Universidad de las Ciencias Informáticas

Facultad 2

Requerimientos de seguridad de autenticación y generación de notificaciones en el Sistema Informático XABAL Arkheia 4.0.

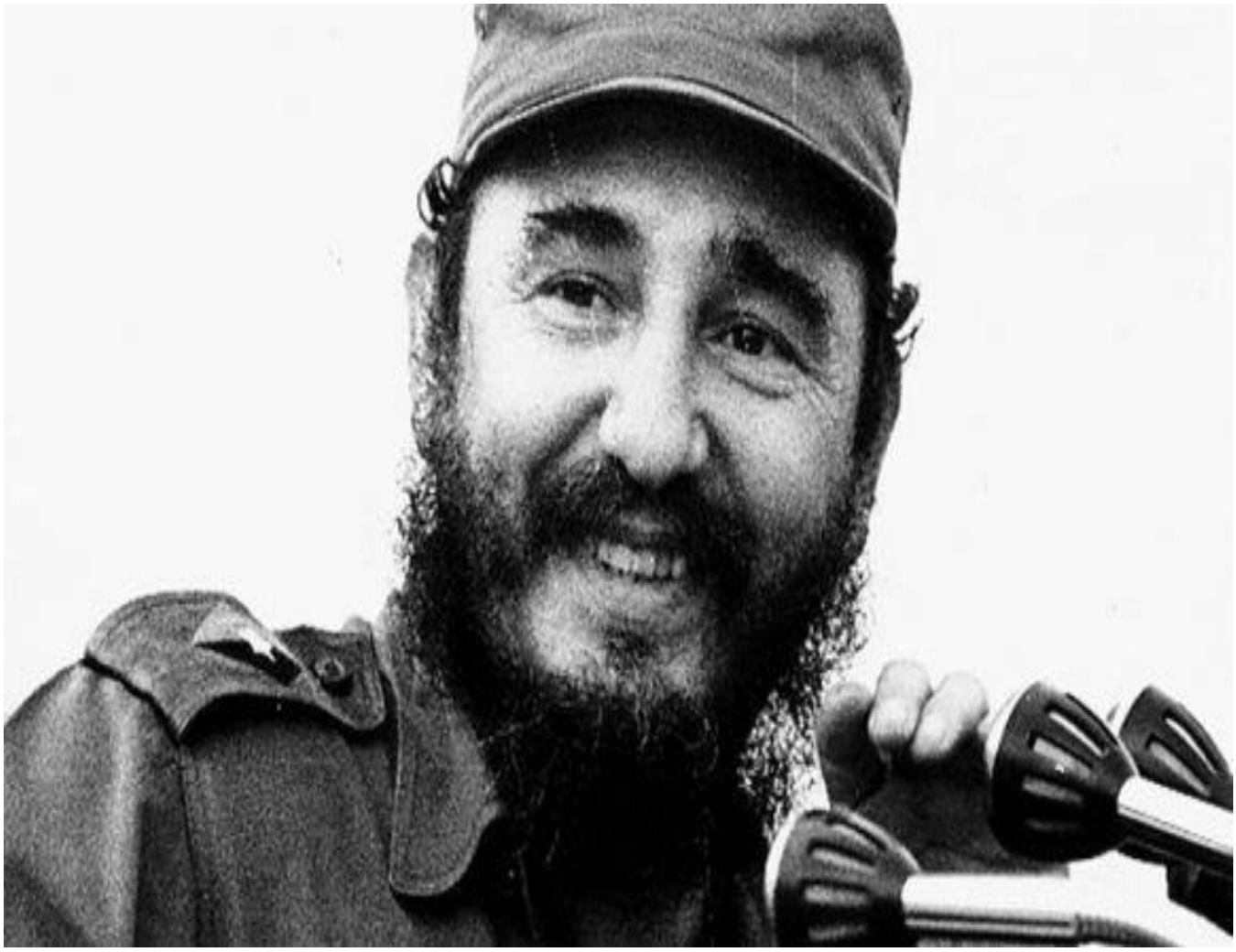
Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Abel Sarria Mediaceja

Tutores: Ing. Pavel Reyes Estévez
MSc. Aurelio Antelo Collado

La Habana, junio 2019

“Año 61 de la Revolución”



¨Nuestra juventud debe procurar adquirir aquellos conocimientos que sean más útiles en cada momento a la nación. Sobre todo, si se tiene en cuenta que estamos entrando en una etapa enteramente nueva¨.

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis que tiene por título: Requerimientos de seguridad de autenticación y generación de notificaciones en el Sistema Informático XABAL Arkheia 4.0 y reconozco 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 junio del año 2019.

_____

Abel Sarria Mediaceja

Autor

_____ _____

Ing. Pavel Reyes Estévez

Tutor

MSc. Aurelio Antelo Collado

Tutor

AGRADECIMIENTOS

A mis familiares que me han apoyado a lo largo de mi carrera en especial a los que están presentes hoy.

A mis amigos por los 5 años que hemos pasado juntos y que siempre han estado en las buenas y en las malas.

A mi novia por su apoyo en estos últimos meses.

A mi tutor Pavel por guiarme y apoyarme desde el primer día en el que nos asignaron la tesis.

A las demás personas que fui conociendo a lo largo de mi carrera, que fueron formando parte de mi círculo de amigos y que hoy están aquí.

A los profesores que contribuyeron a mi formación.

En fin gracias a todos los que de una forma u otra estuvieron presentes en estos 5 años.

DEDICATORIA

A mis familiares en especial a mi mamá que ha sido madre y padre a la vez.

RESUMEN

El Centro de Informatización de la Gestión Documental, es una organización especializada en el desarrollo de sistemas y servicios informáticos integrales de alta calidad y competitividad en la informatización de los procesos de gestión documental. Uno de estos sistemas es el producto XABAL Arkheia 4.0 un sistema para la gestión de documentos de archivos históricos que permite realizar el procesamiento de estos, su digitalización y brindar servicios. El presente trabajo de diploma tiene como título: Requerimientos de seguridad de autenticación y generación de notificaciones en el Sistema Informático XABAL Arkheia 4.0. La investigación se realizó en un tiempo aproximado de 7 meses y propone como objetivo general desarrollar requerimientos de seguridad de software para la gestión de archivos históricos en el sistema informático XABAL Arkheia 4.0 con el fin de mejorar la seguridad en la autenticación. Para la implementación de los requerimientos de seguridad de software antes mencionados fue necesario realizar un estudio sobre los requerimientos que habían sido implementados en la versión anterior del sistema. Durante el desarrollo se utilizaron como lenguajes de desarrollo JavaScript, XML, CSS y HTML5, Visual Paradigm como herramienta para el modelado del negocio y editores de código como Sublime Text y Visual Studio Code. Fue definida una estrategia de pruebas que permitió corroborar el éxito del desarrollo de la solución, en el escenario de validación.

Palabras claves: seguridad de software, gestión documental, informatización, procesos.

ABSTRACT

The Computerization Center for Documentary Management is an organization specialized in the development of high quality computer systems and services of high quality and competitiveness in the computerization of document management processes. One of these systems is the product XABAL Arkheia 4.0, a system for the management of documents of historical archives that allows to process them, digitize them and provide services. The present diploma work has the title: Authentication security requirements and generation of notifications in the XABAL Arkheia 4.0 Computer System. The research was conducted in an approximate time of 7 months and proposes as a general objective to develop software security requirements for the management of historical files in the XABAL Arkheia 4.0 computer system in order to improve security in authentication. For the implementation of the aforementioned software security requirements it was necessary to carry out a study on the requirements that had been implemented in the previous version of the system. During development, JavaScript, XML, CSS and HTML5 were used as development languages, Visual Paradigm as a tool for business modeling and code editors such as Sublime Text and Visual Studio Code. A test strategy was defined to corroborate the success of the development of the solution, in the validation scenario.

Keywords: software security, document management, computerization, processes.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Hoy en día la seguridad informática se ha convertido en una de las principales preocupaciones de las empresas. Por otro lado el uso de la información es cada vez más extenso por lo que los activos a proteger y las vulnerabilidades aumentan, los ciberataques son más frecuentes y complejos, llegando a tener consecuencias muy graves como la revelación de información entre otras, por lo que disponer de profesionales en seguridad que puedan proteger los activos en la red se hace imprescindible en todas la empresas por pequeñas o grandes que estas sean

La seguridad informática es una disciplina que se encarga de proteger la integridad y la privacidad de la información almacenada en un sistema informático. También se ocupa de diseñar los procedimientos, métodos y técnicas, orientados a proveer condiciones seguras y confiables, para el procesamiento de datos en sistemas informáticos (1), por lo que sería de vital importancia para cualquier empresa, institución, negocio, etc., velar por el correcto funcionamiento de los mecanismos establecidos para garantizar la seguridad en dichos sistemas informáticos permitiendo que la información que circula en el entorno donde se encuentra esté asegurada.

La seguridad de la información es el conjunto de medidas preventivas y reactivas de las organizaciones y de los sistemas tecnológicos que permiten resguardar y proteger la información buscando mantener la confidencialidad, la disponibilidad e integridad de datos y de la misma. El concepto de seguridad de la información no debe ser confundido con el de seguridad informática, ya que este último sólo se encarga de la seguridad en el medio informático, pero la información puede encontrarse en diferentes medios o formas, y no solo en medios informáticos. Para el hombre como individuo, la seguridad de la información tiene un efecto significativo respecto a su privacidad, la que puede cobrar distintas dimensiones dependiendo de la cultura del mismo (2).

La información y los procesos que la apoyan, los sistemas y las redes, son bienes importantes de las entidades, por lo que requieren ser protegidos convenientemente frente a amenazas que pongan en peligro la disponibilidad, la integridad, la confidencialidad de la información, la estabilidad de los procesos, aspectos necesarios para alcanzar los objetivos de la organización. La información generalmente es procesada, intercambiada y conservada en redes de datos, equipos informáticos y soportes de almacenamiento, que son parte de lo que se conoce como sistemas informáticos, los mismos están sometidos a potenciales amenazas de seguridad de diversa índole, originadas tanto desde dentro de la propia organización, como desde fuera, procedentes de una amplia variedad de fuentes (3). A los riesgos físicos, entre ellos, accesos no autorizados a la información, catástrofes naturales,

sabotajes, incendios, y accidentes; hay que sumarle los riesgos lógicos como contaminación con programas malignos, ataques de denegación de servicio entre otros, todos estos se pueden disminuir de forma significativa con ello las amenazas y la reducción del impacto sin necesidad de realizar elevadas inversiones ni contar con una gran estructura de personal. Para ello se hace necesario conocer y gestionar de manera ordenada los riesgos a los que está sometido el sistema informático, la elevación de los niveles de seguridad informática se consigue implantando un conjunto de controles, que incluyan políticas, procesos, procedimientos, estructuras organizativas y funciones de hardware y software, los que deben ser establecidos, implementados, supervisados y mejorados cuando sea necesario para cumplir los objetivos específicos de seguridad de la organización.

Para lograr todo esto es necesario que todo sistema informático tenga dentro del mismo los requerimientos de seguridad establecidos para garantizar que no se encuentre comprometida la información de dicho sistema. La identificación y autenticación es la primera línea de defensa para la mayoría de los sistemas computarizados y es un requerimiento que permite prevenir el ingreso de personas no autorizadas (4). Es la base para la mayor parte de los controles de acceso y para el seguimiento de las actividades de los usuarios. Se denomina identificación al momento en que el usuario se da a conocer en el sistema (cuando entra en el campo correspondiente su nombre de usuario); y autenticación a la verificación que realiza el sistema sobre esta identificación (cuando entra en el campo correspondiente su contraseña).

Existen dos razones para autenticar a los usuarios de un sistema: la identidad del usuario es un parámetro para la decisión de control de acceso y la identidad del usuario es registrada cuando se hace el loggin de eventos relevantes para la seguridad en la auditoría (5).

El proceso de autenticación consiste en varios pasos, obtener la información de autenticación de una entidad, analizar los datos, determinar si la información de autenticación es efectivamente asociada a la entidad. Para esto existen diversos mecanismos, pero en esta investigación solo tendremos en cuenta el mecanismo de contraseña secuencia de caracteres generado por el usuario. Esta contraseña es encriptada y luego almacenada, la misma nunca es desencriptada. Cuando el usuario se autentica el sistema puede tomar también en cuenta dónde se encuentra. Identificar el lugar desde el cual un usuario se autentica puede ayudar a resolver disputas sobre la verdadera identidad del mismo (5).

El surgimiento y evolución de las Tecnologías de la Información y las Comunicaciones (TIC), crea un escenario de cambio permanente, donde la rápida capacidad de adaptación e innovación son la clave para el éxito de las organizaciones.

Cuba tiene como objetivo que las TIC se conviertan en un sector de desarrollo estratégico y potencien una economía del conocimiento, por lo que desde hace varios años ha comenzado a dar pasos significativos en la digitalización y preservación de los archivos históricos de las instituciones competentes. Para ello se apoya en sistemas informáticos que garantizan la gestión, conservación y protección de dichos archivos.

La Universidad de las Ciencias Informáticas (UCI), centro educacional vinculado a la producción de software, está inmersa en esta tarea de vital importancia. Uno de sus centros productivos es el Centro de Informatización de la Gestión Documental (CIGED), el mismo se dedica al desarrollo de sistemas y servicios informáticos para: gestión de documentos administrativos, gestión de archivos históricos y gestión bibliotecaria y centros de documentación.

En este centro se desarrolla el sistema informático XABAL Arkheia que contiene un conjunto de reglas generales para la descripción archivística que pueden aplicarse con independencia del tipo documental o del soporte físico de los documentos de archivo. XABAL Arkheia 3.0 es una aplicación multiplataforma dirigida a la conservación y preservación de la memoria histórica de las instituciones de archivo. Permite la gestión de cuadros de clasificación para una institución, su estructura física y la descripción de los documentos de archivo utilizando la norma ISAD (G). Facilita la búsqueda, la interoperabilidad entre sistemas y la gestión de los servicios de información sobre los documentos, a la vez que brinda la posibilidad de extraer caracteres de documentos no manuscritos de las imágenes digitales asociadas a la descripción del documento. Gestiona los procesos de conservación y restauración de los documentos históricos archivando los momentos resultantes del tratamiento ejecutado sobre los mismos, así como los registros ambientales de los locales donde se conserven (6). Todo esto implementando un control en los mecanismos de seguridad a nivel de software.

