Universidad de las Ciencias Informáticas

Facultad 3

Componentes para la gestión de los datos generales del expediente y de los intervinientes en la herramienta informática Expediente Judicial Electrónico

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

Autores:

Linda Liz Vera Solano

Maite Diosdado Jacobo

Tutores:

Ing. Yoslenys Roque Hernández

Ing. Daniel Alberto Ojeda Estrada

La Habana, Cuba

Curso: 2018 - 2019

DECLARACIÓN DE AUTORÍA

Declaramos ser autoras 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 _____.

_____ _____

Maite Diosdado Jacobo

(Autora)

Linda Liz Vera Solano

(Autora)

_____ _____

Ing. Yoslenys Roque Hernández

(Tutor)

Ing. Daniel A Ojeda Estrada

(Tutor)

AGRADECIMIENTOS

Agradezco a mis padres por su apoyo incondicional, y por darme tanta fuerza cuando ya no podía más, y por hacerme entender que la suerte no existe, que los logros vienen acompañados de sacrificio.

A mis abuelos, a mis tías y tíos que de una forma u otra siempre me han ayudado incondicionalmente.

A Arlet por toda su ayuda desde que estoy en la universidad.

A mis tutores Yoslenys y Daniel, por ser los mejores tutores, y por su apoyo incondicional en todo momento.

A mi compañera de tesis por su esfuerzo y dedicación en el desarrollo de la investigación.

A los integrantes del proyecto, por su ayuda en los momentos difíciles.

A la profesora Dariela por su apoyo y ayuda brindada, y a todos los profesores que me ayudaron de una forma u otra.

A todos muchas gracias.

Linda Liz

A mi madre por todo su amor, cariño, apoyo incondicional y comprensión; por ser mi guía, mi ejemplo a seguir y la mejor madre del mundo.

A mi hermano Ariel, que el hecho de saber que puedo ser un ejemplo para él me mantiene firme.

A toda mi familia, mis tíos, mi abuela que de una forma u otra siempre me han apoyado.

A mi novio Royler por todo su amor incondicional, comprensión, apoyo y por hacerme muy feliz.

A mis suegros por todo su cariño, apoyo y hacerme parte de la familia.

A mis maravillosos tutores Yoslenys y Daniel por su gran ayuda, apoyo y dedicación para la realización de este trabajo de diploma.

A mi compañera de tesis por su esfuerzo y dedicación en el desarrollo de la investigación. A todos los integrantes del proyecto que en momentos difíciles tendieron su mano, en especial a Reinier, Yaiset, Luis, Diamicelys.

A mis grandes amigas Patry, Laura, Danet, Haraid, Wendy que han estado conmigo en las buenas y en las malas y con las cuales he pasado excelentes e inolvidables momentos. Las quiero mucho.

A mis compañeros de aula, en especial a los que venimos juntos desde el inicio. A todos los educadores que han contribuido con mi formación profesional, en especial a la profe Dariela, por ser una gran guía, por su preocupación constante, por su dedicación en todo momento, gracias por creer en mí.

A todos los que de una forma u otra han contribuido a la realización del presente trabajo.

A todos aquellos que no he mencionado por falta de espacio pero que estuvieron ahí brindando su ayuda.

A todos, muchas gracias.

Maite

DEDICATORIA

A mis padres y a mis abuelos por ser mi inspiración y a toda mi familia por serlo todo para mí.

A mis amigos por estar ahí cuando los necesito.

A todos muchas gracias por ayudarme a llegar a este día.

Linda Liz

A mi madre por ser mi ejemplo a seguir, por todo su amor. Por ser la mejor madre del mundo.

A mi hermano y mi tío por toda su ayuda brindada.

A Royler, por ser el gran amor de mi vida.
A mis grandes amigos por estar ahí cada día incondicionalmente.

Maite

RESUMEN

El Centro de Gobierno Electrónico de la Universidad de las Ciencias Informáticas surge con el objetivo de informatizar procesos en las instituciones gubernamentales del país y con ello la posibilidad de agilizar los canales de información en las distintas organizaciones. En este centro, se desarrolla una solución informática para los Tribunales Populares Cubanos con el objetivo de informatizar el proceso de creación de los expedientes en las diferentes instancias. Actualmente en los tribunales, los datos del expediente y de los intervinientes se registran de forma manual. La información recopilada no tiene un formato definido y en ocasiones es incompleta lo que retrasa la presentación oportuna de los recursos del caso. Por ello, el presente trabajo de diploma se traza como objetivo principal el desarrollo de los componentes para la gestión de los datos generales del expediente y de los intervinientes de forma tal que contribuya a la estandarización y celeridad en la conformación de los expedientes en los Tribunales Populares Cubanos. Para guiar la propuesta de solución se empleó como metodología de desarrollo de software Proceso Unificado Ágil. Además, se utilizaron las herramientas, lenguajes y tecnologías definidas por el equipo de arquitectura del proyecto Expediente Judicial Electrónico. Al mismo tiempo, para garantizar la calidad del sistema se realizaron las validaciones correspondientes y se aplicaron las pruebas pertinentes, obteniéndose como resultado los componentes para la gestión de los datos generales del expediente y de los intervinientes.

Palabras claves: celeridad, componente, estandarización, Expediente Judicial Electrónico, intervinientes.

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Los Tribunales Populares Cubanos (TPC) componen un sistema de órganos estatales, estructurados con independencia funcional de cualquier otro, y sólo subordinados, jerárquicamente, a la Asamblea Nacional del Poder Popular y al Consejo de Estado. Para la correcta administración de justicia los TPC se encuentran estructurados en tres instancias: Tribunal Supremo Popular, Tribunales Provinciales Populares y Tribunales Municipales Populares. En estas instancias se llevan a cabo diferentes procesos distribuidos en cinco materias[1]: Penal, Administrativo, Civil, Laboral y Económico.

Los TPC tienen pasos de avance en el uso de las tecnologías debido a que el país se ha propuesto desarrollar soluciones internas a problemas de gran impacto en la sociedad, es aquí donde juega un papel fundamental la Universidad de las Ciencias Informáticas (UCI), la cual está organizada por centros productivos que tienen fines específicos. Adscrito a la UCI, el Centro de Gobierno Electrónico (CEGEL) surge con el objetivo de informatizar procesos en las instituciones gubernamentales del país y con ello la posibilidad de agilizar los canales de información y ahorrar tiempo y recursos en las acciones que se realizan en estas organizaciones, tales como el procesamiento de la información generada en procesos judiciales.

Los procesos judiciales llevados a cabo en los TPC implican la atención de un volumen excesivo de expedientes. Los mismos constituyen un instrumento público, que resulta de la agregación de las distintas actuaciones, de las partes y del órgano judicial, en forma de legajo[2] (Web, 2017). En cuanto a la estructura del expediente, cuando se da inicio a un trámite judicial, se le asigna una carátula y una contracarátula al finalizar el cuerpo del documento. En la carátula se le consigna el año de iniciación del expediente, la fecha de entrada, el número del propio expediente y otro conjunto de elementos que varía según la materia pero que son identificativos y propios de cada recurso. Es por ello que los expedientes no siguen el mismo formato, ya que se introducen los mismos datos con disímiles denominaciones en las diferentes salas y materias. El expediente puede demorarse en manos del especialista legal encargado del mismo, debido a la carencia de

[1] Conjunto de los asuntos comprometidos en una rama determinada del derecho, las cuales designan ante todo el género del litigio (Ossorio, 2014).
[2] Conjunto de informaciones, documentos o papeles recopilados, referentes a una persona o un asunto (Oxford, 2018).

datos que son necesarios en el asunto. Esto retrasa la presentación oportuna de los recursos del caso y que el presidente de la sala pertinente demore en tener acceso al mismo. Esta situación se ve agudizada al guardar estos expedientes en estantes, haciendo complejas y tardías las búsquedas, supervisión y el almacenamiento de los mismos, pues el cúmulo de información aumenta considerablemente con el paso del tiempo. Además, el deterioro o pérdida al que se encuentran expuestos los expedientes ocasiona dificultad en el control de la información y por lo tanto lentitud en el trabajo.

Otro aspecto importante que se tiene en cuenta para la conformación del expediente judicial es la información necesaria que identifica a los involucrados en los casos. Datos como el nombre y apellidos o la denominación, la forma de comparecer, la dirección particular o el domicilio legal son requeridos para identificar a las personas naturales o jurídicas que son los intervinientes[3] en el asunto. Todo este proceso de gestión al realizarse de forma manual se basa en gran medida en las acciones de las personas, lo que aumenta la posibilidad de errores humanos y que se invierta tiempo, recursos y materiales, dado este último por la gran dependencia del papel. De no existir estas dificultades los datos en los expedientes seguirían un mismo formato para cada sala o materia lo que traería consigo mayor agilidad, permitiendo que sus trabajadores disminuyan cualquier posibilidad de equivocación.

Ante la problemática anteriormente planteada surge el siguiente **problema a resolver:** ¿Cómo gestionar los datos generales e intervinientes durante la conformación de los expedientes en los Tribunales Populares Cubanos, de forma tal que se contribuya a la estandarización y celeridad en el proceso?

Tomando en cuenta el problema antes propuesto se define como **objeto de estudio:** proceso de desarrollo de expedientes judiciales electrónicos.

De esta forma se determina como **objetivo general:** desarrollar los componentes para la gestión de los datos generales e intervinientes durante la conformación de los expedientes en los Tribunales Populares Cubanos, de forma tal que contribuya a la estandarización y celeridad en el proceso.

Para ello se identifica como **campo de acción:** gestión de los datos generales e intervinientes en el desarrollo del Expediente Judicial Electrónico para los Tribunales Populares Cubanos.

[3] Son las personas que de una forma u otra forman parte de los expedientes que se tramitan y son externos al tribunal. Con excepción de las secretarias y jueces (Ossorio, 2014).

Por lo tanto, se plantea la siguiente **idea a defender**: si se desarrollan los componentes para la gestión de los datos generales e intervinientes durante la conformación de los expedientes en los Tribunales Populares Cubanos, se contribuye a la estandarización y celeridad en el proceso.

Se desglosan del objetivo principal los siguientes **objetivos específicos:**

- Identificar los referentes teóricos en los que se sustenta la propuesta de solución sobre el desarrollo de componentes informáticos dirigidos a la gestión de los datos generales e intervinientes en expedientes judiciales electrónicos.

- Realizar la identificación de requisitos, análisis, diseño e implementación de los componentes para la gestión de los datos generales del expediente y de los intervinientes como aproximación a la implementación.

- Implementar los componentes para la gestión de los datos generales del expediente y de los intervinientes para contribuir a la estandarización y celeridad en la conformación de los expedientes en los Tribunales Populares Cubanos.

- Validar el correcto funcionamiento de los componentes aplicando métricas y pruebas de software para garantizar la calidad de los mismos, así como las variables de la investigación.

Para dar solución a las tareas antes propuestas se definen los métodos científicos utilizados en la investigación, clasificados en teóricos y empíricos:

Métodos teóricos:

- **Histórico-lógico:** empleado en el estudio de la trayectoria y evolución del proyecto XEJEL, además de adquirir conocimientos de la lógica objetiva del desarrollo respecto a cómo se realizan las actividades jurídicas en los TPC.

- **Analítico-sintético:** esta es la unión de dos métodos, el analítico, el cual permitió realizar el estudio teórico de la investigación sobre los Expedientes Judiciales Electrónicos en pequeñas partes para facilitar su estudio, y el sintético que basándose en los elementos analizados con el método anterior ayudó a realizar un

resumen de los elementos más importantes acerca del proceso de desarrollo de estos expedientes.

- **Modelación**: se utilizó el modelo analógico[4] para la realización de los diagramas necesarios en el proceso de desarrollo de software, donde se refleja la estructura de las relaciones, haciendo una representación abstracta de la solución, facilitando así el desarrollo de la misma.

Métodos empíricos:

- **Entrevista**: se utilizó en un intercambio verbal con el cliente para obtener la mayor cantidad de información posible, entender el proceso de negocio, comprender la estructura y funcionamiento de la organización, así como las deficiencias existentes que permitieron definir el problema a resolver y establecer el objeto de estudio (Ver anexo 2).

- **Encuesta:** permitió mediante un cuestionario pre-elaborado (Ver anexo 1), obtener información sobre el tiempo real que demoran los especialistas funcionales jurídicos en conformar los expedientes en los Tribunales Populares Cubanos.

El presente trabajo de diploma está estructurado en 3 capítulos de la siguiente forma:

Capítulo 1. Fundamentación teórica: se aborda en detalle todo lo relacionado con la fundamentación teórica, con el objetivo de definir los elementos teóricos para la realización de la investigación. Se especifican los conceptos asociados al dominio del problema. Se define la metodología a seguir y se caracterizan las herramientas y lenguajes que serán utilizados durante el desarrollo de la solución. El análisis de todos estos elementos hace más comprensible el análisis de la presente tesis.

Capítulo 2. Descripción de la solución propuesta: se presenta la descripción de la solución propuesta a través de la identificación de los requisitos funcionales y no funcionales, igualmente se presenta la arquitectura. A partir de los requisitos identificados se modelan los diagramas de clases del diseño, los cuales permiten obtener una visión más

[4] Representación material de un proceso para entender mejor su origen o funcionamiento (Raviolo, et al., 2010).

clara de la implementación de la solución y los diagramas de componentes donde se representa la relación entre los componentes desarrollados. Por último, se tienen en cuenta los estándares de codificación a tener presentes en la implementación de la solución.

Capítulo 3. Evaluación de la solución propuesta: se evalúa el grado de calidad de los resultados obtenidos en el desarrollo de este trabajo. Esta evaluación se lleva a cabo a partir de la validación del diseño a través de las métricas: Tamaño Operacional de las Clases y Relación entre Clases. Se aplican pruebas de software para verificar y revelar la calidad del producto de software antes de ser entregado al cliente. Por último, se realiza la validación de las variables de la investigación para comprobar en qué grado se cumplirá con la estandarización y la celeridad.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se abordan los conceptos de interés asociados al dominio del problema, se realiza un análisis de la metodología de desarrollo de software empleada, así como del lenguaje de modelado y de la herramienta CASE seleccionada para entender y manejar los artefactos durante el desarrollo del software. También se exponen los patrones de diseño, así como el marco de trabajo y la arquitectura que se emplea en el proceso de construcción del sistema. Además, se profundiza en las características de las herramientas de implementación definidas en el XEJEL que permiten una adecuada construcción del software.

1.1 Conceptos asociados al dominio del problema

A continuación, se presentan algunos de los conceptos asociados al dominio del problema para tener una mejor comprensión de los mismos. Entre ellos se encuentran:

Celeridad: se trata de un término que hace referencia a la velocidad, la premura, la rapidez o la prisa (Pérez Porto, y otros, 2019)**.**

Estandarización: es el acto y el resultado de estandarizar, es decir, ajustar a un estándar. Por lo tanto, implica concertar algo para que resulte coincidente o concordante con un modelo, un patrón o una referencia (Pérez Porto, y otros, 2019).

Expediente: un expediente es un documento que contiene un conjunto ordenado de todos los antecedentes sobre una determinada cuestión. Es un instrumento muy útil en el ámbito administrativo, pues al reunirse todas las pruebas y testimonios, se tiene a mano lo necesario para evaluar un caso, y para decidir en consecuencia (Ossorio, 2014).

Expediente Judicial Electrónico: el conjunto de documentos electrónicos correspondientes a un procedimiento judicial, cualquiera que sea el tipo de información que contenga (MJU, 2012).

Intervinientes: son las personas que de una forma u otra forman parte de los expedientes que se tramitan y son externos al tribunal. Con excepción de las secretarias y jueces (Arango, y otros, 2015).

1.2 Sistemas homólogos

Luego de la investigación de los principales conceptos asociados al dominio del problema, se realiza un estudio de los sistemas similares tanto nacionales como internacionales con el objetivo de definir sus características y la viabilidad de su uso en la solución de la problemática existente.

Gestión Documental Electrónica de Argentina

Es un sistema integrado de caratulación, numeración, seguimiento y registración de movimientos de todas las actuaciones y expedientes del Sector Público Nacional. Gestión Documental Electrónica de Argentina (GDE) está integrado por varios módulos:

- Escritorio Único (EU): permite navegar por todos los módulos que integran el sistema.
- Comunicaciones Oficiales (CCOO): permite la generación, registro y archivo de documentos comunicables.
- Generador Electrónico de Documentos Oficiales (GEDO): permite generar, registrar y archivar todos los documentos oficiales electrónicos.
- Expediente Electrónico (EE): permite la caratulación, vinculación de documentos, pases y consultas de expedientes electrónicos.
- Legajo Único Electrónico: guarda la documentación que certifica a las personas que prestan servicios personales a la Administración Pública (Dirección Nacional de Servicios Digitales, 2017).

