Universidad de las Ciencias Informáticas

Facultad 3

Desarrollo del módulo Revisiones Laborales para el proyecto Sistema de Informatización de la Gestión de las Fiscalía fase II.

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

Autor:

Osvey Sosa Quintana

Tutor: Ing. Yenier Figueroa Machado

Co-tutor: Ing. Annaliet Parra Pérez

Ing. Barbara I González Jorge

Ciudad de La Habana, Curso 2012-2013

Debemos avanzar hacia una explosión masiva del conocimiento, de tecnología de innovación, en función de las necesidades sociales y económicas del país y de la soberanía nacional.

Hugo Chávez Frías

Datos de contacto

Tutor: Ing. Yenier Figueroa Machado

Correo electrónico: yfigueroa@uci.cu

Autor: Osvey Sosa Quintana

Correo electrónico: oquintana@estudiantes.uci.cu

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo al Centro de Gobierno Electrónico de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Firma del Autor: _____. Firma del Tutor: _____.

Resumen

El presente trabajo tiene como objetivo el desarrollo del módulo de Revisiones Laborales perteneciente al departamento de Protección de los Derechos Ciudadanos, para lograr la informatización de los procesos laborales en la Fiscalía General de la República, dando solución a los diferentes problemas que imposibilitan un trabajo eficiente por parte de los fiscales. Para una mejor comprensión de los lectores el trabajo fue dividido en tres capítulos: fundamentación teórica, requisitos y diseño, y por último implementación y prueba. A lo largo del desarrollo de la investigación se reflejan las diferentes actividades realizadas para lograr el objetivo trazado, entre dichas actividades se destacan: el análisis sobre los procesos laborales y los diversos software existentes capaces de lograr informatizar dichos procesos, el estudio sobre las diferentes herramientas y lenguajes que ayudaron a agilizar el desarrollo de la aplicación, la captura de los requisitos a cumplir por parte del sistema a desarrollar y los diferentes diagramas obtenidos en las fases de diseño e implementación. Una vez concluido el desarrollo de la aplicación final, fue validado mediante las pruebas de caja blanca, caja negra y las pruebas de usuario, validando mediante la última las variables de la investigación del trabajo.

Con el desarrollo del software requerido, el autor espera que los fiscales tengan a su disposición una aplicación que los ayude en su trabajo diario, permitiéndoles resolver sus casos en tiempo y forma, además de poder realizar varias labores en el estrecho margen de tiempo con que cuentan, logrando así su trabajo ser mucho más productivo que antes.

Palabras claves: dictamen, diseño, fiscales, implementación, módulo, procesos laborales, promovente, requisitos, sentencia, software.

Índice

Índice de Figuras

Índice de Tablas

Índice de Anexos

Introducción

En la actualidad el mundo está experimentando un cambio debido a la convergencia de las Tecnologías de la Informática y las Comunicaciones (en lo adelante TIC). Dichas tecnologías constituyen el núcleo central de las transformaciones multidimensionales que afrontan la economía y la sociedad, imponiéndoles a las personas modificar no sólo sus hábitos y patrones de conducta, sino incluso, su forma de pensar (1). Las TIC ofrecen un amplio marco de posibilidades, fundamentalmente en el desarrollo de software para la informatización de la sociedad en que vivimos, trayendo como consecuencia un aumento de la productividad y una mejor gestión de las empresas u organizaciones.

Varias son las empresas u organizaciones que lograron informatizar sus procesos mediante aplicaciones, logrando mejoría en los mismos, siendo muchas de estas entidades las encargadas de los procesos realizados en el gobierno. Muchos son los países que han llevado el desarrollo del gobierno electrónico para conseguir una mejor administración mediante la transparencia y el acceso público a la información. Algunos de estos países dispuestos a reforzar la asociación fundamental entre el sector público y sus ciudadanos son: Dinamarca, España, Gran Bretaña y Francia. En América Latina también se ha evidenciado el interés de gestionar sus procesos internos y mostrarlo de cara a los ciudadanos, en países como Colombia, Brasil y Chile.

Cuba no está exenta de estos cambios y ha venido realizando estrategias para lograr la informatización de su sociedad también. Una de estas estrategias es la creación de la Universidad de las Ciencias Informáticas (UCI) y otras entidades, encargadas de la difícil tarea de la informatización de diferentes sectores de la sociedad. En dicha universidad se encuentra la existencia del Centro de Gobierno Electrónico (CEGEL), cuyo objetivo es la creación de soluciones informáticas para dar solución principalmente a problemas nacionales. Uno de los proyectos en desarrollo de este centro es lograr un sistema para la informatización de todos los procesos que se llevan a cabo en la Fiscalía General de la República (en lo adelante FGR) y así lograr agilizar los procesos legales en las áreas que se encuentran en dicha institución.