Actualmente se está desarrollando la versión 4.0 del producto, la cual por estrategia del centro CIGED está implementada sobre el gestor de contenido empresarial Alfresco 5.2, esto trae como consecuencia que exista un cambio en cuanto a la tecnología en la que se desarrollará el producto ya que su versión anterior está desarrollada en Grails que es un framework de desarrollo web. Se pretende que la nueva versión cuente también con los requerimientos de seguridad de autenticación y generación de notificaciones implementados en su versión anterior, los cuales están basados en el certificado del Departamento de Ciberseguridad Tecnológica de la Dirección de Tecnologías y Sistemas del Ministerio del Interior del 2018, al estar implementada sobre Alfresco 5.2 no cuenta con algunos mecanismos de seguridad implementados en la versión anterior según las normas del certificado del Ministerio del Interior . Es de vital importancia, controlar el acceso y mantener la confidencialidad de la información de

los usuarios que se registren y accedan al sistema en su nueva versión ya que este producto se despliega en importantes instituciones del país.

De acuerdo a lo anteriormente expuesto y conociendo la importancia que tiene para un sistema informático la implementación de requerimientos de seguridad se plantea el siguiente **problema a resolver:** ¿Cómo mejorar la seguridad en la autenticación y generación de notificaciones para la gestión de los archivos históricos en el sistema informático XABAL Arkheia 4.0?

El **objeto de estudio** del presente trabajo es la seguridad en la autenticación y generación de notificaciones de los sistemas informáticos de gestión documental.

De acuerdo al problema a resolver planteado se define como **objetivo general**: Implementar los requerimientos de seguridad de software en la autenticación y generación de notificaciones para la gestión de archivos históricos en el sistema informático XABAL Arkheia 4.0.

Se especifica como **campo de acción** la seguridad en la autenticación y generación de notificaciones en el sistema informático XABAL Arkheia 4.0.

Para cumplir con el objetivo general expuesto, se han trazado varios **objetivos específicos** que facilitan y organizan el trabajo:

- ✓ Elaborar el marco teórico de la investigación partiendo del análisis de los antecedentes de las soluciones existentes sobre requerimientos de seguridad en sistemas informáticos.
- ✓ Implementar requerimientos de seguridad informáticos para las funciones de autenticación y generación de notificaciones.
- ✓ Validar el funcionamiento de los requerimientos implementados mediante una estrategia de prueba.

En aras de guiar la realización de la investigación se plantean las siguientes **preguntas de investigación:**

- ✓ ¿Cuáles son los principales conceptos involucrados en el ámbito de seguridad en los sistemas informáticos?
- ✓ ¿Cuál es el procedimiento para la implementación de los requerimientos de seguridad de autenticación en el sistema informático XABAL Arkheia 4.0?
- ✓ ¿Cuáles son los requerimientos de seguridad en la autenticación necesarios para controlar el acceso al sistema informático XABAL Arkheia 4.0?
- ✓ ¿Qué estrategia es necesaria definir para comprobar el correcto funcionamiento de los requerimientos de seguridad implementados en el sistema informático XABAL Arkheia 4.0?

Para dar cumplimiento al objetivo antes descrito se definen las siguientes **tareas de investigación:**

- ✓ Descripción de los conceptos fundamentales para la seguridad de sistemas informáticos.
- ✓ Identificación de los principales procedimientos para la detección de vulnerabilidades en sistemas informáticos.
- ✓ Selección de un procedimiento para la implementación de los requerimientos de seguridad de autenticación en el sistema informático XABAL Arkheia 4.0.
- ✓ Selección de la metodología de desarrollo de software, herramientas, lenguajes y tecnologías a utilizar para el desarrollo de la solución informática que permita la implementación de los requerimientos de seguridad.
- ✓ Descripción de los requerimientos de seguridad en la autenticación para el sistema informático XABAL Arkheia 4.0.
- ✓ Comprobación del funcionamiento de los requerimientos de seguridad implementados para demostrar el correcto funcionamiento del procedimiento seleccionado.

En el desarrollo de la investigación se utilizan los siguientes **métodos de investigación científica:**

Métodos teóricos

Analítico-Sintético: este método fue utilizado para estudiar todas las teorías y documentos referentes a la seguridad en sistemas informáticos, así como la extracción de los elementos y conceptos necesarios para el desarrollo de la investigación.

Histórico-Lógico: este método fue utilizado para estudiar los sistemas relacionados con la seguridad en sistemas informáticos, lo que permitió adquirir conocimiento sobre la forma en que se realizan estas tareas para la definición de los antecedentes de la investigación.

Modelación: este método fue utilizado para la modelación de la propuesta de solución, la creación de los diagramas de casos de uso y modelo de dominio para representar los componentes y relaciones de los requerimientos a implementar.

Métodos empíricos

Simulación: este método fue utilizado para la ejecución de los requerimientos implementados mediante el uso de datos artificiales de prueba, lo que permitió comprobar su funcionamiento para garantizar la calidad de ambos.

Análisis estático: este método fue utilizado para realizar un examen de la estructura de los requerimientos implementados. Su utilización permitió además la aplicación una estrategia de prueba para detectar y corregir errores.

El presente documento está compuesto por introducción, desarrollo, conclusiones y referencias bibliográficas. El contenido del desarrollo está estructurado en tres capítulos que desarrollan su contenido de la siguiente manera:

Capítulo 1. Fundamentación teórica: expone los elementos teóricos de la investigación. Se realiza el estudio e investigación de soluciones informáticas existentes para la implementación de los requerimientos de seguridad informática autenticación y generación de notificaciones. Se determinan la metodología de desarrollo, herramientas y tecnologías que se utilizan para desarrollar la solución propuesta.

Capítulo 2. Elementos fundamentales de la propuesta de solución: expone la propuesta de solución a través de una breve descripción, el modelo de dominio que corresponde directamente a la solución del problema así como los diferentes requisitos y casos de uso correspondientes.

Capítulo 3. Implementación y comprobación de la propuesta de solución: incluye las definiciones y los resultados de las pruebas realizadas al sistema, para comprobar la efectividad de la implantación de los requerimientos de seguridad

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se exponen los aspectos teóricos tratados en el desarrollo de la investigación. Se incluyen los principales conceptos relacionados con el tema del presente trabajo, así como el resultado del estudio realizado sobre las soluciones existentes a nivel nacional e internacional para la implementación de los requerimientos de seguridad informática autenticación y generación de notificaciones. Se describen además, la metodología, herramientas, tecnologías y lenguajes seleccionados para el desarrollo de la solución.

1.1 Aspectos teóricos de la investigación

Para tratar la implementación de los requerimientos de seguridad informática autenticación y generación de notificaciones en contribución a la representación de información, es necesario identificar los conceptos fundamentales que permitan la comprensión del presente trabajo. En esta sección se definen los aspectos teóricos de la investigación.

Seguridad Informática

Proceso de prevenir y detectar el uso no autorizado de un sistema informático. Implica el proceso de proteger contra intrusos el uso de nuestros recursos informáticos con intenciones maliciosas o con intención de obtener ganancias, o incluso la posibilidad de acceder a ellos por accidente (7).Por lo que se comprende la importancia de que exista un correcto funcionamiento en los diferentes mecanismos que se implementan en los actuales sistemas informáticos con el fin de mantener la confidencialidad e integralidad de los datos.

Información

Denominamos al conjunto de datos, ya procesados y ordenados para su comprensión, que aportan nuevos conocimientos a un individuo o sistema sobre un asunto, materia, fenómeno o ente determinado (8). En dependencia del aprovechamiento que hagamos de la información, en este sentido, sería la base racional del conocimiento.

Seguridad de la Información

Se refiere a la protección de los activos de información fundamentales para el éxito de cualquier organización. se define como activos de información correos electrónicos, páginas web, imágenes, bases de datos, faxes, contratos, presentaciones, evaluaciones, contabilidad, documentos legales, etc.,

estos pueden proceder de distintas fuentes de información dentro de la empresa y que pueden encontrarse en diferentes soportes, como papel o medios digitales (3).

Integridad

La integridad es la propiedad que busca mantener los datos libres de modificaciones no autorizadas. La violación de integridad se presenta cuando un empleado, programa o proceso (por accidente o con mala intención) modifica o borra los datos importantes que son parte de la información, así mismo hace que su contenido permanezca inalterado a menos que sea modificado por personal autorizado, y esta modificación sea registrada, asegurando su precisión y confiabilidad (9). La violación de la integridad genera que se pierda información o puede crear una nueva información totalmente diversificada ocasionando que se pierda la veracidad de la misma.

Privacidad

Es el aspecto de la tecnología de la información (TI) que se ocupa de la capacidad que una organización o individuo tiene para determinar qué datos en un sistema informático pueden ser compartidos con terceros (10). Es la capacidad de ejercer un control sobre la información que posee un determinado usuario y que solo pueda acceder la persona autorizada por lo que constituye una de las principales características que debe tener un sistema informático dentro de los requerimientos de seguridad.

Identificación

Es la capacidad de identificar de forma exclusiva a un usuario de un sistema o una aplicación que se está ejecutando en el sistema el cual utiliza el ID de usuario para identificarlo (11). Este proceso se realiza antes de la autenticación y juntos tienen la capacidad de demostrar que un usuario es quien asegura ser.

Autenticación

Existen diversos mecanismos de autenticación, para la presente investigación se hará referencia al proceso de validar la identidad de un usuario mediante un ID de usuario y una contraseña (11).Se considera el primer bloque de seguridad de un sistema informático el cual tenga una gestión de usuario dentro de sus funcionalidades. Por ejemplo, considere el caso de un usuario que se conecta a un sistema especificando un ID de usuario y una contraseña. El sistema autentica al usuario en el momento de la conexión comprobando que la contraseña proporcionada es correcta (11).

1.2 Estudio de los antecedentes de la investigación.

Una vez aclarado los conceptos fundamentales involucrados en la investigación, el siguiente paso consiste en evidenciar como se manifiesta la implementación de los requerimientos de seguridad en algunos sistemas. Esta actividad, destinada fundamentalmente a determinar si estos requerimientos funcionan correctamente así como si un cambio determinado mejora su funcionamiento y resultados, se describe en la presente sección.

Cuando hablamos de seguridad informática sin duda uno de los pioneros en el tema fue James P. Anderson, quien allá por 1980 y a pedido de un ente gubernamental produjo uno de los primeros escritos relacionados con el tema, y es allí donde se sientan también las bases de palabras que hoy suenan como naturales, pero que por aquella época parecían ciencia ficción. El documento se llamó: Computer Security Threat Monitoring and Surveillance, describe ahí la importancia del comportamiento enfocado hacia la seguridad en materia de informática (12).

En la seguridad informática otro aspecto importante corresponde con los servicios de seguridad, ya que mejoran en un buen porcentaje la seguridad de un sistema y el flujo de información de una organización, estos servicios están dirigidos a evitar los ataques de seguridad y utilizan uno o más mecanismos de seguridad para proveer el servicio.

Ya sabemos que unos requerimientos primordiales de los sistemas informáticos que desempeñan tareas importantes son los mecanismos de seguridad adecuados a la información que se intenta proteger; los mismos incluyen al menos un sistema que permita identificar a las entidades (generalmente usuarios) que intentan acceder a la información mediante procesos tan simples como una contraseña o tan complejos como un dispositivo analizador de patrones retinales.

Los sistemas que habitualmente utilizamos los humanos para identificar a una persona, como el aspecto físico o la forma de hablar, son demasiado complejos para una computadora; el objetivo de los sistemas de identificación de usuarios no suele ser identificar a una persona, sino autenticar que esa persona es quien dice ser realmente. Aunque como humanos seguramente ambos términos nos parecerán equivalentes, para un ordenador existe una gran diferencia entre ellos (13).

Los métodos de autenticación se suelen dividir en tres grandes categorías en función de lo que utilizan para la verificación de identidad:

- Algo que el usuario sabe.
- Algo que éste posee.
- Una característica física del usuario o un acto involuntario del mismo.

Es fácil ver ejemplos de cada uno de estos tipos de autenticación: una contraseña es algo que el usuario conoce y el resto de personas no, una tarjeta de identidad es algo que el usuario lleva consigo, la huella

dactilar es una característica física del usuario, y un acto involuntario podría considerarse que se produce al firmar (al rubricar la firma no se piensa en el diseño de cada trazo individualmente) (13).El tipo de autenticación que se tendrá en cuenta para la implementación como requisito de seguridad será la autenticación de un usuario mediante contraseña.

Es el modelo de autenticación más básico el cual consiste en decidir si un usuario es quien dice ser simplemente basándonos en una prueba de conocimiento que priori sólo ese usuario puede superar y desde los primeros sistemas que existieron hasta los más modernos sistemas Unix, esa prueba de conocimiento no es más que una contraseña que en principio es secreta. Evidentemente, esta aproximación es la más vulnerable a todo tipo de ataques, pero también la más barata, por lo que se convierte en la técnica más utilizada en entornos que no precisan de una alta seguridad, como es el caso de los sistemas Unix en redes normales (y en general en todos los sistemas operativos en redes de seguridad media-baja); otros entornos en los que se suele aplicar este modelo de autenticación son las aplicaciones que requieren de alguna identificación de usuarios, como el software de cifrado PGP o el escáner de seguridad NESSUS. También se utiliza como complemento a otros mecanismos de autenticación, por ejemplo en el caso del Número de Identificación Personal (PIN) a la hora de utilizar cajeros automáticos (13).

En todos los esquemas de autenticación basados en contraseñas se cumple el mismo protocolo: las entidades (generalmente dos) que participan en la autenticación acuerdan una clave, clave que han de mantener en secreto si desean que la autenticación sea fiable. Cuando una de las partes desea autenticarse ante otra se limita a mostrarle su conocimiento de esa clave común, y si ésta es correcta se otorga el acceso a un recurso. Lo habitual es que existan unos roles preestablecidos, con una entidad activa que desea autenticarse y otra pasiva que admite o rechaza a la anterior (en el modelo del acceso a sistemas Unix, tenemos al usuario y al sistema que le permite o niega la entrada).

1.2.1 Sistema informático existente.

En el centro CIGED se encuentra el sistema informático XABAL Arkheia 3.0, dirigido a la conservación y preservación de la memoria histórica de las instituciones de archivo (6). La seguridad del sistema a nivel de software está regida por el certificado del Departamento de Ciberseguridad Tecnológica de la Dirección de Tecnologías y Sistemas del Ministerio del Interior del 2018, el cual tiene una normativa para la implementación de mecanismos de seguridad reunidos en los siguientes requerimientos:

Autenticación: Comprobación de la identidad del usuario.

Autorización o control de acceso: Permisos correspondientes para el uso de los recursos informáticos manejados por la solución informática. Se regirán por reglas de control de acceso las cuales serán aprobadas por diferentes niveles (ministerial, órgano central, departamental, etc.)

Auditoría: guardar trazas de todas las operaciones realizadas por el usuario sobre la solución, así como otros eventos de interés, que permitan determinar funcionamientos incorrectos de la solución, así como resolver incidentes de seguridad.

Notificaciones: informar a las personas encargadas ante funcionamiento incorrecto y usos indebidos de la solución informática a través del correo electrónico.