Expediente Judicial Electrónico en España

- El Expediente Judicial Electrónico (EJE) permite que todos los intervinientes judiciales accedan al mismo expediente y documentación, lo que ahorra tiempo, optimiza recursos y agiliza la tramitación de los procedimientos.
- El proceso de implantación del EJE precisa de un trabajo previo de digitalización de todos los documentos de los expedientes, bajo un sistema seguro y de manera certificada, así como la implantación de un sistema de gestión documental.
- La digitalización de los expedientes también mejora la accesibilidad a la información por parte de los profesionales judiciales autorizados, que podrán acceder a un mismo documento desde distintos canales y al mismo tiempo.

- Favorece la comunicación entre órganos jurisdiccionales (Ministerio de Justicia de España, 2016).

Corte Suprema de Justicia (República de Paraguay)

Está compuesto por varios módulos:

- Para el registro de las resoluciones y actuaciones judiciales en el sistema de gestión electrónica.
- Los expedientes se archivan en carpetas electrónicas, donde se incorporan los documentos que se generen en su tramitación, creando un índice electrónico.
- Notificaciones electrónicas: son generadas en el proceso de la gestión de los expedientes.
- La actualización de los registros está a cargo del despacho judicial u oficina administrativa correspondiente. La carpeta electrónica y sus registros deben ser respaldados en forma periódica conforme un plan de mantenimiento que ofrezca redundancia y replicación externa (Corte Suprema de Justicia, 2017).

Lex 100: Sistema de Gestión de Expedientes Judiciales (Madrid, España)

Está compuesto por varios módulos:

- Gestor de documentos y escritos: se encarga de la definición de los modelos de escritos y la operatoria asociada a cada uno de ellos, como por ejemplo las acciones que provocan sobre los expedientes. Igualmente se encarga de todos los mecanismos de fusión entre modelos de escritos y datos del expediente.
- Gestor documental: se encarga de la indexación y almacenamiento en forma documental de toda la información de los expedientes, tanto la incluida en documentos y escritos como en los datos de la causa.
- Gestor de seguridad de acceso y auditoría: se encarga de la seguridad de la aplicación. Esta seguridad se subdivide en tres apartados funcionales:

  a) Control de acceso en función de los premisos y roles.

  b) Mecanismo de auditoria permanente.

  c) Infraestructura de gestión de las contingencias (BASE 100, S.A., 2019).

Sistema para la tramitación de Procesos Penales

Software que facilita la tramitación de los procesos penales en los tribunales del país. Dos de las características fundamentales del sistema son: la 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 limitaciones tales como:

- No obtiene datos de la fase judicial del proceso.
- El nivel de informatización que supone el sistema es mínimo y en el caso de las apelaciones es prácticamente nulo.
- Está programado en Delphi utilizando como sistema de gestión de bases de datos a SQL Server 2000 por lo que no es compatible con el software libre (Gonzáles, 2008).

Sistema Informático para Procedimientos Económicos

Sistema informático concebido para el área de la estadística en el procedimiento Económico. Cuenta con funcionalidades básicas y es lento en cuanto a tiempo de respuesta. En él se insertan los documentos radicados manualmente. La secretaria de estadística recoge el libro de radicación de escritos (LRE) e inserta los datos en la aplicación y cada cinco años borra la información, quedando solamente asentada en los libros. Esta aplicación no cumplió las expectativas iniciales de su creación. Prácticamente no informatiza ningún proceso, pues la radicación se realiza de forma manual y las salvas se guardan en disquetes, tecnología que ha quedado obsoleta en la actualidad. La información almacenada es eliminada cada 5 años, algo ineficiente para un sistema judicial para el cual mantener registros es primordial, ya que la información es el mecanismo principal para la justicia, por lo que tampoco es posible integrar la solución al Sistema de informatización para la Gestión de los Tribunales Populares Cubanos (SITPC) (O.M.J, 2009).

Sistema de informatización para la Gestión de los Tribunales Populares Cubanos

El SITPC permite el intercambio de información entre las diferentes instancias de los TPC de manera que los flujos de trabajo para el control de los procesos, los reportes estadísticos y el control jurisdiccional sean realizados con celeridad.
Características más importantes:

- Captación de los datos en el lugar de origen lo que ayuda a evitar errores en la trascripción de documentos.
- Alerta del vencimiento de los términos guía y alerta a los usuarios sobre el vencimiento de los términos de un expediente evitando el incumplimiento en términos de los actos procesales.
- Control de la consecutividad del proceso para no alterar la tramitación de un expediente.
- Comunicaciones electrónicas.
- Generación dinámica de documentos dentro del sistema, lo cual contribuye a la estandarización de los mismos.
- Turnado automático de los procesos al juez ponente, de acuerdo a criterios configurables.
- Radicación automática, permitiendo la generación de un único número de expediente a nivel de país.
- Enumeración automática de escritos presentados, asuntos radicados y resoluciones dictadas.
- Localización y recuperación rápida de los asuntos por cualquiera de sus datos principales, con las búsquedas simples y avanzadas (González Ochoa, y otros, 2018).

Conclusiones del estudio de los sistemas homólogos

Luego del análisis realizado a los sistemas existentes en el mundo y en Cuba, se propone tener en cuenta algunas de las características de los mismos debido al contenido que aportan. En el caso de los sistemas internacionales ninguno resuelve el problema actual ya que son herramientas que necesitan conexión a internet para acceder a cada una de sus funcionalidades y Cuba tiene limitado su uso. Además, las herramientas son privativas y no se pueden aprovechar sus funcionalidades, ya que no responden a las necesidades propias de los Tribunales Populares Cubanos, por lo que no garantizan la soberanía tecnológica a la que aspira el país. Por otra parte, los sistemas que hasta ahora se han desarrollado en el país están construidos con tecnologías propietarias y que no brindan las funcionalidades que son necesarias para la creación de los expedientes de forma automatizada en las distintas instancias.

Por su parte, el SITPC, está compuesto por 46 módulos, de los cuales se encuentran implementados 7, por lo que el resto aún llevan a cabo el proceso de forma manual. Este sistema no logró estabilizarse por la complejidad y las condiciones de infraestructura tecnológica que presentan los TPC acarreando los problemas que dieron origen a la presente investigación.

1.3 Metodología de desarrollo de software

En el desarrollo de software la necesidad de organizar o estructurar de forma correcta y disciplinada, es uno de los factores más importantes para evitar pérdidas de tiempo y recursos. Con el fin de prevenir tales errores es preciso definir una estrategia que ordene las posibles tareas a desarrollar, así como llevar a cabo una guía de cómo efectuar las actividades, procedimientos y pasos que se deben de seguir para el desarrollo de un software (Cano, 2003).

En el caso de la presente investigación, el proceso es guiado aplicando la metodología definida por la UCI para organizar su proceso de desarrollo de software, la cual es una variación del Proceso Unificado Ágil (AUP). El uso de esta metodología se fundamenta, teniendo en cuenta que los componentes propuestos forman parte de uno de los proyectos de la red de centros productivos de la universidad.

Proceso Unificado Ágil (AUP) variación para la UCI

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo y recursos) 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 de la UCI. Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas, teniendo en cuenta el Modelo CMMI-DEV v1.3. El mismo constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad (Sánchez, 2015).

Con la adaptación de AUP que se propone para la actividad productiva de la UCI:

- Se logra estandarizar el proceso de desarrollo de software, dando cumplimiento además a las buenas prácticas que define CMMI-DEV v1.3.

- Se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos.
- Se redujo a 1 la cantidad de metodologías que se usaban y de más de 20 roles en total que se definían se redujeron a 11 (Sánchez, 2015).

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 7 disciplinas también, pero a un nivel más atómico que el definido en AUP: Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas Internas, Pruebas de Liberación y Pruebas de Aceptación (Sánchez, 2015).

De igual forma se definen por parte de la metodología 4 escenarios para la disciplina de Requisitos. Para el caso de la presente investigación se selecciona el escenario 4. El mismo aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información.

Figura 1: escenario No. 4 para encapsular los requisitos de software

Fuente: (Sánchez, 2015).

1.4 Herramienta CASE *(Computer Aided Software Engineering)*

Constituyen aplicaciones o programas informáticos destinados a aumentar el balance en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas sirven de ayuda en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el diseño de proyectos, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras (Ambler, 2018).

Visual Paradigm v8.0

Constituye una herramienta CASE profesional, que utiliza UML *(Unified Modeling Language)* cómo lenguaje de modelado y que soporta el ciclo de vida completo del desarrollo de software, además de tener la ventaja de ser multiplataforma (Gaines, y otros, 2017). La UCI tiene licencia para su uso y cumple con las políticas de migración a software libre en Cuba. A continuación, se listan algunas características de la misma:

- Ofrece amplias características de modelado de casos de uso incluyendo la función completa de UML, diagrama de casos de uso y editor de flujo de eventos.
- Permite diseñar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. El mismo proporciona abundantes tutoriales del Lenguaje de Modelado, demostraciones interactivas de UML y proyectos UML.
- Se integra con herramientas Java, como son: Eclipse/IBM, NetBeansIDE, entre otras.
- Es apoyado por un conjunto de lenguajes tanto en la generación del código como en la Ingeniería Inversa (Gaines, y otros, 2017).

A partir de los elementos antes expuestos se decide por parte del equipo de arquitectura del proyecto utilizar Visual Paradigm en su versión 8.0, teniendo en cuenta que esta herramienta propicia un conjunto de funcionalidades para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Además, se tuvo en cuenta que la UCI posee una licencia académica para su uso.

Lenguaje de modelado UML (*Unified Modeling Language)* v2.0

UML se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software, no define un proceso de desarrollo específico, tan solo se trata de una notación. Se utiliza para detallar los artefactos en el sistema y definir un sistema de software (Pressman, 2010).

El análisis de este lenguaje se realiza debido a que UML es la notación utilizada por la herramienta CASE que se emplea para la creación de los componentes. Además, es el lenguaje estándar de modelado para software que emplea la metodología AUP, la cual rige

el proceso de desarrollo de software del XEJEL. Los autores de la presente investigación utilizan esta herramienta y lenguaje, teniendo en cuenta que los componentes forman parte del desarrollo del XEJEL.

1.5 Marco de trabajo

Es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar (Acosta, 2017).

Symfony v3.4

Es un marco de trabajo completo diseñado para optimizar el desarrollo de las aplicaciones web basado en el patrón Modelo Vista Controlador. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación (Pacheco, 2013).

Para el desarrollo de la propuesta de solución se definió por parte del equipo de arquitectura del proyecto XEJEL la utilización de Symfony versión 3.4 debido a que es fácil de instalar y configurar en cualquier plataforma. De igual forma las aplicaciones desarrolladas con Symfony son compatibles con la mayoría de las plataformas, bibliotecas e infraestructuras que existen y se adaptan a entornos de negocio en cambio permanente, requiriendo menos esfuerzo para su mantenimiento. Los autores de la presente investigación utilizan este marco de trabajo, teniendo en cuenta las características antes descritas y que los componentes forman parte del desarrollo del XEJEL.

Angular v6

Es un marco de trabajo de JavaScript de código abierto que se utiliza para crear y mantener aplicaciones web de una sola página. Para su funcionamiento, la biblioteca *(rxjs-compat)* lee el HTML *(Hypertext Markup Language)* que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y ofrece una utilidad para aliviar esa transición y poco a poco introducir los cambios que el código necesite.

Este marco de trabajo adapta y amplía el HTML tradicional para servir mejor contenido dinámico. Los principales objetivos de diseño que se siguen con su uso son los siguientes:

- Separar el lado del cliente, de una aplicación del lado del servidor. Esto permite que el trabajo de desarrollo avance en paralelo, y permite la reutilización de ambos lados.
- Guiar a los desarrolladores a través de todo el proceso del desarrollo de una aplicación: desde el diseño de la interfaz de usuario, a través de la escritura de la lógica del negocio, hasta las pruebas.
- Usar la inyección de dependencias, para llevar servicios tradicionales del lado del servidor, tales como controladores dependientes de la vista, a las aplicaciones web del lado del cliente. En consecuencia, gran parte de la carga en el *back-end* (capa de acceso a datos) se reduce, lo que conlleva a aplicaciones web mucho más ligeras (Wilson, 2018).

Para el desarrollo del XEJEL se decide, por parte del equipo de arquitectura, utilizar Angular en su versión 6.0 y por tanto es también usada para el desarrollo de la solución propuesta, teniendo en cuenta que esta forma parte del desarrollo del XEJEL.

Boostrap v4.1

Es un marco de trabajo o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS *(Cascading Style Sheets),* así como extensiones de JavaScript adicionales. A diferencia de muchos marcos de trabajo, solo se ocupa del desarrollo *front-end*[5] .

Boostrap es de código abierto y está disponible en *GitHub*[6]. Además, es modular y consiste esencialmente en una serie de hojas de estilo LESS[7] que implementan la variedad de componentes de la herramienta. Los desarrolladores pueden adaptar el mismo archivo de Boostrap, seleccionando los componentes que deseen usar en su proyecto (Boostrap Team, 2018).

[5] Front-end : capa de presentación (ALEGSA, 2018).
[6] GitHub: plataforma de desarrollo colaborativo (Robles, 2018).
[7] Lenguaje de hojas de estilo (ALEGSA, 2018).

Para el desarrollo del XEJEL se decide la utilización de Bootstrap en su versión 4.1 debido a las nuevas funcionalidades que el mismo integra, por ejemplo: proporciona una galería de iconos de forma gratuita, lo que facilita incorporar cada uno de estos elementos en el XEJEL, brinda la posibilidad de asignarle ancho y altura a los elementos de la página web de forma automática por lo que se adapta a todos los dispositivos. Además, se hace un mejor uso de las transiciones, animaciones y efectos en las páginas web. Los autores de la presente investigación utilizan Boostrap, teniendo en cuenta las características antes descritas y que los componentes forman parte del desarrollo del XEJEL.

1.6 Lenguajes de programación

Constituyen lenguajes artificiales que pueden ser usados para controlar el comportamiento de una máquina, especialmente una computadora. Estos se componen de un conjunto de reglas sintácticas y semánticas que permiten expresar instrucciones que luego serán interpretadas (Domínguez, 2018).

PHP v7.2

PHP (acrónimo de *Hypertext Preprocessor)* es un lenguaje del lado del servidor (esto significa que PHP funciona en un servidor remoto que procesa la página web antes de que sea abierta por el navegador del usuario) especialmente creado para el desarrollo de páginas web dinámicas. Este puede ser incluido con facilidad dentro del código HTML, y permite una serie de funcionalidades extraordinarias.

Algunas de las principales características de este lenguaje se manifiestan a continuación:

- Posee gran número de funciones predefinidas.
- Es un lenguaje de secuencia de comandos de servidor, diseñado específicamente para la web. Dentro de una página web puede incluir código PHP que se ejecute cada vez que se visite una página. El código PHP es interpretado en el servidor web y genera código HTML y otro contenido que el visitante verá.
- Dispone de una conexión propia a todos los sistemas de base de datos (The PHP Group, 2019).

Para el desarrollo del XEJEL se decide la utilización de PHP en su versión 7.2, por parte del equipo de arquitectura del proyecto al que pertenece la propuesta de solución. Los autores de la presente investigación utilizan este lenguaje de programación, teniendo en

cuenta las características antes descritas y que los componentes forman parte del desarrollo del XEJEL.

JavaScript v1.8

Es un lenguaje de programación interpretado, orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (*client-side*), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor (*Server-side JavaScript o SSJS).* Su uso en aplicaciones externas a la web, por ejemplo, en documentos PDF (*Portable Document Format)* (Pérez Valdés, 2007).

jQuery v3.3

JQuery constituye una librería de JavaScript, rápida y concisa que facilita la navegación de un documento HTML, manipulación de eventos, animación e interacción Ajax[8] para desarrollos web rápidos. Lo que la hace tan especial es su sencillez y su reducido tamaño. Entrega una serie de utilidades para programar de forma limpia y libre de errores. Manipula estilos CSS y usa las licencias MIT[9] y GPL[10] permitiendo su uso en proyectos autónomos y privativos (The jQuery Foundation, 2019).

Para el desarrollo del XEJEL se decide la utilización de JavaScript en su versión 1.8, ya que es la seleccionada por el equipo de arquitectura del proyecto al que pertenece la propuesta de solución.

HTML 5

Por sus siglas en inglés de (*HyperText Markup Language)* es el lenguaje básico de la web para crear documentos y aplicaciones para que los desarrolladores lo utilicen en cualquier lugar. HTML en su versión 5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Es un lenguaje simple que sirve para definir otros lenguajes que tienen que ver con el formato de los documentos (Bos, 2019).

[8] JavaScript asíncrono y XML, es una técnica de desarrollo web para crear aplicaciones interactivas (The jQuery Foundation, 2019).
[9] Massachusetts Institute of Technology.
[10] Licencia Pública General de GNU software libre.

Para el desarrollo del XEJEL se decide la utilización de HTML en su versión 5, ya que es la versión seleccionada por el equipo de arquitectura del proyecto al que pertenece la propuesta de solución.

CSS 3

Es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Se usa principalmente para establecer el diseño visual de los documentos web. También permite aplicar estilos no visuales, como las hojas de estilo auditivas.
Está diseñado principalmente para marcar la separación del contenido del documento y la forma de presentación de este. Con esto se pretende mejorar la accesibilidad del documento, permitir que varios documentos HTML compartan un mismo estilo usando una sola hoja de estilos separada en un archivo .css, y reducir la complejidad y la repetición de código en la estructura del documento (Bos, 2019).

