**Universidad de las Ciencias Informáticas**

**Facultad 3**


# Componentes para la gestión de documentos y plantillas en la herramienta informática Expediente Judicial Electrónico

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

**Autor:**
Sergio Orlando Aguilera González

**Tutores:**
Ing. Reinier Fernández Coello
Ing. Yoannys Gustavo Dueñas Pérez

**La Habana, 2019**

# Declaración de Autoría

Declaro ser el único autor del presente trabajo de diploma y reconozco al Centro de Gobierno Electrónico de la Facultad 3 de la Universidad de las Ciencias Informáticas, los derechos patrimoniales del mismo, con carácter exclusivo, para que hagan el uso que estimen pertinente con el mismo.

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

_____

**Firma del autor**
Sergio Orlando Aguilera González

_____

**Firma del tutor**
Ing. Reinier Fernández Coello

_____

**Firma del tutor**
Ing. Yoannys Gustavo Dueñas Pérez

*Frase*

*"Whether you think you can or can't either way you* are *right"*

*Henry Ford*

# Agradecimientos

A mis padres que me han apoyado siempre, sin ellos no hubiera llegado a este momento los amo gracias por todo.

A mis abuelas, sé que me están apoyando desde donde quiera que estén.

A mi novia que me acompañó estos cinco años dándome apoyo y sin ella no sería quien soy hoy.

A mis compañeros de aula.

A mis amigos Alí, Erik, Orel, Yoel, Jose Carlos, Pablix.

A mis tutores Reinier y Yoannys gracias por todo el apoyo que me brindaron durante todo el año.

A la profesora Dariela por su apoyo en estos 5 años.

A los profesores que me impartieron clases en estos 5 años.

# Dedicatoria

*A mis padres por ser incansables en mi educación, este trabajo es dedicado a ellos.*

# Resumen

Actualmente, entre las proyecciones estratégicas del Tribunal Supremo Popular de Cuba se encuentra la informatización de los procesos que se llevan a cabo en los Tribunales Populares Cubanos en sus diferentes instancias. Con el objetivo de contribuir a estas proyecciones, se creó un proyecto de cooperación entre dicha entidad y el Centro de Gobierno Electrónico de la Universidad de las Ciencias Informáticas, surgiendo así la herramienta informática Expediente Judicial Electrónico, la cual contempla la informatización del proceso de creación de los expedientes judiciales. La misma contribuye a agilizar el proceder con los expedientes que intervienen en los procesos judiciales. En base a lo anteriormente descrito, la presente investigación tiene como objetivo desarrollar los componentes para la gestión de documentos y plantillas en dicha herramienta, de forma tal que se contribuya a la disponibilidad y estandarización en el proceso de confección de la documentación de los expedientes en los Tribunales Populares Cubanos. Para lograr este objetivo, se utilizó como guía la metodología de desarrollo el Proceso Unificado Ágil, versión establecida por la Universidad de las Ciencias Informáticas, así como un conjunto de lenguajes y herramientas, permitiendo de esta forma el desarrollo de la presente investigación. Los resultados fueron validados a través de métricas y pruebas de software, donde se pudo comprobar de forma cuantitativa la calidad de los artefactos obtenidos, así como las funcionalidades de los componentes propuestos.

**Palabras claves:** disponibilidad, documentos, estandarización, expedientes judiciales electrónicos y plantillas.

# ÍNDICE DE CONTENIDO

# ÍNDICE DE IMÁGENES

# ÍNDICE DE TABLAS

# Introducción

Las Tecnologías de la Información y las Comunicaciones (TIC) han permitido llevar la globalidad al mundo de la comunicación, facilitando la interconexión entre las personas, así como entre instituciones a nivel mundial, eliminando barreras espaciales y temporales. Cuba no está exenta de este desarrollo, por lo que se encuentra en un proceso de informatización de todo el país, ejemplo de esto es la creación de la Universidad de las Ciencias Informáticas (UCI), la cual está estructurada por varios centros de desarrollo, como es el caso del Centro de Gobierno Electrónico (CEGEL) adscrito a la Facultad 3, que satisface necesidades de clientes gubernamentales, mediante el desarrollo de productos, servicios y soluciones integrales de alta confiabilidad, calidad, competitividad, fidelidad y eficiencia, uno de estos clientes es el Tribunal Supremo Popular (TSP).

Actualmente, entre las proyecciones estratégicas del TSP de la República de Cuba se encuentra la informatización de los procesos que se llevan a cabo en los Tribunales Populares Cubanos (TPC) en sus diferentes instancias, desde la municipal hasta llegar a la instancia suprema. Con el objetivo de contribuir a esta estrategia, se creó un proyecto de cooperación entre el TSP y el CEGEL. Producto de esta colaboración surge así el Expediente Judicial Electrónico (XEJEL) como proyecto, el cual contempla la digitalización de los expedientes judiciales. El mismo surge debido a que hoy en día, llevar el control de los expedientes judiciales en los TPC, es la tarea en la que los especialistas judiciales invierten la mayor parte del tiempo, debido a la documentación de los procesos judiciales que se llevan a cabo. Donde, con cada uno de ellos se confecciona un expediente, legajo o carpeta en el que se asientan, por escrito, las actuaciones o documentos de los actos procesales, de forma cronológica, con los folios enumerados y la carátula, que incluye la información elemental para garantizar su individualización. La clave para mantener el control de un expediente es precisamente el orden cronológico en que se anexan los documentos y su correspondiente foliación. Por tanto, el expediente judicial constituye el registro de la historia de un proceso judicial, todo lo que las partes presentan y lo que el tribunal dispone para resolver un conflicto. Actualmente la forma de incluir un documento a un expediente es cosiendo el borde izquierdo del expediente en partes equidistantes entre sí, con hilo y grandes agujas.

A medida que transcurren los procesos judiciales en los TPC resulta más complicado crear los documentos asociados a cada proceso en cuestión, de esta forma la generación de los mismo se realiza manual, y en muchas ocasiones no conservando las formas, estructuras y estándares para su creación dictados por la Ley del Procedimiento Penal, para el caso de la materia Penal, y la Ley del Procedimiento Civil, Administrativo, Laboral y Económica, para el resto de las materias.

Estos documentos se rigen según las políticas de los TPC y son clasificados como: escritos, autos, sentencias, providencias, oficios, diligencias, citaciones, comunicaciones, actas, mandamientos, informes y notificaciones. Por cada una de estas clasificaciones existen varias sub-clasificaciones, donde se identifican, por ejemplo, un total de 94 tipos de escritos, por lo que considerando el resto de las sub-clasificaciones, se tiene concebido un total de 705 documentos en general hasta la fecha.

Cada uno de estos documentos, según la clasificación, debe poseer una plantilla base que guíe la estructura y confección de cada uno. Debido a las disímiles interpretaciones de la ley a lo largo de todo el país, no existe una estandarización para cada uno de los documentos de una misma clasificación.

Por otra parte, la captación de los datos primarios de cada documento en el lugar de origen acarrea a que estén expuestos a la introducción de errores de repetición en el número de los asientos, tachaduras, saltos en los espacios de las anotaciones y borrones. En ocasiones existen inconsistencias de la información entre documentos de un mismo caso, por ejemplo: números de expedientes y datos de los involucrados, e incluso en algunos de los casos, no coinciden los documentos que pertenecen a una misma tipología según las políticas trazadas por los TPC.

Esta situación se ve agudizada al almacenar los documentos de los expedientes en estantes, donde su búsqueda resulta agotadora, además del deterioro o pérdida al que se encuentran expuestos; lo que ocasiona dificultad en el control de la información y por lo tanto atraso en el trabajo. Por otra parte, la consulta manual de los documentos de un expediente en el tribunal judicial, desde cualquier parte del territorio nacional, es otra de las dificultades que actualmente afrontan los TPC, ya que son objetos tangibles y requieren de un procedimiento bien cuidadoso y lento para hacer posible moverlos a la instancia que se requiera o hacerles adiciones o modificaciones a los mismos.

Ante la problemática anteriormente planteada surge el siguiente **problema a resolver**: ¿Cómo gestionar la documentación de los expedientes judiciales en los Tribunales Populares Cubanos, de forma tal que se contribuya a la disponibilidad y estandarización en el proceso de confección de estos?

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 documentos y plantillas en la herramienta informática Expediente Judicial Electrónico, de forma tal que se contribuya a la disponibilidad y estandarización en el proceso de confección de estos.

Para dar cumplimiento al objetivo propuesto se han derivado un conjunto de **objetivos específicos**, orientados fundamentalmente a proveer los elementos necesarios para la implementación de la solución, los cuales se listan a continuación:

- Identificar los referentes teóricos en los que se sustenta la propuesta de solución sobre el desarrollo de componentes informáticos dirigidos a la gestión de la documentación en expedientes judiciales electrónicos.
- Realizar la especificación de los requisitos, así como el análisis y diseño de los componentes de documentos y plantillas como aproximación a la implementación.

- Implementar los componentes de documentos y plantillas para contribuir a la disponibilidad y estandarización en el proceso de confección de estos.
- Validar la solución propuesta mediante métricas y pruebas de software para garantizar la calidad de los componentes propuestos.
- Validar las variables de la investigación para comprobar cómo la propuesta de solución contribuye a la disponibilidad y estandarización de la documentación en el XEJEL.

Teniendo como referencia el problema a resolver y el objetivo general se plantea la siguiente **idea a defender**: si se desarrollan los componentes para la gestión de documentos y plantillas en la herramienta informática Expediente Judicial Electrónico, se contribuirá a la disponibilidad y estandarización en el proceso de confección de estos.

Para ello se identifica como **campo de acción:** la gestión de la documentación en los expedientes judiciales electrónicos.

Para dar cumplimiento a los objetivos propuestos se definen las siguientes **tareas de la investigación**:

- Definición de los principales conceptos relacionados con el objeto de estudio y campo de acción.
- Descripción del proceso de desarrollo de software y su metodología.
- Caracterización de las tecnologías y las herramientas a utilizar para el desarrollo de la solución.
- Especificación de los requisitos funcionales y no funcionales de la solución.
- Descripción de la arquitectura de software base para la implementación del sistema.
- Caracterización y selección de los patrones de diseño más factibles para la propuesta de solución.
- Realización del diagrama de clases de diseño, modelo de datos y diseño de casos de pruebas.
- Implementación de la propuesta de solución.
- Ejecución de las pruebas de software para evaluar la calidad de la implementación.
- Solución de no conformidades al finalizar cada iteración de las pruebas.
- Validación de las variables de la investigación.

**Posibles resultados:** una vez finalizado el presente trabajo, se tendrá como resultado los componentes para la gestión de documentos y plantillas en la herramienta informática Expediente Judicial Electrónico.

Para dar solución a las tareas antes propuestas se definen los métodos científicos, clasificados en teóricos y empíricos. De los cuales se emplearon:

- **Métodos teóricos:**
  - ✓ **Histórico-lógico:** se empleó para el estudio sobre la trayectoria y evolución de la herramienta informática Expediente Judicial Electrónico y así obtener un conocimiento previo sobre las leyes generales del funcionamiento y desarrollo en el proceder jurídico en los TPC para el caso de los documentos y plantillas.
  - ✓ **Analítico-sintético:** se empleó para el análisis del negocio en general, dividiéndolo en pequeñas partes para facilitar su estudio, y para determinar los elementos más importantes del negocio.
  - ✓ **Modelación:** se utilizó para la realización de los diagramas necesarios en el proceso de desarrollo de software, haciendo una representación abstracta de la solución, facilitando así el desarrollo de la misma.
- **Métodos empíricos:**
  - ✓ **Entrevista:** permitió el intercambio verbal con el cliente para obtener la mayor cantidad de información posible, entender todo 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.
  - ✓ **Encuesta:** permitió mediante un cuestionario pre-elaborado, obtener información sobre el tiempo real, recursos humanos y materiales que utilizan los especialistas funcionales jurídicos para realizar todos los procesos de los documentos y las plantillas en un expediente al estar sujeto a un trámite dentro de lo reglamentado por la ley.

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

**Capítulo 1**: Fundamentación teórica:
En este capítulo se abordan los conceptos necesarios para entender el objetivo fundamental del trabajo, se realiza un estudio de soluciones similares, así como la metodología de desarrollo de software a utilizar y las herramientas para el desarrollo de la solución propuesta. De igual forma se caracterizan los lenguajes de programación del lado del cliente y del lado del servidor a utilizar.

**Capítulo 2**: Descripción de la propuesta de solución:
En este capítulo se presenta una descripción general de los componentes para la gestión de la documentación en el XEJEL. Para ello se lleva a cabo la disciplina de Requisitos, en la cual se engloban los requisitos funcionales y no funcionales de la propuesta de solución. De igual forma se obtienen los artefactos correspondientes a la disciplina de Análisis y diseño, aplicando los patrones de diseño definidos como buenas prácticas durante el ciclo de desarrollo del software, los diagramas de clases de diseño, el modelo de datos y el patrón arquitectónico que da soporte a la propuesta de solución. Por último, se relacionan los estándares de codificación como base a la implementación de los componentes propuestos.

**Capítulo 3**: Validación de la propuesta de solución:

En este capítulo se evalúa el grado de calidad y fiabilidad de los resultados obtenidos en el desarrollo de los componentes para la gestión de la documentación en el XEJEL. 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. Por último, se aplican las disciplinas de pruebas internas y pruebas de liberación que propone la metodología que guía el desarrollo de la solución propuesta, con el fin de verificar y revelar la calidad del producto antes de ser entregado al cliente.

# Capítulo 1: Fundamentación teórica

En el presente capítulo se abordan los conceptos necesarios para entender el objetivo fundamental del trabajo. De igual forma se realiza un estudio de soluciones similares, así como la metodología de desarrollo de software a utilizar y las herramientas para el desarrollo de la solución propuesta. Además, se definen los lenguajes de programación del lado del cliente y del lado del servidor a utilizar.

## 1.1. Conceptos asociados al objeto de estudio

Con el objetivo de lograr un entendimiento del objeto de estudio de la presente investigación, se requiere el conocimiento de los conceptos enunciados a continuación:

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

**Documento electrónico:** Información de cualquier naturaleza en forma electrónica, archivada en un soporte electrónico según un formato determinado y susceptible de identificación y tratamiento diferenciado (Franco Espiño, y otros, 2015).

**Plantilla:** Expresión tipificada de unidades documentales con unas características estructurales, en general homogéneas, de actuaciones únicas o secuenciales, normalmente regulada por una norma de procedimiento, derivada del ejercicio de una misma función y realizadas por un determinado órgano, unidad o persona con competencia para ello (Illescas, 2016).

**Estandarización:** Se denomina estandarización al acto y el resultado de estandarizar: ajustar a un estándar. La estandarización, por lo tanto, implica concertar algo para que resulte coincidente o concordante con un modelo, un patrón o una referencia (Pérez Porto, y otros, 2018).
En la presente investigación se entiende por estandarización de los documentos, al proceso de igualar todos los documentos de un expediente que responden a una misma clasificación, de forma tal que sigan el mismo formato en cuanto al estilo de redacción de los mismos, ya sea teniendo en cuenta aspectos tipográficos como elementos a considerar al concebir cada una de las secciones de los mismos.

**Disponibilidad:** Propiedad o cualidad de los activos de información, mediante la cual aquellas entidades o procesos autorizados tienen acceso a los mismos cuando lo requieren. En el ámbito específico de los documentos electrónicos, aquella propiedad o cualidad de los mismos que permite su localización, recuperación, presentación e interpretación (Franco Espiño, y otros, 2015).
En el caso particular de la presente investigación se tendrá en cuenta la disponibilidad de los documentos en el XEJEL, como la propiedad que permita la localización y recuperación del mismo a cualquier instancia de los TPC, así como de forma concurrente, lo que favorece la búsqueda y

consulta de estos por parte de los especialistas judiciales que intervienen en el proceso que se lleva a cabo en cada expediente.

## 1.2. Estudio de sistemas similares

Luego de la investigación de los principales conceptos, se realiza un estudio de los sistemas similares tanto en Cuba como en el mundo para definir ventajas y desventajas de los mismos, así como la viabilidad de su uso en la solución de la problemática existente.