Integridad de los datos: mecanismo que permita validar todos los datos de entrada a la solución tanto texto como ficheros.

Administración de la seguridad: módulo de administración de la seguridad que permita realizar de forma diferenciada la administración funcional sobre la base del diseño de la seguridad.

Dichos requerimientos se encuentran implementados en el lenguaje de programación Groovy utilizando el framework Grails 2.2.1 para programación web.

Conclusiones de los estudios de los antecedentes de la investigación

El sistema informático analizado anteriormente tiene incorporado los requerimientos de seguridad según la normativa del Ministerio del Interior, estos funcionan correctamente, aunque para este trabajo solo se tendrán en cuenta los requerimientos de autenticación y generación de notificaciones de dicha normativa. Se concluye que llevando acabo todo el proceso de cambio de tecnología que presenta la nueva versión del sistema informático la cual se está desarrollando sobre el gestor de contenido empresarial Alfresco 5.2, se analizó que no tiene activado un mecanismo para bloquear una sesión de usuario dado número de intentos fallidos de acceso al sistema, no evita que un usuario pueda acceder a su sesión encontrándose dicha sesión abierta en otra PC, tampoco establece los requisitos mínimos de longitud de las contraseñas de los usuarios, no existe un mecanismo para notificar al administrador del sistema ante errores en el acceso al mismo, por tanto se podría mejorar la seguridad en la autenticación de la nueva versión incorporándole los requerimientos antes mencionados correspondientes a la normativa del Ministerio del Interior ya que en la versión anterior del producto XABAL Arkheia la implementación de estos requerimientos se realizó con éxito garantizando un correcto acceso a los datos de gran importancia a nivel institucional que son almacenados en el sistema informático.

1.3 Metodología, lenguajes y herramientas de desarrollo.

En aras de preparar el ambiente para la implantación de los requerimientos de seguridad, se realiza la identificación y el análisis de la metodología de desarrollo, lenguaje de programación, entorno de

desarrollo y herramienta de modelado. En esta sección se introduce el análisis de las principales características y ventajas de cada uno de los elementos antes mencionados que justifican su elección como vía para dar solución a la problemática planteada en cumplimiento de los objetivos definidos.

1.3.1 Metodología de desarrollo

Existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de implementación. Por una parte se tienen aquellas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir así como las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos, ya sea debido a la numerosa documentación que generan, al tamaño de los equipos que le utilizan así como otras múltiples causas. En nuestra universidad existen diversos centros productivos cada uno de ellos dedicados a la producción de software y/o servicios por lo que hace que la productividad en la UCI sea cada vez más amplia (14).

Actualmente el desarrollo de software dentro de la actividad productiva de la UCI se caracteriza por el uso de diferentes metodologías de desarrollo entre robustas y ágiles, específicamente las nueve metodologías que se listan a continuación:

- XP
- OPEN UP
- RUP
- BPM
- DAC
- KIMBALL
- SXP
- SCRUM
- NOVA OPEN UP

A pesar de la variedad de metodologías usadas, se ha comprobado que muy pocos proyectos la aplican en su totalidad. Las diferencias entre estas metodologías no radica únicamente en los productos de trabajos que proponen o en sus roles, sino en su forma de planificar el proyecto y realizar las estimaciones del tiempo. Factor determinante en la culminación exitosa de todo desarrollo de software, por lo que uno de los principales problemas detectados es que sin importar la metodología que se usa se está planificando con un único cronograma tipo, además de forzar el método de estimación definido en la Universidad y que responde en su gran mayoría a la metodología RUP (14).

El objetivo de este epígrafe es describir la metodología de desarrollo AUP UCI que es la destinada a emplearse en los proyectos productivos de la UCI. Dicha definición se basa en una variación de la metodología Proceso Unificado Ágil (AUP por sus siglas en inglés) en unión con el modelo CMMI-DEV v1.3 (14), el cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora.

Descripción de las fases

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, la cual se llama Ejecución y se agrega la fase de Cierre (14). Por lo tanto quedaría conformado de la siguiente forma:

- Inicio.
- Ejecución.
- Cierre.

A continuación se describen los procesos realizados en cada una de las fases antes mencionadas:

Inicio

Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto (14).

Ejecución

En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto (14).

Cierre

En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto (14).

Descripción de las disciplinas o artefactos

Se decide para el ciclo de vida de los proyectos de la UCI tener 7 disciplinas, pero a un nivel más atómico que el definido en AUP (14).

Modelado del Negocio

El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para modelar el negocio se proponen las siguientes variantes (14):

1- Casos de Uso del Negocio (CUN).
2- Descripción de Proceso de Negocio (DPN).
3- Modelo Conceptual (MC).

Requisitos

El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto. Existen tres formas de encapsular los requisitos (14):

1- Casos de Uso del Sistema (CUS)
2- Historias de usuario (HU)
3- Descripción de requisitos por proceso (DRP).

Análisis y Diseño

En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis (14).

Implementación

En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema (14).

Pruebas interna

En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posibles componentes de prueba ejecutables para automatizar las pruebas (14).

Pruebas de liberación

Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación (14).

Pruebas de Aceptación

Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (14).

La metodología define cuatro variantes de escenarios que se tienen en cuenta para un correcto modelado del negocio.

Escenario No 1:

Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS. Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema. Es necesario que se tenga claro por el proyecto que los CUN muestran como los procesos son llevados a cabo por personas y los activos de la organización.

Figura 1 Escenario No: 1.

Escenario No 2:

Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS. Este es el escenario que se llevará a cabo en la presente investigación ya que se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información.

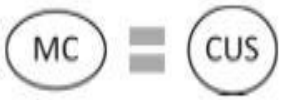


Figura 2 Escenario No: 2

Escenario No 3:

Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP. Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad. Se debe tener presente que este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y la relaciones entre los procesos identificados.

Figura 3 Escenario No: 3

Escenario No 4:

Proyectos que no modelen negocio solo pueden modelar el sistema con HU. Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información.

Figura 4 Escenario No: 4

Conclusiones de la metodología de desarrollo

Después de analizar los diferentes escenarios que nos brinda esta metodología se decidió que el más adecuado para el desarrollo de la siguiente investigación es el escenario 2 por lo descrito anteriormente. Con la adaptación de AUP que se propone para la actividad productiva de la UCI se logra estandarizar el proceso de desarrollo de software, dando cumplimiento además a las buenas prácticas que define CMMI-DEV v1.3. Se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos. Se redujo a 1 la cantidad de metodologías que se usaban y de más de 20 roles en total que se definían se redujeron a 11.

1.3.2 Herramientas CASE

Estas herramientas CASE modelan la información de negocios cuando ésta se transfiere entre distintas entidades organizativas en el seno de una compañía. El objetivo primordial de las herramientas de esta categoría consiste en representar objetos de datos de negocios, sus relaciones, y ayuda a comprender mejor la forma en que fluyen estos objetos de datos entre distintas zonas de negocio en el seno de la compañía. Estas herramientas proporcionan una ayuda importante cuando se diseñan nuevas estrategias para los sistemas de información y cuando los métodos y sistemas no satisfacen las necesidades de la organización (15).

Visual Paradigm 8.0

Visual Paradigm es una potente herramienta multiplataforma de diseño y gestión fácil de usar para sistemas de TI. Visual Paradigm proporciona a los desarrolladores de software la plataforma de desarrollo de vanguardia para crear aplicaciones de calidad más rápido, mejor y más barato. Facilita una excelente interoperabilidad con otras herramientas CASE y la mayoría de los IDE líderes, lo que supera a todo su proceso de desarrollo de Despliegue de Código de Modelo en esta solución integral (16). Por tanto es la herramienta elegida para llevar a cabo la representación de todos los diagramas que se generan en los diferentes artefactos correspondientes a la metodología utilizada.

1.3.3 Editores de código

Sublime Text 2

Sublime Text es un editor de código multiplataforma y ligero (17). Permite tener varios documentos abiertos mediante pestañas, e incluso emplear varios. El sistema de resaltado de sintaxis de Sublime Text soporta un gran número de lenguajes pero para esta investigación solo se tendrán en cuenta HTML, Java, JavaScript, XML (17). Soporta frameworks y auto completar, entre otras funcionalidades. Algunas de sus características son ampliables mediante plugins (17).

Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para plataformas Windows, Linux y macOS, que fue lanzado el 29 de Abril de 2015 (18). Visual Studio Code es gratuito y de código abierto. Tiene soporte nativo para JavaScript, pero no solo esto, posee complementos que permite trabajar con HTML, CSS (19).

A la izquierda despliega un editor de archivos el cual se utilizó para acceder a la estructura de carpeta que presenta Alfresco 5.2. Mientras que a la derecha, muestra el contenido del archivo que estés editando lo cual junto con las múltiples ventanas que permiten trabajar en varios archivos simultáneamente, contribuyó a un mejor desarrollo de la implementación del código.

Por último indicar que, al igual que sucede con otros editores, es necesario añadir algunos complementos para que sea usable.

1.3.4 Lenguaje de programación y entorno de desarrollo

Alfresco 5.2

Software de gestión de contenido empresarial de código abierto que puede trabajar con cualquier tipo de contenido, permitiendo a los usuarios compartir y colaborar con el contenido fácilmente. A menudo, la suma de las partes es mayor que los elementos individuales. Esto es aplicable a la plataforma empresarial digital de Alfresco, que activa procesos y contenido de manera inteligente para acelerar el flujo del negocio. Modular y fácil de integrar y extender, permite desarrollar rápidamente aplicaciones basadas en contenido y procesos, proporcionando una gran experiencia de usuario que permite una verdadera transformación digital del negocio (20). Nos brinda una mayor productividad al tener toda la documentación en un solo repositorio es más fácil encontrar y recuperar la información, agilizando las tareas dentro de la organización y de respuestas a los clientes. También posee una arquitectura escalable lo que le permite soportar las mejoras continuas sin perder la calidad en los servicios que presta. Incluye además un conjunto de Interfaz de Aplicación de Programación (API) para la gestión de sus contenidos como son: Java API para la vinculación con código Java y JavaScript API para la vinculación con código JavaScript.

FreeMarker

Apache FreeMarker es un motor de plantillas: una biblioteca de Java para generar resultados de texto (páginas web HTML, correos electrónicos, archivos de configuración, código fuente, etc.) basados en plantillas y datos cambiantes. Las plantillas están escritas en el lenguaje de plantillas FreeMarker (FTL), que es un lenguaje simple y especializado (no un lenguaje de programación completo como PHP). Generalmente, se usa un lenguaje de programación de propósito general (como Java) para preparar los datos (emitir consultas de base de datos, hacer cálculos de negocios). Luego, Apache FreeMarker muestra los datos preparados usando plantillas. En la plantilla, se está enfocando en cómo presentar los datos, y fuera de la plantilla, se está enfocando en qué datos presentar (21).

Una de las principales ventajas que usa el framework WebScript de este motor es que permite generar diferentes vistas y representaciones como resultado de la ejecución de un mismo servicio. De esta manera se logra la separación definitiva entre la lógica de negocio que se ejecuta durante una petición y el formato de salida de la respuesta generada. Otra característica que posee este motor de plantillas es que es gratis por lo que nos facilita su uso.

JavaScript

JavaScript (JS) es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase, más conocido como el lenguaje de script para páginas web. Es un lenguaje script multiparadigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativa (22). Es el encargado de darle dinamismo a las páginas web trabajando en conjunto con HTML y CSS(los cuales se conceptualizarán más adelante). Alfresco 5.2 posee una API para JavaScript que permite la modificación y creación de nodos, aspectos y propiedades del repositorio del gestor de contenido empresarial:

Se utilizó para incorporar el código de JavaScript dentro del HTML mediante el uso de archivos externos contenedores del código, es la técnica recomendada ya que reduce los tiempos de descarga, incrementa la productividad, poder distribuir y reusar código en cada momento sin comprometer la eficiencia al tener todos los códigos en uno o más archivos externos solo tendríamos que hacer referencia en el HTML a la dirección en que se encuentran dichos archivos para su correcta ejecución.

CSS 3

Este lenguaje es, de hecho, un complemento desarrollado para superar las limitaciones y reducir la complejidad de HTML. CSS fue adoptado como la forma de separar la estructura de la presentación. La versión 3 de CSS permite hacer webs más elaboradas y dinámicas, con mayor separación entre estilos y contenidos. La especificación de HTML5 fue desarrollada considerando CSS3 a cargo del diseño. Debido a esta consideración, la integración entre HTML y CSS es ahora vital para el desarrollo web y esta es la razón por la que cada vez que mencionamos HTML5 también estamos haciendo referencia a CSS3, aunque oficialmente se trate de dos tecnologías completamente separadas. En este momento las nuevas características incorporadas en CSS3 están siendo implementadas e incluidas junto al resto de la especificación en navegadores compatibles con HTML5 (23). Al igual que en JavaScript se considera la técnica de archivos externos contenedores del código para la vinculación con el HTML.

HyperText Markup Language (HTML5)

HTML5 es la quinta revisión y la versión más reciente del estándar HTML en combinación con CSS3 y JavaScript. Ofrece nuevas funciones que proporcionan no solo soporte de medios enriquecidos, sino que también mejoran el soporte para crear aplicaciones web que pueden interactuar con los usuarios, sus datos locales y servidores de forma más fácil y efectiva de lo que antes era posible (24). HTML5 propone estándares para cada aspecto de la web y también un propósito claro para cada una de las tecnologías involucradas (25). Este lenguaje de marcado fue el utilizado para mostrar y estructurar el contenido de las páginas.

XML

XML es un lenguaje de marcado similar a HTML. Significa Extensible Markup Language (Lenguaje de Marcado Extensible) y es una especificación de W3C como lenguaje de marcado de propósito general. Esto significa que, a diferencia de otros lenguajes de marcado, XML no está predefinido, por lo que debes definir tus propias etiquetas. El propósito principal del lenguaje es compartir datos a través de diferentes sistemas, como Internet (26), así como representar información estructurada en la web la cual puede ser procesada, visualizada por muy diversos tipos de aplicaciones y dispositivos. Tiene como ventajas que es fácilmente procesable, separa radicalmente el contenido y el formato de presentación, está diseñado para cualquier tipo de lenguaje y alfabeto. Principales características extensibilidad, estructura y validación.

YUI

YUI es un framework de JavaScript y CSS de código abierto y gratuito para crear aplicaciones web interactivas. Ha demostrado ser escalable, rápido y robusto, es una biblioteca de JavaScript de uso industrial para profesionales, ya sea que se ejecute en dispositivos móviles, en navegadores de escritorio o incluso en el servidor. YUI efectos simples en una web Página y aplicaciones de ingeniería mantenibles, de rendimiento y bien diseñadas. Proporciona una Interfaz de Programación de Aplicaciones (API por sus siglas en inglés) concisa, conveniente e intuitiva que es liviana e increíblemente rápida, además de una infraestructura bien pensada y un conjunto completo de herramientas para ayudar a codificar (27).

