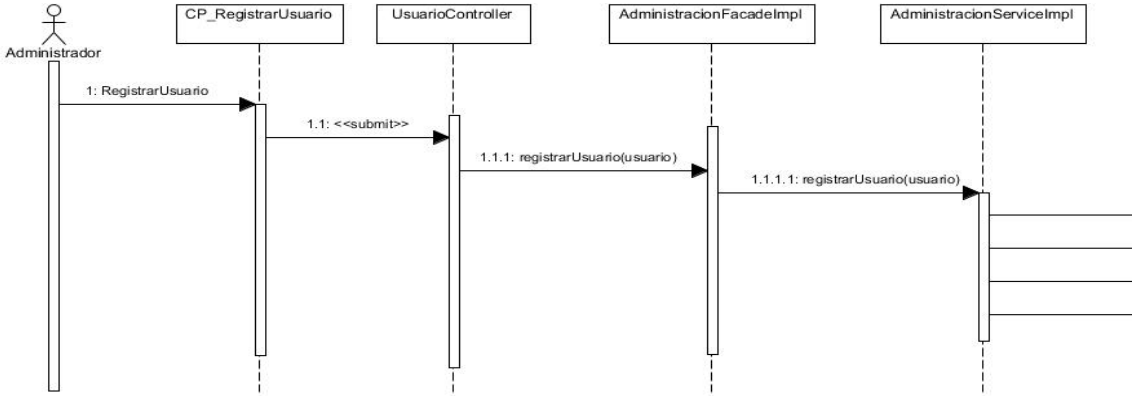


Ilustración 7: Diagrama de Secuencia "Registrar Usuario" (Parte 1)

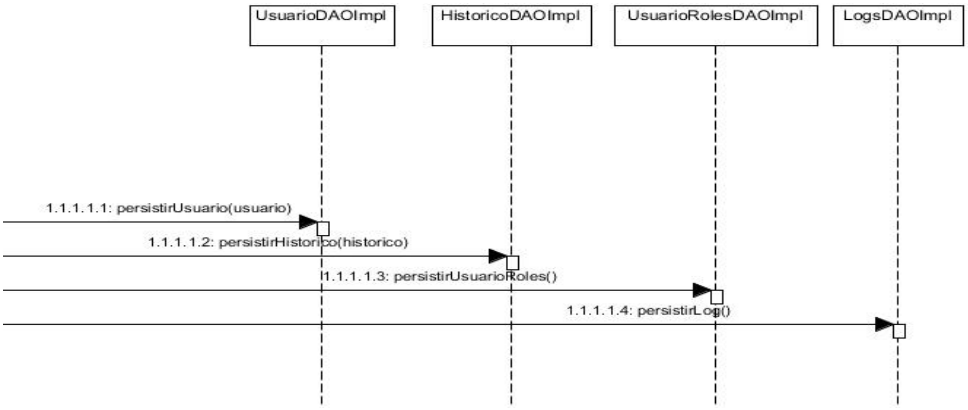


Ilustración 8: Diagrama de Secuencia "Registrar Usuario" (Parte 2)

2.3.8. Modelo de datos

El modelo de datos es un modelo abstracto que representa como se deben usar y representar los datos. Además, describe la estructura de la base de datos, las relaciones entre los datos y las restricciones que deben cumplirse entre ellos. (25) Para la solución propuesta, el modelo de datos se construyó teniendo en cuenta la menor cantidad de campos nulos. *(Ver Ilustración 9)*

El modelo de datos lo conforman 8 tablas: **usuario, usuarioRoles, roles, rolesFuncionalidad, funcionalidad, histórico, logs y configuración**. Además de 2 archivos: **securityConfig.properties y securityUrls.xml.**

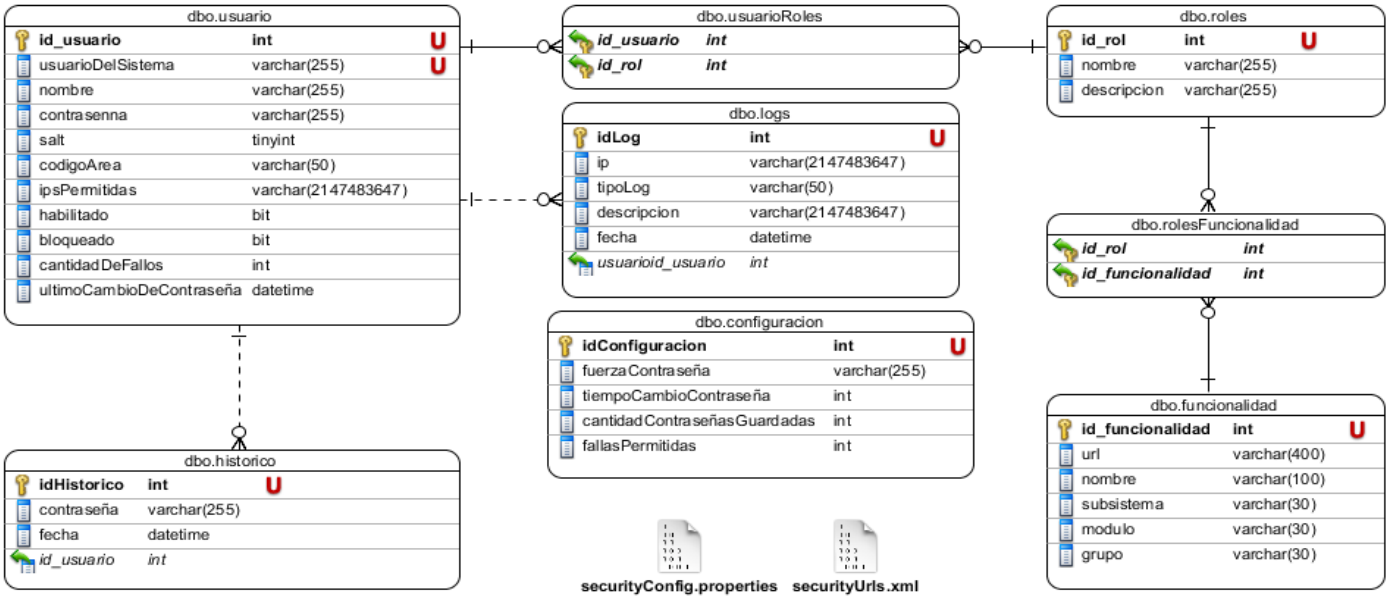


Ilustración 9: Diagrama del modelo de datos del componente de seguridad

A continuación se muestra una breve descripción de la función de las entidades y archivos. (*Ver tabla 3*)

Nombre	Descripción
usuario	Almacena la información de los usuarios.
roles	Almacena los roles del sistema.
funcionalidad	Almacena las funcionalidades que requieren seguridad del sistema.
historico	Almacena las contraseñas proporcionadas por el usuario en el sistema.
configuracion	Almacena configuraciones específicas de seguridad que fueron escritas en el

	fichero securityConfig.
logs	Almacena las acciones ejecutas en el sistema por los usuarios.
usuarioRoles	Almacena la relación entre los usuarios y los roles
rolesFuncionalidad	Almacena la relación entre los roles y las funcionalidades
securityConfig	Almacena configuraciones específicas de seguridad que serán leídas por el contexto de seguridad.
securityUrls	Contiene las funcionalidades que fueron entradas por el cliente y que serán posteriormente persistidas en la tabla funcionalidades.

Tabla 3: Descripción de función de las entidades y archivos

2.3.9. Modelo de dominio

Existen varias formas para describir o expresar el contexto de un sistema en una forma utilizable por los desarrolladores de software. Una de estas vías es mediante la construcción de un modelo de dominio.

