Universidad de las Ciencias Informáticas

Facultad 3

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

Módulo de gestión de auditorías a expedientes en el XABAL eXcriba 4.0

Autor:

Yadira Alarcón Álvarez

Tutores:

Msc. Aurelio Antelo Collado

Ing. Liomar Rodríguez Guerra

La Habana, agosto, 2020

“Año 62 de la Revolución”

Resumen

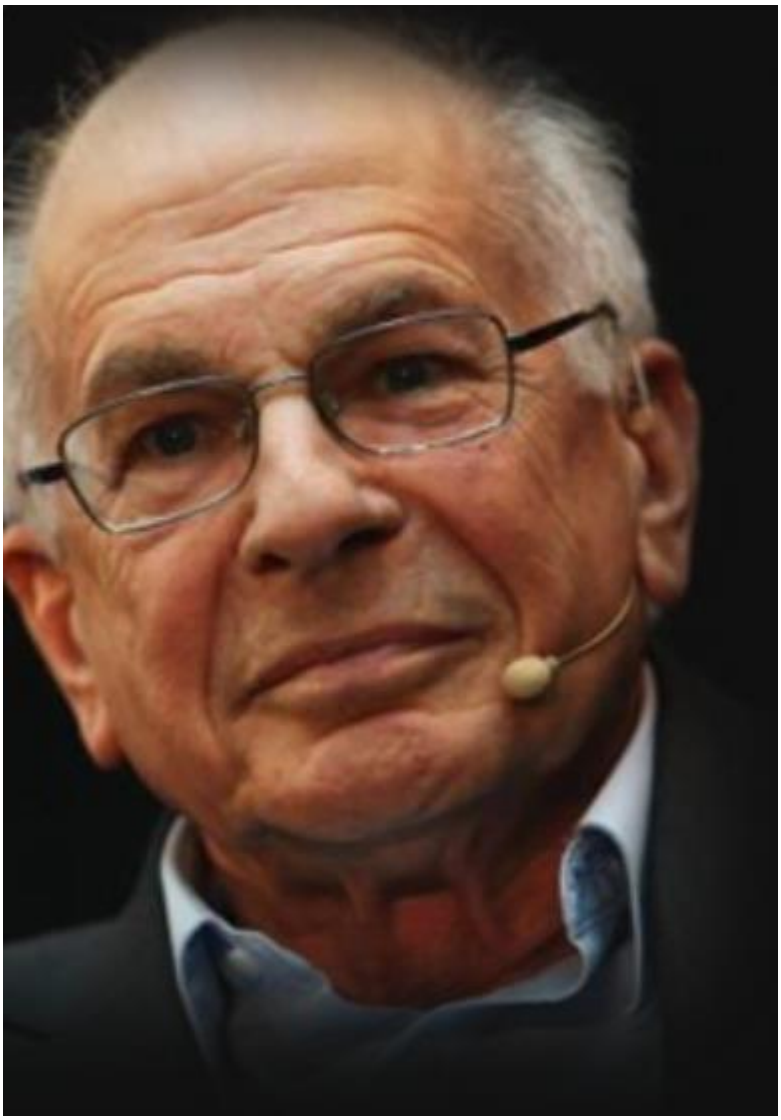
En la actualidad se ha evidenciado un acelerado desarrollo de la ciencia y la tecnología, trayendo consigo un aumento de la producción documental. Este fenómeno no ha sido esquivo a las empresas e instituciones que manejan un gran cúmulo de información. A este problema se le ha dado solución, en alguna medida, con el desarrollo de sistemas de gestión de documentos electrónicos que permitan el control y tratamiento de la documentación.

La presente investigación propone el desarrollo de un módulo para el Gestor de Documentos Administrativos Xabal eXcriba 4.0, el cual cuenta con funcionalidades que permiten llevar a cabo un control sobre las auditorías a expedientes, para ello se realizó un estudio minucioso sobre el funcionamiento de las auditorías. Durante el desarrollo del módulo se utilizaron como lenguajes de desarrollo JavaScript 1.6, XML 1.0, y HTML5 y como herramientas de desarrollo el Visual Paradigm para UML 8.0, Apache Ant 1.9 y el editor de texto Visual Studio Code v1.31.0. La implementación de este módulo permite la accesibilidad de la información, alcanzándose un control estricto de las auditorías a expedientes, facilitando la planificación de las mismas.

Palabras claves: auditoría, módulo, expediente, gestión documental.

“Los momentos finales de una experiencia determinan el recuerdo que conservamos de la misma”

Daniel Kaheman.



Agradecimientos

Agradezco a mis padres, por ser un ejemplo para mí, demostrándome día a día que para alcanzar cualquier sueño hay que perseverar y sacrificarse, les agradezco por estar siempre dispuestos a escucharme,a mi madre gracias por cada lágrima junto a mí después de cada prueba, por su apoyo incondicional, les agradezco que se aventuraran conmigo en esta travesía y me siento muy feliz que hoy podamos sentirnos orgullosos de haberlo logrado.

A mi tía Edilma y a mis hermanitos,gracias por siempre estar,por, la ayuda,por la preocupación en todo momento.

A mi novio,mi titi,mi todo,gracias por el apoyo,gracias por confiar en mi y demostrarme que si podia,gracias por no dejar que nunca me rindiera a pesar de nuestra distancia,gracias por cada mensaje de éxitos antes de cada prueba,gracias por cada noche de estudio y por cada empujoncito que me diste,te amo.

A mi familia gracias por el apoyo.

A mis amigos Annet,Rolando y Yusnaby gracias por cuidarme y quererme ,annecita nunca podia haber elegido major compañera de cuarto y amiga.

A mis compañeros de los dos grupos por lo que pase.

A mis tutores Aurelio y Liomar,muchas gracias por la ayuda.

Ala profesora Dailin mil gracias por todo,gracias por el esmero,por la atención.

Dedicatoria

Dedico esta tesis a las personas más importantes en mi vida, mis padres y mi novio , por impulsarme a seguir adelante, por su amor incondicional y su entrega, por los incontables consejos y por todo su amor.

Índice

ÍNDICE

INTRODUCCIÓN

Introducción

La industria del software es considerada un pilar fundamental para el desarrollo tecnológico de cualquier país. En el mundo se desarrolla a un ritmo vertiginoso, aunque la producción sigue siendo baja y los costos muy elevados. Esta situación se debe, en la mayoría de los casos, a la no aplicación de Técnicas de Ingeniería y Gestión de Software y a la no definición de roles y procesos adecuados en el desarrollo de software. En la actualidad, las empresas de software se esfuerzan por ampliar y mejorar su posición en el mercado, basadas, entre otros aspectos, en la calidad de sus servicios. Las mismas dedican grandes presupuestos para ofrecerles a sus sistemas de software un aseguramiento de la calidad eficiente por el aporte al dinamismo y crecimiento económico que ofrecen. Garantizar la calidad del software es una necesidad, y constituye la conformidad con los requisitos funcionales y de rendimiento explícitamente establecidos, los estándares de desarrollo documentados y las características que se esperan de todo software desarrollado profesionalmente. La calidad de un producto software está principalmente ligada a la calidad del proceso de desarrollo del mismo.

Para lograr un buen aseguramiento de la calidad se realizan auditorías y revisiones de software, estas tienen gran importancia, pues permiten a las organizaciones mejorar su desempeño de forma continua. La auditoría de la información se define como“...un proceso para descubrir ,monitorear y evaluar los flujos y recursos de información, a fin de implementar, mantener o mejorar su gestión en la organización”[1].

Una auditoría se define como un examen **exhaustivo, sistemático y metódico** que se realiza para determinar si las **actividades y resultados relativos a la calidad satisfacen las disposiciones** previamente establecidas y que realmente se llevan a cabo. También se comprueba si estas son las adecuadas para alcanzar los objetivos propuestos por la organización[2] .

En el campo de la gestión de documentos, se puede definir auditoría como “un examen sistemático, planeado y organizado que determina si las actividades y los resultados relacionados con la gestión de documentos cumple con las disposiciones establecidas (métodos, procedimientos, etcétera.), y si éstas se aplican en forma efectiva para alcanzar los objetivos planteados, no sólo por la unidad responsable de la gestión documental, sino por la organización”[3].

Entre los objetivos fundamentales de las auditorías y las revisiones de software, se encuentran determinar e informar sobre el grado de correspondencia existente entre la información cuantificable y los criterios de evaluación establecido. Además, indicar la necesidad de mejoras en un producto, así como el descubrimiento de los defectos e inconsistencias en el sistema y evaluar conformidad con estándares y especificaciones técnicas. Asimismo, es una alta aspiración de toda auditoría que sus resultados contribuyan a la retroalimentación de los clientes de la auditoría, de los auditados y de los propios auditores[2].

INTRODUCCIÓN

Cuba no está ajena en cuanto a desarrollo de soluciones informáticas se trata, puesto que las organizaciones productoras de software se han planteado como meta obtener un producto que cumpla con las normas y estándares internacionales de calidad y para ello forma profesionales calificados en esta rama. Para aumentar la calidad de los productos de software e insertarse en el mercado internacional y satisfacer la creciente demanda del país, se crea el Centro Nacional de Calidad de Software (CALISOFT) en el año 2012. Este centro tiene como objetivo principal ofrecer servicios de evaluación de la calidad de los procesos de desarrollo de software, los procesos de prestación de servicios de tecnología de la información y los productos informáticos. Además, tiene la tarea de establecer estándares técnicos y procedimientos que normalicen la industria informática, dirigiéndola hacia niveles superiores. También sobresale el proyecto de investigación "Modelo de mejora de procesos de software para la industria cubana del software" desarrollado entre CALISOFT y la Universidad de las Ciencias Informáticas (UCI), donde el resultado esperado es la obtención de un Modelo de la Calidad para el Desarrollo de Aplicaciones Informáticas (MCDAI) [4]. La Universidad de las Ciencias Informáticas (UCI) es uno de los máximos exponentes en el desarrollo de la industria del software. La UCI es una universidad productiva, la cual pretende servir de soporte para la industria cubana de la informática, para ello se trabaja en la producción de software y servicios informáticos. Estableciendo que su proceso de desarrollo debe cumplir con cada una de las prácticas genéricas y específicas establecidas para el área de Aseguramiento de la Calidad a Proceso y Producto (PPQA).
En la universidad se encuentra institucionalizado el nivel dos del Modelo Integrado de Capacidad y Madurez (CMMI, del inglés Capability Maturity Model Integration), el cual propone una administración disciplinada de los proyectos donde se establezcan y sigan políticas organizacionales. Haciéndose necesario en los proyectos de la Universidad, realizar revisiones al proceso de desarrollo de software para identificar los defectos que se generan durante todo el ciclo de vida de los mismos. Para el aseguramiento de la calidad del software, la UCI cuenta con una Dirección de Calidad de Software (DCSW). La cual realiza varias actividades para prevenir, controlar y mejorar la calidad de los productos, entre ellas se encuentra las auditorías. Estas son un método probado para eliminar defectos de manera temprana y proporcionar una visión de valor sobre los productos de trabajo y los componentes de producto que están siendo desarrollados y mantenidos.

En la UCI se encuentra el Centro de Informatización de la Gestión Documental (CIGED), dedicado al desarrollo de sistemas y servicios informáticos integrales de alta calidad y competitividad en la informatización de los procesos de gestión documental, cuenta con el producto Gestor de Documentos Administrativos Xabal eXcriba en su versión 4.0, este es un software diseñado para la gestión documental, que permite el trámite de los documentos administrativos que se generan o reciben dentro de las organizaciones a partir de sus funciones, por lo tanto, involucra todas las áreas de una organización, permitiéndoles gestionar de forma correcta la documentación como prueba, testimonio y

evidencia de las actividades organizacionales. La solución de software agiliza el trámite de los documentos, permitiendo controlar el estado de los mismos, evitando la pérdida de información.
El eXcriba 4.0, permite trabajar en un entorno de colaboración entre los usuarios de la organización, compartiendo los documentos con otros usuarios y estableciendo niveles de acceso y permiso.
Además, optimiza la organización de los documentos, posibilita la búsqueda de información y gestiona los documentos durante su ciclo de vida: captura, creación, clasificación, descripción, protección, retención de archivo y expurgo. Las interfaces de usuario de esta aplicación están orientadas a personas que no tienen que poseer un alto grado de conocimiento de informática para trabajar con el sistema.
Actualmente este producto presenta limitantes en el proceso de auditorías, debido a que no existe un módulo que gestione las auditorías a los expedientes. Existe insuficiencias y pérdida de documentos, lo que implica no tener las evidencias necesarias. Existe dependencia del correo electrónico para la circulación de la documentación, lo que conlleva a un aumento del tráfico de red. No presenta un mecanismo de seguimiento y control de las auditorías, la forma en la que se lleva el control de los cambios en el área es poco segura, provocando modificaciones no autorizadas en los documentos. El flujo de información que se realiza es de forma manual lo que trae consigo retrasos en la entrega de documentos y no se conoce oportunamente en cuál paso del proceso se encuentra detenida la información. Los documentos no se encuentran clasificados ni organizados, provocando que se consuma mucho tiempo para su búsqueda. El mecanismo de otorgarle permiso a las auditorías del proyecto es muy engorroso y depende de un administrador del sistema, es por ellos que los documentos los revisores los revisan en el laboratorio donde trabaja el proyecto.

De acuerdo con la situación problemática antes expuesta, en este trabajo se plantea como **Problema científico:** ¿Cómo gestionar un control y seguimiento a las auditorías de calidad realizadas a los expedientes con el producto XABAL-eXcriba 4.0?
Como resultado del análisis del propio problema se definió como **objeto de estudio:** Auditorías de expedientes en el proceso de desarrollo de software.
Para dar solución al problema se plantea como **objetivo general**: Desarrollar el módulo de gestión de auditorías del producto XABAL eXcriba 4.0 que permita a la UCI mantener un control y seguimiento a las auditorías de calidad realizadas a los expedientes.
Se define para la investigación el siguiente **campo de acción:** Automatización de la gestión de las auditorías a expedientes.

Se plantean como **preguntas de la investigación:**

- ¿Cuáles son los referentes teóricos a tener en cuenta para abordar la solución del problema planteado relacionado con la automatización de la gestión de auditorías realizadas a los expedientes en la tecnología de Alfresco 5.2 ?
- ¿Qué propuesta de solución se define para implantar en el módulo gestión de auditorías realizadas a los expedientes en la tecnología de Alfresco 5.2 ?

- ¿Cómo desarrollar un módulo de gestión de auditorías en la tecnología de Alfresco 5.2 ?, que permita la creación, seguimiento y control de auditorías de calidad realizadas a los expedientes?
- ¿Cómo se valida el correcto funcionamiento del módulo de auditorías en la tecnología de Alfresco 5.2 ?

Para dar cumplimiento al objetivo general, se planearon y ejecutaron las siguientes **tareas de investigación:**

- Análisis de los principales conceptos asociados a la gestión de auditorías de calidad realizadas a los expedientes en la tecnología de Alfresco 5.1, para obtener los fundamentos teóricos necesarios para el desarrollo de la solución.
- Análisis de los sistemas homólogos para identificar tendencias actuales relacionadas con las funcionalidades y mecanismos utilizados.
- Análisis de la metodología de software, herramientas informáticas y tecnologías para su posterior utilización en el desarrollo del Módulo de gestión de auditorías de calidad.
- Diseño del mecanismo de control y seguimiento a utilizar en el módulo de gestión de auditorías de calidad.
- Implementación del módulo de gestión de auditorías auditorías basada en el mecanismo de control y seguimiento definido.
- Validación del Módulo de gestión de auditorías en la tecnología de Alfresco 5.1 para comprobar su correcto funcionamiento.

Para llevar a cabo la investigación propuesta se han utilizado los siguientes métodos científicos de investigación:

Métodos teóricos

- **Analítico-Sintético:** permitió analizar los conocimientos generales sobre las auditorías de calidad, estableciendo la extracción de los elementos más significativos, arrojando ideas y conclusiones mucho más prácticas y concretas.
- **Inductivo-deductivo**: se utilizó para establecer una forma de razonamiento acerca de las particularidades de las auditorías de calidad, para reflejar sus puntos comunes.
- **Modelación:** Permitió modelar la realidad de la investigación, los principales elementos que la componen y su funcionamiento, lo cual ayudó a descubrir y estudiar nuevas cualidades y relaciones del objeto de estudio.

Métodos empíricos

- El **criterio de expertos**, para la validación de los aportes fundamentales de la investigación.

- El **método experimental** para comprobar la utilidad de los resultados obtenidos a partir de la implementación del proceso.
- El **estudio de casos** para la valoración de la funcionalidad y eficiencia del sistema propuesto.
- **Observación:** Se utilizó para percibir a partir de la situación real que se investiga, cómo se desarrollan a groso modo los procesos que constituyen el objeto de estudio.
- **Entrevista:** Se aplicó una entrevista informal la directora de la dirección de calidad, las preguntas fueron referentes a las actividades que se desarrollan durante las auditorías de calidad.

El presente trabajo consta de tres capítulos estructurados de la siguiente forma:

Capítulo 1. Fundamentación teórica de la gestión auditorías a expedientes. En este capítulo se abordarán los conceptos fundamentales que permitirán entender las características del sistema que se desea desarrollar y los conceptos relacionados con el tema de investigación, también se analizarán los sistemas homólogos, la definición del entorno de trabajo en el cual se verá reflejado, la fundamentación de las tecnologías, herramientas, metodologías y lenguajes que se emplearán en la construcción del sistema de automatización de auditorías a expedientes.

Capítulo 2. Propuesta de solución del módulo de gestión de auditorías a expedientes. En este capítulo se define la propuesta de solución para el desarrollo del módulo para la gestión de auditorías a expedientes. Se definen los requisitos funcionales y no funcionales capturados dadas las necesidades del cliente para la implementación del sistema, partiendo del estudio del estado del arte realizado en el capítulo anterior. También se definen otros elementos importantes dentro del proceso de desarrollo, tales como: arquitectura, patrones empleados y elementos de la implementación del mismo.

Capítulo 3. Implementación y validación del módulo de gestión de auditorías a expedientes: en este capítulo se explica el proceso de implementación y se plasman los casos de pruebas a los que fue sometido el módulo en cada una de las iteraciones. Se exponen los resultados obtenidos y se muestran las funcionalidades alcanzadas en el período de desarrollo.

CAPÌTULO I

Capítulo I: Fundamentación teórica de la gestión de auditorías a expedientes.

En el desarrollo de todo sistema informático es importante analizar los principales conceptos asociados con el dominio del problema para un mejor modelado de la solución, estudiar los sistemas semejantes que existen en el mundo, al que se necesita; con el objetivo de no duplicar esfuerzos y aprovechar conocimientos acumulados. Es por ello que en este capítulo se brindan unas descripciones generales de los conocimientos relacionados con auditorías de calidad y los conceptos necesarios para la comprensión del problema. Se aborda también sobre los sistemas homólogos existentes vinculados al campo de acción y se describen las tecnologías, herramientas y componentes de la propuesta de solución.

1.1 Conceptos relacionados con las auditorías de calidad.

Para entender mejor las temáticas que serán abordadas en la investigación, se hace necesario conocer un conjunto de conceptos relacionados con el dominio del problema.

Calidad del software

El aseguramiento de la calidad asociado a la producción de software es fundamental, determina la eficacia del proceso de ingeniería que se desarrolla en toda producción de este tipo. Son varias las definiciones que existen en el mundo que se refieren a la calidad de software, tales como: IEEE estándar 610-1900 “la calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario” [5] .

La norma ISO2 8402:1994 plantea que es “el conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas”[6].

Roger Pressman define que es “la concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente, que desea el usuario”[7].

La gestión de la calidad

Aspectos de la función de gestión que determinan y aplican la política de la calidad, los objetivos y las responsabilidades y que lo realiza con medios tales como la planificación de la calidad, el control de la calidad, la garantía de calidad y la mejora de la calidad[8].

Dentro de la gestión de la calidad se observa [9]:

- **Gestión de la calidad de software (ISO 9000):** Conjunto de actividades de la función general de la dirección que determina la calidad, los objetivos y las responsabilidades y se implanta por medios tales como la planificación de la calidad, el control de la

calidad, el aseguramiento (garantía) de la calidad y la mejora de la calidad, en el marco del sistema de calidad.