La FGR es el “órgano del estado al que corresponde el control y la preservación de la legalidad, sobre la base de la vigilancia del estricto cumplimiento de la constitución, las leyes y demás disposiciones legales” (2). Una de las 8 direcciones contempladas dentro de la FGR es el de Protección de los Derechos de los Ciudadanos (PDC), en la cual se encuentra el departamento de Asuntos Civiles, Administrativos y Laborales (desarrollándose en este último los procesos de Revisiones Laborales) (3). Dentro de la

estrategia de informatización de la FGR está el desarrollo de un software para agilizar los procesos laborales y darle solución a los siguientes problemas que imposibilitan el trabajo diario por parte de los fiscales:

- La aplicación existente en la fiscalía no cuenta con un módulo que ayude a los fiscales en los procesos laborales.
- Los procesos son llevados a cabo manualmente, generando una gran cantidad de documentos, por lo que el volumen de información para procesar es muy alto y afecta considerablemente en tiempo y esfuerzo a los fiscales.
- Las consultas a los documentos son engorrosas y es difícil su acceso futuro afectando la disponibilidad de la información.
- La obtención de la información para la toma de decisiones se realiza de manera manual, lo que afecta la capacidad de obtención de la misma en el tiempo requerido. Además se afecta la integridad y confiabilidad de la información para los análisis que se requieren por los niveles de dirección del órgano.
- La planificación de las actividades a largo plazo es realizada de manera empírica, provocando la mayoría de las veces un considerable atraso en los calendarios y afectando el cumplimiento de las demás actividades previstas a desarrollar durante el flujo del proceso.
- Se solapan actividades que la metodología de trabajo propone que deben realizarse de manera escalonada, afectando el cumplimiento de los procedimientos establecidos en las metodologías de trabajo para cada una de las instancias de la fiscalía.
- No existe un estándar para la documentación generada por cada uno de los procesos en las diferentes instancias de la fiscalía, lo que provoca, en ocasiones, que la información no sea suficiente para la tramitación del proceso.

Bajo los hechos planteados es identificado el siguiente **problema a resolver:** ¿Cómo mejorar la gestión del proceso de Revisiones Laborales, de manera que contribuya al control y celeridad del mismo?

De donde se obtiene el **objeto de estudio**: proceso de desarrollo de software de gestión en la informática jurídica.

En función de resolver el problema propuesto se ha definido como **objetivo general**: desarrollar el módulo de Revisiones Laborales para mejorar la gestión del proceso, enmarcándose en el **campo de acción**:

proceso de desarrollo de software de gestión en los procesos fiscales.

Teniendo como **idea a defender** que si se desarrolla el módulo de Revisiones Laborales del SIGEF II se mejora la gestión del proceso, contribuyendo al control y celeridad del mismo.

Para darle cumplimiento al objetivo general propuesto se han definido los siguientes **objetivos específicos:**

1. Elaborar el marco teórico de la investigación mediante un análisis del estado del arte alrededor de los procesos fiscales, metodologías de desarrollo, técnicas y herramientas de software.
2. Realizar el análisis y diseño del módulo Revisiones Laborales del SIGEF II, mediante la metodología de desarrollo de software establecida.
3. Implementar el módulo Revisiones Laborales del SIGEF II, mediante las herramientas y tecnologías seleccionadas.
4. Validar los requisitos, el diseño y la implementación de la solución propuesta mediante métricas y pruebas de software.

Para desarrollar el presente trabajo, se definieron las siguientes **tareas de la investigación** para dar cumplimiento a los objetivos específicos:

1. Análisis del estado del arte de sistemas de gestión de informática jurídica.
2. Análisis del proceso de desarrollo de software en el proyecto Sistema de Informatización de la Gestión de las Fiscalías fase II.
3. Análisis del proceso Revisiones Laborales del área Protección de los Derechos Ciudadanos.
4. Identificación, descripción y validación de los requisitos asociados al módulo Revisiones Laborales del SIGEF II.
5. Modelado de los diagramas de clases y secuencias asociados al módulo Revisiones Laborales del SIGEF II.
6. Implementación de las funcionalidades del módulo Revisiones Laborales del SIGEF II.
7. Diseño de los casos de prueba asociados al módulo Revisiones Laborales del SIGEFII.
8. Ejecución de las pruebas de software al código fuente obtenido en la implementación del módulo

Revisiones Laborales del SIGEF II.