Este modelo describe los conceptos importantes del contexto como objetos del dominio, y enlaza estos objetos unos con otros. La identificación y asignación de un nombre para estos objetos ayuda a completar un glosario de términos que permitirá comunicarse mejor a todas aquellas personas que trabajan en el sistema (22). El modelo de dominio (*ver Ilustración 10*) puede representarse mediante UML, con un Diagrama de Clases en los que se muestran:

- Objetos del negocio que representan elementos que se manipulan en el negocio.
- Objetos del mundo real y conceptos de los que el sistema debe hacer un seguimiento.
- Sucesos que ocurrirán o han ocurrido. (22)

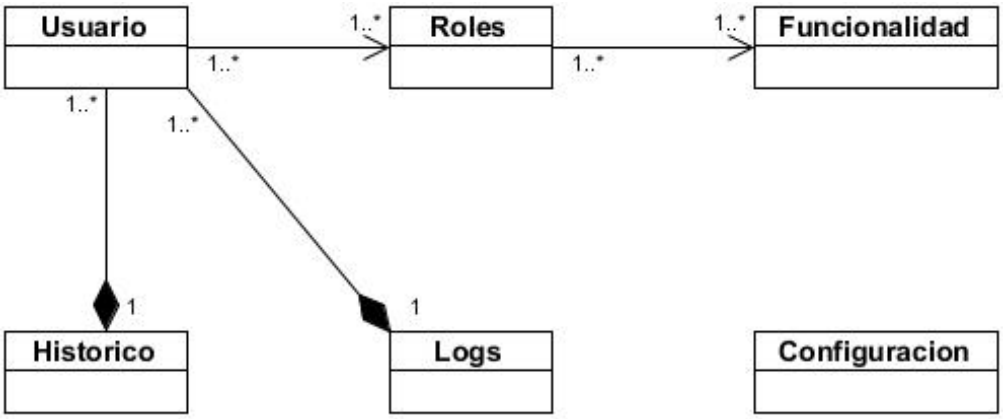


Ilustración 10: Modelo de dominio

2.3.10. Validación del diseño

El diseño está enfocado a convertir los requisitos del cliente en un modelo que al ser implementado, se obtenga el producto deseado. Generalmente constituye el punto de partida para el desarrollo de software una vez especificados los requerimientos del sistema. Evidentemente realizar una validación del mismo para verificar su calidad y flexibilidad, garantiza una buena base para la implementación. Con este objetivo se utilizan un conjunto de métricas de software orientadas a determinar, entre otros aspectos, qué características del modelo de diseño se pueden estimar para comprobar que el sistema será fácil de implementar en cuanto a organización. Entre ellas se seleccionaron las métricas **Tamaño Operacional de Clase (TOC)** y **Relación entre clases (RC)** para validar el componente de seguridad.

Tamaño Operacional de Clase (TOC)

Esta métrica se determina por el número total de operaciones que están encapsuladas dentro de la clase. Grandes valores de esta medida muestran que la clase puede tener demasiada responsabilidad, lo cual reducirá la reusabilidad de la misma y complicará su implementación. Por otro lado, en cuanto menor sea el valor medio para el tamaño más probable es que las clases tengan menos responsabilidad y complejidad y más nivel de reutilización.

Para la evaluación de las clases fueron utilizados umbrales para el tamaño general (*Ver tabla 4*) y la responsabilidad, la complejidad y la reutilización de las clases como atributos de calidad. (*Ver tabla 5*)

Nota: Promedio de operaciones (PO)

Atributo	Categoría	Criterio
Responsabilidad	Baja	<= PO
	Media	Entre PO y 2 * PO
	Alta	> 2 * PO
Complejidad implementación	Baja	<= PO
	Media	Entre PO y 2 * PO
	Alta	> 2 * PO
Reutilización	Baja	> 2 * PO

	Media	Entre PO y 2 * PO
	Alta	<= PO

Tabla 4: Atributos de calidad que afecta esta prueba.

Nombre	Descripción
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Tabla 5: Modo en que son afectados los atributos de calidad

Representación en % de la incidencia de los resultados obtenidos en cada atributo de calidad (ver Ilustración 11)

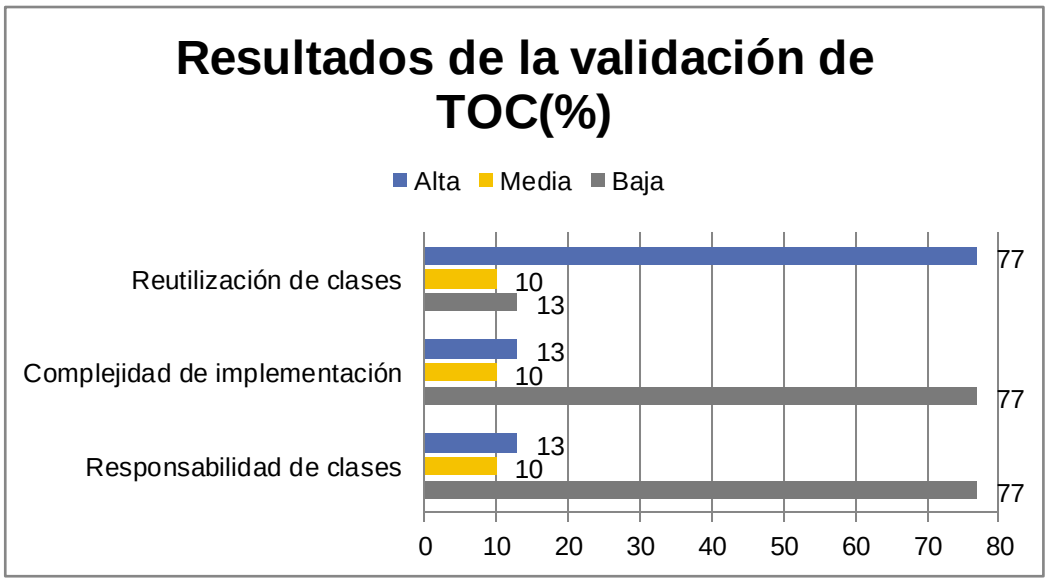


Ilustración 11: Resultados de la validación de "Tamaño Operacional de Clase"

Análisis de los resultados obtenidos en la evaluación de la métrica TOC

Durante la evaluación de la métrica TOC los resultados obtenidos demuestran que los atributos de calidad de las clases se encuentran en un nivel satisfactorio, evidenciándose un alto por ciento de posibilidad de reutilización de clases, dado por un 77% de baja responsabilidad de clases y de complejidad de implementación, con tan solo un 13% de baja reutilización de clases.

Relación entre clases (RC)

Esta métrica está dada por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad. (*Ver tabla 6)*

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 6: Atributos de calidad y el modo en que son afectados

Para la evaluación de las clases fueron utilizados umbrales para el acoplamiento, complejidad de mantenimiento, la reutilización y cantidad de pruebas. (Ver Tabla 7)

Atributo	Categoría	Criterio
Responsabilidad	Ninguna	0
	Baja	1
	Media	2
	Alta	> 2
Complejidad implementación	Baja	<= PO

	Media	Entre PO y 2 * PO
	Alta	> 2 * PO
Reutilización	Baja	> 2 * PO
	Media	Entre PO y 2 * PO
	Alta	<= PO
Reutilización	Baja	<= PO
	Media	Entre PO y 2 * PO
	Alta	> 2 * PO

Tabla 7: Atributos de calidad que afecta esta prueba

Representación en % de la incidencia de los resultados obtenidos en cada atributo de calidad (ver Ilustración 12)

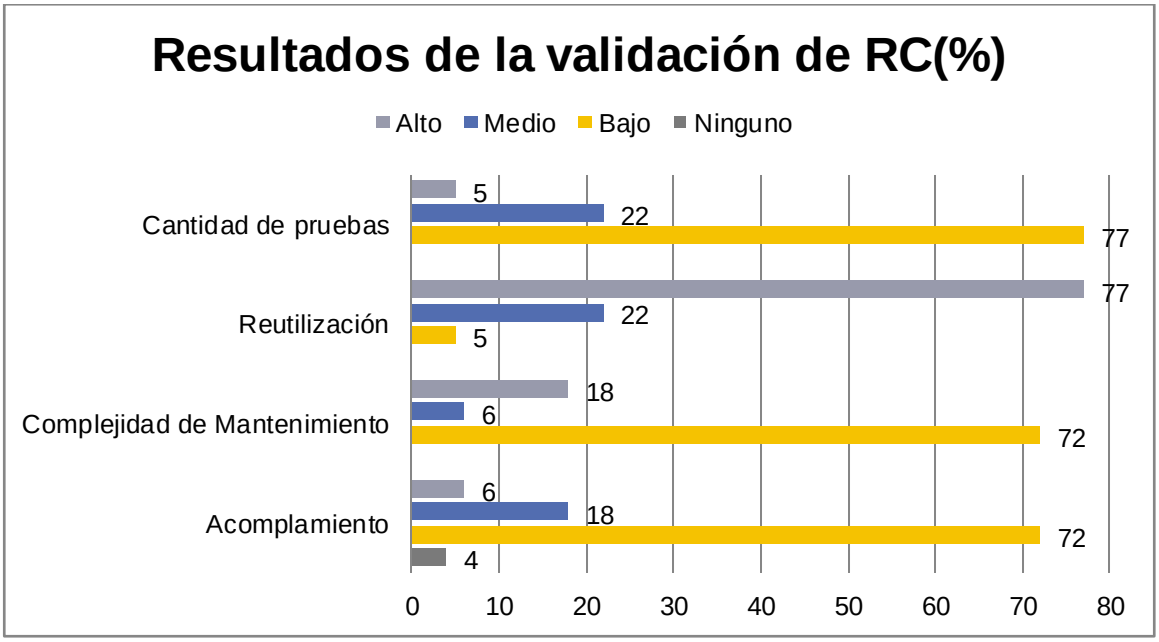


Ilustración 12: Resultados de la validación "Relaciones por Clases"

