UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 3

Centro de Gobierno Electrónico

Componente para la gestión de la seguridad en el Expediente Judicial Electrónico.

Trabajo final presentado en opción al título de

Ingeniero en Ciencias Informáticas

Autores: - Yaraisy de la Caridad González Vargas

- Orel Pérez Camejo

Tutor(es): - Ing. Yaiset Moreno Aleaga

- Ing. Yosviel Domínguez González

- Ing. José Miguel Fabra Gallo

Ciudad de La Habana, 2019

"Todos en este país deberían aprender a programar una computadora ... porque te enseña a pensar."

- Steve Jobs

Dedicatoria

A mi abuela Adela por ser la luz que me guía en el camino más oscuro.

A mi madrina por enseñarme que en la vida es necesario tener metas y luchar por lograrlas.

Yaraisy

A mis padres por ser mi motor impulsor durante estos cinco años, por todo su apoyo, amor y cariño.

Orel

Agradecimientos

A todos los que han hecho posible mi formación profesional a mis tutores, profesores, a mi jefa de año Dariela gracias por el apoyo brindado.

A mis compañeros y amigos con quienes he compartido todos estos años.

A mi compañero de tesis por convertirse en un amigo y calmarme en los momentos difíciles.

A mis dos amigas Rachel e Iliana, por estar a mi lado y quererme como a una hermana.

Gracias mi familia, tíos y tías por todo el cariño y aliento que me han dado.

A mis padres el mayor agradecimiento que les puedo dar es el haberme dado la vida, gracias por su amor incondicional.

A mis hermanos, espero haberles dado el ejemplo que debía darles como hermana mayor.

A mi abuela Mimi, mi viejita, por escuchar mis quejas y pesares, por su dulzura y cariño.

A mi madrina, quien ha estado a mi lado dándome su amor como una madre.

Gracias a mi abuela Adela, mi DAMA DE HIERRO, mi inspiración y ejemplo a seguir.

Yaraisy

A mis padres por estar siempre a mi lado y haberme dado todo el apoyo y afecto necesario para llegar hasta aquí.

A mi abuela por ser mi segunda madre, por haberme consentido en todo y siempre darme fuerzas cuando más falta me hacía.

A mi hermana, por el amor que compartimos.

A todos mis amigos por apoyarme durante todo este tiempo y convertirme en la persona que soy hoy, Yoel, Alí, Sergio, Jose y entre muchos otros.

A mi compañera de tesis por su apoyo incondicional.

Orel

DECLARACIÓN JURADA DE AUTORÍA

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

Para que así conste firmamos la presente a los __ días del mes de ___del año ___.

Firma del autor
Yaraisy de la Caridad González Vargas

Firma del autor
Orel Pérez Camejo

Firma del tutor
Ing. Yaiset Moreno Aleaga

Firma del tutor
Ing. Yosviel Domínguez González

Firma del tutor

Ing. José Miguel Fabra Gallo

RESUMEN

Actualmente Cuba atraviesa por un proceso de informatización de la sociedad para lo que se han creado una serie de proyecciones estratégicas y de las cuales el sector jurídico no se encuentra ajeno. Como colaboración entre el Tribunal Supremo Popular (TSP) y el Centro de Gobierno Electrónico (CEGEL), adscrito a la Universidad de las Ciencias Informáticas (UCI) surge el actual desarrollo del Expediente Judicial Electrónico (XEJEL), para la gestión de los expedientes. El sistema XEJEL en la actualidad carece de mecanismos de seguridad que permitan solo el acceso de las personas autorizadas y controlen las actividades que estas realizan. Como objetivo del presente trabajo de diploma se dispone a desarrollar un componente de seguridad que contribuya al aseguramiento de la integridad y confidencialidad de la información. Para lograr este objetivo se define como guía, la metodología de desarrollo Proceso Unificado Ágil (AUP) en su variación para la UCI, así como también se seleccionaron las herramientas y tecnologías a utilizar durante el desarrollo de la propuesta que brinda solución a la necesidad existente. Los resultados obtenidos fueron validados a través de métricas y pruebas de software, donde se pudo comprobar que se logró un producto de software libre de incongruencias y errores en su funcionamiento.

Palabras Claves: Expediente, Confidencialidad, componente, integridad, seguridad.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Introducción

En esta era, denominada de la información y el conocimiento, no es común hablar de novedad sin mencionar o relacionar a las tecnologías de la información y las comunicaciones (TIC) y es que se puede asegurar que dichas tecnologías ofrecen una respuesta a la mano de quien la posee. Las TIC han impactado todas las áreas del desarrollo social y el sector jurídico no se encuentra exento a ello, ya que varias pueden ser las necesidades que hoy por hoy impulsan a las instituciones del sistema de justicia a modernizarse en cuanto al uso de las TIC entre las que se encuentran: facilitar y hacer más efectiva la tramitación de causas, mejorar la organización del trabajo y productividad de los tribunales, así como optimizar la calidad de la información que es producida en audiencia, entre otras.

En la actualidad, Cuba apuesta por un proceso de informatización de la sociedad, el cual se refleja explícitamente en las políticas y programas del Estado, entre los que se destaca el "Programa Rector de la Informatización de la Sociedad en Cuba". En el mencionado programa, se plasma la necesidad de impulsar el Gobierno Electrónico en la isla e informatizar sectores estratégicos como el sector jurídico. Como resultado de estas proyecciones estratégicas surge en 2012 un proyecto de colaboración entre el Tribunal Supremo Popular (TSP) y el Centro de Gobierno Electrónico (CEGEL), adscrito a la Universidad de las Ciencias Informáticas (UCI). Entre los resultados más significativos de esta cooperación se encuentra el actual desarrollo del Expediente Judicial Electrónico (XEJEL), para la informatización de los expedientes.

Los expedientes constituyen un instrumento, que resulta de la agregación de las distintas actuaciones, de las partes y del órgano judicial, en forma de legajo[1]. En tal sentido, toda la información y documentos que lo conforman es sensible, confidencial y pública, por lo que la seguridad constituye un elemento de vital importancia.

Los Tribunales Populares Cubanos (TPC) están estructurados en 186 instancias en todo el país, divididas en 169 municipales,16 provinciales y un supremo en las cuales trabajan diferentes personas ocupando diferentes cargos, con los que pueden realizar diferentes acciones sobre la información. De la idea anterior se puede inferir, que cualquier sistema para la gestión de los expedientes judiciales, debe garantizar la seguridad de la información desde dos aristas fundamentales: la integridad y la confidencialidad acorde con los principios fundamentales de la Seguridad Informática. Entiéndase la integridad, como la propiedad que busca mantener los datos libres de modificaciones no autorizadas y la confidencialidad como la garantía de que la información es accesible únicamente por las entidades o personas autorizadas.

[1] Conjunto de informaciones, documentos o papeles recopilados, referentes a una persona o un asunto (Oxford, 2018).

Para garantizar la integridad y la confidencialidad de la información en XEJEL, se necesita la identificación de cada persona que acceda al sistema a través de un usuario otorgándole acceso basado en su cargo teniendo en cuenta las acciones que puede realizar según la instancia en la que trabaja.

Ante la problemática anteriormente planteada, surge el siguiente **problema a resolver:** ¿Cómo gestionar la seguridad en el Expediente Judicial Electrónico de manera que contribuya a la integridad y confidencialidad de la información?

Teniendo como **objeto de estudio:** Expedientes judiciales electrónicos.

Siendo el **objetivo general:** Desarrollar un componente de seguridad para el Expediente Judicial Electrónico que contribuya al aseguramiento de la integridad y confidencialidad de la información.

En un **campo de acción** sobre: Gestión de la seguridad en expedientes judiciales electrónicos.

A raíz del objetivo general surgen los siguientes **objetivos específicos:**

- ✓ Establecer los referentes teóricos en los que se sustenta la propuesta de solución.
- ✓ Realizar la identificación de requisitos, análisis y diseño de la propuesta de solución como aproximación a la implementación.
- ✓ Implementar los componentes para contribuir a la gestión de la seguridad en el Expediente Judicial Electrónico.
- ✓ Valorar la viabilidad de la propuesta de solución.

Métodos de investigación

Métodos teóricos:

Histórico - Lógico: Para realizar el estudio del estado del arte, sustentándose en la necesidad de analizar y estudiar la trayectoria que han tenido los sistemas similares ya existentes, tanto nacional como internacional.

Analítico - Sintético: Para la extracción de las características, rasgos y elementos más importantes del negocio para una mayor comprensión del mismo y lograr obtener de manera sintetizada el contenido necesario y suficiente para elaborar conclusiones que brindan solución a la necesidad de la investigación.

Modelación: Para la realización de diagramas y modelos que permitieron obtener un mejor entendimiento ante las necesidades del proyecto y así lograr un correcto diseño e implementación del mismo.

Métodos Empíricos:

Entrevista: Se realizaron entrevistas con el objetivo de obtener información acerca del negocio y de los requerimientos que debía cumplir el software. Se realizó una entrevista no estructurada la cual es mucho más abierta y flexible, sin descuidar el objetivo establecido inicialmente en la investigación.

Estructura capitular

El presente documento está compuesto por tres capítulos los cuales son:

Capítulo 1. Fundamentación teórica: En este capítulo se abordan los fundamentos teóricos relacionados con expediente judicial y la seguridad informática, de igual manera se describe la metodología, las herramientas y las tecnologías utilizadas durante el desarrollo de la propuesta de solución.

Capítulo 2. Propuesta solución: En este capítulo se presenta la descripción general de la propuesta solución. Para dar cumplimiento a ello y teniendo como guía la metodología seleccionada, se lleva a cabo la disciplina de Requisitos la cual permite que se engloben los requisitos funcionales y no funcionales por los cuales se va a regir la propuesta de solución. Se obtienen los artefactos correspondientes a la disciplina de Análisis y diseño. Finalmente, quedan definidos los estándares de implementación a utilizar como base para la implementación.

Capítulo 3. Validación: En este capítulo se valora la calidad de los resultados obtenidos, la cual se lleva a cabo a raíz de la validación del diseño mediante las métricas: Tamaño Operacional de Clases y Relación entre Clases. En último lugar se realizan las pruebas internas y de liberación como disciplinas que propone la metodología y con las cuales se verifica el buen funcionamiento del producto.

Capítulo 1: Fundamentación teórica

Introducción

En el siguiente capítulo se abordarán los fundamentos teóricos relacionados con expediente judicial y la seguridad informática, así como también quedará plasmada la metodología, las herramientas y las tecnologías a utilizar durante el desarrollo de la solución.

1.1 Expediente Judicial Electrónico

Expediente judicial

Expediente Judicial es la acumulación de documentos, escritos de procedimiento y fallos, relativos a un litigio incoado[2] ante una jurisdicción civil, comercial o social, dentro de un legajo en el cual se mencionan los distintos acontecimientos del proceso. (1)

Se le puede definir como el legajo de actuaciones o piezas escritas que registran los actos procesales realizados en un juicio, ordenadas cronológicamente y foliadas en forma de libro, provistas de una carátula destinada a su individualización, está sujeto a normas para su formación y conservación. (2)

De acuerdo a su definición, el expediente judicial es un instrumento, que resulta de la agregación de las distintas actuaciones, de las partes y del órgano judicial, en forma de legajo. El objetivo del expediente judicial, consiste en representar la historia del proceso, mostrando el trabajo profesional y de la autoridad judicial a lo largo de la contienda. (3)

Luego de analizar las posiciones teóricas anteriores, los autores de la investigación se identifican con la definición que establece el expediente como el instrumento que resulta de la agregación de las distintas actuaciones, de las partes y del órgano judicial, en forma de legajo, ya que es la mayormente conocida y utilizada en el contexto judicial cubano.

Expediente judicial electrónico

Es el conjunto de documentos electrónicos correspondientes a un procedimiento judicial, cualquiera que sea el tipo de información que contenga (4). El expediente deja de ser un conjunto de papeles que se traslada, para pasar a ser un conjunto de información en formato electrónico. (5)

Relación con las TIC

Dejando en claro anteriormente lo referido a expediente judicial electrónico, se hace necesario tocar el tema de su relación con las TIC y es que actualmente, el uso de nuevas tecnologías en el sistema judicial puede ayudar a hacer más eficiente y efectiva la gestión de expedientes judiciales, tanto en lo que dice relación con los costos del proceso, los tiempos de demora como el manejo de causas,

[2] Incoar: hacer la primera gestión o trámite en un proceso, pleito o expediente.

como en otras labores administrativas propias de un tribunal, "…se puede señalar en términos generales que las TIC pueden posibilitar grandes ahorros de costos y de tiempos, mediante la automatización de lo repetitivo, el acceso más rápido y seguro a datos, la comunicación más fluida y segura, entre otros aspectos" (6).

1.2 Seguridad en el expediente judicial

El uso de las TIC en los expedientes judiciales puede traer grandes beneficios, pero estos no los excluye de las consecuencias negativas que pueden ocurrir a su vez como pueden ser: la perdida de información, el acceso a la misma de personal no autorizado, entre muchas más, para evitar dichas consecuencias se hace necesario garantizar ciertos mecanismos de seguridad informática. Para una mejor comprensión del término, a continuación, se analizan los conceptos relacionados con Seguridad y Seguridad Informática.

Seguridad

La Seguridad debe ser interpretada como un estado personal que permite percibir que una persona se mueve en un espacio libre de riesgos reales o potenciales, la ausencia o la falta de esta puede originar diversos problemas y daños. (7)

Sentimiento de protección frente a carencias y peligros externos que afecten negativamente la calidad de vida; en tanto y en cuanto se hace referencia a un sentimiento, los criterios para determinar los grados de seguridad pecarán de tener algún grado de subjetividad. (8)

Según el diccionario de la real academia española la seguridad es la cualidad de seguro, el espacio encargado de la seguridad de una persona, de una empresa, etc. (9)

Por lo tanto, la seguridad no es más que proteger cualquier elemento ante la presencia de algún riesgo o amenaza que provoque daños o pérdida a corto, mediano o largo plazo en cualquier escenario que se desarrolle.

Seguridad Informática

La seguridad informática es una disciplina que se encarga de proteger la integridad y la privacidad de la información almacenada en un sistema informático. (10)

La información es un activo que, como otros importantes activos de negocios, tiene valor para una organización y en consecuencia necesita ser debidamente protegido. La seguridad informática protege la información de un amplio rango de amenazas con el objetivo de asegurar la continuidad de negocios, minimizar el daño comercial y maximizar el reembolso de las inversiones y oportunidades comerciales.

La información puede existir en muchas formas, puede ser impresa o escrita en papel, almacenada electrónicamente, trasmitida por correo o usando medios electrónicos, impreso en películas o

hablado en conversación. No importa la forma que tome, el medio por el que se comparta o en el que se almacene, siempre debe ser correctamente protegida.

La seguridad informática se caracteriza como la protección de:

- ✓ La confidencialidad: asegurar que la información es accesible solo para aquellos autorizados a tener acceso;
- ✓ La integridad: salvaguardar la exactitud y totalidad de la información y los métodos de procesamiento;
- ✓ La disponibilidad: asegurar que los usuarios autorizados tengan acceso a la información y activos asociados cuando se requiera. (11)

En términos generales, la seguridad informática puede entenderse como aquellas reglas técnicas y/o actividades destinadas a prevenir, proteger y resguardar lo que es considerado como susceptible de robo, pérdida o daño, ya sea de manera personal, grupal o empresarial. (12) Se logra mediante la implementación de un apropiado sistema de controles, que pudieran ser políticas, prácticas, procedimientos, estructuras organizacionales y funciones de software. Estos controles necesitan ser establecidos para asegurar que los objetivos específicos de seguridad se cumplan (11). En este sentido, es la información el elemento principal a proteger, resguardar y recuperar dentro de las redes institucionales y personales.

Para el componente de Seguridad en cuestión, se tendrán en cuenta que la información almacenadas en los expedientes cumpla con la confidencialidad y la integridad, dichas propiedades se pueden garantizar desde el desarrollo del software, sin embargo, la disponibilidad adquiere su punto cúspide en el momento del despliegue asegurando que el hardware cumpla con las condiciones necesarias para asegurar el respaldo de la información y que los servicios se encuentren activos en todo momento.

1.3 Sistemas similares existentes

Sistemas informáticos existentes en el mundo

El avance tecnológico se hace cada vez más rápido, su introducción en la sociedad tiene un comportamiento ascendente y el sector jurídico forma parte de este acelerado proceso haciendo que cada vez sean más las instituciones jurídicas que busquen refugiarse en ello dado a los grandes aportes que trae consigo. A continuación, se muestra un análisis de las tecnologías existentes en el mundo que permiten la gestión de la seguridad de la información y se muestra una tabla resumen que permite descartar o no la posibilidad de reutilización.