### 1.2.1. Sistemas similares en el mundo

**Sistema de gestión de notificaciones telemáticas (Lexnet):** Este software, desarrollado en España, se basa en un sistema de correo electrónico seguro con firma electrónica, a través del cual el usuario recibe las notificaciones emitidas por el juzgado y presenta los escritos por vía telemática. Posteriormente, y a través del mismo, recibe un resguardo electrónico en el que se acredita que la transmisión se ha efectuado correctamente y se le comunica la fecha efectiva de la presentación de dicho escrito en la oficina judicial o juzgado correspondiente. Su uso se regula en el Real Decreto 84/2007, de 26 de enero, sobre implantación en la administración de justicia del sistema informático de telecomunicaciones Lexnet para la presentación de escritos y documentos, el traslado de copias y la realización de actos de comunicación procesal por medios telemáticos. Dentro del marco funcional que comprende la realización de actos de comunicación, la presentación de escritos con traslado de copias y de escritos iniciadores, el sistema Lexnet posee las siguientes funcionalidades: presentación de documentos y adjuntos al órgano judicial, envío de notificaciones por parte del órgano judicial, emisión de acuses de recibo, acceso autenticado y seguro mediante navegador web, acceso e intercambio de documentos mediante servicios web, gestión de carpetas de usuario, búsqueda y traza de mensajes (IT-CGAE, 2013).

Este sistema se basa solo en la gestión de notificaciones y documentos, funcionalidades necesarias en la solución a implementar, pero que por sí solas no resuelven los problemas de los TPC, sin embargo, aportan ideas a la solución como el intercambio de información mediante servicios web.

**Sistema para la gestión jurídica (Lex-Doctor):** Este completo sistema es utilizado en la amplia mayoría de los estudios jurídicos y asesorías letradas informatizados de la Argentina desarrollado por Sistemas Jurídicos SRL, empresa que tiene un importante posicionamiento en el segmento de los sistemas aplicados a la actividad jurídica en Latinoamérica. Su tecnología de administración de datos, permite manejar grandes volúmenes de información, y organizar grupos de trabajo operando en redes de gran cantidad de terminales. Entre sus principales funcionalidades se encuentran:

- Gestión de expedientes: Procesos judiciales, extrajudiciales, mediaciones y todo tipo de expediente de índole jurídico. Partes, letrados, agendas, pruebas, gestiones, movimientos, audiencias, vencimientos. Manejo total de la procuración.
- Gestión económica: Cuentas por cliente, por expediente, y otros tipos de cuenta definibles por el usuario. Actualizaciones y liquidaciones, con conversión de monedas. Liquidaciones individuales o masivas.

- Gestión documental: Confección automatizada de escritos, cédulas, oficios, mandamientos, telegramas, cartas, y otros documentos, con posibilidad de confección seriada. Almacenamiento de documentos creados por el sistema y por otros programas instalados en la PC.
- Reportes y estadísticas: Confección de listados e informes, con diseños programados o propios, y con posibilidad de exportar a distintos formatos. Evaluación estadística gráfica.
- Acceso remoto: Opcionalmente, permite operar a través de Internet; así, disponiendo de una conexión y una PC, se puede trabajar en línea con la oficina desde cualquier lugar (SRL, 2011).

Lex-Doctor a pesar de ser un sistema con mucho prestigio y aceptación presenta diferentes inconvenientes para su utilización por parte de los TPC, puesto que el mismo está desarrollado sobre software propietario, impidiendo las posibilidades de modificación y adaptación al medio nacional, además centra su trabajo sobre la gestión documental fundamentalmente, y para su utilización es necesario comprar la licencia de uso.

**Expediente Judicial Electrónico en España (EJE):** El expediente judicial electrónico 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 expediente judicial electrónico 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. Además, también favorece la comunicación entre órganos jurisdiccionales (MJU, 2012).

Este sistema, a pesar de tener grandes ventajas, no se ajusta a las leyes y códigos de nuestro país, además de tener un proceso de digitalización, el cual no es objetivo de los TPC en un futuro inmediato.

### 1.2.2. Sistemas similares en Cuba

Cuba no está ajena al avance de las tecnologías, en este sentido se han hecho intentos de informatizar las materias Penal y Económico con tres soluciones informáticas que no han cumplido las expectativas para las que fueron creadas.

**Sistema para la tramitación de procesos penales (SisProp)**: Sistema informático desarrollado en la provincia de Villa Clara en el 2008. La propuesta inicial fue que abarcara la instancia suprema y provincial de la materia penal, desarrollándose solamente la tramitación de los procesos en la última instancia. Facilita la tramitación de los procesos penales, con agilidad y precisión, en las entidades del sistema de tribunales populares del país. Dos de las características fundamentales del sistema son su seguridad dada la naturaleza de la información que se maneja, y la flexibilidad con respecto a cualquier modificación que pudiera sufrir la Ley Procesal Penal. Este software presenta algunas deficiencias como son:

- No capta ningún dato de la fase judicial de la tramitación y decisión del tribunal.
- No aporta estadística, ni información alguna.
- No posee una exhaustiva validación de los datos.
- No se trabajó con un NIP[1] de cada proceso.
- Programado en Delphi, corre sobre SQL Server por lo que no es compatible con el software libre (Castro Morell, y otros, 2008).

Este sistema no cumple con las expectativas para las que fue creado, no es posible integrarlo a la herramienta informática Expediente Judicial Electrónico debido a que las tecnologías de desarrollo no son compatibles con las definidas para dicha herramienta, ya que estas son más avanzadas, además no satisface las necesidades de los TPC en la actualidad.

**Sistema informático (SisEco):** desarrollado en Ciudad de la Habana en el 2002, concebido para el área de la estadística en el procedimiento Económico. El sistema cuenta con funcionalidades muy básicas y es lento en cuanto a tiempo de respuesta. En él se insertan los documentos radicados manualmente y las salvas diariamente se guardan en disquetes. 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. Como principales deficiencias se tiene:

- Inserta documentos radicados manualmente. La secretaria de estadística recoge el libro LRE e inserta los datos en la aplicación y cada cinco años borra los datos de los años anteriores quedando solamente la información asentada en los libros.
- Las salvas se guardan en disquetes (Juiz, 2009).

Al igual que la anterior, esta aplicación no cumplió las expectativas iniciales de su creación. Es lento, característica que no va aparejada con la estadística en tiempo real. La informatización es mínima ya que la radicación se realiza de forma manual. Las salvas se guardan en disquetes, técnica que ha quedado obsoleta en la actualidad, la información almacenada es borrada cada cinco años, algo incomprensible para un sistema judicial para el cual mantener registros es primordial, ya que la información es el mecanismo principal para la justicia. Teniendo en cuenta lo antes explicado se llega a la conclusión de que tampoco es posible integrarlo a la herramienta informática Expediente Judicial Electrónico.

**Sistema de Información para la Gestión de los Tribunales Populares Cubanos (SITPC):** Este sistema, basado en tecnología web, posibilita entre otras funcionalidades generales, la presentación telemática de escritos, la radicación automática de expedientes, el procesamiento de textos con modelos de documentos, y las notificaciones electrónicas a las partes; todo desde la perspectiva del expediente digital (González Ochoa, y otros, 2018).

SITPC en su totalidad está compuesto por 46 módulos, de los cuales se encuentran implementados 7, de la materia Administrativa, el Administrativo, de la materia Económico, el Ejecutivo, de la

[1] Número de identificación permanente o general.

materia Penal, el Ordinario de la instancia provincial y municipal, de la materia Civil, Actos preparatorios y el Ordinario y de la materia Laboral, Disciplina y Derecho Laboral, por lo que el resto de los módulos que responden a los procesos judiciales que se atienden en los TPC aún llevan a cabo el proceso de forma manual, además no satisface las necesidades actuales de los TPC, ya que una parte del expediente se sigue tramitando de forma manual acarreando los problemas que dieron paso a la investigación. Por lo tanto, se concluye que tampoco cumple las necesidades actuales de los TPC.

### 1.3. Metodología de desarrollo de software

Las metodologías de desarrollo de software tienen como objetivo presentar un conjunto de técnicas tradicionales y modernas de modelado de sistemas que permitan desarrollar software de calidad, incluyendo heurísticas de construcción y criterios de comparación de modelos de sistemas (Baltazar, 2012).

Para el desarrollo de esta solución se aplicó la metodología definida por la UCI, la cual es una variación del Proceso Unificado Ágil (AUP) ya que esta se adapta al ciclo de vida productivo de dicha universidad.

AUP es una versión simplificada de Rational Unified Process (RUP). Este define un flujo de trabajo con disciplinas diferentes a las de RUP, aunque conserva sus fases en cada una. La disciplina de modelación incluye la modelación del negocio, requisitos, análisis y diseño. Por otra parte, se integran además la Gestión de Cambios y Gestión de Configuración en una sola disciplina.

AUP-UCI se crea porque, no existía una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Por lo que 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 (Rodríguez Sánchez, 2015).

Entre las principales diferencias presentes en esta versión se encuentran:

- AUP define 4 fases de desarrollo (Inicio, Elaboración, Construcción, Transición), y su versión para la UCI mantiene la fase de Inicio, pero se modifica el objetivo, agrupa las otras 3 fases en una llamada Ejecución e incorpora una llamada Cierre.
- AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), la metodología definida en la UCI tiene 8 disciplinas.
- Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran cada uno de ellos disciplinas. Específicamente en la disciplina Requisitos plantean 4 posibles escenarios para la identificación y descripción de requisitos. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación y la disciplina Despliegue se considera opcional (Rodríguez Sánchez, 2015).

En esta variante se definen 4 escenarios:

1. Proyectos que modelen el negocio con CUN (Caso de Uso del Negocio) solo pueden modelar el sistema con CUS (Casos de Uso del Sistema).
2. Proyectos que modelen el negocio con MC (Modelo Conceptual) solo pueden modelar el sistema con CUS (Casos de Uso del Sistema).
3. Proyectos que modelen el negocio con DPN (Descripción de Proceso de Negocio) solo pueden modelar el sistema con DRP (Descripción de Requisitos por Proceso).
4. Proyectos que no modelen negocio solo pueden modelar el sistema con HU (Historias de usuario) (Rodríguez Sánchez, 2015).

Una vez evaluado el negocio a informatizar, se obtuvo un negocio muy bien definido y se concibió que la propuesta no modela negocio ya que solo se centraría en llevar el expediente de forma electrónica, obviando como tal los procesos judiciales a los que se adscriben cada una de las materias de los TPC. Por otra parte, el cliente estará acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos, además de ser un proyecto no muy extenso, por lo que se puede modelar el sistema a través de las HU utilizando el escenario 4 que define AUP-UCI.

## 1.4. Herramientas y lenguajes

En el presente epígrafe se caracterizan las herramientas y lenguajes utilizados para el desarrollo de la propuesta de solución:

### 1.4.1. Visual Paradigm

Es una herramienta CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) profesional que soporta el desarrollo de software desde el análisis y diseño orientado a objeto, construcción, pruebas y despliegue. Entre muchas de sus ventajas permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Además, permite la importación y exportación de archivos XML de diferentes versiones. Presenta licencia gratuita y comercial. Es multiplataforma, fácil de instalar y actualizar, compatible entre ediciones.

Entre sus principales características se encuentran:

- Soporte de UML versión 2.1.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Modelado colaborativo con CVS y Subversion (control de versiones).
- Ingeniería de ida y vuelta.
- Ingeniería inversa - Código a modelo, código a diagrama.
- Generación de código - Modelo a código, diagrama a código.
- Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.

- Diagramas de flujo de datos.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- Generador de informes.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML (Visual Paradigm, 2017).

Para la realización del XEJEL, se decide utilizar la herramienta Visual Paradigm for UML en su versión 8.0, debido a la fácil integración de esta herramienta con el lenguaje PHP y a la experiencia del equipo de desarrollo en su empleo, lo que posibilita reducir el tiempo en el proceso de desarrollo del software, ya que no hay necesidad de capacitar a los desarrolladores. Además, la UCI posee una licencia académica para su uso.

### 1.4.2. Angular

Angular es un marco de trabajo que facilita la creación de aplicaciones web, desarrollado en TypeScript, de código abierto, el cual combina plantillas declarativas, inyección de dependencia, herramientas de extremo a extremo y mejores prácticas integradas para resolver los desafíos de desarrollo web. Los bloques de construcción básicos de una aplicación Angular son NgModules, que proporcionan un contexto de compilación para los componentes. NgModules recopila códigos relacionados en conjuntos funcionales; una aplicación Angular se define por un conjunto de NgModules.

Una aplicación siempre tiene al menos un módulo raíz que habilita el arranque, y normalmente tiene muchos más módulos de características.

- Los componentes definen vistas, que son conjuntos de elementos de pantalla que Angular puede elegir y modificar según la lógica y los datos de su programa.
- Los componentes utilizan servicios, que proporcionan una funcionalidad específica que no está directamente relacionada con las vistas. Los proveedores de servicios se pueden inyectar en los componentes como dependencias, haciendo que su código sea modular, reutilizable y eficiente.
- Tanto los componentes como los servicios son simples clases, con decoradores que marcan su tipo y proporcionan metadatos que le indican a Angular cómo usarlos.
- Los metadatos de una clase de componente lo asocian con una plantilla que define una vista. Una plantilla combina HTML ordinario con directivas angulares y un marcado de enlace que permite a Angular modificar el HTML antes de representarlo.
- Los metadatos para una clase de servicio proporcionan la información que Angular necesita para que esté disponible para los componentes a través de la inyección de dependencia (DI).
- Los componentes de una aplicación típicamente definen muchas vistas, ordenadas jerárquicamente. Angular proporciona el servicio de enrutador para ayudarlo a definir rutas de

navegación entre vistas. El enrutador proporciona sofisticadas capacidades de navegación en el navegador (Angular, 2018).

Para el desarrollo del XEJEL se decide, por parte del equipo de arquitectura, utilizar Angular en su versión 6.0 y, por ende, es la utilizada para el desarrollo de la solución propuesta.

### 1.4.3. TypeScript

TypeScript es un lenguaje de programación de alto nivel que implementa muchos de los mecanismos más habituales de la programación orientada a objetos, pudiendo extraer grandes beneficios que serán especialmente deseables en aplicaciones grandes, capaces de escalar correctamente durante todo su tiempo de mantenimiento. La característica fundamental de *TypeScript* es que compila en *Javascript* nativo, por lo que se puede usar en todo proyecto donde se esté usando *Javascript*. Dicho con otras palabras, cuando se usa *TypeScript* en algún momento se realiza su compilación, convirtiendo su código a *Javascript* común. El navegador, o cualquier otra plataforma donde se ejecuta *Javascript*, nunca llegará a enterarse que el código original estaba escrito en *TypeScript*, porque lo único que llegará a ejecutar es el *Javascript* resultante de la compilación (Guerra, 2016).

Para el desarrollo del XEJEL se decide la utilización de 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.

### 1.4.4. JQuery

JQuery es una librería de JavaScript. Esta librería de código abierto, simplifica la tarea de programar en JavaScript y permite agregar interactividad a un sitio web sin tener conocimientos del lenguaje. Basados en esta librería**,** existe una infinita cantidad de *plugins* creados por desarrolladores de todo el mundo**.** Estos *plugins* resuelven situaciones concretas dentro del maquetado de un sitio, por ejemplo: un menú responsive, una galería de fotos, un carrusel de imágenes, un *slide*[2], un *header*[3] que cambia de tamaño, el deslizamiento del *scroll*[4] al hacer clic en un botón (anclas HTML), la transición entre páginas y miles de efectos más (Chuburu, 2018).

Para el desarrollo del XEJEL se decide la utilización de JQuery en su versión 3.3, ya que es la seleccionada por el equipo de arquitectura del proyecto debido a las nuevas funcionalidades que trae definidas, por ejemplo, el cálculo para asignar el alto y el ancho en el diseño de la página web se hace más preciso evitando el redondeo. Además, se introduce una manera sencilla de mostrar datos con el bucle (*for*), facilitando la programación de las instrucciones en los métodos.