9. Validación de los resultados obtenidos a través de pruebas de caja negra a las funcionalidades del módulo Revisiones Laborales del SIGEF II, a fin de comprobar su correcto funcionamiento.
10. Validación de la propuesta de solución de la investigación frente al problema planteado.

Métodos teóricos:

Analítico-Sintético: se emplea para el análisis de toda la información referente a la investigación, y así llegar a obtener conclusiones necesarias del trabajo.

Histórico-Lógico: para el estudio de toda la bibliografía y documentación existente de los procesos laborales en la fiscalía, así como los sistemas de gestión existentes relacionados.

Modelación: para la elaboración de los diferentes diagramas que ayudaron a una mejor comprensión de la lógica del proceso, así como de las funcionalidades que debe cumplir el sistema.

Métodos empíricos:

Entrevista: para la recopilación de información, con el fin de entender el flujo de trabajo, llegar a una comprensión del negocio y recoger los requisitos que el sistema debe cumplir.

Estructura del documento

Capítulo 1. Fundamentación teórica. El capítulo 1 refleja el análisis de algunos de los sistemas de gestión fiscal desarrollados en el mundo, además de las herramientas, metodologías y lenguajes usados para poder darle solución al problema planteado. También se exponen los principales conceptos sobre el negocio y los métodos utilizados para el análisis y desarrollo de la solución propuesta.

Capítulo 2. Requisitos y Diseño. El capítulo 2 muestra los requisitos funcionales y no funcionales con los cuales debe cumplir el sistema a desarrollar, así como la especificación de los requisitos y su validación. Por otra parte también se visualiza el diseño realizado, los diagramas de clases necesarios como el de clases persistentes, el de la vista, el de las clases controladoras, el de clases del repositorio y los diagramas de secuencia, haciendo uso de los patrones de diseño. También contiene la validación del diseño mediante métricas propuestas por Pressman.

Capítulo 3. Implementación y Prueba. El capítulo 3 abarca el desarrollo de todo el trabajo correspondiente con la implementación del sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ejecutables y similares. También se muestran los resultados de las pruebas

realizadas al sistema, principalmente las pruebas de caja blanca y caja negra, las cuales validarán la solución propuesta en esta investigación.

1. Capítulo I. Fundamentación Teórica

1.1. Introducción

En el presente capítulo se tratan todos los elementos teóricos que se tuvieron en cuenta a lo largo de este trabajo para lograr realizar las fases de requerimientos, diseño, implementación y pruebas del sistema a desarrollar. También quedan recogidos los conceptos fundamentales, principalmente del negocio, permitiendo un mejor entendimiento de los procesos en el área de las revisiones laborales. Se recogen las características principales de varios software jurídicos desarrollados en el mundo que sirvieron de análisis para llegar al desarrollo del sistema requerido. Por otra parte, también contiene un análisis de las tecnologías, herramientas, metodología y lenguajes para lograr el desarrollo del mismo.

1.2. Conceptos fundamentales

Persona jurídica: es un sujeto de derechos y obligaciones que existe físicamente pero no como individuo, sino como institución y que es creada por una o más personas físicas para cumplir un objetivo social que puede ser con o sin ánimo de lucro (4).

Proceso judicial: conjunto de actuaciones que realiza un tribunal de justicia en un procedimiento judicial desde su inicio hasta que se dicta sentencia (4).

Proceso penal: es el conjunto de actos conforme a los cuales el juez, aplicando la ley, resuelve el conflicto de intereses sometido a su conocimiento por el Ministerio Público (4).

Revisiones Laborales: “es una revisión de una sentencia firme dictada en otro proceso, por la existencia de causas que la hacen injustas. Las revisiones solo se realizan en la FGR de conjunto con el Tribunal Supremo Popular” (5).

Promovente: persona natural que presenta una queja o petición (5).

Fiscal: “es quien representa y ejerce el ministerio público en los tribunales”. En las causas criminales, ejerce la acción acusatoria pública. En general, vela por la pureza del procedimiento, es decir, por el cumplimiento de todos los requisitos y actuaciones previstas por la ley (5).

Dictamen: es un juicio que se emite sobre una cosa (5).

Ampliación: la ampliación en el ámbito laboral es un documento solicitado por el Tribunal Supremo Popular aclarando varios datos faltantes o que no se entendieron del dictamen (5).

Sentencia: aquella en que el juzgador, concluido el juicio, resuelve finalmente sobre el asunto principal (5).