Política de calidad (ISO 9000): Directrices y objetivos generales de una organización, relativos a la calidad, tal como se expresan formalmente por la alta dirección [10].
La gestión de la calidad se aplica normalmente a nivel de empresa. También puede haber una gestión de calidad dentro de la gestión de cada proyecto.

Aseguramiento de la calidad

El aseguramiento de la calidad establece la infraestructura de apoyo a los métodos sólidos de la ingeniería de software, la administración racional de proyectos y las acciones de control de calidad, todo de importancia crucial si se trata de elaborar software de alta calidad. Además, el aseguramiento de la calidad consiste en un conjunto de funciones de auditoría y reportes para evaluar la eficacia y completitud de las acciones de control de calidad. El objetivo del aseguramiento de la calidad es proveer al equipo administrativo y técnico los datos necesarios para mantenerlo informado sobre la calidad del producto, con lo que obtiene perspectiva y confianza en que las acciones necesarias para lograr la calidad del producto funcionan. Por supuesto, si los datos provistos a través del aseguramiento de la calidad identifican los problemas, es responsabilidad de la administración enfrentarlos y aplicar los recursos necesarios para resolver los correspondientes a la calidad. Para mejorar la calidad los procesos de desarrollo de software se definen diferentes métodos/procedimientos con el objetivo de desarrollar un software de alta calidad[11].

Aseguramiento de la Calidad de Procesos y Productos (PPQA)

El Aseguramiento de la Calidad de Procesos y Productos (PPQA), es el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza que el software cumplirá los requisitos dados de calidad. Este aseguramiento se diseña para cada aplicación antes de comenzar a desarrollarla y no después. El aseguramiento de la calidad del software engloba: un enfoque de gestión de calidad, revisiones técnicas formales aplicables en el proceso de software, el control de la documentación del software, los cambios realizados y mecanismos de generación de informes. Además, provee toda la gestión a la planta de personal, así como define los objetivos verificables dentro de los procesos y productos de trabajo para su evaluación[12].

Las revisiones del software

Una revisión de software es "un proceso o reunión en la que un producto de software es examinado por un proyecto personal, los administradores, usuarios, clientes, representantes de los usuarios, u otros participantes para comentarios o aprobación. En este contexto, el término "producto de software" significa "cualquier documento técnico o un documento parcial, producida como un entregable de una actividad de desarrollo de software", y puede incluir documentos tales como contratos, planes de proyectos y presupuestos, documentos de requerimientos y especificaciones, diseños, código fuente, documentación del usuario, soporte y mantenimiento de documentación, planes de prueba, especificaciones de prueba, normas, y cualquier otro tipo de producto de trabajo especializado[13].

Pressman (2010) define las revisiones de software como un "..."filtro "para el proceso de ingeniería del software. Estas se aplican en varios momentos del desarrollo del software y sirven para detectar errores y defectos que puedan así ser eliminados. Las revisiones del software sirven para "purificar" las actividades de ingeniería del software que suceden como resultado del análisis, el diseño y la codificación"[7].

Como se ilustra en la figura 1 las revisiones pueden ser de dos tipos, dinámicas y estáticas. Las primeras son las que detectan los defectos ejecutando el software, fundamentalmente son las ejecutadas en la fase de prueba del proyecto. Las segundas son visuales y se realizan sin necesidad de que el software esté ejecutándose. Ambas son de suma importancia y una combinación adecuada puede detectar gran cantidad de defectos y por tanto mejorar la calidad del producto final [14].

Figura #1 Tipos de revisión.

Tienen como objetivos [7]**:**

- Señalar la necesidad de mejoras en el producto.
- Continuar las partes de un producto en las que no es necesaria o no es deseable una mejora.

- Conseguir un trabajo técnico de una calidad más uniforme o al menos más predecible, que la que puede ser conseguida sin revisiones, con el fin de hacer más manejable el trabajo técnico.

Las revisiones se clasifican en informales y formales. Las primeras se realizan de forma constante y sin ellas la programación y comprensión de un proyecto serían prácticamente imposible.

Revisión Técnica Formal (RTF)

"Una Revisión Técnica Formal (RTF) es el proceso de evaluar de manera estricta requerimientos, especificaciones, arquitecturas, diseños, código, planes de prueba, procedimientos y otros artefactos, estableciendo el nivel de conformidad, completez, correctez, trazabilidad, estilo y otros criterios que permitirán detectar discrepancias y proponer acciones de mejora". Este proceso involucra la planeación de la revisión, la preparación individual del revisor para entender el contexto del sistema, la inspección en sí y la retroalimentación[15].

En su aplicación incluye diferentes roles: moderador, secretario, revisor, lector y productor, los cuales generalmente son jugados por una o dos personas (el desarrollador y algún revisor del mismo equipo o externo). Las RTF reducen el costo de desarrollo, la propagación de defectos y el mantenimiento, aumentan la satisfacción del cliente y mejoran la moral entre los practicantes.

Durante años se han sugerido una serie de pautas para la RTF de avances dentro de los Sistemas de Software (SS), pero son pocas las veces que se especifica qué revisar, cómo y sobre todo cuales parámetros de calidad se están asociando, ya que la importancia de las RTF radica no sólo en el hecho de que deben automatizarse, sino en la elección correcta de los elementos sobre los cuales se hará la revisión[15].

Objetivos de las RTF [16].

Descubrir errores en la función lógica o implementación en cualquier representación del software.

- Verificar que el software en revisión satisface sus requisitos.
- Garantizar que el software se ha representado de acuerdo con los estándares predefinidos.
- Lograr un desarrollo de software uniforme.
- Hacer proyectos manejables.

Las Revisiones del software se pueden dividir en tres categorías[17]**:**

- **Revisiones por pares de software** se llevan a cabo por el autor del producto de trabajo, o por uno o más colegas del autor, para evaluar el contenido técnico y / o la calidad de la obra.
- **Opiniones de administración de software** se llevan a cabo por los representantes de la dirección para evaluar el estado de los trabajos realizados y para tomar decisiones con respecto a las actividades aguas abajo.

- **Exámenes de auditoría de software** se llevan a cabo por personal externo para el proyecto de software, para evaluar el cumplimiento con las especificaciones, normas, acuerdos contractuales, u otros criterios.

Auditoría

Una auditoría es una inspección o examen que se realiza en una empresa u organización para verificar el correcto funcionamiento de la misma en distintos aspectos. Se suele llevar a cabo por personas ajenas a la entidad que está siendo objeto del examen, denominados auditores. En el caso de la auditoría interna de calidad, es la que se lleva a cabo en las empresas para controlar la perfecta implantación de la norma ISO 9001 en la organización. La realización de estas auditorías internas de un modo periódico es un requisito dentro de la propia norma ISO 9001 y se entiende como un componente básico para comprobar el grado de desempeño de la norma.

Con una auditoría interna de calidad, además de evaluar el funcionamiento de los sistemas implantados, se identifican las desviaciones que pueden existir e introducir medidas correctivas. Esto permite asegurar que el sistema de gestión de calidad sigue las directrices establecidas y que, por tanto, los clientes pueden seguir depositando su confianza en los productos o servicios de la empresa [18].

Tipos de auditorías de calidad [18]

- **Auditoría interna:** Es realizada por agentes propios de la empresa en los que se tiene cierto grado de confianza. Con esta auditoría se aportan datos interesantes que pueden delatar errores que posteriormente generan acciones correctivas, prevención y mejora.
- **Auditoría externa:** En ella actúan clientes de la organización o por personas que trabajan de parte de éste. En muchos de los casos, se basa principalmente en conocer las expectativas de los clientes de primera mano, es decir encuestándolos o preguntándoles acerca de la calidad de nuestro producto. También puede ser realizada por organizaciones competentes en el área de la certificación de calidad que interviene en los procesos de la empresa a fin de constatar la misma.

Objetivos de las auditorías de calidad [2].

- Se centra en la observación y análisis del sistema con miras a mejorar la calidad en dichos sistemas, procesos y organización.
- Se centra en la observación de condiciones o situaciones de cualquier naturaleza, a veces dadas por errores o desperfectos, con un grado de evaluación y detalle minucioso proporcional a la importancia de la calidad de los procesos que se requiere.
- Se centra en el estudio de muestras extraídas de pruebas de producción en los procesos.
- Se realizan comparaciones de acuerdo a las referencias existentes de la producción anterior que permitan determinar diferencias, similitudes y/o mejoras.

Lista de chequeo

Es un formato creado para realizar actividades repetitivas, controlar el cumplimiento de una lista de requisitos o recolectar datos ordenadamente y de forma sistemática. Se usa para hacer comprobaciones sistemáticas de actividades o productos asegurándose de que el trabajador o inspector no olvide nada importante [19].

Principales objetivos de las listas de chequeo:

- Realizar actividades en las que es importante que no se olvide ningún paso y/o deben hacerse las tareas con un orden establecido.
- Realiza inspecciones donde se debe dejar constancia de cuáles han sido los puntos inspeccionados.
- Verificar o examinar artículos.
- Examinar o analizar la localización de defectos. Verificar las causas de los defectos.
- Recopilar datos para su futuro análisis.

1.2 Sistemas homólogos

En la actualidad son varias las compañías que en distintos lugares del mundo han desarrollado sistemas automatizados para la gestión de auditorías, los cuales son ampliamente utilizados en prestigiosas empresas. Algunos de estos son:

MKinsight ™

Es un sistema integral de gestión de auditoría altamente configurable, potente y fácil de usar. A partir de los auditores individuales al estado MKinsight Auditoría ™ Instituciones es fácil de usar, sencillo de implementar y asequible sea cual sea el tamaño de su equipo de auditoría [20].

Características principales:

Base de datos integrados a la perfección individual. Totalmente configurable terminología y flujo de trabajo. Seguridad completa de datos cifrados. Interfaz de usuario amigable. Programa de Desarrollo Robusto. Multi-Lenguaje.

Funciones Claves:

- Planificación de la auditoría.
- Programación de Auditoría.
- Auditoría de Gestión.
- Informar el Rendimiento.
- Gestión del riesgo.
- Información Integral.
- Tiempo de grabación.
- Recomendación

- Acción de Seguimiento.
- Cuestionarios en línea.
- Bibliotecas.

Gespro

Esta Suite para la Dirección Integrada por Proyectos se presenta en un modelo de negocios basado en servicios que combinan el uso de una solución informática para la dirección integrada de proyectos y un sistema de formación especializada en gestión de proyectos. Esta combinación posibilita no sólo la informatización de las organizaciones sino también la mejora integral de los procesos de planificación, control y seguimiento de proyectos[21].

Características del ecosistema [21]:

- La planificación y el control y seguimiento de los proyectos y de los recursos asociados a los mismos, alineadas con la proyección estratégica de las organizaciones.
- La planificación del alcance y el tiempo, de la gestión de recursos humanos y sus competencias, de la gestión de riesgos, así como la financiera de los proyectos y de logística y la de gestión de recursos compartidos.
- El control y seguimiento de proyectos a través de la combinación de un cuadro de mando integral y un sistema para el diseño dinámico de reportes que permiten el acceso a la información del estado de proyectos con diferentes niveles de detalles de la información.
- Gestión documental con facilidades para la gestión del expediente de los proyectos.

Funcionalidades principales [21] :

- **Entorno web** con punto acceso único a todas las funcionalidades de la suite.
- **Gestión de Portafolios de Proyectos:** Permite la gestión de portafolios de proyectos. Plan de proyecto, la ejecución del plan de proyecto y en el control integrado de cambios.
- **Gestión de la Calidad:** Permita el control y seguimiento de defectos, solicitudes de cambios y no conformidades en el desarrollo de los proyectos con facilidades para el análisis de las principales deficiencias.
- **Gestión de Riesgos:** Procesos para la planificación de la gestión de riesgos, identificación de los riesgos, análisis cualitativo de los riesgos, análisis cuantitativo de los riesgos, planificación de las respuestas a los riesgos, y monitoreo y control de los riesgos.

SE Audit

SE Audit es un sistema 100% WEB, multiusuario y multidepartamental, que incorpora herramientas de: organización, clasificación y búsqueda. Características que dan al producto simplicidad, agilidad,

confiabilidad y eficiencia. La inversión en SoftExpert dependerá del tamaño de la empresa y la metodología de implementación [22].

El software proporciona el trabajo en equipo, a través de un práctico mecanismo de control de pendencias, denominado Team Workflow, que notifica vía e-mail, el momento exacto, a los responsables por actividades pendientes, exhibe estas pendencias y autoriza el registro de las firmas electrónicas y demás informaciones aplicables a cada etapa del proceso. Este mecanismo asegura la agilidad y el compromiso con el cumplimento de los plazos en todas las etapas del proceso de auditoría [22].

SE Audit le da a la organización total flexibilidad para establecer:

- Requisitos normativos que serán auditados.
- Criterios para cualquier tipo de auditoría adoptada por la organización.
- Equipos de auditores internos y externos que quedarán responsables por la conducción del proceso de auditoría.
- Cursos y competencias (CHA - Conocimientos, Habilidades y Aptitudes) necesarias para que cada auditor, sea él interno o externo, pueda asumir una responsabilidad en la realización de la auditoría, tal como anexar registros que comprueben su cualificación.
- Etapas las cuales cada clasificación de auditoría será sometida.

De las herramientas analizadas anteriormente se pueden mencionar algunas funcionalidades que tienen en común y coinciden además con características que son deseadas para el sistema de gestión de auditorías, estas particularidades son:

- Administración de documentos y archivos
- Notificaciones e-mail
- Sistema de acceso basado en roles
- Generación de reportes

Durante la investigación se estudiaron los sistemas MKinsight ™, Gespro y SE Audit concluyéndose que ninguno puede ser adaptado a las necesidades de la investigación, por lo que es necesario desarrollar el módulo de gestión de auditorías para el GDA eXcriba. Por otra parte, los sistemas que realizan mecanismo de seguimiento y control: MKinsight ™ y SE Audit son softwares propietarios, lo que constituye un aspecto negativo para el país y para la UCI, pues estos se encuentran inmersos en un proceso de independencia tecnológica y traerían consigo gastos innecesarios por la obtención de las licencias. Por otra parte, algunos sistemas no se pueden desplegar en GNU/Linux. Para una mejor comprensión se muestra una comparación de estos sistemas en la siguiente tabla:

Sistema	Web	Sistema Operativo	Mecanismo de seguimiento	Licencia
MKinsight ™	SI	Microsoft Windows	SI	Privativa
Gespro	SI	Multiplataforma	No	Libre
SE Audit	Si	Multiplataforma	Si	Privativa

Tabla 1. Comparación entre los sistemas estudiados. Fuente: elaboración propia.

1.3 Metodología de desarrollo de software

Las metodologías de desarrollo de software ofrecen una guía para encaminar de manera correcta el trabajo y así lograr un producto de calidad. Estas van orientando cómo separar el proyecto por etapas, las tareas que se irán haciendo en dichas etapas y las herramientas que se podrán emplear[23].

Una metodología de desarrollo de software se refiere a un framework que es usado para estructurar, planear y controlar el proceso de desarrollo en sistemas de información. A lo largo del tiempo, una gran cantidad de métodos han sido desarrollados diferenciándose por su fortaleza y debilidad. Uno de los aspectos más importantes para elegir una metodología para el desarrollo de un software, es determinar el alcance que tendrá, saber cómo es el proyecto que se va a desarrollar y cuál es el tiempo con que se dispone[23].

Metodología AUP-UCI

La metodología de trabajo para guiar el cumplimiento de los objetivos propuestos fue “Metodología de Desarrollo para la Actividad Productiva de la UCI (AUP-UCI)”, pues es la metodología que se encuentra definida en el proyecto, la cual tiene como eje principal la aplicación de técnicas ágiles incluyendo[23]:

- Desarrollo dirigido por pruebas.
- Modelado ágil.
- Gestión de cambios ágil.
- Refactorización de Base de datos para mejorar la productividad.

Esta metodología tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, para ello para ello se seguirá el modelo CMMIDEV v1.36, el cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora.

Las fases de trabajo, las disciplinas y los escenarios para modelar el sistema en los proyectos se describen a continuación[23]:

Fases:

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.

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.

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

Disciplinas:

Modelado de 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.

Requisitos: el esfuerzo principal en la disciplina 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.

Análisis y diseño: 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.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.

Implementación: en la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.

Pruebas internas: 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.

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.

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.

Roles definidos por AUP

- Administrador de proyecto: maneja a los miembros, planea, maneja y asigna los recursos.
- Ingeniero de procesos: desarrolla, adapta y apoya sus materiales del proceso del software.
- Desarrollador: escribe, testea y construye software.
- Administrador de Base de Datos: diseña, prueba, desarrolla, y apoya los esquemas de la BD.
- Modelador ágil: crea y desarrolla modelos, bosquejos o los archivos de la herramienta CASE.
- Administrador de la configuración: es responsable de proporcionar la infraestructura total y el ambiente del CM al equipo de desarrollo.
- Administrador de pruebas: responsable del éxito de la prueba, incluyendo el planeamiento, la gerencia, y la defensa para la prueba y las actividades de la calidad.
- Probador: encargado de ejecutar las pruebas.

Escenarios para modelar el sistema en los proyectos:

Escenario no 1: Proyectos que modelen el negocio con Caso de Uso del Negocio (CUN) solo pueden modelar el sistema con Caso de Uso del Sistema (CUS).
Escenario no 2: Proyectos que modelen el negocio con Modelo Conceptual solo pueden modelar el sistema con CUS.
Escenario no 3: Proyectos que modelen el negocio con Descripción de Proceso de Negocio (DPN) solo pueden modelar el sistema con Descripción de Requisitos por Proceso (DRP).
Escenario no 4: Proyectos que no modelen negocio solo pueden modelar el sistema con Historias de Usuario (HU).
Para la realización de este trabajo se empleó el escenario No. 2 de la metodología AUP-UCI. El cual 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 se modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información [21].Además se abarcan las disciplinas Requisitos, Análisis y diseño, Implementación y Pruebas internas. La disciplina Requisitos se aplica al realizar el levantamiento de requisitos funcionales y no funcionales, modelándolos utilizando la variante descripción de casos de usos. Al modelar el sistema mediante el diagrama de dominio, además de especificar la arquitectura que se utiliza, se abarca la disciplina de Análisis y diseño. El desarrollo de la solución cumple con lo planteado en la disciplina de Implementación y la creación de los casos de prueba abarca la disciplina Pruebas internas. Además se selecciona la fase de ejecución para el desarrollo de la propuesta de solución.

1.4 GDA eXcriba o Gestor de Documentos Administrativos Xabal eXcriba 4.0

El GDA eXcriba o Gestor de Documentos Administrativos eXcriba, es un sistema basado en el ECM Alfresco, que lleva su contenido a lo largo de su ciclo de vida, permitiendo la gestión de los múltiples documentos de trabajo, ya sean documentos de archivo o administrativos. El Gestor de Documentos Administrativos Xabal eXcriba 4.0 rige su funcionamiento por la Norma ISO 15489 para la gestión documental y usa como núcleo de su sistema informático el Gestor de Contenido Empresarial (ECM) Alfresco Community en su versión 5.2f. Es un producto informático genérico, que se ha propuesto desde su comienzo el cumplimiento cabal de normas, desde su concepción hasta la personalización y despliegue del producto mediante soluciones a la medida en distintas organizaciones [24].