[2] Diapositiva

[3] Encabezamiento

[4] Desplazamiento

### 1.4.5. Boostrap

Bootstrap es un framework CSS[5] desarrollado inicialmente (en el año 2011) por Twitter que permite dar forma a un sitio web mediante librerías CSS que incluyen tipografías, botones, cuadros, menús y otros elementos que pueden ser utilizados en cualquier sitio web. Es una excelente herramienta para crear interfaces de usuario limpias y totalmente adaptables a todo tipo de dispositivos y pantallas, sea cual sea su tamaño. Además, ofrece las herramientas necesarias para crear cualquier tipo de sitio web utilizando los estilos y elementos de sus librerías. Desde la aparición de Bootstrap 3 el framework se ha vuelto más compatible con desarrollo web responsive, entre otras características se han reforzado las siguientes:

- Soporte bastante bueno (casi completo) con HTML5[6] y CSS3[7], permitiendo ser usado de forma muy flexible para desarrollo web con unos excelentes resultados.
- Se ha añadido un sistema GRID[8] que permite diseñar tablas de 12 columnas donde se debe plasmar el contenido.
- Boostrap 3 establece Media Queries[9] para 4 tamaños de dispositivos diferentes dependiendo del tamaño de su pantalla, estas Media Queries permiten desarrollar para dispositivos móviles y tablets de forma mucho más fácil.
- Boostrap 3 también permite insertar imágenes responsive, es decir, con solo insertar la imagen con la clase “img-responsive” las imágenes se adaptarán al tamaño (3schools.com, 2019).

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: brinda la posibilidad de asignarle una altura y ancho a los elementos de la página web de forma automática por lo que se adaptará a todos los dispositivos. Además, se hace un mejor uso de las transiciones, animaciones y efectos en las páginas web. También incorpora el tratamiento de los *tooltips*, de forma tal que, al colocar el puntero sobre un elemento, muestra de una manera diferente el texto o título correspondiente. Por último, proporciona una galería de iconos de forma gratuita, lo que facilita incorporar cada uno de estos elementos en el XEJEL.

### 1.4.6. JavaScript

JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Su función es ampliar las funcionalidades de HTML, permitiendo interactuar con el navegador de manera dinámica y eficaz, proporcionando a las páginas web dinamismo y vida. JavaScript es soportado por la

[5] Hojas de estilo en cascada (CSS por sus siglas en inglés Cascade)
[6] Sigla en inglés de HyperText Markup Language (Lenguaje de Marcas de Hipertexto).
[7] Siglas en inglés de Casacade Style Sheet (Estilo de hojas en cascada)
[8] Sistema de cuadrícula de 12 columnas para crear objetos tablas en una página web.
[9] Módulo CSS3 que permite adaptar la representación del contenido a características del dispositivo

mayoría de los navegadores como Internet Explorer, Mozilla Firefox, Netscape, Opera (Pérez Valdés, 2007).

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.

### 1.4.7. HTML

HTML es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet. Se trata de la sigla que corresponde a Lenguaje de Marcas de Hipertexto por sus siglas en inglés, HyperText Markup Language, que podría ser traducido como Lenguaje de Formato de Documentos para Hipertexto.

Es un lenguaje muy simple y general que sirve para definir otros lenguajes que tienen que ver con el formato de los documentos. Permite ciertos códigos que se conocen como scripts, los cuales brindan instrucciones específicas a los navegadores que se encargan de procesar el lenguaje. Entre los scripts que pueden agregarse, los más conocidos y utilizados son JavaScript y PHP[10] (Berciano Alonso, 1999).

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.

### 1.4.8. CSS

Las Hojas de estilos en cascada (CSS por sus siglas en inglés de Cascade Style Sheet) es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes (Lázaro, 2010).

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.

### 1.4.9. Symfony

Symfony es un marco de trabajo diseñado para optimizar el desarrollo de las aplicaciones web, gracias a sus características. Para empezar, 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).

Symfony ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server y se puede ejecutar tanto en plataformas *nix

[10] *Hypertext Preprocessor*

(Unix, Linux, etc.) como en plataformas Windows. A continuación, se muestran algunas de sus ventajas:

- Fácil de instalar y configurar en la mayoría de las plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales, y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Fácil de extender, lo que permite su integración con bibliotecas desarrolladas por terceros (Pacheco, 2013).

Para el desarrollo del XEJEL se decide la utilización de Symfony en su versión 3.4 debido a las nuevas funcionalidades que el mismo integra, por ejemplo: posee la facilidad de gestionar los localizadores uniformes de recursos (URL por sus siglas en inglés) inteligentes, permitiendo configurar direcciones amigables en las páginas de la aplicación. El uso de controladores que gestionan estas rutas es ideal para crear aplicaciones web a medida complejas de forma muy ordenada y eficiente. Además, permite su integración con las bibliotecas de otros fabricantes y así aprovechar las ventajas en la parte del *front-end* de nuevas librerías tanto en JavaScript tipo Angular, jQuery, entre otras. Igualmente, como cualquier pre-procesador CSS y todas las nuevas API de HTML5. Por último, es la versión LTS[11] más actual, lo que facilita incorporar cada uno de estos elementos en el XEJEL.

### 1.4.10. PHP

Procesador de Hipertextos (PHP por sus siglas en inglés de *Hypertext Preprocessor*) es un lenguaje interpretado en el lado del servidor que se caracteriza por su potencia, versatilidad, robustez y modularidad. Los programas escritos en PHP son embebidos directamente en el código HTML y ejecutados por el servidor Web a través de un intérprete antes de transferir al cliente que lo ha solicitado un resultado en forma de código HTML puro. Al ser un lenguaje que sigue las corrientes de código abierto, son totalmente accesibles de forma gratuita en la red. Es un lenguaje multiplataforma, que puede funcionar sobre la mayoría de los servidores Web y brinda soporte a más de 20 tipos de base de datos.

Su rapidez en la ejecución y los bajos requisitos de consumo en los sistemas donde es desplegado lo hacen uno de los preferidos por los desarrolladores. Dispone de una conexión nativa a los principales sistemas de base de datos utilizados actualmente tales como Postgres, MySQL, Oracle, Microsoft SQL Server, lo cual permite la creación de aplicaciones web robustas. Su mayor ventaja

[11] Soporte a largo plazo (**LTS**, por sus siglas en inglés de: *Long Term Support*)

radica en ser un lenguaje libre, por lo que se convierte en una alternativa de muy fácil acceso, además de contar con una comunidad de desarrolladores que intercambian experiencias lo que facilita la rápida solución de problemas sin costo alguno (Sæther Bakken, y otros, 2002).
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.

### 1.4.11. Doctrine

Doctrine es un potente y completo sistema para el mapeo de objeto relacional (ORM por sus siglas en inglés de Object Relational Mapping) para PHP con un DBAL[12] incorporado el cual permite trabajar con un esquema de base de datos como si fuese un conjunto de objetos. Está inspirado en Hibernate, que es uno de los ORM más populares y grandes que existen y nos brinda una capa de abstracción de la base de datos muy completa. Posee la habilidad de escribir opcionalmente las consultas de la base de datos utilizando la programación orientada a objetos debido a que Doctrine utiliza el patrón Active Record para manejar las bases de datos. Esto les brinda una alternativa poderosa a diseñadores de SQL que mantiene un máximo de flexibilidad sin requerir la duplicación de código innecesario. La característica más importante es que te da la posibilidad de escribir consultas de base de datos en un lenguaje propio llamado Doctrine Query Language (DQL) (Doctrine-project.org, 2018).
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 propuesta de solución como resultado de la presente investigación.

### 1.4.12. Apache

Apache es un servidor web de código libre para la transferencia de hipertextos (*Hypertext Transfer Protocol*, HTTP por sus siglas en inglés) para plataformas Unix, Windows y Macintosh. Su implementación se realiza de forma colaborativa, con prestaciones y funcionalidades equivalentes a las de los servidores comerciales. El proyecto está dirigido y controlado por un grupo de voluntarios de todo el mundo que, usando Internet y la web para comunicarse, planifican y desarrollan el servidor y la documentación relacionada.
Principales características:

- Es un servidor de web conforme al protocolo HTTP.
- Soporta tanto host basados en IP como hosts virtuales.
- Apache soporta autentificación básica basada en la Web.
- Modular: Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos.

[12] DBAL: Database Abstraction Layer (Capa de abstracción de base de datos)

- Extensible: gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor (del Castillo San Félix, 2000).

Para el desarrollo del XEJEL se decide la utilización de 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, por lo que la convierte en la versión más rápida de Apache.

### 1.4.13. PostgreSQL

PostgreSQL es un poderoso sistema de base de datos relacional de objetos de código abierto. Tiene más de 15 años de desarrollo activo y una arquitectura comprobada que le ha valido una sólida reputación de fiabilidad, integridad de datos y corrección. Sus principales características son: llaves ajenas o llaves foráneas, disparadores, vistas, integridad transaccional, acceso concurrente multiversión (no se bloquean las tablas, ni siquiera las filas, cuando un proceso escribe), capacidad de albergar programas en el servidor en varios lenguajes. Herencia de tablas, tipos de datos y operaciones geométricas. A continuación, se muestran algunas de sus ventajas:

- Es altamente escalable en cuanto a cantidad de datos y usuarios.
- Es multiplataforma.
- Ofrece una documentación bien organizada, pública y libre.
- Presenta conectores de datos foráneos que permiten añadir y consultar fuentes de datos externas desde PostgreSQL (PostgreSQL, 2018).

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.

### 1.4.14. NetBeans

NetBeans es un entorno de desarrollo integrado (IDE por sus siglas en inglés de *Integrated Development Environment*), de código abierto dedicado a proporcionar productos de desarrollo de software sólidos (el IDE de NetBeans y la plataforma de NetBeans) que atienden las necesidades de los desarrolladores, usuarios y las empresas que confían en NetBeans como base para sus productos; en particular, para permitirles desarrollar estos productos de forma rápida, eficiente y fácil al aprovechar las fortalezas de la plataforma Java y otros estándares relevantes de la industria. El IDE de NetBeans proporciona soporte para varios lenguajes (PHP, JavaFX, C / C ++, *JavaScript*, entre otros), así como varios marcos de trabajo (Oracle, 2018).

Para el desarrollo del XEJEL 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, lo cual permite realizar tareas de control de versiones directamente desde el proyecto dentro del IDE.

### 1.4.15. Git

Git es un sistema de control de versiones gratuito, de código abierto, y se distribuye, o sea, todos los desarrolladores tienen el historial completo de su repositorio de códigos a nivel local. Esto hace

que el clon inicial del repositorio sea más lento, pero las operaciones subsiguientes, como cometer, culpar, difuminar, fusionar y registrar mucho más rápido. Incluye las funcionalidades de crear ramas y *merges*[13], además de reescribir historiales de repositorios, lo cual ha dado como resultado muchas herramientas innovadoras y eficaces. Básicamente, funciona del siguiente modo:

1. Crea un "repositorio" (proyecto) con una herramienta de alojamiento de Git.
2. Copia (o clona) el repositorio a una máquina local.
3. Añade un archivo al repositorio local y confirma (*commit*) los cambios.
4. Envía (*push*) los cambios a la rama principal.
5. Realiza cambios en un archivo con una herramienta de alojamiento de Git y se confirman.
6. Extrae (*pull*) los cambios a una máquina local.
7. Crea una rama (*branch*, versión), se realiza algún cambio y se confirma.
8. Fusiona (*merge*) a una rama con la rama principal (Murua, y otros, 2019).

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 como resultado de la presente investigación.

**Conclusiones del capítulo**

En este capítulo se abordaron los principales conceptos asociados a los expedientes judiciales electrónicos que facilitaron una mejor comprensión del objeto de estudio de la investigación. Por otra parte, el estudio referente a las soluciones similares permitió concluir que no existe una herramienta capaz de satisfacer las necesidades de los TPC, teniendo en cuenta la soberanía tecnológica por la que opta el país. Por último, el estudio de las características de la metodología de desarrollo de software, las herramientas y lenguajes de programación a utilizar fundamenta su selección por parte del equipo del XEJEL para el desarrollo de los componentes para la gestión de la documentación en dicha herramienta.

[13] Hacer fusiones entre diferentes ramas

# Capítulo 2: Descripción de la propuesta de solución

En el presente capítulo se abordan los principales elementos relacionados con la propuesta de solución. Se realiza una descripción general del posible resultado, luego se procede a realizar la disciplina de Requisitos abarcando los requisitos funcionales y no funcionales de la propuesta de solución. Por otro lado, se obtienen los artefactos correspondientes a la disciplina de Análisis y diseño, aplicando los patrones de diseño definidos como buenas prácticas durante el ciclo de desarrollo del software, los diagramas de clases de diseño, el modelo de datos y el patrón arquitectónico que da soporte a la propuesta de solución. Por último, se definen los estándares de codificación como base a la implementación de los componentes propuestos.

## 2.1 Descripción general de la propuesta de solución

La herramienta informática XEJEL contribuye a agilizar el proceder con los expedientes que intervienen en los procesos judiciales. El empleo del expediente judicial en formato electrónico se materializa en la sustitución de su tradicional legajo por su equivalente en formato digital.

En base a lo anteriormente descrito y con el fin de darle cumplimiento a los objetivos previamente planteados en la presente investigación, la propuesta de solución permitirá la gestión de todo tipo de documento, generándose propiamente en el sistema o adjuntándolo al mismo. En ambos casos se debe tener en cuenta el tipo de documento y la clasificación del mismo, en caso de generarse propiamente en el sistema, este debe ser capaz de mostrarle una plantilla al usuario con el fin de que se pueda asociar esa plantilla a esa clasificación que le corresponde. Esta plantilla una vez creada será la utilizada en todo momento cuando se desee adicionar un documento con la misma clasificación. Una vez creados estos documentos, el sistema debe dar la posibilidad de listarlos, una vez realizado un criterio de búsqueda, con el fin de visualizarlos, editarlos y guardarlos para futuros cambios, o simplemente pasarlos a definitivo. En caso de optar por adjuntar el documento, el sistema permitirá cargar el documento ya previamente creado y trasladado al tribunal para ser adjuntado al expediente en un momento determinado. Por último, los documentos de cada uno de los expedientes estarían disponibles, para todo el personal jurídico con permiso hacia los mismos, desde cualquier parte del territorio nacional a través de la herramienta informática, en caso de que decidan realizar una búsqueda para su consulta con el fin de tomar una decisión sobre el proceso en cuestión, garantizando su preservación y durabilidad a través de los mecanismos establecidos por parte de los TPC.

## 2.2 Requisitos

La tarea principal de la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto (Rodríguez Sánchez, 2015).

Los requisitos para un software son descripciones de lo que el sistema debe hacer: el servicio que ofrece y las restricciones en su operación. Estos reflejan las necesidades de los clientes por un sistema que atienda cierto propósito (Sommerville, 2011).

Para la obtención o captura de requisitos en la propuesta de solución se emplearon las técnicas:

- **Entrevista:** Es una técnica de gran utilidad para obtener información cualitativa como opiniones o descripciones subjetivas de actividades. Es una técnica muy utilizada, y requiere una mayor preparación y experiencia por parte del analista (Sommerville, 2011). Esta técnica se evidencia durante el levantamiento de información realizado en las reuniones ejecutadas con el cliente, donde se formularon un conjunto de preguntas con el objetivo de entender las necesidades del mismo y concebir un lenguaje común entre el cliente y el equipo de desarrollo de la propuesta de solución (ver anexo 1).
- **Tormenta de ideas:** Es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios (Pressman, 2010). Esta técnica se evidencia durante las reuniones con el cliente donde, luego de un diálogo entre ambas partes, se obtuvo como resultado un conjunto de acuerdos con el fin de refinar las necesidades del cliente, estos acuerdos se muestran en el anexo 2 del propio documento.

### 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 o software 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, cumplimiento, seguridad u otra índole (Pressman, 2010).

Una vez aplicada las técnicas de obtención de Requisitos se identificaron 14 requisitos funcionales para la propuesta de solución, en la siguiente tabla se muestran las descripciones de cada uno de los requisitos.