Auto: forma de resolución judicial, fundada, que decide cuestiones secundarias, previas, incidentales o de ejecución, para las que no se requiere sentencia (5).

1.3. Escenario actual del proceso

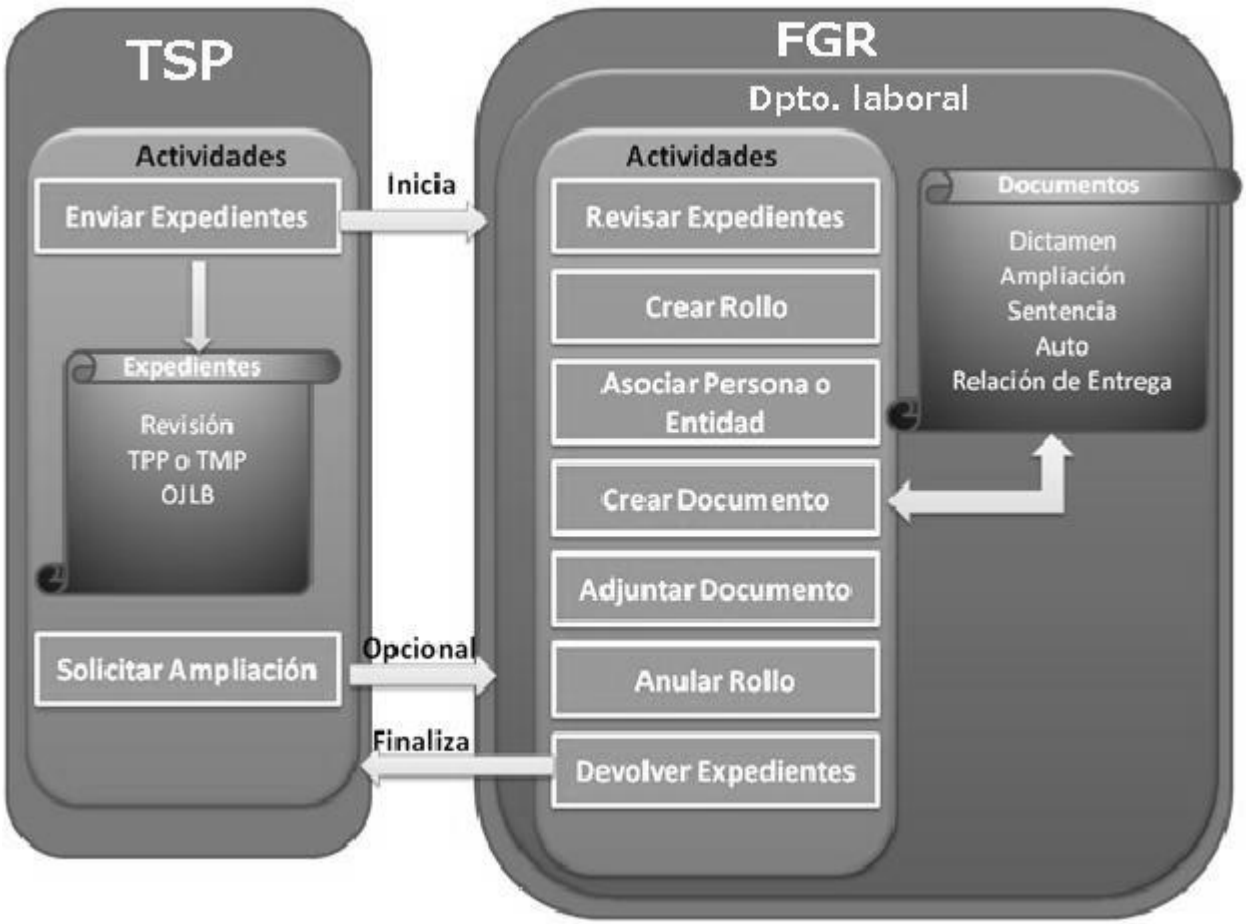


Figura 1: descripción del proceso de Revisiones Laborales

En materia laboral el único proceso que se realiza en la fiscalía es la de revisión laboral, el cual se encarga de todas las revisiones solicitadas por el tribunal de la FGR, las mismas son realizadas una vez que el Tribunal Supremo Popular (en lo adelante TSP) encuentre, o se le presenten pruebas por parte de

una institución o personal inmerso en el proceso fiscal anterior, en donde se dio una sentencia por parte del tribunal popular provincial o municipal (en lo adelante TPP y TMP respectivamente), y dichas pruebas puedan dar lugar a un cambio en la sentencia de este proceso. Las revisiones laborales son promovidas solo por el TSP, por lo que se inicia el proceso en la FGR una vez que llega del TSP un expediente de revisión. De conjunto con este expediente, llega también el expediente del proceso que se va a revisar (expediente del TPP o TMP, y el expediente del Órgano de Justicia Laboral de Base (en lo adelante OJLB)). Del expediente del OJLB solo interesa saber si está o no dicho expediente, y del expediente del TPP o TMP (uno o el otro según donde se haya desarrollado el proceso laboral que se va a revisar).