El sistema se compone de varios módulos, los cuales se enuncian a continuación sus características técnicas :

Módulo	Detalles técnicos	Descripción
ECM Alfresco	Desarrollado en Java. Se utilizó la liberación (release) "Community" de la versión 3.0 del Administrador de Contenidos Empresariales Alfresco[1].	Núcleo del producto de software eXcriba, modificado en correspondencia con las necesidades propias del GDA eXcriba.
Interfaz Web	Aplicación web desarrollada con PHP v5.3.0. Se utilizaron los frameworks CodeIgniter v1.7.2 del lado del servidor y JQuery v1.3.2 del lado del cliente.	Interfaz web sencilla de utilizar que permite la interacción de los usuarios correspondientes con el repositorio documental que proporciona el ECM Alfresco.
Módulo para la Gestión de Usuarios y Grupos de Usuarios	Aplicación de escritorio o "Desktop Application" desarrollada en Java.	Aplicación de escritorio que facilita la creación y eliminación de usuarios y grupos de usuarios al repositorio documental.
Módulo para la Gestión de Tipologías Documentales	Aplicación de escritorio o "Desktop Application" desarrollada en Java.	Aplicación de escritorio que facilita la gestión de las tipologías documentales correspondientes al tipo de información que se maneja en el repositorio documental, para una mejor estructuración y clasificación de los documentos dentro del repositorio.

Figura #2. Módulos de eXcriba[25].

1.5 Alfresco

Alfresco es un Sistema de Administración de Contenidos Empresariales, libre, basado en estándares abiertos y de escala empresarial. Fue desarrollado en el 2005 por el equipo más experimentado del sector proveniente de Documentum, Vignette e Interwoven[26].

Alfresco permite la gestión del ciclo de vida de los contenidos, organiza y facilita la gestión de contenidos de todo tipo, facilita el trabajo colaborativo y provee un repositorio fuente basado en últimas tecnologías y estándares. Cuenta con dos versiones disponibles, una versión Community,

gratuita, con licencia GPL7 y una versión Enterprise, que requiere una suscripción anual de pago y permite disponer de garantía del fabricante, así como a las actualizaciones intermedias.

La arquitectura de Alfresco está basada en un repositorio de contenidos, gestionando el almacenamiento de la información en cualquiera de sus formatos nativos, indexando y categorizando los contenidos para su rápida búsqueda y localización, y almacenando los metadatos en un sistema gestor de base de datos (SGBD). Alfresco propone una arquitectura state-of-the-art8 usando Spring, Hibernate, Lucene, servicios web, REST9, entre otras tecnologías. Dentro de las facilidades que provee Alfresco se pueden encontrar[26]:

- Inmediata localización de documentos.
- Búsqueda precisa de los documentos por contenido o nombre.
- Drástico recorte del espacio de almacenamiento y reaprovechamiento del mismo.
- Eliminación de los documentos duplicados.
- Total control y seguridad de acceso y alteración de documentos.
- No existen documentos extraviados o perdidos.
- Mejora de la calidad y el servicio ofrecido.

ECM Community Alfresco 5.2f

La actual versión de Alfresco presenta las siguientes mejoras:

- Plataforma al aire libre.
- Herramientas de soporte en la consola de administración.
- Eliminación de GhostScript.
- Nuevas API REST.
- Nuevo explorador de API REST.
- Mejoras administrativas.
- Facetas de selección múltiple.
- Acciones de archivos a granel en los resultados de búsqueda

Alfresco incorpora un metamodelo en su estructura, es decir, la capacidad para declarar nuevos modelos de contenido. Básicamente un modelo contenido es la estructura que define qué información acompañará a los documentos digitales. Su definición permite dotar de semántica a los documentos para que posteriormente se puedan implementar búsquedas y procesos eficientes e “inteligentes” [25].

1.5.1 Auditorías en Alfresco

Una de las funcionalidades con la que habitualmente debe contar un Sistema Gestor de Contenidos (SGC), es la capacidad de auditar las acciones realizadas sobre el repositorio de documentos, y la obtención de estadísticas sobre los movimientos realizados en los documentos almacenados.

Alfresco incluye una estructura que posibilita crear una pista de auditoría que permite recopilar la información necesaria para obtener estadísticas e informes sobre los movimientos en el repositorio de contenidos. Esta estructura aprovecha la arquitectura Spring de la aplicación, para utilizar un conjunto de interceptores que permiten controlar los servicios de Alfresco a nivel de método. Alfresco incluye, entre sus funciones, la capacidad de auditar las acciones realizadas sobre el repositorio de documentos, y la obtención de estadísticas sobre los movimientos realizados en los documentos almacenados [13]. La auditoría en Alfresco no está basada en la definición de políticas, a través de comportamientos añadidos a cada uno de los servicios. Es una auditoría de alto nivel aplicada en la capa de servicios públicos del repositorio (Content Services). En esta capa las acciones del usuario o la aplicación contra el repositorio generan transacciones y eventos de seguridad. Las acciones fallidas generan excepciones que se registran como otra transacción. Si ha habido vuelta atrás, no serán grabadas, sino se reflejarán en la base de datos. Las transacciones, eventos de seguridad y excepciones son llamadas al bean (código), capturadas por el interceptor de auditoría. Este interceptor genérico soporta auditoría para cualquier servicio público configurado en el descriptor public-services-context.xml con la anotación PublicService, concediéndole acceso de lectura. También se configura cada método para describir su comportamiento, definiendo si es auditable o no, y qué será auditado, mediante[26]:

- Clave de nodo (key node reference).
- Clave de cadena de descripción internacional: es un argumento o valor de retorno opcional que filtra auditoría basada en type/aspect/path.
- Posición de argumentos y nombres de parámetros de los métodos, para filtrar por valores especificados.

Se puede activar o desactivar auditoría para un servicio (por defecto, o todos los servicios) y el método de un servicio. También se puede incluir o excluir para el método de un servicio tipos de objetos, objetos basados en rutas, instancias de objetos e instancias de objetos basados en valores de propiedades[26].

Para realizar la configuración de la auditoría existe un fichero de nombre auditConfig.xml donde se pueden definir qué servicios y qué métodos se van a auditar, en el momento de prescindir de un servicio se puede crear un bloque donde se indican cuáles son los métodos a auditar dentro de ese servicio, o se puede auditar simplemente todos los métodos de ese servicio, poniendo el valor all al atributo mode del elemento service. Observando que el Alfresco cuenta con una estructura integrada a él, que le permite crear registros o trazas de auditorías de manera automática, solo bastaría con especificarle que servicios y métodos dentro de los auditables por esta aplicación, son los que se desean auditar [13].

1.6 Lenguajes de programación

Para la implementación de las funcionalidades en Alfresco el equipo de desarrollo utilizó JavaScript 1.6 para el desarrollo en el lado del servidor y del cliente, en este último también se hizo uso del lenguaje HTML 5 (HyperText Markup Language) y XML 1.0 (Extensible Markup Language) para realizar los modelos de contenido y los servicios de datos, y para el modelado de procesos se utilizó Lenguaje de modelado de procesos BPMN v 2.0.

Lenguaje de modelado de procesos BPMN v 2.0

Business Process Model and Notation (BPMN), en español Modelo y Notación de Procesos de Negocio, es una notación gráfica estandarizada que permite el modelado de procesos de negocio, en un formato de flujo de trabajo (workflow). BPMN fue inicialmente desarrollada por la organización Business Process Management Initiative (BPMI), y es actualmente mantenida por el Object Management Group (OMG), después de la fusión de las dos organizaciones en el año 2005[28].

Su versión actual es la v2.0.2, publicada en 2013, que contiene una mejora menor sobre la versión del 2011 v2.0, respecto a formatos de intercambio.

El principal objetivo de BPMN es proporcionar una notación gráfica estándar que sea fácilmente legible y entendible por parte de todos los involucrados e interesados del negocio (stakeholders). Entre estos interesados están los analistas de negocio (quienes definen y redefinen los procesos), los desarrolladores técnicos (responsables de implementar los procesos) y los gerentes y administradores del negocio (quienes monitorizan y gestionan los procesos). En síntesis, BPMN tiene la finalidad de servir como lenguaje común para cerrar la brecha de comunicación que frecuentemente se presenta entre el diseño de los procesos de negocio y su implementación, esto facilitará una mejor comprensión de lo que se realiza[28].

El modelado en BPMN se realiza mediante diagramas muy simples con un conjunto muy pequeño de elementos gráficos. Con esto se busca que para los usuarios del negocio y los desarrolladores técnicos sea fácil entender el flujo y el proceso.

Las cuatro categorías básicas de elementos son:

- Objetos de Flujo: Eventos, Actividades, Rombos de control de flujo (gateways).
- Objetos de Conexión: Flujo de Secuencia, Flujo de Mensaje, Asociación.
- Carriles de nado (swimlanes): Piscina, Carril.
- Artefactos: Objetos de Datos, Grupo, Anotación.

Estas cuatro categorías de elementos nos dan la oportunidad de realizar un diagrama simple de procesos de negocio (en inglés Business Process Diagram, BPD). En un BPD se permite definir un tipo personalizado de objeto de flujo o un artefacto, si con ello se hace el diagrama más comprensible [29].

HTML 5

Esta especificación define la quinta mayor revisión del núcleo del lenguaje de la Word Wide Web: el lenguaje de marcado de hipertexto (HTML). En esta versión, se introducen nuevas características para ayudar a los autores de aplicaciones Web, se introducen nuevos elementos basados en la investigación de las prácticas prevalecientes de autoría, y se ha prestado especial atención a la definición de los criterios de conformidad clara para los agentes de usuario en un esfuerzo por mejorar la interoperabilidad.

El código HTML es un lenguaje muy simple y fácil de interpretar en términos generales, por ejemplo: negrita indica que los navegadores web visuales deben mostrar el texto en negrita; entonces podemos decir que estas marcas o etiquetas son como instrucciones a las que obedece el navegador para determinar la forma en la que debe aparecer. Este lenguaje se evidencia dentro de la solución en la presentación al cliente o sea en las vistas de las páginas web[28].

JavaScript 1.6

Es un lenguaje de programación interpretado, o sea, no requiere compilación. Es utilizado especialmente en páginas web embebido en el código HTML o similares. La mayoría de los navegadores pueden interpretar los códigos JavaScript incluidos en las páginas web.

Por existencia de distintas versiones de JavaScript incompatibles, el World Wide Web Consortium (W3C) diseñó un estándar llamado DOM (Document Object Model) que incorpora Internet Explorer 6 en adelante, Opera versión 7 en adelante y Mozilla con el objetivo de lograr mejor compatibilidad entre navegador y página web[28].

Con JavaScript se pueden extender las posibilidades de las páginas web como, por ejemplo, evitar que se pueda copiar el texto de una página, botones para agregar automáticamente una página a favoritos, crear barras de scroll, abrir popups, cambiar el puntero del mouse, rotar banners, validar formularios, etc. En la solución se pone de manifiesto este lenguaje en la implementación de las páginas web que serán vistas por el cliente, además de los formularios y las tablas [30].

XML 1.0

XML significa lenguaje de marcas generalizado (Extensible Markup Language). Es un lenguaje usado para estructurar información en un documento o en general en cualquier fichero que contenga texto, como por ejemplo ficheros de configuración de un programa o una tabla de datos. Ha ganado muchísima popularidad en los últimos años debido a ser un estándar abierto y libre, creado por el Consorcio Word Wide Web, W3C, en colaboración con un panel que incluye representantes de las principales compañías productoras de software[28].

Es un lenguaje de meta-marcado**,** ya que se usa para describir metadatos a través de etiquetas de marcado. Provee un conjunto de reglas simples para diseñar formatos de texto que permiten estructurar los datos. Provee un método uniforme para describir e intercambiar datos estructurados**.**

Describe estructura y semántica, no formato**.** Este lenguaje no solo presenta el contenido de un texto, sino que lo dota de significado, además de que el contenido de un documento es separado de cualquier noción de presentación[28].

Es un lenguaje extensible, es decir, que, a diferencia del HTML, permite definir etiquetas y una estructura propia del documento, a través de un conjunto de reglas definidas en un documento DTD (Document Type Definition). Es un formato abierto que puede ser interpretado por cualquier aplicación que reconozca su lenguaje [31] .

1.7 Herramientas de desarrollo Visual Paradigm v 8.0

CASE es un acrónimo para Computer Aided Software Engineering, que en español significa Ingeniería de Software Asistida por Computadora. Se considera una herramienta CASE a un grupo de métodos, utilidades y técnicas que facilitan en la automatización del ciclo de vida del desarrollo de software, en su totalidad o en alguna de sus fases. Estas herramientas facilitan la construcción de prototipos, el desarrollo conjunto de aplicaciones, perfeccionan y estandarizan la documentación. Además, aumentan la portabilidad de las aplicaciones, facilitan la reutilización de componentes y viabilizan un desarrollo y refinamiento visual de las aplicaciones mediante el manejo de gráficos. Al utilizar una herramienta CASE en el desarrollo de un proyecto se garantiza un aumento en la calidad y, por consiguiente, un aumento de la productividad. Para el desarrollo de la solución se utilizó Visual Paradigm for UML 8.0[32].

Visual Paradigm es una herramienta multiplataforma, característica que la favorece y que viene muy acorde a la migración al software libre que lleva a cabo Cuba. Es una herramienta UML profesional que soporta el ciclo de vida del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Se integra con varios IDEs (Entorno de Desarrollo Integrado) y soporta múltiples usuarios trabajando sobre un mismo proyecto. Ofrece interoperabilidad entre diagramas, ya que permite a partir de un diagrama obtener otro que guarde relación con el mismo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación [33].

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto que incluye diversas características tales como la herencia múltiple, funciones, restricciones, etc. El desarrollo de PostgreSQL no es manejado por una empresa o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre o apoyados por organizaciones comerciales. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group) [34] .

Visual Studio Code v1.31.0

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y MacOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. Es compatible con varios lenguajes de programación y un conjunto de características que pueden o no estar disponibles para un idioma dado[32].

VS Code es un editor de código fuente sofisticado que admite muchas funcionalidades practicas al momento de trabajar con el código. Estas son algunas de ellas:

- Lenguajes de programación: La edición de código no está limitada para C# y VB (lenguajes propietarios de Microsoft) si no que de nueva cuenta el Open Source está en el paquete: Java, Go, C, C++, Ruby, Python, PHP, Perl, JavaScript, Groovy, Swift, PowerShell, Rust, DockerFile, CSS, HTML, XML, JSON, Lua, F#, Batch, SQL.

- Multiplataforma: Fue creado y diseñado para que funcione en los tres sistemas operativos mayormente utilizados: Windows, Linux y Mac OS.
- Plugins: VS Code es una herramienta que se actualiza constantemente, tiene la posibilidad de adaptar plugins para trabajar con el cómputo en la nube de Microsoft Azure y desplegar proyectos directamente.
- Acerca de los Proyectos: Las plantillas que se encuentran en Visual Studio ayudan a construir la estructura base de los proyectos, con VS Code también podemos crearlos, pero siempre desde cero.

Por lo anteriormente expuesto se selecciona el Visual Studio Code v1.31.0 como editor de código por la flexibilidad que brinda para el desarrollo de la investigación en curso como son las extensiones pues desde el mismo editor se puede acceder a una gran variedad de extensiones y complementos que permitirá dotarle de las funciones y características necesarias para el desarrollo del módulo [35].

1.8 Conclusiones del capítulo

Con la investigación realizada en el presente capítulo se concluye que:

- El análisis de los conceptos fundamentales relacionados con las auditorías, gestión documental y sistemas de gestión documental, permitió mayor claridad y dominio de las terminologías involucradas al problema identificado.
- Al estudiar las posibles soluciones que existen, sus ventajas y desventajas, se concluyó que debe ser realizado un módulo para gestionar las auditorías y se pueda mantener un control y seguimiento a las auditorías de calidad realizadas a los expedientes electrónicos

- El estudio de las tecnologías, herramientas, lenguajes y metodologías permite conocer las bondades de estas para lograr mayor acomplamiento de la solución con el sistema XABAL eXcriba 4.0.

CAPÍTULO II

Capítulo 2: Propuesta de solución del módulo de gestión de auditorías a expedientes.

En el presente capítulo se abordan temas relacionados con la propuesta de solución para responder a la situación problemática en cuestión. Además, se realizan varias tareas fundamentales, como son obtener un listado de las funcionalidades que el sistema que se analiza y diseña deberá tener, llamadas comúnmente requisitos funcionales. Se representan las acciones y las relaciones con las personas que las realizan, y lograr así un consenso con el cliente, se presentan los resultados obtenidos en el escenario No. 2 correspondiente a la metodología AUP-UCI, donde se exponen los artefactos generados durante su puesta en práctica.

2.1 Descripción de la propuesta de solución.

El sistema informático que se propone, constituye un factor fundamental para el proceso de auditoría de expedientes. La propuesta de solución permitirá guardar todos los resultados encontrados en cada auditoría y luego se genera el dictamen técnico como resumen de la misma. El módulo presenta varias funcionalidades, entre las que se encuentra la de crear una nueva auditoría, así como asignar los permisos a los diferentes artefactos que lo componen sin necesidad de contactar al equipo de desarrollo. Este pretende satisfacer la necesidad de gestionar los permisos sobre las auditorías de expedientes de forma automática en el momento de actualizar la lista de miembros del área, satisfaciendo así las necesidades actuales del proceso.

El sistema debe permitir crear una o más auditorías, luego de registrarse la solicitud se copia en el espacio de trabajo del usuario una carpeta con todos los documentos necesarios. Los documentos se copian dependiendo del tipo de auditoría que se va a realizar. Posteriormente se emite la minuta de reunión de inicio, documento que registra las principales cuestiones tratadas en la reunión. Al finalizar la realización de la minuta de reunión se revisa el expediente de proyecto y se realizan las listas de chequeo, dependiendo del tipo de auditoría. Las listas de chequeo describen los indicadores necesarios para evaluar los documentos. Luego se genera un dictamen técnico, este documento es la constancia de que la proyecto le fueron realizadas la auditoría. Además, el sistema es capaz de administrar los usuarios, gestionando de cada uno sus permisos. Los usuarios desempeñarán rol dependiendo de la lista de datos definida por calidad. En esta lista de datos se define que permiso debe tener cada rol para cada uno de los artefactos del expediente, el mismo puede crear contenido y modificarlo siempre que sea de su propiedad.

2.2 Modelo Conceptual

Un modelo conceptual es una representación de conceptos de un dominio del problema. La designación de modelo conceptual ofrece la ventaja de subrayar fuertemente una concentración en los conceptos del dominio, no en las entidades del software[23].

El modelo conceptual permite la descomposición global del conjunto de la información de una organización mediante la utilización de los conceptos más importantes que ella maneje, son conceptos del mundo real, es decir son modelos de análisis[23] . Este modelo se describe mediante diagramas de UML (especialmente mediante diagramas de clases). Estos diagramas permiten una primera aproximación a la realidad que se quiere manejar con un sistema informático, mostrando las relaciones que existen entre los conceptos mediante asociaciones. En la Figura #3 se presenta el modelo conceptual de la propuesta de solución.