Análisis de los resultados obtenidos en la evaluación de la métrica RC

Luego de aplicarse la métrica de diseño RC y obtenidos los resultados de la evaluación del instrumento de medición de la métrica, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 93,75 % de las clases empleadas poseen menos de 3 dependencias de otras clases lo que lleva a evaluaciones positivas de los atributos de calidad involucrados (bajo acoplamiento con 72 %, baja complejidad de mantenimiento con un 72 %, baja cantidad de pruebas con 77 % y alta reutilización con un 77 %). Favoreciendo de esta manera la reutilización de las clases así como la modificación e implantación del diseño.

Matriz de inferencia de indicadores de calidad

La matriz de inferencia permite evaluar positiva o negativamente los resultados obtenidos de las relaciones atributo/métrica en una escala numérica, donde el valor 1 representa un resultado “positivo” y el valor 0 “negativo”. Si la métrica no influye en el atributo de calidad la relación es considerada como nula y es representada con un guion simple (-). Al promediar por cada atributo las relaciones no nulas, se obtiene un resultado general que al ubicarse en el rango de evaluación (valor <=0.4: “Mala”, valor >0.4 y valor<0.7: “Regular”, valor>=0.7: “Buena”) permite arribar a conclusiones sobre la calidad del diseño propuesto. Teniendo en cuenta que los resultados arrojados con la aplicación de las métricas fueron positivos para todos los atributos, la matriz de la inferencia queda elaborada de la siguiente manera. (*Ver tabla 8)*

Atributos\Métricas	TOC	RC	Promedio
Responsabilidad	1	(-)	1
Complejidad de implementación	1	(-)	1
Reutilización	1	1	1
Acoplamiento	(-)	1	1
Complejidad de mantenimiento	(-)	1	1
Cantidad de prueba	(-)	1	1

Tabla 8: Matriz de Inferencia de indicadores de calidad

El promedio general es 1 y teniendo en cuenta el rango de evaluación se concluye que el componente de seguridad para Quarxo Fase 2 presenta un diseño con buena calidad.

2.4. Conclusiones del capitulo

Se realizaron actividades correspondientes a las Disciplinas Modelamiento del Negocio, Requisitos y Análisis y diseño del Modelo de desarrollo del CEIGE, obteniéndose un modelo de diseño de buena calidad, lo cual facilitará el desarrollo de la disciplina Implementación.

Capítulo 3: Implementación y Validación del Componente de Seguridad

3. Introducción

El presente capítulo abarca las disciplinas de Implementación y Pruebas Internas correspondientes a la fase Desarrollo del Modelo de Desarrollo del CEIGE versión 1.1. Durante la implementación se expondrán los estándares de implementación utilizados, la descripción de las clases principales y otros aspectos relacionados con la utilización de Spring Security 3.1.0. Se realizan pruebas de caja negra y pruebas de caja blanca utilizando JUnit (pruebas unitarias de java) para evaluar la calidad del producto implementado. Finalmente, se validan las variables de investigación con el objetivo de comprobar si la solución es eficiente.

3.1. Implementación

La implementación constituye una de las disciplinas más importantes del modelo de desarrollo del centro, en ella se toma como punto de partida lo arrojado como resultado en el diseño y se implementa el sistema en términos de componentes como ficheros de código binario, código fuente, scripts, ejecutables, entre otros. Su importancia se debe a que se obtiene como consecuencia un sistema ejecutable, siendo uno de los principales objetivos en el desarrollo de software. (21)

3.1.1. Estándares de codificación

Los estándares de codificación se definen por el equipo de desarrollo para lograr estandarización en la programación del software. Estos se basan en la estructura y apariencia física de un programa con el fin de facilitar la lectura, comprensión, mantenimiento del código, reutilización a lo largo del proceso de desarrollo de un software y no en la lógica del programa. La generalización de aspectos tan simples como el trato de las mayúsculas, ayuda a eliminar conflictos de funcionalidades implementadas con nombres iguales y guían de forma clara el proceso de desarrollo. (28)

3.1.1.1. Convenciones de nomenclatura

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto con minúscula, en caso de ser una palabra compuesta se empleará la notación *PascalCasing*[7]. (29)

Ejemplo: UsuarioController.java, RolController.java

Los nombres de los métodos y los atributos de las clases, así como los nombres de ficheros de código *JavaScript* y sus funciones y variables internas comienzan con la primera letra en minúscula, en caso de que sea un nombre compuesto se empleará notación *CamelCasing*[8] (29)

Ejemplo: registrarRol.js, registrarUsuario.js

3.1.1.2. Nomenclatura según el tipo de clase

Las clases que se encuentran dentro del paquete *controller* se nombran adicionándoles el sufijo **Controller**, con el objetivo de identificarlas como controladores de *Spring.*

Ejemplo: RolController.java, UsuarioController.java

Las clases que se encuentran dentro del paquete *facade* se nombran adicionándoles como sufijo del nombre la palabra ***Facade***. Para el paquete *impl* se les nombra igual que la interfaz que implementan y se le agrega como sufijo la palabra *Impl*. (29)

Ejemplo: AdministracionFacade.java, AdministracionFacadeImpl.java

Las clases que se encuentran dentro del paquete *service* se nombran adicionándoles como sufijo del nombre la palabra ***Service***. Para el paquete *impl* se les nombra igual que la interfaz que implementan y se le agrega como sufijo la palabra *Impl*. (29)

Ejemplo: AdministracionService.java, AdministracionServiceImpl.java

[7] Plantea que los identificadores y nombres de variables, métodos y funciones que están compuestos por múltiples palabras juntas, iniciarán cada palabra con letra mayúscula.
[8] Plantea que los identificadores y nombres de variables, métodos y funciones que están compuestos por múltiples palabras juntas, iniciarán cada palabra con letra mayúscula excepto la primera palabra.

Las clases que se encuentran dentro del paquete *dao* se nombran adicionándoles como sufijo del nombre las siglas **Dao.** Para el paquete *impl* se les nombra igual que la interfaz que implementan y se le agrega como sufijo la palabra *Impl*. (29)

Ejemplo:

- ConfiguracionDao, ConfiguracionDaoImpl

3.1.2. Descripción de clases y funcionalidades

Basado en el diseño de las clases perteneciente al Componente de Onyx realizado en el capítulo anterior, a continuación se describirán los atributos y métodos de las que se consideran más importantes desde el punto de vista funcional asociadas a la *Administración* del *Componente*. (*Ver tablas 9, 10, 11*)

Nombre: UsuarioController	
Tipo de clase: *Controller*	
Atributo	**Descripción**
administracionFacade	Instancia de la clase fachada de la Administración.
passwordEncoder	Instancia de la clase PasswordEncoder de Spring Security
Métodos	**Descripción**
@RequestMapping(method = RequestMethod.POST, value = "/registrarUsuario") public registrarUsuario(HttpServletRequest request, HttpServletResponse response): void	Registra un Usuario.
@RequestMapping(method = RequestMethod.POST, value = "/asignarCredencial") public asignarCredencial(HttpServletRequest request, HttpServletResponse response) : void	Asigna una credencial (contraseña) al usuario que previamente fue buscado.
@RequestMapping(method = RequestMethod.POST,	Desbloquea un usuario que previamente fue

value = "/desbloquearUsario") public desbloquearUsario(HttpServletRequest request, HttpServletResponse response) : void	buscado.