El fiscal crea un rollo para almacenar todo lo referente al proceso, en este rollo quedan registrados todos los datos anteriores y los documentos que se generen en el proceso. La principal función del fiscal en este proceso es emitir su criterio al respecto mediante un dictamen, para lo que cuenta con solo 7 días para, pasado ese tiempo el proceso continúa su curso normal en el TSP. Una vez que el fiscal emite el dictamen, lo envía al TSP. En casos excepcionales si el TSP considera que es necesario argumentar alguna cuestión del dictamen, solicita al fiscal la ampliación de dicho dictamen. El fiscal puede realizar tantas ampliaciones como solicite el TSP (no se pueden realizar ampliaciones en paralelo). Cuando el fiscal termina de estudiar el proceso y ha emitido el dictamen, devuelve los expedientes al TSP. Para ello registra los expedientes en un documento para que quede como constancia que han sido entregados. En este documento pueden ir relacionados más de un proceso de revisión. Dicho documento es llevado al TSP de conjunto con los expedientes para que sea firmado. Una vez que el TSP emite la sentencia del proceso de revisión, envía una copia a la fiscalía. En casos excepcionales en los que haya existido algún error o haya faltado algún elemento en la sentencia, el TSP emite un Auto aclarando la situación dada y envía una copia a la fiscalía. Si el fiscal considera relevante lo tratado en la sentencia puede realizar un comentario al respecto. En casos de que la revisión no siga su curso por algún motivo los rollos no pueden ser eliminados, solo son anulados dejando constancia de la existencia del proceso desarrollado (5).

1.4. Valoración del estado del arte.

1.4.1. Aplicaciones existentes

En todas partes del mundo los gobiernos se han interesado desde un inicio en tener sus sistemas de gestión para la informatización de sus procesos fiscales, garantizando así eficiencia en los procesos que realizan. Entre las aplicaciones estudiadas se encuentran los siguientes:

Legal Advisor:

En Argentina se desarrolló **Legal Advisor (LEAD),** que consiste en un sistema informático jurídico que tiene por objeto asistir a los operadores de la justicia como jueces, fiscales y defensores del fuero penal en el proceso de individualización de la pena. Fue implementado para actuar sobre tres tipos de delitos pero puede ser extendido a otros procesos penales (6). Algunas de sus características son:

- ✓ Predictivo: Estima la posible pena a otorgar para un determinado caso.
- ✓ Explicativo: Justifica las decisiones con jurisprudencia.
- ✓ Adaptativo: Corrige los criterios de decisión en función de los resultados reales.
- ✓ Pragmático: Tiene utilidad práctica para los operadores judiciales (6).

National Criminal Justice Reference Service:

En EE.UU se fundó en 1972 el NCJRS (National Criminal Justice Reference Service), financiado con recursos federales, que ofrece información sobre el abuso de sustancias con el fin apoyar la investigación, la política y el desarrollo de programas en todo el mundo. En este sitio se pueden encontrar información sobre eventos relacionados con la justicia como conferencias, librerías con información criminal, ofrece información sobre recursos legislativos, publicaciones, programas y da la posibilidad de hacer preguntas y obtener respuestas (7).

Lex-Doctor:

Lex-Doctor es el sistema de gestión jurídica más difundido en toda América Latina, adaptándose al procedimiento de todas las jurisdicciones. Este sistema permite desde la informatización de una oficina judicial individual, hasta la informatización integral del Poder Judicial, abarcando la gestión de causas, la publicación de expedientes para consulta en línea y en tiempo real, con tecnología de última generación, escalable y adaptable a las necesidades de cada organismo (8).

Los sistemas anteriormente anunciados fueron desarrollados para lograr eficiencia en todos los procesos jurídicos que abarcan su sociedad, y que les permita un mejor control y una mayor calidad en el desarrollo de los mismos. Estos sistemas son muy buenos en el trabajo para el que fueron desarrollados, pero nuestro país no puede darle utilidad ya que son software privativos, algunos son solo de información y no de gestión, y cada uno de ellos responden mayormente a las especificidades del sistema judicial del país donde fue desarrollado. Además no cuentan con todas las funcionalidades que requiere el sistema judicial

cubano, como manejar procesos relacionados con las revisiones laborales, buscar documentos por diferentes criterios o generar reportes de los rollos confeccionados de las revisiones laborales realizadas.