Conclusiones del capítulo

En el capítulo se elaboró el marco teórico de la investigación a partir del análisis de los antecedentes de las soluciones existentes orientadas a los sistemas de seguridad informáticos. Para el cumplimiento del

objetivo se identificaron y definieron los conceptos fundamentales para obtener conocimientos previos para la implantación de los requerimientos de seguridad en la autenticación y generación de notificaciones en el sistema informático XABAL Arkheia 4.0 que permitieron la descripción del objeto de estudio de la investigación. Para preparar el ambiente de desarrollo se realizó el estudio de la metodología AUP UCI la cual es la que se usa en la universidad con el fin de estandarizar los procesos en el desarrollo de software y servicios. En apoyo a la metodología se usó la herramienta CASE Visual Paradigm 8.0 que permite el modelado de artefactos como el modelo de dominio para la comprensión de la propuesta de solución y los diagramas de uso para la comprensión del negocio. También se caracterizaron los lenguajes de programación CSS, JavaScript, HTML5 para la implementación de los requerimientos de seguridad así como los editores de códigos con los cuales se llevó a cabo dicha implementación.

CAPÍTULO 2. ELEMENTOS FUNDAMENTALES DE LA PROPUESTA DE SOLUCIÓN

Introducción

En el presente capítulo se describen los principales artefactos relacionados con el análisis de la propuesta de solución al problema identificado y señalado en la actual investigación. Se muestran los diferentes diagramas generados por las disciplinas de la metodología utilizada para lograr el desarrollo como el modelo de dominio para entender los principales conceptos involucrados para la solución informática, la especificación de requerimientos y la descripción de los casos de uso para lograr un modelado del negocio. Por tanto se llevó a cabo todo el proceso de análisis para lograr la implementación de la propuesta de solución.

Propuesta de solución

A partir del estudio de los antecedentes y consecuentemente con el objetivo de la investigación, se concluye que es necesario implementar los requerimientos de seguridad en la autenticación y notificaciones que permitan mejorar la seguridad informática del sistema en el proceso de autenticación. Con lo propuesto para la implementación de los requerimientos antes mencionados el usuario tendrá la posibilidad de contar con una solución informática que incorporará un mecanismo para el bloqueo de sesión de un usuario ante tres intentos fallidos de acceso al sistema, no permitirá el acceso de un usuario con sesión abierta en el servidor, cumplirá con los requisitos mínimos de longitud de contraseñas. La solución informática incorporará requerimientos para notificaciones los cuales permitirán ante cualquier fallo de los mecanismos anteriores mencionados enviarlas ante funcionamiento incorrecto o uso indebido de la solución informática a la persona encargada.

Modelo de dominio

Es una representación visual de clases conceptuales o de objetos reales en un dominio de interés. No constituyen clases de software, son representaciones de los conceptos involucrados para la comprensión de la solución y las asociaciones establecidas describen las relaciones entre los conceptos (28). Por tanto, un modelo de dominio constituye una representación conceptual de las relaciones entre los objetos involucrados en el proceso de desarrollo del sistema.

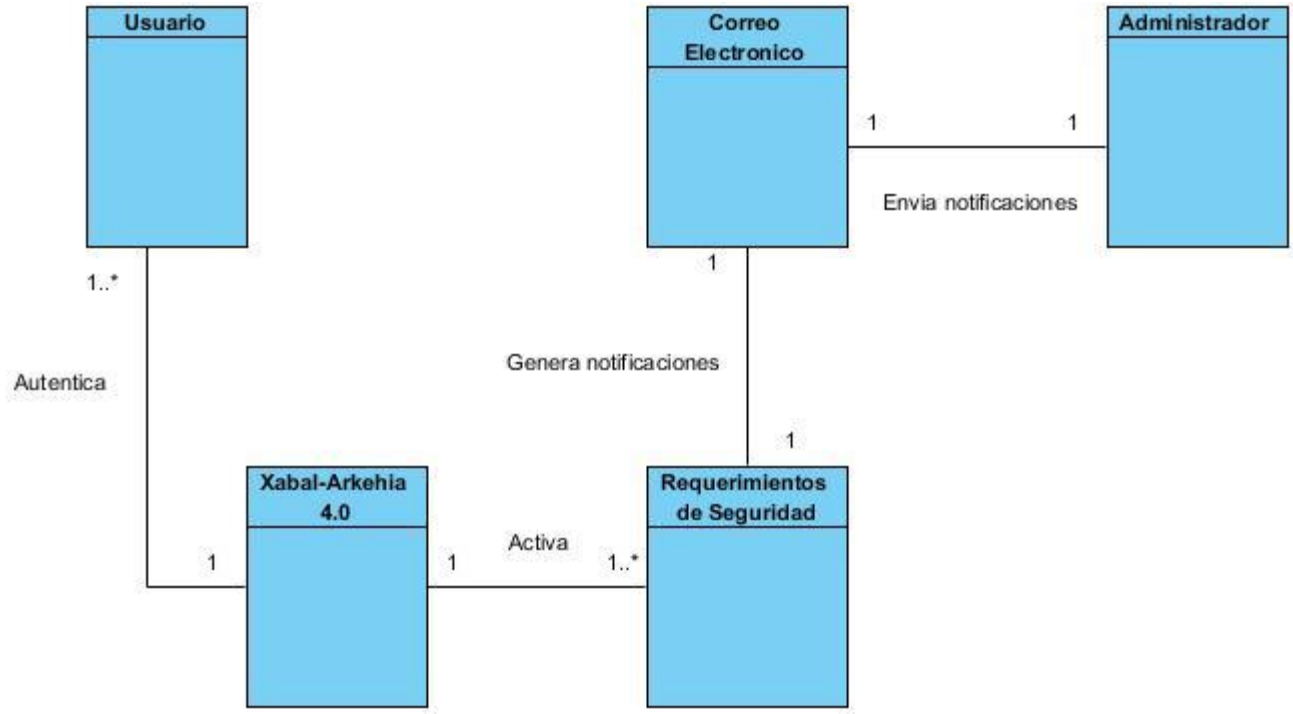


Figura 5 Diagrama del Modelo de Dominio.

La figura 1 muestra los principales conceptos y como están relacionados entre ellos para lograr una mejor comprensión de lo que se plantea alcanzar con el desarrollo de la presente investigación. A continuación se hace una breve descripción de las clases mostradas anteriormente en el diagrama.

Descripción de las clases de dominio

Usuario: persona que se autentica en el sistema con el fin de utilizar los servicios que brinda.

XABAL Arkheia 4.0: sistema para la gestión de documentos de archivos históricos que permite realizar el procesamiento de estos, su digitalización y brindar servicios.

Requerimientos de seguridad: conjunto de mecanismos implementados para lograr la integridad y confidencialidad de los datos de los usuarios que se encuentran registrados en el sistema.

Correo electrónico: servicio de red utilizado para notificar al administrador del sistema ante la detección de errores de autenticación.

Administrador: persona encargada de controlar el acceso de los usuarios al sistema y encargada de velar por el funcionamiento de los requerimientos de seguridad.

2.1 Requerimientos funcionales

Los requerimientos funcionales son declaraciones de los servicios que proveerá el sistema, de la manera en que éste reaccionará a entradas particulares (28).Por tanto se considera fundamental a la hora de identificarlos que respondan a todas las funcionalidades que será capaz de realizar la solución informática.

Muchos de los problemas de la ingeniería de software provienen de la imprecisión en la especificación de requerimientos. Para un desarrollador de sistemas es natural dar interpretaciones de un requerimiento ambiguo con el fin de simplificar su implementación. Sin embargo, a menudo no es lo que el cliente desea. Se tienen que estipular nuevos requerimientos y se deben hacer cambios al sistema, retrasando la entrega de éste e incrementando el costo (28).

En principio, la especificación de requerimientos funcionales de un sistema debe estar completa y ser consistente. La compleción significa que todos los servicios solicitados por el usuario están definidos. La consistencia significa que los requerimientos no tienen definiciones contradictorias.

El sistema informático XABAL Arkheia 4.0 incluirá los requerimientos de seguridad en la especificación de requisitos del proyecto y se implementarán durante todo el ciclo de desarrollo. Para el presente trabajo sólo se tendrán en cuenta los requerimientos de autenticación y notificaciones.

Autenticación:

No.	Nombre	Descripción
RF1	Bloquear cuenta de usuario.	El sistema debe permitir el bloqueo de la cuenta del usuario ante tres intentos fallidos de acceso.
RF2	Establecer contraseña	El sistema debe permitir que las contraseñas establecidas cumplan con los requisitos de longitud como mínimo 8 caracteres.
RF3	Cambiar contraseña	El sistema debe permitir que la nueva contraseña cumpla con los requisitos de longitud como mínimo 8 caracteres.
RF4	Controlar historial de contraseña	El sistema debe mantener el historial de contraseñas de los usuarios con un mínimo de 24 contraseñas. No se le permitirá el uso

		de ninguna contraseña almacenada en el historial.
RF5	Destruir múltiples sesiones	El sistema solo debe permitir acceder a un mismo usuario a la vez desde diferentes lugares.

Tabla 1 Requerimientos funcionales de autenticación.

Notificaciones:

No.	Nombre	Descripción
RF6	Generar notificación de intentos de contraseña incorrectas	El sistema envía un correo al administrador ante tres intentos de inicio de sesión con contraseña incorrecta.
RF7	Generar notificación de usuario inexistente.	El sistema envía un correo al administrador ante una repetición de usuario inexistente desde una misma dirección IP.
RF8	Generar notificación de inicio de sesión con usuario activo.	El sistema envía un correo al administrador ante intento de inicio de sesión con usuario que posee sesión activa en el servidor.

Tabla 2 Requerimientos funcionales de notificaciones.

2.2 Requerimientos no funcionales

Representan características generales y restricciones de la aplicación o sistema que se esté desarrollando. Suelen presentar dificultades en su definición dado que su conformidad o no conformidad podría ser sujeto de libre interpretación, por lo cual es recomendable acompañar su definición con criterios de aceptación que se puedan medir (29). Por tanto son requerimientos que imponen restricciones en el diseño y la implementación siendo en su conjunto cualidades o propiedades que el producto debe tener para lograr la satisfacción del cliente.

No.	Nombre	Descripción
RNF1	Seguridad lógica y de datos	1-El sistema debe asegurar que los datos estén protegidos del acceso no autorizado. 2- El sistema debe permitir deshabilitar la sesión después de un tiempo de inactivo.

		3-Los identificadores de las sesiones cambiaran en cada sesión iniciada o terminada por el usuario. 4- Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador.
RNF2	Usabilidad	1-Utilizar el idioma español para los mensajes y textos de la interfaz. 2-Las páginas que requieran autenticarse para acceder a ella tendrán un vínculo para cerrar sesión.
RNF3	Portabilidad	1-Se puede utilizar el sistema en sistemas operativos Windows y GNU/Linux. Se recomienda GNU/Linux. .
RNF4	Soporte	1-La estación de trabajo del cliente debe tener instalado un navegador web. Se recomienda utilizar Mozilla Firefox 60.X o superior
RNF5	Legales	1-Las herramientas seleccionadas para la implementación de los requerimientos de seguridad están respaldadas por licencias libres, bajo condiciones de software libre.

Tabla 3 Requerimientos no funcionales

2.3 Diagrama de caso de uso

Un caso de uso se define como un conjunto de acciones realizadas por el sistema que dan lugar a un resultado observable. Especifica un comportamiento que el sujeto puede realizar en colaboración con uno o más actores, pero sin hacer referencia a su estructura interna (30). Por tanto los casos de uso son los principales medios para capturar la funcionalidad del sistema desde la perspectiva del usuario así como especificar la relación que existe entre el mismo y los usuarios y/u otros sistemas.

Los diagramas de caso de uso son uno de los cinco tipos de diagramas en UML para modelar aspectos dinámicos de sistemas (diagramas de actividad, diagramas de estados, diagramas de secuencia y diagramas de colaboración son otros cuatro tipos de diagramas en UML para modelar los aspectos

dinámicos de un sistema). Son importantes para modelar el comportamiento de un sistema, un subsistema o una clase. Cada uno muestra un conjunto de casos de uso, actores y sus relaciones.
Estos diagramas se encuentran compuesto principalmente por dos tipos de nodos:
Actor: representa cualquier elemento que intercambia información con el sistema.
Caso de uso: es una secuencia de intercambios en diálogo con el sistema que se encuentra relacionadas por su comportamiento.
Los arcos entre los actores y los casos de uso se denominan arcos de comunicación, los cuales son una forma de representar la interrelación que existen entre los elementos del diagrama.

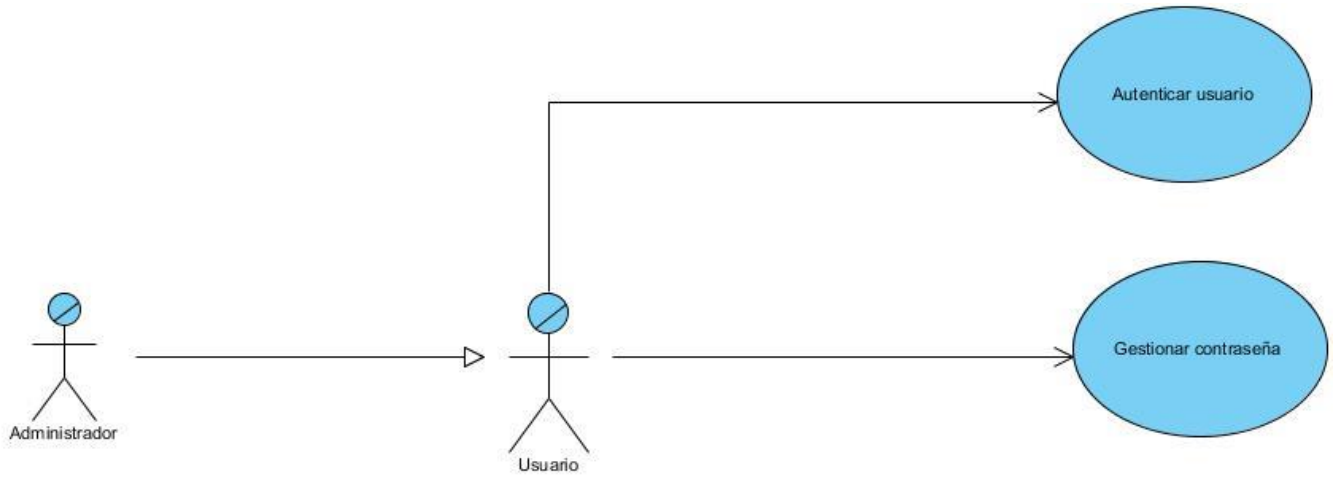


Figura 6 Diagrama de Casos de Uso

2.3.1 Descripción de los actores

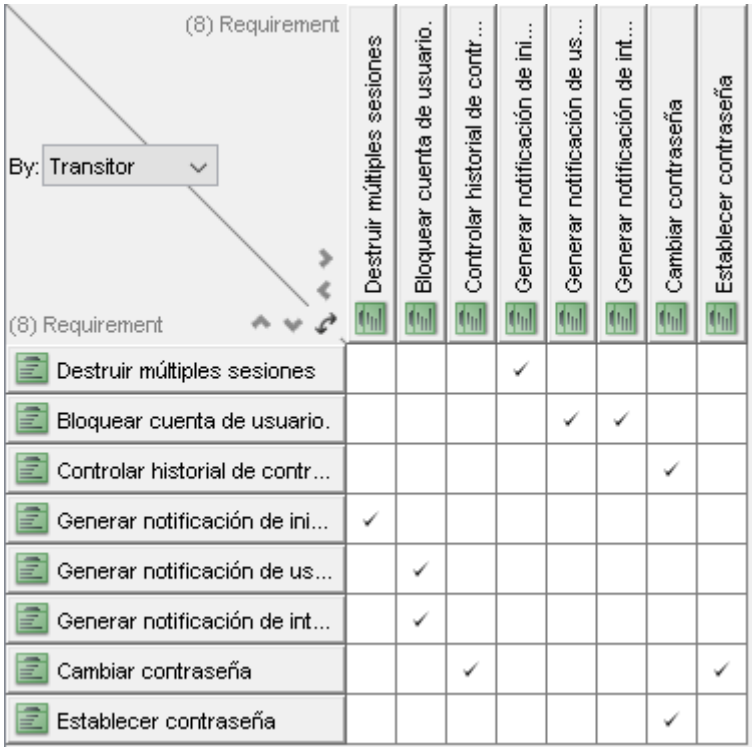
Actor	Objetivo
Usuario	Realizar las actividades dirigidas a autenticarse en el sistema, gestionar su contraseña, modificar su perfil y cerrar su sesión.
Administrador	Realizar las actividades dirigidas a administrar el sistema XABAL Arkheia 4.0.