Sistema de gestión procesal de justicia (AVANTIUS): es un producto español de gestión integral de expedientes judiciales, que permite que los distintos órganos judiciales, fiscales, forenses o cualquier otro profesional interno o externo puedan incorporarse a un proceso con la debida seguridad, pudiendo intervenir en un expediente único para su resolución. Este sistema permite el

registro y reparto de todos los documentos que pueden presentarse ante la Administración de Justicia permitiendo la tramitación de dichos asuntos. A su vez permite la remisión de información a otras oficinas y servicios jurisdiccionales. El sistema está totalmente integrado con la fiscalía y posee interoperabilidad con otros organismos. Cuenta además con un sistema integrado de señalamientos y permite las búsquedas de la información y la explotación de la información registrada. (13)

Sistema de gestión de notificaciones telemáticas (LEXNET): es una plataforma española de intercambio seguro de información entre los órganos judiciales y una gran diversidad de operadores jurídicos que, en su actividad diaria, necesitan intercambiar documentos judiciales como notificaciones, escritos y demandas. El funcionamiento de Lexnet se basa en un sistema de correo electrónico seguro, con firma electrónica, a través del cual el usuario recibe las notificaciones emitidas por el juzgado y presenta los escritos por vía telemática. Posteriormente, y a través del mismo sistema, recibe un resguardo electrónico en el que se acredita que la transmisión se ha efectuado correctamente y se le comunica la fecha efectiva de la presentación de dicho escrito en la oficina judicial o juzgado correspondiente. El sistema Lexnet posee las siguientes funcionalidades:

- ✓ Presentación de documentos y adjuntos al órgano judicial.
- ✓ Envío de notificaciones por parte del órgano judicial.
- ✓ Emisión de acuses de recibo. Acceso autenticado y seguro mediante navegador Web.
- ✓ Acceso e intercambio de documentos mediante Web Services.
- ✓ Gestión de carpetas de usuario.
- ✓ Búsqueda y traza de mensajes.

El principal problema de este sistema es que para llevar a cabo el proceso de firma de la comunicación por quien la envía, utiliza un binario ActiveX (nativo de determinadas versiones de Windows y de Internet Explorer). Esto implica problemas de evolución de la infraestructura informática, de falta de interoperabilidad con otras plataformas cliente como puedan ser MacOS o Linux, con otros navegadores como Firefox, Opera, Safari y Chrome/Chromium, e incluso algunas versiones de Microsoft Internet Explorer. (14)

Sistema de gestión procesal (MINERVA-NOJ): es una aplicación de gestión procesal aplicada en España, que soporta la tramitación de la información relativa a los procedimientos judiciales, de forma que cualquier órgano judicial implicado en la tramitación de un determinado procedimiento, pueda acceder a la información asociada al mismo con las garantías de reserva, control y confidencialidad requeridas. El sistema permite además, la obtención de todo tipo de libros, consultas y estadísticas para los órganos y sedes en los que está implantado, dando cobertura a cualquier tipo de órgano judicial existente: desde los unipersonales (Juzgados de Primera Instancia, de Instrucción) hasta los órganos colegiados (Audiencias Provinciales, Tribunales Superiores de Justicia), además de los Órganos Centrales (Tribunal Supremo y Audiencia Nacional), y oficinas de

apoyo (Oficina de Asistencia a las Víctimas de la Audiencia Nacional). El sistema permite la adecuación del aplicativo a la estructura organizativa de la Oficina Judicial, lo que implica la posibilidad de configurar todo tipo y número de servicios comunes procesales. Facilita la gestión de usuarios permitiendo a los secretarios judiciales definir el perfil de trabajo de cada persona, en función de los órganos y tipos de procedimientos a los que puede acceder para tramitar, consultar y otras acciones. La tramitación guiada, implica que, una vez realizado un trámite, el sistema propone inmediatamente el siguiente o permite elegir entre varios posibles. La desventaja principal de este sistema es que es nativo del sistema operativo Windows y que para su uso se necesita pagar la licencia ya que es un software privativo. (15)

Sistema para la gestión jurídica (LEX-DOCTOR): herramienta informática argentina que asimila la experiencia de más de 700 Juzgados, Tribunales, Receptorías Generales de Expedientes, y otras Oficinas Judiciales de distintos Poderes Judiciales del país.

Provee innovaciones técnicas de gran envergadura, proyectadas desde nuestra experiencia única a nivel nacional, de publicar Oficinas Judiciales para consulta por internet o red privada, en línea y en tiempo real, con costos extremadamente reducidos, y extraordinaria eficiencia.

Entre sus generalidades se encuentran:

- ✓ Especialmente desarrollado para el manejo integral de Juzgados, Tribunales, Receptorías Generales de Expedientes, Defensorías, Fiscalías, Cámaras, y demás Oficinas Judiciales y dependencias similares del Poder Judicial.
- ✓ Es multinstancia y fuero[3], destacándose la alta parametrización y configuración de sus elementos para someterlo a las particularidades propias de cada organismo.
- ✓ Cumple funciones integrales de gestión jurídica, incluyendo procesamiento de textos, actualizaciones monetarias, agenda y vencimientos, listados e informes de todas sus tablas, manejo de lápiz óptico para operaciones masivas, administración y consulta de doctrina, legislación y jurisprudencia.
- ✓ Permite el manejo y seguimiento de cada causa a partir de su ingreso en la mesa de entradas hasta la finalización del trámite procesal. (16)

Sistema Automatizado de Gestión Judicial (SAGJ): es un conjunto de elementos orientados al almacenamiento, procesamiento, administración de datos y consulta de información, el cual fue desarrollado esencialmente para la gestión electrónica y digital de los expedientes que cursan trámite ante los tribunales del Órgano Judicial de la República de Panamá.

La plataforma tecnológica de este sistema de información está constituida por un grupo de módulos o aplicaciones informáticas que pueden trabajar de forma independiente y en conjunto, para brindar servicios integrales a los usuarios de la Administración de Justicia, tales como, el módulo de Reparto

[3] Fuero: Compilación o código general de leyes

del Registro Único de Entrada (RUE), la solución informática para la tramitación del Expediente Judicial Electrónico (EJE), el Tarjetero Electrónico y las Certificaciones de Depósito Judicial (CDJ), entre otros. (17)

Sistemas informáticos existentes en Cuba

Sistema para la Tramitación de Procesos Penales (SISPROP): es un software que facilita la tramitación de los procesos penales, con agilidad y precisión, en las entidades del Sistema de Tribunales Populares del país. Dos de las características fundamentales del sistema son su seguridad dada la naturaleza de la información que se maneja, y la flexibilidad con respecto a cualquier modificación que pudiera sufrir la Ley de Procedimiento Penal. El sistema presenta un grupo de limitaciones que se resumirán a continuación:

El sistema no obtiene datos de la fase judicial del proceso. No aporta estadística, ni información alguna y al no haberse programado la introducción de los datos de la fase judicial señalados en el punto uno no es solucionable. El nivel de informatización que supone el sistema es mínimo y en el caso de las apelaciones es prácticamente nulo. El sistema no valida casi ningún dato. Este sistema está programado en Delphi utilizando como sistema de gestión de bases de datos a SQL Server 2000 por lo que queda claro que no es compatible con el software libre. (18)

Sistema de Información para la Gestión de los Tribunales Populares Cubanos (SITPC): Consiste en una aplicación Web que permite a los usuarios (fiscales, jueces, abogados, secretarios y personas naturales o jurídicas), de acuerdo a los permisos otorgados, realizar todos los trámites procesales que corresponden legalmente, y a los funcionarios correspondientes controlar en tiempo real el funcionamiento de la actividad judicial. Es un sistema que posibilita entre otras funcionalidades generales, la presentación telemática de escritos, la radicación automática de expedientes, el procesamiento de textos con modelos de documentos, y las notificaciones electrónicas a las partes; todo desde la perspectiva del expediente digital.

Dentro de las características más importantes del sistema se encuentran:

- ✓ Captación de los datos en el lugar de origen lo que ayuda a evitar errores en la trascripción de documentos.
- ✓ Alerta a los usuarios sobre el vencimiento de los términos de un expediente evitando el incumplimiento en la realización de los actos procesales.
- ✓ Control de la consecutividad del proceso que permite no alterar la tramitación de un expediente.
- ✓ Generación dinámica de documentos, lo cual contribuye a su estandarización.
- ✓ Turnado[4] automático o manual de los procesos al juez ponente, de acuerdo a criterios configurables.

[4] Asignación a un juez de un expediente.

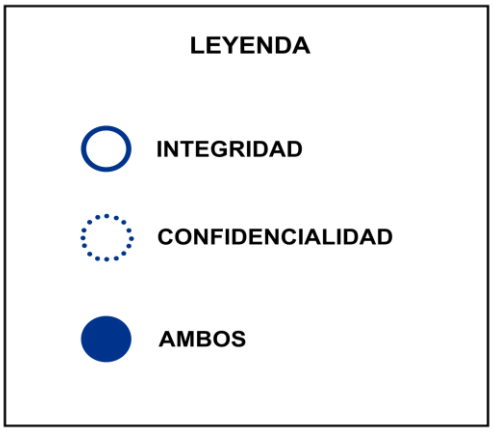
- ✓ Radicación automática, permitiendo la generación de un único número de expediente a nivel de país.
- ✓ Enumeración automática de escritos presentados y resoluciones dictadas. (19)

El estudio realizado sobre los sistemas similares queda resumido en la tabla 1, el mismo permitió tener una visión más clara sobre la situación existente puesto a que ninguno de los sistemas analizados ofrece un módulo que sea capaz de integrarse a el sistema XEJEL y responder ante la necesidad actual. Para los sistemas Avantius web y Lexnet se pudo ver que estos además de no ser privativos, aseguran la integridad y la confidencialidad, pero no atienden la gestión de estructura, indicador a tener en cuenta en el momento de reutilizar o desarrollar el nuevo componente de seguridad. Automáticamente se procedió a descartar los sistemas Minerva-noj, Lex-doctor y SAGJ dado su estado privativo. El sistema SISPROP tampoco se tuvo en cuenta como opción, pues el mismo no garantiza la confidencialidad. Finalmente, el sistema que cumple con todos los indicadores analizados es SITPC, pero a pesar de ello fue desarrollado con tecnología que actualmente es obsoleta, no obstante, se tuvieron en cuenta las buenas prácticas y ventajas que este ofrece, entre las que podemos mencionar: la necesidad de mantener la gestión de las estructuras, el acceso a la aplicación basado en el cargo de los usuarios, así como la gestión de las acciones que se pueden realizar en el sistema y la gestión de los usuarios.

Tabla 1 Resumen según indicadores

Nombre del Sistema	Nivel de acceso por roles	Usuarios	Estructuras	Funcionalidades	Es privativo
AVANTIUS WEB	SI	SI	NO	SI	NO
LEXNET	SI	SI	NO	SI	NO
MINERVA-NOJ	SI	SI	SI	SI	SI
LEX-DOCTOR	SI	SI	SI	SI	SI
SAGJ	SI	SI	SI	SI	SI
SISPROP	SI	SI	NO	NO	NO
SITPC	SI	SI	SI	SI	NO

Fuente: Elaboración propia



1.4 Metodología de desarrollo

Para dar respuesta al problema existente se hace necesario definir los pasos a seguir seleccionando la metodología adecuada. Siendo, las metodologías un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. La selección de la metodología a emplear, propone los roles que el equipo de desarrollo deberá cubrir, las actividades que debe cumplir cada uno de ellos, los artefactos que serán generados de cada tarea, así como el registro de cada detalle de la información que se irá generando. (20)

1.4.1 Metodología AUP-UCI

El Proceso Unificado Ágil de Scott Ambler o Agile Unified Process (AUP) en inglés es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP.

Al no existir una metodología de software universal en los proyectos desarrollados en la UCI, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable, se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva. Con dicha adaptación se logra estandarizar el proceso de desarrollo de software, dando cumplimiento además a las buenas prácticas que define CMMI-DEV[5] v1.3 las cuales se centran en el desarrollo de productos y servicios de calidad. Se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos. (20)

Descripción de las Fases de AUP-UCI

- ✓ Inicio: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto
- ✓ 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. (20)

Se define utilizar esta metodología en su escenario número 4 el cual aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. Este escenario no modela el negocio solo pueden modelar el sistema a través de Historias de usuario. Además, el cliente siempre estará acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos.

1.5 Tecnologías y herramientas

Realizar el estudio de las herramientas y tecnologías que se utilizarán para el desarrollo de la propuesta de solución es de vital importancia, comenzando por conocer que las herramientas son programas, aplicaciones o simplemente instrucciones que ofrecen la posibilidad de realizar varias funcionalidades con diferentes propósitos. (21) Mientras que, las tecnologías son el conjunto de conocimientos técnicos, ordenados científicamente que permiten diseñar y crear bienes y servicios. (22) La correcta selección de las herramientas y tecnologías informáticas a través de un estudio profundo de las características que las identifican, permiten agilizar y facilitar el desarrollo de los

[5] Integración de sistemas modelos de madurez de capacidades para desarrolladores o Capability Maturity Model Integration for Development (CMMI-DEV)

sistemas informáticos. A continuación, se presentan los referentes teóricos de las diferentes herramientas y tecnologías utilizadas durante el desarrollo de la solución.

1.5.1 Lenguajes de programación

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina. (23)

PHP 7.2

PHP (acrónimo recursivo de *PHP:* Hypertext Preprocessor *o* Preprocesador de Hipertexto) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML[6].

Lo que distingue a PHP del lenguaje utilizado del lado del cliente como Javascript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP. Lo mejor de utilizar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales. (24)

PHP puede emplearse en todos los sistemas operativos principales, incluyendo Linux, muchas variantes de Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS y probablemente otros más. PHP admite la mayoría de servidores web de hoy en día, incluyendo Apache, IIS, y muchos otros.

De modo que, con PHP, se tiene la libertad de elegir el sistema operativo y el servidor web. Además, se tiene la posibilidad de utilizar programación por procedimientos o programación orientada a objetos (POO), o una mezcla de ambas. (25)

TypeScript 2.7

TypeScript es un lenguaje de programación de código abierto desarrollado por Microsoft, el cual cuenta con herramientas de programación orientada a objetos, muy favorable si se tienen proyectos grandes. Anders Hejlsberg, arquitecto principal del desarrollo del lenguaje de programación C#, es el principal participante en el desarrollo de este lenguaje.

TypeScript convierte su código en Javascript común. Es llamado también Superset de Javascript, lo que significa que si el navegador está basado en Javascript, este nunca llegará a saber que el código original fue realizado con TypeScript y ejecutará el Javascript como lenguaje original. (26)

[6] Lenguaje de Marcas de Hipertexto (HyperText Markup Language).

CSS 3

CSS significa Cascade Style Sheets, también llamado Hojas de Estilo en Cascada. CSS es un lenguaje de marcado que se emplea para dar formato a un sitio web. Es decir, funciona en conjunto con los archivos HTML. El lenguaje CSS3 se puede aplicar en la misma hoja en la que estás desarrollando un documento HTML, pero por motivos de productividad se suele realizar en un documento aparte con la extensión .css. Este documento se puede vincular a cada página HTML que conforme el sitio web, es por ello que es más útil realizar los estilos por separado. (27)

HTML 5

HTML, que significa Lenguaje de Marcado para Hipertextos (HyperText Markup Language) es el elemento de construcción más básico de una página web y se usa para crear y representar visualmente una página web. HTML le da "valor añadido" a un texto estándar en español e inglés. Hipertexto se refiere a enlaces que conectan una página Web con otra, ya sea dentro de una página web o entre diferentes sitios web ya que los vínculos son un aspecto fundamental de la Web. (28)

HTML5 es la quinta revisión (mayor) de este estándar. Las principales novedades que trae son nuevas etiquetas para conseguir la Web Semántica (que los elementos o etiquetas aporten significado y no solo contenido) y nuevas APIs[7] para permitir funcionalidades avanzadas de JavaScript. (29)

1.5.2 Lenguaje de modelado

UML 2.0

Para el modelado de los procesos de la solución se utiliza el Lenguaje de Modelado Unificado UML (Unified Modeling Language), dicho lenguaje incluye las clases, esquemas de base de datos y componentes reutilizables de software en un lenguaje determinado y soporta el paradigma orientado a objetos.

El Lenguaje de Modelado Unificado (UML), es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad, respaldado el mismo por el Grupo de Administración de Objetos OMG (por sus siglas en inglés de Object Management Group). UML tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases y objetos, hasta la implementación y configuración con los diagramas de despliegue. (30)

En resumen, UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software y está compuesto por diversos elementos gráficos que se combinan para conformar diagramas, los cuales tiene la finalidad de presentar diversas perspectivas de un sistema. (31)

[7] Siglas de Interfaz de Programación de Aplicaciones ('Application Programming Interface')

Visual Paradigm 8.0

Visual Paradigm es una herramienta CASE[8] de modelado visual que facilita la construcción de artefactos en un proceso de desarrollo de software mediante el Lenguaje de Modelado Unificado (UML). Esta herramienta resulta ser una potente multiplataforma y a su vez es fácil de usar. Proporciona a los desarrolladores de software una plataforma de desarrollo para construir aplicaciones de calidad más óptimas y de menor costo. Posee capacidades de ingeniería directa e inversa, puede integrarse a los principales entornos de desarrollo y posee disponibilidad de múltiples versiones para cada necesidad. (32)

Las ventajas que proporciona Visual Paradimg para UML son:

- ✓ Dibujo: facilita el modelado de UML, ya que proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta.
- ✓ Corrección sintáctica: controla que el modelado con UML sea correcto.
- ✓ Coherencia entre diagramas: al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades.
- ✓ Integración con otras aplicaciones: permite integrarse con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad.
- ✓ Trabajo multiusuario: permite el trabajo en grupo, proporcionando herramientas de compartición de trabajo.
- ✓ Reutilización: facilita la reutilización, ya que disponemos de una herramienta centralizada donde se encuentran los modelos utilizados para otros proyectos.
- ✓ Generación de código: permite generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software.
- ✓ Generación de informes: permite generar diversos informes a partir de la información introducida en la herramienta. (33)

1.5.3 Marcos de trabajo

Symfony 3.4

Symfony es un framework (marco de trabajo) muy completo, que está diseñado para optimizar el desarrollo de aplicaciones web. Este framework separa la lógica del negocio, la lógica del servidor y la presentación de la aplicación web. También mediante el uso de varias herramientas reduce el tiempo de desarrollo de una aplicación web compleja. A parte de esto, para el desarrollador supone una gran ventaja su uso, ya que, al automatizar las tareas más comunes, dicho desarrollador puede dedicarse por completo a los aspectos específicos de cada aplicación.