1.5. Características del proyecto SIGEF II

El proyecto SIGEF II está dirigido a adicionar nuevas funcionalidades al Sistema de Informatización de la Gestión de las Fiscalías fase I antes implementado, lo que permitirá aumentar el nivel de informatización de los procesos fiscales, condicionando un incremento en la calidad de la tramitación, supervisión y control en tiempo real de los procesos fiscales. Además, una reducción de los términos de las actividades y diligencias practicadas, mayor control de la observancia de la garantía de los procesados, utilización óptima de la fuerza fiscal, así como la utilización de documentos en formato duro (3).

La conformación de la plataforma para el desarrollo será Linux Apache Postgres PHP, la cual se selecciona tomando como punto de partida los requisitos no funcionales, de manera que se tribute a la política de migración hacia software libre que tiene la FGR, enmarcada en la estrategia del país que tiene su base en la soberanía tecnológica (3).

Por varias características identificadas se define que en el desarrollo de la aplicación cada módulo hará uso del patrón modelo vista controlador (MVC).

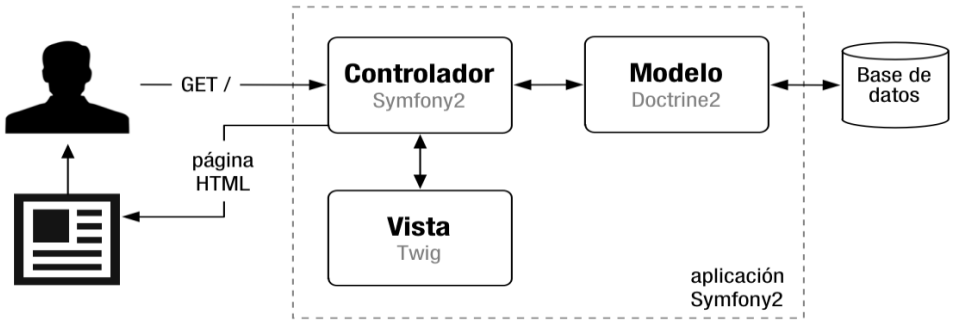


Figura 2: representación vertical de la arquitectura de cada módulo.

El proyecto SIGEFII cuenta con la implementación de un subsistema denominado **Arquitectura Base** de apoyo a la hora del desarrollo de los distintos módulos, recogiendo el diseño e implementación de todas las funcionalidades comunes en los módulos. Estas funcionalidades son representadas en componentes, entre los cuales se encuentran: componentes para la creación de objetos entre niveles y módulos,

asegurar un manejo transaccional adecuado, el control de excepciones, trazas, mensajería, las funcionalidades visuales que se identifique como recurrentes, garantizar la integrabilidad, seguridad y enrutamiento. De esta manera no solo se reduce la implementación a los requisitos funcionales básicos, sino que el código es altamente reutilizable y se limitan los errores o vulnerabilidades en el mismo (9).

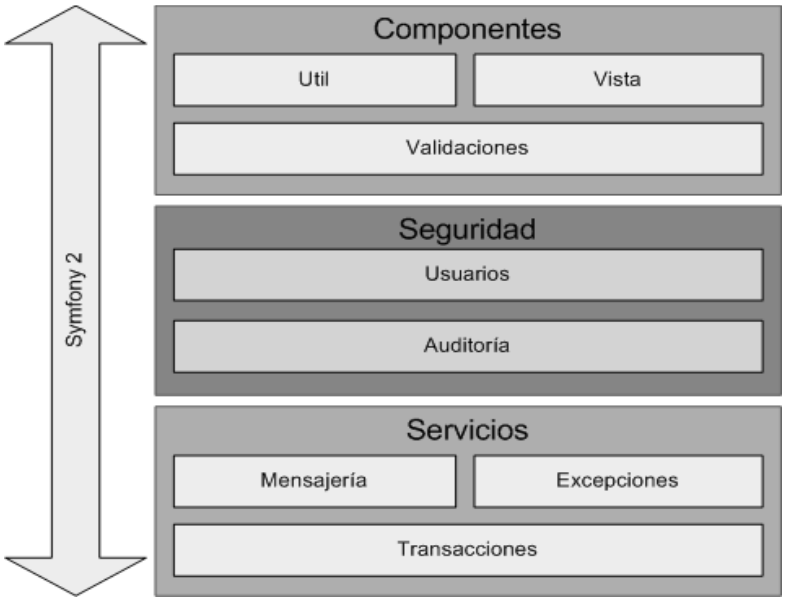


Figura 3: representación en capas del subsistema de arquitectura base.