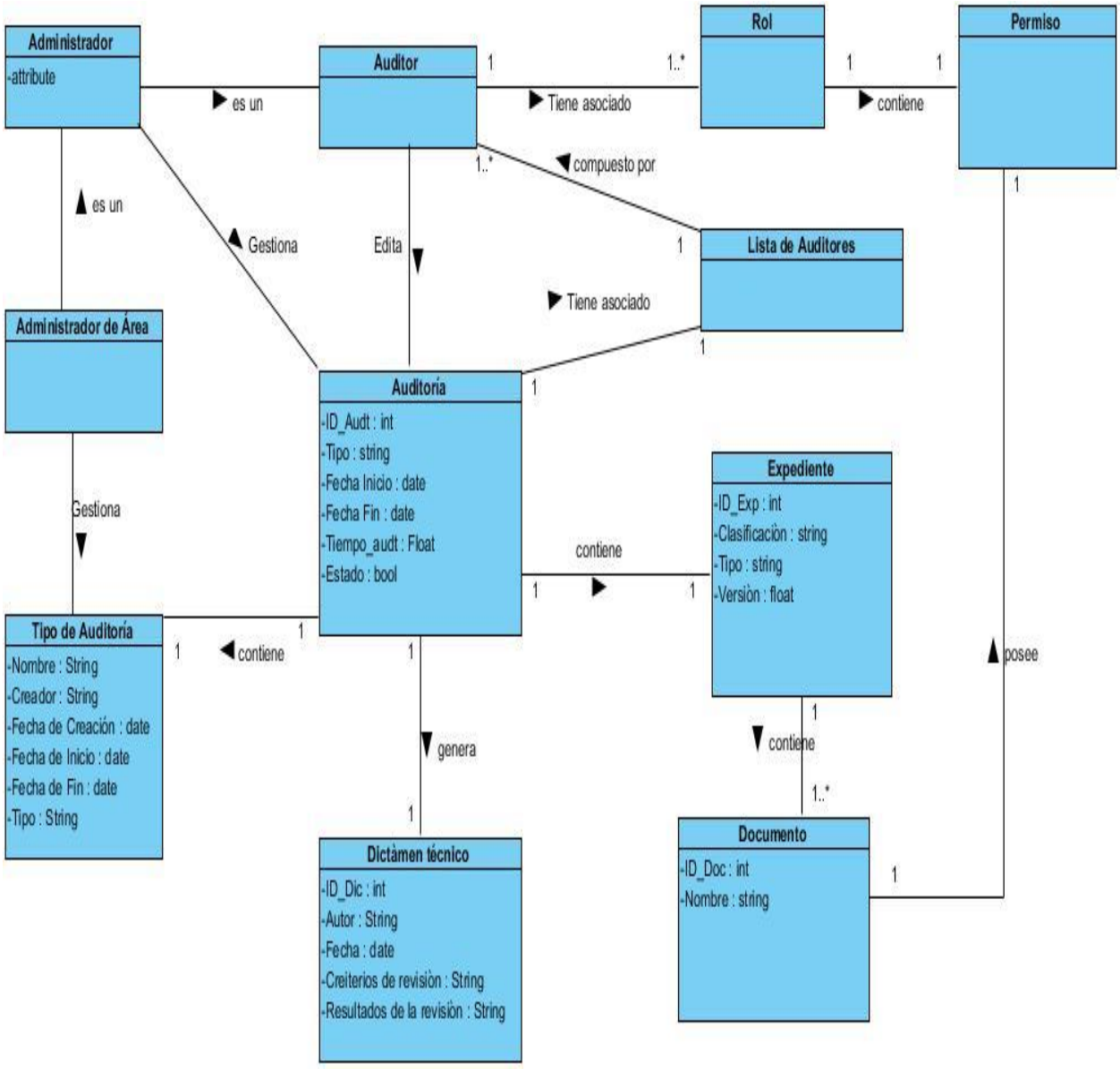


Figura #3 Modelo Conceptual.

Conceptos identificados:

Tipo de auditoría: es la estructura lógica que contiene información de las auditorías a la cual puede acceder el administrador del área.

Auditor: Persona con la competencia para llevar a cabo una auditoría.

Administrador: Posee control total del Sistema, por lo que puede realizar cualquier acción.

Administrador del Área: Es un administrator que posee permisos para gestionar los tipos de auditorías.

Expediente: Es la estructura lógica de carpetas que contiene documentos referentes a un proyecto de software.

Documento: Archivo asociado al Proyecto el cuál contiene información del mismo.

Dictamen Técnico: Documento que contiene toda la información del resultado de la auditoría.

Lista de auditores: es el listado donde se almacenan auditores.

Auditoría: es la estructura lógica que contiene información de las auditorías, a la cual accede el auditor para la creación de una nueva auditoría.

Rol: Cargo que puede ocupar el auditor.

Permiso: Nivel de acceso que tiene ese Rol.

2.3 Definición de Requisitos

Un requisito es una “condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado”[36]. También se define como las condiciones que debe cumplir o poseer un sistema o uno de sus componentes para satisfacer un contrato, una norma o una especificación. Puede verse como una declaración abstracta de alto nivel de un servicio que el sistema debe proporciona[36].

2.3.1 Técnicas para la captura de los requisitos

En el proceso de desarrollo de un sistema, sea o no para la web, el equipo de desarrollo se enfrenta al problema de la identificación de requisitos. La definición de las necesidades del sistema es un proceso complejo, pues en él hay que identificar los requisitos que el sistema debe cumplir para satisfacer las necesidades de los usuarios finales y de los clientes. Para realizar este proceso, no existe una única técnica estandarizada y estructurada que ofrezca un marco de desarrollo que garantice la calidad del resultado. Existe en cambio un conjunto de técnicas, cuyo uso proponen las diferentes metodologías para el desarrollo de aplicaciones web. Se debe tener en cuenta que la selección de las técnicas y el éxito de los resultados que se obtengan, depende en gran medida tanto del equipo de análisis y desarrollo, como de los propios clientes o usuarios que en ella participen[36].

Dentro de las técnicas de captura de requisitos existentes, se utilizaron para la obtención de los requisitos de esta investigación las siguientes:

- **Análisis de sistemas existentes:** Durante la investigación se realizó un estudio de aplicaciones similares a la que se necesitaba obtener, en las cuales se observaron los diseños de sus interfaces, las funcionalidades que ofrecen, el grado de dificultad a la hora de interactuar con la aplicación, los colores que emplean, entre otros rasgos importantes que contribuyeran a obtener un producto con la mejor calidad.
- **Tormenta de ideas:** Se efectuaron reuniones con el equipo de desarrollo del proyecto eXcriba y a partir de un conjunto de ideas propuestas principalmente por los analistas, se definieron los primeros requisitos con los que debería cumplir el módulo. Se estipuló conveniente una nueva reunión incluyendo algunos integrantes del proyecto para acumular más opiniones y definir una versión final de los requisitos del módulo.
- **Entrevistas:** Se realizaron entrevistas al equipo de desarrollo del GDA eXcriba en las cuales se hicieron una serie de preguntas con el objetivo de obtener toda la información posible sobre las funcionalidades del módulo a desarrollar.

2.3.2 Requisitos Funcionales

Los requisitos funcionales son tareas, capacidades o condiciones que el sistema debe cumplir. Dichas tareas dirigen un entendimiento de cuál será el impacto del software sobre el negocio, qué quieren los clientes y cómo los usuarios finales interactuarán con dicho software. Deben ser lo más completos, claros y precisos posibles[37].

A partir de las técnicas de capturas de requisitos realizadas y las actuales se identificaron los siguientes requisitos.

- RF_1: Crear auditoría.
- RF_2: Eliminar auditoría.
- RF_3: Mostrar auditoría.
- RF_4: Modificar auditoría.
- RF_5: Filtrar por nombre de auditoría.
- RF_6: Filtrar por creador de auditoría.
- RF_7: Filtrar por fecha de creación de auditoría.
- RF_8: Filtrar Por Tipo de Expediente.
- RF_9: Filtrar Por Clasificación del Expediente.
- RF_10: Filtrar por Versión.
- RF_11: Crear tipo de auditoría.
- RF_12: Eliminar tipo de auditoría.
- RF_13: Mostrar detalles del tipo de auditoría.

- RF_14: Modificar tipo de auditoría.
- RF_15: Filtrar por nombre del tipo de auditoría.
- RF_16: Filtrar por creador del tipo de auditoría.
- RF_17: Filtrar por fecha de creación de auditoría.
- RF_18: Filtrar por fecha de inicio del tipo de auditoría.
- RF_19: Filtrar por fecha fin del tipo de auditoría.
- RF_20: Filtrar por tipo de auditoría.
- RF_21: Adicionar documentos asociados.
- RF_22: Eliminar documentos asociados.
- RF_23: Modificar documentos asociados.
- RF_24: Buscar documentos asociados.
- RF_25: Asignar lista de auditores a una auditoría.
- RF_26: Crear lista de auditores.
- RF_27: Editar lista de auditores.
- RF_28: Ver detalles de la lista de auditores.
- RF_29: Eliminar lista de auditores.
- RF_30: Buscar lista de auditores.
- RF_31: Asignar roles a un auditor.
- RF_32: Editar roles de un auditor.
- RF_33: Eliminar roles de un auditor.
- RF_34: Mostar roles de un auditor.
- RF_35: Bloquear documentos durante el período de auditoría.
- RF_36: Añadir dictamen técnico a partir de una plantilla.
- RF_37: Imprimir dictamen técnico.

2.3.3 Requisitos no Funcionales

Los requisitos no funcionales definen cuáles son las propiedades con las que el sistema debe contar, así como sus restricciones fundamentales. Este tipo de requisitos garantizan que el producto sea fácil de usar, seguro y atractivo. Hacen referencia a las propiedades emergentes de un sistema como la fiabilidad, el tiempo de respuesta, rendimiento y la seguridad [38].

- **Usabilidad**

RNF_1: Utilizar el idioma español para los mensajes y textos de la interfaz.

RNF_2: El módulo puede ser utilizado por personas con conocimientos básicos en el manejo de computadoras. Se utilizarán diversas palabras que permitan al usuario saber dónde debe buscar la información deseada.

- **Fiabilidad**

RNF_3: La precisión y exactitud de las respuestas del módulo se corresponden con la calidad y exactitud de la información insertadas por los usuarios del sistema y la información contenida en el Gestor de Documentos Administrativos Xabal eXcriba 4.0.

- **Portabilidad**

RNF_4: Se puede utilizar el módulo en sistemas operativos Windows y GNU/Linux. Se recomienda utilizar los de GNU/Linux.

- **Legales**

RNF_5: Las herramientas seleccionadas para el desarrollo del módulo están respaldadas por licencias libres, bajo las condiciones de software libre.

2.4 Definición de los casos de uso del sistema

Un caso de uso especifica una secuencia de acciones, incluyendo variantes, que el sistema puede llevar a cabo, y que producen un resultado observable de valor para un actor concreto.

A continuación, se declaran los casos de uso para el módulo.

CU_1: Gestionar auditoría.
CU_2: Gestionar tipo de auditoría.
CU_3: Gestionar lista de auditores.
CU_4: Gestionar roles de un auditor.
CU_5: Gestionar documentos asociados.
CU_6: Asignar lista de auditores a una auditoría.
CU_7: Bloquear documentos durante el período de auditoría.
CU_8: Añadir dictamen técnico.
CU_9: Imprimir dictamen técnico.
CU_10: Filtrar elementos.

2.4.1 Definición de los actores del sistema.

Un actor es un conjunto coherente de roles que los usuarios de casos de uso desempeñan cuando interaccionan con estos casos de uso. Los actores definidos en la investigación se pueden encontrar en la Tabla 2.

Asesor de Planificación	Es el usuario del sistema que ocupa la responsabilidad de gestionar los tipos de auditorías.
Auditor	Es el usuario perteneciente al área de Dirección de Calidad de la UCI que posee los permisos para gestionar las auditorías.

Tabla 2. Definición de los actores.

2.4.2 Diagrama de casos de uso del sistema.

Un diagrama de casos de uso del sistema representa gráficamente a los procesos y su interacción con los actores. En la figura#4 se aprecia el diagrama de caso de uso del sistema propuesto:

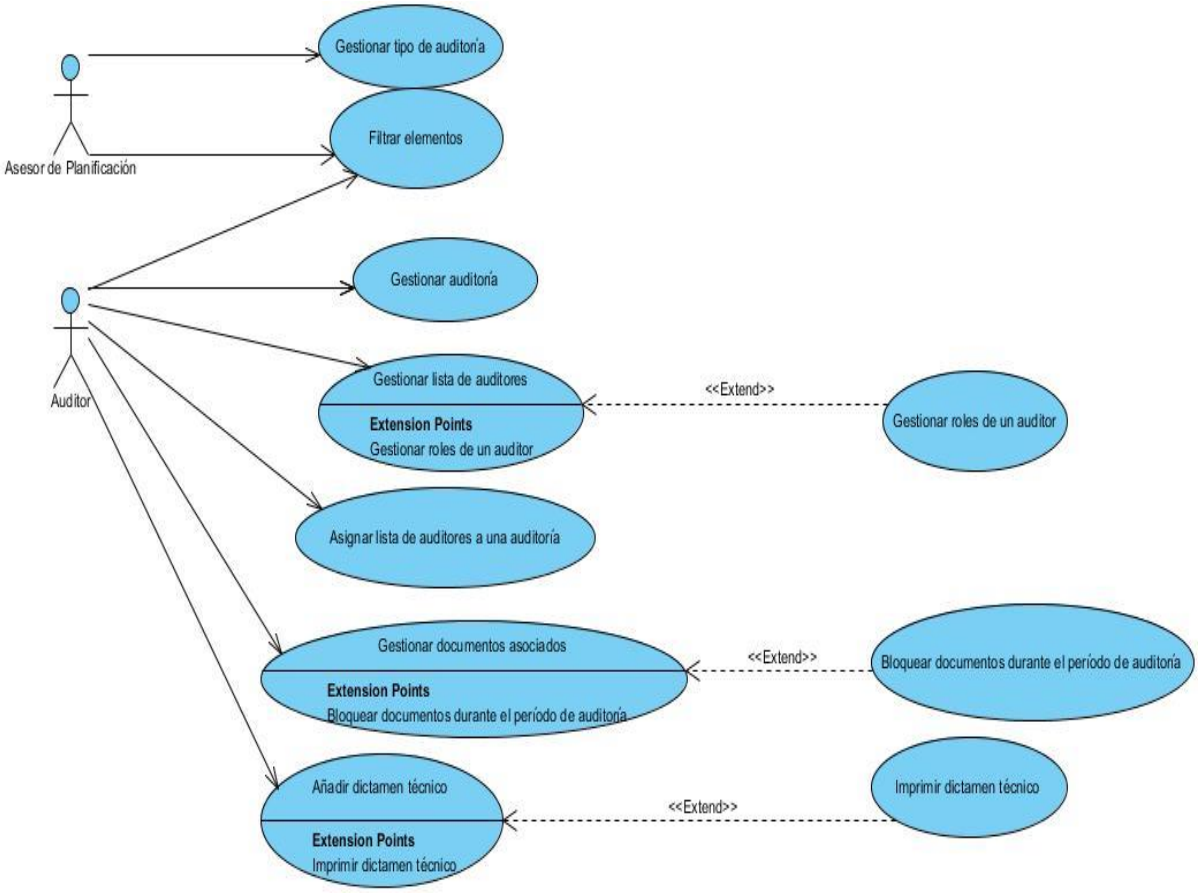


Figura #4 Diagrama de casos de uso.

2.4.3 Descripción de casos de uso del sistema.

A continuación, se expone la descripción del caso de uso Gestionar tipo de auditoría identificado como uno de los casos de uso arquitectónicamente significativos en el sistema. El resto de las descripciones por su extensión se encuentran en el Anexo 2 del presente documento.

Caso de Uso	Gestionar tipo de auditoría
Objetivo	Crear, modificar, listar o mostrar el tipo de auditoría.
Actores	Asesor de Planificación: (Inicia) Crea, modifica o elimina los elementos del tipo de auditoría.

Resumen	El caso de uso inicia en cuanto el actor seleccione crear, modificar o eliminar un elemento de los tipos de auditorías y el sistema permite que se efectúe la acción seleccionada.
Complejidad	Media
Prioridad	Alta
Precondiciones	1. El actor consta de los permisos necesarios y esta autenticado en el sistema. 2. El actor se encuentra en el entorno Lista de Tipos de auditorías. 3. El actor selecciona el elemento que desea actualizar.
Post condiciones	1. Un tipo de auditoría fue creado. 2. Un tipo de auditoría fue modificado. 3. Un tipo de auditoría fue listado. 4. Un tipo de auditoría fue mostrado.

Flujo de eventos		
Flujo básico: Gestionar tipo de auditoría.		
	Actor	**Sistema**
1	Crea, modifica o elimina un elemento.	
2		El sistema muestra las acciones permitidas: • Si desea adicionar un elemento, ver sección 1:” Adicionar elemento.” • Si desea modificar un elemento, ver sección 2:” Modificar elemento.” • Si desea eliminar un elemento, ver sección 3:” Eliminar elemento.”
		Termina el caso de uso.
Sección 1: “Crear elemento”		
Flujo básico		
	Actor	**Sistema**

1	Selecciona el botón “Crear Tipo de Auditoría”.	
2	Inserta los valores solicitados.	
3	Presiona el botón “Aceptar”.	
4		Valida la información insertada. Si hay datos incompletos, ver Flujo Alterno 2: “Existen campos vacíos”. Si hay datos incorrectos, ver Flujo Alterno 3: “Existen campos incorrectos.”
5		Crea el nuevo tipo de auditoría.
6		Termina el caso de uso.

Flujos alternativos

1. **Cancelar Operación.**

	Actor	**Sistema**
1	Selecciona la opción cancelar.	
2		Elimina todos los datos de entrada y regresa a la vista anterior.
3		El caso de uso termina.

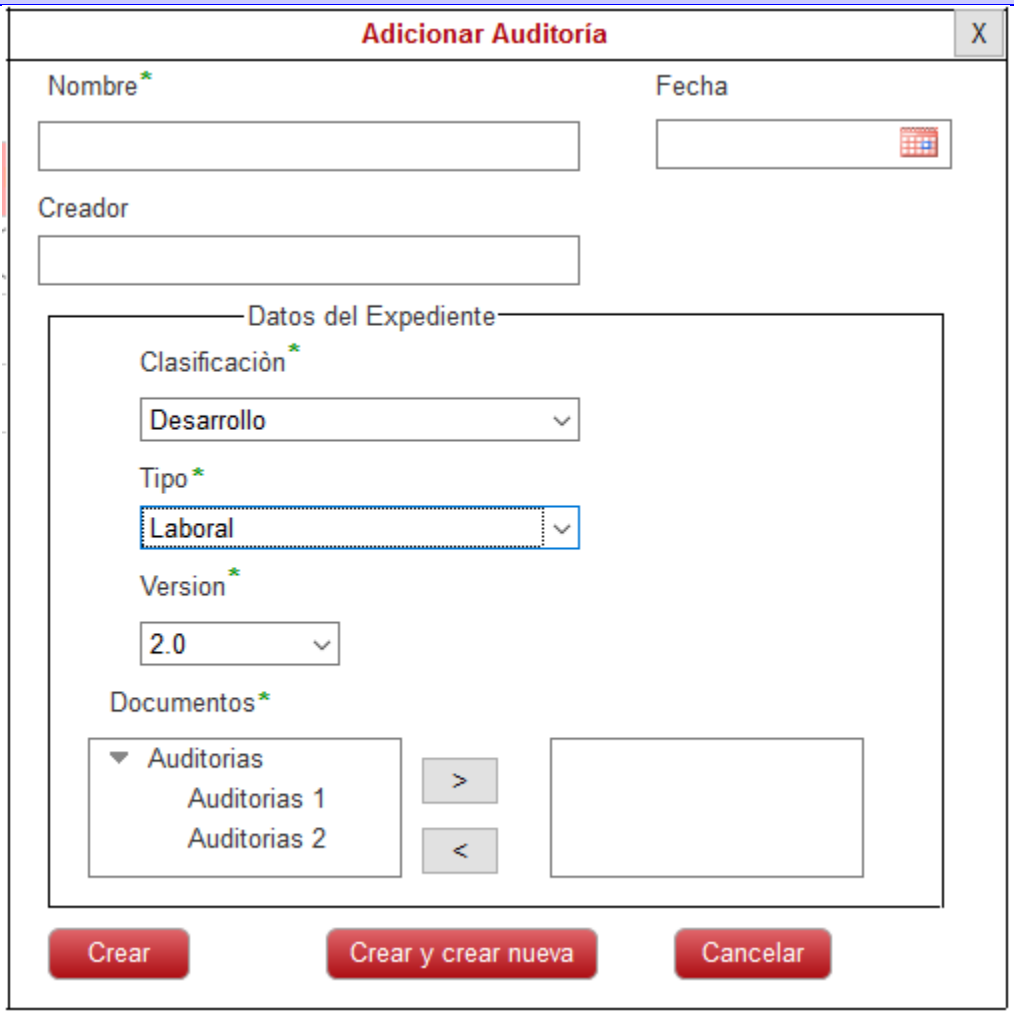
2. **Existen campos vacíos.**

	Actor	**Sistema**
1		Muestra el mensaje de error “Existen campos vacíos.”
2		Muestra un indicador al lado de los campos vacíos.

3. **Existen campos incorrectos.**

	Actor	**Sistema**
1		Muestra el mensaje de error “Existen campos incorrectos.”