Para el desarrollo del XEJEL se decide la utilización de CSS en su versión 3, ya que es la versión seleccionada por el equipo de arquitectura del proyecto al que pertenece la propuesta de solución.

Typescript v3.2

Es un lenguaje de programación libre y de código abierto desarrollado por Microsoft. Principalmente se caracteriza por implementar las ventajas de un lenguaje orientado a objetos y las ventajas de JavaScript. Este lenguaje de programación está especialmente indicado para desarrollar proyectos de una cierta envergadura, teniendo en cuenta la integración con distintos editores y entornos de desarrollo (Microsoft, 2018).

Para el desarrollo del XEJEL se decide utilizar TypeScript en su versión 3.2, ya que es el requerido para la versión de Angular seleccionada por el equipo de arquitectura del proyecto al que pertenece la propuesta de solución.

1.7 Sistema Gestor de Bases de Datos

Es un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan. Son programas que admiten crear y conservar las bases de datos asegurando su integridad, confidencialidad y seguridad (Alvarez, 2012).

Un Sistema Gestor de Bases de Datos (SGBD) permite definir, crear y mantener las bases de datos, además permite la autonomía entre los datos y los programas de aplicación, minimizar las redundancias, garantiza la integridad, la seguridad y protección de los datos (Alvarez, 2012).

PostgreSQL v10.1

Es uno de los sistemas gestores de bases de datos de código abierto más avanzado del mundo. Además, constituye un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en Postgres. Dentro de sus principales ventajas se encuentran:

- Soporte de protocolo de comunicación encriptado por SSL[11].
- Extensiones para alta disponibilidad, nuevos tipos de índices, datos espaciales, minería de datos.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos (The PostgreSQL Global Development Group, 2019).

Para el desarrollo del XEJEL se decide la utilización de PostgreSQL en su versión 10.1, ya que es la seleccionada por el equipo de arquitectura del proyecto al que pertenece la propuesta de solución como resultado de la presente investigación.

PgAdmin3 v1.22

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 diversas necesidades, desde la escritura de simples consultas SQL (*Structured Query Language)* a la elaboración de bases de datos complejas. La interfaz gráfica es compatible con todas las características de postgreSQL (Page, y otros, 2018). Se utiliza PgAdmin en su versión 1.22 por su compatibilidad con la versión de PostgreSQL seleccionada por el equipo de arquitectura del proyecto XEJEL al cual pertenece la actual propuesta de solución.

[11] Seguridad de la capa de transporte (ALEGSA, 2018).

1.8 Mapeador Objeto-Relacional (ORM)

Permite convertir los datos de los objetos en un formato correcto para poder guardar la información en una base de datos (mapeo) creándose una base de datos virtual donde los datos que se encuentran en la aplicación, quedarán vinculados a la base de datos. Este consiste en transformar toda la información que se recibe de la base datos, principalmente en tablas, en los objetos de la aplicación y viceversa (Zoega, 2010).

Doctrine v2.4

Es un mapeador de objetos-relacional (ORM por sus siglas en idioma inglés) escrito en PHP que proporciona una capa de persistencia para objetos PHP. Es una capa de abstracción que se sitúa justo encima de un SGBD. Dentro de sus principales características se encuentran:

- Bajo nivel de configuración que necesita para empezar un proyecto. Puede generar clases a partir de una base de datos existente y después el programador puede especificar relaciones y añadir funcionalidad extra a las clases autogeneradas.
- Posibilidad de escribir consultas de base de datos utilizando un dialecto de SQL denominado DQL *(Doctrine Query Language)* que está inspirado en *Hibernate* (Java)[12] .
- Una función compilar que combina varios archivos PHP del marco de trabajo en uno solo para evitar el descenso de rendimiento que provoca incluir varios archivos PHP (Doctrine Team, 2019).

Para el desarrollo del XEJEL se decide la utilización de Doctrine en su versión 2.4, ya que es la seleccionada por el equipo de arquitectura del proyecto al que pertenece la presente propuesta de solución.

1.9 Servidor web

Sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red al navegador de un usuario. Este intercambio es mediado por el navegador y el servidor que se comunican el uno con el otro mediante HTTP[13].Se pueden utilizar varias tecnologías en

12 Es una herramienta de mapeo objeto-relacional (ORM) para la plataforma Java (Doctrine Team, 2019).

13 Hypertext Transfer Protocol.

el servidor para aumentar su potencia más allá de su capacidad de entregar páginas HTML (Netcraft, 2018).

Apache v2.4

Apache, es un servidor web que tiene como principales características las siguientes:

- Multiplataforma.
- Es una tecnología gratuita de código abierto.
- Es un servidor altamente configurable de diseño modular.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor (The Apache Software Foundation, 2019).

Para el desarrollo del XEJEL se selecciona Apache en su versión 2.4, ya que presenta mejoras en cuanto al rendimiento al minimizar el consumo de memoria y en las concurrencias de las peticiones. Además, es el que define el equipo de arquitectura del proyecto XEJEL al cual responde la presente solución.

1.10 Sistema para el control de versiones

Los sistemas para el control de versiones registran los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo de tal manera que sea posible recuperar versiones especificas más adelante. Un sistema de control de versiones (SCV), es un software que controla y organiza las distintas revisiones que se realizan (Durante Lerate, y otros, 2009).

Git v2.17

Es un software de control de versiones diseñado por Linus Torvalds. Fue creado teniendo en cuenta la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida. Algunas de las características más importantes son:

- Rapidez en la gestión de ramas: un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente.
- Gestión distribuida: los cambios se importan como ramas adicionales y pueden ser fusionados de la misma manera como se hace en la rama local.

- Gestión eficiente de proyectos grandes y realmacenamiento periódico en paquetes (Robles, 2018).

Para el desarrollo del XEJEL se decide utilizar Git en su versión 2.17, ya que es la seleccionada por el equipo de arquitectura del proyecto al que pertenece la propuesta de solución.

GitLab v11.5

Es otra plataforma donde se puede alojar el código fuente de forma pública o privada, y mediante el controlador de versiones Git, además, se puede trabajar a nivel colaborativo en diferentes proyectos con distintos desarrolladores. Constituye una potente solución de software libre que permite crear y gestionar repositorios de código y documentos. Con GitLab se puede crear un servicio en un servidor o utilizando los servicios que ofrece para gestionar con facilidad cambios y versiones de los ficheros (Hoffman, 2019).

Para el desarrollo del XEJEL se decide utilizar GitLab en su versión 11.5, ya que es la seleccionada por el equipo de arquitectura del proyecto al que pertenece la presente propuesta de solución.

1.11 Entorno de desarrollo integrado

Es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software. Normalmente, un Entorno de desarrollo integrado (IDE) consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de estos tienen auto-completado inteligente de código (*IntelliSense*). Algunos IDE contienen un compilador, un intérprete, o ambos, tales como NetBeans y Eclipse (Ramos Salavert, y otros, 2011).

NetBeans v8.2

Es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. Permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para

interactuar con las API de NetBeans. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole otros nuevos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software (Domínguez, 2018).

Se decide la utilización de NetBeans en su versión 8.2, ya proporciona soporte para el cliente de control de versiones seleccionado por parte del equipo de arquitectura del proyecto XEJEL al cual pertenece la presente solución.

1.12 Conclusiones del capítulo

El análisis de los principales conceptos asociados al dominio del problema facilitó la mejor comprensión del presente trabajo. Además, el establecimiento del estado del arte permitió obtener las características de los sistemas actuales tanto nacionales como internacionales que utilizan expedientes judiciales, lo que ratificó la necesidad de crear una solución que resuelva las necesidades específicas de los TPC. Por otra parte, la selección de la metodología AUP en su variación para la UCI pone en línea la presente solución con las políticas que sigue la Universidad en el uso de la misma para obtener productos de mayor calidad que satisfagan las necesidades del cliente. También el empleo de la herramienta CASE Visual Paradigm 8.0 ayuda a automatizar el proceso de desarrollo del software y los marcos de trabajo Angular 6 y Symfony 3.4 facilitan la construcción de los componentes propuestos. La selección de las herramientas y lenguajes de la presente solución están en correspondencia con las definidas por el equipo de arquitectura del proyecto XEJEL lo que garantiza la soberanía tecnológica a la que aspira el país.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

En el presente capítulo se presenta la descripción de la propuesta de solución. Se identifican los requisitos funcionales y no funcionales, obteniéndose la base de la arquitectura del sistema, así como particularidades del diseño e implementación. A partir de los requisitos identificados se modelan los diagramas de clases del diseño los cuales permiten obtener una visión más clara de la implementación del sistema. Además, se obtiene el diagrama de componentes que representa la relación y las dependencias entre los mismos.

2.1 Descripción de la propuesta de solución

El sistema XEJEL tiene como objetivo informatizar el proceso de creación de los expedientes en las diferentes instancias de los TPC. Al conformar los casos se hace necesario el registro de los datos generales que son distintivos e identificativos en las diferentes salas, materias e instancias de los TPC, así como la modificación y consulta sobre ellos. Además, sobre los intervinientes, debe especificarse si es una persona natural o jurídica para poder registrar o modificar la información que debe quedar plasmada en el expediente.

La propuesta de solución incluye un componente para la gestión de los datos de los expedientes para cada materia y procedimiento. En este componente quedan registrados los datos propios del asunto, siendo de vital importancia, la fecha de entrada del expediente, el número del mismo y el estado en el que se encuentra. Además, podrán ser detallados los datos de un determinado expediente que se desee a través de la búsqueda automática del mismo. También, se gestionan en el componente de intervinientes, los datos que identifican a las partes que interceden en el asunto. Asimismo, se muestra el listado de personas naturales y/o jurídicas que forman parte de ese expediente.

2.2 Requisitos del software

Un requisito es la condición o capacidad que un usuario necesita para poder resolver un problema o lograr un objetivo. También se define como la condición o capacidad que debe exhibir o poseer un sistema para satisfacer un contrato, estándar, especificación, u otra documentación formalmente impuesta (Sanchez, 2018).

2.2.1 Requisitos funcionales

Los requisitos funcionales (RF) de un sistema, son aquellos que describen cualquier actividad que este deba realizar, en otras palabras, el comportamiento o función particular de un sistema cuando se cumplen ciertas condiciones. Por lo general, estos deben incluir funciones desempeñadas por pantallas específicas, descripciones de los flujos de trabajo a ser desempeñados por el sistema y otros requerimientos de negocio (Pressman, 2010). El equipo de desarrollo de la presente investigación, identificó 22 (RF), en la Tabla 1 se muestra la especificación de estos requisitos:

Tabla 1: especificación de RF de los componentes gestionar datos generales de los expedientes y los intervinientes.

No.	Nombre	Descripción
RF/1	Registrar expediente Penal	El sistema permitirá registrar los datos de un expediente Penal.
RF/2	Modificar expediente Penal	El sistema permitirá modificar los datos de un expediente Penal seleccionado.
RF/3	Registrar expediente Económico	El sistema permitirá registrar los datos de un expediente Económico.
RF/4	Modificar expediente Económico	El sistema permitirá modificar los datos de un expediente Económico seleccionado.
RF/5	Registrar expediente Administrativo	El sistema permitirá registrar los datos de un expediente Administrativo.
RF/6	Modificar expediente Administrativo	El sistema permitirá modificar los datos de un expediente Administrativo seleccionado.
RF/7	Registrar expediente Civil	El sistema permitirá registrar los datos de un expediente Civil.
RF/8	Modificar expediente Civil	El sistema permitirá modificar los datos de un expediente Civil seleccionado.
RF/9	Registrar expediente Laboral	El sistema permitirá registrar los datos de un expediente Laboral.

RF/10	Modificar expediente laboral	El sistema permitirá modificar los datos de un expediente Laboral seleccionado.
RF/11	Buscar expediente	El sistema permitirá buscar los expedientes a los que tenga permisos el usuario autenticado dependiendo de los criterios de búsqueda que sean seleccionados.
RF/12	Consultar datos del expediente	El sistema permitirá consultar los datos identificativos de un expediente determinado.
RF/13	Registrar interviniente persona natural	El sistema permitirá registrar los datos correspondientes a una persona natural como interviniente en un expediente.
RF/14	Modificar interviniente persona natural	El sistema permitirá modificar los datos correspondientes a una persona natural como interviniente en un expediente.
RF/15	Registrar interviniente persona jurídica	El sistema permitirá registrar los datos correspondientes a una persona jurídica como interviniente en un expediente.
RF/16	Modificar interviniente persona jurídica	El sistema permitirá modificar los datos correspondientes a una persona jurídica como interviniente en un expediente.
RF/17	Eliminar interviniente	El sistema permitirá eliminar intervinientes en un expediente determinado.
RF/18	Listar intervinientes	El sistema permitirá listar los intervinientes asociados a un expediente determinado.
RF/19	Buscar intervinientes	El sistema permitirá buscar los datos de un interviniente en un expediente determinado.
RF/20	Adicionar datos de la deuda	El sistema permitirá adicionar los datos de una deuda o varias deudas en los expedientes de la materia Económica.
RF/21	Modificar datos de la deuda	El sistema permitirá modificar los datos de la deuda seleccionada en los expedientes de la materia Económica.

RF/22	Eliminar deuda	El sistema permitirá eliminar la deuda seleccionada en los expedientes de la materia Económica.

Fuente: elaboración propia

2.2.2 Requisitos no funcionales

Los requisitos no funcionales representan características generales y restricciones de la aplicación o sistema que se esté desarrollando. Suelen presentar dificultades en su definición dado que su conformidad o no conformidad podría ser sujeto de libre interpretación, por lo cual es recomendable acompañar su definición con criterios de aceptación que se puedan medir (Pressman, 2010).

El equipo de arquitectura del proyecto XEJEL definió los requisitos no funcionales del sistema en el documento "*CEGEL_XEJEL_Especificacion_de_requisitos_de_software".* Seguidamente se presenta un resumen de los mismos teniendo en cuenta que forman parte de la solución del presente trabajo de diploma.

Usabilidad

RnF 1. Requisito de usabilidad 1

En el sistema se deben visualizar todos los mensajes en idioma español. La tipografía debe ser uniforme, de un tamaño adecuado.

RnF 2. Requisito de usabilidad 2

El sistema debe facilitar la interacción con el usuario, posibilitando la utilización de combinaciones de teclas, podrán utilizarse los campos de selección en la interfaz en los casos que sea posible y se agruparán los vínculos y botones por grupos funcionales siempre que se cumpla con las pautas de diseño de las interfaces.

RnF 3. Requisito de usabilidad 3

El sistema debe ofrecer una interfaz amigable, fácil de operar. Igualmente tiene que mantener la línea de diseño establecida la cual mantiene la uniformidad y representatividad de la solución. Las interfaces deben poseer un diseño sencillo, con pocas entradas, permitiendo un balance adecuado entre funcionalidad y simplicidad de tal manera que no se haga difícil para los usuarios la utilización del mismo.

Confiabilidad

RnF 4. Requisito de confiabilidad 1

El sistema debe permitir la visualización de la información necesaria para advertir cualquier excepción en el mismo. La información detallada es almacenada en registros de los diferentes componentes de la aplicación.

RnF 5. Requisito de confiabilidad 2

El sistema debe estar disponible durante el horario laboral. En caso de ser necesaria una actualización al software, esta se hará mayormente en horario no laboral, en caso contrario, la misma no debe durar más de 24 horas, teniendo en cuenta que previamente se deben haber validado y probado las modificaciones realizadas.

Eficiencia

RnF 6. Requisito de eficiencia 1

Los procesos del sistema que se implementan con transacciones donde se modifica la base de datos, deben tener tiempos de respuesta no mayores a los 3 segundos. En el caso de la obtención de información a través de transacciones que involucran consultas a la base de datos, el tiempo está dado por el volumen de la misma, por lo cual se implementará en todo momento buscando simplificar al máximo, los datos que se consulten, de manera que la cifra no exceda los 10 segundos.

Restricciones del diseño y la implementación

RnF 7. Requisito de restricción de diseño e implementación 1

La PC cliente debe poseer resolución de 1024 por 768 pixeles o superior.

RnF 8. Requisito de restricción de diseño e implementación 2

El sistema se debe desarrollar utilizando el lenguaje de programación del lado del servidor: PHP 7.2+. Lenguaje de programación del lado del cliente: HTML5, CSS3 y JavaScript, TypeScript. Debe estar instalado en el servidor Alternative PHP Cache, así como todas las bibliotecas y configuraciones de las que depende el funcionamiento correcto de Symfony3 y PostgreSQL 10+.

Interfaz de usuario

RnF 9. Requisito de interfaz de usuario 1

En el sistema se debe evitar el uso de letras en negrita, estás se usarán solo en los encabezados de las tablas y el título de los grupos de las funcionalidades que aparecen en el menú lateral.