Este subsistema está formado por componentes donde su primer nivel es precisamente una capa con ese nombre, la que encapsula una serie de funcionalidades de uso común en toda la aplicación relacionadas con la capa de presentación de cada módulo, además de configuraciones para el uso de variables globales y validaciones. La capa de seguridad se encarga de la gestión de usuarios, mostrar información de las trazas de usuarios y aspectos importantes como el control de acceso a cada objeto de la aplicación. El siguiente nivel se encarga de garantizar el cumplimiento de conceptos paralelos a todo el sistema como la creación de servicios, mensajería, tratamiento de excepciones y manejo transaccional garantizando la integridad de los datos (9).

1.5.1. Modelo de desarrollo de software

La selección de un modelo de desarrollo a utilizar depende del tipo de aplicación y del contexto de desarrollo, si se tienen los requerimientos bien definidos o si lo que se quiere es asegurar mayor robustez

durante el proceso de desarrollo.

Teniendo en cuenta un estudio y análisis de la utilización de metodologías y modelos de desarrollo de software en el mundo, en conjunto con las características que posee el proyecto SIGEG II en estos momentos, se decide emplear como guía para el desarrollo de la aplicación propuesta el modelo de desarrollo de software utilizado en CEIGE. Este modelo constituye un caso de éxito, incorporando los distintos subprocesos dictados por el nivel dos de CMMI, certificación obtenida por dicho centro en julio de 2011 y reconocida por el SEI (Software Engineering Institute). Dicho modelo tiene como principios la orientación a dominios del negocio, la especialización y reutilización de los recursos humanos en varios proyectos y el aumento de la productividad. El conjunto de fases propuesta abarca el total de acciones que se realizan en las distintas líneas de desarrollo para la elaboración del servicio o producto final, sin embargo, se debe adaptar a las características particulares de los proyectos, que pueden realizar o no determinadas fases, así como la elaboración de determinados artefactos del total que se definen. Las fases propuestas por este modelo son (Estudio preliminar, Modelado del negocio, Requisitos, Análisis y Diseño, Implementación, Pruebas internas y Pruebas de liberación) (10).

El presente trabajo se centrará en las fases de Requisitos, Análisis y Diseño, Implementación y Pruebas. Todas estas fases permitirán la elaboración de los artefactos necesarios para lograr el producto o la aplicación final. Los artefactos a generar son los prototipos no funcionales de las interfaces, el documento de especificación de requisitos, el modelo de diseño, los diagramas de clases persistentes, repositorio, controladoras, vista, así como los diagramas de secuencia, el modelo de implementación (diagrama de despliegue y diagrama de componentes) y los ficheros de código.

1.6. Formulación de la propuesta de solución

Las herramientas y lenguajes que se reflejan a continuación son las utilizadas en el proyecto SIGEFII, las cuales fueron seleccionadas mediante un profundo análisis por los arquitectos del proyecto, siendo estas las más calificadas para lograr agilizar el trabajo por las características que presentan y las ventajas que brindan a lo largo del proceso de desarrollo de software. Por otra parte en este epígrafe también se recoge una breve síntesis teórica de los principales patrones, métricas y técnicas con las cuales se estará trabajando en el capítulo 2.

1.6.1. Herramientas

AxureRP 5.5:

AxureRP es una herramienta que permite a los diseñadores de aplicaciones crear prototipos no funcionales, diagramas de flujo y especificaciones para las aplicaciones y sitios web. Permite la creación de maquetas o prototipos con lógica condicional, contenido dinámico, animaciones, con solo arrastrar y soltar los campos. Exporta los prototipos en HTML y JavaScript permitiendo ser visualizados por cualquier navegador web. Añade notas y páginas a los widgets proporcionando más contexto a los wireframes. Organiza las anotaciones en campos personalizables para ayudar a manejar la información y estandarizar la documentación que genera (11).

AxureRP es seleccionado dada la necesidad de construir prototipos no funcionales de manera rápida y eficiente para poder mostrarle al cliente un ejecutable de la aplicación futura, dejándole ver así la interacción entre las interfaces mediante la navegación que se realiza por cada una de las funcionalidades del módulo. También ayuda a los desarrolladores a la hora de la implementación, ya que emula principalmente los efectos que debe realizar la vista, como son los widgets visibles o no a partir de criterios de selección.