[8] Siglas en inglés de Computer Aided Software Engineering lo que en español sería Ingeniería de Software Asistida por Ordenador.

Symfony está completamente desarrollado en PHP y es compatible con la mayoría de gestores de base de datos, y además se puede ejecutar tanto en plataformas Linux o Unix, como en plataformas Windows.

Algunas de las principales características de Symfony:

- ✓ Es fácil de instalar y configurar en la mayoría de plataformas.
- ✓ Independiente del sistema gestor de base de datos.
- ✓ Utiliza programación orientada a objetos y características como los espacios de nombres.
- ✓ Fácil de usar, aunque preferiblemente para el desarrollo de grandes aplicaciones web.
- ✓ Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- ✓ Preparado para aplicaciones empresariales y adaptable a las políticas de cada empresa.
- ✓ Código fácil de leer que incluye comentarios de phpDocumentor.
- ✓ Potente línea de comandos que facilitan la generación de código, esto es fundamental para ahorrar tiempo de trabajo. (34)

Bootstrap 4.1

Bootstrap es un framework CSS desarrollado inicialmente (en el año 2011) por Twitter que permite dar forma a un sitio web mediante librerías CSS que incluyen tipografías, botones, cuadros, menús y otros elementos que pueden ser utilizados en cualquier sitio web. Aunque el desarrollo del framework Bootstrap fue iniciado por Twitter, fue liberado bajo licencia MIT[9] en el año 2011 y su desarrollo continúa en un repositorio de GitHub.

Bootstrap es una excelente herramienta para crear interfaces de usuario limpias y totalmente adaptables a todo tipo de dispositivos y pantallas, sea cual sea su tamaño. Además, Bootstrap ofrece las herramientas necesarias para crear cualquier tipo de sitio web utilizando los estilos y elementos de sus librerías. (35)

Angular 6.0

Angular es un framework de desarrollo creado por Google. La finalidad de Angular es facilitar el desarrollo de aplicaciones web SPA[10] y además brindar herramientas para trabajar con los elementos de una web de una manera más sencilla y óptima. Otro propósito que tiene Angular es la separación completa entre el front-end y el back-end en una aplicación web.

Ventajas de Angular

[9] Instituto Tecnológico de Massachusetts (**MIT**, Massachusetts Institute of Technology).
[10] SPA es el acrónimo de 'Single Page Application'.

- ✓ Lenguaje Typescript, tiene una sintaxis muy parecida a Java, con tipado estático.
- ✓ Basado en componentes, es decir, podemos escribir componentes web con vista y lógica para después reutilizarlos en otras páginas.
- ✓ Comunidad muy grande con multitud de tutoriales y librerías.
- ✓ Inyección de dependencias, un patrón de diseño que se basa en pasar las dependencias directamente a los objetos en lugar de crearlas localmente.
- ✓ Programación reactiva, la vista se actualiza automáticamente a los cambios.
- ✓ Dispone de asistente por línea de comandos para crear proyectos base (Angular cli).
- ✓ Se integra bien con herramientas de testing[11].
- ✓ Se integra bien con Ionic[12], para adaptar aplicaciones web a dispositivos móviles. (36)

1.5.4 Librerías

jQuery 3.3

jQuery es una librería JavaScript open-source, que funciona en múltiples navegadores, y que es compatible con CSS3. Su objetivo principal es hacer la programación "scripting"[13] mucho más fácil y rápida del lado del cliente. Con jQuery se pueden producir páginas dinámicas, así como animaciones parecidas a Flash en relativamente corto tiempo. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio. (37)

Doctrine 2.0

Doctrine Project es un conjunto de librerías PHP entre las que se encuentra su mapeo objeto-relacional ORM por sus siglas en inglés (Object-Relational Mapper). Las entidades en Doctrine 2 son objetos PHP que contienen variables (propiedades) que se guardan y devuelven a una base de datos a través del Entity Manager de Doctrine, una implementación del patrón de mapeador de datos. Su principal característica es la poca configuración que hace falta para empezar un proyecto.

Un ORM (Object Relational Mapping) es una técnica de programación que convierte datos entre sistemas de tipos de un lenguaje de programación orientado a objetos y una base de datos relacional. El resultado es una base de datos orientada a objetos que puede usarse desde el lenguaje de programación utilizado: las clases son tablas de la base de datos y los objetos son registros. Así resulta más fácil crear y manipular tablas y datos. (38)

[11] Pruebas de calidad del software.
[12] Ionic es una herramienta, gratuita y Open Source, para el desarrollo de aplicaciones híbridas basadas en HTML5, CSS3, JavaScript o Typescript.
[13] Tipo de lenguaje de programación que está diseñado para integrarse y comunicarse con otros lenguajes de programación.

1.5.5 Gestor de base de datos: PostgreSQL 10.1

PostgreSQL es un poderoso sistema de base de datos relacional de objetos de código abierto. Tiene más de 15 años de desarrollo activo y una arquitectura comprobada que le ha valido una sólida reputación de fiabilidad, integridad de datos y corrección. Se ejecuta en todos los principales sistemas operativos, incluidos Linux, UNIX (AIX, BSD, HP-UX, macOS, Solaris) y Windows.

Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multi-hilos para garantizar la estabilidad del sistema. Admite conjuntos de caracteres internacionales, codificaciones de caracteres multibyte, Unicode, y es compatible con la configuración regional para la clasificación, la distinción entre mayúsculas y minúsculas y el formateo. Es altamente escalable tanto en la gran cantidad de datos que puede administrar como en la cantidad de usuarios simultáneos que puede acomodar. (39)

Herramienta para la gestión de base de datos: PgAdmin4 v4.8

Es una aplicación de diseño y manejo de bases de datos para su uso con postgreSQL. Se puede utilizar para manejar postgreSQL 7.3 y superiores y funciona sobre casi todas las plataformas. Este software fue diseñado para responder a las necesidades de todos los usuarios, desde la escritura de simples consultas SQL (Structured Query Language)[14] a la elaboración de bases de datos complejas. La interfaz gráfica es compatible con todas las características de postgreSQL y facilita la administración.

Una característica interesante de pgAdmin4 es que, cada vez que se realiza alguna modificación en un objeto, escribe la(s) sentencia(s) SQL correspondiente(s), lo que hace que, además de una herramienta muy útil, sea a la vez didáctica. También incorpora funcionalidades para realizar consultas, examinar su ejecución (como el comando explain) y trabajar con los datos. (40)

1.5.6 Entorno de desarrollo integrado: NetBeans 8.0

Netbeans es un entorno de desarrollo gratuito y de código abierto. Permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, como aplicaciones Web, o para dispositivos móviles. Da soporte a las siguientes tecnologías, entre otras: Java, PHP, Groovy, C/C++, HTML5. Además, puede instalarse en varios sistemas operativos: Windows, Linux, Mac OS.

Entre sus características principales se pueden encontrar:

- ✓ Suele dar soporte a casi todas las novedades en el lenguaje PHP. Cualquier avance del lenguaje es rápidamente soportado por Netbeans.
- ✓ Asistentes para la creación y configuración de distintos proyectos, incluida la elección de algunos frameworks.

[14] En español, Lenguaje de Consulta Estructurado.

- ✓ Simplifica la gestión de grandes proyectos con el uso de diferentes vistas, asistentes de ayuda, y estructurando la visualización de manera ordenada, lo que ayuda en el trabajo diario.
- ✓ Herramientas para depurado de errores: el debugger que incluye el IDE[15] es bastante útil para encontrar dónde se producen errores.
- ✓ Optimización de código: por su parte el Profiler ayuda a optimizar las aplicaciones e intentar hacer que se ejecuten más rápido y con el mínimo uso de memoria.
- ✓ Acceso a base de datos: el propio Netbeans se puede conectar a distintos sistemas gestores de bases de datos, como pueden ser Oracle, MySql y demás, y ver las tablas, realizar consultas y modificaciones, y todo ello integrado en el propio IDE.
- ✓ Se integra con diversos servidores de aplicaciones, de tal manera que se pueden gestionar desde el propio IDE: inicio, parada, arranque en modo debug, despliegues. (41)

1.5.7 Herramientas para el control de versiones

Git

Git es un sistema de control de versiones distribuido, gratuito y de código abierto, diseñado para manejar todo, desde proyectos pequeños hasta proyectos muy grandes con rapidez y eficiencia.

Git es fácil de aprender y tiene un rendimiento increíblemente rápido. Supera las herramientas de gestión de configuración de software (Software Configuration Management, SCM por sus siglas en inglés) como Subversion, CVS, Perforce y ClearCase con características como bifurcaciones locales baratas, áreas de preparación convenientes y múltiples flujos de trabajo. (42)

La principal diferencia entre Git y cualquier otro sistema de control de versiones (Subversion y amigos incluidos) es la forma en que Git piensa acerca de sus datos. Conceptualmente, la mayoría de los otros sistemas almacenan información como una lista de cambios basados en archivos. Estos otros sistemas (CVS, Subversion, Perforce, Bazaar, etc.) consideran la información que almacenan como un conjunto de archivos y los cambios realizados en cada archivo a lo largo del tiempo (esto se describe comúnmente como control de versiones basado en delta). (43)

GitLab

GitLab es una potente solución de software libre aplicable solo para Git que va a permitir a crear y gestionar repositorios de código y documentos. Con GitLab se podrá crear un servicio en el propio servidor o utilizando los servicios que ofrece para gestionar con facilidad cambios y versiones de los propios ficheros o de los ficheros de otras personas. Especialmente utilizado por equipos de desarrolladores de aplicaciones cuando están repartidos a lo largo del mundo. (44)

[15] En inglés Integrated Development Environment lo que se traduce al español como entorno de desarrollo integrado o entorno de desarrollo interactivo.

1.5.8 Servidor web: Apache 2.4

Apache es un poderoso servidor web, cuyo nombre proviene de la frase inglesa “a patchy server” y es completamente libre, ya que es un software de código abierto. Una de las ventajas más grandes de Apache, es que es un servidor web multiplataforma, es decir, puede trabajar con diferentes sistemas operativos y mantener su excelente rendimiento.

Desde el año 1996, es el servidor web más popular del mundo, debido a su estabilidad y seguridad. Apache sigue siendo desarrollado por la comunidad de usuarios desarrolladores que trabaja bajo la tutela de Apache Software Foundation. Entre las principales características de Apache, se encuentran las siguientes:

- ✓ Soporte de seguridad SSL y TLS.
- ✓ Puede realizar autentificación de datos utilizando SGDB.
- ✓ Puede dar soporte a diferentes lenguajes, como Perl, PHP, Python y otros. (45)

1.6 Arquitectura de software

La Arquitectura de Software es la organización fundamental de un sistema representada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución (46).

A continuación, se presentan los referentes teóricos de las dos arquitecturas que se seleccionaron para el desarrollo de la propuesta de solución.

Arquitectura basada en capas

La arquitectura basada en capas se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada.

El estilo de arquitectura basado en capas se identifica por las siguientes características:

- ✓ Describe la descomposición de servicios de forma que la mayoría de la interacción ocurre solamente entre capas vecinas.
- ✓ Las capas de una aplicación pueden residir en la misma máquina física (misma capa) o puede estar distribuido sobre diferentes computadores (n-capas).
- ✓ Los componentes de cada capa se comunican con otros componentes en otras capas a través de interfaces muy bien definidas.
- ✓ Este modelo ha sido descrito como una “pirámide invertida de re-uso” donde cada capa agrega responsabilidad y abstracción a la capa directamente sobre ella. (47)

Arquitectura basada en componentes

Una arquitectura basada en componentes describe una aproximación de ingeniería de software al diseño y desarrollo de un sistema. Esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Esto provee un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma como se comparte el estado.

El estilo de arquitectura basado en componentes tiene las siguientes características:

- ✓ Es un estilo de diseño para aplicaciones compuestas de componentes individuales.
- ✓ Pone énfasis en la descomposición del sistema en componentes lógicos o funcionales que tienen interfaces bien definidas.
- ✓ Define una aproximación de diseño que usa componentes discretos, los que se comunican a través de interfaces que contienen métodos, eventos y propiedades. (48)

1.7 Modelos de control de acceso

En cualquier sistema informático, especialmente si es multiusuario, el control de acceso sobre usuarios y recursos del mismo es un pilar fundamental para su seguridad. Por ello, es de gran importancia contar con mecanismos que proporcionen una apropiada segregación de privilegios y permisos de usuario, así como de la administración de los mismos y de los elementos relacionados. Un modelo de control de acceso (MCA) es un conjunto definido de criterios que un administrador del sistema utiliza para definir derechos/permisos de los usuarios del/al sistema. Los MCA son metodologías en las que se gestiona y organiza la admisión a áreas físicas, y lo que es más importante, a los sistemas informáticos (49).

En dependencia de las restricciones y control de acceso al sistema, se pueden encontrar tres tipos principales de MCA:

Control de Acceso Obligatorio (Mandatory Access Control, MAC): en el modelo de control de acceso obligatorio se asignan las funciones de los usuarios estrictamente de acuerdo a los deseos del administrador del sistema. Este es el método de control de acceso más restrictivo, porque el usuario final no puede establecer controles de acceso en los archivos. El Control de acceso obligatorio es muy popular en ambientes/instalaciones altamente secretas, como la industria de defensa donde los archivos «perdidos» pueden afectar a su seguridad nacional (50).

Control de Acceso Discrecional (DAC): el control discrecional de acceso está en el otro extremo del espectro de acceso, diferente del modelo de acceso obligatorio, ya que es el menos restrictivo de los tres modelos. En el marco del modelo de acceso discrecional el usuario final tiene total libertad para asignar los derechos a los objetos que desea.

Este nivel de control completo sobre los archivos puede ser peligroso porque si un atacante o algún Malware compromete la cuenta a continuación, el usuario malicioso o código tendrá un control completo también (50).

Control de Acceso Basado en Roles (RBAC): la función de control de acceso basado en permisos o Roles crea la asignación de derechos/permisos de acceso a funciones o trabajos específicos dentro de la empresa; RBAC asigna funciones a los usuarios, con lo que le concede privilegios. Este modelo de control de acceso a las funciones de manera efectiva en las organizaciones reales es, debido a que a los archivos y los recursos se le asignan los permisos de acuerdo a las funciones que lo requieran. Por ejemplo, un administrador del sistema puede crear una función de acceso para los gerentes solamente. Así, un usuario se le tendría que ser asignado el papel de un gerente para utilizar esos recursos (50).

Se quiere lograr con la propuesta de solución, una aplicación que gestione el acceso de las personas según su cargo, otorgándole los permisos correspondientes al mismo, por lo que, para su desarrollo, se determinó utilizar el modelo RBAC debido a que, de los tres modelos analizados, este es el que más se ajusta según sus características y funcionamiento a la necesidad existente.

1.8 Patrones de diseño

Los patrones de diseño son unas técnicas para resolver problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. (49)

Patrones GRASP

Los patrones GRASP[16] describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. (50)

Los patrones GRASP definidos a utilizar en el diseño de la propuesta de solución son:

Experto: El experto en información es el principio básico de asignación de responsabilidades. 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. De este modo se obtendrá un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento) (51).

[16] Patrones Generales de Software para Asignación de Responsabilidades (GRASP por sus siglas en inglés).

Controlador: El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado (51).

Creador: El patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases (51).

Fabricación: La fabricación se da en las clases que no representan un ente u objeto real del dominio del problema, sino que se ha creado intencionadamente para disminuir el acoplamiento, aumentar la cohesión y/o potenciar la reutilización del código. Las clases de fabricación casi siempre se dividen atendiendo a su funcionalidad, dicho con otras palabras, se confeccionan clases destinadas a conjuntos de funciones (51).

Patrones GoF

Los patrones GoF[17] son patrones de diseño; existen 23 patrones de este tipo y se clasifican según su propósito en Creacionales, Estructurales y de Comportamiento. Teniéndose que:

- ✓ Creacionales: Conciernen al proceso de creación de objetos.
- ✓ Estructurales: Tratan con la composición de las clases o los objetos.
- ✓ Comportamiento: Caracterizan la vía en que las clases u objetos interactúan y distribuyen las responsabilidades. (52)

Los patrones GoF tienen como características que:

- ✓ Son soluciones concretas, o sea, proponen soluciones a problemas concretos y no a las teorías genéricas.
- ✓ Son soluciones técnicas basadas en Programación Orientada a Objetos (POO).
- ✓ Se utilizan en situaciones frecuentes ya que se basan en la experiencia acumulada al resolver problemas reiterativos.
- ✓ Favorecen la reutilización de código. (53)

De este grupo de patrones se define utilizar el patrón:

Observador: Permite observar los cambios producidos por un objeto, de esta forma, cada cambio que afecte el estado del objeto observado lanzará una notificación a los observadores; a esto se le conoce como Publicador-Suscriptor. Es uno de los principales patrones de diseño utilizados en interfaces gráficas de usuario (GUI), ya que permite desacoplar al componente gráfico de la acción a realizar (54).