RnF 10. Requisito de interfaz de usuario 2

Son usados los siguientes *widgets*[14] en los formularios: *checkbox* cuando se tiene menos de 4 elementos posible a seleccionar y se pueden seleccionar varios, en caso de que se deba seleccionar solo uno, se usará *radio button*; *select* para seleccionar elementos de una lista en las que existan más de 3 posibles elementos; *textfield* para los campos de texto y *textarea* para textos de mayor longitud. Todos los campos tendrán un *label*, el mismo se encontrará encima del *widget* y contendrán (*) de color rojo en caso de ser obligatorio el campo, los mensajes de validaciones irán de color rojo y se mostrarán debajo de *widget* y de igual forma, pero de un color más opaco se mostrará algún mensaje de ayuda del campo.

RnF 11. Requisito de interfaz de usuario 3

Las listas se mostrarán en tablas, las mismas se paginarán de 10 elementos por páginas siempre del lado del servidor, las acciones en lotes y demás acciones generales sobre los elementos de la misma, irán ubicadas al lado superior izquierdo de la tabla, las acciones individuales sobre los elementos se ubicarán en cada una de las filas, siempre al final, en la última columna, en forma de botones.

RnF 12. Requisito de interfaz de usuario 4

Los botones tendrán dos tonalidades de colores, los de acciones primarias, que son aquellas acciones que terminan o completan una acción, lo que siempre se quiere lograr, serían los que llamen la atención a la vista y los secundarios, que son las acciones que se pueden hacer, pero no es lo que debe de ocurrir normalmente, ejemplo un cancelar, irán pintados de un color más claro.

RnF 13. Requisito de interfaz de usuario 5

[14] Es un elemento de una interfaz que muestra información con la cual el usuario puede interactuar (ALEGSA, 2018).

Los formularios diseñan el estilo de *wizard*[15], los mismos tendrán los botones de navegabilidad en la esquina inferior derecha, dando la posibilidad de navegar en ambos sentidos y pudiendo cancelar en todo momento.

2.3 Historias de usuario

Constituyen descripciones cortas y esquemáticas que resumen la necesidad concreta de un usuario al utilizar un producto o servicio, así como la solución que la satisface. Su función principal es identificar problemas percibidos, proponer soluciones y estimar el esfuerzo que requieren implementar las ideas propuestas (Solvingadhoc, 2017). En la presente solución se definieron 22 historias de usuario de la cuales se presentan, por su alta complejidad, las correspondientes al RF1 y al RF13:

Tabla 2: HU Registrar expediente Penal correspondiente al RF1

Número: 1	**Requisito:** Registrar expediente Penal
Programador: Linda Liz Vera Solano	**Iteración Asignada:** 1
Prioridad: alta	**Tiempo Estimado:** 24 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 22 horas
Descripción: funcionalidad que permite registrar los datos del expediente que se presenta para la materia Penal. **Campos:** **Materia:** campo de selección donde se debe escoger la materia Penal para que se muestren los datos correspondientes a esta materia. Es obligatorio. **Procedimiento:** campo de selección que se activa después de seleccionar la materia y lista los procedimientos que corresponden a la materia seleccionada. Es obligatorio. **No. EFP/denuncia:** campo de texto para registrar el número del EFP o denuncia del expediente. Es obligatorio. **Órgano de Instrucción/PNR:** campo de texto donde se registra el Órgano de Instrucción o PNR de donde se presenta el expediente. Es obligatorio. **Folios útiles:** campo numérico donde se registra la cantidad de folios que tiene el expediente. Es obligatorio.	

[15] Asistente (ALEGSA, 2018).

Tipo de Recurso: campo de selección donde se escoge el tipo de recurso. No obligatorio.

Delitos/Incidentes: campo de selección múltiple que permite escoger los delitos o incidentes por los que se presenta el expediente. Es obligatorio.

Circular: campo de selección que permite escoger la circular asociada a un delito o incidente. No obligatorio.

Fecha de entrada: campo de selección, se escoge de un calendario la fecha de entrada del expediente. Es obligatorio.

Estado actual: campo de selección para escoger un estado en el que se encuentra el expediente. Campo obligatorio.

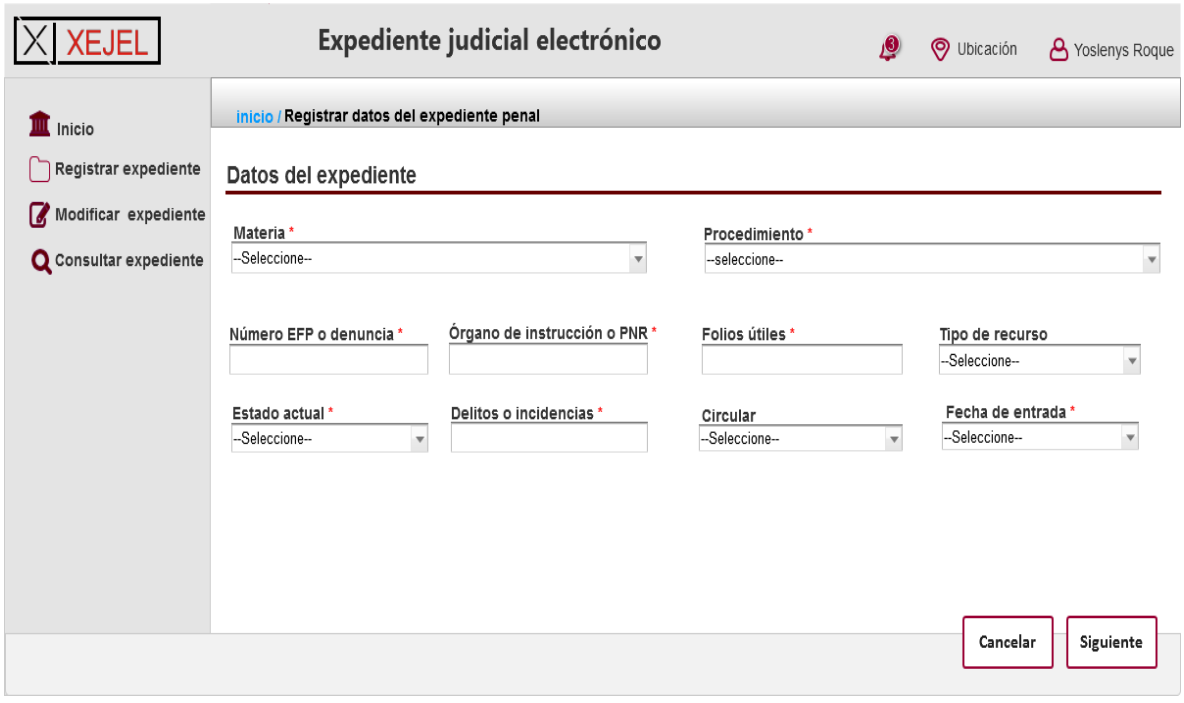
Botones:

Cancelar: permite regresar a la interfaz principal, sin tener en cuenta los cambios realizados.

Siguiente: permite validar y registrar los datos de la interfaz y muestra la interfaz siguiente.

Observaciones: al registrar un expediente Penal se necesita llenar los campos obligatorios.

Prototipo de interfaz gráfica de usuario:



Fuente: elaboración propia

Tabla 3: HU Registrar interviniente persona natural correspondiente al RF13

Número: 13	**Requisito:** Registrar interviniente persona natural
Programador: Maite Diosdado Jacobo	**Iteración Asignada:** 1
Prioridad: alta	**Tiempo Estimado:** 24 horas
Riesgo en Desarrollo: N/A	**Tiempo Real:** 21 horas

Descripción: funcionalidad que permite registrar los datos correspondientes a una persona natural como interviniente en un expediente.

Campos:

Nombre y apellidos: campo de texto para introducir el nombre y los apellidos de la persona natural. Es obligatorio.

Carné de Identidad: campo numérico para introducir el número de carné de identidad de la persona natural. No es obligatorio.

Sexo: campo de selección para escoger el sexo de la persona natural que se presenta. No es obligatorio.

Edad: campo numérico para captar la edad de la persona natural que se presenta. No es obligatorio.

Forma de comparecer: campo de selección para escoger la forma de comparecer de la persona que se está registrando en el expediente. Es obligatorio.

Medida cautelar: campo de selección que permite registrar la medida cautelar a la que está sujeta la persona que se está registrando. No obligatorio.

Tipo de sujeto: campo de selección que permite escoger el tipo de sujeto al que pertenece la persona que se está registrando. No obligatorio.

Ocupación: campo de texto para especificar la ocupación de la persona que se registra. No obligatorio.

OACE: campo de selección que permite escoger el organismo de la administración central del estado al que pertenece la persona que se está registrando. No obligatorio.

Contacto: campo de texto para introducir los datos de contactos (teléfono, correo, fax) de la persona que se está registrando. No obligatorio.

Dirección: campo de texto que permite introducir la dirección de la persona que se está registrando. No obligatorio.

Botones:

Cancelar: opción que descarta todos los cambios realizados y sale de la interfaz.

Adicionar: opción que valida que los datos de la interfaz sean correctos y registra los datos del interviniente.

Observaciones: al registrar un interviniente persona natural se necesita llenar los campos obligatorios.

Prototipo de interfaz gráfica de usuario:

Fuente: elaboración propia

2.4 Arquitectura del sistema

El diseño arquitectónico es la primera etapa en el proceso de construcción del software. Constituye el enlace crucial entre el diseño y la ingeniería de requerimientos, ya que identifica los principales componentes estructurales en un sistema y la relación entre ellos. Su salida consiste en un modelo que describe la forma en que se organiza el sistema como un conjunto de componentes en comunicación (Sommerville, 2005).

La solución propuesta tiene como base la arquitectura del XEJEL, separada en *frontend* y *backend.*

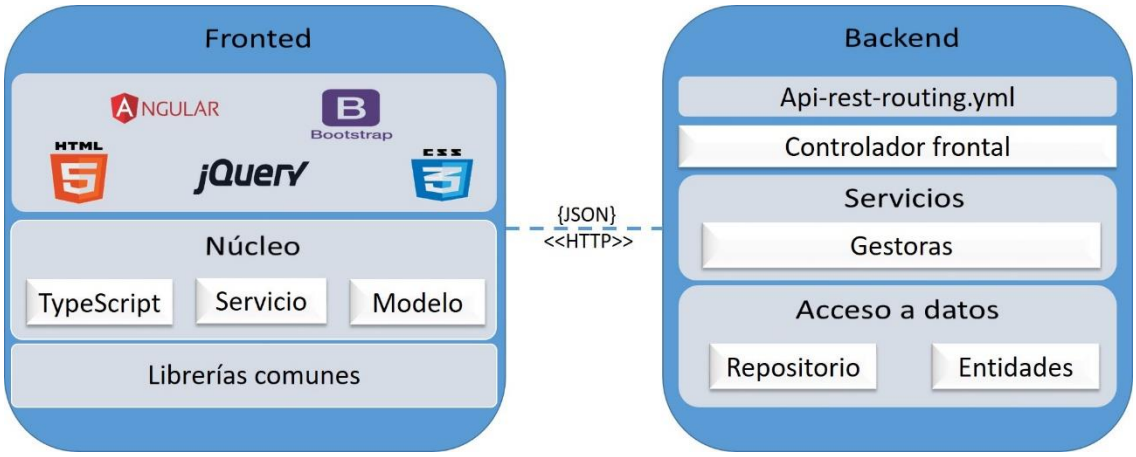


Figura 2: arquitectura del XEJEL

Fuente: elaboración propia

En la figura 2 se muestran los elementos de esta arquitectura y las partes donde inciden los componentes propuestos. El *frontend* es la parte del software que interactúa con los usuarios y el *backend* es la parte que procesa la entrada desde el *frontend.* La idea general es que el *frontend* sea el responsable de recolectar los datos de entrada del usuario y los transforma ajustándolos a las especificaciones que demanda el *backend* para poder procesarlos, devolviendo generalmente una respuesta que el *frontend* recibe y expone al usuario de una forma entendible. Para la solución propuesta la conexión entre ambas partes se realiza a través del protocolo HTTP intercambiando datos en formato JSON[16].

2.4.1 Arquitectura frontend

Un componente es un módulo de software que puede ser código fuente, código binario, un ejecutable, o una librería con una interfaz definida. Además, se representan las dependencias entre componentes o entre un componente y la interfaz de otro, es decir uno de ellos usa los servicios o facilidades del otro (Cillero, 2019).

En la figura 3 se muestra la descripción detallada de cómo se representa en el *frontend* del XEJEL la arquitectura de la presente propuesta de solución.

[16] Notación de objetos JavaScript, por sus siglas en inglés JavaScript Object Notation (ALEGSA, 2018).

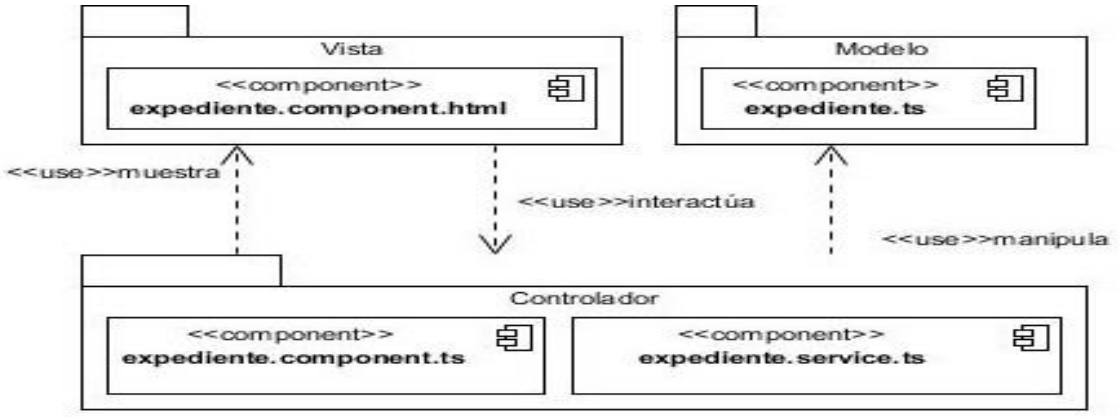


Figura 3: arquitectura *fronted* del componente Registrar expediente

Fuente: elaboración propia

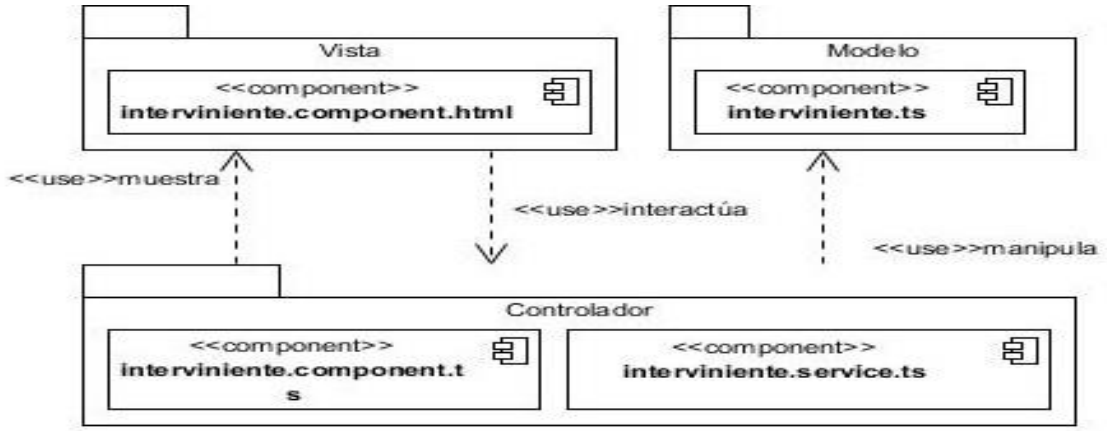


Figura 4: arquitectura *fronted* para el componente Registrar interviniente

Fuente: elaboración propia

En las figuras el modelo para cada componente está compuesto por los archivos *expediente.model.ts e interviniente.model.ts*, los cuales contienen la información que se le solicita al usuario para registrar los datos del expediente o de los intervinientes. Por otra parte, la vista se compone por los archivos *expediente.component.html* e *interviniente.componente.html,* estos representan la información visible al usuario e interactúan con funciones específicas del controlador. El controlador está compuesto por los archivos *expediente.component.ts* e *interviniente.componente.ts*, encargados de interactuar con el modelo y mostrar la información a la vista.

2.4.2 Arquitectura backend

A partir de los diagramas correspondientes a las figuras 5 y 6, en la parte del *backend* los componentes propuestos inciden en las capas controladora, gestora y acceso a datos. Estos componentes son implementados sobre la arquitectura *backend* del XEJEL.