Tabla 4 Descripción de los actores.

2.4 Matriz de trazabilidad

Relaciona dos elementos esenciales para la buena ejecución de las labores de un proyecto: los requisitos establecidos para dicha ejecución y el valor que cada uno de ellos agrega al conjunto del

proceso. Es una herramienta clave para la ingeniería de los proyectos, así como para el seguimiento de los diversos elementos que los componen (31).



	Destruir múltiples sesiones	Bloquear cuenta de usuario.	Controlar historial de contr...	Generar notificación de ini...	Generar notificación de us...	Generar notificación de int...	Cambiar contraseña	Establecer contraseña
Destruir múltiples sesiones				✓				
Bloquear cuenta de usuario.					✓	✓		
Controlar historial de contr...							✓	
Generar notificación de ini...	✓							
Generar notificación de us...		✓						
Generar notificación de int...		✓						
Cambiar contraseña			✓					✓
Establecer contraseña							✓	

Figura 7 Matriz de trazabilidad requisito-requisito.

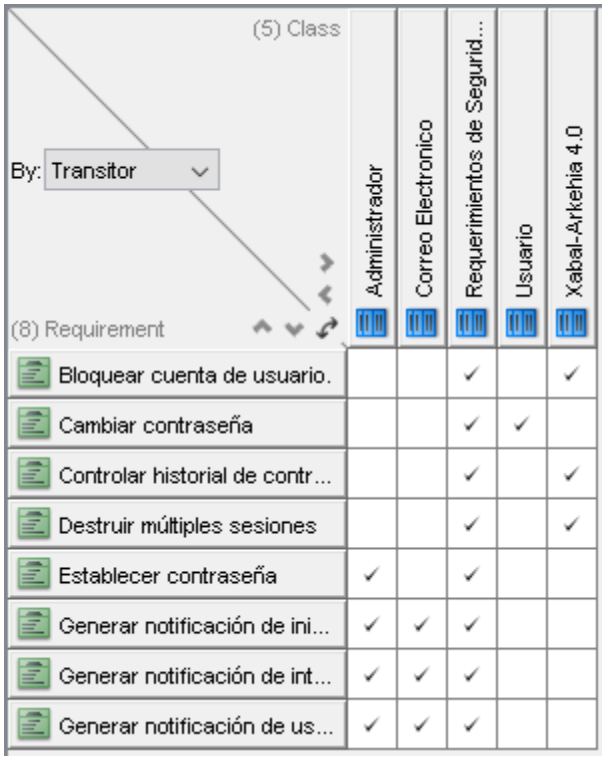


Figura 8 Matriz de trazabilidad modelo conceptual-requisito.

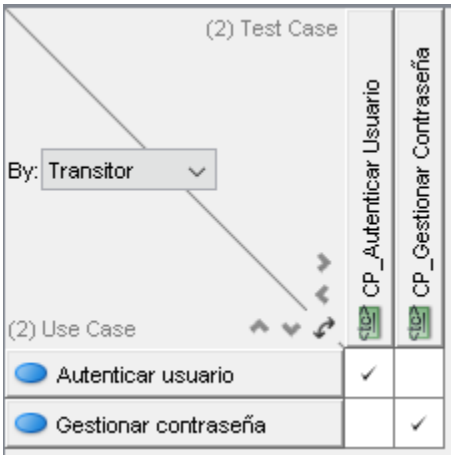


Figura 9 Matriz de trazabilidad caso de uso _ caso de prueba.

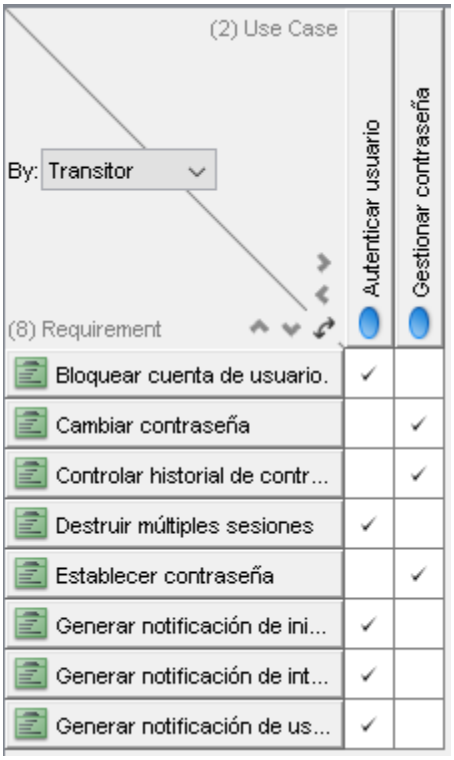


Figura 10 Matriz de trazabilidad casos de uso- requisitos.

Se muestran 4 matrices de trazabilidad se especifica cada una a continuación en el orden en que aparecen las figuras:

- Matriz de trazabilidad requisito-requisito la cual muestra la dependencia que existe entre cada uno de los requisitos.
- Matriz de trazabilidad requisito-modelo de dominio la cual muestra la relación que existe entre los requisitos y las principales clases representadas en el modelo conceptual.
- Matriz de trazabilidad caso de uso-caso de prueba la cual muestra como cada caso de uso está relacionado con el caso de prueba que le corresponde.
- Matriz de trazabilidad requisito-caso de uso la cual muestra como cada requisito está agrupado en el caso de uso correspondiente.

2.5 Especificación de los Casos de Uso

CU 1 Autenticar usuario

Objetivo	Usuario: (Inicia) Se autentica en el sistema
Actor	Usuario

Resumen	El usuario introduce los datos para autenticarse, se verifica que los datos introducidos son correctos y que el usuario cuenta con los permisos necesarios para autenticarse en el sistema. En caso contrario no se permite la autenticación del usuario y se genera una notificación al administrador del sistema a través del correo electrónico.
Complejidad	Media
Prioridad	Alta
Precondiciones	No aplica
Postcondiciones	Usuario autenticado en el sistema.

Flujo de eventos

Flujo básico Autenticar usuario

	Actor	**Sistema**
		Muestra una interfaz solicitando los siguientes datos: • Nombre de usuario. • Contraseña. Muestra la siguiente opción: - Acceder.
1.	Introduce los datos solicitados.	
2.	Selecciona la opción “Acceder”.	
3.		Comprueba que los campos obligatorios no estén vacíos.
4.		Comprueba en la Base de Datos (BD) que el usuario corresponde con un usuario del sistema y verifica que la contraseña pertenezca a ese usuario.
5.		Comprueba en la BD que el usuario está activado.

6.		Comprueba en la BD que la contraseña del usuario no haya expirado.
7.		Comprueba en la BD que no exista otra sesión abierta para ese usuario en la aplicación usando el mismo usuario y contraseña.
8.		Almacena en la BD la fecha y la hora en la que inició sesión y el código del usuario autenticado.
9.		Carga la configuración del usuario autenticado según el rol que tenga en el sistema, usando el código del usuario.
10.		Muestra en el menú principal las opciones a las que tiene acceso el usuario autenticado.
11.		Termina el caso de uso.
Flujos alternos		
3a No se han introducido todos los datos obligatorios		
	Actor	**Sistema**
1.		Comprueba que los campos usuario y/o contraseña estén vacíos.
2.		Señala los campos que deben ser llenados y muestra un mensaje indicando que faltan datos por introducir. Ir a la acción 1.
4a Los datos introducidos son incorrectos		
	Actor	**Sistema**
1.		Comprueba que los datos introducidos no estén correctos.
2.		Muestra un mensaje indicando que existen datos incorrectos, y señala los datos que están incorrectos.

		Ir a la acción 1. Dado tres intentos de inicio de sesión con datos incorrectos, el sistema activará un mecanismo de bloqueo de usuarios y generará una notificación al administrador del sistema al igual que si existe una repetición de acceso al sistema por parte de un usuario inexistente desde una misma dirección IP.
5a Usuario desactivado		
	Actor	**Sistema**
		Comprueba en la BD que el usuario está desactivado.
1.		Muestra un mensaje indicando que su cuenta ha sido desactivada. Ir a la acción 1.
6a Contraseña expirada		
	Actor	**Sistema**
		Comprueba en la BD que la contraseña del usuario haya expirado.
2.		Ir a la acción 1.
7a Usuario autenticado en otra sesión		
	Actor	**Sistema**
		Comprueba en la BD que existe otra sesión abierta para ese usuario en la aplicación.
3.		Muestra un mensaje indicando que el usuario ya está autenticado en la aplicación. Genera una notificación al administrador después de un intento de inicio de sesión con usuario que posee sesión activa en el servidor.

		Ir a la acción 1.
Relaciones	**CU incluidos**	No aplica.
	CU extendidos	No aplica.
Requisitos no funcionales	No aplica.	
Asuntos pendientes	No aplica.	

Tabla 5 CU1 Autenticar usuario.

CU 2 Gestionar contraseña

Objetivo	Establecer y cambiar la contraseña de un usuario.	
Actores	Administrador: (Inicia) Establece contraseñas en el sistema. Usuario: Cambia la contraseña en el sistema.	
Resumen	El actor solicita realizar una acción sobre la contraseña de un usuario seleccionado, el sistema muestra el formulario que permite realizar la acción solicitada. Las acciones son: establecer y cambiar la contraseña de un usuario. El actor introduce los datos especificados. Se almacenan los datos en la BD.	
Complejidad	Media.	
Prioridad	Media.	
Precondiciones	Debe estar mostrado el listado de los usuarios. Debe existir un usuario seleccionado. El usuario debe estar activado.	
Postcondiciones	Contraseña actualizada.	
Flujo de eventos		
Flujo básico Establecer contraseña		
	Actor	**Sistema**

1.	Selecciona crear un nuevo usuario en las herramientas de administración que tiene el sistema	
2.		Permite realizar las siguientes acciones sobre una contraseña: - Establecer contraseña. - Cambiar contraseña. *Ver Sección 1: Cambiar contraseña.*
3.	Selecciona la opción “Establecer contraseña”.	
4.		Muestra un formulario con los siguientes datos: • Contraseña. • Confirmar contraseña. Muestra las siguientes opciones: -Aceptar. -Cancelar.
5.	Introduce los datos del usuario.	
6.	Selecciona la opción “Aceptar”.	
7.		Comprueba que se han introducido todos los datos obligatorios, teniendo en cuenta que son todos y que la contraseña cumpla con los requisitos de longitud mínimo de 8 caracteres
8.		Comprueba que los campos ¨Contraseña¨ y ¨Confirmar contraseña¨ son iguales.
9.		Codifica la contraseña utilizando un algoritmo de encriptación.
10.		Almacena en la BD los datos de la contraseña.
11.		Muestra un mensaje de confirmación indicando que la contraseña ha sido establecida satisfactoriamente.

12.		Termina el caso de uso.
Flujos alternos		
5*a Opción "Cancelar"		
	Actor	**Sistema**
1.	Selecciona la opción "Cancelar".	
2.		Retorna a la página que le dio origen.
7a No se han introducido todos los datos obligatorios		
	Actor	**Sistema**
1.		Comprueba que no se han introducido los campos obligatorios.
2.		Señala los campos que deben ser llenados y muestra un mensaje indicando que existen errores en el formulario. Ir a la acción 4.
8a Contraseñas incorrectas		
	Actor	**Sistema**
1.		Comprueba que los campos ¨Contraseña¨ y ¨Confirmar contraseña¨ no son iguales.
2.		Señala los campos incorrectos y muestra un mensaje indicando que las contraseñas no coinciden. Ir a la acción 4.
Sección 1: "Cambiar contraseña**"**		
Flujo básico Cambiar contraseña		
	Actor	**Sistema**
1.	Selecciona la opción "Cambiar contraseña".	

2.		Muestra un formulario con los siguientes datos: • Contraseña anterior. • Nueva contraseña. • Confirmar contraseña. Muestra las siguientes opciones: Aceptar. Cancelar.
3.	Introduce los datos solicitados.	
4.	Selecciona la opción “Aceptar”.	
5.		Comprueba que se han introducido todos los datos obligatorios, teniendo en cuenta que son todos y que la contraseña posee los requisitos de longitud mínimo de 8 caracteres
6.		Comprueba que la Contraseña anterior coincide con la contraseña del usuario autenticado.
7.		Comprueba que los campos ¨Nueva contraseña¨ y ¨Confirmar contraseña¨ son iguales.
8.		Codifica la contraseña utilizando un algoritmo de encriptación.
9.		Actualiza en la BD los datos de la contraseña.
10.		Muestra un mensaje de confirmación indicando que la contraseña ha sido cambiada satisfactoriamente.
11.		Termina el caso de uso.
Flujos alternos		
3*a Opción “Cancelar”		
	Actor	**Sistema**
3.	Selecciona la opción “Cancelar”.	