Tabla 9 : Descripción de la clase "UsuarioController"

Nombre: AdministracionFacade	
Tipo de clase: *Interface*	
Atributo	**Descripción**
administracionService	Instancia de la clase service de la Administración.
Métodos	**Descripción**
public leerXML_security() : List<Funcionalidad>	Permite leer el XML, previamente realizado por el cliente según el estándar y conformar un objeto Java con la misma estructura.

Tabla 10 : Descripción de la clase "AdministracionFacade"

Nombre: RolController	
Tipo de clase: *Controller*	
Atributo	**Descripción**
administracionFacade	Instancia de la clase fachada de la Administración.
Métodos	**Descripción**
@RequestMapping(value = "/cargarArbolFuncionalidades", method = RequestMethod.GET)	Devuelve un JSON[9] con todas las funcionalidades que fueron leídas del XML y trasladadas a objetos java.

[9] JSON, acrónimo de JavaScript Object Notación, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

cargarArbolFuncionalidades(HttpServletRequest request,HttpServletResponse response) : String	
@RequestMapping(method = RequestMethod.POST, value = "/registrarRol") public registrarRol(HttpServletRequest request, HttpServletResponse response) : void	Registra un rol en el sistema

Tabla 11 : Descripción de la clase "RolController"

3.1.3. Aspectos fundamentales de la implementación

Spring Security es un framework que proporciona servicios de seguridad para aplicaciones J2EE, con un mecanismo de seguridad declarativo e independiente del entorno donde se despliega la aplicación. (30)

El núcleo de la seguridad del componente Onyx se centra en el desarrollo de las funcionalidades brindadas por el framework Spring Security, el mismo provee mecanismos de autenticación y autorización basados en el uso de una serie de filtros de seguridad que configurados debidamente elevan la seguridad de la aplicación.

3.1.3.1. Conformación del archivo XML de funcionalidades

Para cargar las funcionalidades que serán protegidas (interceptadas), por el framework de seguridad es necesaria la conformación de un XML (*securityUrls.xml*) con las funcionalidades que el cliente determine necesarias para la protección. (Ver Ilustración 13)

```
<global>
    <mapping subsystem="Onyx" module="Seguridad">
            <agrupation name="Usuario">
                <url name="Registrar Usuario">/mostrarRegistrarUsuario.htm</url>
                <url name="Actualizar Usuario">/actualizarUsuario.htm</url>
                <url name="Desbloquear Usuario">/desbloquearUsario.htm</url>
                <url name="Asignar Contrasenna">/asignarCredencial.htm</url>
            </agrupation>
            <agrupation name="Rol">
                <url name="Registrar Rol">/mostrarRegistrarRol.htm</url>
                <url name="Actualizar Rol">/actualizarRol.htm</url>
            </agrupation>
            <agrupation name="Buscador">
                <url name="Buscador General">/mostrarBuscador.htm</url>
            </agrupation>
            <agrupation name="Estadisticas">
                <url name="Funcionalidades por roles">/mostrarEstRolesXFuncionalidades.htm</url>
                <url name="Usuarios por roles">/mostrarEstCantidadUsuariosXRoles.htm</url>
            </agrupation>
            <agrupation name="Reportes">
                <url name="Generar reporte de logs">/mostrarReporteLogs.htm</url>
            </agrupation>
    </mapping>
</global>
```

Ilustración 13: Ejemplo del archivo XML de funcionalidades securityUrls.xml

A continuación se muestra una breve descripción de cada una de las etiquetas del XML de funcionalidades. (Ver tabla 12)

Nombre	Descripción
global	Agrupa los subsistemas.
mapping	Contiene los atributos que determinan el subsistema y modulo.
agrupation	Agrupa la(s) url(s) por un grupo determinado.
url	Contiene la(s) url(s) así como el nombre con él se le mostrara el cliente.

Tabla 12: Descripción de los tag del XML

3.1.3.2. Adaptación del archivo de configuración

Con el objetivo de hacer más dinámico el proceso de configuración del contexto del framework de seguridad se creó el archivo *securityConfig.properties* que almacena una serie de variables que influyen en el comportamiento y nivel de alcance de la seguridad, estas variables cuentan con valores por defecto, los cuales pueden ser modificados por el cliente a su consideración. (*Ver Ilustración 14*)

```
#########################################################################################################
###   Archivo de configuración de las variables utilizadas por la Administración de Seguridad Onyx   ###
#################                                                                   ####################


###     Expresión regular con que debe cumplir la contraseña entrada por el usuario
onyx.fuerzaContrasenna=[a-zA0-Z9]+

###     Tiempo en dias máximo que el usuario podra seguir usando la misma contraseña
onyx.tiempoCambioContrasenna=30

###     Cantidad de registros con las contraseñas del usuario que seran guardadas en el historial,
###     cuando el usuario desee cambiar su contraseña no podra ser con ninguna de estas y cuando lo haga
###     se eliminara el registro más antiguo y se adicionara normalmente la nueva
onyx.cantidadContrasennasGuardadas=3

###     Fallas permitidas durante el intento de logueo en la aplicación
onyx.fallasPermitidas=2
```

Ilustración 14: Archivo de configuración securityConfig.properties

A continuación se muestra una breve descripción de cada uno de las variables del archivo de configuración. (Ver tabla 13)

Variable	Descripción
fuerzaContrasenna	Expresión regular con que debe cumplir la contraseña entrada por el usuario.
tiempoCambioContrasenna	Tiempo en días máximo que el usuario podrá seguir usando la misma contraseña.
cantidadDeContrasennasGuardadas	Cantidad de registros con las contraseñas del usuario que serán guardadas en el historial, cuando el usuario desee cambiar su contraseña no podrá ser con ninguna de estas y cuando lo haga se eliminara el registro más antiguo y se adicionara normalmente la nueva.
fallasPermitidas	Fallas permitidas durante el intento de autenticación en la aplicación.

Tabla 13: Descripción del archivo de configuración

3.1.3.3. Funcionamiento de Spring Security

A continuación se describe el funcionamiento general de este framework desde el componente de seguridad así como de los principales elementos que lo componen.

3.1.3.3.1. Gestión de las peticiones encaminadas a entrar al sistema

Las peticiones realizadas a la página principal de la aplicación deben cruzar antes por una infraestructura de seguridad que despliega spring security, la misma se pone en marcha cuando el usuario realiza la petición, la cual es interceptada por el *AuthenticationManager*[10] que le solicita al usuario las credenciales, que no son más que el usuario y la contraseña, luego de que el mismo las provea, estas son validadas contra la base de datos, si son válidas, la petición es intervenida una vez más, en este caso por el *AccessDecisionManager*[11] que valida si el usuario tiene los permisos necesarios para acceder al recurso solicitado, en caso de tenerlos se le da acceso al sistema (*ver Ilustración 15*).

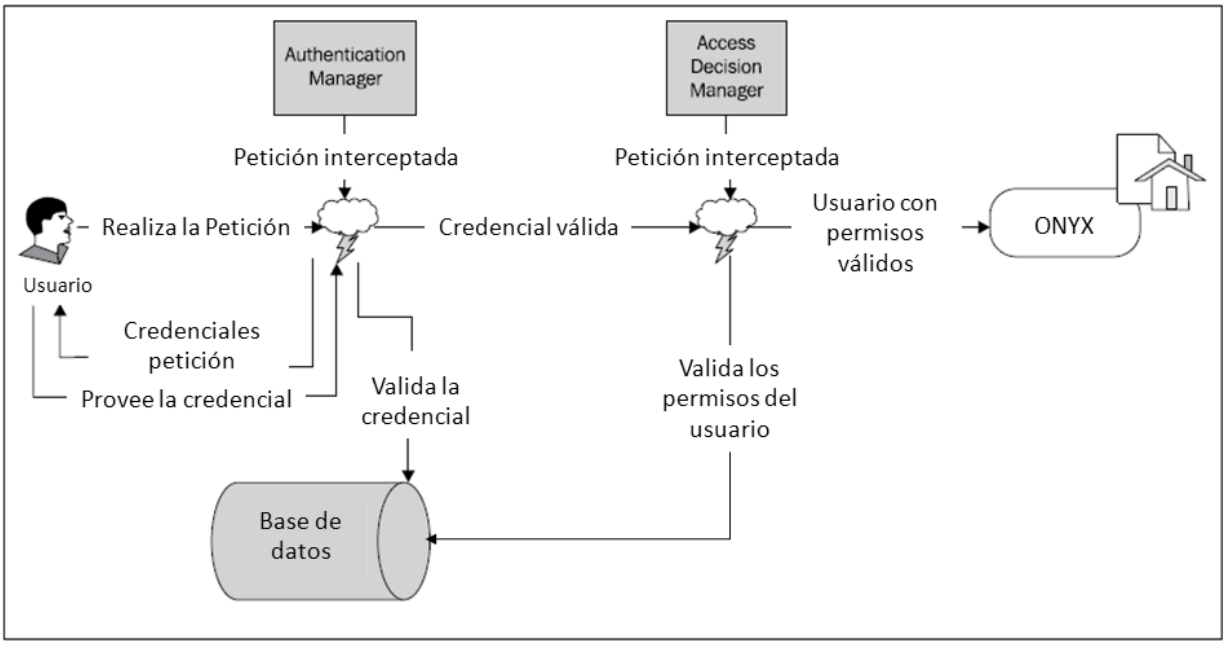


Ilustración 15: Gestión de las peticiones encaminadas a entrar al sistema

3.1.3.3.2. Filtros de seguridad

Parte del mecanismo de seguridad puede verse como un filtro que delega responsabilidades en un conjunto de filtros, que aseguran entre todos las funcionalidades del sistema. Spring Security define una