**Tabla 1:** descripción de los Requisitos funcionales

| No | Nombre | Descripción |
|---|---|---|
| RF1. | Buscar documentos del expediente | El sistema permitirá buscar los documentos de un expediente, dando la posibilidad de aplicar criterios de búsquedas según crea pertinente |
| RF2. | Listar documentos en definitivo | El sistema permitirá listar todos los documentos en estado definitivo, asociados a un expediente |
| RF3. | Descargar documento en definitivo | El sistema permitirá descargar los documentos en estado definitivo, asociados a un expediente |
| RF4. | Eliminar documentos en definitivo | El sistema permitirá eliminar los documentos en definitivo, asociados a un expediente |

| | | |
|---|---|---|
| RF5. | Listar documentos en edición | El sistema permitirá listar todos los documentos en edición, asociados a un expediente |
| RF6. | Eliminar documentos en edición | El sistema permitirá eliminar los documentos en edición, asociados a un expediente |
| RF7. | Cargar documento en el sistema | El sistema permitirá cargar un documento, desde fuera del sistema, a un expediente seleccionado |
| RF8. | Listar documentos a adjuntar en el sistema | El sistema permitirá listar los documentos, previamente cargados, de un expediente determinado |
| RF9. | Eliminar documentos a adjuntar en el sistema | El sistema permitirá eliminar los documentos, previamente cargados, de un expediente determinado |
| RF10. | Adjuntar documentos en el sistema | El sistema permitirá adjuntar los documentos, previamente cargados, de un expediente determinado |
| RF11. | Adicionar documento en el sistema | El sistema permitirá adicionar un documento al expediente previamente seleccionado |
| RF12. | Modificar documento en el sistema | El sistema permitirá modificar el documento seleccionado, teniendo en cuenta la plantilla previamente cargada en el sistema |
| RF13. | Guardar documento en edición | El sistema permitirá guardar el documento seleccionado, para su posterior modificación |
| RF14. | Pasar documento a definitivo | El sistema permitirá pasar a definitivo el documento seleccionado sin posibilidad de modificación |

**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 los que tienen vinculación directa a los componentes realizados.

**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 será 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.

**RnF 6.** Requisito de confiabilidad 3

Se debe configurar un sistema de respaldo ante fallos del servidor que garantice una copia de la aplicación, así como del estado diario de la base de datos.

**Eficiencia**

**RnF 7.** 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 8.** Requisito de restricción de diseño e implementación 1

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

**RnF 9.** 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 10.** Requisito de interfaz de usuario 1

Se usarán los siguientes widgets 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 2

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 3

Los botones tendrán dos tonalidades de colores, los de acciones primarias, que serían 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.

### 2.2.3. Validación de los requisitos

El resultado del trabajo realizado es una consecuencia de la Ingeniería de Requisitos (especificación del sistema e información relacionada) y es evaluada su calidad en la fase de validación. La validación de requisitos examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos, y que el resultado del trabajo se ajusta a los estándares establecidos para el proceso, el proyecto y el producto (Pressman, 2010). Para la validación de los requisitos se tuvieron en cuenta las técnicas de:

- **Creación de prototipos:** En esta aproximación a la validación, se muestra un modelo ejecutable del sistema en cuestión a los usuarios finales y clientes. Así, ellos podrán experimentar con este modelo para constatar si cubre sus necesidades reales (Sommerville, 2011). En la presente investigación se realizó un modelo ejecutable utilizando el componente para que los clientes interactuaran con él y así poder determinar si cumple sus necesidades. En el epígrafe 2.2.4 se hace referencia a las Historias de Usuario como producto de trabajo de la disciplina de Requisitos, en la misma se detallan los prototipos obtenidos para la propuesta de solución para el caso del requisito que se toma como ejemplo en este documento.
- **Generación de casos de prueba:** Los requerimientos deben ser comprobables. Si las pruebas para los requerimientos se diseñan como parte del proceso de validación, esto revela con frecuencia problemas en los requerimientos. Si una prueba es difícil o imposible de diseñar, esto generalmente significa que los requerimientos serán difíciles de implementar, por lo que deberían reconsiderarse. El desarrollo de pruebas a partir de los requerimientos del usuario antes de escribir cualquier código es una pieza integral de la programación extrema (Pressman, 2010). En el Capítulo 3 del presente documento se relacionan los casos de pruebas generados en esta técnica.
- **Revisiones formales de los requisitos:** se realizaron revisiones formales de cada requisito, por parte del cliente y el equipo de desarrollo, validando que la interpretación de cada una de las descripciones no sea ambigua, ni presenten omisiones o errores.

Para medir la calidad de la especificación de los requisitos de software se aplicó la métrica Calidad de la especificación (CE). El empleo de esta métrica permite obtener un alto nivel de entendimiento y precisión de los requisitos, para llegar a ello, se debe primeramente calcular el total de requisitos de software como se muestra a continuación:

**Nr:** total de requisitos de software.
**Nf:** cantidad de requisitos funcionales.
**Nnf:** cantidad de requisitos no funcionales.

$$Nr = Nf + Nnf$$

Sustituyendo los valores en la ecuación, se obtiene:

$$Nr = 14 + 12$$

$$Nr = 26$$

Finalmente, para calcular la Especificidad de los Requisitos (ER) se tuvo en cuenta la ausencia de ambigüedad en los mismos de forma tal que se obtuviera una sola interpretación, lo que requiere que cada uno sea descrito utilizando un término único, para ello se realiza la siguiente operación:
**Nui:** número de requisitos para los cuales todos los revisores tuvieron interpretaciones idénticas.

$$Q1 = Nui / Nr$$

Para sustituir los valores de las variables en la ecuación se tuvo en cuenta que, de los requisitos especificados para el desarrollo de la propuesta de solución, cuatro de ellos causaron contradicción

(RF1, RF4, RF10 y RF12) en sus interpretaciones. Por tanto, la variable Q1 obtiene el siguiente valor:

**Q1 = 22/ 26**

**Q1 = 0.84**

Es importante aclarar que mientras más cerca de 1 está el valor de Q1, menor es la ambigüedad. Teniendo en cuenta el resultado anterior, igual a 0.84, se concluye que el 84% de los requisitos se obtuvieron una sola interpretación. Los cuatro requisitos identificados como ambiguos fueron modificados y validados para garantizar su correcta interpretación, llegando al resultado ideal de Q1=1.

### 2.2.4. Historias de usuario

Las Historias de Usuario (HU), son pequeñas descripciones de los requerimientos de un cliente. Su utilización es común cuando se aplica marcos de entornos ágiles. Además, construyen entendimiento compartido a través de la colaboración, con palabras e imágenes. En otras palabras, son discusiones acerca de las soluciones a problemas para la organización, los clientes y los usuarios, que llevan a acuerdos sobre los cuales construir lo deseado (TENSTEP.INC, 2016).

En total se definieron 14 HU, a continuación se describe una HU de prioridad Alta para el cliente, obtenida en el desarrollo de la propuesta de solución, el resto se pueden consultar en el documento (Aguilera González, 2018) creado por el autor de la presente investigación:

**Tabla 2:** historia de usuario para el RF1 “Buscar documentos del expediente”

| | |
|---|---|
| **Número:** 1 | **Requisito:** Buscar documentos del expediente |
| **Programador:** Sergio Orlando Aguilera González | **Iteración Asignada:** 1 |
| **Prioridad:** Alta | **Tiempo Estimado:** 24 horas |
| **Riesgo en Desarrollo:** | **Tiempo Real:** 24 horas |

**Descripción:**
Funcionalidad que permite buscar los documentos de un expediente, dando la posibilidad de aplicar criterios de búsquedas según crea pertinente.

**Campos:**

- Tipo de documento: Campo que permite entrar el nombre del o los documentos a filtrar
- Clasificación: Campo que permite entrar la clasificación de un documento a filtrar
- Fecha: Campo que permite seleccionar la fecha de creación del documento a filtrar

**Botones:**

- Filtrar: Opción que permite aplicar un filtro a la lista de documentos mostrada
- Aceptar: Opción que permite iniciar la búsqueda

**Observaciones:** N/A

**Prototipo elemental de interfaz gráfica de usuario:**


**Fuente:** Elaboración propia

### 2.3 Arquitectura del sistema

En esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales (Rodríguez Sánchez, 2015).

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, 2011).

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

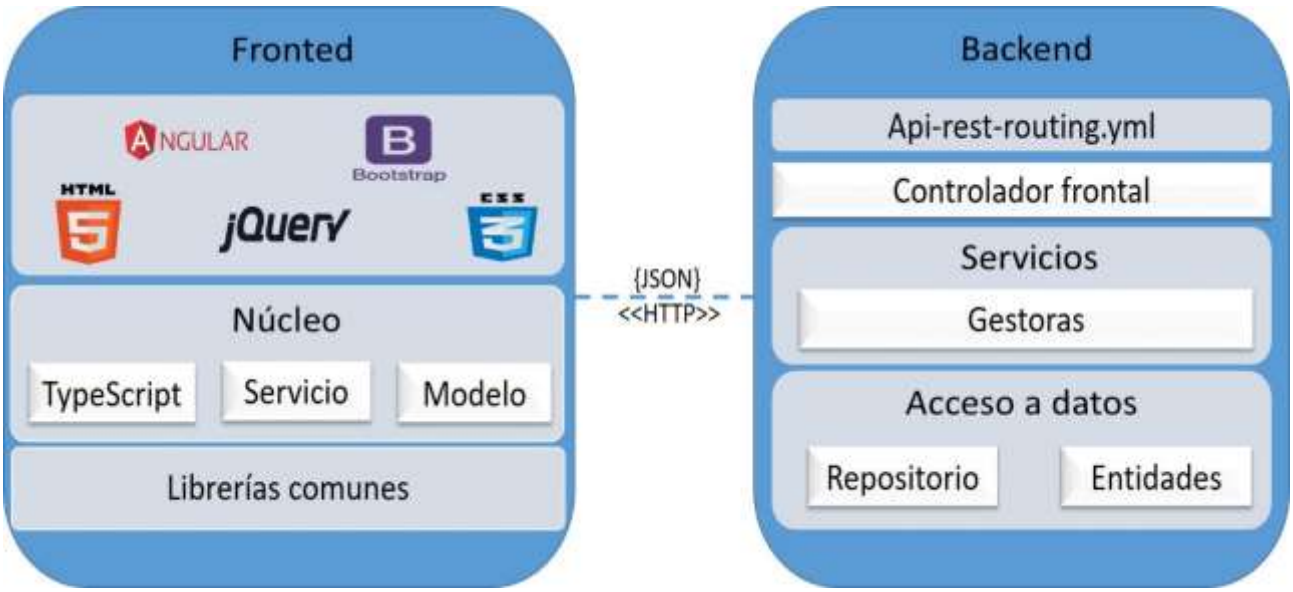


**Figura 1:** arquitectura del XEJEL

**Fuente:** elaboración propia

En la figura mostrada anteriormente se exponen 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[14].

### 2.3.1. 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, 2017).

La solución propuesta tiene como base la arquitectura *frontend* del XEJEL, la cual se basa en el uso del patrón arquitectónico Modelo Vista Controlador (MVC).

Este patrón separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El modelo contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia. La vista, o interfaz de usuario, compone la información que se envía al cliente y los mecanismos de interacción con éste. El controlador, actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno (Álvarez, 2014).

[14] (Notación de objetos JavaScript, por sus siglas en inglés JavaScript Object Notation)

En la siguiente figura se muestra una descripción detallada de cómo inciden los componentes propuestos en dicha arquitectura:

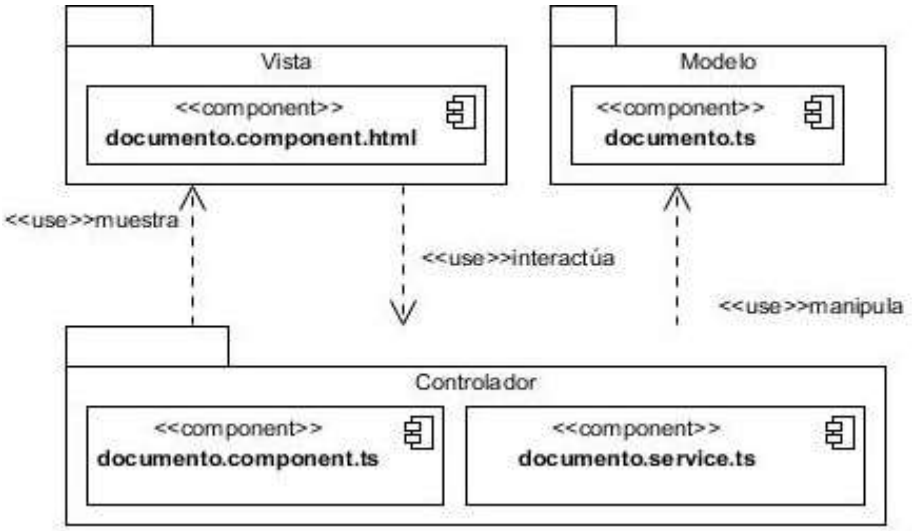


**Figura 2:** arquitectura *frontend* de la HU_ Buscar documentos del expediente

**Fuente:** elaboración propia

En la figura el modelo para el componente referente a la HU_ Buscar documentos del expediente está compuesto por el archivo *documento.ts*, el cual contiene la información que se le solicita al usuario para mostrar los datos del documento. Por otra parte, la vista se compone por el archivo *documento.component.html*, este representa la información visible al usuario e interactúan con funciones específicas del controlador. El controlador está compuesto por el archivo *documento.component.ts*, encargado de interactuar con el modelo y mostrar la información a la vista.

### 2.3.2. Arquitectura backend

En la parte del *backend,* la propuesta de solución incide en las capas controladora, gestora y acceso a datos, la misma se basa sobre la arquitectura *backend* del XEJEL a través del uso del patrón arquitectónico N Capas.

La arquitectura basada en N Capas se enfoca principalmente en el agrupamiento de funcionalidad relacionada dentro de una aplicación en distintas capas que son colocadas verticalmente una encima de otra. La funcionalidad dentro de cada capa se relaciona con un rol o responsabilidad especifica. El dividir en capas una aplicación, permite la separación de responsabilidades lo que proporciona una mayor flexibilidad y un mejor mantenimiento (Muñoz Serafin, 2018).

En la siguiente figura se muestra una descripción detallada de cómo inciden los componentes propuestos en dicha arquitectura:

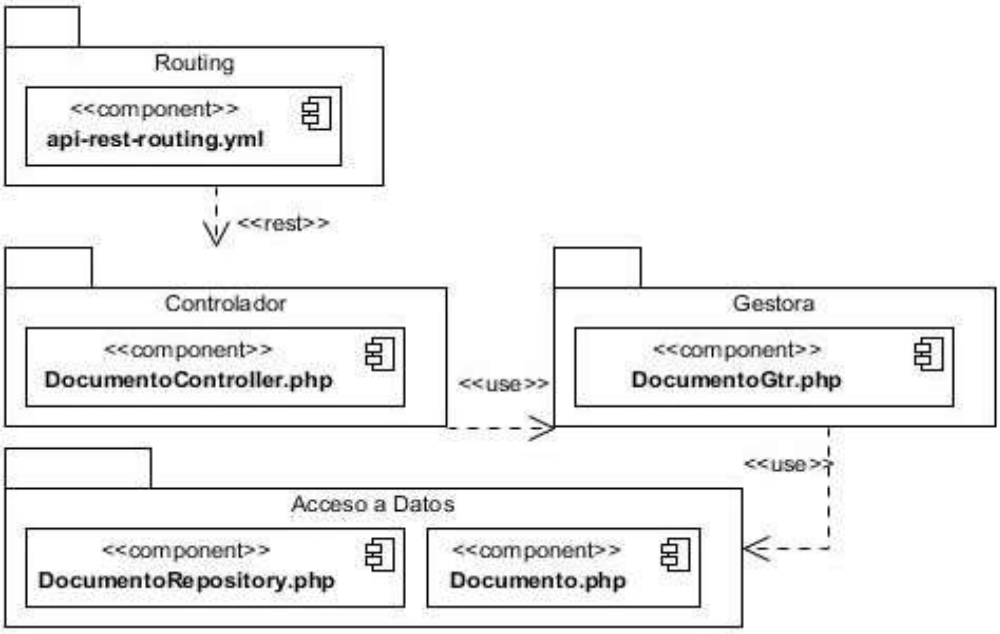


**Figura 3:** arquitectura backend para la HU_ Buscar documentos del expediente

**Fuente:** elaboración propia

En la figura presentada anteriormente, la clase *DocumentoController.php* de la capa Controladora, es la encargada de controlar el flujo de información en el sistema y usa el componente de la capa Gestora mediante la clase *DocumentoGtr.php*, para gestionar los datos en el modelo, esta capa a su vez, usa el componente de la capa de Acceso a Datos a través de las *DocumentoRepository.php y Documento.php,* esta última posee los datos del documento en el sistema una vez mapeada la base de datos de la propuesta de solución. La relación entre los componentes del *frontend* y del *backend* se ve reflejada en el componente *api-rest.routing.yml*, el cual será el encargado de dirigir las peticiones que llegan desde el *frontend* hacia la clase controladora que corresponda.