4.		Retorna a la página que le dio origen.
5a No se han introducido todos los datos obligatorios		
	Actor	**Sistema**
1.		Comprueba que no se han introducido los campos obligatorios.
2.		Señala los campos que deben ser llenados y muestra un mensaje indicando que existen errores en el formulario. Ir a la acción 2.
6a Contraseña anterior incorrecta		
	Actor	**Sistema**
1.		Comprueba que la Contraseña anterior no coincide con la contraseña del usuario autenticado.
2.		Señala el campo incorrecto y muestra un mensaje indicando que la contraseña anterior está incorrecta. Ir a la acción 2.
7a Contraseñas incorrectas		
	Actor	**Sistema**
3.		Comprueba que los campos ¨Nueva contraseña¨ y ¨Confirmar contraseña¨ no son iguales.
4.		Señala los campos incorrectos y muestra un mensaje indicando que las contraseñas no coinciden. Ir a la acción 2.
Relaciones	**CU incluidos**	No aplica.

	CU extendidos	No aplica.
Requisitos no funcionales	No aplica.	
Asuntos pendientes	No aplica.	

Tabla 6 CU2 Gestionar contraseña

2.6 Descripción de la arquitectura

La Arquitectura de Software se refiere a las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos (32).
Para la implementación de los requerimientos de seguridad autenticación y generación de notificaciones se realizó un estudio sobre la arquitectura en la que está basado Alfresco 5.2 teniendo como resultado que se basa en la arquitectura en capas.

El Patrón de arquitectura por capas es una de las técnicas más comunes que los arquitectos de software utilizan para dividir sistemas de software complicados. Al pensar en un sistema en términos de capas, se imaginan los principales subsistemas de software ubicados de la misma forma que las capas de un pastel, donde cada capa descansa sobre la inferior. En este esquema la capa más alta utiliza varios servicios definidos por la inferior, pero la última es inconsciente de la superior. Además, normalmente cada capa oculta las capas inferiores de las siguientes superiores a esta.
Los beneficios de trabajar un sistema en capas son (33):

- Se puede entender una capa como un todo, sin considerar las otras.
- Las capas se pueden sustituir con implementaciones alternativas de los mismos servicios básicos.
- Se minimizan las dependencias entre capas.
- Las capas posibilitan la estandarización de servicios.

Capa de Presentación

Referente a la interacción entre el usuario y el software. Puede ser tan simple como un menú basado en líneas de comando o tan complejo como una aplicación basada en formas. Su principal responsabilidad es mostrar información al usuario, interpretar los comandos de este y realizar algunas validaciones simples de los datos ingresados (33). Logra una correcta interacción entre el par usuario-aplicación a

través del uso del framework YUI para lograr una interfaz ricamente interactiva, también utiliza la librería Aikau para el uso de widgets y temas. Se pueden ver de manifiesto en la implementación de los requerimientos de seguridad en los formularios con el uso de los componentes Form-runtime así como las peticiones Ajax lo cual permite representar visualmente toda la información necesaria consultada y/o generada por la interacción entre el usuario y la aplicación.

Capa de Reglas de Negocio

También denominada Lógica de Dominio, esta capa contiene la funcionalidad que implementa la aplicación. Involucra cálculos basados en la información dada por el usuario y datos almacenados y validaciones. Controla la ejecución de la capa de acceso a datos y servicios externos. Se puede diseñar la lógica de la capa de negocios para uso directo por parte de componentes de presentación o su encapsulamiento como servicio y llamada a través de una interfaz de servicios que coordina la conversación con los clientes del servicio o invoca cualquier flujo o componente de negocio (33). Se pone de manifiesto en los servicios de autenticación a los cuales se les integraron los requerimientos de seguridad y también tiene la posibilidad de comunicarse con servicios externos en este caso el correo para notificar al administrador ante fallos en la autenticación.

Capa de Datos

Esta capa contiene la lógica de comunicación con otros sistemas que llevan a cabo tareas por la aplicación. Estos pueden ser monitores transaccionales, otras aplicaciones, sistemas de mensajerías, etc. Para el caso de aplicaciones empresariales, generalmente está representado por una base de datos, que es responsable por el almacenamiento persistente de información. Esta capa debe abstraer completamente a las capas superiores (negocio) del dialecto utilizado para comunicarse con los repositorios de datos (33). Esta capa se pone de manifiesto en los modelos de contenidos y servicios de datos.

2.6.2 Patrones de diseño

Los patrones de diseño son soluciones para problemas típicos y recurrentes que enfrentan los programadores al desarrollar una aplicación. Son soluciones probadas y documentadas que explican cómo resolver un determinado problema bajo determinadas circunstancias. Por lo que constituyen una guía para el rápido desarrollo de software. Por tanto los patrones ayudan a estandarizar el código, haciendo que el diseño sea más comprensible para otros.

Patrón de diseño GRASP

Describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (34) (35). De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento). Los patrones analizados en esta sección son: Experto, Creador, Alta Cohesión, Bajo Acoplamiento y Controlador.

Experto

El experto en información establece el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo (35). Por tanto el uso de este patrón tiene como resultado que los objetos tengan la posibilidad de realizar las acciones implementadas utilizando su propia información. Se puede evidencia en la funcionalidad enviar correo ya que en esta se encuentran las variables con los valores necesarios para realizar la acción de enviarlo al administrador.

Creador

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental es encontrar un creador que se debe conectar con el objeto producido en cualquier evento (35).

Alta Cohesión

La alta cohesión indica que los datos y responsabilidades de una entidad están fuertemente ligados a la misma en un sentido lógico. La información que maneja una entidad de software tiene que estar conectada lógicamente con esta, no debe existir entidades con atributos que describan comportamientos que en realidad no le corresponden (35). Por tanto el uso de este patrón garantiza que se guarde una alta relación entre sus funcionalidades, manteniendo el enfoque a su único propósito.

Bajo acoplamiento

El bajo acoplamiento sostiene la idea de mantener las entidades y clases lo menos ligadas entre sí posible, de tal forma que en caso de producirse modificaciones en alguna de ellas, la repercusión sea la mínima posible en el resto de las entidades o clases (35). Por tanto el acoplamiento en términos de software, se refiere a la relación que se guardan entre los módulos de un sistema y la dependencia entre ellos. Este patrón se evidencia en el archivo notificaciones.js donde existe una función para detectar si

el usuario que intenta acceder es inexistente y otra para mandar el correo al administrador, ya que una modificación en este proceso de detectar al usuario no repercute sobre la acción de enviar el correo.

Controlador

El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado (34).

Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control (34).

Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento. Se manifiesta en el fichero login.js que se encarga de controlar todas las acciones que se realizan en la autenticación.

Conclusiones del capítulo

En este capítulo se realizó el análisis de la propuesta de solución, para ello se elaboró un modelo de dominio donde se mostraron los principales conceptos y la relación que existe entre ellos, se presentaron los requisitos funcionales y no funcionales, se definieron los casos de usos, así como su diagrama y descripción. Lo anteriormente planteado permitió un mejor entendimiento del funcionamiento del negocio, posibilitando la implementación de los requerimientos de seguridad en la autenticación y generación de alarmas en el sistema informático XABAL Arkheia 4.0.

CAPÍTULO 3. IMPLEMENTACIÓN Y COMPROBACIÓN DE LA PROPUESTA DE SOLUCIÓN

Introducción

En el presente capítulo se describe la implementación de los requerimientos de seguridad en la autenticación y generación de notificaciones así como los resultados de la integración de dichos requerimientos en el sistema informático. Se traza un plan de pruebas para comprobar el funcionamiento de la implementación y garantizar que el mismo sea correcto para finalmente mostrar los resultados de estas pruebas.

3.1 Implementación

La disciplina de implementación que propone la metodología AUP (UCI) se define en cómo después de haberse realizado el Análisis y Diseño se obtiene toda la información para dar paso a la modificación del sistema.

Con estos resultados fue realizada la implementación de los requerimientos de seguridad en la autenticación y generación de notificaciones a través de la utilización de archivos contenedores del código como JavaScript, HTML, XML, así como plantillas FTL. Con el estudio de estos requerimientos en la aplicación existente se logró integrarlos a la versión XABAL Arkheia 4.0 a pesar de estar basada en otra tecnología.

El resultado fue un sistema que cuenta con mecanismos para lograr una mejora en la seguridad de la solución informática durante la autenticación, para mitigar los ataques de fuerza bruta se implementó un mecanismo para el bloqueo de usuario así como informar mediante el servicio de correo al administrador tras existir cualquier error producido en el proceso de autenticarse un usuario en el sistema.

3.2 Diagrama de despliegue

Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos (36).

Nodo: es un elemento de hardware o software. Esto se muestra con la forma de una caja en tres dimensiones.

Nodo como contenedor: un nodo puede contener otros elementos, como componentes o artefactos.

Asociación: representa una ruta de comunicación entre los nodos.

Se pueden ver representados todos estos elementos en la siguiente figura

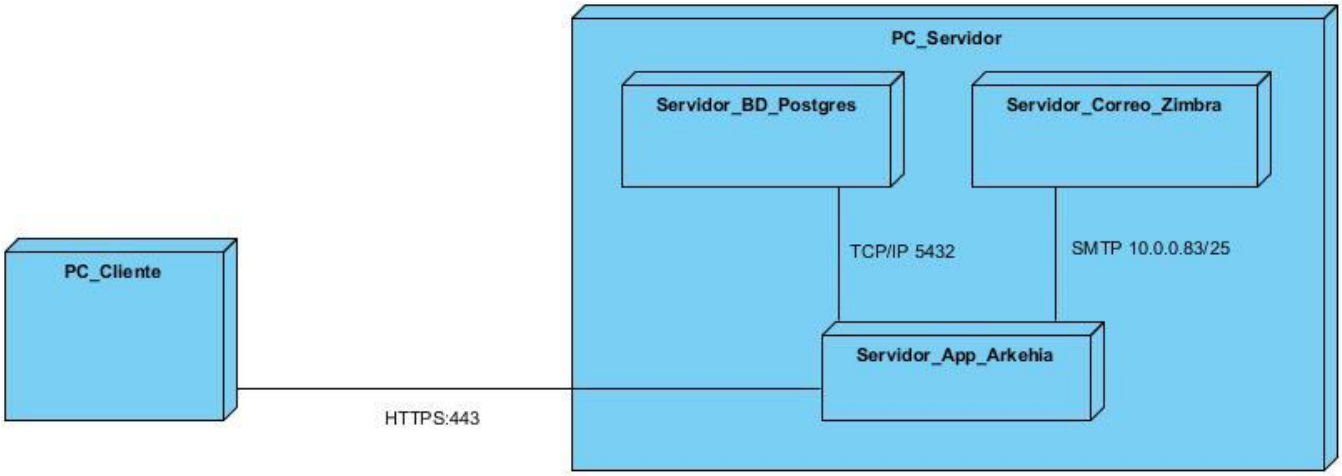


Figura 11 Diagrama de despliegue

Como se muestra en la figura, la distribución física del sistema en tiempo de ejecución consta de cinco nodos entre ellos un nodo contenedor. En uno se encuentra la PC cliente, que debe tener instalado un navegador web, el cual hace peticiones a la PC servidor donde se encuentra la aplicación XABAL Arkheia 4.0 usando el protocolo de comunicación HTTPS por el puerto 443. El sistema establece comunicación con el servidor de base dato PostrgresSQL utilizando el protocolo TCP/IP por el puerto 5432 a la hora de realizar la autenticación ya que la misma almacena datos de usuario como la contraseña y el nombre de usuario de esta forma se define la asociación entre la PC servidor con la base de dato y con la aplicación. En la PC Servidor como nodo contenedor se tiene la base de datos, la aplicación, y la conexión con el servicio de correo electrónico, al cual accede por SMTP por el host 10.0.0.83 y puerto 25, pero de forma separada cada servicio es independiente, pero en la misma PC.

3.3 Pruebas de software

Las pruebas de software comprenden el conjunto de actividades que se realizan para identificarse posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación, por medio de pruebas sobre el comportamiento del mismo (37).

Los sistemas informáticos, programas y aplicaciones han crecido a niveles inimaginables en complejidad e interoperabilidad, con lo cual también se han incrementado las posibilidades de defectos, a simple vista insignificantes, pero que pudieran adquirir proporciones catastróficas (37).

Por tanto se basan en plantearse una estrategia de prueba con el fin de reducir al mínimo las posibilidades de errores de software.

3.3.1 Estrategia de pruebas

Una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requerirán. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados. Una estrategia de prueba de software debe ser suficientemente flexible para promover un uso personalizado de la prueba. Al mismo tiempo, debe ser suficientemente rígida para alentar la planificación razonable y el seguimiento de la gestión conforme avanza el proyecto (32).Por tanto la estrategia de prueba permite definir las técnicas y criterios a tener en cuenta para realizar las pruebas al componente desarrollado.

La estrategia de prueba define:

- Técnicas de pruebas (manual o automática) y herramientas a ser usadas.
- Qué criterios de éxitos y culminación de la prueba serán usados.
- Consideraciones especiales afectadas por requerimientos de recursos o que tengan implicaciones en la planificación.

3.3.2 Niveles de pruebas

Las pruebas se aplican a diferentes tipos de destinos, en fases o niveles diferentes de esfuerzo de trabajo. Estos niveles suelen distinguirlos los roles que están más capacitados para diseñar y dirigir las pruebas, donde las técnicas son más adecuadas para realizar la prueba en cada nivel. Es importante asegurarse de hay un equilibrio entre los diferentes esfuerzos de trabajo (38). Los niveles de prueba que existen son:

- Prueba del desarrollador.
- Prueba independiente.
- Prueba de unidad.
- Prueba de regresión.
- Prueba de integración.
- Prueba del sistema.
- Prueba de aceptación.

Se tendrán en cuenta para la presente investigación los siguientes niveles de prueba:

Prueba del desarrollador

En este nivel se utilizará el método de caja blanca para examinar cada una de las posibles entradas y determinar el flujo de ejecución con el fin de verificar que los valores de salida sean correctos.

Prueba de regresión

Se utilizan para averiguar si una aplicación existente todavía funciona como se esperaba después de haber sido actualizada o modificada. Es vital llevar a cabo tales pruebas cada vez que el código ha cambiado (39). Por tanto se realizan para verificar que una vez se hayan detectado errores después que sean corregidos no se afecte la calidad del sistema.

Pruebas realizadas según la metodología de desarrollo:

Prueba interna

En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas (14). En esta se generan distintos artefactos en el caso de esta investigación serán los casos de prueba.

Prueba de liberación

Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación (14).

Prueba de aceptación

Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (14). Se hizo entrega del sistema modificado al cliente, y este realizó un acta de aceptación confirmando que cumple con el objetivo previsto.

3.3.3 Tipos de pruebas

Un grupo de actividades de pruebas pueden tener por objeto comprobar el sistema en base a un motivo u objetivo específico.