Otros patrones

[17] Gang of Four,por sus siglas GoF (Grupo de los cuatro nombrado así por los cuatro autores del libro Desing Patterns)

Inyección de dependencias: Este es un patrón de diseño de software usado en la Programación Orientada a Objetos, que trata de solucionar las necesidades de creación de los objetos de una manera práctica, útil, escalable y con una alta versatilidad del código (55).

1.1 Conclusiones parciales

En este capítulo se realizó un estudio sobre los principales conceptos asociados al objeto de estudio para una mejor comprensión del mismo. El estudio de los sistemas existentes a nivel internacional y nacional permitió confirmar la necesidad de crear un nuevo componente de seguridad para XEJEL, ya que ninguno satisfacía las necesidades planteadas; además permitió detectar las buenas prácticas de estos sistemas y reutilizarlas en la implementación del nuevo componente de seguridad que satisfaga los requerimientos del expediente judicial electrónico. Finalmente, el estudio de las características de la metodología de desarrollo, las herramientas y tecnologías, así como las arquitecturas y patrones a utilizar, permitió tener mejor conocimiento de los mismos y garantizar su correcta utilización en el desarrollo de la propuesta de solución.

Capítulo 2: Propuesta de solución

Introducción

En este capítulo se presenta la descripción general de la propuesta solución. Para dar cumplimiento a ello y teniendo como guía la metodología seleccionada se lleva a cabo la disciplina de Requisitos la cual permite que se engloben los requisitos funcionales y no funcionales por los cuales se va a regir la propuesta de solución. Se obtienen los artefactos correspondientes a la disciplina de Análisis y diseño. Finalmente, quedan definidos los estándares de implementación a utilizar como base para la implementación.

2.1 Descripción general de la propuesta de solución

Con el desarrollo del sistema Expediente Judicial Electrónico se tiene como objetivo informatizar la gestión de los expedientes que se tramitan en los TPC, dichos expedientes constituyen el instrumento fundamental que resulta de la agregación de las distintas actuaciones y en tal sentido toda la información y documentos que lo conforman es sensible y confidencial, por lo que el desarrollo de un componente que gestione la Seguridad resulta de vital importancia, garantizando la misma desde dos aristas fundamentales : la integridad y la confidencialidad acorde con los principios fundamentales de la Seguridad Informática. Para ello es necesario que en la solución propuesta queden definidas las estructuras de los tribunales, así como las funcionalidades o acciones que pueden realizarse sobre los expedientes las cuales quedan asociadas a los roles o cargos que se tienen en cada una de las estructuras. Estos roles se les asignan a los usuarios los cuales se traducen en las personas que van a tener acceso al sistema. De modo que, cuando un usuario se autentique en el sistema, lo primero que se muestra es un menú izquierdo mostrando todas las funcionalidades que va a poder realizar y una vez seleccionada una de ellas el sistema muestra solo los expedientes a los que va a tener acceso dicho usuario.
La unión de estos cuatro componentes (Usuarios, Roles, Estructuras y Funcionalidades) garantiza el cumplimiento de la integridad y confidencialidad de la información almacenada en los expedientes que se tramitan en cada una de las estructuras. La figura 1 evidencia lo anteriormente descrito.

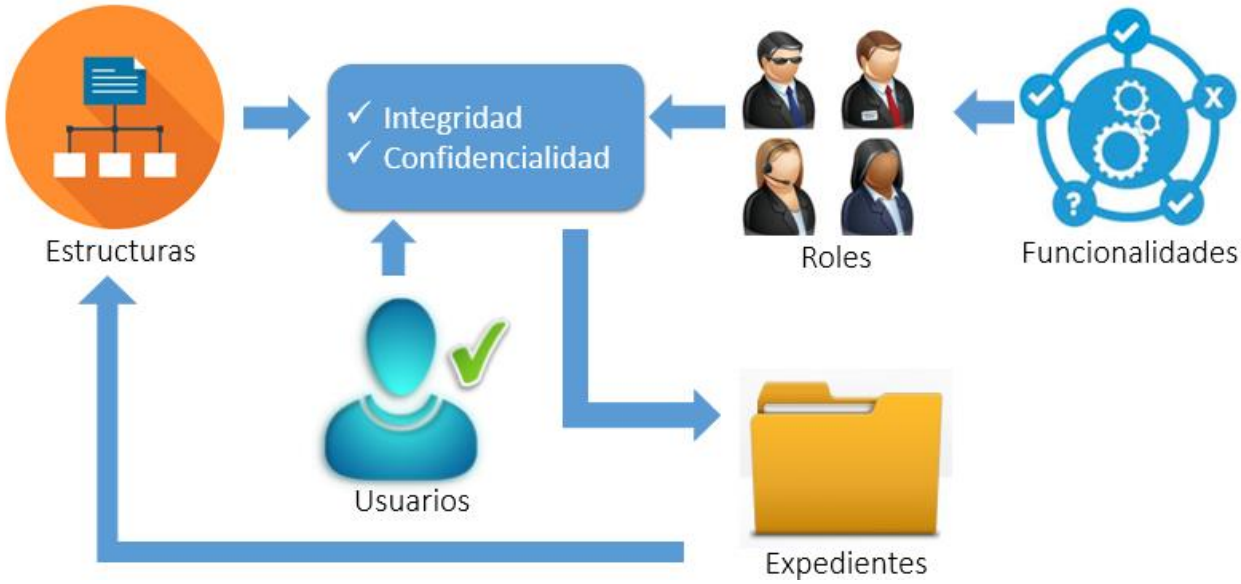


Figura 1 Descripción gráfica de la propuesta de solución

Fuente: Elaboración propia

2.2 Requisitos

Para un correcto diseño e implementación de la propuesta solución es necesario conocer cuáles son los requisitos del cliente, conociendo que: los requisitos para un sistema son descripciones de lo que el sistema debe hacer, el servicio que ofrece y las restricciones en su operación. Tales requisitos reflejan las necesidades de los clientes por un sistema que atienda cierto propósito, como sería controlar un dispositivo, colocar un pedido o buscar información. En algunos casos, un requisito es simplemente un enunciado abstracto de alto nivel en un servicio que debe proporcionar un sistema, o bien, una restricción sobre un sistema. En el otro extremo, consiste en una definición detallada y formal de una función del sistema. (56)

2.2.1 Requisitos funcionales

Los requisitos funcionales son enunciados acerca de servicios que el sistema debe proveer, de cómo debería reaccionar el sistema a entradas particulares y de cómo debería comportarse el sistema en situaciones específicas. En algunos casos, los requisitos funcionales también explican lo que no debe hacer el sistema. (56)

Para identificar los requisitos funcionales correspondientes al componente para la gestión de la seguridad en XEJEL, se realizó un levantamiento de información el cual quedó plasmado en el documento “CEGEL_XEJEL_Componete_Seguridad_Levantamiento_de_información” a través de aplicación de la técnica de entrevista con el cliente, logrando obtener un total de 24 requisitos funcionales contenidos en “CEGEL_XEJEL_Componete_Seguridad_Especificación_de_requisitos”. A continuación, se numeran los requisitos funcionales (RF) y se muestra una descripción de los mismos:

Tabla 2 Especificación de Requisitos funcionales

Nº	Nombre	Descripción
RF1	Autenticar usuario	Permite a los usuarios acceder al sistema a partir de los permisos otorgados.
RF2	Adicionar usuario	Permite adicionar los usuarios que podrán acceder al sistema.
RF3	Modificar usuario	Permite modificar el usuario.
RF4	Activar usuario	Permite activar un usuario en el sistema.
RF5	Desactivar usuario	Permite desactivar un usuario en el sistema.
RF6	Listar usuario	Permite mostrar la lista de usuarios en el sistema.
RF7	Asignar rol a usuario	Permite asignarle un rol a un usuario que se está adicionando o modificando en el sistema.
RF8	Eliminar rol del usuario	Permite eliminar un rol que tenga un usuario que se está modificando.
RF9	Adicionar rol	Permite adicionar un rol en el sistema.
RF10	Modificar rol	Permite modificar un rol en el sistema.
RF11	Activar rol	Permite activar un rol en el sistema.
RF12	Desactivar rol	Permite desactivar un rol en el sistema.
RF13	Listar roles	Permite mostrar la lista de roles que existen en el sistema.
RF14	Crear estructura	Permite crear una estructura.
RF15	Modificar estructura	Permite modificar una estructura.
RF16	Activar estructura	Permite activar una estructura.
RF17	Desactivar estructura	Permite desactivar una estructura.
RF18	Listar estructura	Permite listar las estructuras del sistema.
RF19	Adicionar funcionalidad	Permite adicionarle al usuario funcionalidades en el sistema.
RF20	Modificar funcionalidad	Permite modificar las funcionalidades que tendrá el usuario en el sistema.
RF21	Activar funcionalidad	Permite activar una funcionalidad.

RF22	Desactivar funcionalidad	Permite desactivar una funcionalidad.
RF23	Listar funcionalidad	Permite listar las funcionalidades que tendrá el usuario en el sistema.
RF24	Eliminar funcionalidad	Permite eliminar una funcionalidad del sistema.

Fuente: Elaboración propia

2.2.2. Requisitos no funcionales

Describen aspectos del sistema que son visibles por el usuario que no incluyen una relación directa con el comportamiento funcional del sistema. Los requisitos no funcionales incluyen restricciones como el tiempo de respuesta (desempeño), la precisión, recursos consumidos, seguridad. (57)

Estos requisitos se encuentran referenciados en el documento "CEGEL_XEJEL_Componente_Seguridad_Especificacion_de_requisitos". En el anexo 1, se muestran algunos de los que mayor vínculo tienen con el desarrollo del componente para la gestión de la seguridad en XEJEL.

2.3 Historias de usuario

Historias de usuario es la técnica utilizada en el escenario 4 de la metodología AUP-UCI y se utiliza para encapsular los requisitos del software. No son más que tarjetas en las cuales se describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla (58). En el caso del componente para gestión de la seguridad de XEJEL se crearon 24 historias de usuario, las cuales se encuentran reflejadas en el documento *"*CEGEL_XEJEL_Componete_Seguridad_Historias_de_usuario*"* mostrándose a continuación una de estas. En este caso en este caso se toma como ejemplo la HU referente al RF_Adicionar usuario, siendo "usuario" lo que necesita una persona para poder acceder al sistema y a partir del mismo se le asignan las funcionalidades y roles que definen su nivel de acceso a la información por lo que el requisito funcional mencionado es considerado el más importante ya que da entrada a una nueva persona al sistema.

Tabla 3 HU_Adicionar usuario

Número: 2	**Requisito:** Adicionar usuario
Programador: Orel Pérez Camejo	**Iteración Asignada: 1**
Prioridad: Alta	**Tiempo Estimado:** 14 horas
Riesgo en Desarrollo: CEGEL_XEJEL_Plan de Riesgo	**Tiempo Real:** 14 horas

Descripción:

Permite adicionar los usuarios que podrán acceder al sistema.

Campos:

Nombre: Campo de texto obligatorio que permite introducir el nombre del usuario que se va a adicionar al sistema.

Apellidos: Campo de texto obligatorio que permite introducir el apellido del usuario que se va a adicionar al sistema.

Usuario: Campo de texto obligatorio que permite introducir el nombre de usuario de quien se va a adicionar al sistema.

Contraseña: Campo de texto obligatorio que permite introducir la contraseña del usuario que se va a adicionar al sistema.

Confirmar contraseña: Campo de texto obligatorio que permite confirmar la contraseña que se está introduciendo al sistema.

Recibir por correo: Campo de selección obligatorio que permite seleccionar si se desea recibir por correo las alertar, las notificaciones o ambas.

Botones:

Adicionar: Opción que permite adicionar el usuario en el sistema.

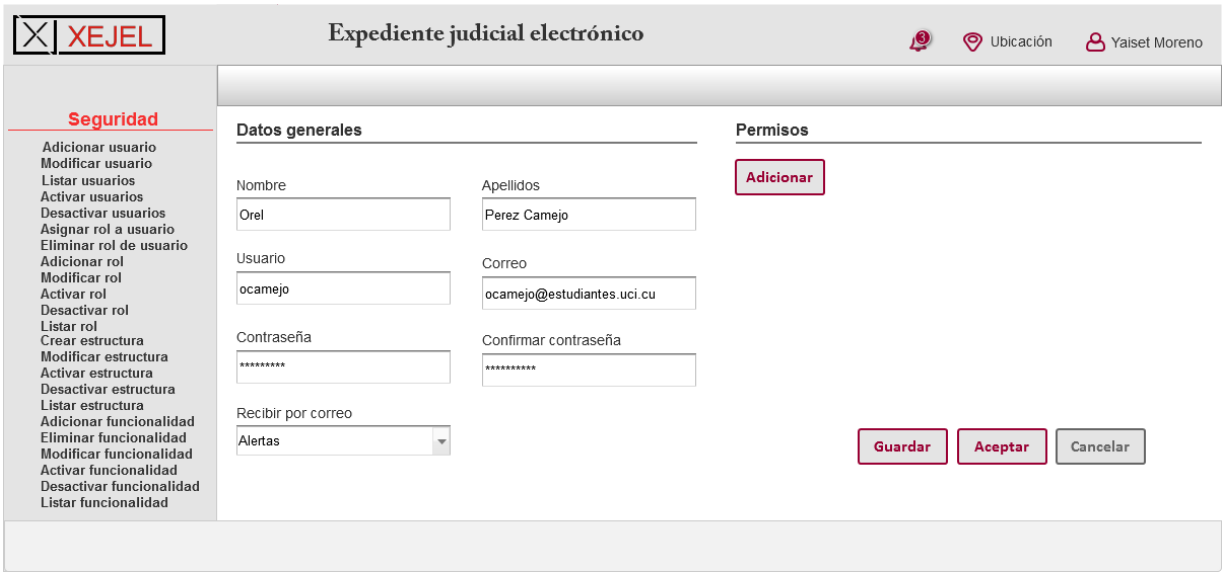
Aceptar: Opción que permite guardar los cambios y salir de la interfaz.

Guardar: Opción que permite guardar los cambios sin salir de la interfaz.

Cancelar: Opción que permite cancelar los cambios realizados y salir de la interfaz.

Observaciones: N/A

Prototipo elemental de interfaz gráfica de usuario:



Fuente: Elaboración propia

2.4 Análisis y diseño

2.5.1.1 Arquitectura base

Como arquitectura base para el desarrollo de XEJEL se ha definido una arquitectura en capas por parte del back-end. Dicha arquitectura brinda entre sus beneficios que se puede entender una capa como un todo, sin considerar a las otras; las capas se pueden sustituir con implementaciones alternativas de los mismos servicios básicos, además, se minimizan dependencias entre capas. En esta arquitectura la capa Controlador es la encargada de recibir las peticiones que provienen del front-end, esta accede a la capa Contenedor de servicios en la que se encuentran contenidas las clases Gestoras y la lógica del negocio de la aplicación. La capa Contenedor de servicios a su vez accede a la capa Modelo la cual interactúa con la capa Acceso a datos a través del patrón Fábrica. Además, esta arquitectura cuenta con tres capas transversales Caché, Trazas y Seguridad, las cuales abarcan todas las anteriormente mencionadas.

Para el front-end se define una arquitectura basada en componentes la cual ofrece como beneficios: la facilidad de instalación, la escalabilidad entre componentes, reduce los costos de desarrollo y mantenimiento, y la reusabilidad. Esta arquitectura está compuesta por módulos, donde cada uno de ellos contiene al menos un componente. Los componentes se relacionan con las plantillas y viceversa, además las directivas interactúan con las plantillas dotando a estas de funcionalidades. Un componente puede hacer uso de otro componente o de un servicio, esto último se logra a través del patrón inyección de dependencia.

A continuación, se muestra en figuras, el comportamiento de estas arquitecturas.