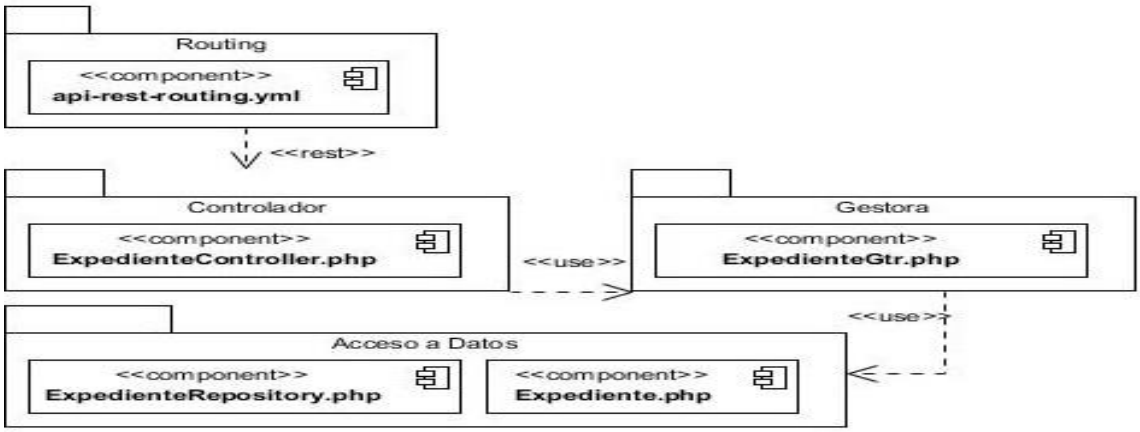


Figura 5: arquitectura *backend* para el componente Registrar expediente

Fuente: elaboración propia

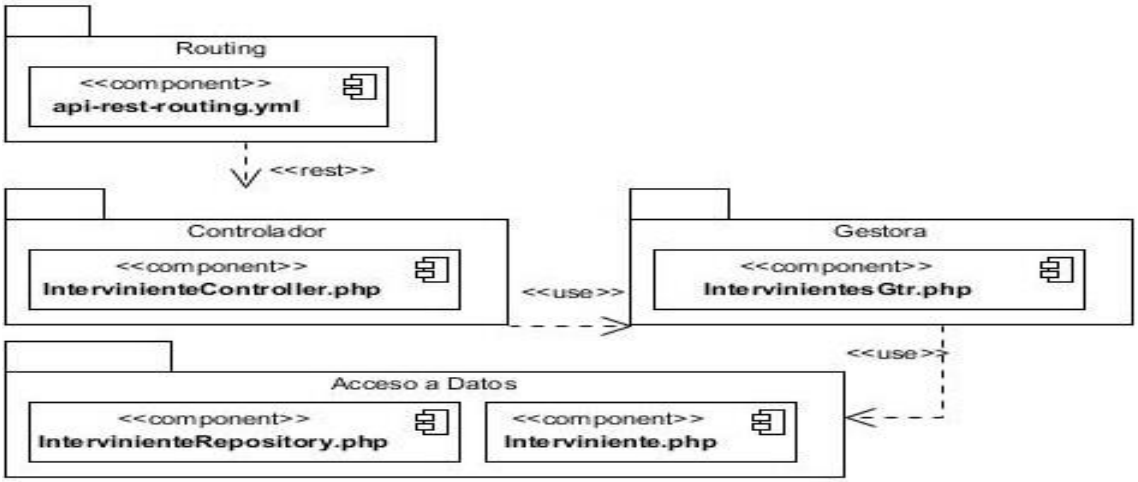


Figura 6: arquitectura *backend* para el componente Registrar interviniente

Fuente: elaboración propia

En las figuras 5 y 6 los componentes *ExpedienteController* e *IntervinienteController* de la capa Controladora usan los componentes de la capa Gestora *ExpedienteGtr* e *IntervinienteGtr*, los cuales usan a los componentes de la capa de Acceso a Datos

ExpedienteRepository e *IntervinienteRepository*. La relación entre los componentes del *frontend* y del *backend* se ve reflejada en el componente *api-rest.routing.yml.*

2.5 Patrones de diseño

Los patrones de diseño son técnicas para resolver problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. 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 (Pressman, 2010).

2.5.1 Patrón Modelo-Vista-Controlador (MVC)

Fue una de las primeras ideas en el campo de las interfaces gráficas de usuario y uno de los primeros trabajos en describir e implementar aplicaciones de software en términos de sus diferentes funciones (Nube Colectiva, 2019). De manera genérica, los componentes de MVC se definen como sigue:

Modelo: es un objeto que representa cierta información del dominio. Constituye un objeto no visual y contiene todos los datos y comportamiento del mismo (Nube Colectiva, 2019).

Vista: la vista representa la muestra del modelo mediante la interfaz visual (Nube Colectiva, 2019).

Controlador: se encarga de cambiar cualquier información existente en el modelo. Toma las entradas del usuario, manipula el modelo y actualiza la vista apropiadamente (Nube Colectiva, 2019). El uso de este patrón MVC se muestra en la Arquitectura *frontend* representada en las figuras 3 y 4.

2.5.2 Patrón N Capas

El patrón arquitectónico N Capas es una extensión del patrón Capas tradicional (ayuda a estructurar las aplicaciones que se pueden descomponer en grupos de sub-tareas en la que cada grupo está en un nivel particular de abstracción). En el nivel más alto y abstracto, la vista de arquitectura lógica de un sistema puede considerarse como un conjunto de servicios relacionados agrupados en diversas capas (Escalante, 2014). Los componentes de la presente propuesta de solución son implementados sobre la arquitectura *backend* del XEJEL a través del uso del patrón arquitectónico N Capas (Ver figuras 5 y 6).

2.5.3 Patrones GRASP[17]

Los patrones GRASP son una ayuda en el aprendizaje que permiten al desarrollador entender lo esencial del diseño de objetos y a aplicar el razonamiento de una forma metódica, racional y explicable (Pressman, 2010). A continuación se describen los utilizados en la propuesta de solución:

Patrón Controlador: 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 (Larman, 2016). Este patrón se evidencia en la solución a través de las clases controladoras que se encuentran en la carpeta *Controller* en el *backend*, y en la estructura de Symfony3 se evidencia en el uso del controlador frontal que se encuentra en la carpeta web de cada proyecto creado con este marco de trabajo. Para una mayor comprensión consultar la figura 7, en la cual se muestra el uso del patrón en la clase *ExpedienteController.php* correspondiente al componente de expediente de la propuesta de solución:

```php
class ExpedienteController extends BaseApiController
{

    /** @author Maite Diosdado <mdiosdado@estudiantes.uci.cu> ...*/
    public function listaProcedimientosXMateriaAction($idMateria, ExpedienteGtr $expedienteGtr){...}

    /** @author Maite Diosdado <mdiosdado@estudiantes.uci.cu> ...*/
    public function obtenerexpedienteAction($id, ExpedienteGtr $expedienteGtr){...}

    /** @author Maite Diosdado <mdiosdado@estudiantes.uci.cu> ...*/
    public function registrarExpAction(Request $request, ExpedienteGtr $expedienteGtr)
    {
        $expediente = $request->request->all();
        $idExpediente = $expedienteGtr->registrarExp($expediente, $this->getUser());

        return $this->view($idExpediente, StatusCodes::HTTP_OK);
    }
```

Figura 7: patrón controlador. Clase *ExpedienteController.php*

Fuente: elaboración propia

Patrón Experto: 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

[17] General Responsibility Assignment Software Patterns

encapsulada (Larman, 2016). 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 se engloban en la propuesta de solución. En la siguiente figura se muestra la clase entidad *ExpedienteController.php*, la cual es la experta en información para el tratamiento de los expedientes en el XEJEL.

```php
/** Expediente ...*/
abstract class Expediente
{
    const FORMAT_NUMERO_ANNO = '%s/%s';
    const FORMAT_NUMERO = '%s';
    /** @var int ...*/
    private $id;

    /** @var \DateTime ...*/
    private $fechaCreacion;

    /** @var \DateTime ...*/
    private $fechaEntrada;

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

    /** @var int ...*/
    private $foliosutiles;

    /** @ORM\ManyToOne(targetEntity= "AppBundle\Entity\Comun\TipoMateria") ...*/
    private $tipoMateria;

    /** @ORM\ManyToOne(targetEntity="AppBundle\Entity\Comun\TipoProcedimiento") ...*/
    private $tipoProcedimiento;

    /** @ORM\ManyToOne(targetEntity="AppBundle\Entity\Seguridad\Estructura", inversedBy="expediente") ...*/
    private $sala;
```

Figura 8: patrón experto. Clase *Expediente.php*

Fuente: elaboración propia

Patrón Creador: ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases (Larman, 2016). Este patrón se evidencia en la propuesta de solución en las clases gestoras en el momento de crear nuevas instancias de las clases entidades, en la figura 9 se muestra como, desde la clase *ExpedienteGtr.php* se hace instancia de la clase *Expediente.php* al crear un objeto de tipo expediente.

```php
$expediente = new ExpedientePenal();
$expediente->setTipoMateria($tipoMateria);
$expediente->setFechaCreacion($fechaActual);
$expediente->setFechaEntrada($fechaEntrada);
$expediente->setTipoProcedimiento($tipoProcedimiento);
$expediente->setSala($estructura);
$estadoExpediente->setExpediente($expediente);
```

Figura 9: patrón creador. Clase *ExpedienteGtr.php*

Fuente: elaboración propia

Patrón Bajo acoplamiento: este patrón se basa en la baja dependencia que debe existir entre las clases, además está estrechamente relacionado con los patrones Experto o Alta Cohesión (Larman, 2016). En el proyecto XEJEL se evidencia este patrón a través del marco de trabajo Symfony, el cual favorece ampliamente el bajo acoplamiento de las clases en el sistema, ya que a cada clase se le asignan solamente las responsabilidades necesarias de manera que no dependan en gran medida de otras (Ver figura 6).

Patrón Alta cohesión: este patrón tiene como propósito asignar responsabilidades de manera que la cohesión siga siendo alta (Larman, 2016). El marco de trabajo Symfony favorece la alta cohesión asignando responsabilidades a las clases de tal manera que estas se encuentren estrechamente relacionadas entre sí y no lleguen a realizar un trabajo excesivo. En el proyecto XEJEL se evidencia este patrón a través de la interrelación que existe entre las clases controladora, las clases gestora y las funcionalidades del sistema, la primera encargada de manejar la lógica de presentación y el flujo de los datos provenientes de la vista y la segunda encargada de manejar la lógica del negocio de cada funcionalidad (Ver figura 6).

2.5.4 Patrones GoF[18]

Los patrones GoF son soluciones concretas y técnicas basadas en la Programación Orientada a Objetos (POO). Se utilizan en situaciones frecuentes, debido a que se basan en la experiencia acumulada al resolver problemas reiterativos. Ayudan a construir software basado en la reutilización y a construir clases reutilizables (Pérez, 2013).

Patrón Decorador: su principal objetivo es añadir responsabilidades a objetos concretos de manera dinámica y transparente sin afectar a otros objetos. Este patrón brinda más flexibilidad que la herencia estática y evita que las clases más altas en la jerarquía estén demasiado cargadas de funcionalidad y sean complejas (Gamma, y otros, 1994). La aplicación de este patrón en el XEJEL se evidencia en la implementación de los componentes en el *frontend* como se muestra en la siguiente figura, ya que se le añaden funcionalidades a la clase: adicionar-litigante,component.html :

[18] Gang of Four (ALEGSA, 2018).

```
@Component({
  selector: 'app-adicionar-litigante',
  templateUrl: './adicionar-litigante.component.html',
  styleUrls: ['./adicionar-litigante.component.scss']
})
```

Figura 10: patrón decorador. Clase *Adicionar-interviente. component.ts*

Fuente: elaboración propia

Patrón observador: permite observar los cambios producidos por un objeto. Es uno de los principales patrones de diseño utilizados en interfaces gráficas de usuario, ya que permite desacoplar al componente gráfico de la acción a realizar (Larman, 2016). Este patrón es empleado en el proyecto XEJEL en varias partes de la implementación de los componentes en el *frontend*, ejemplo al enviar los datos de la vista al controlador, este último hace la petición de un servicio a través de una inyección y se queda a la espera de la respuesta que este servicio debe proveer para así emitir o no un evento. En el método que se muestra en la siguiente figura se evidencia la aplicación de este patrón:

```
onSubmit() {
  this.litiganteService.insertarLitigante(this.litigante, this.idExpediente)
    .subscribe(response => {
      this.eventoGuardar.emit(this.idExpediente);
    });
}
```

Figura 11: patrón observador

Fuente: elaboración propia

Patrón fabricación pura: 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 pura casi siempre se dividen atendiendo a su funcionalidad, dicho con otras palabras, se confeccionan clases destinadas a conjuntos de funciones (Larman, 2016). Generalmente se considera que la fabricación es parte de la capa de servicios orientada a objetos de alto nivel en una arquitectura. Este patrón se evidencia en las clases útiles implementadas en el XEJEL, las cuales forman parte de los servicios de la aplicación. En la figura que se muestra a continuación se observa la clase *EstructuraUtil.php* de la cual la propuesta de solución se nutre de los servicios que posee la misma al reflejar el tratamiento con las salas y tribunales en el sistema en el momento de retornar el identificador del tribunal o la sala en la que se está trabajando.

```php
class EstructuraUtil
{
    const TRIBUNAL_ACTUAL = 'tribunal-actual';
    const SALA_ACTUAL = 'sala-actual';

    /** Retorna el id del tribunal en el q se esta trabajando ...*/
    public static function getTribunalActual(){...}

    /** Retorna el id de la sala en la q se esta trabajando ...*/
    public static function getSalaActual(){...}

    /** Retorna si se esta trabajando en una sala, o aun no se he antrado al alguna ...*/
    public static function isInSala(){...}

    /** Metodo que retorna la header de key, la q se especifica en el parametro ...*/
    private static function getValue($key){...}
}
```

Figura 12: patrón Fabricación pura. Clase *EstructuraUtil.php*

Fuente: elaboración propia

2.5.5 Otros patrones

Patrón Inyección de dependencias: 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 (Pratt, 2013). En el método del XEJEL que se muestra en la siguiente figura se evidencia la aplicación de este patrón ya que se le pasa al constructor un conjunto de parámetros que luego serán utilizados en su implementación:

```typescript
constructor(private serviceExpediente: ExpedienteService,
            private fechaservice: FechaUtil,
            private activatedRoute: ActivatedRoute,
            private notificationService: NotificationsService) {...}
```

Figura 13: patrón Inyección de dependencias. Clase *datos-expediente.component.ts*

Fuente: elaboración propia

2.6 Diagrama de clases del diseño

El diagrama de clases recoge las clases de objetos y sus asociaciones. En este diagrama se representa la estructura y el comportamiento de cada uno de los objetos del sistema y sus relaciones con los demás objetos. Con el fin de facilitar la comprensión del diagrama, se pueden incluir paquetes como elementos del mismo, donde cada uno de ellos agrupa un conjunto de clases (Cillero, 2017).

En los anexos 3 y 4, se muestran los diagramas de clases del diseño para los componentes Registrar expediente y Registrar interviniente. En ambos se evidencian las principales

clases de la vista, las variables y métodos que las componen y la relación entre ellas. Además, queda evidenciado el uso del protocolo HTTP para intercambiar datos en formato JSON con las clases del negocio que a su vez permiten la relación con las clases entidades.

2.7 Modelo de datos

El modelo de datos se utiliza para describir la estructura lógica y física de la información persistente gestionada por el sistema. Se utiliza para definir la correlación entre las clases de diseño persistentes y las estructuras de datos persistentes, y para definir las estructuras de datos persistentes (Tecnologías-Información, 2018). A continuación, se muestra el modelo de datos correspondiente a la solución propuesta.