Un tipo de prueba se centra en un objetivo de prueba en particular, que puede ser cualquiera de los siguientes (40):

- Una función a realizar por el software.
- Una característica de calidad no funcional, tales como la fiabilidad o la usabilidad.
- La estructura o arquitectura del software o sistema.

Pruebas funcionales

Las pruebas funcionales se basan en funciones y prestaciones descritas en documentos o entendidas por los colaboradores y en su interoperabilidad con sistemas específicos, pueden llevarse a cabo en todos los niveles de prueba (40).

Las técnicas basadas en la especificación sirven para obtener condiciones de prueba y casos de prueba a partir de la funcionalidad de un software o sistema. Tienen en cuenta el comportamiento externo del software (40).

Pruebas no funcionales

Las pruebas no funcionales incluyen, pero sin limitarse a ello, pruebas de rendimiento, pruebas de carga, pruebas de estrés, pruebas de usabilidad, pruebas de mantenibilidad, pruebas de fiabilidad y pruebas de portabilidad. Estas pruebas se refieren a cómo funciona el sistema (40).

Pueden ejecutarse en todos los niveles de prueba ya que son las pruebas necesarias para medir las características de los sistemas y software que pueden cuantificarse según una escala variable, tales como los tiempos de respuesta en el caso de las pruebas de rendimiento (40). Por tanto las pruebas no funcionales tienen en cuenta el comportamiento externo del software y en la mayoría de los casos lo hacen mediante técnicas de diseño de pruebas de caja negra.

Pruebas de estructura

Son las más idóneas, después de las técnicas basadas en la especificación, para ayudar a medir la exhaustividad de las pruebas mediante una evaluación de la cobertura de un tipo de estructura (40).

Pueden ejecutarse en todos los niveles de prueba se basan en la arquitectura del sistema, como por ejemplo la jerarquía de llamadas (40).

Pruebas asociadas a cambios

Una vez detectado y corregido un defecto, el software debe volver a probarse para confirmar que el defecto original ha sido corregido con éxito. A esto se le denomina confirmación. La depuración (localizar y corregir defectos) es una actividad de desarrollo, no una actividad de pruebas. Las pruebas de regresión son pruebas reiteradas de un programa ya probado, después de haber sido modificado, con vistas a localizar defectos surgidos o no descubiertos como resultado del cambio o de los cambios. Estos defectos pueden estar en el software objeto de las pruebas, o en cualquier otro componente de software asociado o no asociado. Se realizan cuando el software, o su entorno, sufren modificaciones. El alcance de las pruebas de regresión depende del riesgo de no encontrar defectos en el software que antes funcionaba (40).

Después de haber hecho un estudio de los diferentes tipos de pruebas según el International Software Testing Qualifications Board que fue la referencia bibliográfica utilizada se decidió que las pruebas que serían aplicadas son de tipo funcionales por lo descrito anteriormente.

3.3.4 Métodos de pruebas

Son las técnicas aplicadas para definir una estrategia en función de la validación y verificación del sistema diseñado para descubrir fallos. En el caso de esta investigación se describirá el proceso que se llevó a cabo para obtener los resultados de los métodos de prueba caja blanca y caja negra.

Prueba de Caja negra

Las pruebas de caja negra, también llamadas pruebas de comportamiento, se enfocan en los requerimientos funcionales del software; es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa. Las pruebas de caja negra no son una alternativa para las técnicas de caja blanca. En vez de ello, es un enfoque complementario que es probable que descubra una clase de errores diferente que los métodos de caja blanca. Las pruebas de caja negra intentan encontrar errores en las categorías siguientes: funciones incorrectas o faltantes (32).

- Errores de interfaz.
- Errores en las estructuras de datos o en el acceso a bases de datos externas.
- Errores de comportamiento o rendimiento.
- Errores de inicialización y terminación.

Técnicas de prueba de caja negra:

- Partición de equivalencias
- Análisis de valores borde
- Tablas de decisión
- Transición entre estados
- Pruebas de casos de uso

La partición de equivalencia es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos de los que pueden derivarse casos de prueba. Un caso de prueba ideal descubre de primera mano una clase de errores (por ejemplo, procesamiento incorrecto de todos los datos carácter) que de otro modo podrían requerir la ejecución de muchos casos de prueba antes de observar el error general. El diseño de casos de prueba para la partición de equivalencia se basa en una evaluación de las clases de equivalencia para una condición de entrada. Con los conceptos introducidos

en la sección precedente, si un conjunto de objetos puede vincularse mediante relaciones que son simétricas, transitivas y reflexivas, se presenta una clase de equivalencia. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Por lo general, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana (32).

Prueba de Caja blanca

La prueba de caja blanca, en ocasiones llamada prueba de caja de vidrio, es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba. Al usar los métodos de prueba de caja blanca, puede derivar casos de prueba que (32):

- Garanticen que todas las rutas independientes dentro de un módulo se revisaron al menos una vez.
- Revisen todas las decisiones lógicas en sus lados verdadero y falso.
- Ejecuten todos los bucles en sus fronteras y dentro de sus fronteras operativas.
- Revisen estructuras de datos internas para garantizar su validez.

Prueba de ruta básica

La prueba de ruta o trayectoria básica es una técnica de prueba de caja blanca propuesta por primera vez por Tom McCabe. El método de ruta básica permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño de procedimiento y usar esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para revisar el conjunto básico tienen garantía para ejecutar todo enunciado en el programa, al menos una vez durante la prueba. Primeramente se debe denotar el grafo de flujo (32).

Notación del grafo de flujo

- **Nodo del grafo de flujo(N):** representa uno o más enunciados de procedimientos.
- **Aristas(A):** representan el flujo de control y son análogas a las flechas.
- **Regiones:** representan las áreas acotadas por los nodos y las aristas.

Una vez realizada la notación del grafo de flujo se procede a calcular la complejidad ciclomática.

Complejidad ciclomática

Es una métrica que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa. Esto

indica el límite superior para el número de pruebas que se deben realizar, para asegurar que se ejecuta cada sentencia al menos una vez. La complejidad ciclomática tiene fundamentos en la teoría de gráficos y proporciona una medición de software extremadamente útil. La complejidad se calcula en una de tres formas (32):

- El número de regiones del gráfico de flujo corresponde a la complejidad ciclomática.
- La complejidad ciclomática V (G) para un gráfico de flujo G se define como V (G) = A–N + 2 donde A es el número de aristas del gráfico de flujo y N el número de nodos del gráfico de flujo.
- La complejidad ciclomática V (G) para un gráfico de flujo G también se define como V (G) = P + 1 donde P es el número de nodos predicado contenidos en el gráfico de flujo G.

-Luego se determina un conjunto básico de caminos linealmente independientes: el valor de V (G) es el número de caminos linealmente independientes de la estructura de control del programa.

-Obtención de casos de prueba: se realizan los casos de pruebas que forzarán la ejecución de cada camino del conjunto básico Los datos deben elegirse de modo que las condiciones en los nodos predicados se establezcan de manera adecuada conforme se prueba cada ruta. Cada caso de prueba se ejecuta y compara con los resultados esperados. Una vez completados todos los casos de prueba, el examinador puede estar seguro de que todos los enunciados del programa se ejecutaron al menos una vez.

3.3.5 Casos de pruebas

Son un conjunto de valores de entrada, precondiciones de ejecución, resultados esperados y Postcondiciones de ejecución, desarrollados para un objetivo particular de condición de prueba, tal como para ejercer una ruta de un programa en particular o para verificar el cumplimiento de un requisito específico (41).

Los casos de prueba son sin duda uno de los elementos más importantes cuando a pruebas de software se refiere, dado que son un elemento que día a día le dan más valor al software, tanto así que entre más casos y de mayor calidad se posean casos el software va a adquirir más valor, adicionalmente cabe mencionar que un caso de prueba de ejecución manual bien elaborado es insumo clave en un proceso de automatización de pruebas, también se vuelven un valor agregado que apoya a los procesos de formación dentro de un área de calidad al ser fuente de información explicita del negocio. Los indicadores de resultados y la forma en que se puede medir el desempeño de un equipo de pruebas de software se mide en base a casos de pruebas (41).

A continuación se describen los casos de pruebas para cada caso de uso, especificando el escenario, la descripción, observaciones, la respuesta del sistema y el flujo central, así como los resultados obtenidos una vez ejecutado el caso de prueba y las condiciones que deben cumplirse para que este se ejecute. La técnica utilizada para mostrar la información en las tablas fue particiones equivalentes del método de prueba de caja negra.

CP1_Autenticar usuario

Escenario	Descripción	Observaciones	Respuesta del sistema	Flujo central
EC1.1.Autenticar usuario.	El escenario de prueba permite autenticar al usuario al ingresar los datos en el formulario.	N/A	El sistema entra a la página de inicio.	1-Se introducen los datos correctamente. 2-Se presiona el botón Registrarse. 3-Se muestra la página de inicio dentro del sistema.
EC1.2.Campos obligatorios vacíos.	El escenario de prueba permite al usuario validar los campos vacíos.	Valor(vacío)	El sistema señala los campos obligatorios vacíos y muestra el mensaje: "Valores de autenticación incorrectos."	1-Se introducen los datos correctamente. 2-Se presiona el botón Registrarse. 3-Se muestra la página de inicio dentro del sistema.

EC1.3 Validar campos incorrectos.	El escenario de prueba permite al usuario validar los campos del formulario.	Valor (cualquier nombre de usuario o contraseña incorrecto).	El sistema señala los campos con valores incorrectos y muestra el mensaje: "Valores de autenticación incorrectos."	1-Se introducen los datos correctamente. 2-Se presiona el botón Registrarse. 3-Se muestra la página de inicio dentro del sistema.

Tabla 7 Caso de Prueba Autenticar usuario

CP2_Gestionar contraseña

Escenario	Descripción	Observaciones	Respuesta del sistema	Flujo central
EC2.1.Establecer contraseña.	El escenario de prueba permite al administrador establecer de contraseña.	N/A	El sistema establece una contraseña al usuario en cuestión.	1-Selecciona la opción crear usuario 2-Se introducen los datos correctamente. 3-Se presiona el botón aceptar.
EC2.2.Cambiar contraseña.	El escenario de prueba permite al usuario cambiar de contraseña.	N/A	El sistema establece una contraseña al usuario en cuestión.	1-Selecciona la opción crear usuario 2-Se introducen los datos correctamente. 3-Se presiona el botón aceptar.

EC2.3.Campos obligatorios vacíos.	El escenario de prueba permite al usuario validar los campos vacíos.	Valor(vacío)	El sistema señala los campos obligatorios vacíos y muestra el mensaje: "Rellene este campo."	1-Selecciona la opción crear usuario 2-Se introducen los datos correctamente. 3-Se presiona el botón aceptar.
EC2.4.Validar campos incorrectos.	El escenario de prueba permite al usuario validar los campos del formulario.	Valor(menor a 8 caracteres)	El sistema señala los campos obligatorios vacíos y muestra el mensaje: "El campo debe tener al menos 8 caracteres."	1-Selecciona la opción crear usuario 2-Se introducen los datos correctamente. 3-Se presiona el botón aceptar. .
EC2.5.Cancelar acción.	El escenario de prueba permite al usuario autenticado cancelar la acción.	N/A	El sistema vuelve a la página de inicio.	1-Selecciona la opción crear usuario 2-Se introducen los datos correctamente. 3-Se presiona el botón cancelar.

Tabla 8 Caso de Prueba Gestionar contraseña

3.3.6 Resultados de las pruebas realizadas

Según la norma ISO 9000:2005 una No Conformidad es un incumplimiento de un requisito del sistema, sea este especificado o no (42).

Pueden ser de dos tipos:

• **No conformidad mayor:** ausencia o fallo en implantar y mantener uno o más requisitos del sistema de gestión de la calidad, o una situación que pudiera, basándose en evidencias o evaluaciones objetivas,

crear una duda razonable sobre la calidad de lo que la organización está suministrando. Las entidades certificadoras no pueden conceder el certificado mientras exista una no conformidad mayor (42).

• **No conformidad menor (o solamente no conformidad):** es una no conformidad detectada, que por sus características no llega a la gravedad de la anterior (42).

Las organizaciones deben poner en marcha métodos de medida y análisis que les permitan detectar las no conformidades, mediante parámetros y puesta en marcha de acciones que minimicen dichas no conformidades y tiendan a su eliminación (42).

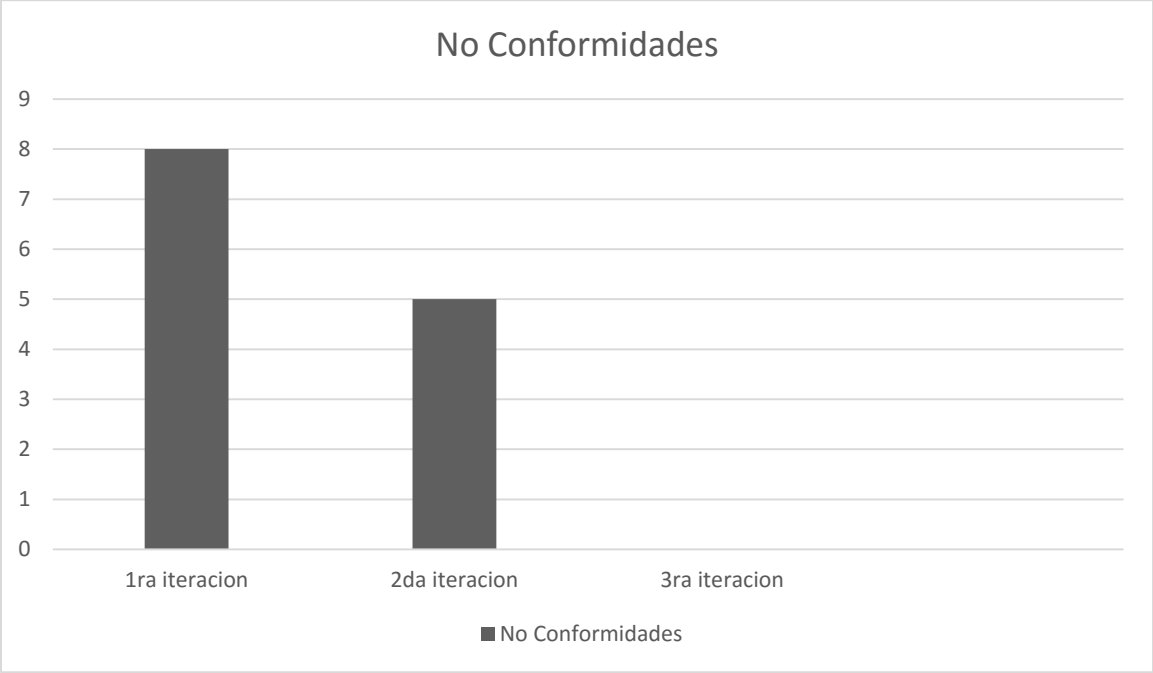


Figura 12 Resultados de las pruebas realizadas.