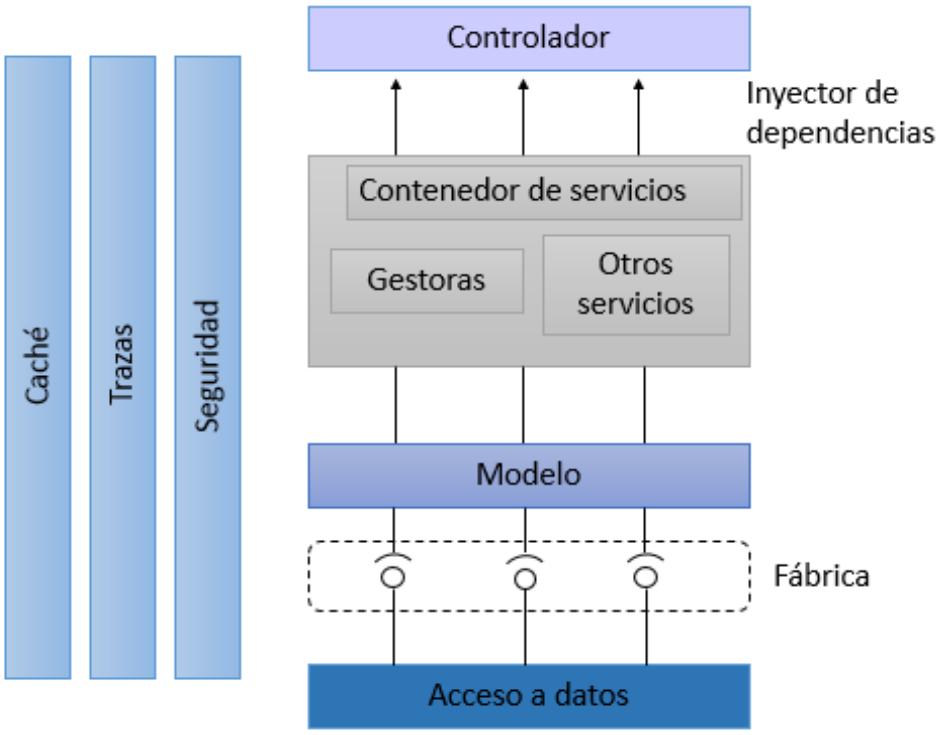


Figura 2. Back-end (Symfony)

Fuente: Elaboración propia

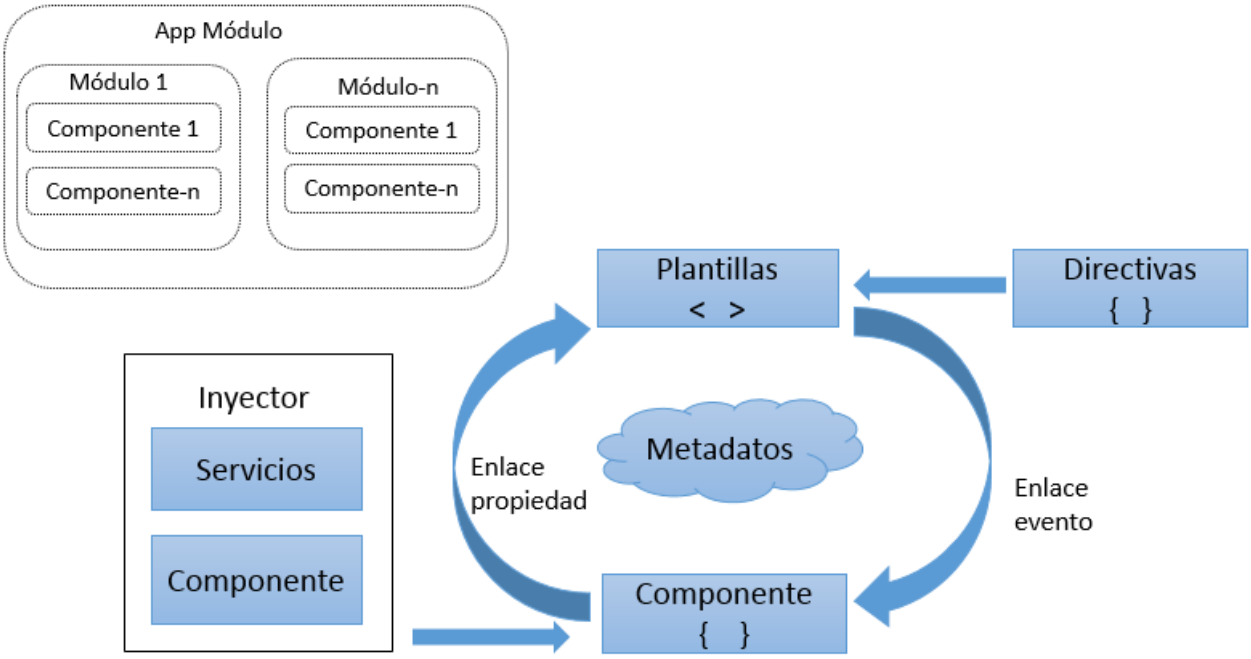


Figura 3 Front-end (Angular)

Fuente: Elaboración propia

2.5.2 Diagrama de componentes

Un diagrama de componentes muestra las organizaciones y las dependencias entre tipos de componentes. Representa las dependencias entre componentes software, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables (59). A continuación, se muestra el diagrama de componentes elaborado para el desarrollo del componente de seguridad de XEJEL. Como se observa en la Figura 4 en el front-end la plantilla interactúa con el controlador, este a su vez manipula el modelo y devuelve una respuesta a la plantilla. Por otra parte, la Figura 5 muestra que en el back-end el Routing redirecciona a la capa controlador, esta a su vez accede a la gestora y finalmente la gestora a la capa de acceso a datos.

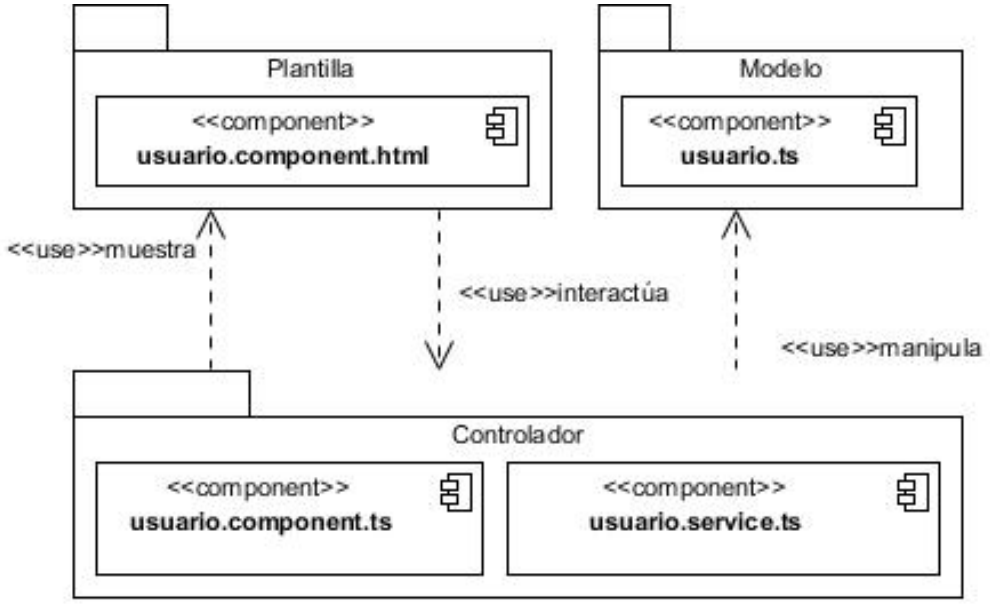


Figura 4 Diagrama de componentes front-end

Fuente: Elaboración propia

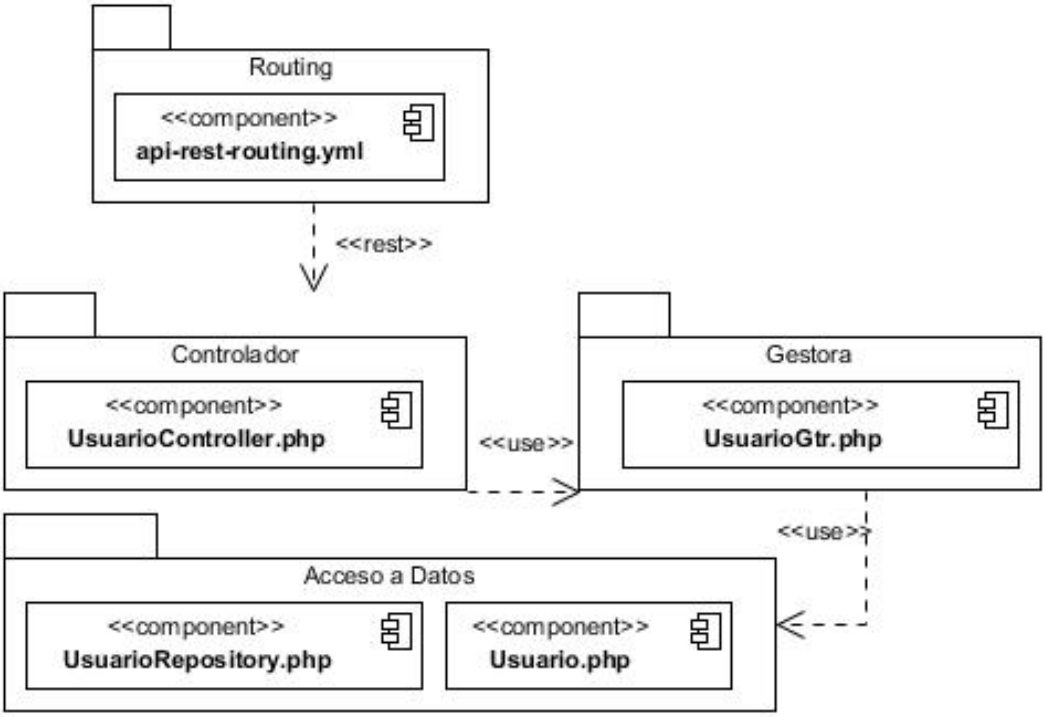


Figura 5 Diagrama de componentes back-end

Fuente: Elaboración propia

2.5.3 Diagrama de clases del diseño

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea una vista lógica de la información que se maneja en el sistema, los componentes que se encargarán del funcionamiento y la relación entre uno y otro. (60) Resumiendo, son interacciones entre nodos y arcos, que

generalmente representan interacciones, relaciones, interfaces y colaboraciones entre las clases, interfaces, notas, restricciones, paquetes y demás elementos que conforman un programa orientado a objetos. A continuación, se muestra el diagrama de clases del diseño elaborado para el RF_ Adicionar usuario.

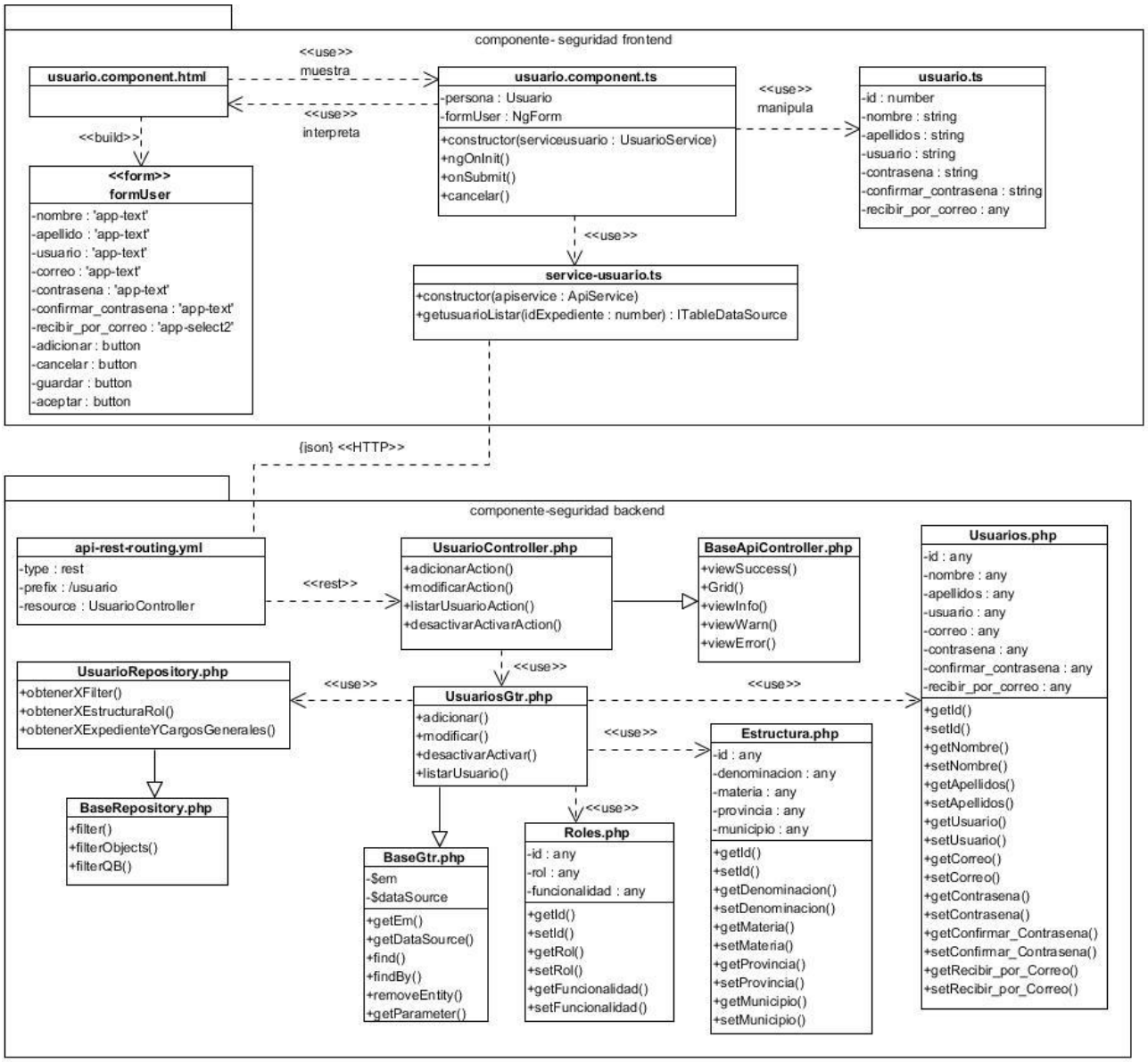


Figura 6 Diagrama de clases del diseño

Fuente: Elaboración propia

2.5.4 Patrones de diseño

Patrones GRASP

Experto: Este patrón se pone en práctica en la solución en las clases entidades que son las expertas en información y las encargadas de manejar la lógica del negocio que comprenden cada uno de los conceptos que engloba la propuesta de solución. La figura mostrada a continuación evidencia el uso de este patrón.

```php
/** @ORM\Entity ...*/
class User extends BaseUser
{
    /** @var int ...*/
    protected $id;

    /** @var string ...*/
    private $nombre;

    /** @var string ...*/
    private $apellidos;

    /** @var bool ...*/
    private $notificarPorCorreo;

    /** @ORM\OneToMany(targetEntity= "AppBundle\Entity\Seguridad\UsuarioRolEstructura", mappedBy="usuario", cascade={"all"}) ...*/
    private $rolEstructura;

    /** @ORM\OneToMany(targetEntity= "AppBundle\Entity\Seguridad\Turnado", mappedBy="usuario", cascade={"all"}) ...*/
    private $turnado;

    /** @ORM\OneToMany(targetEntity= "AppBundle\Entity\Comun\TribunalActuante", mappedBy="usuario", cascade={"all"}) ...*/
    private $miembroTribunalesActuantes;

    /** @ORM\OneToMany(targetEntity= "AppBundle\Entity\Comun\Notificacion", mappedBy="usuario", cascade={"all"}) ...*/
    private $notificaciones;
```

Figura 7 Patrón Experto. Clase User.php

Fuente: Elaboración propia

Controlador: Este patrón se pone de manifiesto en el sistema XEJEL en la utilización de varias clases controladoras como son EstructuraController.php, FuncionalidadController.php, RolController.php, UsuarioController.php. las mismas se encuentran en el directorio xejel-backend/src/AppBundle/Controller y son las encargadas de interactuar con las clases gestoras después de una petición y enviar los datos a la api-rest-routing.yml para que se comunique con el frontend a través de los servicios. Seguidamente se muestra la clase UsuarioController correspondiente al componente de usuario en la propuesta de solución.

```php
class UsuarioController extends BaseApiController
{


    /** @Rest\Post("/") ...*/
    public function crearUsuarioAction(Request $request, UsuarioGtr $usuarioGtr){...}


    /** @Rest\Put("/{id}") ...*/
    public function actualizarUsuarioAction($id, Request $request, UsuarioGtr $usuarioGtr){...}

    /** @Rest\Post("/listadoUsuario") ...*/
    public function listadoUsuarioAction(Request $request, UsuarioGtr $usuarioGtr, UsuarioTm $usuarioTm){...}

    /** @Rest\Patch("/") ...*/
    public function activarUsuarioAction(Request $request, UsuarioGtr $usuarioGtr){...}

    /** @Rest\Get("/get/{id}") ...*/
    public function obtIdAction($id, UsuarioGtr $usuario){...}

    /** @Rest\Get("/listadoEstructUser") ...*/
    public function listEstrcuUserAction(UsuarioGtr $usuarioGtr){...}

}
```

Figura 8 Patrón Controlador. Clase UsuarioController.php

Fuente: Elaboración propia

Creador: En la implementación del XEJEL este patrón se evidencia en las clases Gestoras (ejemplo UsuarioGtr.php) en el momento de crear nuevas instancias de las clases Entidades, como se muestra en la figura a continuación:

```
$user = $userManager->createUser();
$user->setNombre($usuario["nombre"]);
$user->setApellidos($usuario["apellidos"]);
$user->setUsername($usuario["username"]);
$user->setEmail($usuario["email"]);
$user->setPlainPassword($usuario["password"]);
$user->setNotificarPorCorreo($usuario["notificar_por_correo"]);

$userManager->updateUser($user);
```

Figura 9 Patrón Creador. Clase UserGtr.php

Fuente: Elaboración propia

Fabricación: Generalmente se considera que la fabricación es parte de la capa de servicios orientada a objetos de alto nivel en una arquitectura y se utilizan para acceder de la capa Controladora a la Gestora y también para acceder a las clases Repositorio para el acceso a datos, implementadas en XEJEL, ejemplo de esto es cuando la clase UsuarioGtr.php hace uso de la clase UsuarioRepository.php para acceder a los datos.

```
public function listAllUsuarios($array = null)
{
    $filter = array();
    $data = $this->getEm()->getRepository( className: User::class)->obtenerXFilter($array, ResultType::QueryBuilderType);

    return $this->getDataSource()->setData($data);

}
```

Figura 10 Patrón Fabricación. Clase UsuarioGtr.php

Fuente: Elaboración propia

Patrones GoF

Observador: Este patrón es empleado en todos los componentes del frontend, este último hace la petición de un servicio a través de una inyección y se queda a la espera de la respuesta que dicho servicio debe proveer para así emitir o no un evento. Se evidencia en la comunicación del frontend con el backend. En el sistema XEJEL se hizo uso de este patrón y un ejemplo de ello es en la comunicación de la clase UsuarioController.php a través de la api-rest-routing.yml con la clase service-usuarios.ts. La siguiente figura evidencia el uso de este patrón.

```
onSubmit(ir = false) {
  this.usuarioServiceService.save(this.usuario)
    .subscribe( next: response => {
      if (response) {
        this.notificationService.success( message: 'app.module.usuario.msg.guardado');
        if (ir) {
          this.guardar.emit();
        } else {
          this.aceptar.emit();
        }
      }
    });
}
```

Figura 11 Patrón Observador. Clase gestionar-usuario.component.ts

Fuente: Elaboración propia

Otros patrones

Inyección de dependencias: En XEJEL este patrón se utiliza obteniendo una instancia del Contenedor de Servicios a través del método get() a cualquier clase derivada de xejel-backend/src/AppBundle/Controller, también su utilización se evidencia en los constructores como se muestra en la siguiente figura.

```
constructor(private usuarioServiceService: UsuarioServiceService,
            private rolService: RolService,
            private alertService: AlertService,
            private notificationService: NotificationsService,
            private estructuraService: EstructuraService,
            private activatedRoute: ActivatedRoute,
            private route: Router) {
  super();
  this.usuario = new Usuario();
  this.rol = new Rol();
  this.notificarPorCorreo = this.usuarioServiceService.NotificarCorreo();
  this.selecRoles = this.rolService.AsignarRoles();
  this.selecEstructura = this.estructuraService.AsignarEstructura();

}
```

Figura 12 Patrón Inyección de dependencia. Clase gestionar-usuario.component.ts

Fuente: Elaboración propia

2.5.5 Modelo de datos

El modelado de datos es una manera de estructurar y organizar los datos para que se puedan utilizar fácilmente por las bases de datos (61). A continuación se muestra el modelo de datos utilizado para el componente de seguridad en XEJEL.