2		Muestra un indicador sobre los campos incorrectos.
Relaciones	**CU Incluidos**	No aplica
	CU Extendidos	No aplica



Sección 2:" Modificar elemento"		
Flujo básico		
	Actor	**Sistema**

1	Presionar el ícono de “Editar” del elemento escogido.	
2		Muestra un formulario con los datos del elemento escogido.
3	Inserta los valores que desea actualizar.	
4	Presiona el botón “Aceptar”.	
5		Valida la información insertada. Si hay datos incompletos, ver Flujo Alterno 2: “Existen campos vacíos”. Si hay datos incorrectos, ver Flujo Alterno 3: “Existen campos incorrectos.”
6		Actualiza los elementos del elemento con los datos insertados.
7		Termina el caso de uso.

Flujos alternativos

1. **Cancelar Operación.**

	Actor	**Sistema**
1	Selecciona la opción cancelar.	
2		Elimina todos los datos de entrada y regresa a la vista anterior.
3		El caso de uso termina.

2. **Existen campos vacíos.**

	Actor	**Sistema**
1		Muestra el mensaje de error “Existen campos vacíos.”

2		Muestra un indicador al lado de los campos vacíos.

3. **Existen campos incorrectos.**

	Actor	Sistema
1		Muestra el mensaje de error “Existen campos incorrectos.”
2		Muestra un indicador sobre los campos incorrectos.

Relaciones	**CU Incluidos**	No aplica
	CU Extendidos	No aplica

Sección 3: “Eliminar elemento”		
Flujo básico		
	Actor	**Sistema**
1	Presiona el icono “Eliminar” del elemento escogido.	

2		El sistema muestra una interfaz de confirmación para eliminar.
3	Selecciona el botón “Eliminar”.	
4		El sistema elimina el elemento escogido.
5		Se actualiza la Lista de Tipos de auditorías.
6		Termina el caso de uso.

Flujos alternativos

1. **Cancelar Operación.**

	Actor	Sistema
1	Selecciona la opción cancelar.	
2		Elimina todos los datos de entrada y regresa a la vista anterior.
3		El caso de uso termina.

Relaciones	**CU Incluidos**	No aplica
	CU Extendidos	No aplica

Prototipo elemental de interfaz gráfica de usuario

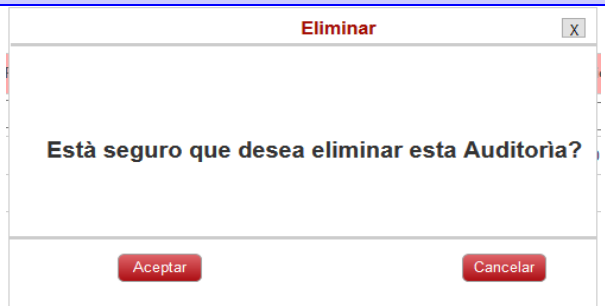


Tabla 3: Descripción del CU: Gestionar tipo de auditoría**.**

2.5 Matriz de Trazabilidad

En este epígrafe se muestra mediante una matriz de trazabilidad cómo son recogidos todos los requisitos funcionales en los casos de uso identificados. La matriz de trazabilidad consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo qué objetivos cubre cada requisito para detectar inconsistencias u objetivos no cubiertos.

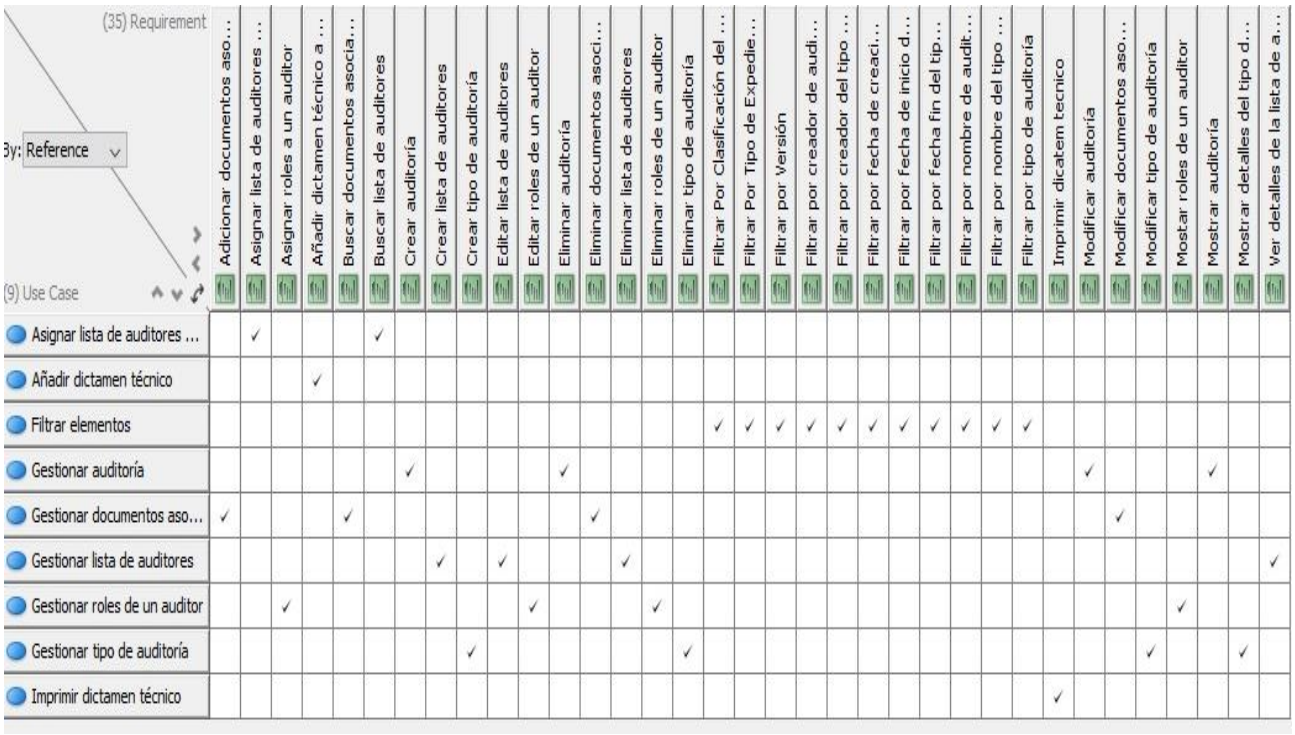


Figura #5 Matriz de trazabilidad de RF-CU.

Como se puede observar en la tabla anterior cada requisito identificado ha sido cubierto por al menos un caso de uso del sistema, lo que significa que la especificación de caso de uso propuesta satisface todas las necesidades del cliente.

2.6 Descripción de la arquitectura

La arquitectura de software de un sistema de programa o computación es la estructura del sistema, la cual comprende los componentes del software, las propiedades de esos componentes visibles externamente, y las relaciones entre ellos[39].

Para el desarrollo del Gestor de Documentos Administrativos Xabal eXcriba 4.0 se utilizó la arquitectura en capas o n-capas, como también es conocida, y por consiguiente el desarrollo del Módulo de gestión de auditorías a expedientes del XABAL eXcriba 4.0 también utilizará la misma.

2.6.1 Arquitectura en Capas

La arquitectura en capas es un estilo de programación, cuyo objetivo primordial es la separación de la capa de presentación, la capa web y la capa de acceso a repositorios, con el propósito de simplificar la comprensión y la organización del desarrollo del sistema. Este patrón reduce las dependencias, ya que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores. La arquitectura propuesta añade una gran flexibilidad al diseño de la aplicación, así como una interoperabilidad en entornos distribuidos con un nivel de abstracción superior[39].

Las tres capas que se definieron para el desarrollo del módulo son: Presentación, Web y Acceso a Repositorio.

Capa de Presentación

En esta capa se encuentra el conjunto de interfaces de usuario, que les hace posible al cliente y la aplicación establecer la comunicación, manipular los datos, así como representar en términos de componentes visuales, toda la información necesaria, consultada y/o generada por el par aplicación usuario. Algunos de los componentes son: ObjectFinder, FormRuntime, FormUI, jQuery y AjaxRequest. El framework que se usa es el YUI. Además, cuenta con una capa de componentes desarrollados por Alfresco Inc que hace función tanto de utilitarios como de integración con algunas restricciones del diseño como la internacionalización y la comunicación con los servicios Rest de la capa web. Durante el desarrollo del módulo esta capa se pone de manifiesto en los formularios, tablas y páginas web[39].

Capa Web

En esta capa se ejecutan todos los procesos de negocio que han sido previamente implementados, se preparan a su vez las transformaciones de datos, sirviendo como un mediador entre las demandas del cliente y las respuestas de los datos. Controla y dirige el flujo de la aplicación en sentido general. Esta capa no maneja una lógica de negocio fuerte, pero sí tiene implementado servicios para la gestión de la autenticación, la configuración de la caché que será enviada a la presentación, también tiene la posibilidad de comunicarse con más de un repositorio, así como con servicios externos, ella se evidencia en los servicios de presentación. Su función principal es la de adecuar los datos transformando la información del repositorio en estructuras web entendibles por un navegador de archivos (html, js, css, imágenes). El framework principal de esta capa es Spring Surf[39].

Capa de Acceso a Repositorio

Esta capa es la más compleja y extensa dentro de la plataforma eXcriba, en ella se implementan un conjunto de servicios que permiten manipular de forma distribuida y a través de la red la información no estructurada y sus metadatos. En el repositorio es posible hacer búsquedas, sobre los contenidos a través del motor Lucene, exporta todas las funcionalidades a través de Rest.Esta capa se pone de manifiesto en los modelos de contenido y servicios de datos[39].

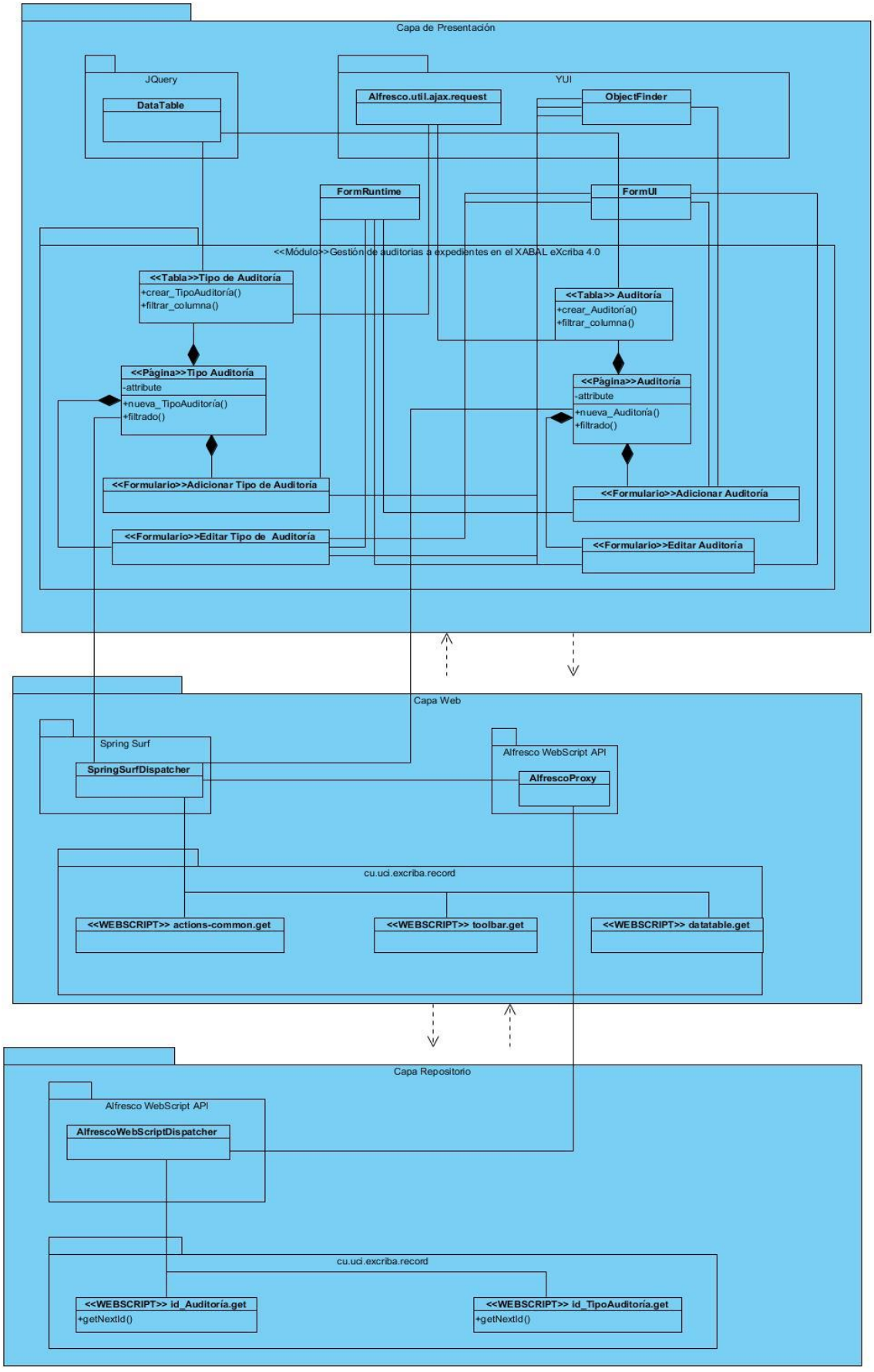


Figura #6 Diagrama de arquitectura.

2.7 Patrones de diseño

Un patrón de diseño es una abstracción de una solución en un nivel alto. Los patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo, desde el análisis hasta el diseño y desde la arquitectura hasta la implementación.

El patrón que se utilizó fue el GRASP (General Responsibility Assignment Software Patterns), los Patrones Generales de Software para Asignar Responsabilidades describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones Dentro de los patrones GRASP utilizados se destacan cuatro, los cuales están relacionados entre sí, pues en la práctica, el nivel de acoplamiento no se puede considerar de manera aislada a otros principios como el Experto o Alta Cohesión[40]:

Experto:

Es el encargado de asignar una responsabilidad al experto en información: la clase que tiene la información necesaria para realizar la responsabilidad. Nos indica, por ejemplo, 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[41].

El patrón experto posibilita que se mantenga el encapsulamiento de la información, pues los objetos utilizan su propia información para llevar a cabo las tareas. Normalmente, esto conlleva un bajo acoplamiento, lo que da lugar a sistemas más robustos y más fáciles de mantener, además se distribuye el comportamiento entre las clases que contienen la información requerida, por tanto, se estimula las definiciones de clases más cohesivas y ligeras que son más fáciles de comprender y manejar. Se soporta normalmente una alta cohesión. Este patrón se pone de manifiesto en la funcionalidad de crear la auditoría. Como se aprecia en figura #7.

```
24          <div class="align-fix">
25              <div class="span9" style="position:relative;z-index: 0;">
26                  <div id="panel1" class="ui-widget ui-panel" style="width:100%;height: 100%; margin-bottom: 18px;">
27                      <div class="ui-widget-header ui-panel-header ui-helper-clearfix">
28                          <span>Adicionar Auditoria</span>
29                      </div>
30
31                      <#--panel-->
32                      <div class="ui-panel-content">
33                      <div class="span6">
34                      <label class="control-label" for="nombre">Nombre <span class="mandatory-indicator" >*</span></label>
35                      <div class="control-group">
36                                      <div class="controls">
37                                          <input type="text" id="nombre" class="input-text-100" placeholder="" name="nombre
38                                      </div>
39                      </div>
40                      </div>
41                        <div class="span6">
42                      <label class="control-label" for="Fecha">Fecha <span class="mandatory-indicator">*</span></label>
43                      <div class="control-group">
44                                      <div class="controls">
```

Figura #7 Funcionalidad de crear auditoría.

Bajo Acoplamiento:

La función de este patrón es asignar una responsabilidad de manera que el acoplamiento permanezca bajo. Un elemento con bajo acoplamiento no depende de demasiados elementos, estos elementos pueden ser clases, subsistemas, sistemas, etcétera.

Este patrón permite que en caso de modificarse algún elemento sus cambios no afecten a otro componente, además son fáciles de entender de manera aislada y conveniente para la reutilización. En la figura se muestra el código utilizado[41].

```
var n=search.findNode(args.n);
model.result =jsonUtils.toJSONString(N.properties);
```

Figura # 8: Ejemplo de patrones GRASP: Bajo acoplamiento.

Alta cohesión:

GDA eXcriba caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen todo un trabajo enorme, de modo que la información que almacenan debe ser coherente y debe estar relacionada con la clase, permitiendo mejorar la claridad con que se entiende el diseño. Grady Booch [42]señala que se da una alta cohesión funcional cuando los elementos de un componente (clase, por ejemplo) "colaboran para producir algún comportamiento bien definido"[41] . En la figura # 9 se muestra el código utilizado:

```
function getInformationFromPMDataList(p, siteNodeRef)
{
  var datalist = assocs["gep:projectMember"][0];
     if(datalist && datalist.length > 0){
      datalist = datalist[0];
    }
    else
    {
      logger.log("No se ha encontrado la lista de datos");
      return;
    }
    var node, result = {}, rol;
    for(var i in datalist.children)
    {
        node = datalist.children[i];
        var roles = node.properties['xfpm-dl:projectRoles'];
        var member = node.assocs['xfpm-dl:member'][0];
        for(var r in roles)
        {
          rol = roles[r];
          if(result[rol])
          {
            result[rol].push(member.nodeRef);
          }
          else
          {
             result[rol] = [member.nodeRef];
          }
        }
    }
    return result;
}
```

Figura # 9: Ejemplo de patrones GRASP: Alta cohesión.

2.8 Conclusiones del capítulo

- Con la culminación del presente capítulo se realizó el levantamiento de requisitos a partir de las técnicas de capturas de requisitos análisis de sistemas existentes, tormenta de ideas, entrevistas. La matriz de trazabilidad permitió comprobar que todos los requisitos se incluyeran en un CUS para dar cumplimiento a las exigencias del cliente.
- La descripción de CUS y los diagramas de clases de diseño a partir del lenguaje de modelado UML orientaron efectivamente la implementación del sistema, su documentación y comportamiento acorde a los requisitos.
- El análisis de la arquitectura y los patrones de diseño permitió conocer cuáles eran los utilizados por el sistema, permitiendo lograr una estandarización en el proceso de integración.

CAPÍTULO III

Capítulo 3: Implementación y validación del módulo de gestión de auditorías a expedientes.

En el presente capítulo se muestra cómo está implementado el sistema y bajo qué condiciones se hizo. Además, se realizó la selección del tipo de prueba y las técnicas que se utilizaron para comprobar la validez del módulo. Posteriormente se describieron los valores empleados para las pruebas y para concluir una evaluación de su ejecución y los resultados obtenidos.

3.1.1Implementación.

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios (24).

3.1.1Estándares de codificación.

Los estándares de código son parte de las llamadas buenas prácticas o mejores prácticas. Estas son un conjunto no formal de reglas que han ido surgiendo en las distintas comunidades de desarrolladores con el paso del tiempo y las cuales, bien aplicadas pueden incrementar la calidad del código. El Módulo de gestión de auditorías a expedientes en el XABAL eXcriba 4.0 se integra a un producto estable existente, por lo tanto, la codificación se realizó según el estándar utilizado en la implementación de XABAL eXcriba 4.0. A continuación, se muestran los estándares de codificación que se utilizaron para la implementación del módulo[43]:

Identación y líneas: usar una Identación sin tabulaciones, con un equivalente a cuatro espacios. El uso de las llaves es seguido del nombre del método. La longitud de las líneas de código es aproximadamente de 75-80 caracteres para mantener la legibilidad del código[43].

Ejemplo:

```
function getInformationFromPMDataList(p, siteNodeRef)
{ \\

{
```

Figura # 10: Estándares de codificación Identación y líneas.

Variables: las variables no deben contener caracteres extraños, rigiéndose por la nomenclatura camelCase, en caso de contenerlos eliminar los acentos de los mismos.

```
function omitirAcentos(text) {
    var acentos = "ÃÀÁÄÂÈÉËÊÌÍÏÎÒÓÖÔÙÚÜÛãàáäâèéëêìíïîòóöôùúüûÑñÇç";
    var original = "AAAAAEEEEIIIIOOOOUUUUaaaaaeeeeiiiioooouuuunncc";
    for (var i=0; i<acentos.length; i++) {
        text = text.replace(acentos.charAt(i), original.charAt(i));
    }
    return text;
}
```

Figura #11: Estándares de codificación Variables.