[10] *AuthenticationManager:* Es una clase de Spring Security especialmente preparada para administrar todas las posibles autenticaciones al sistema.
[11] *AccessDecisionManager:* Clase del framework de seguridad destinada a realizar una intercepción entre los permisos del usuario y los recursos solicitados, permitiendo identificar si tiene acceso o no al sistema.

cadena de filtros, cada uno de ellos es usado para interceptar las peticiones del usuario y realizar un pre o post-procesamiento de la información disponible o simplemente redireccionar a un lugar específico, el paso de un filtro a otro se realiza mediante la llamada al método *doFilter*. (*Ver Ilustración 16*)

Este framework es totalmente flexible permitiendo modificar o extender la cadena de filtros para personalizarlos de manera que se ajusten a las necesidades de la aplicación. (30)

El componente de seguridad Onyx hace uso de 14 filtros de seguridad, 4 de ellos fueron especialmente creados para cubrir las necesidades propias del sistema y asegurar partes especificas del mismo a las que el framework no era capaz de llegar, estas necesidades van a ser expuestas desde el punto de vista de su solución más adelante, además se adaptó el comportamiento de otros 3 filtros para asegurar la compatibilidad con el sistema y se dejó con el funcionamiento estándar a los restantes 7 filtros.

En la *ilustración 17* se muestra el orden de ejecución de estos filtros de seguridad y posteriormente una breve explicación del funcionamiento de los mismos.

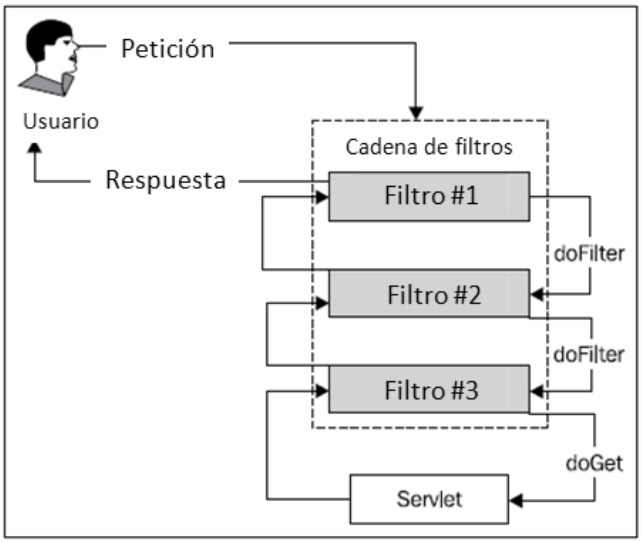


Ilustración 16: Gestión de las peticiones en el sistema

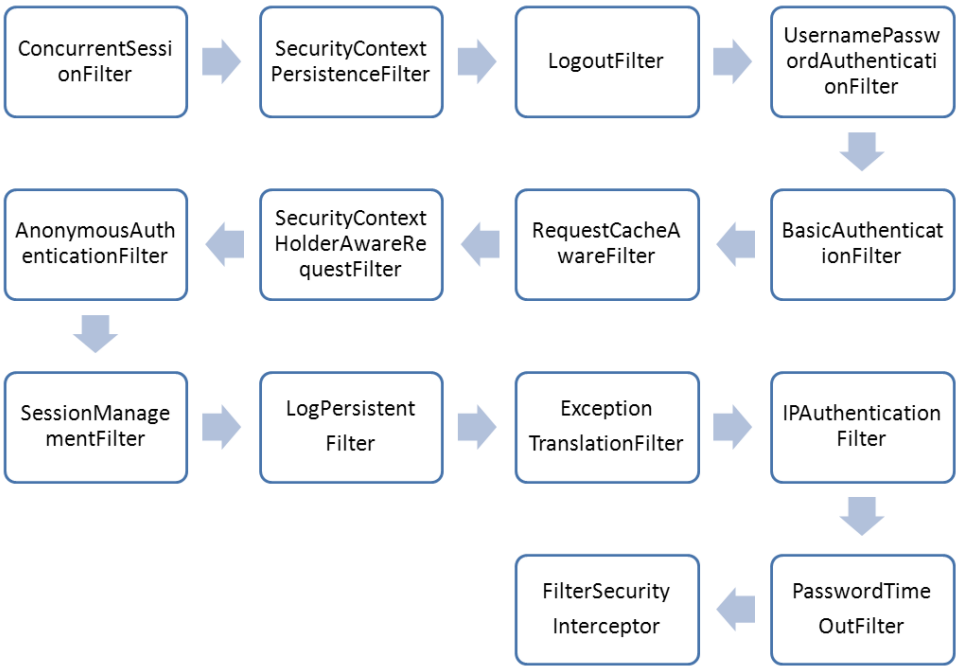


Ilustración 17: Filtros por los que pasa la petición antes de ser atendida

A continuación se brindara una breve explicación del funcionamiento de cada uno de los filtros:

Filtro No. 1. **ConcurrentSessionFilter:** Comprueba el tiempo que el usuario ha pasado inactivo en el sistema. Si ese tiempo llega a superar el establecido, se inactiva la sesión provocando la carga de la vista de autenticación, obligando al usuario a confirmar que aún está ahí.

Filtro No. 2. **SecurityContextPersistenceFilter:** Es responsable de la carga y almacenamiento del contexto de seguridad, este contexto representa el usuario asegurado y la sesión autenticada.

Filtro No. 3. **LogoutFilter:** Atiende la petición de deslogueo, la cual está habilitada para todos los usuarios autenticados en el sistema desde cualquier página. Su función es destruir la sesión del usuario y re-direccionarlo a la página de autenticación. (31)

Filtro No. 4. **UsernamePasswordAuthenticationFilter:** Espera por una petición a la URL virtual utilizada para la autenticación basada en formularios con nombre de usuario y contraseña (por defecto, */j_spring_security_check*), e intenta autenticar al usuario si hay una coincidencia. (31)

Filtro No. 5. **BasicAuthenticationFilter:** Vigila los encabezados de autenticación básica HTTP y los procesa. (31)

Filtro No. 6. **RequestCacheAwareFilter:** Se utiliza después de un inicio de sesión satisfactorio para reconstituir la petición original del usuario que fue interceptada por una solicitud de autenticación. (30)

Filtro No. 7. **SecurityContextHolderAwareRequestFilter:** Envuelve el HttpServletRequest con una subclase que extiende de HttpServletRequestWrapper. Esto proporciona un contexto adicional para solicitar procesadores más abajo en la cadena de filtros. (31)

Filtro No. 8. **AnonymousAuthenticationFilter:** Si el usuario no ha sido autenticado para el tiempo en que este filtro es llamado, un token de autenticación se asocia con la solicitud que indica que el usuario es anónimo. (31)

Filtro No. 9. **SessionManagementFilter:** Gestiona la administración de la sesión basado en los usuarios autenticados, ayudando a asegurarse de que todas las sesiones asociadas con un solo usuarios están siendo analizadas. (31)

Filtro No. 10. **ExceptionTranslationFilter:** Este filtro se encarga del enrutamiento por defecto y la delegación de las excepciones que se producen durante el procesamiento de una petición. (31)

Los últimos 4 filtros que se muestran a continuación fueron creados para cubrir necesidades específicas del cliente, así como elevar la seguridad de la aplicación:

Filtro No. 11. **LogPersistentFilter:** Se encarga de registrar en la base de datos todo lo que está ocurriendo en el sistema, desde las peticiones más simples, hasta las más complejas, llevando un registro del usuario, IP, hora y fecha en la que realizó la petición.