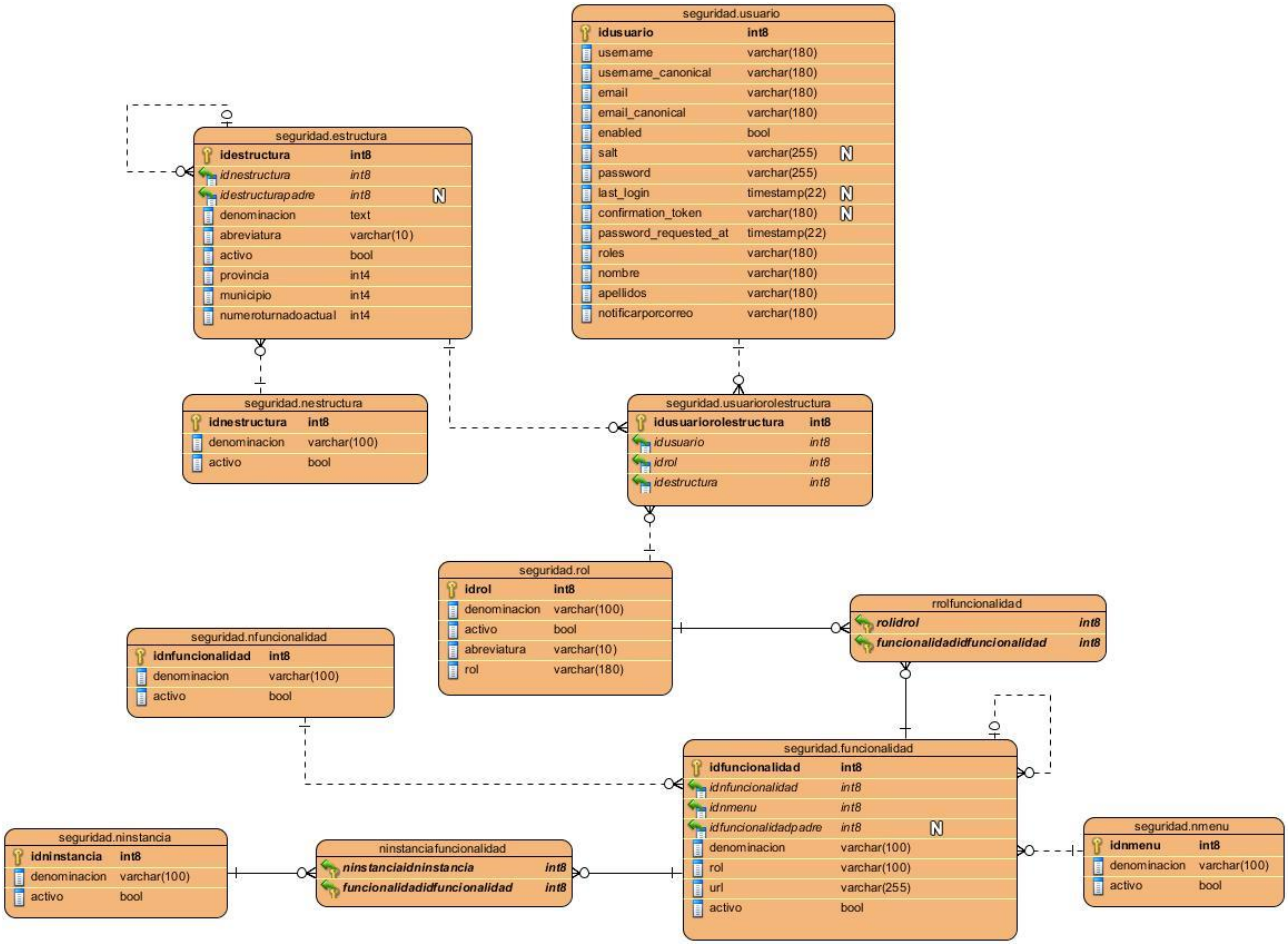


Figura 13 Modelo de datos

Fuente: Elaboración propia

2.5 Implementación

2.5.1 Estándares de codificación

El propósito fundamental de los estándares de codificación es que el proyecto tenga una arquitectura y un estilo consistente, independiente, con lo cual el sistema resulte fácil de entender y por supuesto fácil de mantener (62).

En el desarrollo del Componente se utilizarán los estándares de codificación plasmados en el documento *"CEGEL_XEJEL_Componete_Seguridad_Estandares_de_codificacion_para_PHP"* realizado por el equipo de desarrollo del proyecto en conjuntos con los definidos a continuación.

Identación

La identación suele estar compuesta por 4 espacios. Las tabulaciones no están permitidas.

Cabecera del archivo

Todos los archivos .php tienen que iniciar con una cabecera específica indicando información de la versión, autor y fecha de modificación de los últimos cambios, así como otros datos que se consideren necesarios. Cada equipo decide si desea agregar más información.

```
/*
* @Clase controladora acusado. "UsuarioController.php"
* @modificado: 1 de enero del 2019
* @autor: Orel Pérez Camejo
```

Clases

Las clases serán colocadas en un archivo.php, donde estará el código de la clase. El nombre de la clase será el mismo del archivo. Los nombres de las clases deben de iniciar con letra mayúscula. Si un nombre de la clase se comprende de más de una palabra, la primera letra de cada nueva palabra debe capitalizarse. No se permiten las letras capitalizadas sucesivas; por ejemplo, una clase "User" no se permite, mientras "User" es aceptable.

Estilo y reglas de escritura de código PHP

Nombres de variables

Los nombres de variables deben ser intuitivos y concisos. No utilizar grandes frases ni abreviaturas. Los nombres de variables pueden empezar con una letra en minúscula, pero si estas tienen más de una palabra, cada nueva palabra debe iniciar con letra mayúscula ("symfonyPdf ").

Contantes

Todas las letras pertenecientes al nombre de una constante deben aparecer en mayúsculas y deben de tener como prefijo el nombre de la clase a la que pertenecen. Estas pueden contener tanto caracteres alfanuméricos como barras bajas (_). Los números están permitidos.

Definición de las funciones

Los nombres de funciones pueden contener únicamente caracteres alfanuméricos. Los nombres de la función pueden empezar en letras minúsculas. Cuando un nombre de función consiste en más de una palabra, la primera letra de cada nueva palabra debe estar en mayúsculas. Ejemplo:

```
listarUsuarios () {
        función ();
}
```

Siempre incluir las llaves

En todo momento a la hora de codificar un bloque de instrucciones, éste debe ir encerrado entre llaves, aun cuando conste de una sola línea. Ejemplo:

```
if ($algo) {
    función ();
```

```
}
```

Llaves

Las llaves de apertura irán al final de la sentencia que delimitan y las de cierre estarán alineadas con el inicio de la sentencia en una nueva línea. Ejemplo:

```
foreach ($algo) {
        if ($algo) {
          función ();
    }
}
```

No utilizar variables sin inicializar:

Si no se tiene control sobre el valor de una variable, se debe verificar que esté inicializada. Ejemplo:

```
if (isset($cliente) && $cliente == 5)
```

Pero sólo se debe usar esta opción cuando no se tenga el control o no se esté seguro del valor que pueda tener la variable (Como en variables que llegan por parámetro o por GET).

1.2 Conclusiones parciales

La entrevista con el cliente permitió sentar las bases y el buen entendimiento de las necesidades reales del cliente. Quedó documentada toda la información relativa a la propuesta de solución obtenida a través del levantamiento de requisitos y la elaboración de las historias de usuarios. A partir de la descripción de las arquitecturas y la realización de los artefactos correspondientes, queda bien elaborada la propuesta para la correcta implementación del componente para la gestión de la seguridad en XEJEL teniendo en cuenta el buen uso de los patrones de diseño y los estándares de codificación ya definidos. Una vez realizado el diseño y la implementación han quedado sentadas las bases para realizar la validación del diseño y las pruebas internas necesarias para garantizar el buen funcionamiento de la solución.

Capítulo 3: Validación

Introducción

En este capítulo se valora la calidad de los resultados obtenidos, la cual se lleva a cabo a raíz de la validación del diseño mediante las métricas: Tamaño Operacional de Clases y Relación entre Clases. En último lugar se realizan las pruebas internas y de liberación como disciplinas que propone la metodología y con las cuales se verifica el buen funcionamiento del producto.

3.1 Validación del diseño

3.1.1 Métricas para la validación del diseño

Una métrica de software es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software con medidas que pueden ayudar al desarrollador a juzgar la calidad de un diseño a nivel de componente (63). Su objetivo lo inclinan a mejorar la comprensión de la calidad del producto, a estimar la efectividad del proceso y mejorar la calidad del trabajo.

Una vez realizado el diseño del componente en cuestión, se llevó a cabo el proceso de validación para lo que se hizo necesario estudiar las diferentes métricas, quedando seleccionadas dos de ellas: Tamaño Operacional de Clases y Relaciones entre clases. Las cuales están diseñadas para cumplir con los siguientes atributos de calidad:

- ✓ **Responsabilidad:** Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- ✓ **Complejidad de implementación:** Consiste en el grado de dificultad que tiene implementado un diseño de clases determinado.
- ✓ **Reutilización:** Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- ✓ **Acoplamiento:** Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- ✓ **Complejidad del mantenimiento:** Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- ✓ **Cantidad de pruebas:** Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

Tamaño Operacional de clase (TOC)

La métrica Tamaño Operacional de Clases se basa en contar la cantidad de atributos y la cantidad de operaciones que tiene una clase individual y el promedio que presenta el sistema en su totalidad (64). En la Tabla 4 se muestran los atributos de calidad por los que se evalúa esta métrica, así como el modo en que dichos atributos afectan a la misma.

Tabla 4 Comportamiento de la métrica TOC.

Atributo que afecta	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase

Una vez analizado el resultado obtenido, si el mismo posee valores grandes significa que la clase posee un alto nivel de responsabilidad, a raíz de esto la reutilización disminuirá por lo que la implementación se hará mucho más difícil. Los umbrales definidos para medir los atributos de calidad se muestran en la Tabla 5 donde se emplea la abreviatura Prom como promedio.

Tabla 5 Rango de valores para la evaluación de los atributos de calidad relacionados con la métrica TC

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

La siguiente tabla muestra las clases a las que se le aplicó la métrica y los resultados obtenidos para cada uno de los atributos analizados, en la misma se utilizaron las abreviaturas:

Re: para el atributo de Responsabilidad

Ci: para el atributo de Complejidad de implementación

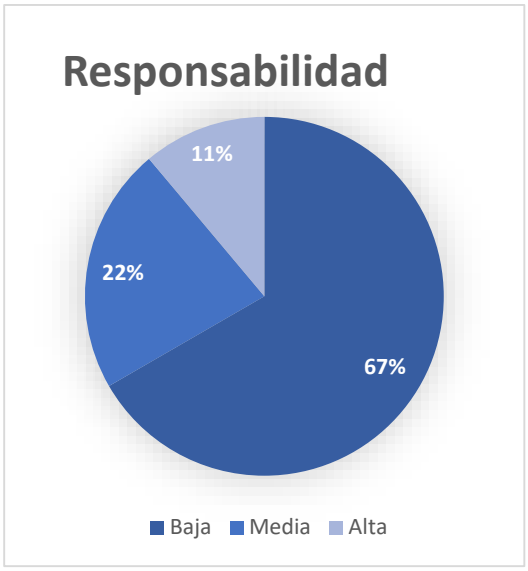
R: para el atributo de Reutilización

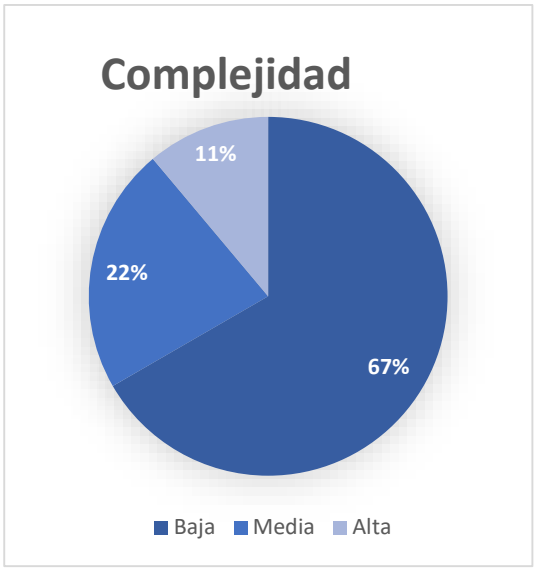
Tabla 6 Resultados obtenidos a partir de la aplicación de la métrica TC.

Componente	Clase	Cantidad de Procedimientos	Re	Ci	R
Usuario	BaseApiController	5	Baja	Baja	Alta
Usuario	UsuarioController	4	Baja	Baja	Alta
Usuario	BaseGtr	6	Baja	Baja	Alta
Usuario	UsuariosGtr	4	Baja	Baja	Alta
Usuario	BaseRepository	3	Baja	Baja	Alta
Usuario	UsuarioRepository	0	Baja	Baja	Alta
Usuario	Estructura	10	Media	Media	Media
Usuario	Roles	6	Baja	Baja	Alta
Usuario	Usuario	16	Alta	Alta	Baja

Fuente: Elaboración propia

Para la realización de este ejemplo se utilizaron 9 clases, obteniéndose un total de 58 procedimientos con un promedio de los mismos igual a 6,44 aproximadamente. Analizando los resultados obtenidos se tiene que un 11% de las clases tienen responsabilidad y complejidad alta, para otro 22% la responsabilidad y la complejidad es media y para el 67% restante se tiene que dichos atributos se comportan de una manera baja. Para el atributo de reutilización se obtuvo como resultado que el 50% de las clases poseen una alta reutilización, el 33% es media, mientras que el 17% presentan baja reutilización. Dichos resultados son positivos para la validación del diseño puesto a que al presentar baja responsabilidad y complejidad acopladas a un elevado nivel de reutilización se permite la realización de una implementación sencilla. Los gráficos de pastel presentados a continuación muestran los resultados obtenidos.





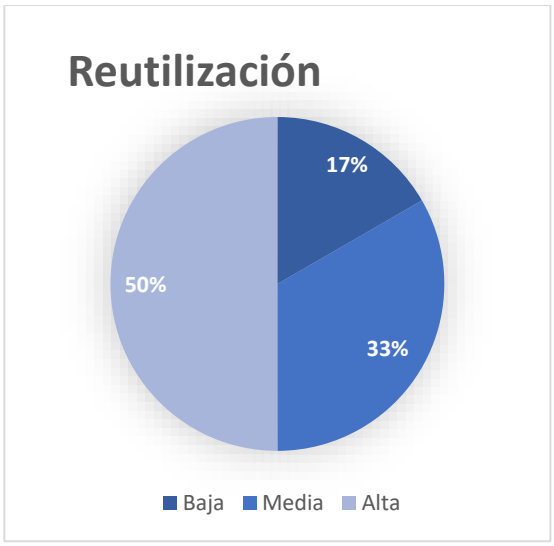


Figura 14 Resultados de la métrica TOC

Fuente: Elaboración propia

Relaciones entre clases (RC)

La métrica Relaciones entre clases está dada por el número de relaciones de uso de una clase con otra y evalúa los atributos mostrados en la Tabla 7 donde también se ve evidenciado el modo en que estos afectan a la métrica (64).

El valor obtenido a raíz de la aplicación de la métrica RC debe ser directamente proporcional al acoplamiento y a la complejidad de mantenimiento e inversamente proporcional al nivel de reutilización. Lo que significa que, si existe un acoplamiento y una complejidad de mantenimiento alto, el nivel de reutilización del código disminuirá.

Tabla 7 Comportamiento de la métrica RC.