1.1 Modelo de Despliegue.

El modelo de despliegue consta de uno o más nodos, dispositivos (nodos estereotipados sin capacidad de proceso en el nivel modelado de abstracción) y conectores, entre nodos, y entre nodos y dispositivos. El modelo de despliegue también correlaciona procesos con los elementos de proceso, permitiendo la distribución de comportamiento entre los nodos que se deben representar[44].

Un **modelo de despliegue** puede tener las siguientes **propiedades:**

- Introducción: Una descripción textual que sirve como breve introducción al modelo.
- Nodos: Elementos de proceso en el sistema.
- Dispositivos: Dispositivos físicos, que no tienen ninguna capacidad de proceso (al nivel modelado de abstracción), que dan soporte a los nodos de procesador.
- Conectores: Conexiones entre nodos, y entre nodos y dispositivos. Los conectores pueden tener información asociada en relación con la capacidad o ancho de banda del conector.

Métodos para el despliegue de extensiones de Alfresco.

Existen varias maneras para realizar un despliegue de una extensión para Alfresco, el método concreto depende de la complejidad del desarrollo y del servidor de aplicaciones que utilicemos. Así, por ejemplo, si la personalización consiste solamente en cambios en fichero de configuración o propiedades bastaría con colocarlos en el classpath del servidor de aplicaciones. A continuación, se plantean algunos de los métodos de despliegue de extensiones de Alfresco[44]:

- Extensión de Alfresco usando librerías JAR.
- Extensiones del Cliente Web.
- Extensiones fuera de librerías JAR.
- Extensión con Módulos AMP.

3.2.1 Extensión con Módulos AMP

Siempre es recomendable el uso de módulos de extensión AMP (Alfresco Module Package) para las nuevas funcionalidades. Los módulos de extensión AMP son ficheros que contienen una colección

de código, XML, imágenes, CSS, y otros que extienden la funcionalidad proporcionada por el repositorio de Alfresco. Estos contenidos están distribuidos dentro de una estructura como la siguiente [44]:

Figura #12: Estructura de extensión con módulos AMP

Esta no es una estructura obligatoria, puede faltar cualquier directorio o estar vacío. Cada uno de ellos será mapeado dentro de un Alfresco WAR usando la Herramienta de Manejo de Módulos (MMT):

- /config: El contenido será mapeado dentro del directorio /WEB-INF/classes en el fichero WAR. En este directorio se situarán los ficheros que definirán la configuración Spring y la del Interfaz de Usuario. En este directorio debe estar también el fichero module-contex.xml con la configuración Spring del módulo.
- /lib: El contenido será mapeado en el directorio en /WEB-INF/lib. Deberá contener los ficheros JAR de nuestro modulo.
- /licenses: Si el módulo usa alguna librería que necesite de una licencia, estas deberán estar situadas en este directorio.
- /web/jsp: El contenido será mapeado con el directorio /jsp del fichero WAR. El directorio debe contener todos los JSP nuevos o modificados del módulo.
- /web/css: El contenido será mapeado con el directorio /css del fichero WAR. El directorio debe contener las hojas de estilo de nuestro módulo.

- /web/images: El contenido será mapeado en el directorio /images del WAR. Contendrá las imágenes que use nuestro módulo.
- /web/scripts: Los contenidos serán mapeados en el directorio /scripts del WAR. Los ficheros con código JavaScript que use el módulo deberán ser puestos aquí.

3.3 Diagrama de componentes

El diagrama de despliegue es usado para visualizar los elementos físicos en los que el sistema va a funcionar. Seguidamente se presenta cómo estará desplegado el sistema.

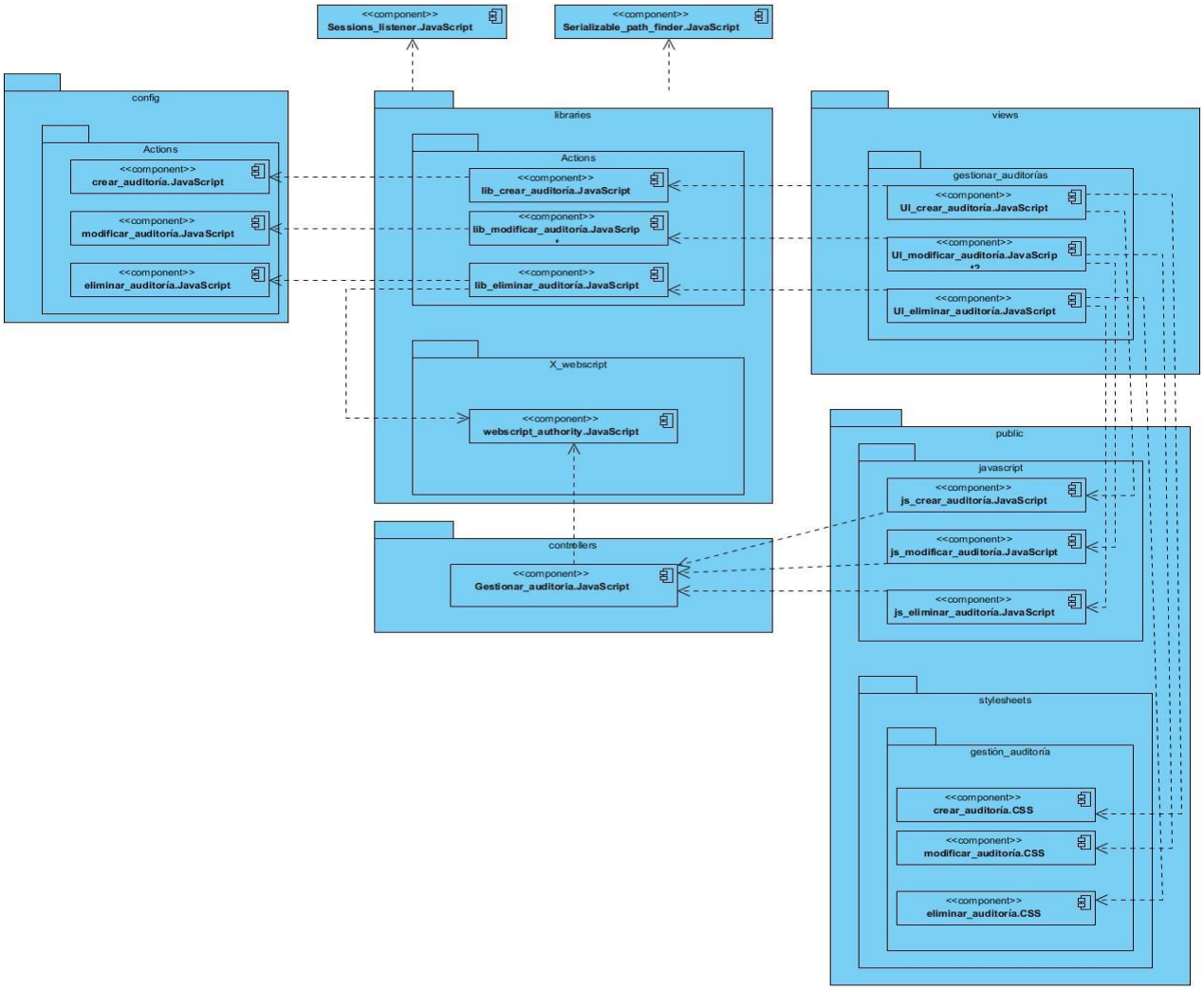


Figura #13 Diagrama de Componentes.

El módulo x-authority-manager contiene todos los componentes agrupados en dos paquetes fundamentales: JavaScript y resource. El paquete JavaScript conteniene todos los componentes que se procesan del lado del servidor mientras que el paquete resource contiene los componentes que se

procesan del lado del cliente. Dentro del paquete JavaScript se encuentra el componente config, donde se definen las configuraciones para ejecutar las acciones o funcionalidades que brinda el módulo, en el componente libraries se encuentran las clases encargadas de construir las vistas asociadas a cada una de las funcionalidades del módulo (paquete actions) y además las clases encargadas de hacer las llamadas a los servicios web (paquete x-webscript).Además, en el paquete JavaScript se encuentran el componente views, que contiene las interfaces visuales que se mostrarán al usuario en el navegador y el componente controllers que contiene la clase GestionarAuditoría la cual coordina la ejecución de las diferentes operaciones que se pueden llevar a cabo en el módulo. En el componente resource se encuentran los paquetes images, javascript y css. En images se encuentran las imágenes utilizadas para decorar las interfaces del módulo, javascript contiene los script que serán ejecutados en el cliente (navegador web) durante la interacción del usuario con las interfaces que brinda el módulo y el componente css contiene las hojas de estilo que se le aplican a las interfaces que serán mostradas al usuario.

3.4 Diagrama de despliegue

El diagrama de despliegue es usado para visualizar los elementos físicos en los que el sistema va a funcionar. Seguidamente se presenta cómo estará desplegado el sistema.

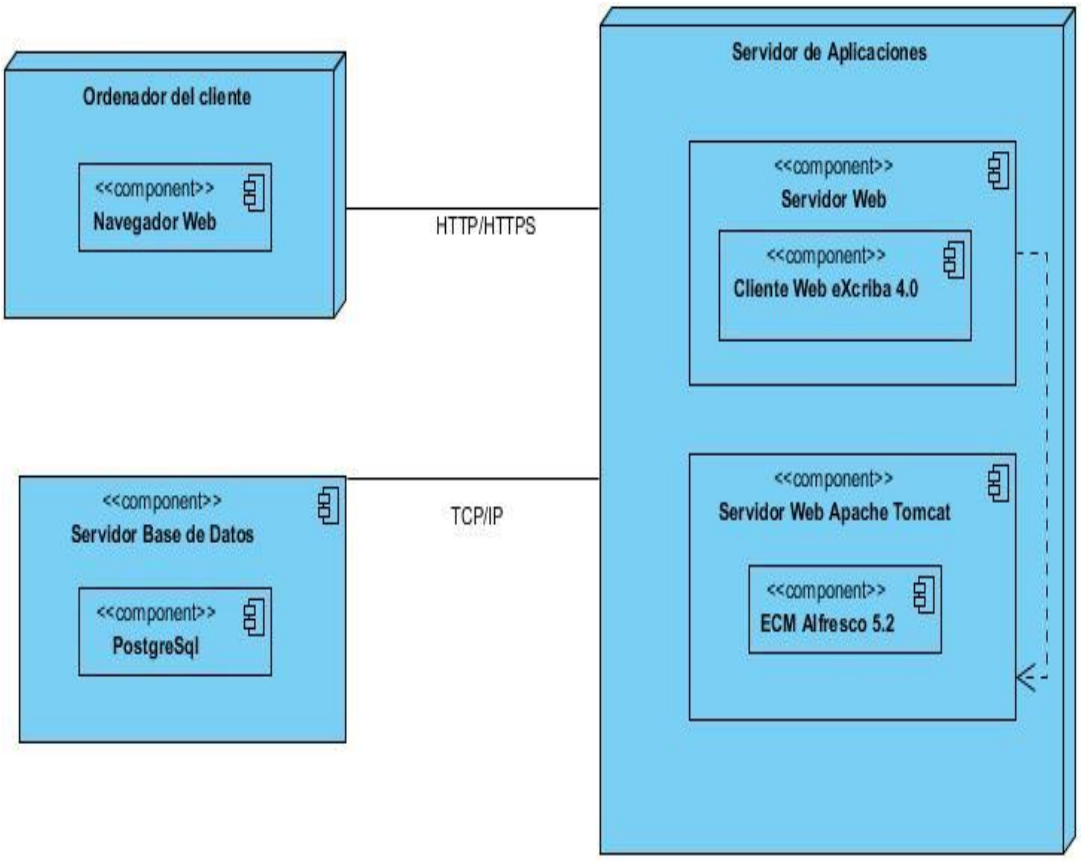


Figura #14 Diagrama de Despliegue.

A continuación, se detalla una breve descripción de cada uno de los nodos presentes en el diagrama de despliegue de la Figura #14.

Ordenador del cliente: en este nodo es donde procesan todas las interfaces de usuario que han sido previamente implementadas. Se encuentra alojado el navegador web mediante el cual el usuario accederá a las interfaces del módulo utilizando el Protocolo de transferencia de hipertexto, por sus siglas en inglés (HTTP) o Protocolo seguro de transferencia de hipertexto, por sus siglas en inglés (HTTPS).

Servidor de aplicaciones: En este nodo estará alojado el GDA eXcriba, en el que se integra la propuesta de solución, y su núcleo el ECM Alfresco, además de la capa de servicios que estará incluida en el propio software.

Servidor de base datos: es donde se encuentran los componentes asociados a la base de datos del sistema alojado en el PostgreSQL 9.4.

3.5 Validación del diseño

Una métrica es un instrumento que permite evaluar el software al inicio del proceso, que cuantifica además un criterio y persigue comprender mejor la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo realizado al nivel de proyecto. La aplicación de métricas al diseño de un producto de software constituye un elemento fundamental a la hora de evaluar la calidad del mismo. Con el fin de desarrollar un diseño robusto y sencillo se realizó la validación del mismo utilizando las métricas Tamaño Operacional de Clase (TOC) y Relaciones entre clases (RC) [44].

Clase	Cantidad de relaciones de uso
Administrador	3
Administrador del Área	2
Tipo de Auditoría	2
Auditor	4
Auditoría	6
Dictamen Técnico	1
Rol	2
Lista de auditores	2
Expediente	2
Documento	2
Permiso	2

Tabla 5: Clases y Relaciones de uso (RC).

Clase	Cantidad de Procedimientos
Administrador	12
Administrador del Área	14
Tipo de Auditoría	17
Auditor	15
Auditoría	11
Dictamen Técnico	11
Rol	13
Lista de auditores	18
Expediente	13
Documento	15
Permiso	19

Tabla 6: Clases y cantidad de procedimientos (TOC).

Atributos de calidad	Descripción
Acoplamiento	Dependencia o interconexión de una clase o estructura de clase respecto a otra.
Complejidad Mantenimiento	Nivel de esfuerzo necesario para sustentar, mejorar o corregir el diseño de software propuesto. Puede influir significativamente en los costes y la planificación del proyecto.
Reutilización	Significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de software.
Cantidad de Pruebas	Número o grado de esfuerzo necesario para realizar las pruebas de calidad al producto diseñado.
Responsabilidad	Responsabilidad que posee una clase en un marco conceptual correspondiente al modelado de la solución propuesta.
Complejidad de implementación	Grado de dificultad que tiene implementar un diseño de clases determinado

Tabla 7. Atributos que posibilita medir las métricas TOC y RC.

	Categoría	Criterio
Responsabilidad	Alta	>2*prom
	Media	Entre prom y 2*prom
	Baja	<=prom

	Categoría	Criterio
Complejidad de implementación	Alta	>2*prom
	Media	Entre prom y 2*prom
	Baja	<=prom

	Categoría	Criterio
Reutilización	Alta	<=prom
	Media	Entre prom y 2*prom
	Baja	>2*prom

Figura #15. Atributos de calidad para las pruebas.

3.5.1 Tamaño Operacional de las Clases (TOC)

Para determinar el valor de los atributos de calidad, se debe determinar la cantidad de procedimientos que posee cada una de las clases a medir. Una vez determinado la CP se procede a calcular el promedio del mismo y, según los criterios expuestos, se determina la incidencia de los atributos de calidad en cada una de las operaciones de las clases[32]. La aplicación del instrumento de evaluación de la métrica TOC en el diseño del sistema propuesto, arrojó los resultados expuestos en el gráfico de la figura #16.

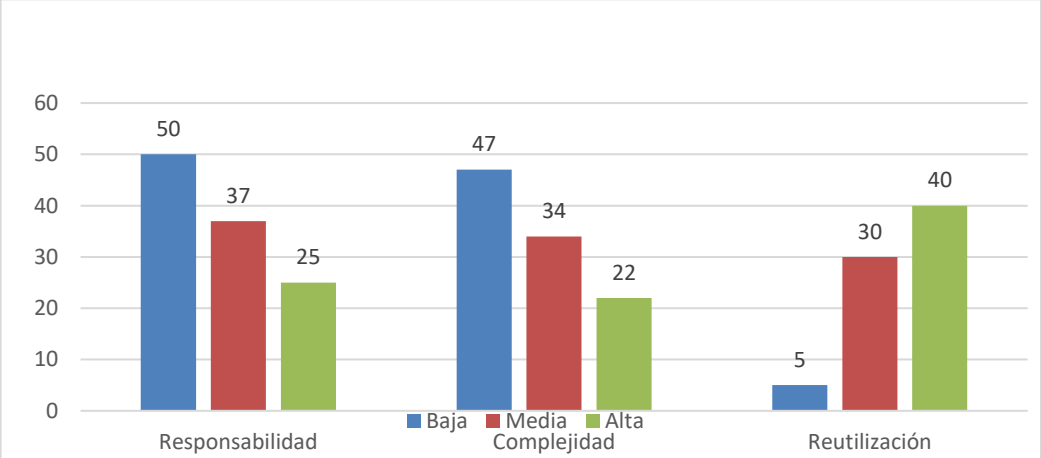


Figura #16. Resultado de la métrica TOC (Fuente: elaboración propia).

- **Responsabilidad:** luego de aplicar la métrica se obtuvieron resultados satisfactorios que reflejan una responsabilidad bajo con un valor del 50%.

- **Complejidad de Implementación**: después de haberse realizado la medición de la métrica, arrojó también resultados positivos ya que la complejidad de las clases es baja en un 47 %.

- **Reutilización:** se obtuvieron valores que según muestra la gráfica de la figura anterior se comporta en un nivel alto con un 40%.

3.5.2 Relaciones entre clase (RC)

Atributo de calidad	Descripción
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase

Tabla 8. Modo en que se afectan los atributos de calidad.

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

	Categoría	Criterio
	ninguno	0
Acoplamiento	Baja	1
	Media	2
	Alta	>2

	Categoría	Criterio
Complejidad de mantenimiento	Alta	>2*prom
	Media	Entre prom y 2*prom
	Baja	<=prom

	Categoría	Criterio
Reutilización	Alta	<=prom
	Media	Entre prom y 2*prom
	Baja	>2*prom

	Categoría	Criterio
Cantidad de pruebas	Alta	>2*prom
	Media	Entre prom y 2*prom
	Baja	<=prom

Figura #17. Umbrales utilizados.

La métrica RC está dada por el número de relaciones de uso de una clase con otra. Permite evaluar el acoplamiento, la complejidad de mantenimiento, la reutilización y la cantidad de pruebas de unidad necesarias para probar una clase, teniendo en cuenta las relaciones existentes entre ellas. Para determinar el grado de afectación de los atributos de calidad que mide la métrica RC es necesario

determinar la Cantidad de Relaciones de Uso (CRU) que posee cada una de las clases a medir. Una vez determinada la CRU, se procede a calcular el promedio de las mismas y teniendo ambos valores según los criterios expuestos, se determina la incidencia de los atributos de calidad en cada una de las clases*[32]*. La aplicación del instrumento de evaluación de la métrica RC en el diseño del sistema propuesto, arrojó los resultados expuestos en el gráfico de la figura 18.