Filtro No. 12. **IPAuthenticationFilter:** Autoriza o no el acceso de un usuario al sistema según su dirección IP del mismo.

Filtro No. 13. **PasswordTimeOutFilter:** Verifica que al usuario que está autenticándose no se le haya caducado la contraseña, en caso de que si haya ocurrido se le solicita que la cambie, y al mismo tiempo chequea que no repita ninguna de las contraseñas que están almacenadas en el histórico del usuario, el número de contraseñas almacenadas en el histórico es totalmente configurable y posee el valor 3 por defecto.

Filtro No. 14. **FilterSecurityInterceptor:** Al ser el último filtro, es el que decide si el usuario entra o no al sistema, el mismo se basa en la verificación del rol o roles del usuario para con la funcionalidad a la que desea acceder, en caso de que algún rol tenga permisos de acceso a dicha funcionalidad, se le da acceso, en caso contrario se le deniega el mismo.

3.2. Pruebas

El principal objetivo de esta disciplina es de evaluar la calidad del producto que se desarrolló. Se realiza a través de diferentes fases mediante la aplicación de pruebas concretas para validar que las suposiciones hechas en el diseño y que los requerimientos se estén cumpliendo satisfactoriamente. (29)

Esto significa que se verifica el funcionamiento del producto como se diseñó y que los requerimientos han sido cumplidos satisfactoriamente.

Se aplicarán los métodos de pruebas adaptados a este nivel, como son los métodos de prueba: Caja Blanca (usando la técnica “camino básico”), para comprobar los caminos lógicos del software y Caja Negra, para probar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. (32)

3.2.1. Pruebas unitarias

El objetivo de las pruebas unitarias es el aislamiento de partes del código y la demostración de que estas partes no contienen errores. Las pruebas unitarias son una forma de probar el correcto funcionamiento de un módulo de código, esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. (32)

Las pruebas unitarias realizadas a los servicios del sistema sirven para validar que las salidas son correctas y aseguran al desarrollador que la solución no presenta errores en la lógica de programación y que las respuestas son las correctas ante una entrada de datos determinada. (32)

3.2.1.1. Pruebas de Caja Blanca

Para las pruebas de Caja Blanca se utilizó el framework *JUnit*, el cual brinda un conjunto de bibliotecas que se integran fácilmente al IDE de desarrollo seleccionado: *Eclipse*. *JUnit* permite la realización de las pruebas a los métodos de las clases implementadas. Para hacer uso de este framework se definieron los siguientes pasos:

- Crear un caso de prueba por cada clase implementada.
- Definir los métodos a probar dentro de cada caso de prueba.
- Realizar pruebas a los métodos. (32)

A continuación se muestran los resultados de aplicar las pruebas a la clase más significativa que presenta la administración, *UsuarioController,* el resultado de la prueba realizada a todos los métodos de esta clase fue satisfactorio (*ver ilustración 18*).

El flujo de trabajo completo para la realización de la prueba puede ser consultado en el **anexo 22.**

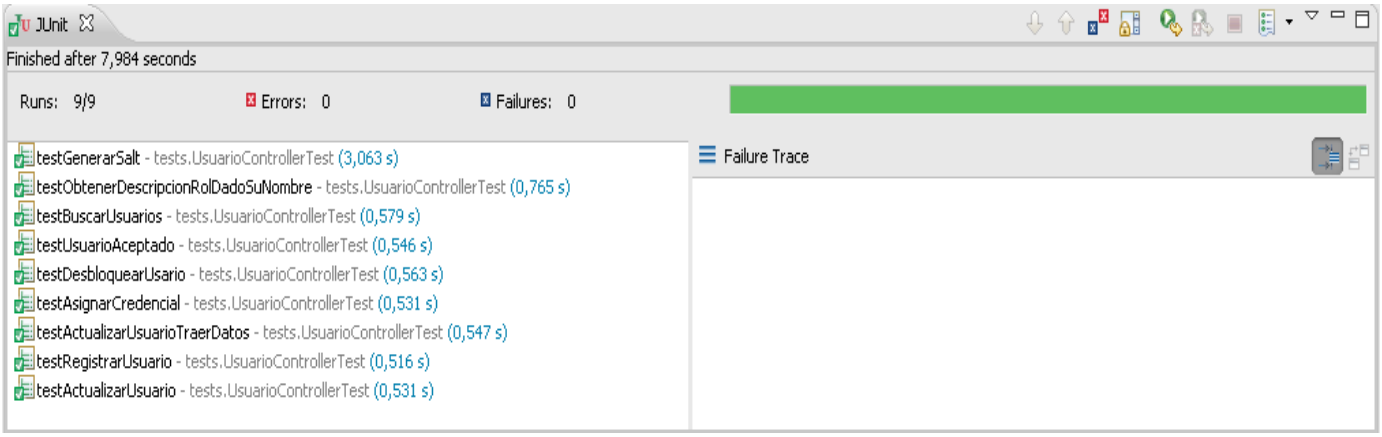


Ilustración 18: Resultado de la prueba realizada

3.2.2. Pruebas de Caja Negra

Las pruebas de Caja Negra o pruebas Funcionales se realizan sobre la interfaz del software, entendiendo por interfaz las entradas y salidas del mismo. No es necesario conocer su lógica de funcionamiento, únicamente la funcionalidad que debe realizar. También conocidas como Pruebas de Comportamiento, estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente se ve como una "Caja Negra" cuyo comportamiento solo puede ser determinado mediante el estudio de las entradas y salidas obtenidas a partir de ellas. (33)

No obstante, como el estudio de todas las posibles entradas y salidas de un programa sería impracticable, se seleccionó un conjunto de ellas sobre las que se realizan las pruebas. Para seleccionar el conjunto de entradas y salidas se consideró que en todo programa existe un conjunto de entradas que causan un comportamiento erróneo en el sistema, y como consecuencia producen una o varias salidas que revelan la presencia de defectos o errores.

Las técnicas de prueba no se hallan de forma aislada, sino como un conjunto integrado de acciones que combinadas permitirán verificar y evaluar la calidad de software. Entre las técnicas para desarrollar la prueba de Caja Negra se encuentran:

- ✓ *Técnica de la Partición de Equivalencia:* divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- ✓ *Técnica del Análisis de Valores Límites:* prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- ✓ *Técnica de Grafos de Causa-Efecto:* permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones. (33)

De las técnicas mencionadas se aplicará la de **Partición de Equivalencia** ya que es una de las más efectivas. Permite definir casos de prueba que declaren clases de errores, reduciendo el número de casos de prueba a desarrollar para demostrar que las funciones del software son operativas; las entradas se aceptan de forma adecuada y se producen salidas correctas.

Para verificar que la aplicación se comporta según los requerimientos establecidos por el cliente, se diseñan casos de pruebas usando el método de Caja Negra. Para ello se realizan casos de pruebas para el requisito "*Registrar Usuario*", los cuales se pueden consultar en el documento de tesis en formato digital (**Ver Anexo 1**).

Las pruebas realizadas fueron satisfactorias desde el punto de vista interno y funcional, abarcando requerimientos, funciones y lógica interna. Se aplicaron los métodos de Caja Negra y Caja Blanca para validar tanto la interfaz como el correcto funcionamiento interno del software.

3.3. Validación de las variables de la investigación

La investigación desarrollada plantea como idea a defender que: "Con la implementación del nuevo componente de seguridad del sistema Quarxo se asegura mayor rendimiento, independencia en las configuraciones y una adecuada auditoría en la gestión de la seguridad". A continuación se evalúan las variables rendimiento, independencia y auditoría con el uso del nuevo componente de seguridad para Quarxo en el BNC. Para esta evaluación se tomaran en cuenta una serie de indicadores con el objetivo de obtener una visión global del comportamiento de estas variables.

El objetivo fundamental de mejorar el rendimiento es minimizar el retraso observado por el usuario entre el momento en que hace clic en el enlace y en el que la página finalmente aparece, así como agilizar el proceso que se realiza en la dicha página.