Atributo que afecta	Modo en que lo afecta

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

Tabla 8 Rango de valores para la evaluación de los atributos de calidad relacionados

	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de Mantenimiento	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.
Cantidad de Pruebas	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

Fuente: Elaboración propia

La Tabla 8 anteriormente expuesta muestra los rangos de valores en los que se van a evaluar los atributos de la métrica y se utiliza la abreviatura Prom para el promedio. A continuación, se muestran las clases tomadas como ejemplo para la demostración de la misma y los resultados arrojados una vez aplicada. En la siguiente tabla se utilizan las abreviaturas:

Ac: para el atributo de Acoplamiento

Cm: para el atributo de Complejidad de mantenimiento

R: para el atributo de Reutilización

Cp: para el atributo de Cantidad de pruebas

Tabla 9 Resultados obtenidos a partir de la aplicación de la métrica RC.

Componente	Clase	Cantidad de Relaciones de Uso	Ac	Cm	R	Cp
Usuario	BaseApiController	1	Bajo	Baja	Alta	Baja
Usuario	UsuarioController	1	Bajo	Baja	Alta	Baja
Usuario	BaseGtr	1	Bajo	Baja	Alta	Baja
Usuario	UsuariosGtr	5	Alto	Alta	Baja	Alta
Usuario	BaseRepository	1	Bajo	Baja	Alta	Baja
Usuario	UsuarioRepository	2	Medio	Media	Media	Media
Usuario	Estructura	1	Bajo	Baja	Alta	Baja
Usuario	Roles	1	Bajo	Baja	Alta	Baja
Usuario	Usuario	1	Bajo	Baja	Alta	Baja

Fuente: Elaboración propia

Una vez estudiados los resultados obtenidos se entiende que para un 56% de las clases el acoplamiento resulta ser bajo, para un 11% es alto mientras que el restante 33% de las clases no poseen este atributo de calidad. La complejidad de mantenimiento de las clases tiene un comportamiento bajo en su mayoría representando el 89% de las mismas, teniendo el restante 11% de las clases una complejidad alta. Los mismos valores antes mencionados se obtuvieron para el atributo referente a la cantidad de pruebas. Por su parte, el atributo de reutilización resulta tener un nivel alto en un 89% y bajo en un 11%. Los datos anteriores indican que se realizó un buen diseño ya que, debido al bajo acoplamiento unido a la baja complejidad de mantenimiento y la alta reutilización se podrá realizar una implementación fácil y un sencillo mantenimiento, también gracias a los resultados obtenidos para la cantidad de pruebas se requerirá de menos esfuerzo a la hora de realizar las pruebas unitarias.

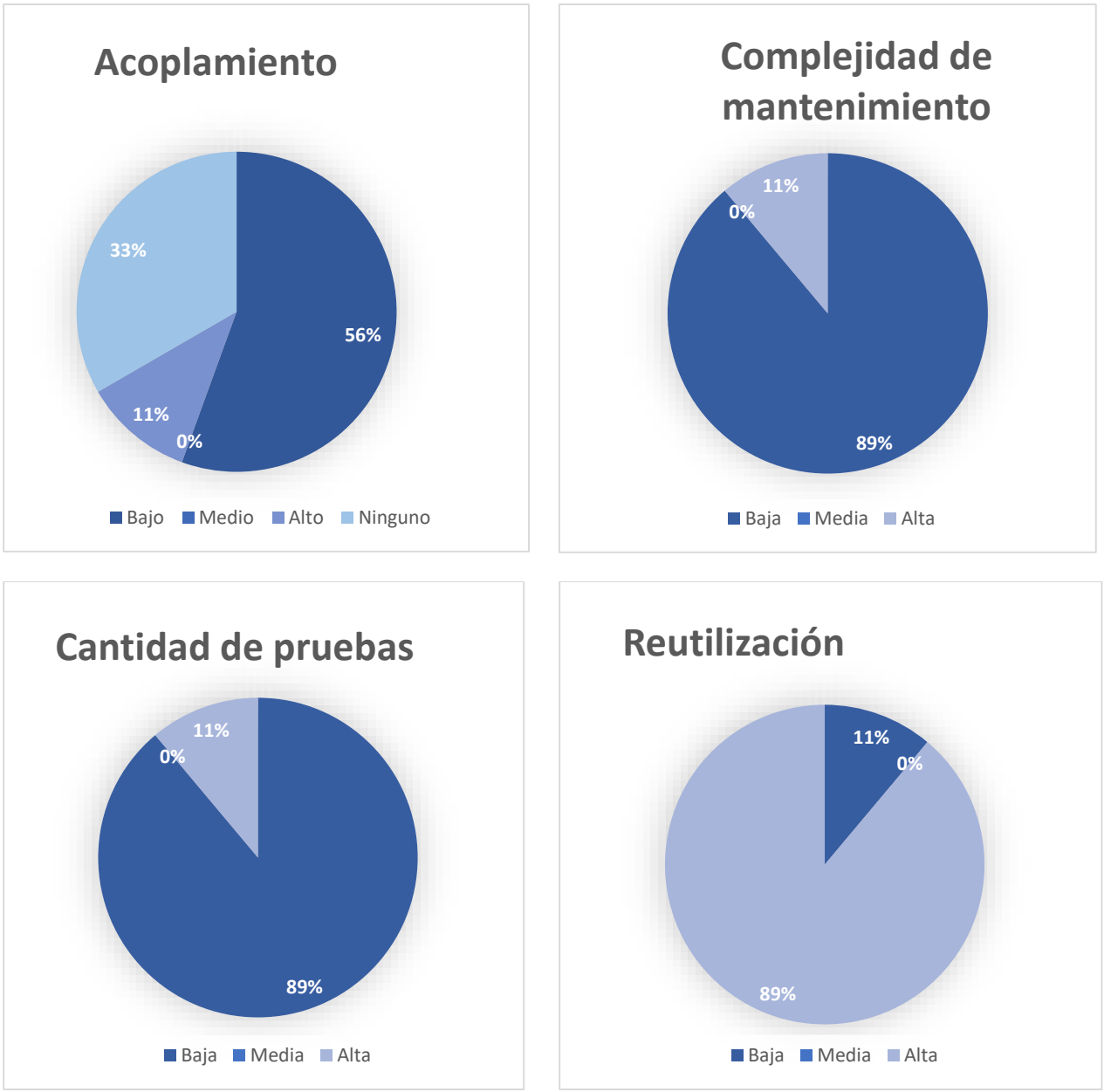


Figura 15 Resultados de la métrica RC

Fuente: Elaboración propia

3.2 Pruebas internas

Una vez analizada y realizada la validación del diseño y siguiendo la guía ofrecida por la metodología seleccionada se hace necesario realizar la misma operación, pero para la implementación mediante pruebas internas. En la disciplina Pruebas internas 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 (58). Seguidamente se detallan las pruebas realizadas durante la aplicación de esta disciplina teniendo como objetivo garantizar la calidad del resultado obtenido.

3.2.1 Pruebas unitarias

Las pruebas unitarias se centran en un componente de software individual, estructuras de datos y funcionalidad del componente, en un esfuerzo por descubrir errores locales del componente (65). Para la aplicación de estas pruebas se ha definido utilizar el método de caja blanca el cual se presenta a continuación.

Método de caja blanca

Las pruebas de caja blanca (también conocidas como pruebas de caja de cristal o pruebas estructurales) se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente. El método de caja blanca garantiza que se ejerciten por lo menos una vez todas las rutas independientes de cada módulo, programa o método, así como todas las decisiones lógicas en las vertientes verdadera y falsa, todos los bucles en sus límites operacionales y las estructuras internas de datos para asegurar su validez (65). Para la aplicación de este método se seleccionó la técnica de ruta básica debido a que permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto básico. A continuación, se muestra el código de la función crearUsuario() seleccionado como ejemplo en este caso para llevar a cabo dicha técnica.

```php
/**
 * @param $usuario
 * @return mixed
 */
function crearUsuario($usuario)
{
    $userManager = UtilRepository2::getContainer()->get( id: 'fos_user.user_manager');// 1
    $usernameExist = $userManager->findUserByUsername(trim($usuario['username']));// 1
    $emailExist = $userManager->findUserByEmail(trim($usuario['email']));// 1

    if ($usernameExist /* 2 */ || $emailExist /* 3 */) {
        return false;// 4
    }
    /**
     * @var User $user
     */
    $user = $userManager->createUser();// 5
    $user->setNombre($usuario["nombre"]);// 5
    $user->setApellidos($usuario["apellidos"]);// 5
    $user->setUsername($usuario["username"]);// 5
    $user->setEmail($usuario["email"]);// 5
    $user->setPlainPassword($usuario["password"]);// 5
    $user->setNotificarPorCorreo($usuario["notificar_por_correo"]);// 5

    $userManager->updateUser($user);// 5
    $this->actualizarPermisos($user, $usuario);


    return true;// 5
}// 6
```

Figura 16 Método crearUsuario()

Fuente: Elaboración propia

Para poder calcular la complejidad ciclomática es necesario realizar a partir del código anterior un grafo de flujo representado mediante nodos, aristas y regiones y que queda de la siguiente manera:

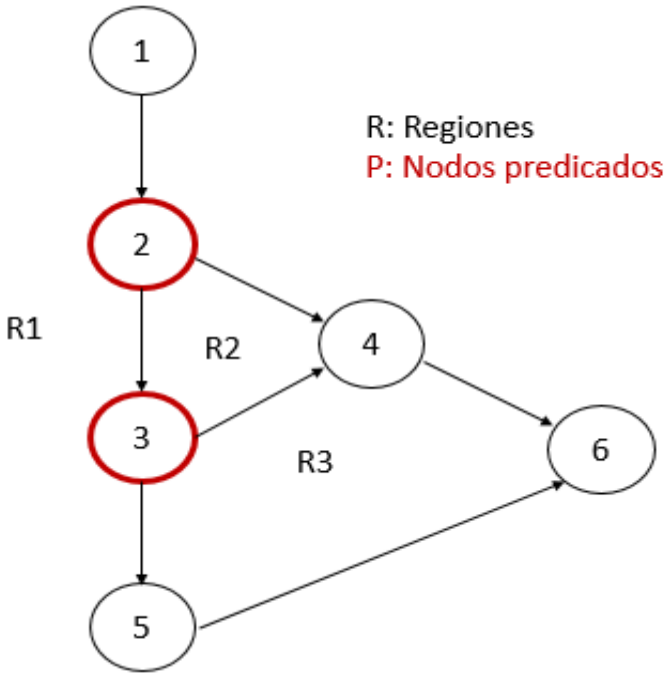


Figura 17 Grafo de flujo correspondiente al método crearUsuario()

Fuente: Elaboración propia

El valor calculado como complejidad ciclomática define el número de rutas independientes del conjunto básico de un programa y ofrece un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Ya realizado el grafo, se procede a el cálculo de la complejidad ciclomática la cual se realiza mediante las siguientes tres variantes:

Variante 1:
V(G)= A-N+2

V(G)= 7-6 +2

V(G)= 3

Siendo A la cantidad de aristas y N la cantidad de nodos.

Variante 2:
V(G)=P+1

V(G)=2+1

V(G)=3

Siendo P, la cantidad de nodos predicados.

Variante 3:
V(G)= R

V(G)=3

Siendo R, el número de regiones.

En cada una de las variantes anteriores la complejidad ciclomática resultó ser igual a 3, lo que significa que existen 3 rutas básicas y que esa misma cantidad es el límite mínimo de casos de prueba que se le pueden aplicar al código.

A continuación, se muestran las rutas básicas obtenidas por las que puede circular el flujo:

Ruta #1: {1,2,4,6}

Ruta #2: {1,2,3,4,6}

Ruta #3: {1,2,3,5,6}

Una vez obtenidos las rutas básicas anteriormente señaladas, se procede a realizar los casos de prueba para cada una de las mismas, seguidamente se presenta un ejemplo aplicado sobre la ruta #1:

Tabla 10 Caso de prueba para la ruta básica #1

Caso de prueba: Ruta básica #1	
Entrada	Nombre de usuario y correo.
Condiciones	El usuario ya existe.
Resultados esperados	El sistema muestra un mensaje indicando que ya el usuario existe.

Fuente: Elaboración propia

Una vez realizados todos los casos de prueba obtenidos a partir de la aplicación de la técnica de ruta básica sobre todos los métodos del sistema, es necesario concluir que el mismo ejercita al menos una vez todas las sentencias, así como todas las decisiones lógicas en las vertientes verdaderas y falsas, además, se determinó que el sistema se encuentra libre de códigos innecesarios y ciclos infinitos.

3.2.2 Pruebas funcionales

Las pruebas funcionales son pruebas diseñadas tomando como referencia las especificaciones funcionales de un componente o sistema. Se realizan para comprobar si el software cumple con las funciones esperadas (66). Se define utilizar para la realización de estas pruebas el método de Caja negra.

Método de caja negra

El método de caja negra se lleva a cabo sobre la interfaz del software. Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la

información externa se mantiene (66). Para la aplicación de este método se utilizó la técnica de partición de equivalencia. A continuación, se muestra la descripción de las variables como primer paso a llevar a cabo en la realización del diseño de casos de pruebas (DCP) presentado en el Anexo 1, en este caso para el RF_Adicionar usuario. Los restantes DCP se encuentran en los artefactos elaborados por los autores de la presente investigación:

- ✓ “CEGEL_XEJEL_Componente_Seguridad_Autenticar_Usuario_DCP”
- ✓ “CEGEL_XEJEL_Componente_Seguridad_Estructura_DCP”
- ✓ “CEGEL_XEJEL_Componente_Seguridad_Funcionalidad_DCP”
- ✓ “CEGEL_XEJEL_Componente_Seguridad_Rol_DCP”
- ✓ “CEGEL_XEJEL_Componente_Seguridad_Usuario_DCP”

Tabla 11 Descripción de las variables.

No	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de texto	No	Campo de texto que permite redactar el nombre del usuario. Admite una cadena de letras, números o cualquier otro carácter hasta un máximo de 80.
2	Apellido	Campo de texto	No	Campo de texto que permite redactar el apellido del usuario. Admite una cadena de letras, números o cualquier otro carácter hasta un máximo de 80.
3	Usuario	Campo de texto	No	Campo de texto que permite darle un nombre al usuario en el sistema. Admite una cadena de letras, números o cualquier otro carácter hasta un máximo de 80.
4	Correo	Campo de texto	No	Campo de texto que permite introducir el correo del usuario en el sistema. Tendrá un formato como el siguiente ejemplo: ocamejo@xxxx.xx
6	Contraseña	Campo de texto	No	Campo de texto que permite introducir la contraseña del usuario. Admite una cadena de letras, números o cualquier otro carácter desde un mínimo de 8 hasta un máximo de 30 caracteres.

7	Confirmar contraseña	Campo de texto	No	Campo de texto que permite introducir la contraseña del usuario. Admite una cadena de letras, números o cualquier otro carácter desde un mínimo de 8 hasta un máximo de 30 caracteres. Debe ser igual a la contraseña anterior.
8	Recibir por correo	Campo de selección	No	Campo de selección que permite seleccionar si se quieren recibir por correo las alertas, notificaciones, ambas o nada.

Fuente: Elaboración propia

Para la aplicación de este método se realizaron un total de 3 iteraciones con el objetivo de garantizar el correcto funcionamiento del componente ante diferentes situaciones y asegurar un resultado satisfactorio. La siguiente Figura 18 muestra el número total de No conformidades (NC) detectadas en cada iteración.