## 2.4 Patrones de diseño

Un patrón de diseño describe una estructura que resuelve un problema de diseño en particular dentro de un contexto específico y en medio de fuerzas que pueden tener un impacto en la manera en que se aplica y utiliza el patrón (Pressman, 2010). A continuación, se describen cada uno de los patrones utilizados en la propuesta de solución:

### 2.4.1. Patrones GRASP

Los Patrones Generales de Software para Asignación de Responsabilidades (GRASP por sus siglas en inglés de *General Responsibility Assignment Software Patterns*) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Larman, 2016). A continuación, se describen los patrones GRASP utilizados:

**Patrón Controlador:** El objetivo de este patrón es asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Delega la responsabilidad en otras clases con las que mantiene un modelo de alta cohesión. 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 dicho marco de trabajo. Para una mayor comprensión consultar la figura 1. A continuación se muestra la clase *DocumentoController.php* correspondiente al componente de documento de la propuesta de solución:

```
class DocumentoController extends BaseApiController
{
    /** @Rest\Post("/definitivo/{idexpediente}") ...*/
    public function documentosEnDefinitivoAction($idexpediente, Request $request, DocumentoGtr $documentoGtr, DocumentoTm $documentoTm){...}
    /** @Rest\Get("/documentosclasificacion/listar/{idExpediente}/{idTipoDocumento}") ...*/
    public function listaDocumentosClasificacionAction($idExpediente, $idTipoDocumento, Request $request, DocumentoGtr $documentosGtr){...}
    /** @Rest\Get("/listar-enEdicion-get/{idexpediente}") ...*/
    public function getDocumentoEdicionAction($iddocumento, DocumentoGtr $documentosGtr){...}
    /** @Rest\Post("/crear-doc-edicion/{idExpediente}/{idTipoDocumentoClasificacion}") ...*/
    public function crearDocEnEdicionAction($idExpediente, $idTipoDocumentoClasificacion, Request $request, DocumentoGtr $documentoGtr){...}
    /** @Rest\Post("/adjuntar/{idexpediente}/{idTipoDocumentoClasificacion}") ...*/
    public function subirDocumentosAction($idexpediente, $idTipoDocumentoClasificacion, Request $request, DocumentoGtr $documentoGtr, FtpGtr $ftpGtr){...}
    /** @Rest\Get("/descargar-documento/{iddoc}") ...*/
    public function descargarDocumentoAction($iddoc, FtpGtr $ftpGtr){...}
    /** @Rest\Delete("/{id}") ...*/
    public function eliminarDocumentoAction($id, DocumentoGtr $documentoGtr){...}
    /** @Rest\Get("/listar-doc-edicion/{idexpediente}") ...*/
    public function documentosEnEdicionGetAction($idexpediente, DocumentoGtr $documentoGtr){...}
    /** @Rest\Get("/{iddocumento}") ...*/
    public function obtenerDocumentoEdicionAction($iddocumento, DocumentoGtr $documentoGtr){...}
    /** @Rest\Put("/{iddocumento}") ...*/
    public function guardarDocEnEdicionAction($iddocumento, Request $request, DocumentoGtr $documentoGtr){...}
    /**@Rest\Get("/regenerarDoc/{iddocumento}") ...*/
    public function regenerarDocumentoEdicionAction($iddocumento, DocumentoGtr $documentoGtr){...}
    /** @Rest\Put("/pasar-definitivo-doc/{iddocumento}") ...*/
    public function pasarDefinitivoDocAction($iddocumento, Request $request, DocumentoGtr $documentoGtr){...}
}
```

**Figura 4:** patrón Controlador. Clase *DocumentoController.php*

**Fuente:** elaboración propia

**Patrón Experto:** Este patrón define la clase experta en información, dado que esta es la que contiene toda la información para cumplir con la responsabilidad. De este modo se obtendrá un diseño con mayor cohesión y así la información se mantiene 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 engloban en la propuesta de solución. En la siguiente figura se muestra la clase entidad *Documento.php*, la cual es la experta en información para el tratamiento de los documentos en el XEJEL.

```php
class Documento{
    /** @var int ...*/
    private $id;

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

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

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

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

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

    /** @ORM\ManyToOne(targetEntity= "AppBundle\Entity\Comun\Expediente", inversedBy="documentos") ...*/
    private $expediente;

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

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

    public function __construct(){...}

    /**Metodos get y set de cada uno de los atributos ...*/
}
```

**Figura 5:** patrón Experto. Clase *Documento.php*

**Fuente:** elaboración propia

**Patrón Creador:** Permite 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. A continuación, se muestra como desde la clase *DocumentoGtr.php,* se hace instancia de la clase *Documento.php* al crear un objeto de tipo documento.

```php
$documento = new Documento();
$documento->setTipoEstadoDocumento($this->getEm()->find( className: TipoEstadoDocumento::class,  id: TipoEstadoDocumento::EDICION));
$documento->setFechaRadicacion(FechaUtil::actual());
$documento->setExpediente($this->getEm()->find( className: Expediente::class, $idExpediente));
$tipoDocumentoClasificacion = $this->getEm()->find( className: TipoDocumentoClasificacion::class, $idTipoDocumentoClasificacion);
$documento->setTipoDocumentoClasificacion($tipoDocumentoClasificacion);
$documento->setNombredocumento($tipoDocumentoClasificacion->getDenominacion());

$plantilla = $this->plantillaGtr->generate($tipoDocumentoClasificacion->getPlantilla(), $idExpediente);
$documento->setPlantilla($plantilla);
```

**Figura 6:** patrón Creador. Clase *DocumentoGtr.php*

**Fuente:** elaboración propia

**Patrón Bajo acoplamiento:** Este patrón plantea la baja dependencia que debe existir entre las clases y está estrechamente relacionado con los patrones Experto o Alta cohesión (Larman, 2016). Symfony favorece ampliamente el bajo acoplamiento de las clases en el sistema, dado que a cada clase se le asignan solamente las responsabilidades necesarias de manera que no dependan en gran medida de otras (ver figura 3).

**Patrón Alta cohesión:** Este patrón permite 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 (ver figura 3).

**Patrón Fabricación pura:** Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar; su objetivo es devolver una instancia de múltiples tipos de objetos, que normalmente provienen de una misma clase padre y sólo se diferencian entre ellos por algún aspecto de comportamiento. Las clases de fabricación pura casi siempre se dividen atendiendo a su funcionalidad (Larman, 2016). Este patrón se evidencia en las clases útiles (*NombreClaseUtil.php*) 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 *FechaUtil.php* de la cual la propuesta de solución se nutre de los servicios que posee la misma al reflejar el tratamiento con las fechas en el sistema.

```php
/** Class FechaUtil ...*/
class FechaUtil
{
    const SUNDAY_DAY = 'sunday';
    const SATURDAY_DAY = 'saturday';
    const FORMAT_D_M_Y = 'd/m/Y';
    const FORMAT_D_MA_Y = 'd/M/Y';
    const FORMAT_D_MA = 'd \d\e M'; //Ex: 30 de Nov

    /** Retorna la fecha actual ...*/
    public static function actual($format = null){...}

    /** @param $day ...*/
    public static function next($day, $today = true){...}

    /** Retorna los dias de diferencia entre una fecha segun calendario natural ...*/
    public static function restarFechaNatural(\DateTime $pfecha1, \DateTime $pfecha2){...}

    /** @param \DateTime $fecha fecha a convertir a string ...*/
    public static function toString(\DateTime $fecha){...}
}
```

**Figura 7:** patrón Fabricación pura. Clase *FechaUtil.php*

**Fuente:** elaboración propia

### 2.4.2. Patrones GoF

En el libro “Design Patterns: Elements of Reusable Object Oriented Software” escrito por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, se realiza una recopilación de 23 patrones de diseño aplicados usualmente por expertos diseñadores de software orientado a objetos. Desde luego que estos no son los inventores ni los únicos involucrados, pero fue el punto de partida para difundirse con más fuerza la idea de patrones de diseño Gang of Four (GoF), conocidos de igual forma como patrones de la pandilla de los cuatro (Gamma, y otros, 1994). A continuación, se describen los patrones GoF utilizados:

**Patrón Decorador:** permite 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 se evidencia en la implementación de los componentes en el *frontend* como se muestra en la siguiente figura:

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

**Figura 8:** patrón Decorador. Clase *adicionar-documento.component.ts*

**Fuente:** elaboración propia

**Patrón Observador:** permite observar los cambios producidos por un objeto, de esta forma, cada cambio que afecte el estado del objeto observado lanzará una notificación a los observadores; a esto se le conoce como Publicador-Suscriptor. Es uno de los principales patrones de diseño utilizados en interfaces gráficas de usuario, ya que permite desacoplar al componente gráfico de la acción a realizar (Larman, 2016). Este patrón es empleado en el *frontend*, 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 dicho 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:

```
aceptar() {
    switch (this.radioMarcado) {
        case 'adjuntar': {
            this.router.navigate( commands: [this.rutas.routes.edit_documentos, this.idExpediente]);
            break;
        }
        case 'generar': {
            this.documentoService.crearDocEdicion(this.idExpediente, this.idTipoDocumentoClasificacic
                .subscribe( next: value => {
                    this.eventoGenerarDocumento.emit( value: {
                        'idExpediente': this.idExpediente,
                        'idDocumento': value
                    });
                });
            break;
        }
    }
}
```

**Figura 9:** patrón Observador. Clase *adicionar-documento.component.ts*

**Fuente:** elaboración propia

### 2.4.3. Otros patrones

**Patrón Inyección de dependencias:** 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 Angular los servicios son clases TypeScript, su propósito es contener la lógica de negocio y las clases para el acceso a datos. Estas clases se pueden instanciar desde cualquier otro fichero que las importe. Angular facilita hacer uso de este patrón a través de la inyección del objeto desde el

constructor de las clases TypeScript. En el método que se muestra en la siguiente figura se evidencia la aplicación de este patrón:

```
constructor(private documentoService: DocumentoService,
            private activatedRoute: ActivatedRoute,
            private router: Router) {
    super();
}
```

**Figura 10:** patrón Inyección de dependencias. Clase *adicionar-documento.component.ts*

**Fuente:** elaboración propia

### 2.5 Diagrama de clases del diseño

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema, muestra los elementos que lo forman (clases e interfaces) y las relaciones entre ellos (asociaciones). Las clases se representan mediante una caja subdividida en tres partes: en la superior se muestra el nombre, en el medio los atributos y en la inferior las operaciones o métodos. Las asociaciones entre dos clases se representan mediante una línea que las une y esta puede tener una serie de elementos gráficos que expresan características particulares de la asociación (Pressman, 2010). A continuación, se muestra el diagrama de clase del diseño para el componente Documentos.

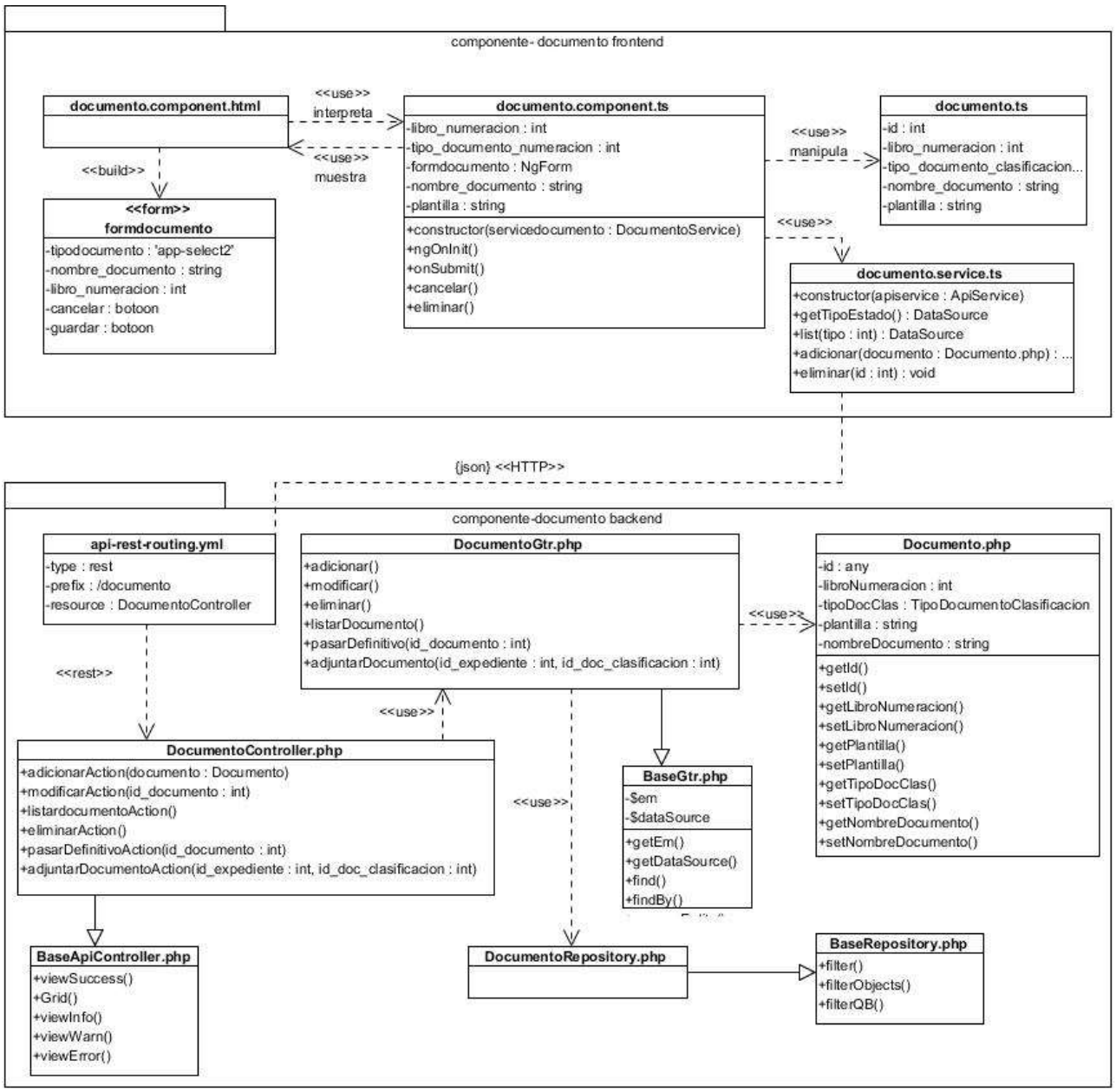


**Figura 11:** diagrama de clases del diseño para el componente de Documentos.

**Fuente:** elaboración propia

En el diagrama de clases del diseño se evidencia cómo la vista interactúa con la controladora, esta a su vez manipula al modelo y se nutre de un servicio ejecutado mediante una petición realizada al *backend* a través del protocolo HTTP. Esta petición llega al *api-rest-routing* el cual redirecciona hacia la controladora, esta a su vez hace uso de los servicios de la gestora encargadas de manipular las clases entidades mapeadas desde la base de datos.