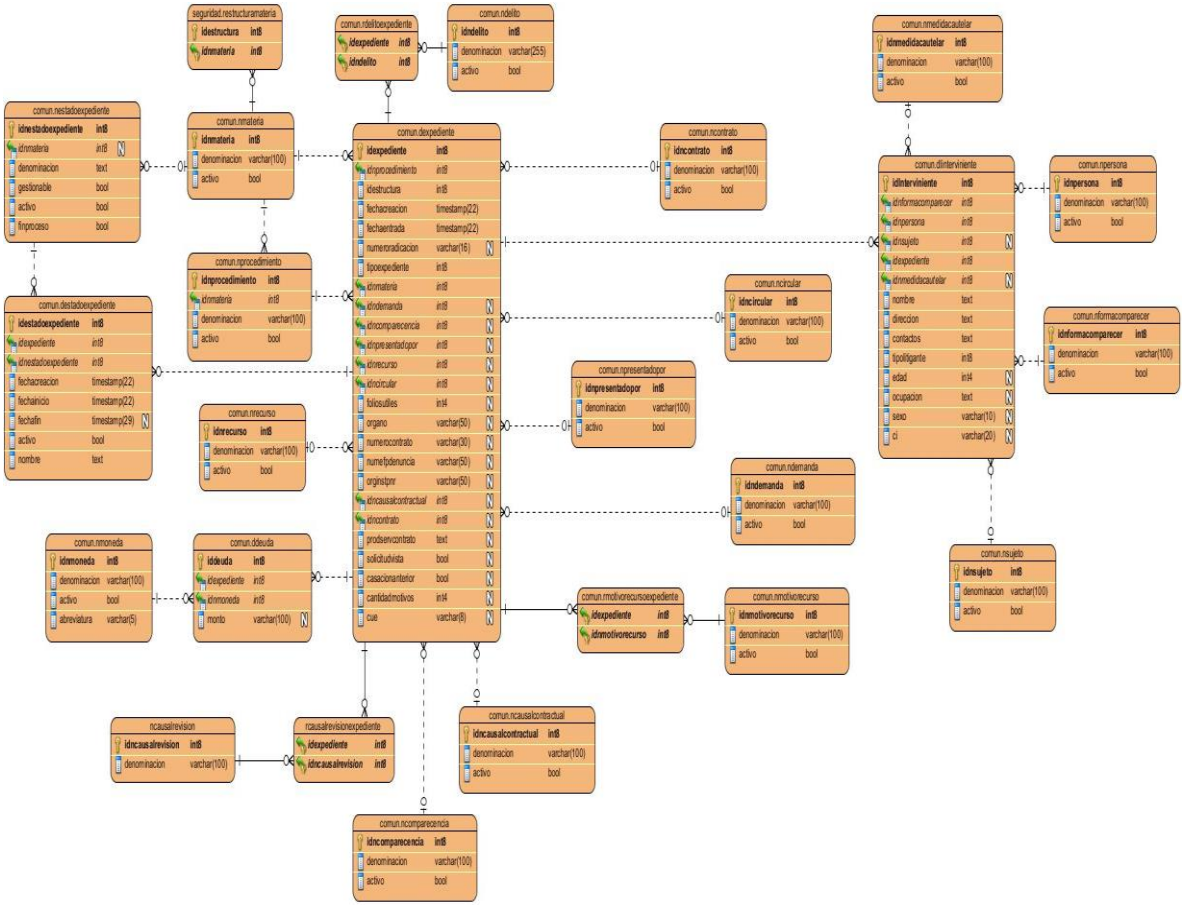


Figura 14: modelo de datos

Fuente: elaboración propia

El modelo de datos obtenido cuenta con un total de 26 tablas, de ellas 17 son

nomencladores encargados de gestionar conceptos específicos del negocio, anteriormente predefinidos, por ejemplo, la tabla *comun_nmateria*, que se relaciona con la tabla *comun_nestado* y se definen los tipos de estados que tiene el expediente.

2.8 Estándares de codificación

Los estándares de codificación son normas que deben seguirse durante todo el ciclo de implementación del sistema (Ohmyroot, 2017). Para la presente solución los estándares fueron seleccionados por el equipo de arquitectura y diseño del proyecto. Esta información se encuentra plasmada íntegramente en el documento "*CEGEL_XEJEL_Estandares_de_codificacion_para_PHP.doc*". A continuación, se describen los utilizados en la implementación de los componentes de la presente investigación:

Declaración de clases

Las Clases deben ser nombradas de acuerdo a la convención de nombres.
Cada clase debe contener un bloque de documentación acorde con el estándar de *PHPDocumentor*.
Todo el código contenido en una clase debe ser separado con cuatro espacios.
En la figura 15, se muestra un ejemplo de una declaración de clase teniendo en cuenta los aspectos definidos anteriormente:

```
class ExpedienteController extends BaseApiController
{
    /**
     * @author Linda Liz Vera <llvera@estudiantes.uci.cu>
     * @Rest\Get("/getprocxmateria/{idMateria}")
     * @Rest\View(serializerGroups={"default"})
     * @param $idMateria
     * @param ExpedienteGtr $expedienteGtr
     * @return \FOS\RestBundle\View\View
     * @throws \Doctrine\DBAL\DBALException
```

Figura 15: declaración de clases

Fuente: elaboración propia

Funciones y métodos

Los nombres de funciones pueden contener únicamente caracteres alfanuméricos. Los guiones bajos (_) no están permitidos. Los números están permitidos en los nombres de las funciones, pero no se aconseja en la mayoría de los casos.

Los nombres de funciones deben empezar siempre con una letra minúscula. Cuando un nombre de función consiste en más de una palabra, la primera letra de cada nueva palabra debe estar en mayúsculas. Esto es llamado comúnmente como formato "*camelCase*".
En la figura que se muestra a continuación queda evidente el uso de lo descrito anteriormente en el nombre de la función *adicionarExpedienteAction()*.

```
public function adicionarExpedienteAction(Request $request, ExpedienteGtr $expedienteGtr)
{
    $expediente = $request->request->all();
    $expedienteGtr->insertarExpediente($expediente);

    return $this->viewSuccess( msg: 'Se ha creado el expediente',  code: StatusCodes::HTTP_CREATED);
}
```

Figura 16: funciones y métodos

Fuente: elaboración propia

Constantes

Las constantes pueden contener tanto caracteres alfanuméricos como barras bajas (_). Los números están permitidos. Todas las letras pertenecientes al nombre de una constante deben aparecer en mayúsculas. Las palabras dentro del nombre de una constante deben separarse por barras bajas (_). En la imagen que se muestra a continuación se evidencia el uso de las reglas descritas.

```
const ADMINISTRATIVA = 1;
const CIVIL = 2;
const ECONOMICO = 3;
const LABORAL = 4;
const PENAL = 5;
```

Figura 17: definición de contantes

Fuente: elaboración propia

Ubicación y denominación de archivos

La denominación y ubicación de los archivos siguen las convenciones establecidas por el equipo de arquitectura. Ejemplo:
Para las clases controladoras se usa el sufijo *Controller*: Ver figura 15, para las clases de gestión del negocio se usa el sufijo Gtr como se muestra en la figura 18.

```
class ExpedienteGtr extends BaseGtr
```

Figura 18: denominación de las clases gestoras

Fuente: elaboración propia

Para los *table model* definidos para los *grid* [19] se usará el sufijo Tm:

```
class LitiganteTm extends BaseTableModel
```

Figura 19: denominación de los *table model*

Fuente: elaboración propia

Para las clases repositorio se usará el sufijo *Repository*:

```
class LitiganteRepository extends BaseRepository
```

Figura 20: denominación de las clases *Repository*

Fuente: elaboración propia

Estilo y reglas de escritura de código PHP

Nombres de variables

Los nombres deben ser descriptivos y concisos. No usar ni grandes frases ni pequeñas abreviaciones para las variables. Siempre es mejor saber qué hace una variable con solo conocer su nombre. Esto aplica para los nombres de variables, funciones, argumentos de funciones y clases. Los nombres de las variables y de las funciones pueden iniciar con letra minúscula, pero si estas tienen más de una palabra, cada nueva palabra debe iniciar con letra mayúscula. Las constantes deben escribirse siempre en mayúsculas y tanto estas, como las variables globales, deben de tener como prefijo el nombre de la clase a la que pertenecen. Ver figura 16.

Las definiciones de la función

Los nombres de las funciones pueden contener sólo caracteres alfanuméricos. Los mismos

[19] Función que divide un área de trabajo en cuadrículas. Cuadrícula para presentar datos en forma de tabla (ALEGSA, 2018).

siempre deben empezar en letras minúsculas y si tiene más de una palabra, la primera letra de cada nueva palabra debe capitalizarse. Ver figura 16.

Llamadas a funciones

Deben llamarse las funciones sin los espacios entre el nombre de la función, el paréntesis de la apertura, y el primer parámetro; los espacios entre las comas y cada parámetro, y ningún espacial entre el último parámetro, el paréntesis del cierre, y el punto y coma. Ver figura 16.

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. Ver figura 16.

No utilizar variables sin inicializar

Si no se tiene control sobre el valor de una variable, se debe verificar que esté inicializada. Esto lo permite PHP utilizando el *isset* [20]como se muestra en la figura 21.

```
if (isset($litigante['edad']) && $litigante['edad'] != null) {
    $litiganteNuevo->setEdad($litigante['edad']);
} else {
    $litiganteNuevo->setEdad(null);
}
```

Figura 21: inicialización de variables

Fuente: elaboración propia

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

Instrucción *"switch"*

Al utilizar esta instrucción es recomendable seguir el estilo que se muestra en la figura 22:

[20]Evaluar si una variable está definida (ALEGSA, 2018).

```
switch ($evento.id) {
    case 1:
        this.activado = 'ad';
        break;
    case 2:
        this.activado = 'cv';
        break;
```

Figura 22: instrucción *switch*

Fuente: elaboración propia

2.9 Interfaces de la solución

El resultado del proceso de implementación se traduce en componentes funcionales que son integrados entre sí y a su vez al sistema XEJEL. Los especialistas judiciales, después de estar autenticados en el sistema solo pueden acceder a las funcionalidades que les competen y que se encuentran en el menú lateral que se señala en el anexo 5. La interfaz de ese anexo muestra el formulario donde se adicionan los datos de un expediente según la materia y el procedimiento al que pertenezca. En el anexo 6 se muestra el formulario para adicionar una persona natural al expediente del que forma parte como interviniente en el asunto.

2.10 Conclusiones del capítulo

El empleo de la metodología AUP en su variación para la UCI, permitió organizar el desarrollo de la solución y generar los productos de trabajo necesarios correspondientes a las etapas de requisitos y de análisis y diseño. La arquitectura del sistema dividida en *frontend* y *backend* permite la ampliación de las funcionalidades de la solución y la reutilización de los componentes de la presente solución en aplicaciones futuras. Por otra parte, los patrones de diseño y estándares de codificación, garantizaron la obtención de una solución con poca dependencia entre clases, flexible al mantenimiento y a la introducción de cambios. El empleo de los estándares de codificación facilita la lectura, comprensión y mantenimiento del código para los desarrolladores.

CAPÍTULO 3: VALIDACIÓN Y PRUEBA

En este capítulo se evalúa el grado de calidad y fiabilidad de los requisitos obtenidos en el desarrollo de la investigación, donde esta evaluación se lleva a cabo a partir de la validación del diseño a través de las métricas aplicadas a las clases. Además, se aplican pruebas de software para verificar y revelar la calidad de la solución propuesta antes de ser entregada al cliente.

3.1 Técnicas de validación de requisitos

Con el objetivo de ratificar que los requisitos del software obtenidos definen el sistema que el cliente desea se llevó a cabo un proceso de validación de los mismos, para el cual se emplearon las siguientes técnicas:

Revisiones de los requisitos: se realizaron revisiones a cada uno de los requisitos por parte del equipo de desarrollo. Las revisiones internas generaron un total de 5 no conformidades de tipo técnicas y de ortografía, las cuales fueron corregidas satisfactoriamente. Con el equipo de análisis y líder de proyecto se realizó la revisión en la cual se añadieron detalles a 3 requisitos funcionales y se aprobaron finalmente los mismos, generándose de este encuentro la firma del documento "*Especificación_de_requisitos_de_software"* por parte del cliente.

Construcción de prototipos: se mostró al cliente un modelo ejecutable que permitió tener una visión preliminar de cómo se verían estos componentes en el sistema y a través de la interacción con los mismos se comprobó la satisfacción y aprobación del cliente, los mismos fueron aprobados satisfactoriamente.

Generación de casos de prueba: como parte del proceso de validación de los requisitos funcionales fueron diseñados casos de pruebas para cada requisito.

3.2 Métricas aplicadas a los requisitos

Con el objetivo de medir la calidad de la especificación de los requisitos se aplicó una de las métricas propuestas para la metodología AUP (UCI), Calidad de la especificación (CE). Para obtener cuán entendibles y precisos son los requisitos, primeramente, se calcula el total de requisitos de la especificación como se muestra a continuación:

Nr: el total de requisitos de especificación.

Nf: cantidad de requisitos funcionales.

Nnf: cantidad de requisitos no funcionales.

$$Nr = Nf + Nnf$$

Como resultado de la sustitución de los valores, para el XEJEL se obtiene:

$$Nr = 22 + 13$$
$$Nr = 35$$

Para determinar, finalmente, la Especificidad de los Requisitos (ER) o ausencia de ambigüedad en los mismos se realiza la siguiente operación:

$$ER = Nui / Nr$$

Donde *Nui* es el número de requisitos para los cuales todos los revisores tuvieron interpretaciones idénticas. Mientras más cerca de 1 esté el valor de ER, menor será la ambigüedad.

Para el caso de los requisitos obtenidos para el XEJEL, 5 produjeron contradicción en las interpretaciones. Sustituyendo las variables se obtiene:

$$ER = 30 / 35$$
$$ER = 0.86$$

Arrojando un resultado final satisfactorio, indicando que el grado de ambigüedad de los requisitos es bajo (14%) ya que el 86% son entendibles. Los requisitos ambiguos fueron modificados y validados para garantizar una correcta interpretación.

3.3 Validación del diseño

Con el objetivo de validar el diseño realizado se aplicaron las siguientes métricas de diseño: Relaciones entre Clases (RC) y Tamaño Operacional de Clase (TOC).

3.3.1 Relación entre clases (RC)

Permite evaluar el acoplamiento, la complejidad de mantenimiento, la reutilización y la cantidad de pruebas de unidad necesarias para probar una clase teniendo en cuenta las relaciones existentes entre ellas (Pressman, 2010). La figura que se muestra a continuación representa la cantidad de clases por cantidad de relaciones de usos que poseen.

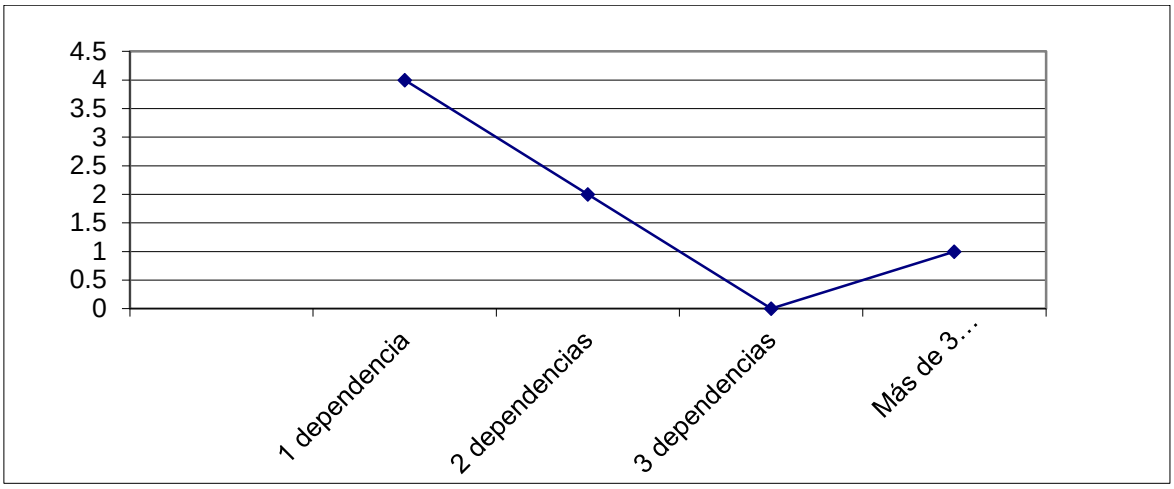


Figura 23: representación de la cantidad de clases por cantidad de relaciones de usos que poseen

Fuente: elaboración propia

En la gráfica se observa como 4 clases presentan 1 sola dependencia, 2 clases presentan 2 dependencias, no existe ninguna clase con 3 dependencias y existe 1 clase con más de tres dependencias.

En la figura 24 que se muestra a continuación se representa en porciento (%) el nivel de acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización de las clases al aplicar la métrica RC.

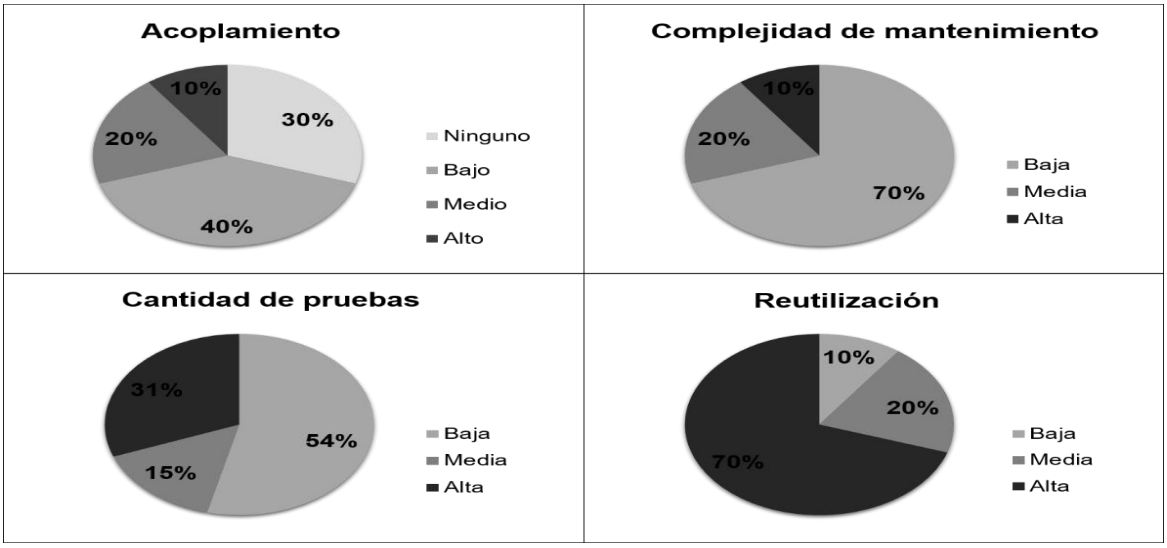


Figura 24: representación en porciento (%) del nivel de acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización de las clases

Fuente: elaboración propia

Una vez aplicada la métrica y teniendo en cuenta el umbral definido para validar el diseño (Bien [0.1; 0.3], Regular [0.4; 0.7] y Mal [0.8; 1]), se obtiene como resultado que las clases del diseño promueven el bajo acoplamiento, la complejidad de mantenimiento y la cantidad de pruebas no son altas y en consecuencia el grado de reutilización es mayor.