A continuación se realizará una comparación entre el actual sistema de seguridad del banco y el nuevo componente de seguridad. (Ver tablas 14, 15, 16, 17, 18, 19)

Tiempo de respuesta			
No	**Actividades**	**Componente en Quarxo**	**Componente Onyx**
1	Registrar rol, Actualizar rol	Implica la carga del árbol de funcionalidades de la base de datos del banco. Cada vez que se despliega un nodo del árbol se realiza una carga lazy de las funcionalidades requeridas.	Carga las funcionalidades del XML que previamente fue conformado por el cliente.
	OBSERVACIONES	El uso excesivo de peticiones en la página dificulta la renderización de la misma y por lo tanto la navegabilidad en ella.	

Tabla 14: Comparación entre el componente de Quarxo y Onyx con respecto a "tiempo de respuesta"

Escenario 1: Se realiza una prueba mediante la carga del árbol de funcionalidades en los dos sistemas, debido a que Onyx carga todo el árbol de funcionalidades de un XML, y el componente de seguridad de Quarxo carga nodo a nodo según petición a la base de datos, se tomaron 16 funcionalidades para ejecutar la prueba.

Para medir el tiempo de respuesta de cada petición se utilizó la herramienta ***WebInject*** (**Ver Anexo 2**).

Tiempos de respuesta		
Indicadores	**Componente de Quarxo**	**Componente Onyx**
Mayor tiempo de respuesta	0.543	0.021
Menor tiempo de respuesta	0.010	0,001
Promedio de tiempos de respuesta	0.046	0.021
Tiempo total (segundos)	**2.918**	**0.021**

Tabla 15: Tiempos de respuesta del componente de Quarxo y Onyx

Agregar que, el tiempo de renderización se ve optimizado en el componente de seguridad Onyx (**Ver Anexo 18**) debido al reemplazo del framework de decoración visual Dojo por Bootstrap, este último está basado completamente en HTML5 y CSS3, por lo que gran parte de la estrategia para renderizar se encuentra en el propio navegador.

Las restantes áreas serán analizadas a continuación:

Navegabilidad			
No	**Actividades**	**Componente en Quarxo**	**Componente Onyx**
1	Buscar usuarios, Buscar roles	Los buscadores están ubicados en vistas diferentes.	Todos los buscadores están en una misma vista (ventana emergente), incluido un buscador para logs.
	OBSERVACIONES	El uso de diferentes vistas para mostrar contenidos comunes o que presentan algún grado de relación, afecta la navegabilidad y por lo tanto la usabilidad del sistema.	

Tabla 16: Comparación entre el componente de Quarxo y Onyx con respecto a "navegabilidad"

Usabilidad			
No	**Actividades**	**Componente en Quarxo**	**Componente Onyx**
1	Selección de funcionalidades para un rol en árbol.	Es necesario desplegar cada nodo del árbol hasta encontrar la funcionalidad requerida.	Cuenta con un campo para la búsqueda de funcionalidades en el árbol, permitiendo además buscar agrupaciones completas.
	OBSERVACIONES	El proceso de búsqueda nodo a nodo se hace bastante complejo cuando es necesario asignar muchas funcionalidades a un rol, consumiendo mucho tiempo mientras se recorre el árbol.	

Tabla 17: Comparación entre el componente de Quarxo y Onyx con respecto a "usabilidad"

Independencia			
No	**Actividades**	**Componente en Quarxo**	**Componente Onyx**
1	Inicialización o arranque del componente	Dependiente de varios módulos del sistema Quarxo	No depende de ningún módulo del sistema Quarxo
	OBSERVACIONES	La dependencia de módulos que no contribuyen directamente al negocio, impidiendo que el componente sea reutilizable.	
2	Administración del componente	Prevalece dentro del sistema bancario Quarxo	Es totalmente independiente al sistema bancario Quarxo
	OBSERVACIONES	Los administradores tienen acceso a todos los módulos del sistema Quarxo sin ningún propósito.	

Tabla 18: Comparación entre el componente de Quarxo y Onyx con respecto a "independencia"

Control de contenido			
No	**Actividades**	**Componente en Quarxo**	**Componente Onyx**
1	Consulta de estadísticas	No cuenta con ningún visualizador de estadísticas	Cuenta con un visualizador de estadísticas que permite la impresión y exportación de las mismas.
	OBSERVACIONES	El control de estadísticas permite obtener una visión global del comportamiento y tendencia de los datos.	
2	Registro de logs	No cuenta con registro de logs	Permite el registro y la búsqueda de logs
	OBSERVACIONES	La falta de registro de logs atenta contra el control de las operaciones.	

Tabla 19: Comparación entre el componente de Quarxo y Onyx con respecto a "control de contenido"

3.4. Conclusiones del capítulo

Este capítulo centra sus elementos fundamentales en la implementación y validación del componente de seguridad Onyx, teniendo como premisa la calidad de la solución propuesta.

Fueron abordados temas tales como los estándares de codificación empleados. Se realizó una breve descripción de algunas clases esenciales que intervienen en el desarrollo de la solución describiéndose sus funcionalidades, atributos y descripción de los mismos. La solución de seguridad fue probada mediante pruebas de Caja Blanca, obteniéndose las respuestas correctas ante una entrada de datos determinada y que la solución no presenta errores en la lógica de programación; Caja Negra, verificándose que la aplicación se comporta según los requerimientos establecidos por el cliente.

Se evaluaron las variables consideradas en la investigación, demostrando que la solución desarrollada es apta para reemplazar el sistema de seguridad de Quarxo, permitiendo disminuir los tiempos de respuesta y al mismo tiempo ganar en usabilidad, navegabilidad y más funcionalidades.

Conclusiones

La investigación desarrollada y los resultados obtenidos permiten a los autores plantear las siguientes conclusiones:

- ✓ En el estudio de los mecanismos para la aplicar seguridad en aplicaciones bancarias, se descartó la utilización de las aplicaciones informáticas no ajustadas al negocio, por no corresponderse con la solución que se necesita. Se utilizó el framework Spring Security como herramienta especializada para la gestión de la seguridad.
- ✓ El proceso de desarrollo de software fue guiado por el *Modelo de Desarrollo de Software para el CEIGE*, cumpliendo con los elementos que dicho modelo propone y logrando una efectiva ingeniería de software.
- ✓ Las tecnologías seleccionadas para el desarrollo de la solución están acordes a los requerimientos del cliente, a las políticas del centro y a los estándares internacionales.
- ✓ Se obtuvo el nuevo componente de seguridad para Quarxo, probado mediante pruebas de caja Blanca y Caja Negra. Fue desarrollado con tecnologías libres, y provisto de nuevos elementos que tributan a una mayor eficiencia en la gestión del mismo.

Recomendaciones

A partir del estudio realizado en la presente investigación, teniendo en cuenta las experiencias obtenidas a lo largo de su desarrollo, los autores proponen las siguientes recomendaciones:

- ✓ Desarrollar una funcionalidad de notificaciones, que sea capaz de avisarle a los administradores del sistema cuando un usuario se ha bloqueado y cuando se ha terminado de generar un reporte.
- ✓ Desarrollar un mecanismo que permita actualizar en tiempo real las estadísticas mostradas en el sistema.
- ✓ Desarrollar un mecanismo que permita seleccionar visualmente los filtros que van a ser aplicados al sistema del cliente y aplicarles la configuración mínima necesaria para su correcto funcionamiento.
- ✓ Adicionar una funcionalidad que permita la visualización de los datos de la sesión de los usuarios que están autenticados en el sistema cliente.

Bibliografía consultada

Erich Gamma, R. H. (2007). *Design Patterns. Addison-Wesley.*

Larman, C. (2008). *UML y Patrones. Prentice Hall.*

Lowagie, B. (2010). *IText in Action.*

McGraw-Hill Interamericana, R. S. (2009). *INGENIERÍA DEL SOFTWARE. Un enfoque práctico.*

Mularien, P. (2010). *Spring Security 3.*

Murphy, P. T. (n.d.). *Spring Persistance with Hibernate.*

O'Reilly Media Inc, M. A. (2008). *Dojo: The Definitive Guide.*

Peter Tahchiev, F. L. (2010). *JUnit in Action. Manning Publications Co.*

Rod Johnson, J. H. (n.d.). *Spring, Reference Documentation.*

Taylor, B. A. (n.d.). *Spring Security, Reference Documentation.*

Referencias Bibliográficas

1. **Definicion.de.** Definicion.de. [Online] [Cited: Enero 2013, 22.] http://definicion.de/seguridad-informatica/.