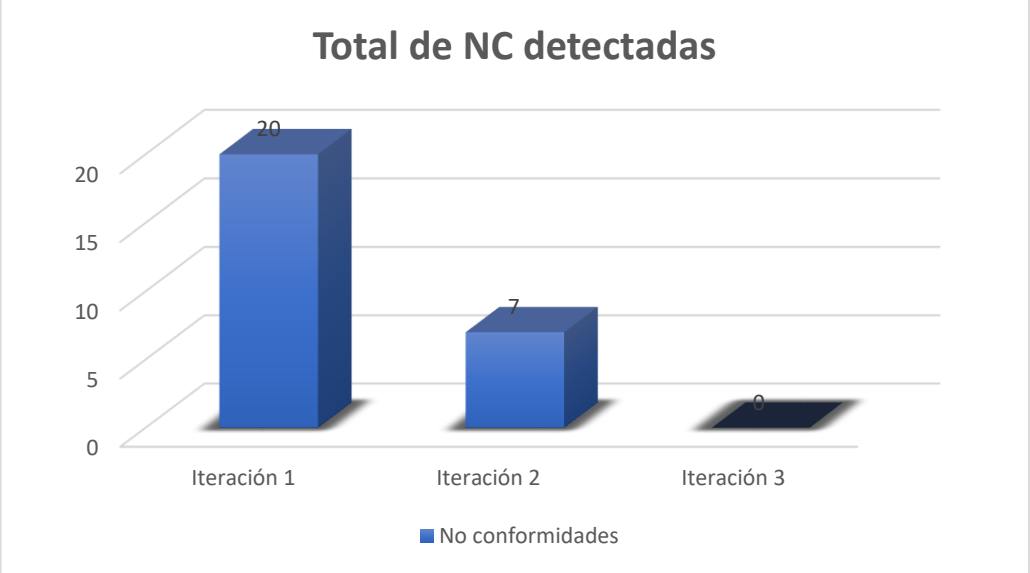


Figura 18 Total de NC detectadas por cada iteración

Fuente: Elaboración propia

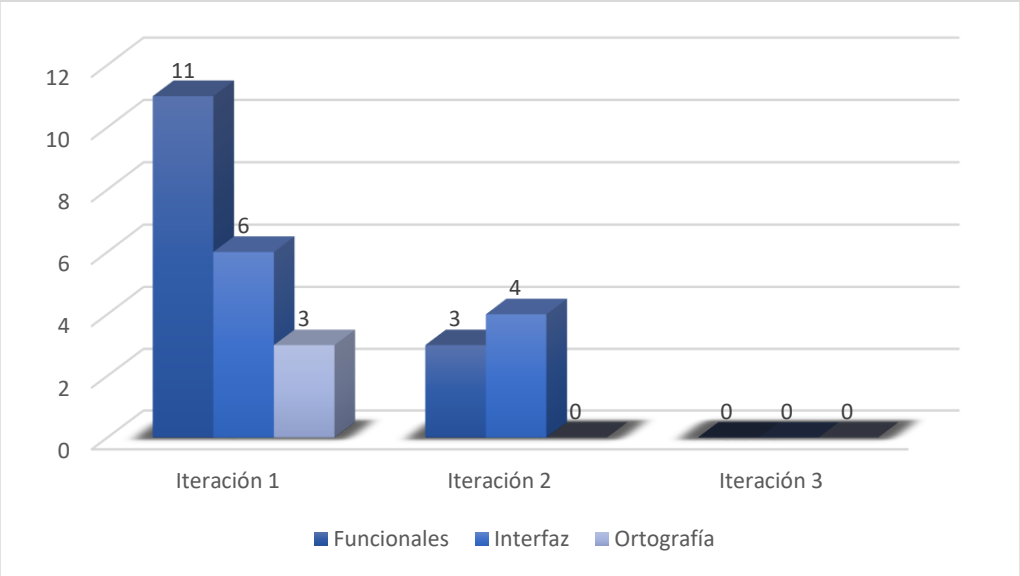


Figura 19 Total de NC por tipo de clasificación en cada iteración

Fuente: Elaboración propia

La Figura 19 muestra las NC detectadas según su clasificación las cuales fueron: Funcionales, Ortografía e Interfaz. Una vez corregidas las NC y producto a que no se detectaron NC en una tercera iteración se determinó proceder hacia la disciplina de pruebas de liberación propuesta por la metodología seleccionada.

3.3 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 (58). El proceso de pruebas de liberación por el cual se sometió el funcionamiento de la propuesta de solución fue llevado a cabo por el grupo de Calidad del centro CEGEL, el cual es el encargado de evaluar el producto antes de ser entregado al cliente. Para dicho proceso se utilizaron los DCP utilizados durante la realización de las pruebas funcionales pertenecientes a la disciplina de pruebas internas. La siguiente gráfica muestra los resultados obtenidos según las iteraciones realizadas.

Figura 20 Total de NC detectadas en las pruebas de liberación

Fuente: Elaboración propia

Durante la disciplina de pruebas de liberación se detectaron en una primera iteración un total de dos NC clasificadas en Recomendación y Correspondencia con otro artefacto. Finalmente, en una segunda iteración no se detectaron NC por lo que se procedió a redactar el acta de liberación por parte del grupo de calidad y la cual se presenta en el Anexo 2. La mencionada acta de liberación corrobora que el producto ya se encontraba listo para ser entregado al cliente.

1.3 Conclusiones parciales

En el presente capítulo se realizó la validación del diseño mediante el uso de métricas permitió obtener un diseño sencillo y de buena calidad con clases reutilizables y con poca dependencia entre sí. Por otra parte, la ejecución de las pruebas unitarias y funcionales para la validación de la implementación permitieron demostrar que las funcionalidades externas e internas satisfacen los requisitos definidos permitiendo la entrada de datos adecuados y produciendo salidas correctas. Finalmente, las pruebas de liberación evidenciaron que los entregables cumplen con las expectativas y necesidades del cliente.

Conclusiones generales

Con la realización del presente trabajo de diploma se desarrolló un componente de seguridad para el actual sistema en desarrollo XEJEL, el cual protege la información que en este se maneja de ataques externos o accesos no autorizados. Los resultados obtenidos permiten concluir:

- ✓ Se establecieron los referentes teóricos en los que se sustenta la propuesta de solución relacionados con el objeto de estudio, además, sobre las diferentes herramientas informáticas para la gestión judicial, así como la metodología, las herramientas y las tecnologías a utilizar durante el desarrollo quedando de esta forma sentadas las bases de la investigación.
- ✓ La identificación de requisitos, así como el análisis y diseño realizado de los componentes usuarios, roles, estructuras y funcionalidades de la propuesta de solución, permitieron obtener una aproximación a la implementación y establecer los elementos necesarios para la misma.
- ✓ La implementación de los componentes permitió contribuir a la gestión de la seguridad en el Expediente Judicial Electrónico garantizando el cumplimiento de la integridad y confidencialidad de la información.
- ✓ Las métricas para la validación del diseño utilizadas y las pruebas internas realizadas permitieron valorar la viabilidad de la propuesta de solución, quedando corroborada en la disciplina de pruebas de liberación la buena calidad de los entregables al cliente.

Recomendaciones

- ✓ Se recomienda la integración de los componentes de la presente investigación al resto de los componentes del Expediente Judicial Electrónico para lograr la gestión adecuada de la seguridad de la información almacenada en el sistema.

Referencias

1. **Enciclopedia jurídica. [En línea] 2014. [Citado el: 2 de noviembre de 2018.] http://www.enciclopedia-juridica.biz14.com/d/expediente/expediente.htm.**

2. Antonio, Franciskovic Rojas. LOS EXPEDIENTES JUDICIALES: EXPERIENCIAS DE ANTAÑO Y HOGAÑO. [En línea] 2013. [Citado el: 2 de noviembre de 2018.] http://www.derecho.usmp.edu.pe/sapere/ediciones/edicion_6/articulos/4_Los_expedientes_judiciales.Experiencias_de_antano_y_hogano.pdf.

3. FUDE. FUDE Cursos a Distancia. [En línea] 2018. [Citado el: 2 de noviembre de 2018.] https://www.educativo.net/articulos/que-es-un-expediente-judicial-796.html.

4. Justicia, Ministerio de. *EXPEDIENTE JUDICIAL ELECTRÓNICO* . España : s.n., 2014.

5. GALINDO, FERNANDO. *E- Judicial en España* . Zaragoza : s.n., 2015.

6. CEJA. Perspectivas de Uso e Impactos de las TIC en la Administración de Justicia en América Latina. [En línea] http://biblioteca.cejamericas.org/bitstream/handle/2015/3955/Libroblancoe-justicia.pdf?sequence=1&isAllowed=y.

7. ASPECTOS GENERALES DE SEGURIDAD INFORMÁTICA. [En línea] 2006. http://repositorio.utc.edu.ec/bitstream/27000/536/1/T-UTC-1052%281%29.pdf.

8. DefiniciónABC. [En línea] 2018. http://www.definicionabc.com/social/seguridad.php.

9. Española, Asociación de Academia de la Lengua. Real Academia Española. *Diccionario de la lengua española.* [En línea] 2018. [Citado el: 26 de marzo de 2019.] https://dle.rae.es/?id=XTrIaQd.

10. Merino, Julián Pérez Porto y María. Definición de. *Definición de Seguridad Informática.* [En línea] 2008. https://definicion.de/seguridad-informatica/.

11. Yudith, Doina. Instituciones. *Direccion Nacional Seguridad y Proteccion MINSAP.* [En línea] 19 de enero de 2019. [Citado el: 26 de marzo de 2019.] https://instituciones.sld.cu/dnspminsap/seguridad-informatica/.

12. HUGO. HUGO TESIS. [En línea] 2008. https://problema.blogcindario.com/2008/10/00014-marco-teorico.html.

13. Tracasa. *AVANTIUS Sistema de Gestión Procesal de Justicia.* [En línea] https://tracasa.es/proyectos/avantius-sistema-de-gestion-procesal-de-justicia/.

14. ingenio2010. [En línea] 2018. [Citado el: 29 de noviembre de 2018.] http://www.ingenio2010.com/index.php/Lexnet.

15. Minerva-noj. [En línea] 2018. [Citado el: 26 de noviembre de 2018.] http://lider1.dia.fi.upm.es:8080/pubby/page/Minerva_noj.

16. Lex-Doctor. [En línea] 2018. [Citado el: 26 de noviembre de 2018.] http://www.lex-doctor.com/lex-doctor-abogados-lex-doctor-10.php.

17. Órgano judicial. *Sistema Automatizado de Gestión Judicial (SAGJ).* [En línea] 2018. [Citado el: 26 de noviembre de 2018.] https://www.organojudicial.gob.pa/servicios/sistema-automatizado-de-gestion-judicial-sagj.

18. Guadarramas, MSc. José R Castro Morell y MSc. Daniel E. González. *Sistema para la tramitación de procesos penales.* Las Villas : Universidad Central "Marta Abreu", 2008.

19. Ochoa, Yosviel Domínguez González y Darian González. *SITPC, UNA HERRAMIENTA INFORMÁTICA CUBANA PARA LA ADMINISTRACIÓN DE LA JUSTICIA.* La Habana : s.n., 2018.

20. Sánchez, Tamara Rodríguez. *Metodología de desarrollo para la Actividad productiva UCI.* 2014.

21. Sistemas Master. *Definición de Herramienta.* [En línea] 2019. [Citado el: 28 de marzo de 2019.] https://sistemas.com/herramienta.php#ixzz2qwSpRx1E.

22. Búa, Manuel Torres. El método de proyectos en tecnología. [En línea] 30 de abril de 2014. [Citado el: 28 de marzo de 2019.] https://www.edu.xunta.es/espazoAbalar/sites/espazoAbalar/files/datos/1464945204/contido/index.html.

23. Leyva, Enrique Santos. *Módulo de Seguridad para el Sistema de Medición Arex.* s.l. : Universidad de las Ciencias Informáticas, 2016.

24. php. *¿Qué es php?* [En línea] [Citado el: 25 de marzo de 2019.] https://www.php.net/manual/es/intro-whatis.php.

25. php. *¿Qué puede hacer php?* [En línea] [Citado el: 25 de marzo de 2019.] http://php.net/manual/es/intro-whatcando.php.

26. Caceres, Martin. DevCode. *¿Qué es TypeScript?* [En línea] [Citado el: 4 de noviembre de 2018.] https://devcode.la/blog/que-es-typescript/.

27. web, Desarrollo. MDN web docs moz://a. *CSS.* [En línea] 18 de marzo de 2019. [Citado el: 4 de noviembre de 2018.] https://developer.mozilla.org/es/docs/Web/CSS.

28. MDN web docs moz://a. *HTML.* [En línea] 18 de marzo de 2019. [Citado el: 11 de abril de 2019.] https://developer.mozilla.org/es/docs/Web/HTML.

29. Fernández, Alberto. Quora. *¿Qué es HTML5 y para qué sirve?* [En línea] 9 de agosto de 2016. [Citado el: 4 de noviembre de 2018.] https://es.quora.com/Qu%C3%A9-es-HTML5-y-para-qu%C3%A9-sirve.

30. Urgelles, Verónica Hernández. *Sistema informático para la realización de auditorías de Seguridad Informática en CEGEL.* s.l. : Universidad de las Ciencias Informáticas, 2016.

31. Maceo, Jorge Enrique Ricardo y Lara, Jorge Fonseca. *Desarrollo del Componente de Seguridad Onyx para el sistema Quarxo fase 2.* s.l. : Universidad de las Ciencias Informáticas, 2014.

32. Visual Paradigm Product Overview. [En línea] [Citado el: 20 de septiembre de 2018.] http://www.visualparadigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html..

33. Guión Visual Paradigm for UML. [En línea] 2013. [Citado el: 20 de septiembre de 2018.] http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf..

34. Pineda, Juan Miguel Vergara. CORIAWEB. [En línea] 2016. [Citado el: 6 de noviembre de 2018.] https://www.coriaweb.hosting/symfony-principales-caracteristicas/.

35. Fontela, Alvaro. Raiola Networks. *¿Que es Bootstrap?* [En línea] 16 de julio de 2015. [Citado el: 4 de noviembre de 2018.] https://raiolanetworks.es/blog/que-es-bootstrap/.

36. Angular One framework.Mobile & desktop. [En línea] [Citado el: 29 de marzo de 2019.] https://angular.io/tutorial.

37. jQuery write less, do more. *What is jQuery?* [En línea] 2019. [Citado el: 7 de noviembre de 2018.] https://jquery.com/.

38. Lázaro, Diego. Introducción a Doctrine ORM. *Explicación y funcionamiento del mapa de objetos relacional Doctrine ORM y sus principales características.* [En línea] 2018. [Citado el: 4 de noviembre de 2018.] https://diego.com.es/introduccion-a-doctrine-orm.

39. Montalvo, Alvaro Mauricio Carrera. *SISTEMA DE GEOLOCALIZACIÓN PARA SELECCIONAR CURSOS FORMATIVOS EN LA CIUDAD DE QUITO.* Quito : FACULTAD DE INGENIERÍA Y CIENCIAS, 2018.

40. pgAdmin . *PostgreSQL Tools.* [En línea] 2016. [Citado el: 13 de junio de 2019.] https://www.pgadmin.org/.

41. Calenda, Maia. GENBETA:dev. *NetBeans.* [En línea] 9 de enero de 2014. [Citado el: 26 de marzo de 2019.] https://www.genbeta.com/desarrollo/netbeans-1.

42. Git. [En línea] [Citado el: 26 de marzo de 2019.] https://git-scm.com/.

43. Git. *What is Git?* [En línea] [Citado el: 26 de marzo de 2019.] https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F.

44. Crespo, Álvaro González. Qué es GitLab. [En línea] 2016. [Citado el: 19 de enero de 2019.] https://es.linkedin.com/learning/gitlab-esencial/que-es-gitlab.

45. Foundation, Apache Software. APACHE HTTP Server Proyect. *What is the Apache HTTP Server Project?* [En línea] [Citado el: 19 de enero de 2019.] http://httpd.apache.org/ABOUT_APACHE.html.

46. Pérez, Alina Dolores Rodríguez y Rojas, Luis Guillermo Silva. Revista Cubana de Informática Médica. *Arquitectura de software para el sistema de visualización médica Vismedic.* [En línea] 8 de abril de 2016. [Citado el: 1 de abril de 2019.] http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1684-18592016000100006.

47. Pelaez, Juan Carlos. [En línea] [Citado el: 19 de enero de 2019.] https://geeks.ms/jkpelaez/2009/05/30/arquitectura-basada-en-capas/.

48. —. [En línea] [Citado el: 20 de enero de 2019.] https://geeks.ms/jkpelaez/2009/04/18/arquitectura-basada-en-componentes/.

49. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Element of Reusable Object-Oirented Software. *Design Patterns.* 1995.

50. C, Oscar Sandoval. EL MUNDO DEL INFORMATICO. [En línea] Mayo de 2007. https://jorgesaavedra.wordpress.com/2007/05/08/patrones-grasp-patrones-de-software-para-la-asignacion-general-de-responsabilidadparte-ii/.

51. Larman, Craig. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. 3ra .* s.l. : Prentice-Hall, 2016.

52. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Desing Patterns.* s.l. : Addison Wesley, 1995.

53. Mariñán, Martín Pérez. Patrones de Diseño(Design Patterns). [En línea] [Citado el: 21 de marzo de 2019.] https://studylib.es/doc/8483023/patrones-de-dise%C3%B1o--design-patterns--.

54. Larman, Craig. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. 3ra.* s.l. : Prentice-Hall, 2016.

55. Pratt, Michael. Praπ. *Praπ.* [En línea] 16 de abril de 2013. [Citado el: 5 de marzo de 2019.] http://www.michael-pratt.com/blog/13/Patrones-de-Diseno-Inyeccion-de-Dependencias/..

56. Sommerville, Ian. *Ingenieria de Software-Somerville 9na.* Naucalpan de Juárez : Addison-Wesley, 2011. México.

57. Quiroga, Juan Pablo. ***Requerimientos funcionales y no funcionales.*** **s.l. : Universidad de los Andes.**

58. Sánchez, Tamara Rodríguez. ***Metodología de desarrollo para la Actividad Productiva de la UCI.*** **Habana : s.n., 2014.**

59. Rumbaugh, James, Jacobson, Ivar y Booch, Grady. ***El lenguaje unificado de modelado.Manual de referencia.*** **Madrid : Addison Wesley, 1998.**

60. Jacobson, Ivar y otros. ***Object-Oriented Software Engineering—A Use Case.*** **1992.**

61. Tecnologías. ***Modelado de datos.*** **[En línea] [Citado el: 26 de febrero de 2019.] https://www.tecnologias-informacion.com/modeladodatos.html.**

62. Muñoz, Isabela. Prezi. ***Estandares de Codificacion.*** **[En línea] 16 de febrero de 2016. [Citado el: 19 de febrero de 2019.] https://prezi.com/ngfpqli1fklt/estandares-de-codificacion/.**

63. Pressman, Roger S. ***Ingenieria.del.Software.-.Roger.Pressman.7th.*** **s.l. : Mc Graw Hill, 2010.**

64. Lorenz, M. y Kidd, J. ***Object-Oriented Software Metrics.*** **s.l. : Prentice-Hall, 1994.**

65. PRESSMAN, ROGER S. ***Ingenieria de Software Un Enfoque Práctico 7th.*** **2010.**

66. Pressman, Roger S. ***Ingeniería del Software, un enfoque práctico. 7ma edición.*** **s.l. : Mc Graw Hill, 2010.**