3.3.2 Tamaño operacional de clase (TOC)

Permite medir la responsabilidad, la complejidad de implementación y la reutilización de las clases del diseño. Es importante destacar que, para esta métrica, la responsabilidad y la complejidad son inversamente proporcionales a la reutilización, por lo que, a mayor responsabilidad y complejidad de implementación de una clase, menor es su nivel de reutilización (Pressman, 2010).

En la figura 25 que se muestra a continuación se evidencian los resultados obtenidos de la aplicación de la métrica TOC:

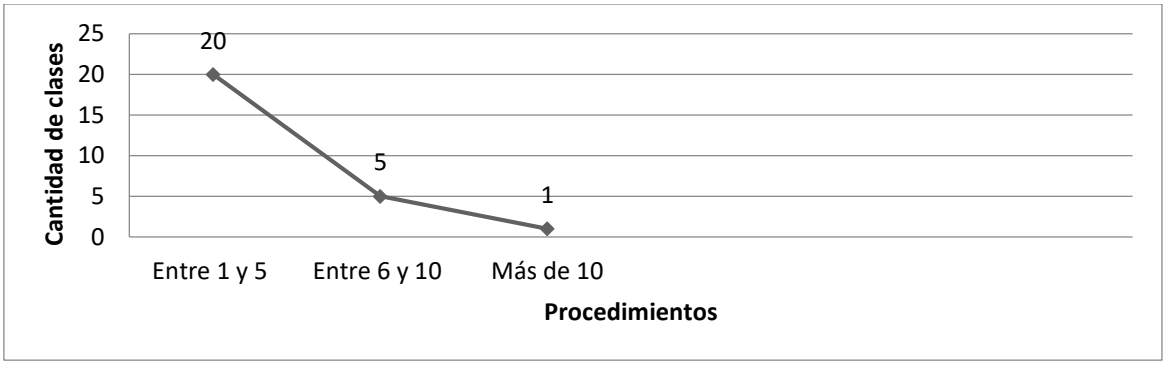


Figura 25: representación de la cantidad de clases por cantidad de procedimientos que contienen

Fuente: elaboración propia

En la gráfica se observa que existe 20 clases que presentan entre 1 y 5 procedimientos, 5 clases que tienen entre 6 y 10 procedimientos, y 1 clase que tiene más de 10 procedimientos.

En la figura 26 que se muestra a continuación se representa en porciento (%) el nivel de responsabilidad, complejidad de implementación y reutilización de las clases al aplicar la métrica TOC.

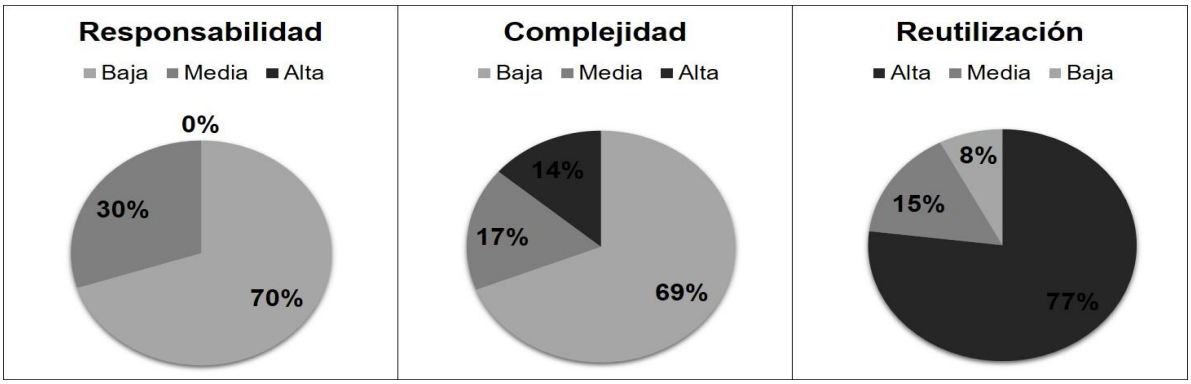


Figura 26: representación en porciento (%) del nivel de responsabilidad, complejidad de implementación y reutilización de las clases

Fuente: elaboración propia

Luego de aplicada la métrica se observa que las clases del diseño no se encuentran sobrecargadas en cuanto a responsabilidades y el nivel de complejidad de las mismas no es elevado, teniendo una alta reutilización.

De manera general, los resultados obtenidos de la aplicación de las métricas TOC y RC demuestran que el diseño no es complejo, que las clases presentan bajo acoplamiento y un alto grado de reutilización, lo que trae consigo que las clases puedan reutilizarse favoreciendo la implementación.

3.4 Pruebas de software

Proceso que permite verificar y revelar la calidad de un producto software antes de ser entregado al cliente. Es una fase en el desarrollo de software que consiste en probar las aplicaciones construidas (Institute Puig Castelar, 2018). La metodología seleccionada para guiar el proceso de desarrollo de software define las pruebas como una de las disciplinas a tener en cuenta. La misma abarca las pruebas internas, las pruebas de liberación y las pruebas de aceptación que serán analizadas en los epígrafes posteriores.

3.4.1 Pruebas internas

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

automatizar las pruebas (Sánchez, 2015).

Pruebas unitarias

Las pruebas unitarias se realizan sobre las funcionalidades internas de un módulo y se encargan de comprobar los caminos lógicos, ciclos (bucles) y condiciones que debe cumplir el programa (Pressman, 2010).

Para aplicar las pruebas unitarias, los autores de la presente investigación definen utilizar el método de Caja blanca. Con el empleo de este método es posible desarrollar casos de prueba que garanticen la ejecución, al menos una vez, de los caminos independientes (Pressman, 2010). Para aplicar este método se define la técnica de ruta básica.

Técnica de ruta básica

La técnica de ruta básica es empleada en el método de Caja blanca, la misma tiene como objetivo comprobar que cada camino se ejecute independiente de un componente o programa, obteniéndose una medida de la complejidad lógica del diseño. Esta técnica de prueba debe ser utilizada para evaluar la efectividad de los métodos asociados a una clase, con el objetivo de asegurar que cada camino independiente sea ejecutado por lo menos una vez en el sistema (Pressman, 2010).

Pressman propone como estrategia para aplicar la ruta básica, realizar un análisis de la complejidad ciclomática de cada procedimiento que componen las clases del sistema; una vez concluido este paso se selecciona el método con valor de contener errores, además de que ofrece una medida del número de pruebas que deben diseñarse para validar la correcta implementación de una determinada función.

Para obtener los casos de prueba a partir de la técnica ruta básica, se debe construir el grafo de flujo para cada uno de los métodos desarrollados, a continuación se muestra un ejemplo de la aplicación de la técnica correspondiente al código del método *insertarLitigante()* que es uno de los métodos más complejos desde el punto de vista de implementación y que se muestra en la siguiente figura:

Figura 27: método *insertarLitigante()* utilizado como ejemplo para la técnica de ruta básica

Fuente: elaboración propia

A continuación se muestra, en la figura 28, el grafo de flujo obtenido a partir del método anteriormente descrito.

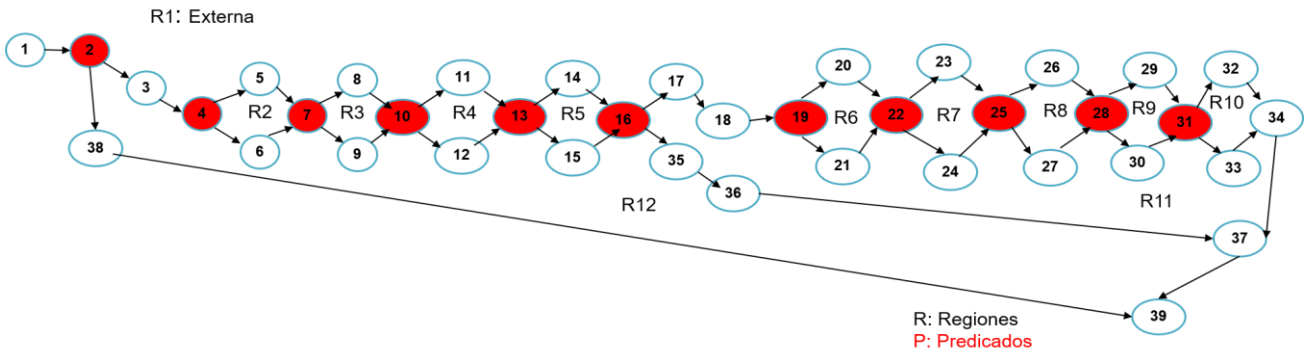


Figura 28: grafo del flujo del método *insertarLitigante()*

Fuente: elaboración propia

Luego se determina la complejidad ciclomática V (G) del grafo resultante, la cual es un indicador del número de caminos independientes que existen en un grafo, es decir, es cualquier camino dentro del código que introduce por lo menos un nuevo conjunto de sentencias de proceso o una nueva condición. La complejidad ciclomática puede ser calculada de 3 formas:

1. **V(G)** = R
2. **V(G)** = E – N + 2
3. **V (G)** = P + 1

Conociendo que:

- **G:** Grafo de flujo (grafo)
- **R:** El número de regiones contribuye a estimar el valor de la complejidad ciclomática.
- **E:** Número de aristas
- **V(G):** Complejidad ciclomática
- **N:** Número de nodos del grafo
- **P:** Número de nodos predicados (nodos de donde parten al menos dos aristas).

Realizando los cálculos correspondientes se obtiene por cualquiera de las variantes el siguiente resultado:

V(G) = R	**V(G)** = E – N + 2	**V(G)** = P + 1
V(G) = 12	**V(G)** = 49 – 39 + 2	**V(G)** = 11 + 1
	V(G) = 12	**V(G)** = 12

Una vez calculada la complejidad ciclomática, el valor obtenido representa el límite superior de pruebas que debe aplicarse (Pressman, 2010). Por lo que el conjunto de caminos básicos son:

Camino básico 1: 1-2-38-39
Camino básico 2: 1-2-3-4-5-7-8-10-11-13-14-16-35-36-37-39
Camino básico 3: 1-2-3-4-5-7-8-10-11-13-14-16-17-18-19-20-22-23-25-26-28-29-31-32-34-37-39
Camino básico 4: 1-2-3-4-6-7-8-10-11-13-14-16-35-36-37-39
Camino básico 5: 1-2-3-4-5-7-9-10-11-13-14-16-35-36-37-39
Camino básico 6: 1-2-3-4-5-7-9-10-12-13-15-16-35-36-37-39
Camino básico 7: 1-2-3-4-6-7-9-10-12-13-15-16-35-36-37-39
Camino básico 8: 1-2-3-4-6-7-9-10-12-13-15-16-17-18-19-21-22-24-25-27-28-30-31-33-34-37-39
Camino básico 9: 1-2-3-4-5-7-9-10-12-13-15-16-17-18-19-21-22-24-25-27-28-30-31-33-34-37-39
Camino básico 10: 1-2-3-4-5-7-8-10-12-13-15-16-17-18-19-21-22-24-25-27-28-30-31-33-34-37-39
Camino básico 11: 1-2-3-4-5-7-9-10-12-13-14-16-17-18-19-21-22-24-25-27-28-30-31-33-34-37-39
Camino básico 12: 12:1-2-3-4-5-7-8-10-11-13-15-16-17-18-19-21-22-24-25-27-28-30-31-33-34-37-39

Luego se definen los casos de prueba para cada uno de los caminos básicos obtenidos, en cada una de las funciones de la solución. A continuación, solamente se presenta el caso de prueba definido para la ruta independiente 1.

Tabla 4: caso de prueba de la ruta independiente 1

Descripción	Se inserta el interviniente asociado al expediente.
Condición de ejecución	El interviniente ya debe encontrarse en el expediente
Entradas	Se introduce el mismo nombre y apellidos o denominación del interviniente y la misma forma de comparecer.
Resultados esperados	Se debe mostrar una alerta especificando que el interviniente ya se encuentra en el expediente.

Fuente: elaboración propia

Resultados al aplicar la técnica de ruta básica

Esta técnica se aplicó a los métodos de las clases controladoras; estas clases fueron seleccionadas debido a que engloban las funcionalidades del sistema. Para comprobar la fiabilidad del código fuente se realizaron dos iteraciones completas en busca de errores de codificación, como condición de parada se tuvo en cuenta lo definido como resultado esperado en los casos de prueba. Se realizó una segunda iteración donde se erradicaron las No conformidades, por lo que se puede concluir que luego de realizar la prueba de caja blanca, donde se diseñaron y ejecutaron los casos de prueba correspondientes, se logró asegurar el cumplimiento del proceso de mejora del código.

Pruebas funcionales

Las pruebas funcionales son pruebas diseñadas tomando como referencia las especificaciones funcionales de un componente o sistema (lo que se va a testear, el software o una parte de él). Se realizan para comprobar si el software cumple las funciones esperadas (Pressman, 2010).

Método de Caja negra

El método de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La misma no es una alternativa a las técnicas de método de caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de caja blanca. Estas pruebas permiten encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación (Pressman, 2010).

Para llevar a cabo el método de Caja negra se utiliza la técnica de Partición equivalente que a continuación se describe.

Técnica de prueba: partición equivalente

Esta técnica divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. El método se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de prueba que deben desarrollarse. Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entrada. (Pressman, 2010)
Para aplicar esta técnica, se deben primeramente realizar el Diseños de casos prueba (DCP) con el objetivo de obtener un conjunto de pruebas que tengan la mayor probabilidad de descubrir los defectos del software.

Un DCP es, en ingeniería del software, un conjunto de condiciones o variables bajo las cuales un analista determinará si una aplicación o una característica de éstos es parcial o completamente satisfactoria (Pressman, 2010).

Con el objetivo de comprobar el funcionamiento de los componentes, se realizaron un total de 3 iteraciones de pruebas, obteniéndose los resultados que se muestran en la siguiente figura:

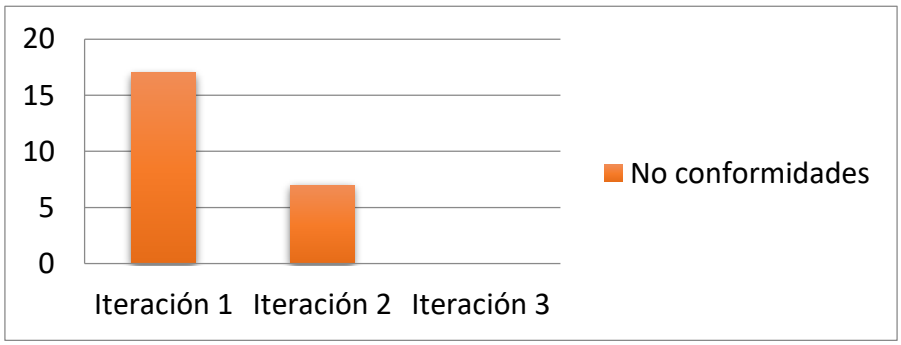


Figura 29: representación de la cantidad de no conformidades por iteraciones

Fuente: elaboración propia

En una primera iteración se detectaron un total de 17 no conformidades (9 de validación, 3 de ortografía y 5 de interfaz). En la segunda iteración se manifestaron 7 no conformidades donde aún existían 5 errores de validación y 2 errores ortográficos. Finalmente, en una tercera iteración se obtuvieron resultados satisfactorios al no mostrar no conformidades.

3.4.2 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 (Sánchez, 2015). Las pruebas realizadas a los componentes de la propuesta de solución se desarrollaron en un total de 3 iteraciones obteniéndose los resultados que se muestran en la siguiente figura:

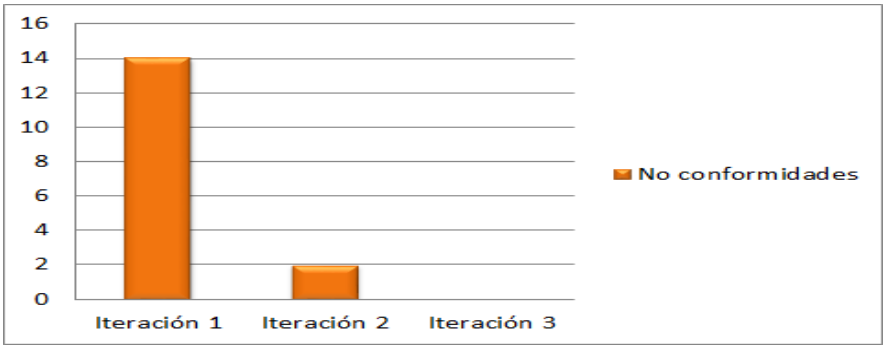


Figura 30: representación de la cantidad de no conformidades por iteraciones

Fuente: elaboración propia

En una primera iteración se detectaron un total de 14 no conformidades (10 errores de validación y 4 de correspondencia entre artefactos). En la segunda iteración se obtuvieron 2 no conformidades donde se revelaron errores de validación. Finalmente, en una tercera iteración se obtuvieron resultados satisfactorios ya que no se detectaron no conformidades. Como resultado de este proceso se generó el Acta de liberación por parte del equipo de Calidad del centro CEGEL (Ver anexo 7).