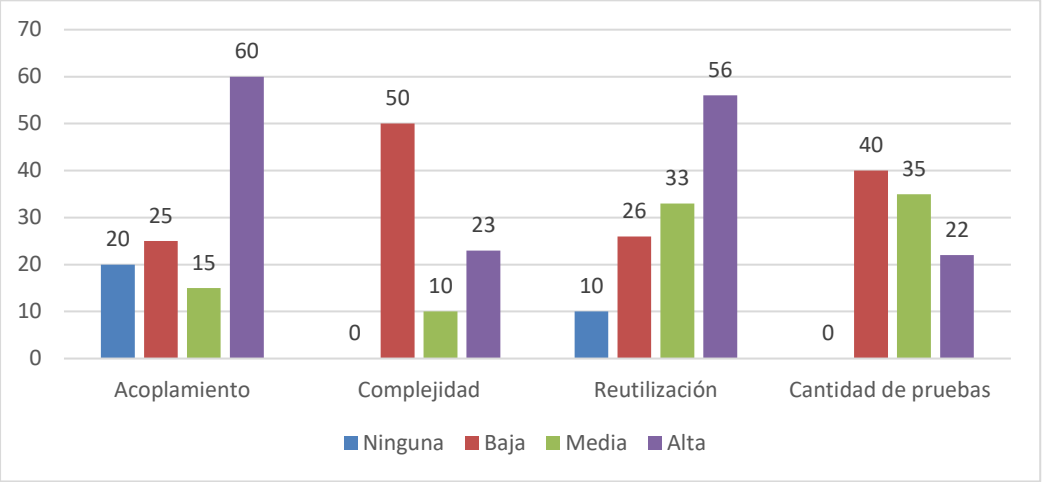


Figura #18. Resultados de las métricas RC (Fuente: elaboración propia).

- **Acoplamiento:** según los resultados que se muestran, el 60% de las clases posee alto acoplamiento.
- **Complejidad de mantenimiento:** según los resultados que se muestran en la figura, el 50% de las clases se comportan satisfactorias.
- **Reutilización:** según los resultados que se mostraron en la figura, el 56% de las clases tiene un alto grado de reutilización.
- **Cantidad de pruebas:** luego de aplicar la métrica se obtuvo que el 40% de las clases poseen un bajo grado de esfuerzo a la hora de realizar cambios, rectificaciones y pruebas de software.

Según lo analizado anteriormente, los valores de RC se comportan de forma satisfactoria, en cada una de los resultados de interés para el diseño propuesto. Estos expresan que las clases del diseño presentan alto acoplamiento, parámetro que no afecta el diseño, teniendo en cuenta las características del negocio. Por otra parte, se muestra que la complejidad de mantenimiento y la cantidad de pruebas son bajas, en consecuencia, el grado de reutilización es alto.

3.6 Prueba de software

Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación.Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un sistema. Básicamente es una fase en el desarrollo de software que consiste en probar la aplicación construida. Se integran dentro de las

diferentes fases del ciclo de la Ingeniería de Software ejecutando un programa que mediante técnicas experimentales se trata de descubrir los errores que tiene el sistema. Para determinar el nivel de calidad se deben efectuar las medidas o pruebas que permiten comprobar el grado de cumplimiento respecto a las especificaciones iniciales del producto, siendo el resultado observado y registrado[45].

3.6.1. Tipos de pruebas

Existen en la actualidad diferentes tipos de pruebas que se le aplican a los softwares para comprobar su calidad, entre estos tipos se encuentran: las pruebas de caja blanca y las pruebas de caja negra.

3.7 Pruebas de caja negra

Las pruebas de caja negra también denominadas pruebas de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Esta prueba intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructura de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y terminación[45].

Técnica de partición de equivalencia

La técnica de partición de equivalencia es considerada una de las técnicas más efectivas, pues permite examinar los valores válidos e inválidos de las entradas existentes en el software. Esta técnica se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar[45].

Una clase de equivalencia representa un conjunto de estados válidos o no válidos 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 lógica. Por tal razón se utilizó la técnica de partición de equivalencia para la confección de los casos de pruebas, ya que permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico[43].

A continuación, se muestra la descripción de las variables para el CU "Gestionar Auditoría", el resto de las descripciones se pueden observar en el Anexo2 de la presente investigación.

Luego de definir el tipo de variables y las descripciones se procede a realizar el diseño de caso de pruebas para estos casos de uso, en las tablas se muestra el Diseño de caso de pruebas para cada caso de uso respectivamente.

Nombre de la variable	Clasificación	Valor Nulo	Descripción
Nombre	Campo de texto	No	En este campo solo pueden introducirse letras

Fecha	Campo de fecha	Si	En este campo se escoge la fecha en el calendario.
Creador	Campo de texto	No	En este campo solo pueden introducirse letras
Clasificación Expediente	Lista de selección única	No	En este campo solo puede escogerse entre las opciones disponibles.
Tipo Expediente	Lista de selección única	No	En este campo solo puede escogerse entre las opciones disponibles.
Versión Expediente	Lista de selección única	No	En este campo solo puede escogerse entre las opciones disponibles.
Adjunto	Lista de selección única	No	En este campo solo puede escogerse entre las opciones disponibles.

Tabla 9: Descripción de las variables

Nombre del Nombre del Caso de Uso: Gestionar Auditoría.										
Esc	Descrip	Nombre	Fecha	Creador	Clasf Exp	Tipo Expe	Ver Expe	Adj	Resp del sistema	Flujo
EC1. Adicionar Auditoría con datos correctos.	Este escenario de prueba permite al usuario adicionar una nueva auditoría.	V Audit1	V 31/06/ 2020	V Julio	V Desarr ollo	V Laboral	V 2.0	V Doc1	Adiciona el elemento correcta mente.	1. Seleccion a la opción crear Auditoría. 2. Seleccion a los datos. 3. Seleccion a la opción aceptar.
EC 2. Adicionar elemento con campos vacíos.	Este escenario de prueba permite al usuario adicionar un nuevo elemento y dejar campos obligatori os vacíos.	I 	V 23/03/ 2020	V Elena	I 	I 	V 3.3	V Doc2	Muestra un mensaje indicando que existen campos vacíos.	1. Seleccion a la opción nuevo elemento. 2. Seleccion a los datos. 3. Seleccion a la opción aceptar.

Tabla 10: Caso de prueba: Gestionar Auditoría-Escenario: Adicionar Auditoría.

Nombre del Nombre del Caso de Uso: Gestionar Auditoría.										
Esc	Descrip	Nom	Fecha	Creador	Clasif Expp	Tipo Expe	Ver Expe	Adj	Resp sist	Flujo central

EC1. Modificar Auditoría.	Este escenario de prueba permite al usuario modificar una auditoría.	V	V	V	V	V	V	V	Modifica el elemento correctamente.	1.Al acceder al sistema, se selecciona la opción Modificar Auditoría. Aparece una vista para insertar los datos que son modificables en la auditoría. 2. Inserta los datos que se van a modificar en la auditoría. 3. Selecciona el botón Aceptar que aparece en la parte inferior del formulario.
		Audit 6	31/06/ 2020	Pedro	Desarrollo	Jurídico	2.0	Doc1		
EC 2. Modificar elementos con campos vacíos.	Este escenario de prueba permite al usuario modific	I	V	V	I	I	V	V	Muestra un mensaje indicando que existen	1.Al acceder al sistema, se selecciona la opción Modificar Auditoría. Aparece una vista para insertar los datos

	ar un elemento y dejar campos obligatorios vacíos.		23/03/2020	Jorge			3.3	Doc2	campos vacíos.	que son modificables en a auditoría. 2. Inserta los datos que se van a modificar en la auditoría, pero deja campos que son obligatorios vacios. 3. Selecciona el botón Aceptar.

Tabla 11: Caso de prueba: Gestionar Auditoría -Escenario:Modificar Auditoría.

Durante la ejecución de estas pruebas se realizaron cuatro iteraciones, donde los errores detectados fueron solucionados conllevando a que en la 4ta iteración no se detectaron errores. Entre los errores detectados se encuentran: errores ortográficos, de validación, de interfaz. A continuación, se presentan los resultados obtenidos.

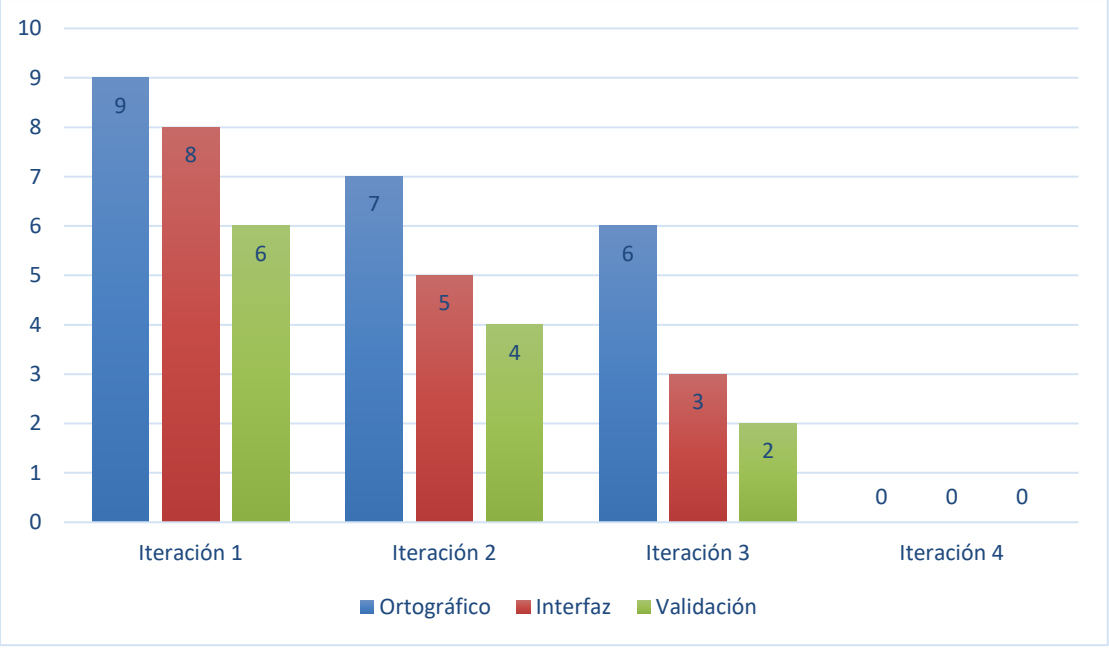


Figura #19: Resultado de las pruebas de caja negra.

3.8 Método de caja blanca

En la prueba de caja blanca se realiza un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del programa, bucles y condiciones; y examinado el estado del

programa en varios puntos. Para realizar estas pruebas el equipo de desarrollo se puede apoyar en varias técnicas para validar la solución[23].

- Prueba de condición: ejercita las condiciones lógicas contenidas en el módulo de un programa.
- Prueba de flujo de datos: se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.
- Prueba de bucles: es una técnica que se centra exclusivamente en la validez de las construcciones de bucles.
- Prueba del camino básico: permite obtener una medida de la complejidad lógica de un diseño, centrándose en encontrar y probar los caminos básicos por los que circula el flujo.

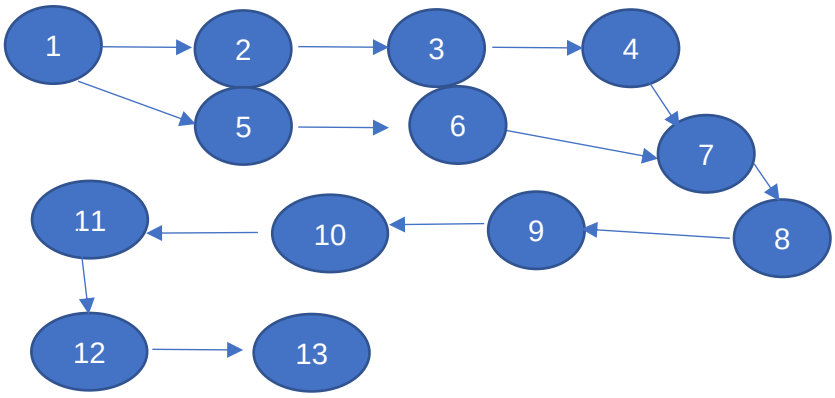
Para la realización de los casos de prueba de caja blanca se decidió realizar las pruebas del camino básico debido a que garantiza que se verifiquen todos los caminos por los que circula el flujo del algoritmo. Los pasos que se siguen para aplicar esta técnica son[46]:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.
5. A continuación, se muestra una figura con las sentencias de código enumeradas del procedimiento realizado sobre el método getNodo(uuid).

```
1  function getNodo(uuid) {
2      var referenceType = "node";
3      var reference = ["workspace", "SpacesStore", uuid];
4      var node = search.findNode(referenceType, reference);
5      return node;
6  }
7
8
9  try {
10     var uuid = json.get('uuid');
11     var nodo= getNodo(uuid);
12     var NombAudit= nodo.properties['audit:NombAudit'];
13     var autorAudit= nodo.properties['audit:autorReporte'];
14     var tablaParaAuditoria= nodo.properties['audit:selectAuditoria']
15     model.NombAudit= NombAudit;
16     model.autorAudit= autorAudit;
17     model.tablaParaAuditoria=tablaParaAuditoria;
18
19 } catch (e) {
20     model.error = e.message;
21 }
22
```

1 2 3 4 5 6 7 8 9 10 11 12 13

Figura #20. Método getNodo(uuid).



Figura# 21. Grafo del camino básico del método agregarUsuario().

La complejidad ciclomática se basa en la teoría gráfica y se calcula de dos maneras distintas, donde, para que el cálculo sea correcto, todas deben arrojar el mismo resultado: El número de regiones corresponde a la complejidad ciclomática “V (G)”. 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 (nodos que tienen más de una arista de salida) contenidos en el gráfico de flujo G.

V (G) = A – N + 2 V (G) = 1 +1

V (G)= 13-13+2 V (G) =1+1

V (G)= 2 V (G) =2

El valor V (G) expresa la cantidad de caminos linealmente independientes de la estructura de control del programa, por lo que se definen los siguientes 3 caminos:

Camino básico #1: 1-2-3-4-7-8-9-10-11-12-13

Camino básico #2: 1-5-6-7-8-9-10-11-12-13

Obtención de casos de prueba: cada camino independiente es un caso de prueba a realizar, de forma que los datos introducidos provoquen que se visiten las sentencias vinculadas a cada nodo del camino. A continuación, en la tabla 11 se muestra el CP diseñado para el camino básico # 2:

Caso de prueba: Gestionar Auditoría	
Entrada	Datos de la Auditoría
Condiciones	Que exista la Auditoría
Resultados esperados	No se obtiene la información de la Auditoría

Tabla 12. Caso de prueba de caja blanca para el camino básico #2

Una vez ejecutados todos los casos de pruebas obtenidos a través de la aplicación de la técnica camino básico para todos los métodos del sistema, se concluye que el código generado está libre de ciclos infinitos y códigos innecesarios.

3.9 Conclusiones del capítulo

- En el capítulo se diseñaron y aplicaron los casos de prueba usados durante el desarrollo de la solución y se describieron las pruebas a las que fue sometida la aplicación, con la finalidad de asegurar que esta cumpla con los requerimientos funcionales.
- En la mayoría de los casos se obtuvieron resultados satisfactorios, lo que permitió asegurar el correcto funcionamiento del módulo de auditoría desarrollado en el eXcriba.
- Con la elaboración del diagrama de despliegue se logró ilustrar la organización física del módulo.

CONCLUSIONES GENERALES

Conclusiones generales

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

- El estudio de los mecanismos existentes para la gestión de auditorías permitió conocer las principales características de estos, identificando así los requisitos funcionales y no funcionales propuestos en la presente investigación para el desarrollo del módulo.
- La selección de la metodología, herramienta y lenguaje de modelado permitió el diseño de un módulo capaz de satisfacer las necesidades planteadas en el problema.
- La implementación de cada una de las funcionalidades definidas para la solución utilizando las herramientas y lenguajes de programación permitieron obtener un producto a la altura de las necesidades del proyecto eXcriba.
- La aplicación de las pruebas permitió corregir las no conformidades detectadas en el módulo de manera satisfactoria, lo que contribuyó a la obtención de un producto final listo para ser desplegado en la versión 4.0 del GDA Xabal eXcriba.
- Los resultados de la validación del diseño demostró alto acoplamiento entre las clases y el grado de reutilización de estas es alto.

RECOMENDACIONES

Recomendaciones

El objetivo de este trabajo ha sido logrado, no obstante, se brindan las siguientes recomendaciones

- Realizar pruebas de rendimientos al módulo con un gran volumen de información en un entorno real.
- Adicionar la funcionalidad que permita registrar las auditorías sobre la modificación de permisos que se han realizado sobre un documento en específico.
- Valorar la inclusión de la solución en la evolución al eXcriba 4.2.
- Utilizar el trabajo como material de aprendizaje para aquellas personas que deseen realizar una aplicación con funcionalidades similares.

ANEXOS

Anexos

Anexo 1: Matriz de trazabilidad de RF-RF.

ANEXOS

Caso de Uso	Gestionar roles de un auditor.
Objetivo	Crea,modifica o elimina los roles de un auditor
Actores	Auditor: (Inicia) Crea, modifica o elimina un rol.
Resumen	El caso de uso inicia en cuanto el actor seleccione crear, modificar o eliminar un rol y el sistema permite que se efectúe la acción seleccionada.
Complejidad	Media
Prioridad	Alta
Precondiciones	4. El actor consta de los permisos necesarios y esta autenticado en el sistema. 5. El actor se encuentra en el entorno Roles. 6. El actor selecciona el rol que desea actualizar.
Post condiciones	1. Un rol fue creado. 2. Un rol fue modificado. 3. Un rol fue listado. 4. Una auditoría fue mostrada.

Flujo de eventos

Flujo básico: Gestionar roles de un auditor.

	Actor	**Sistema**
1	Crea, modifica o elimina un elemento.	
2		El sistema muestra las acciones permitidas: • Si desea adicionar un elemento, ver sección 1:" Adicionar elemento." • Si desea modificar un elemento, ver sección 2:" Modificar elemento." • Si desea eliminar un elemento, ver sección 3:" Eliminar elemento."
		Termina el caso de uso.

Sección 1: "Adicionar elemento"

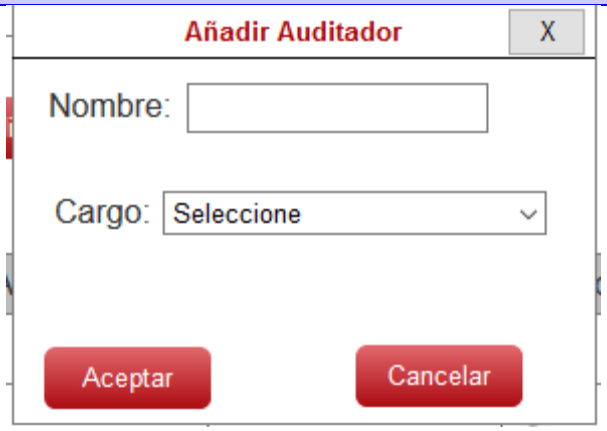
Flujo básico		
	Actor	**Sistema**
1	Selecciona el botón “Nuevo elemento”.	
2		Muestra un formulario solicitando los siguientes datos: -Miembro -Roles
3	Inserta los valores solicitados.	
4	Presiona el botón “Aceptar”.	
5		Valida la información insertada. Si hay datos incompletos, ver Flujo Alterno 2: “Existen campos vacíos”. Si hay datos incorrectos, ver Flujo Alterno 3: “Existen campos incorrectos.”
6		Crea el nuevo usuario con los datos insertados.
7		Actualiza los permisos en los expedientes de proyecto vinculados a la Lista de auditores.
8		Termina el caso de uso.
Flujos alternativos		
Cancelar Operación.		
	Actor	**Sistema**
1	Selecciona la opción cancelar.	
2		Elimina todos los datos de entrada y regresa a la vista anterior.
3		El caso de uso termina.
Existen campos vacíos.		
	Actor	**Sistema**

1		Muestra el mensaje de error “Existen campos vacíos.”
2		Muestra un indicador al lado de los campos vacíos.

Existen campos incorrectos.

	Actor	Sistema
1		Muestra el mensaje de error “Existen campos incorrectos.”
2		Muestra un indicador sobre los campos incorrectos.

Relaciones	**CU Incluidos**	No aplica
	CU Extendidos	Gestionar lista de auditores.



Sección 2:” Modificar elemento”		
Flujo básico		
	Actor	**Sistema**
1	Presionar el icono de “Editar”.	
2		Muestra un formulario con los datos del elemento escogido.

3	Inserta los valores que desea actualizar.	
4	Presiona el botón “Aceptar”.	
5		Valida la información insertada. Si hay datos incompletos, ver Flujo Alterno 2: “Existen campos vacíos”. Si hay datos incorrectos, ver Flujo Alterno 3: “Existen campos incorrectos.”
6		Actualiza los elementos de la Lista de Datos de auditores.
7		Actualiza los permisos en los expedientes de proyecto vinculados a la Lista de auditores.
8		Termina el caso de uso.