### 2.6 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). En la figura que se muestra a continuación se relacionan las estructuras de datos persistente como resultado de la propuesta de solución:

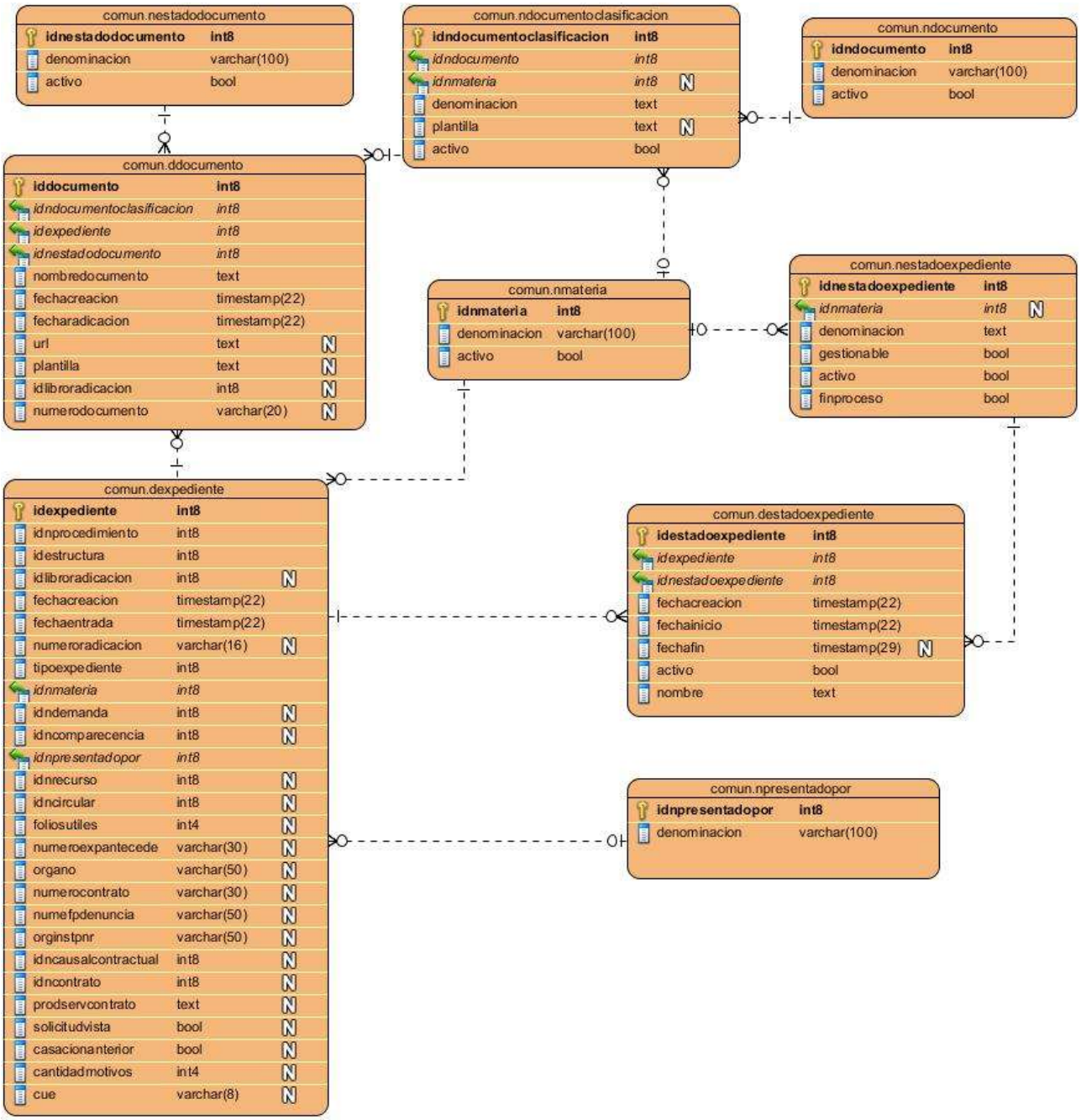


**Figura 12:** modelo de datos para la propuesta de solución

**Fuente:** elaboración propia

El modelo de datos obtenido cuenta con un total de 9 tablas, de ellas 6 son nomencladores encargados de gestionar conceptos específicos del negocio anteriormente predefinido, por ejemplo: la tabla *comun.ndocumento* se encargará de nomenclar los tipos de documentos que se pueden gestionar en el sistema. Además se cuenta con 3 tablas encargadas de guardar la información referente a cada concepto que intervienen en la propuesta de solución, por ejemplo: la tabla *comun.ddocumento*, en la cual se guardará la información persistente al gestionar los documentos de un expediente en el sistema.

## 2.7 Implementación

En la implementación, a partir de los resultados de la disciplina de Análisis y diseño, se construye el sistema (Rodríguez Sánchez, 2015). Para ello a continuación se describen los estándares de codificación a tener en cuenta en el desarrollo de la propuesta de solución.

### 2.7.1. Estándares de codificación

Los estándares de codificación son normas que deben seguirse durante todo el ciclo de implementación del sistema (ohmyroot, 2017).

Los mismos fueron seleccionados por el equipo de arquitectura y diseño del proyecto, donde esta información se encuentra plasmada íntegramente en el documento (Aleaga, 2018) del expediente del proyecto XEJEL. A continuación, se describen algunos de los estándares más utilizados en la implementación de los componentes para la gestión de los documentos y plantillas en el XEJEL:

**Declaración de clases:**

- Las Clases deben ser nombradas de acuerdo a la convención de nombres.
- La llave "{" deberá escribirse siempre en la línea debajo del nombre de la clase ("*one true brace*").
- 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. Únicamente una clase está permitida por archivo PHP. Incluir código adicional en archivos de clase está permitido pero esta desaconsejado. En archivos de ese tipo, dos líneas en blanco deben separar la clase de cualquier código PHP adicional en el archivo de clase.

A continuación, se muestra un ejemplo de una declaración de clase que es permitida:

```
class DocumentoController extends BaseApiController
```

**Figura 13:** 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 función, 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".
- Por norma general, se recomienda la elocuencia. Los nombres de función deben ser lo suficientemente elocuentes como para describir su propósito y comportamiento.

A continuación, se muestra un ejemplo de nombres de funciones admisibles:

```php
public function guardarDocEdicion($iddocumento, $data)
{
    $documento = $this->getDocumentoEdicion($iddocumento);
    $documento->setPlantilla($data['plantilla']);

    $this->getEm()->persist($documento);
    $this->getEm()->flush();

    return $documento->getId();
}
```

**Figura 14:** nombre de funciones

**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 (_). Por ejemplo:

```php
const MONDAY_DAY = 'lunes';
const TUESDAY_DAY = 'martes';
const WEDNESDAY_DAY = 'miércoles';
const THURSDAY_DAY = 'jueves';
const FRIDAY_DAY = 'viernes';
const SATURDAY_DAY = 'sábado';
const SUNDAY_DAY = 'domingo';
const FORMAT_D_M_Y = 'd/m/Y';
const FORMAT_D_MA_Y = 'd/M/Y';
const FORMAT_D_MA = 'd \d\e M';
```

**Figura 15:** constantes

**Fuente:** elaboración propia

Las constantes deben ser definidas como miembros de clase con el modificador "const". Definir constantes en el alcance global con la función "define" está permitido, pero no recomendado.

**Comentarios en las funciones**: Todas las funciones deben tener un comentario, antes de su declaración, explicando qué hacen. Ningún programador debería tener que analizar el código de una función para conocer su utilidad. Tanto el nombre como el comentario que acompañe a la función deben bastar para ello.

**Ubicación y denominación de archivos**: Se ubicarán los archivos según las convenciones establecidas por Symfony 3.4 o según las especificaciones del equipo de arquitectura. Para la denominación de los archivos se seguirán las convenciones establecidas a continuación:

- Para las clases controladoras se usará el sufijo Controller
- Para las clases de gestión del negocio se usará el sufijo Gtr
- Para las clases que definen las tablas en el sistema se usará el sufijo Tm
- Para las clases repositorio se usará el sufijo Repository

**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 inician con letra minúscula, pero si estas tienen más de una palabra, cada nueva palabra debe iniciar con letra mayúscula. Las constantes deben de 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 14).

**Definiciones de la función:** Los nombres de la función pueden contener solo caracteres alfanuméricos. Los nombres de la función siempre deben empezar en letras minúsculas. Cuando un nombre de la función consiste de más de una palabra, la primera letra de cada nueva palabra debe capitalizarse (ver figura 14).

**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 14).

**Siempre incluir las llaves:** En todo momento a la hora de codificar un bloque de instrucciones, este debe ir encerrado entre llaves, aun cuando conste de una sola línea (ver figura 14).

**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 de la siguiente manera:

```
if (isset($filters['materia']))
```

**Figura 16:** verificar si las variables están inicializadas

**Fuente:** elaboración propia

Pero solo se debe usar esta opción cuando no se tenga el control o no se esté seguro del valor que pueda tener la variable (Como las variables que llegan por parámetro o por peticiones GET, SET, PUT, entre otras).

**Conclusiones del capítulo**

La descripción general de la propuesta de solución permitió obtener una visión de los flujos de información y las actividades que engloban el negocio. Además, a través de la especificación de los requisitos e historias de usuarios, se documentó toda la información relativa a la propuesta de solución. Por su parte, el diseño de la propuesta de solución permitió concebir los elementos necesarios para la implementación de los componentes de gestión de los documentos y plantillas. De igual forma, el estudio de la arquitectura definida para el XEJEL facilitó abstraerse del problema en el que se enmarca la presente investigación, y a su vez identificar cómo interactúa el sistema en el que se desenvuelve la propuesta actual. Por último, la identificación de los estándares de codificación favoreció sentar las normas a seguir durante todo el ciclo de la implementación.

# Capítulo 3: Evaluación de la solución propuesta

En este capítulo se evalúa el grado de calidad y fiabilidad de los resultados obtenidos en el desarrollo de los componentes para la gestión de la documentación en el XEJEL. 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. Por último, se aplican las disciplinas de pruebas internas y pruebas de liberación que propone la metodología que guía el desarrollo de la solución propuesta, con el fin de verificar y revelar la calidad del producto antes de ser entregado al cliente.

## 3.1 Validación del diseño

Para validar el diseño realizado se aplicaron un conjunto de métricas. Los resultados obtenidos por cada una de las métricas utilizadas se muestran a continuación.

### 3.1.1. Métrica Tamaño Operacional de las Clases

Las métricas basadas en el Tamaño Operacional de las Clases (TOC) se basan en contar la cantidad de atributos y la cantidad de operaciones que tiene una clase individual y el promedio que presenta el sistema en su totalidad. Se evalúa a partir de los siguientes atributos de calidad (Lorenz, y otros, 1994):

- Responsabilidad
- Complejidad de implementación
- Reutilización

Una vez analizado el indicador tamaño de clase, si el valor resultante tiende al crecimiento, es probable que la clase posea un alto grado de responsabilidad; en consecuencia, el nivel de reutilización sería mínimo y la implementación altamente compleja. Para medir los atributos de calidad se definieron los umbrales que se muestran en la Tabla 2. En la misma se emplean la abreviatura Prom (promedio).

**Tabla 3:** métrica TOC. Categoría por atributos y criterio de evaluación.

| Atributo de calidad | Categoría | Criterio |
|---|---|---|
| Responsabilidad | Baja | TOC < =Promedio (Prom). |
| | Media | TOC Entre Prom. y 2* Prom. |
| | Alta | TOC >2*Prom. |
| Complejidad de implementación | Baja | TOC < =Prom. |
| | Media | Entre Prom. y 2* Prom. |
| | Alta | TOC >2*Prom. |
| Reutilización | Baja | TOC >2*Prom. |
| | Media | TOC Entre Prom. y 2* Prom. |
| | Alta | TOC < =Prom. |

**Fuente:** (Lorenz, y otros, 1994)

A continuación, se muestran las medidas de los parámetros de calidad obtenidas luego de aplicar la métrica TOC al diseño de clases de los componentes como resultado de la presente investigación, teniendo en cuenta las categorías por atributos y los criterios de evaluación de la tabla anterior. En el caso de la siguiente tabla se utilizaron las abreviaturas:

- **Rp** para el atributo de Responsabilidad
- **Ci** para el atributo de Complejidad de implementación
- **R** para el atributo de Reutilización

**Tabla 4:** resultados obtenidos luego de aplicada la métrica TOC

| No. | Clase | Cantidad de procedimientos | Rp | Ci | R |
|---|---|---|---|---|---|
| 1 | documento.component.html | 12 | Media | Media | Media |
| 2 | documento.component.ts | 5 | Baja | Baja | Alta |
| 3 | documento.ts | 1 | Baja | Baja | Alta |
| 4 | documento.service.ts | 5 | Baja | Baja | Alta |
| 5 | formdocumento | 3 | Baja | Baja | Alta |
| 6 | api-rest-routing.yml | 2 | Baja | Baja | Alta |
| 7 | DocumentoGtr.php | 11 | Media | Media | Media |
| 8 | Documento.php | 17 | Alta | Alta | Baja |
| 9 | DocumentoController.php | 12 | Media | Media | Media |
| 10 | BaseGtr.php | 7 | Baja | Baja | Alta |
| 11 | BaseApiController.php | 6 | Baja | Baja | Alta |
| 12 | DocumentoRepository.php | 2 | Baja | Baja | Alta |
| 13 | BaseRepository.php | 25 | Alta | Alta | Baja |
| 14 | editarplantilla.component.html | 3 | Baja | Baja | Alta |
| 15 | editarplantilla.component.ts | 5 | Baja | Baja | Alta |
| 16 | app-editor | 8 | Baja | Baja | Alta |
| 17 | PlantillaGtr.php | 5 | Baja | Baja | Alta |
| 18 | PlantillaController.php | 3 | Baja | Baja | Alta |

**Fuente:** elaboración propia

Al aplicar la métrica TOC a todas las clases del modelo de diseño obtenido, y después de haber estudiado los resultados se obtuvieron para cada atributo de calidad los siguientes resultados:


**Figura 17:** resultado de la métrica TOC

**Fuente:** elaboración propia

Después de evaluar los resultados obtenidos de cada uno de los atributos de calidad que mide esta métrica, se obtuvieron los siguientes resultados:

- El 56% de las clases poseen una baja responsabilidad y complejidad de pruebas.
- El 56% de las clases poseen una alta reutilización.

Estos resultados demuestran la calidad del diseño de la solución propuesta, ya que, al obtener bajos índices de responsabilidad y complejidad, unidos a un alto grado de reutilización de las clases se facilita en gran medida la implementación de los componentes como resultados de la propuesta de solución.

### 3.1.2. Métrica Relación entre Clases

El resultado de la aplicación de la métrica Relación entre Clases (RC) está dado por el número de relaciones de uso que se establecen entre una clase y las demás clases existentes. Se evalúa a partir de los siguientes atributos de calidad (Lorenz, y otros, 1994):

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

Para obtener un nivel óptimo de relación entre clases, el valor obtenido al aplicar dicha métrica debe ser directamente proporcional al acoplamiento y a la complejidad de mantenimiento; además debe ser inversamente proporcional al nivel de reutilización del código. Para aplicar la métrica RC es necesario categorizar cada una de las clases según la cantidad de relaciones que esta contenga.

A continuación, se muestra una tabla con las categorías para clasificar cada uno de los atributos de calidad anteriormente mencionados, así como el criterio de evaluación. En la misma se emplean la abreviatura Prom (promedio).

**Tabla 5:** métrica RC. Categoría por atributos y criterio de evaluación

| Atributo | Categoría | Criterio |
|---|---|---|
| Acoplamiento | Ninguno | RC = 0 |
| | Bajo | RC = 1 |
| | Medio | RC = 2 |
| | Alto | RC > 2 |
| Complejidad de Mantenimiento | Baja | RC <= Prom |
| | Media | Prom<=RC<= 2* Prom |
| | Alto | RC >2* Prom |
| Reutilización | Baja | RC >2* Prom |
| | Media | Promedio< RC <= 2 *Promedio |
| | Alto | RC <= Promedio |
| Cantidad de pruebas | Baja | RC <= Promedio |
| | Media | Promedio<= RC <2*Promedio |
| | Alto | RC >= 2*Promedio |

**Fuente:** (Lorenz, y otros, 1994)

A continuación, se muestran las medidas de los parámetros de calidad obtenidas luego de aplicar la métrica TOC al diseño de clases de los componentes como resultado de la presente investigación, teniendo en cuenta las categorías por atributos y los criterios de evaluación de la tabla anterior. En el caso de la siguiente tabla se utilizaron las abreviaturas:

- **Ac** para el atributo de Acoplamiento.
- **Cm** para el atributo de Complejidad de mantenimiento.
- **R** para el atributo de Reutilización.
- **Cp** para el atributo Cantidad de prueba.