Con el objetivo de que las funcionalidades correspondientes a los requerimientos de seguridad autenticación y generación de notificaciones tengan un correcto funcionamiento, se realizaron las pruebas en tres iteraciones. Como se observa en la figura en la primera iteración fueron detectadas ocho no conformidades las cuales fueron solucionadas en la propia iteración. En la segunda iteración fueron detectadas cinco no conformidades asociadas a errores de interfaz siendo satisfactoriamente solucionadas. En una tercera y última iteración no se detectó ninguna no conformidad por lo que el sistema mostró un buen funcionamiento y se considera terminada la implementación de los requerimientos de seguridad.

Conclusiones del capítulo

En este capítulo se describió como se llevó a cabo el proceso de implementación de los requerimientos de seguridad autenticación y notificaciones en el sistema informático XABAL Arkheia 4.0. Se planteó un plan de pruebas a realizar para comprobar el funcionamiento de la fase de implementación describiendo cada uno de los procesos durante el desarrollo del plan de prueba trazado, obteniendo al final el resultado de las pruebas a las que fue sometido el sistema.

CONCLUSIONES GENERALES

Durante el desarrollo de la investigación se planteó la necesidad de mejorar los requerimientos de seguridad en el sistema informático XABAL Arkheia 4.0 dándole así cumplimiento al objetivo propuesto.

El análisis de los principios teóricos de la investigación, enfocado a los requerimientos de seguridad en la autenticación y generación de notificaciones, permitió tener un mejor entendimiento de los procesos de este tipo en la universidad.

Se generaron los artefactos correspondientes al modelado del negocio donde se describen y se diseñan las funcionalidades del proceso.

El estudio de la versión anterior sistema informático XABAL Arkheia 3.0 permitió analizar los requerimientos de seguridad en la autenticación y generación de notificaciones posibilitando la realización de la implementación en la nueva versión, adaptada a las características del entorno actual de la institución.

El empleo de un proceso de desarrollo guiado por la metodología, herramientas y tecnologías libres favoreció el trabajo del equipo y trajo como resultado la obtención de un sistema que cumple con los requerimientos definidos.

Los requerimientos de seguridad que se implementaron son un aspecto que no incluía el sistema en cuestión dentro de sus funcionalidades, por lo que constituyen funcionalidades novedosas para este sistema.

El diseño y ejecución de los casos de prueba al software, permitieron identificar errores de implementación en la aplicación, que fueron corregidos posteriormente dándole así cumplimiento al objetivo general de esta investigación.

RECOMENDACIONES

- Configuraciones del servidor tomcat:

  -Uso de SSL obligatorio para forzar que todas las conexiones se realicen cifradas, redirigiendo siempre solicitudes http a https.

  -Utilizar cookies declaradas como HTTPonly para proteger a las mismas de lecturas y escrituras por parte de scripts del cliente. De esta forma solo el servidor y el navegador tendrán acceso a las cookies, dificultando así robos de sesión y otro tipo de problemas de seguridad.

  -Restringir IP/redes desde los que se puede acceder y su dominio.

- Una vez terminado el sistema en su totalidad se recomienda la certificación del Departamento de Ciberseguridad Tecnológica de la Dirección de Tecnologías y Sistemas del Ministerio del Interior del 2018.

REFERENCIAS

1. **Rodríguez, Angelica.** TrustDimension. *TrustDimension.* [En línea] TrustDimension, 18 de enero de 2016. [Citado el: 12 de enero de 2019.] https://www.trustdimension.com/la-importancia-de-la-seguridad-informatica/.

2. **Lemus, Rodolfo Godoy.** *Seguridad de la Información: Revista de la Segunda Cohorte del Doctorado en Seguridad Estratégica.* Guatemala : Segunda Cohorte Doctorado en Seguridad Estratégica, 2014.

3. **Torres, Jaime A.** *Sistema de Gestión de Seguridad de la Información.* Ecuador : s.n., 2016.

4. **Borghello, Cristian.** Segu.Info. [En línea] [Citado el: 21 de enero de 2019.] https://www.segu-info.com.ar/logica/identificacion.htm.

5. **Universidad de Uruguay.** *Seguridad Informática.* Uruguay : s.n., 2017.

6. **UCI.** Uci. [En línea] Universidad de las Ciencias Informaticas. [Citado el: 14 de enero de 2019.] https://www.uci.cu/investigacion-y-desarrollo/productos/xabal/arkheia-30.

7. **Valencia, Universidad Internacional de.** Universidad Internacional de Valencia. [En línea] VIU, 21 de marzo de 2018. [Citado el: 12 de enero de 2019.] https://www.universidadviu.com/la-seguridad-informatica-puede-ayudarme/.

8. **Significados.** Significados. *Significados.* [En línea] Significados, 16 de febrero de 2017. [Citado el: 13 de enero de 2019.] https://www.significados.com/informacion/.

9. **Santana, Charly.** SlideShare. *SlideShare.* [En línea] Linkedln, 1 de enero de 2011. [Citado el: 13 de enero de 2019.] https://es.slideshare.net/CharlySantana1/integridad-de-la-informacion.

10. **Center, Search Data.** TechTarget. *TechTarget.* [En línea] TechTarget S.A de C.V, 1 de junio de 2014. [Citado el: 14 de enero de 2019.] https://searchdatacenter.techtarget.com/es/definicion/Privacidad-de-datos-privacidad-de-informacion.

11. **IBM.** IBM Knowledge Center. [En línea] IBM. [Citado el: 13 de 1 de 2019.] https://www.ibm.com/support/knowledgecenter/es/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009740_.htm.

12. **Google Sites.** Seguridad Informatica. [En línea] Google, 14 de marzo de 2017. [Citado el: 17 de enero de 2019.] https://sites.google.com/site/seguridadinformatica323/-bienvenida.

13. **RedIris.** [En línea] Red.es. [Citado el: 19 de enero de 2019.] https://www.rediris.es/cert/doc/unixsec/node14.html.

14. **UCI.** Metodología de desarrollo para la Actividad productiva de la UCI. [aut. libro] Tamara Rodríguez Sánchez. La Habana : s.n., 2015.

15. **Asensio, Rafael Menéndez-Barzanallana.** Ingeniería del software. [En línea] 4 de 10 de 2018. [Citado el: 14 de 1 de 2019.] https://www.um.es/docencia/barzana/IAGP/Enlaces/CASE_principales.html.

16. **Visual Paradigm.** Visual Paradigm Product Overview. [En línea] [Citado el: 15 de 1 de 2019.] https://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html.

17. **Manuel, F.** Genebeta. [En línea] WebEdia, 9 de febrero de 2012. [Citado el: 15 de marzo de 2019.] https://www.genbeta.com/herramientas/sublime-text-un-sofisticado-editor-de-codigo-multiplataforma.

18. **Sandra, Jonathan.** DwES. [En línea] 2019. [Citado el: 10 de diciembre de 2018.] http://www.alba9.es/Joomla/index.php/publicaciones/23-ventajas-y-desventajas-de-visual-studio-code.html.

19. **Atareado .** El atareado linux para legos. [En línea] Disqus, 18 de mayo de 2018. [Citado el: 30 de noviembre de 2018.] https://www.atareao.es/software/programacion/visual-studio-code/.

20. **Alfresco Community Edition.** Alfresco. [En línea] Alfresco software,Inc. [Citado el: 20 de enero de 2019.] https://www.alfresco.com/es/plataforma/alfresco-community-edition.

21. **Freemarker.** <#freemarker>. [En línea] [Citado el: 23 de enero de 2019.] https://freemarker.apache.org/.

22. **Mozilla.** MDN web docs. [En línea] Mozilla, 25 de septiembre de 2018. [Citado el: 23 de enero de 2019.] https://developer.mozilla.org/es/docs/Web/JavaScript.

23. **Gouchat, Juan Diego.** *El Gran Libro de HTML5,CSS3 y JavaScript.* Barcelona : Publidisa, 2012. 978-84-267-1782-5.

24. **Mozilla.** MDN web docs. [En línea] 25 de enero de 2014. [Citado el: 17 de enero de 2019.] https://developer.mozilla.org/es/docs/HTML/HTML5/Introducci%C3%B3n_a_HTML5.

25. **Gauchat, Juan diego.** *El gran libro de HTML5,CCS3 y JavaScript.* Barcelona : Marcombo S.A. ISBN : 978-84-267-1782-5.

26. **Mozilla org.** MDN Web Docs. [En línea] 23 de marzo de 2016. [Citado el: 23 de enero de 2018.] https://developer.mozilla.org/es/docs/Web/XML/Introducci%C3%B3n_a_XML.

27. **Gupta, Yogendra.** Github. [En línea] Github Inc, 13 de febrero de 2015. [Citado el: 24 de enero de 2019.] https://github.com/yui/yui3/wiki/FAQ#what-is-yui.

28. **CAROLINA MARTÍNEZ, I.** *Modelo Conceptual / Modelo de Dominio.* Caracas : Universidad Simón Bolívar, 2010.

29. **Google sites.** Metodología Gestión de Requerimientos. [En línea] [Citado el: 21 de enero de 2019.] https://sites.google.com/site/metodologiareq/capitulo-ii/tecnicas-para-identificar-requisitos-funcionales-y-no-funcionales.

30. **Guilherme Siqueira Simões, Carlos Eduardo Vazquez.** La oficina de proyectos de informática. [En línea] 6 de febrero de 2016 . [Citado el: 25 de enero de 2019.] www.pmoinformatica.com/2015/05/requerimientos-no-funcionales-ejemplos.html.

31. **José, Francisco.** *Fundamentos de la vista de casos de uso.* Salamanca : Universidad de Salamanca, 2018.

32. **Universidad de Barcelona.** OBS-Business School. [En línea] Universidad de Barcelona, 2018. [Citado el: 23 de marzo de 2019.] https://www.obs-edu.com/int/blog-project-management/herramientas-esenciales/cual-es-la-utilidad-de-la-matriz-de-trazabilidad.

33. **Pressman, Roger S.** *Ingeniería de Software enfoque práctico 7ma edición.* México : McGraw-Hil, 2010. ISBN: 978-607-15-0314-5.

34. **Lizardo, Maria Eugenia Arevalo.** *Introducción al Patrón de Arquitectura por Capas.* 2010.

35. **Astudillo, Marcello Visconti y Hernan.** Repositorio institucional de la Universidad de Las Tunas. [En línea] [Citado el: 15 de marzo de 2019.] http://roa.ult.edu.cu/bitstream/123456789/401/1/08-Patrones.pdf.

36. **LARMAN, C.** *UML y Patrones.* Prentice Hall:New York : s.n., 1999.

37. **Sparks Systems.** [En línea] 12 de marzo de 2015. [Citado el: 25 de abril de 2019.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.

38. **Oficina de proyectos de informática.** La web sobre gerencia de proyectos de informática, software y tecnología. [En línea] 2012. [Citado el: 15 de abril de 2019.] http://www.pmoinformatica.com/p/pruebas-de-software.html.

39. **IBM Corp.** Concepto niveles de prueba. [En línea] 2006. [Citado el: 25 de marzo de 2019.] https://cgrw01.cgr.go.cr/rup/RUP.es/SmallProjects/core.base_rup/guidances/concepts/levels_of_test_8A878577.html#.

40. **Llontop, Alexis.** Aditec. [En línea] [Citado el: 12 de abril de 2019.] https://adictec.com/que-son-pruebas-de-regresion/.

41. **International Software Testing Qualifications Board.** *Programa de estudio de nivel básico.* s.l. : Foundation Level Syllabus, 2010. ISTQB-2010.

42. **Testing Colombia.** [En línea] 18 de enero de 2016. [Citado el: 25 de abril de 2019.] https://www.testingcolombia.com/casos-de-prueba-que-son-como-se-hacen-y-para-que-sirven/.

43. **Asociación Española para la Calidad.** [En línea] 2018. [Citado el: 1 de mayo de 2019.] https://www.aec.es/web/guest/centro-conocimiento/no-conformidad.

44. **Mozilla.** MDN web docs. [En línea] Mozilla, 24 de febrero de 2014. [Citado el: 12 de enero de 2019.] https://developer.mozilla.org/es/docs/Rhino.

45. **Alfreco Enterprise.** Alfresco Documentation. [En línea] [Citado el: 22 de enero de 2019.] https://docs.alfresco.com/5.0/concepts/aikau-intro.html.

46. **Universidad a distancia de Madrid.** [En línea] [Citado el: 21 de marzo de 2019.] http://www.mundolinux.info/que-es-xml.htm.

47. **FA, Ruben.** Genbeta. [En línea] 24 de febrero de 2010. [Citado el: 14 de abril de 2019.] https://www.genbeta.com/desarrollo/patrones-de-diseno-que-son-y-por-que-debes-usarlos.

48. **C, Larman.** *UML y Patrones.* New York : Prentice Hall, 1999.

49. **EXE.** Curso de Introducción a Java. [En línea] [Citado el: 13 de enero de 2019.] http://www.mundojava.net/caracteristicas-del-lenguaje.html?Pg=java_inicial_4_1.html.

GLOSARIO DE TÉRMINOS

Login: iniciar sesión.

UCI: Universidad de las Ciencias Informáticas.

TIC: Tecnologías de la Información y las Comunicaciones.

CIGED: Centro de Informatización de la Gestión Documental.

AUP: Agile Unified Process, es una metodología de desarrollo de software de tipo ágil.

AUP-UCI: Proceso Unificado Ágil para el Desarrollo de la Actividad Productiva en la Universidad de las Ciencias Informáticas.

Norma ISAD (G): son las siglas en inglés de General International Standard Archival Description (Norma Internacional General de Descripción Archivística), publicada por el Consejo Internacional de Archivos (CIA) en 1994.

ID: es la abreviatura del vocablo inglés Identification, que traducido al idioma español significa identificación.

CMMI-DEV v1.3: Integración de sistemas modelos de madurez de capacidades o Capability Maturity Model Integration (CMMI) es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software.

MVC: Modelo Vista Controlador.

W3C: son las siglas de World Wide Web Consortium, un consorcio fundado en 1994 para dirigir a la Web hacia su pleno potencial mediante el desarrollo de protocolos comunes que promuevan su evolución y aseguren su interoperabilidad.

GRASP: son patrones generales de software para asignación de responsabilidades.

HTTPS: Protocolo seguro de transferencia de hipertexto.

TCP/IP: Protocolo de Control de Transmisión/Protocolo de Internet.

SMTP: Protocolo de comunicación que permite el envío de correos electrónicos en internet.

ISO 9000-2005: es una Norma Internacional que detalla los fundamentos de los Sistemas de Gestión de la Calidad.