Flujos alternativos

1. **Cancelar Operación.**

	Actor	**Sistema**
1	Selecciona la opción cancelar.	
2		Elimina todos los datos de entrada y regresa a la vista anterior.
3		El caso de uso termina.

2. **Existen campos vacíos.**

	Actor	**Sistema**
1		Muestra el mensaje de error “Existen campos vacíos.”
2		Muestra un indicador al lado de los campos vacíos.

3. **Existen campos incorrectos.**

	Actor	**Sistema**

1		Muestra el mensaje de error “Existen campos incorrectos.”
2		Muestra un indicador sobre los campos incorrectos.
Relaciones	**CU Incluidos**	No aplica
	CU Extendidos	Gestionar lista de auditores.

Descripción Sección 3: “Eliminar elemento”		
Flujo básico		
	Actor	**Sistema**
1	Presiona el icono “Eliminar” del elemento seleccionado.	
2		El sistema muestra una interfaz de confirmación para eliminar.
3	Selecciona el botón “Eliminar”.	
4		El sistema elimina el elemento y se actualiza la lista que contiene la Lista de Artefactos.
		Actualiza los permisos en los expedientes de proyecto vinculados a la Lista de Auditores.

5		Termina el caso de uso.

Flujos alternativos

1. **Cancelar Operación.**

	Actor	Sistema
1	Selecciona la opción cancelar.	
2		Elimina todos los datos de entrada y regresa a la vista anterior.
3		El caso de uso termina.

2. **Existen campos vacíos.**

	Actor	Sistema
1		Muestra el mensaje de error “Existen campos vacíos.”
2		Muestra un indicador al lado de los campos vacíos.

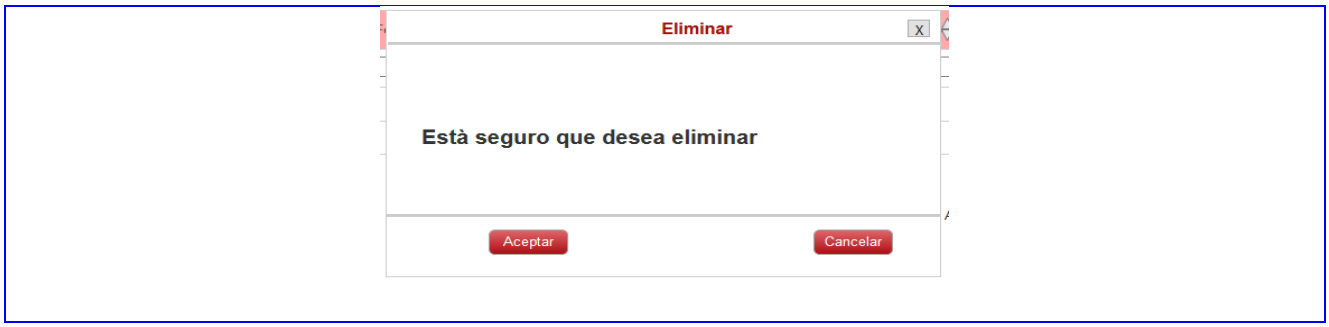
3. **Existen campos incorrectos.**

	Actor	Sistema
1		Muestra el mensaje de error “Existen campos incorrectos.”
2		Muestra un indicador sobre los campos incorrectos.

Relaciones	**CU Incluidos**	No aplica
	CU Extendidos	Gestionar lista de auditores.

Prototipo elemental de interfaz gráfica de usuario

ANEXOS



Anexo 2: Descripción del CUS "Gestionar roles de un audito".

Nombre de la variable	Clasificación	Valor Nulo	Descripción
Nombre	Campo de texto	No	En este campo solo pueden introducirse letras
Tipo	Lista de selección única	No	En este campo solo puede escogerse entre las opciones disponibles.
Expediente	Lista de selección única	No	En este campo solo puede escogerse entre las opciones disponibles.
Fecha Inicio	Campo de fecha	Si	En este campo se escoge la fecha en el calendario.
Fecha Fin	Campo de fecha	Si	En este campo se escoge la fecha en el calendario.
Grupo Revisor	Lista de selección única	No	En este campo solo puede escogerse entre las opciones disponibles.

Anexo 3: Descripción de las variables:CU: Gestionar Tipo de Auditoría.

Nombre del Caso de Uso: Gestionar Tipo de Auditoría.									
Escenario	Descripción	Nombre	Tipo	Expediente	Fecha Inicio	Fecha Fin	Grupo Revisor	Resp del sistema	Flujo central
		V	V	V	V	V	V		

ANEXOS

EC1. Adicionar Tipo de Auditoría con datos correctos	Este escenario de prueba permite al usuario adicionar una nueva auditoría.	Audit1	Interna	Laboral	23/03/2020	31/09/2020	Grupo 1	Adiciona el elemento correctamente.	4. Selecciona la opción crear Tipo de Auditoría. 5. Selecciona los datos. 6. Selecciona la opción aceptar.
EC 2. Adicionar elemento con campos vacíos.	Este escenario de prueba permite al usuario adicionar un nuevo elemento y dejar campos obligatorios vacíos.	V	I	V	I	I	V	Muestra un mensaje indicando que existen campos vacíos.	4. Selecciona la opción nuevo elemento. 5. Selecciona los datos 6. Selecciona la opción aceptar.
		Audit2		Laboral			Grupo 2		

ANEXOS

Nombre del Caso de Uso: Gestionar Tipo de Auditoría.									
Escenario	Descripción	Nombre	Tipo	Expediente	Fecha Inicio	Fecha Fin	Grupo Revisor	Resp sistema	Flujo central
EC1. Modificar un Tipo de Auditoría.	Este escenario de prueba permite al usuario modificar una auditoría.	V Audit1	V Interna	V Laboral	V 23/03/2020	V 31/09/2020	V Grupo1	Modifica el elemento correctamente.	1.Al acceder al sistema, se selecciona la opción Modificar tipo de Auditoría. Aparece una vista para insertar los datos que son modificables en el tipo de auditoría. 2. Inserta los datos que se van a modificar en el tipo de auditoría. 3. Selecciona el botón Aceptar que aparece en la parte inferior del formulario.
EC 2. Modificar elemento con campos vacíos.	Este escenario de prueba permite al usuario adicionar un nuevo elemento y dejar campos	V Audit2	I	V Laboral	I	I	V Grupo2	Muestra un mensaje indicando que existen campos vacíos.	1.Al acceder al sistema, se selecciona la opción Modificar tipo de Auditoría. Aparece una vista para insertar los datos que son modificables en el tipo de auditoría.

	obligatorios vacíos.								2. Inserta los datos que se van a modificar en el tipo de auditoría, pero deja campos que son obligatorios vacios. 3. Selecciona el botón Aceptar que aparece en la parte inferior del formulario.

Anexo 4: Caso de prueba: Gestiona Tipo de auditoría.

BIBLIOGRAFÍA

Bibliografía

- Luis David Fernández Valderrama. GESTIÓN DOCUMENTAL. [Consultado 23 mayo 2020]. Disponible en: http://www.sociedadelainformacion.com/12/Gestion Documental.pdf.
- Fabiola Evelyn Andrade Sánchez, Johanna Agip Valverde. Gestión por procesos (BPM) usando mejora continua y reingeniería de procesos de negocio. Enero 2011. [Consultado 10 marzo 2020]. Disponible en: http://www.cybertesis.edu.pe/sisbib/2007/agip_vj/html/index-frames.html.
- Munwar Shariff. Alfresco Enterprise Content Management Implementation. Packt Publishing, Diciembre 2006.ISBN: 1-904811-11-6.
- Jeff Potts. Alfresco Developer Guide. Packt Publishing, Octubre 2008. ISBN: 1847193110.
- Hammer, M. (1990) – "Re-engineering Work: Don't Automate, Obliterate", Harvard Business Review, pp 104-112, July-August. [Consultado Noviembre 14, 2019]. Disponible en: http://www.kmbook.com/bpr.htm.
- DESARROLLO, E. de, 2017. Manual de Usuario de eXcriba.
- UCI, 2007. Universidad de las Ciencias Informáticas. [en línea], Disponible en: http://www.uci. cu/investigación-y-desarrollo/productos/xabal/excriba-31.
- RODRÍGUEZ, T., 2014. Metodología de desarrollo para la Actividad productiva de la UCI., pp. 1-16.
- ALFRESCO, 2018. Alfresco Content Services [en línea]. 2018. S.l.: s.n. [Consulta: 22 enero 2020]. Disponible en: https://www.alfresco.com/platform/content-services-ecm.
- ATHENTO, [sin fecha]. La gestión documental inteligente ayuda a las entidades públicas con el registro de su correspondencia. [en línea]. España: Disponible en: http://www.athento.com.
- BERGLJUNG, M., 2011. Alfresco 3 Business Solutions. Birmingham: Packt Publishing Ltd. ISBN 184951-335-X.
- BHANDARI, A., CHOUDHARY, V. y MAJMUDAR, P., 2012. Alfresco Share Enterprise Collaboration and Efficient Social Content Management. Birmingham: s.n. ISBN 978-1-84951-710-2.
- BUSTELO RUESTA, C., 2000. Gestión documental en las empresas: una aproximación práctica. Bilbao: s.n.

BIBLIOGRAFÍA

- CAMPILLO TORRES, I., 2010. Sistema de Gestión Integral de Documentos de archivo para empresas de la construcción del territorio de Camagüey. Granada: Editorial de la Universidad de Granada.
- CARUANA, D., NEWTON, J., FARMAN, M., UZQUIANO, M. y ROAST, K., 2010. Professional Alfresco: Practical Solutions for Enterprise Content Management. S.l.: John Wiley and Sons. ISBN 1-118-05717-1.

REFERENCIAS

Referencias Bibliográficas

[1] «ISO 9001 - Implementación y certificación de la norma 9001», *Normas ISO*. [En línea]. Disponible en: https://www.normas-iso.com/iso-9001/. [Accedido: 25-feb-2020].

[2] «En qué consiste una auditoría de la norma de calidad ISO 9001», *Innova Consult*. [En línea]. Disponible en: https://innovaconsult.es/auditoria-iso-9001/. [Accedido: 25-feb-2020].

[3] «requisitos del software iee - Buscar con Google». [En línea]. Disponible en: https://www.google.com/search?sxsrf=ALeKk01bqzUfJTRJglXjwadX8WuviEmfEw%3A1583893385661&ei=iUtoXunxJ66xggfQurWoDA&q=requisitos+del+software+iee&oq=requisitos+del+software+iee&gs_l=psy-ab.3..33i22i29i30l6.65586.68285..69417...0.2..0.141.541.0j4.....0....1..gws-wiz.....0i71j0j0i22i30.kFBkHd41BXQ&ved=0ahUKEwip2vTUrpHoAhWumOAKHVBdDcUQ4dUDCAo&uact=5. [Accedido: 10-mar-2020].

[4] Y. Lazo Alvarado *et al.*, «Proceso de aseguramiento de la calidad para un modelo de la calidad en Cuba», *Rev. Cuba. Cienc. Inform.*, vol. 10, n.º 0, pp. 124-137, may 2016.

[5] «¿Que es la Calidad de Software?», *Calidad De Software*, 07-nov-2012. [En línea]. Disponible en: https://calidadesoftwareutem.wordpress.com/2012/11/07/que-es-la-calidad-de-software/. [Accedido: 25-feb-2020].

[6] «UNE-EN ISO 8402:1995 GESTION DE LA CALIDAD Y ASEGURAMIENTO DE ...» [En línea]. Disponible en: https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0013715. [Accedido: 25-feb-2020].

[7] R. S. Pressman, *INGENIERIA DE SOFTWARE*. McGraw-Hill Interamericana de España S.L., 2010.

[8] «Pero ¿Qué es la Gestión de la Calidad?ISO 9001 calidad. Sistemas de Gestión de Calidad según ISO 9000.» [En línea]. Disponible en: http://iso9001calidad.com/que-es-la-gestion-de-la-calidad-23.html. [Accedido: 25-feb-2020].

[9] «Pero ¿Qué es la Gestión de la Calidad?ISO 9001 calidad. Sistemas de Gestión de Calidad según ISO 9000.» [En línea]. Disponible en: http://iso9001calidad.com/que-es-la-gestion-de-la-calidad-23.html. [Accedido: 25-feb-2020].

[10] «ISO 9000:2015(es), Sistemas de gestión de la calidad — Fundamentos y vocabulario». [En línea]. Disponible en: https://www.iso.org/obp/ui/#iso:std:iso:9000:ed-4:v1:es. [Accedido: 25-feb-2020].

[11] «Proceso de aseguramiento de la calidad para un modelo de la calidad en Cuba». [En línea]. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992016000600010. [Accedido: 25-feb-2020].

[12] A. arevalomaria, «CMMI. Nivel 2 Gestionado: Aseguramiento de la Calidad de Productos y Procesos (PPQA)», *Maria Eugenia Arevalo Lizardo*, 25-jun-2011. [En línea]. Disponible en: https://arevalomaria.wordpress.com/2011/06/25/cmmi-nivel-2-gestionado-aseguramiento-de-la-calidad-de-productos-y-procesos-ppqa/. [Accedido: 25-feb-2020].

[13] MARCO POLO SILVA SEGOVIA, «tecnicas de revisión del software», 08:36:26 UTC.

[14] «Tipos de revisiones Flashcards by Jose Carranza | Brainscape». [En línea]. Disponible en: https://www.brainscape.com/flashcards/tipos-de-revisiones-3620018/packs/5522634. [Accedido: 25-feb-2020].

[15] «Revisiones Técnicas Formales - EcuRed». [En línea]. Disponible en: https://www.ecured.cu/Revisiones_T%C3%A9cnicas_Formales. [Accedido: 24-feb-2020].

[16] «» revisiones de software // Jojooa». [En línea]. Disponible en: http://jojooa.com/analisis-y-diseno-de-sistemas/revisiones-de-software/. [Accedido: 25-feb-2020].

[17] «» revisiones de software // Jojooa». [En línea]. Disponible en: http://jojooa.com/analisis-y-diseno-de-sistemas/revisiones-de-software/. [Accedido: 25-feb-2020].

[18] «¿En qué consiste una auditoría interna de calidad?», *CTMA Consultores*, 12-abr-2017. [En línea]. Disponible en: https://ctmaconsultores.com/auditoria-interna-calidad/. [Accedido: 25-feb-2020].

[19] «Check list / Listas de chequeo: ¿Qué es un checklist y cómo usarlo? : PDCA Home». [En línea]. Disponible en: https://www.pdcahome.com/check-list/. [Accedido: 25-feb-2020].

[20] «MKinsight», *Capterra*. [En línea]. Disponible en: https://www.capterra.es/software/106582/mkinsight. [Accedido: 01-mar-2020].

REFERENCIAS

[21] «MGPI - Maestría en Gestión de Proyectos Informáticos». [En línea]. Disponible en: https://gespro.uci.cu/. [Accedido: 25-feb-2020].

[22] «Software para la Gestión de Servicios de TI - ITSM | SoftExpert». [En línea]. Disponible en: https://www.softexpert.com/es/solucao/gestion-servicios-ti-itsm/. [Accedido: 01-mar-2020].

[23] «¿Qué es un diagrama de clases? - Culturación». [En línea]. Disponible en: https://culturacion.com/que-es-un-diagrama-de-clases/. [Accedido: 10-mar-2020].

[24] «EXCRIBA 3.1 | Universidad de las Ciencias Informáticas». [En línea]. Disponible en: https://www.uci.cu/investigacion-y-desarrollo/productos/xabal/excriba-31. [Accedido: 25-feb-2020].

[25] «Gestor de Documentos Administrativos eXcriba - EcuRed». [En línea]. Disponible en: https://www.ecured.cu/Gestor_de_Documentos_Administrativos_eXcriba. [Accedido: 11-sep-2020].

[26] «Alfresco software y servicios | ECM | BPM». [En línea]. Disponible en: https://www.alfresco.com/es/. [Accedido: 11-sep-2020].

[27] «Alfresco software y servicios | ECM | BPM». [En línea]. Disponible en: https://www.alfresco.com/es/. [Accedido: 27-feb-2020].

[28] «Adoo grady booch». [En línea]. Disponible en: https://es.slideshare.net/camposei1/adoo-grady-booch. [Accedido: 11-sep-2020].

[29] «Activiti BPM: por qué y para qué (caso de implantación)», *AITOR RODRÍGUEZ WEBLOG*, 16-jun-2016. [En línea]. Disponible en: http://aitorrm.github.io/bpm/activiti_bpm_platform_implantacion/. [Accedido: 25-feb-2020].

[30] «JavaScript», *Documentación web de MDN*. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/JavaScript. [Accedido: 27-feb-2020].

[31] «El lenguaje extensible de marcas (XML) 1.0 - Extensible Markup Language (XML) 1.0». [En línea]. Disponible en: http://www.sidar.org/recur/desdi/traduc/es/xml/xml1/index.html. [Accedido: 27-feb-2020].

[32] «AEC - Validacion de diseño». [En línea]. Disponible en: https://www.aec.es/web/guest/centro-conocimiento/validacion-de-diseno. [Accedido: 11-sep-2020].

[33] «Visual Paradigm - EcuRed». [En línea]. Disponible en: https://www.ecured.cu/Visual_Paradigm. [Accedido: 27-feb-2020].

[34] «¿Qué es PostgreSQL? - Para qué sirve, Características e Instalación». [En línea]. Disponible en: https://blog.infranetworking.com/servidor-postgresql/. [Accedido: 27-feb-2020].

[35] «Visual Studio Code - Code Editing. Redefined». [En línea]. Disponible en: https://code.visualstudio.com/. [Accedido: 27-feb-2020].

[36] «Capítulo 3 DEFINICIÓN REQUERIMIENTOS - Metodología Gestión de Requerimientos». [En línea]. Disponible en: https://sites.google.com/site/metodologiareq/capitulo-iii. [Accedido: 11-sep-2020].

[37] «Capítulo 3 DEFINICIÓN REQUERIMIENTOS - Metodología Gestión de Requerimientos». [En línea]. Disponible en: https://sites.google.com/site/metodologiareq/capitulo-iii. [Accedido: 11-sep-2020].

[38] «Ingeniería del software | Proceso de desarrollo de software | Software». [En línea]. Disponible en: https://www.scribd.com/doc/134177165/Ingenieria-del-software. [Accedido: 28-feb-2020].

[39] «Arquitectura basada en capas. – Blog de Juan Peláez en Geeks.ms». [En línea]. Disponible en: https://geeks.ms/jkpelaez/2009/05/30/arquitectura-basada-en-capas/. [Accedido: 11-sep-2020].

[40] «¿Qué son los patrones de diseño de software? – Consultoría y Servicios IT para empresas | Profile Software Services». [En línea]. Disponible en: https://profile.es/blog/patrones-de-diseno-de-software/. [Accedido: 11-sep-2020].

[41] «Patrones de diseño de software - DevExperto, por Antonio Leiva». [En línea]. Disponible en: https://devexperto.com/patrones-de-diseno-software/. [Accedido: 11-sep-2020].

[42] Israel Campos E, «Adoo grady booch», 22:45:28 UTC.

REFERENCIAS

[43] «Estándar de codificación Java (AMAP.CodificacionJava) - XWiki». [En línea]. Disponible en: https://amap.cantabria.es/amap/bin/view/AMAP/CodificacionJava. [Accedido: 11-sep-2020].

[44] «Diagrama de despliegue - EcuRed». [En línea]. Disponible en: https://www.ecured.cu/Diagrama_de_despliegue. [Accedido: 11-sep-2020].

[45] «Tipos de Pruebas de Software | OpenWebinars». [En línea]. Disponible en: https://openwebinars.net/blog/tipos-de-pruebas-de-software/. [Accedido: 11-sep-2020].

[46] «Pruebas de caja blanca - EcuRed». [En línea]. Disponible en: https://www.ecured.cu/Pruebas_de_caja_blanca. [Accedido: 11-sep-2020].