**Tabla 6:** resultados obtenidos luego de aplicada la métrica RC

| No. | Clase | Cantidad de Relaciones de Uso | Ac | Cm | R | Cp |
|---|---|---|---|---|---|---|
| 1 | documento.component.html | 1 | Bajo | Baja | Alta | Baja |
| 2 | documento.component.ts | 2 | Medio | Media | Media | Media |
| 3 | documento.ts | 0 | Ninguno | Baja | Alta | Baja |
| 4 | documento.service.ts | 0 | Ninguno | Baja | Alta | Baja |
| 5 | formdocumento | 0 | Ninguno | Baja | Alta | Baja |
| 6 | api-rest-routing.yml | 1 | Bajo | Baja | Alta | Baja |
| 7 | DocumentoGtr.php | 2 | Medio | Media | Media | Media |
| 8 | Documento.php | 0 | Ninguno | Baja | Alta | Baja |
| 9 | DocumentoController.php | 1 | Bajo | Baja | Alta | Baja |

| | | | | | | |
|---|---|---|---|---|---|---|
| 10 | BaseGtr.php | 0 | Ninguno | Baja | Alta | Baja |
| 11 | BaseApiController.php | 0 | Ninguno | Baja | Alta | Baja |
| 12 | DocumentoRepository.php | 1 | Bajo | Baja | Alta | Baja |
| 13 | BaseRepository.php | 0 | Ninguno | Baja | Alta | Baja |
| 14 | editarplantilla.component.html | 1 | Bajo | Baja | Alta | Baja |
| 15 | editarplantilla.component.ts | 2 | Medio | Media | Media | Media |
| 16 | app-editor | 1 | Bajo | Baja | Alta | Baja |
| 17 | PlantillaGtr.php | 2 | Medio | Media | Media | Media |
| 18 | PlantillaController.php | 1 | Bajo | Baja | Alta | Baja |

**Fuente:** elaboración propia

Después de haber aplicada la métrica, se tomaron todos los resultados obtenidos individualmente y se agruparon para ser analizados de manera general, promediando los valores obtenidos por categoría en cada atributo, los resultados se muestran en la siguiente figura:

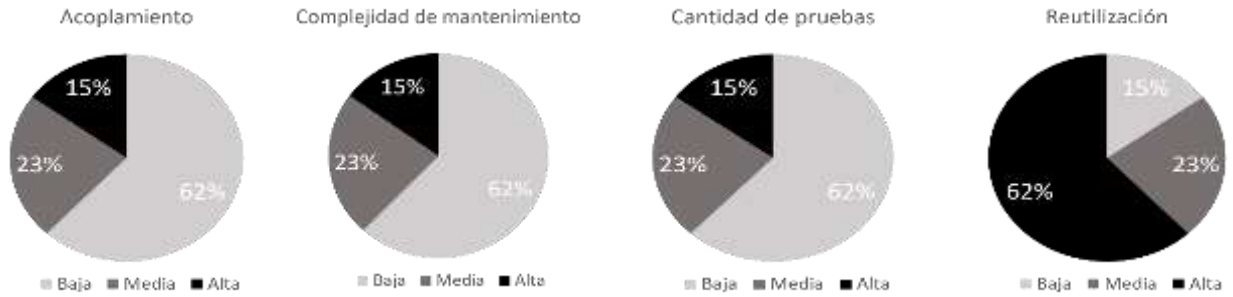


**Figura 18:** resultado de la métrica RC

**Fuente:** elaboración propia

Después de evaluar los resultados obtenidos de cada uno de los atributos de calidad que mide esta métrica, se obtuvieron los siguientes resultados:

- El 62% de las clases tienen un bajo acoplamiento, baja complejidad de mantenimiento y un porciento bajo de cantidad de pruebas.
- El 62% de las clases poseen una alta reutilización.

Como se puede observar en la figura el acoplamiento posee un nivel bajo por lo que existe poca dependencia entre las clases trayendo como consecuencia una alta probabilidad de reutilización. También se puede apreciar que existe un bajo nivel de complejidad de mantenimiento por lo que a la hora de optimizar métodos y demás operaciones no es necesario realizar una gran cantidad de pruebas, lo que minimiza el tiempo de implementación y pruebas de los componentes propuestos.

### 3.2 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 (Rodríguez Sánchez, 2015). A continuación, se detallan las pruebas llevadas a cabo durante esta disciplina, con el objetivo de asegurar la calidad del resultado de la implementación.

### 3.2.1. 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, el autor de la presente investigación define 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[15] (Pressman, 2010). Para aplicar este método se define la técnica de ruta básica, que se describe a continuación:

**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. Esta métrica se calcula sobre un grafo y se puede realizar mediante tres formas distintas:

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 [16] incluidos en el grafo.

Una vez calculada la complejidad ciclomática, el valor obtenido representa el límite superior de pruebas que deberán aplicarse (Pressman, 2010).

En la siguiente figura se muestra el método *eliminarDoc()* de los requisitos: Eliminar documento en

[15] Camino que recorre por lo menos una nueva arista en el grafo de flujo.

[16] Está caracterizado porque dos o más aristas emergen de él.

edición y Eliminar documento en definitivo. Se selecciona el mismo como ejemplo porque es uno de los métodos que al calcular la complejidad ciclomática se obtuvo como resultado mayor a 1, además que en él se ejecutan dos requisitos.

```
public function eliminarDoc($id)
{
    $documento = $this->getEm()->find( className: Documento::class, $id);
    $estadoDocumento = $documento->getTipoEstadoDocumento()->getId();
    $accion = 0;
    if ($estadoDocumento == TipoEstadoDocumento::DEFINITIVO) {
        $documento->setTipoEstadoDocumento($this->getEm()->find( className: TipoEstadoDocumento::class,  id: TipoEstadoDocumento::DESACTIVADO));
        $this->getEm()->persist($documento);
    } else {
        $this->getEm()->remove($documento);
        $accion = 1;
    }
    $this->getEm()->flush();
    return $accion;
}
```

(1) (2) (3) (4) (5) (6) (7)

**Figura 19:** método *eliminarDoc()* utilizado como ejemplo para la técnica de ruta básica

**Fuente:** elaboración propia

A continuación, se realiza el grafo de flujo para el método previamente descrito:

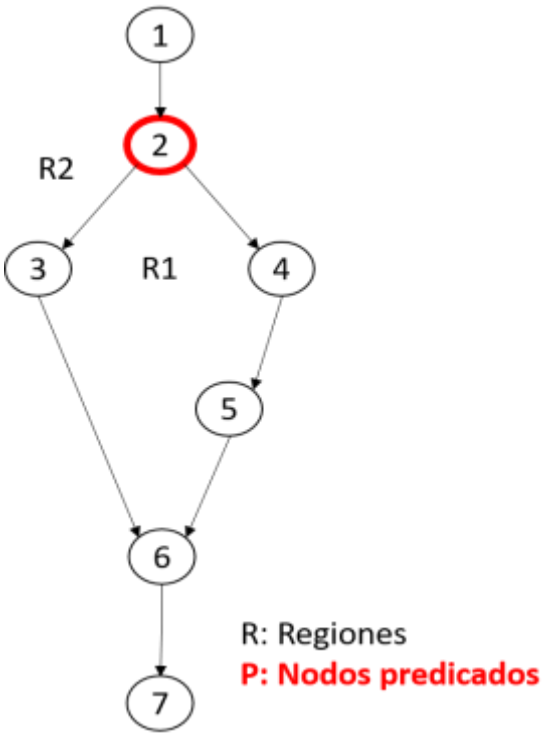


**Figura 20**: grafo de flujo a partir del método *eliminarDoc()*

**Fuente:** elaboración propia

A dicho grafo de flujo se le aplicó la métrica de complejidad ciclomática, calculada por las tres vías conocidas, obteniéndose los siguientes resultados:

- **V(G)** = R = 2
- **V(G)** = 7 – 7 + 2 = 2

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

Después de calcular la complejidad del grafo, se pudo comprobar que los resultados obtenidos son iguales a 2, por tanto, se deben realizar 2 casos de pruebas, uno por cada ruta independiente. Las rutas independientes resultantes fueron:

1) 1-2-3-6-7
2) 1-2-4-5-6-7

En las tablas que se muestran a continuación se encuentran los casos de pruebas correspondientes a cada una de las rutas básicas obtenidas anteriormente.

**Tabla 7:** caso de prueba de la ruta independiente 1

| **Caso de prueba: Ruta independiente #1** | |
|---|---|
| **Descripción:** El documento a eliminar se encuentra en estado definitivo, por cuestiones de seguridad y modo de trabajo de los TPC se desactivan en el sistema. | |
| **Entrada** | El identificador del documento. |
| **Condición de ejecución** | El documento está en estado definitivo. |
| **Resultados esperados** | Se desactiva el documento. |

**Fuente:** elaboración propia

**Tabla 8:** caso de prueba de la ruta independiente 2

| **Caso de prueba: Ruta independiente #1** | |
|---|---|
| **Descripción:** El documento a eliminar se encuentra en estado de edición en el sistema y posteriormente se elimina. | |
| **Entrada** | El identificador del documento. |
| **Condición de ejecución** | El documento está en estado en edición. |
| **Resultados esperados** | Se elimina el documento. |

**Fuente:** elaboración propia

**Resultados al aplicar la técnica de ruta básica**

Esta técnica se aplicó a los métodos de las clases controladoras y gestoras; dichas clases fueron seleccionadas debido a que engloban las funcionalidades del sistema se realizaron pruebas manuales utilizando un modelo ejecutable al eliminar los documentos en el sistema. En el caso de prueba # 1 se introdujo un documento a eliminar con estado definitivo, de tal forma que al hacer la comprobación se desactivara dicho documento. Una vez ejecutado este caso de prueba el resultado fue el esperado. En el caso de prueba # 2, se introdujo un documento con estado en edición, de tal forma que al hacer la comprobación se eliminara dicho documento en el sistema.

Una vez ejecutados todos los casos de pruebas obtenidos en esta técnica, se concluye que los mismos fueron probados satisfactoriamente, demostrando que todas las rutas de este código se ejecutaron al menos una vez.

### 3.2.2. Pruebas funcionales

Las pruebas funcionales son pruebas diseñadas tomando como referencia las especificaciones funcionales de un componente o sistema. Se realizan para comprobar si el software cumple las funciones esperadas (Pressman, 2010).

Para llevar a cabo estas pruebas se tuvo en cuenta el método de Caja negra que a continuación se describe.

**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 de equivalencia que a continuación se describe.

**Técnica de prueba: Partición de equivalencia**

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). Como primer paso en el DCP, se muestra a continuación la descripción de las variables para el caso del requisito funcional “Buscar documento en el expediente”:

**Tabla 9:** descripción de las variables del RF “Buscar documentos en el expediente”

| No | Nombre de campo | Clasificación | Valor Nulo | Descripción |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 1 | Documento | Campo de selección | Si | Campo de selección único que muestra una lista para escoger la clasificación del documento que se quiere filtrar al realizar la búsqueda. |
| 2 | Fecha | Campo fecha | Si | Campo de fecha que permite introducir la fecha de creación del documento al filtrar la búsqueda de los documentos. |
| 3 | Clasificación | Campo de selección | Si | Campo de selección único que muestra una lista para escoger la clasificación del documento que se quiere al filtrar la búsqueda. |

**Fuente:** elaboración propia

Luego de obtener la descripción de cada una de estas variables, se procede a realizar cada uno de los escenarios a probar en el sistema. Para el caso del requisito en cuestión, en el anexo 3 se muestran estos escenarios, el resto se pueden consultar en el artefacto generado (Aguilera González, 2019) que engloba todos los DCP de los requisitos funcionales de los componentes propuestos, los mismo elaborados por el autor de la presente investigación.

Para aplicar el método de caja negra a la solución, se efectuaron un total de 3 iteraciones para poder alcanzar resultados satisfactorios, atendiendo al correcto comportamiento del sistema ante diferentes situaciones. En la figura que se muestra a continuación, se realiza un resumen mediante una gráfica con el número total de No conformidades (NC) por iteración.

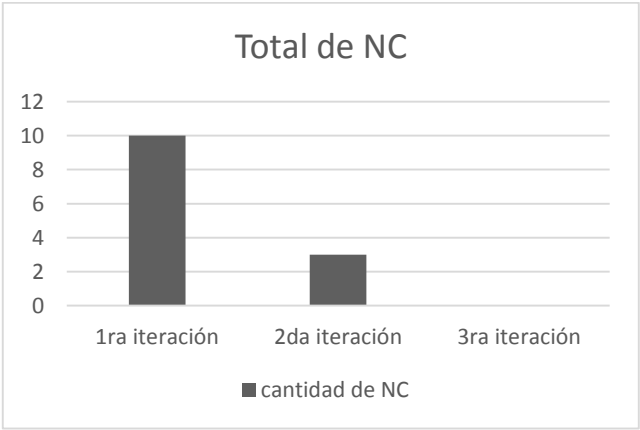


**Figura 21:** total de NC detectadas por cada iteración

**Fuente:** elaboración propia

Estas NC se clasificaron en: Opciones que no funcionan, Ortografía e Interfaz. En las dos primeras iteraciones la tendencia de NC fue de Opciones que no funcionan. En cada una se corrigieron las NC dando paso a que en la tercera iteración no se detectaran NC, lo que trajo consigo que se procediera a la disciplina de pruebas de liberación propuesta por la metodología AUP-UCI. A continuación, se muestra un resumen de las NC por tipo de clasificación en cada iteración de

prueba:

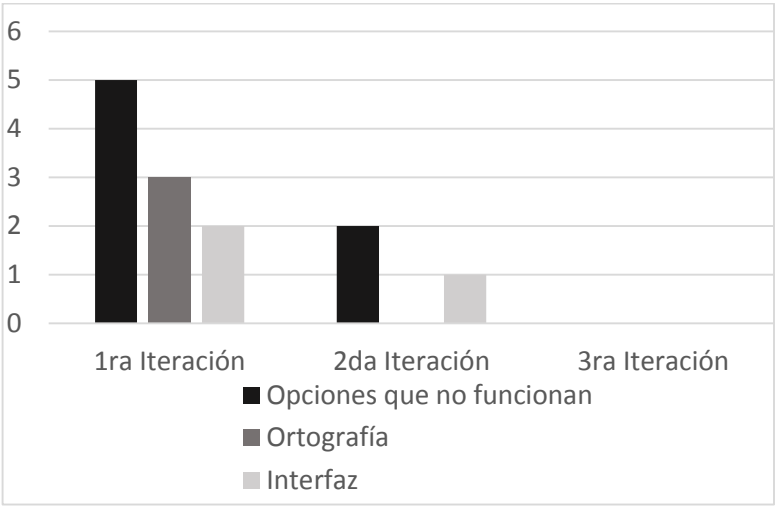


**Figura 22:** total de NC por tipo de clasificación en cada iteración

**Fuente:** elaboración propia

### 3.3 Pruebas de liberación

Esta etapa de prueba es ejecutada por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación (Rodríguez Sánchez, 2015). Para el caso de la solución propuesta se sometió el funcionamiento de ambos componentes a un proceso de pruebas de liberación por el grupo de Calidad CEGEL, encargados de evaluar cada uno de los entregables al cliente. Para este proceso se utilizaron los mismo DCP ejecutados en las pruebas funcionales durante la disciplina de pruebas internas. A continuación, se muestra una figura con las iteraciones y la cantidad de No conformidades detectadas.

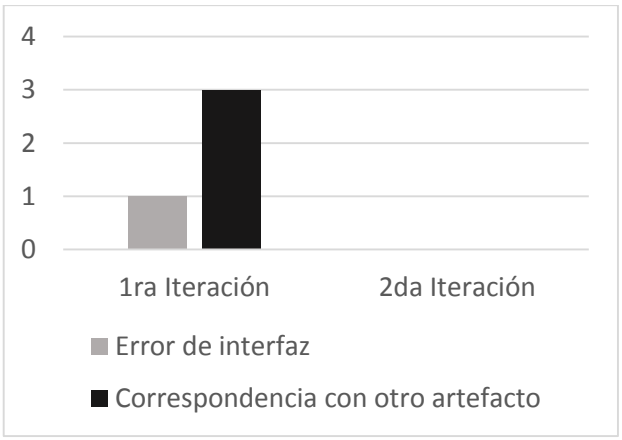


**Figura 23:** total de NC detectadas en las pruebas de liberación

**Fuente:** elaboración propia

La liberación del software llevada a cabo por el grupo de calidad de CEGEL demostró que los componentes cumplían con las necesidades requeridas y estaban listos para ser entregados al cliente. Luego de terminadas las pruebas se emitió el acta de liberación, corroborando que los componentes estaban listos para ser utilizado por el cliente. En el anexo 5 se puede consultar el acta emitida por esta entidad.