2. —. definicion.de. [Online] [Cited: Noviembre 15, 2012.] http://definicion.de/seguridad/.

3. **INSTITUTE, I. G. COBIT.** IGC. [Online] [Cited: Enero 18, 2013.] http://www.itgi.org. ISBN 1-933284-37-4.

4. **M. E. Kabay, PhD [,CISSP-ISSMP].** *Identification, Authentication and Authorization on the World Wide Web.* s.l. : International Computer Security Association, 2009.

5. **Ecured.** ECURED. [Online] [Cited: Diciembre 12, 2012.] http://www.ecured.cu/index.php/Control_de_acceso..

6. **Jorge L. Martín Correa, Alejandro.** *Módulo de seguridad para el sistema informático del Banco Nacional de Cuba.* La Habana : s.n., 2009.

7. **desarrolloweb.** Desarrolloweb.com. [Online] [Cited: Enero 19, 2013.] http://www.desarrolloweb.com/articulos/996.php.

8. **Netinfo.** Netinfo. [Online] [Cited: 01 15, 2013.] http://www.netinfo.com.cy/default.asp?id=275.

9. **websense.** Websense. [Online] [Cited: Enero 10, 2013.] http://www.websense.com/content/Home.aspx.

10. **González, Reinaldo Pelaez.** *Drivers de autenticación para el sistema de seguridad de ACAXIA.* La Habana : s.n., 2011.

11. **Baryolo, MSc. Oiner Gómez.** *SISTEMA DE GESTIÓN INTEGRAL DE SEGURIDAD ACAXIA.* La Habana : UCI, 2013.

12. **ENTIDADES, CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DE.** *MODELO DE DESARROLLO DE SOFTWARE.* La Habana : Subdirección de Producción, 2012.

13. **Reyes, Daylen de la Caridad Barbán and Rodríguez, Roberto Rodríguez.** *Diseño e implementación del subsistema Vencimiento del sistema Quarxo para el Banco Nacional de Cuba.* La Habana : Universidad de las Ciencias Informáticas, 2011.

14. El mundo Informático. [Online] http://jorgesaavedra.wordpress.com/2007/05/08/patrones-grasp-patrones-de-software-para-la-asignacion-general-de-responsabilidadparte-ii/.

15. **Torres, Hector Peña.** *Desarrollo del subsistema Reportes de Quarxo.* La Habana : s.n., 2012.

16. **Alain L. Piñero Añon, Ing. Yadira Calimano Meneses., Ing. Adolfo Miguel Iglesias Chaviano.** *Diseño e implementación del módulo Captación de Mensajes en el sistema Quarxo.* La Habana : UCI, 2011.

17. Ecured. *Ecured.* [Online] [Cited: 03 20, 2013.] http://www.ecured.cu/index.php/Visual_Paradigm.

18. **Jorge Antonio Fariñas Acosta, Ing. Yulier Matías León.** *Diseño e Implementación del módulo Interfaces del plugins Génesis.* La Habana, Cuba : UCI, 2011.

19. **html5rocks.** HTML5rocks. [Online] [Cited: 01 17, 2013.] http://www.html5rocks.com/en/tutorials/developertools/sourcemaps/.

20. **dev.w3.** Dev.w3. [Online] [Cited: 02 10, 2013.] http://dev.w3.org/html5/spec/Overview.html#html-vs-xhtml.

21. **W3schools.** w3schools. [Online] [Cited: 02 10, 2013.] http://www.w3schools.com/html5/html5_reference.asp.

22. **w3.** w3. *Cascading Style Sheets (CSS) Snapshot 2010.* [Online] [Cited: 04 03, 2013.] http://www.w3.org/TR/css-2010/#css.

23. **imaginanet.** Imaginanet. [Online] [Cited: 02 14, 2013.] http://www.imaginanet.com/blog/html5-frameworks-html5-boilerplate-y-twitter-bootstrap.html.

24. **Sommerville, I.** *Ingeniería del software .* s.l. : Pearson Educación., 2005.

25. IEEE. IEEE Standard Glossary of Software Engineering Terminology. [Online] http://standards.ieee.org/findstds/standard/610.12-1990.html.

26. **Lara, Rafael Bello.** *Arquitectura de Software v0.2.*

27. **Juntadeandalucia.** juntadeandalucia. [Online] 03 04, 2013. http://www.juntadeandalucia.es/servicios/madeja/contenido/subsistemas/arquitectura/capa-acceso-datos.

28. **Rumbaugh, J., I. Jacobson, and G. Booch.** *UML. El lenguaje de modelado unificado. Manual de referencia. 2000, Addison Wesley.*

29. **Jacobson, I., G. Booch, J. Rumbaugh.** *EL PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE.* s.l. : PERSON EDUCACIÓN.S.A., 2008.

30. **Pressman, R.** *Ingeniería del Software: Un Enfoque Práctico.* 2008.

31. **Elmasri, R.** *Fundamentos de sistemas de bases de datos. 2009: Addison-Wesley.* 2009.

32. **Cruz, Reiniel Enrique Santa.** *DESARROLLO DEL SUBSISTEMA MENSAJERÍA SLBTR VERSIÓN 1.1 DE QUARXO.* La Habana : UCI.

33. **Chaviano, Adolfo Miguel Iglesia.** *Estándares de codificación Proyecto Sistema automatizado de la gestión bancaria.*

34. **Vargas, Cesar.** FLUX it. [Online] 06 19, 2011. [Cited: 04 06, 2013.] http://wiki.fluxit.com.ar/display/PUBLIC/Spring+Security.

35. **Mularien, Peter.** *Spring Security 3.* s.l. : Gagandeep Singh, 2010.

36. **Myers, G., C. de Filecich, y A. Filevich.** *El arte de probar el software.* 2010.

37. **Marañón, Gonzalo Álvarez.** [Online] [Cited: Noviembre 2012, 21.] http://www.iec.csic.es/CRIPTONOMICON/autenticacion/control.html.

38. **Sergi Quiñonero.** Sergi Quiñonero Blog Page. [Online] [Cited: Enero 20, 2013.] http://www.sergiquinonero.net/diferentes-tipos-de-ataques-en-paginas-y-aplicaciones-web.html.

39. **CS.** ComeSpan. [Online] [Cited: Enero 15, 2013.] http://www.comespam.com/2011/09/12/estrategia-de-seguridad-web-monitorizando-con-el-barracuda-web-filter/.

40. **javaencastellano.** Javaencastellano. [Online] [Cited: 04 01, 2013.] http://javaencastellano-hibernate.blogspot.com/2009/09/criteria-query-en-hibernate.html.

41. **jstree.** Jstree. [Online] [Cited: 04 02, 2013.] http://www.jstree.com/.

42. **Jacobson, I., G. Booch, J. Rumbaugh.** Scientificcommons. [Online] [Cited: 04 10, 2013.] http://en.scientificcommons.org/6960494.

43. **Lara, Rafael Bello.** *0120_ARQUITECTURA DE SOFTWARE VISTA ENTORNO DE DESARROLLO TECNOLÓGICO.* 2009.

44. **Mularien, Peter.** *Spring Security 3.* s.l. : Packt, 2013.

45. Mynitor. [Online] [Cited: 5 2, 2013.] http:mynitor.com.

46. masterdisseny. [Online] [Cited: 5 20, 2013.] http://www.masterdisseny.com/master-net/articulos/art0080.php3.

47. *Architecture Working Group, I., IEEE Recommended Practice for Architectural Description of Software-Intensive Systems.* s.l. : IEEE Standards Association, 2010. IEEE Standard IEEE Std 1471-2000.

48. **Barnes, D.J. y M. Kölling.** *Programación orientada a objetos con Java.MANEJO EFICIENTE DEL TIEMPO.* 2008.

49. **Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides.** *Design Patterns. Addison-Wesley.* 2007.

50. **O'Reilly Media Inc, Matthew A. Russell.** *Dojo: The Definitive Guide.* 2008.

51. **McGraw-Hill Interamericana, Roger S. Pressman.** *INGENIERÍA DEL SOFTWARE. Un enfoque práctico.* 2009.

52. **Lowagie, Bruno.** *IText in Action.* 2010.

53. **Peter Tahchiev, Felipe Leme, Vincent Massol y Gary Gregory.** *JUnit in Action. Manning Publications Co.* 2010.