3.4.3 Pruebas de aceptación

Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar las funciones y tareas para las cuales fue construido (Sánchez, 2015). Esta prueba fue ejecutada en el entorno de producción por parte del equipo de análisis del sistema XEJEL y por los tutores de la presente investigación que ejercen como clientes en este caso. El resultado de esta prueba generó un Acta de Aceptación validada por el jefe del proyecto XEJEL (Ver anexo 8).

3.5 Validación de las variables de la investigación

Las deficiencias descritas en la introducción del presente trabajo de diploma afectan directamente el cumplimiento de las actividades fundamentales en los TPC. Para demostrar la existencia del problema y aplicar los instrumentos seleccionados se definieron las siguientes variables:

Variable dependiente: la estandarización y celeridad en la conformación de los expedientes.
Variable independiente: gestionar los datos generales del expediente y de los intervinientes.

A partir de los resultados obtenidos mediante la encuesta realizada a los especialistas judiciales del Tribunal Municipal Popular de Cienfuegos y el TSP (Ver anexo 1) y una vez realizadas las pruebas correspondientes a los componentes de la presente investigación se evidencia una reducción considerable del tiempo en el registro de la información. Los resultados se muestran en la siguiente tabla:

Tabla 5: validación de las variables de la investigación

Indicador	Antes	Después
Tiempo de búsqueda	El tiempo de búsqueda de un expediente de forma manual es aproximadamente de 5 a 10 minutos.	El tiempo de búsqueda de un expediente en el sistema es de 66,7 segundos.
Tiempo para recopilar datos	El tiempo para recopilar datos de un expediente de forma manual es aproximadamente de 30 minutos a 1 hora.	El tiempo para recopilar los datos de un expediente en el sistema es de 58,6 segundos.
Tiempo de Asignación de datos	El tiempo de asignación de datos en un expediente de forma manual es aproximadamente de 1	El tiempo de asignación de datos en el sistema es de 55,6 segundos.

	hora.	
Tiempo para completar datos de los Intervinientes	El tiempo para completar datos de los intervinientes aproximadamente es de 10 a 30 minutos.	El tiempo para completar los datos de los intervinientes en el sistema es de 49,7 segundos.
Estandarización de la información	No se tiene un formato estándar para cada tipo de expediente, por lo que se introduce los mismos datos con otras denominaciones.	Se tiene una estructura para cada tipo de expediente y se guardan en un mismo formato.

Fuente: elaboración propia

Luego de realizada la evaluación de las variables, analizados los resultados y las características de la solución de la presente investigación se demuestra la validez de la idea a defender al plantear que el desarrollo de los componentes para la gestión de los datos generales e intervinientes durante la conformación de los expedientes en los Tribunales Populares Cubanos, contribuyen a la estandarización y celeridad en el proceso.

3.6 Conclusiones del capítulo

La aplicación de técnicas de validación de requisitos permitió ratificar la correspondencia de los mismos con las solicitudes del cliente. Por su parte, la validación del diseño mediante las métricas TOC y RC permitió obtener, de forma general, el grado de complejidad de implementación y mantenimiento, de responsabilidad, reutilización, acoplamiento y la cantidad de pruebas necesarias para realizar a las clases, lo que trae consigo que las clases puedan reutilizarse, favoreciendo la implementación del sistema. Además, el método de caja negra permitió comprobar que las funciones son operativas a través de la interfaz del software, manteniendo así la integridad de la información externa. Por su parte, el método de caja blanca permitió comprobar internamente las funciones de los componentes, facilitando la detección de no conformidades para su corrección. Se realizó una valoración del comportamiento de las variables que forman parte del problema de la investigación, demostrando que los componentes desarrollados contribuyen a la celeridad de los procesos y la estandarización de la información.

CONCLUSIONES GENERALES

- La elaboración del marco teórico de la investigación mediante el estudio y el análisis de los principales referentes teóricos en los que se sustenta la investigación, facilitó la adquisición de los conocimientos necesarios para la solución propuesta.
- La especificación de los requisitos, así como el análisis y diseño de los componentes para gestión de los datos generales del expediente y de los intervinientes, permitieron obtener una aproximación para concebir los elementos necesarios de la implementación de los mismos.
- Con el desarrollo de los componentes para la gestión de los datos generales del expediente y de los intervinientes se logró contribuir a la estandarización y celeridad en la conformación de los expedientes en los Tribunales Populares Cubanos.
- Con la validación de las variables de la investigación y los resultados obtenidos a través de las métricas de validación del diseño y las pruebas de software se pudo comprobar de forma cuantitativa la calidad de los artefactos obtenidos durante el desarrollo de la solución propuesta.

RECOMENDACIONES

- Se recomienda la integración de los componentes de la presente investigación a los componentes de documento y notificaciones para lograr la total gestión de los datos que conforman al expediente.

BIBLIOGRAFÍA

Acosta, David L. 2017. Framework Design: A Role Modeling Approach. [En línea] 2017. http://sedici.unlp.edu.ar/handle/10915/19306.

Aleaga, Yaiset Moreno. 2018. CEGEL_XEJEL_Especificacion_de_requisitos_de_software. [En línea] 2018.

ALEGSA. 2018. Alegsa.com.ar. *Alegsa.com.ar.* [En línea] 2018. http://www.alegsa.com.ar.

Alvarez, Miguel Angel. 2012. DesarrolloWeb.com. *DesarrolloWeb.com.* [En línea] 2012. https://desarrolloweb.com/manuales/tutorial-sql.html.

Ambler, Scott W. 2018. Algunos lineamientos útiles para trabajar y seleccionar herramientas CASE de manera ágil. *Agile Module.* [En línea] 2018. http://www.agilemodeling.com/essays/simpleTools.htm#SelectingCASE.

Arango, Lopera y Antonio, Nelson. 2015. *El interviniente, Punibilidad y prinicpio de igualdad en el derecho Penal Colombiano.* Medellín : s.n., 2015.

Avila, José Herrera. 2012. issuu. [En línea] 2012. https://issuu.com/dr.marcomendozacorbetto/docs/id99-xv2_-_introducci__n_al_derecho.

BASE 100, S.A. 2019. Grupo Base 100 LEX 100. *BASE 100.* [En línea] 2019. https://www.base100.com/es/soluciones/lex100.html.

Boostrap Team. 2018. Bootstrap. *Bootstrap.* [En línea] 2018. https://getbootstrap.com/docs/4.3/getting-started/introduction/.

Bos, Bert. 2019. w3schools. *W3C.* [En línea] 2019. https://www.w3.org/Style/CSS/Overview.en.html.

Cano, José Hilario, Patricio Letelier Torres, Grupo ISSI. 2003. Metodologías ágiles en el desarrollo de software. [En línea] 12 de noviembre de 2003. http://issi.dsic.upv.es/archives/f-1069167248521/actas.pdf.

Center, IBM Knowledge. 2016. Patrón de diseÑo modelo-vista-controlador. [En línea] 2016. https://www.ibm.com/support/knowledgecenter/es/SSZLC2_8.0.0/com.ibm.commerce.developer.doc/concepts/csdmvcdespat.html.

Cillero, Manuel. 2017. Diagrama de clases del diseño. [En línea] 2017. https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-clases/.

—. 2017. Diagrama de despliegue. [En línea] 2017. https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-despliegue/.

—. 2017. Diagrama de secuencia. [En línea] 2017. https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-interaccion/diagrama-de-secuencia/.

—. 2017. mi circunstancia digital. [En línea] 2017. https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-componentes/.

—. 2019. Mi circunstancia digítal. *Mi circunstancia digítal.* [En línea] 2019. https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-componentes/.

Corte Suprema de Justicia. 2017. *Corte Suprema de Justicia de la República de Paraguay.* Asunción - Paraguay : s.n., 2017.

Dirección Nacional de Servicios Digitales. 2017. Argentina.gob.ar. [En línea] 2017. https://www.argentina.gob.ar/.

DISQUS. 2019. synfony.es. [En línea] 2019. https://symfony.es/noticias/2017/10/10/nuevo-en-symfony-34.

Doctrine Team. 2019. Doctrine 2 ORM. *Doctrine 2 ORM.* [En línea] 2019. https://www.doctrine-project.org/projects/doctrine-orm/en/2.6/index.html.

Domínguez, Dorado M. 2018. *Todo Programación. Nº 13. Págs. 32-34.* s.l. : Editorial Iberprensa (Madrid), 2018.

Durante Lerate, Rosa María, y otros. 2009. *Sistemas para el Control de Versiones.* España : s.n., 2009.

Escalante, Lain Cárdenas. 2014. *El patrón de arquitectura n-capas con orientación al dominio.* Universidad César Vallejo, Trujillo, Perú : s.n., 2014.

Gaines, Jeff, Boyd, Geraldine y Copley, Della. 2017. Visual Paradigm Online. *Visual Paradigm Online.* [En línea] 2017. https://online.visual-paradigm.com/es/features/.

Gamma, Erich, y otros. 1994. *"Design Patterns: Elements of Reusable Object Oriented Sofware".* s.l. : Grady Booch, 1994.

Gonzáles. 2008. *SISPROP Sistema para la tramitación de Procesos Penales.* 2008.

González Ochoa, Darián y Domínguez González, Yosviel. 2018. *SITPC, una herramienta informática cubana para la administración de la justicia.* La Habana : s.n., 2018.

Hoffman, Reid. 2019. LinKedln Corporation. *LinKedln Corporation.* [En línea] 2019. https://www.linkedin.com/learning/gitlab-esencial/que-es-gitlab.

Institute Puig Castelar. 2018. Pruebas de aplicaciones web. *Pruebas de aplicaciones web.* [En línea] 2018. https://elpuig.xeill.net/Members/vcarceler/asix-m09/uf1/nf1/a5.

justicia, Avanzando hacia una verdadera. 2016. *(Avanzando hacia una verdadera justicia, 2016).* 2016.

Larman, Craig. 2016. *UML y Patrones. Una introducción al análisis y diseño orientado a*

objetos y al proceso unificado. 3ra . s.l. : Prentice-Hall, 2016.

Microsoft. 2018. Typescript. *Typescript.* [En línea] 2018. https://www.typescriptlang.org/.

Ministerio de Justicia de España. 2016. *Informe de Modernización judicial en España.* Madrid : s.n., 2016.

MJU, Ministerio de Justiica de España. 2012. *EXPEDIENTE JUDICIAL ELECTRÓNICO Informes de Modernización Judicial en España.* España : Ministerio de Justicia, 2012.

Netcraft. 2018. Web Server Survey. *Web Server Survey.* [En línea] 2018. www.webServer.com.

Nube Colectiva. 2019. NC (Nube colectiva). *NC (Nube colectiva).* [En línea] 2019. http://blog.nubecolectiva.com/que-es-mvc-modelo-vista-controlador-y-otros-detalles/.

O.M.J. 2009. *Sistema Informático para Procedimientos Económicos.* 2009.

Ohmyroot. 2017. Estandares de codificación. *Estandares de codificación.* [En línea] 12 de enero de 2017. https://www.ohmyroot.com/buenas-practicas-legibilidad-del-codigo/.

Ossorio, Manuel. 2014. *Diccionario de Ciencias Jurídicas, Políticas y Sociales.* Guatemala : Datascan, 2014.

Oxford. 2018. Spanish Oxford Living Dictionaries. *Spanish Oxford Living Dictionaries.* [En línea] 2018. [Citado el: 13 de Septiembre de 2018.] https://es.oxforddictionaries.com/definicion/legajo.

Pacheco, Nacho. 2013. Documentación de Symfony. [En línea] 2013. http://gitnacho.github.io/symfony-docs-es/.

Page, Dave, Saito, Hiroshino y Yeatman, Mark. 2018. PgAdmin. *PgAdmin.* [En línea] 28 de Noviembre de 2018. http://www.pgadmin.org/.

Pérez Porto, Julian y Gardey, Ana. 2019. Definición. de. *Definición. de.* [En línea] 2019. [Citado el: 23 de Enero de 2019.] https://definicion.de/celeridad/.

—. 2019. Definición.de. *Definición.de.* [En línea] 2019. [Citado el: 23 de Enero de 2019.] https://definicion.de/estandarizacion/.

Pérez Valdés, Damián. 2007. maestrosdelweb. *maestrosdelweb.* [En línea] 3 de julio de 2007. http://www.maestrosdelweb.com/que-es-javascript/.

Pérez, Mariñan. 2013. Patrones de Diseño. [En línea] 2013. https://www.ecured.cu/Patrones_Gof.

Porto, Julián Pérez. 2010. *Definición de expediente (https://definicion.de/expediente/).* 2010.

Pratt, Michael . 2013. Pratт. *Pratт.* [En línea] 16 de abril de 2013. http://www.michael-pratt.com/blog/13/Patrones-de-Diseno-Inyeccion-de-Dependencias/.

Pressman, Roger S, Ph.D. 2010. *Ingenieria de Software. Un enfoque práctico. Septima Edición.* Mexico, D.F : The Me Graw Hill Companies, 2010.

Ramos Salavert, Isidro y Lozano Pérez, María Dolore. 2011. *Ingeniería del software y bases de datos: tendencias actuales.* 2011. págs. Entornos de Desarrollo Integrados, pág. 78».

Raviolo, Andrés, Ramírez, Paula y López, Eduardo. 2010. *Enseñanza y aprendizaje del concepto de modelo científico a través de analogías.* Argentina : s.n., 2010.

Reynoso, Carlos. 2011. Introducción a la Arquietectura de Software. *Introducción a la Arquietectura de Software.* [En línea] 2011. http://carlosreynoso.com.ar/archivos/arquitectura/Arquitectura-software.pdf.

Robles, Victor. 2018. ¿Que es Git y para que sirve? [En línea] 2018. https://victorroblesweb.es/2018/04/28/que-es-git-y-para-que-sirve/.

Sanchez, Ruby. 2018. Especificación de Requisitos Software según el estándar de IEEE 830. [En línea] 2 de octubre de 2018. https://www.academia.edu/6647065/Especificaci%C3%B3n_de_Requisitos_Software_seg%C3%BAn_el_est%C3%A1ndar_de_IEEE_830.

Sánchez, Tamara Rodríguez. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI. .* La Habana. Cuba : s.n., 2015.

Sequera , Santiago . 2019. Hispavista. [En línea] 2019. http://elrincondelainformaticauc.blogdiario.com/1465338887/informatica-juridica-de-gestion/.

Solutions, Axure Software. 2019. PURCHASE AXURE RP 8. [En línea] 2019. sales@axure.com.

Solvingadhoc. 2017. Solvingadhoc. *Solvingadhoc.* [En línea] diciembre de 2017. https://solvingadhoc.com/las-historias-usuario-funcion-agilidad/.

Sommerville, Ian. 2005. *Ingeniería de software. Séptima Edición.* Madrid España : PEARSON EDUCATION S.A, 2005.

Symfony Cloud. 2019. TWIG the flexible, fast and secure template for engine for php. [En línea] 2019. https://twig.symfony.com/doc/2.x/.

Tecnologías-Información. 2018. Modelo de datos. *Modelo de datos.* [En línea] 2018. https://www.tecnologias-informacion.com/modeladodatos.html.

The Apache Software Foundation. 2019. Apache HTTP Server Project. *Apache HTTP Server Project.* [En línea] 2019. https://httpd.apache.org/docs/2.4/es/.

The jQuery Foundation. 2019. Jquery write less do more. *Jquery write less do more.* [En

línea] 2019. https://blog.jquery.com/2018/01/19/jquery-3-3-0-a-fragrant-bouquet-of-deprecations-and-is-that-a-new-feature/.

The PHP Group. 2019. PHP. *PHP.* [En línea] 2019. https://www.php.net/docs.php.

The PostgreSQL Global Development Group. 2019. PostgreSQL. *PostgreSQL.* [En línea] 2019. https://www.postgresql.org/docs/10/release-10-1.html.

Web, Xinergia. 2017. FUDE. *FUDE.* [En línea] 2017. [Citado el: 13 de Septiembre de 2018.] https://www.educativo.net/articulos/que-es-un-expediente-judicial-796.html.

Wilson. 2018. AngularJS. [En línea] 2018. https://docs.angularjs.org/api.

Zoega, Geir Valle. 2010. *Mapeador de objetos realacionales ORM.* 2010.