### 3.4 Validación de las variables de la investigación

Teniendo en cuenta la idea a defender que se plantea en la presente investigación, para analizar la relación causa efecto entre la variable independiente "como gestionar la documentación de los expedientes judiciales en los Tribunales Populares Cubanos" y la variable dependiente "que contribuya a la disponibilidad y estandarización en el proceso de confección de estos", el autor de la presente investigación define un conjunto de criterios de medida que permiten validar cómo a través de la propuesta de solución se logra la relación entre ambas variables. La obtención de estos criterios se realiza a partir de las principales deficiencias identificadas en la situación problemática y de los resultados de la encuesta realizada a los especialistas funcionales judiciales de los TPC (ver anexo 4).

**Criterios de medida definidos:**

- Acceso simultaneo de los documentos en un expediente: este criterio incide en la disponibilidad de cada uno de los documentos generados o adjuntados a un expediente, ya que al existir una sola copia física puede estar en un solo lugar en un mismo intervalo de tiempo.
- Consulta de los documentos de un expediente: este criterio incide en la disponibilidad de cada uno de los documentos generados o adjuntados a un expediente, facilitando su búsqueda en caso de consultarlo para emitir un criterio sobre el proceso en cuestión.
- Conservación de la documentación asociada a un expediente: este criterio incide en la disponibilidad de la documentación asociada al proceso de un expediente, a través de las diferentes salvas de sistema de respaldo ante fallos del servidor que garantice una copia de la aplicación, así como del estado diario de la base de datos.
- Captura de los datos al registrar un documento: Este criterio influye en la estandarización de la documentación en los TPC, ya que actualmente este proceso al hacerse de forma manual, existe inconsistencia de los datos en un mismo tipo de documento.
- Generación de los documentos: mide la facilidad al generar una plantilla base para cada uno de los documentos, según su clasificación, contribuyendo a la estandarización de la documentación en los TPC.

En la siguiente tabla se evalúan cada uno de los criterios de medida definidos. La evaluación se realiza a través de un antes (TPC sin una herramienta informática que contribuya a la gestión de la documentación de los procesos judiciales) y un después (TPC con una herramienta informática que contribuye a la gestión de la documentación de los procesos judiciales), con el propósito de verificar cómo, mediante la solución propuesta, se contribuye a la disponibilidad y estandarización en el proceso de confección de la documentación en los TPC, asociadas a un proceso judicial. Los datos tenidos en cuenta en la comparación fueron tomados a partir de una encuesta realizada a los especialistas funcionales de los TPC, en una primera presentación de la solución propuesta hacia los especialistas encargados del manejo de la documentación que se genera en los TPC (ver anexo 3).

**Tabla 10:** validación de las variables de investigación

| Atributo | Antes | Después |
|---|---|---|
| Acceso simultaneo de los documentos en un expediente | A partir de la encuesta realizada, el 100% de los encuestados plantean que no es posible el acceso simultaneo de los documentos en un expediente, ya que, al existir una sola copia en formato físico, solo puede estar a la misma vez en un solo lugar, y si este se encuentra entregado a una de las partes, en caso de que la otra parte solicite el mismo documento, según la ley, posee entre 1 y 3 días para su devolución. | El acceso simultáneo se beneficia ya que, al existir un documento digital, pueden acceder al mismo simultáneamente todas las partes que intervienen en el proceso y que tienen acceso a esta documentación. |
| Consulta de los documentos de un expediente | El 56% de los encuestados plantean que la consulta de los documentos de un expediente se dificulta, ya que estos se archivan en grandes estantes de los tribunales, dificultando su proceder al consultarlo para emitir un criterio sobre el proceso en cuestión. | La consulta de los documentos se favorece con la propuesta de solución, ya que no es necesario acceder a un local con grandes estantes a consultar un documento, con solo aplicar criterios de búsqueda, se devolverá el documento que se desea consultar. |
| Conservación de la documentación asociada al proceso de un expediente | Actualmente, según el 64% de los encuestados, los documentos están expuestos al deterioro, ya que su conformación se realiza de forma manual y se conservan en los estantes del tribunal en formato duro, donde, el grado de humedad y otras causas hacen que se deteriore el papel o que se contaminen con bacterias y hongos, una vez infestados los | Se disminuye el uso del papel y el deterioro de la documentación, ya que, al llevar el expediente de forma digital, la documentación se archivaría en un programa informático que permite el almacenamiento y conservación de la misma. Por otra parte, se tendrá en cuenta un sistema de respaldo ante |

| | | |
|---|---|---|
| | archivos, hay que desinfectarlos con productos específicos utilizando químicos o fumigar el local para la desinfección de los documentos, lo que trae consigo riesgos que conlleva el uso de los plaguicidas para la salud y el ambiente. | fallos del servidor que garantice una copia de la aplicación así como del estado diario de la base de datos. |
| Generación de los documentos | De acuerdo al 52% de los encuestados plantean que, debido a la generación manual de los documentos, en muchas ocasiones no conservan las formas, estructuras y estándares dictados por la ley para su creación. | El sistema genera todos los documentos en formato pdf, teniendo como base una plantilla generada por el sistema, una vez conformada por el presidente de sala o del tribunal, conservando las formas, estructura y estándares dictados por la ley para la creación de estos documentos. |
| Captura de los datos al registrar un documento | El 64% de los encuestados plantean que los documentos, al generarse de forma manual y debido a errores humanos, están expuesto a tachaduras, borrones y en algunos de los casos, los datos de la documentación no concuerdan en un mismo expediente. | Al contar con un editor de plantillas, los usuarios manipulan los documentos de manera digital, donde los datos comunes del proceso se generan automáticamente, por lo que se eliminan las tachaduras y los borrones. |

**Fuente:** elaboración propia

La comparación realizada en la tabla anterior, a través de los criterios de medida antes definidos, demuestra cómo, con el desarrollo de los componentes para la gestión de documentos y plantillas en la herramienta informática XEJEL, se contribuirá a la disponibilidad y estandarización en el proceso de confección de estos.

**Conclusiones del capítulo**

En el presente capítulo se realizaron las métricas de validación del diseño, que permitieron definir de forma cuantitativa la calidad del mismo en el desarrollo de los componentes para la gestión de la documentación en XEJEL. Por otra parte, la ejecución de pruebas unitarias para la validación del

código, reflejaron que las operaciones internas se ajustan a las especificaciones y que las funcionalidades internas se ejecutan de forma adecuada. De igual forma, la ejecución de pruebas funcionales, evidenció que las funciones externas son operativas, que las entradas se aceptan de forma adecuada y que se producen salidas correctas, obteniendo finalmente una aplicación que satisface los requisitos definidos. Las pruebas de liberación evidenciaron que los componentes cumplen con las necesidades requeridas para ser entregado al cliente. Por último, la relación causa efecto entre la variable independiente y la variable dependiente, a través de varios criterios de medidas definidos, demostró que con la propuesta de solución se contribuye a la disponibilidad y estandarización de la documentación en el XEJEL.

# Conclusiones generales

Al concluir la investigación para el desarrollo de los componentes para la gestión de documentos y plantillas en la herramienta informática Expediente Judicial Electrónico, se pudo arribar a las siguientes conclusiones:

- La identificación de los referentes teóricos en los que se sustenta la propuesta de solución sobre el desarrollo de componentes informáticos dirigidos a la gestión de la documentación en expedientes judiciales electrónicos permitió sentar las bases de la investigación.
- La especificación de los requisitos, así como el análisis y diseño de los componentes de documentos y plantillas, permitieron obtener una aproximación y concebir los elementos necesarios para la implementación de estos componentes.
- La implementación de los componentes de documentos y plantillas favoreció la obtención un módulo funcional, contribuyendo a la disponibilidad y estandarización en el proceso de confección de estos.
- La validación de los resultados obtenidos a través de las métricas de validación del diseño y las pruebas de software permitió comprobar de forma cuantitativa la calidad de los artefactos obtenidos durante el desarrollo de la solución propuesta.
- La validación de las variables de la investigación permitió corroborar cómo se contribuye a la disponibilidad y estandarización de la documentación en el XEJEL.

# Recomendaciones

Se recomienda la integración de los componentes obtenidos como resultados de la presente investigación, con los componentes de:

- Datos generales del expediente, de forma tal que se logre asociar documentos a un expediente desde que se registra y cuando se tramita en el XEJEL.
- Notificación, de forma tal que cada vez que se tramite un expediente, se les notifique a las partes que intervienen en el proceso sobre dicha tramitación.

# Referencias bibliográficas


- **3schools.com. 2019.** 3schools.com. [En línea] 2019. https://www.w3schools.com/bootstrap4/.
- **Aguilera González, Sergio Orlando. 2019.** *Diseño de casos de prueba. Componente documento .* La Habana : Universidad de las Ciencias Informáticas, 2019.
- **—. 2018.** *Historias de usuario. Componentes documentos y plantillas para el XEJEL.* La Habana : Universidad de las Ciencias Informáticas, 2018.
- **Aleaga, Yaiset Moreno. 2018.** *CEGEL-SITPC-Estándares de Codificación para PHP.* 2018.
- **Álvarez, Daniel José Salas. 2010.** Estándares de codoficacion Java. [En línea] 2010. http://www.aves.edu.co/ovaunicor/recursos/view/265.
- **Álvarez, Miguel Angel. 2014.** desarrolloweb.com. [En línea] 2014. https://desarrolloweb.com/articulos/que-es-mvc.html.
- **Angular. 2018.** Angular.io. [En línea] 2018. https://angular.io/guide/architecture.
- **Berciano Alonso, Alonso. 1999.** [En línea] 1999. http://platea.pntic.mec.es/~abercian/guiahtml/comienzo.htm.
- **Castro Morell, Daniel E. y González Guadarrama, José R. 2008.** *Sistema para la tramitación de procesos penales.* 2008.
- **Cavenago, Almeira AS. Pérez. 2007.** *Arquitectura de software: Estilos y Patrones.* Argentina : Universidad Nacional de la Patagonia San Juan Bosco, 2007.
- **Cillero, Manuel. 2017.** Diagrama de componentes. [En línea] 2017. https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-componentes/.
- **Crespo, Álvaro González. 2016.** Qué es GitLab. [En línea] 2016. [Citado el: 19 de enero de 2019.] https://es.linkedin.com/learning/gitlab-esencial/que-es-gitlab.
- **DataPrinx.** Introducción al diseño de bases de datos. [En línea] http://www.dataprix.com/1-introduccion-diseno-bases-datos.
- **del Castillo San Félix, Alvaro. 2000.** El servidor de web Apache: Introducción práctica: Apache 1.x y 2.0 alpha. [En línea] 2000. http://acsblog.es/articulos/trunk/LinuxActual/Apache/html/x31.html.
- **Doctrine-project.org. 2018.** Doctrine project. What is Doctrine? [En línea] 2018. [Citado el: 14 de Noviembre de 2018.] http://docs.doctrine-project.org/en/latest/tutorials/getting-started.html.

- **Franco Espiño, Beatriz y Pérez Alcázar, Ricard. 2015.** Redrta.org. *Redrta.org.* [En línea] 2015. http://mgd.redrta.org/directrices-administracion-de-documentos-electronicos/mgd/2015-01-27/161143.html.
- **Gamma, Erich, y otros. 1994.** *"Design Patterns: Elements of Reusable Object Oriented Sofware".* s.l. : Grady Booch, 1994.
- **González Ochoa, Darián y Domínguez González, Yosviel. 2018.** *SITPC, una herramienta informática cubana para la administración de la justicia.* La Habana : s.n., 2018.
- **Guerra, Enrique Fernandez. 2016.** Desarrolloweb.com. [En línea] 2 de junio de 2016. [Citado el: 30 de noviembre de 2018.] https://desarrolloweb.com/articulos/introduccion-a-typescript.html.
- **Illescas, Eduardo Díaz-Meco. 2016.** Tc-abodagos.es. [En línea] 15 de junio de 2016. [Citado el: 30 de noviembre de 2018.] https://tc-abogados.es/que-es-un-documento-electronico/.
- **ISO/IEC. 2005.** *INGENIERÍA DE SOFTWARE - CALIDAD DEL PRODUCTO. PARTE 1: MODELO DE LA CALIDAD.* La Habana : Cuban National Bureau of Standards, 2005.
- **IT-CGAE. 2013.** *Manuel de usuario. LEXNET.* s.l. : Red Abogacía Española, 2013. pág. 25.
- **Juiz, O. 2009.** *INFORME DEL SISTEMA SISECO.* 2009.
- **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.
- **Lorenz, M. y Kidd, J. 1994.** *Object-Oriented Software Metrics.* s.l. : Prentice-Hall, 1994.
- **Lucid Software Inc. 2018.** Lucidchart. [En línea] 2018. https://www.lucidchart.com/pages/es/qu%C3%A9-es-la-notaci%C3%B3n-de-modelado-de-procesos-de-negocio.
- **MJU, Ministerio de Justiica de España. 2012.** *EXPEDIENTE JUDICIAL ELECTRÓNICO Informes de Modernización Judicial en España.* España : Ministerio de Justicia, 2012.
- **Muñoz Serafin, Miguel. 2018.** *Introducción al desarrollo de aplicaciones N-Capas con tecnologías Microsoft.* 2018.
- **Murua, Juan, Fernández, Alejandro y Eguiluz, Javier. 2019.** *Pro Git, el libro oficial de Git.* 2019.
- **Ochando, Blázquez. 2014.** Modelo entidad-relación. Fundamentos y Diseño de Bases de Datos. [En línea] 2014. http://ccdoc-basesdedatos.blogspot.com/2013/02/modelo-entidad-relacion-er.html.
- **ohmyroot. 2017.** Estandares de codificación. [En línea] 12 de enero de 2017. https://www.ohmyroot.com/buenas-practicas-legibilidad-del-codigo/.

- **Oracle. 2018.** Netbeans.org. *NetBeans IDE Features.* [En línea] 2018. [Citado el: 20 de Febrero de 2019.] https://netbeans.org/features/index.html.
- **Pablo, José. 2017.** ¿Qué es y para qué sirve un modelo conceptual? [En línea] 2017. http://www2.chj.gob.es/albufera/01_WEB_ED/01_AV_DSAV/04_GA/01_MC/9-Metodologia/9-4_Fase_3.htm.
- **Pacheco, Nacho. 2013.** Documentación de Symfony. [En línea] 2013. http://gitnacho.github.io/symfony-docs-es/.
- **Pérez Porto, Julián y Gardey, Ana Gardey. 2018.** Definición.de. *Definición.de.* [En línea] 2018. https://definicion.de/estandarizacion/.
- **Pérez Valdés, Damián. 2007.** maestrosdelweb. *maestrosdelweb.* [En línea] 3 de julio de 2007. http://www.maestrosdelweb.com/que-es-javascript/.
- **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. 2010.** *Ingeniería de Software. Un enfoque práctico.* Septima . México DF : McGraw-Hill INTERAMERICA EDITORES, 2010.
- **Rodríguez Sánchez, Tamara. 2015.** *Metodología de desarrollo para la Actividad productiva UCI.* La Habana : s.n., 2015.
- **Sæther Bakken, Stig, Aulbach, Alexander y Schmid, Egon. 2002.** *Manual de PHP.* 2002.
- **Sommerville, Ian. 2011.** *Ingeniería de Software.* 9na. México : Addison-Wesley, 2011. ISBN: 978-607-32-0603-7.
- **SRL, E. S. J. 2011.** Lex-Doctor, GESTIÓN JURÍDICA. [En línea] 2011. http://www.lexdoctor.com/productos_estudiosjuridicos_info.php.
- **Tecnologías-Información. 2018.** Modelo de datos. [En línea] 2018. https://www.tecnologias-informacion.com/modeladodatos.html.
- **TENSTEP.INC. 2016.** [En línea] 2016. https://www.tenstep.ec/portal/articulos-boletin-tenstep/41-scrum/253-scrum-como-escribir-historias-de-usuarios-sin-morir-en-el-intento.
- **Velasco, Rubén. 2017.** PHP 7.2, todas las novedades de esta nueva versión de PHP. [En línea] 13 de noviembre de 2017. [Citado el: 20 de septiembre de 2018.] https://www.redeszone.net/2017/11/13/novedades-php-7-2/.
- **Visual Paradigm. 2019.** [En línea] 2019. http://www.visual-paradigm.com/.