Visual Paradigm for UML 8.0:

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este software de modelado ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas, generar documentación, así como la generación de código inverso. Esta herramienta también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML (12). Posee varias características importantes como son (multiplataforma, interoperabilidad, modelamiento de los requisitos, colaboración de equipo, generación de documentos, editor de detalles de casos de uso, ingeniería de código, modelado de procesos del negocio y modelamiento de bases de datos) (13).

Visual Paradigm for UML se selecciona por la fuerte integración que posee con el lenguaje UML. Con esta herramienta se busca agilizar el diseño gracias a la interfaz sencilla que posee. Además, permite el trabajo en colaboración de los diseñadores mediante su propio controlador de versiones. También fue muy influyente que la universidad cuente con la licencia para su uso.

Netbeans 7.3:

NetBeans es un IDE de código abierto con una gran base de usuarios, que permite que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. NetBeans es un producto libre y gratuito sin restricciones de uso. Aumenta el rendimiento y la experiencia de programar, además de escanear de forma inteligente el proyecto para corregir cualquier tipo de fallo (14). Tiene fuerte integración el marco de trabajo Symfony2, con los demás lenguajes propuestos y con otras herramientas que facilitan el desarrollo de aplicación de gran envergadura como los Sistemas de Control de Versiones, ayudando al trabajo en colaboración.

NetBeans fue seleccionado por el conocimiento que tenía de su uso el equipo de desarrollo, lo que permite agilizar el trabajo y lograr mayor productividad en el mismo. Además permite poder hacer llamadas a las funcionalidades del marco de trabajo sin tener que acudir al uso de los comandos por consola, lo que haría un poco más lento y tedioso el desarrollo de la aplicación.

PostgreSQL 9.1:

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD[1], lo que significa que se puede disponer de su código fuente, modificarlo a voluntad y redistribuirlo libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multi-hilos para garantizar la estabilidad del sistema (15).

Presenta propiedades como atomicidad, consistencia y aislamiento. La atomicidad asegura la realización de una operación; la consistencia posibilita la ejecución de aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos; y el aislamiento asegura que una operación no pueda afectar a otras, de esta manera dos transacciones sobre la misma información no genera error (16).

PostgreSQL es seleccionado porque brinda seguridad mediante el trabajo con la base de datos, como lo es la confidencialidad de los datos, que es un requisito importante que necesita el cliente. También cuenta con un gran prestigio a nivel internacional y tiene una fuerte integración con la herramienta de gestión gráfica seleccionada.

[1] BSD (Berkeley Software Distribution): Es una licencia de software libre permisiva que permite el uso del código fuente en software no libre. El software modificado puede redistribuirse sin restricciones, como el usuario escoja, libre o propietario.

PgAdmin 3 -1.14.2:

Esta herramienta posee una interfaz gráfica que soporta todas las características de PostgreSQL y facilita enormemente la administración, e incluso pueden realizarse cambios en el modelo de datos desde la herramienta de manera muy sencilla. PgAdmin 3 está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL[2] simples hasta desarrollar bases de datos complejas y facilita enormemente la administración (17).

PgAdmin 3 es seleccionado gracias a la fuerte integración que tiene con el Sistema Gestor de Base de Datos escogido, y por poseer características que permiten que el trabajo con los datos sea de manera fácil y sencilla.

Apache 2.2.22:

Apache está diseñado para ser un servidor web potente y flexible, continuamente actualizado y adaptado a los nuevos protocolos (HTTP) y puede funcionar en la más amplia variedad de plataformas y entornos. Apache se ha adaptado siempre a una gran variedad de entornos a través de su diseño modular el cual permite elegir las características del servidor seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor. Tiene capacidad para servir páginas tanto de contenido estático, como de contenido dinámico a través de otras herramientas soportadas que facilitan la actualización de los contenidos mediante bases de datos, ficheros u otras fuentes de información (18).

Es un software de libre distribución, que publica su código fuente, lo que permite que cualquiera pueda modificarlo y colaborar así a su desarrollo. Tiene interfaz con todos los sistemas de autenticación. Facilita la integración de plugins de los lenguajes de programación de páginas web dinámicas más comunes. Tiene integración en estándar del protocolo de seguridad SSL y provee interfaz a todas las bases de datos (18).

Apache es seleccionado por la necesidad de un servidor web compatible con la plataforma de desarrollo, que permita rapidez en la navegación y la seguridad e integridad de los datos con los que se trabajen. Además apache permite configurar los módulos a utilizar en el servidor, logrando desechar elementos

---

[2] SQL: el lenguaje de consulta estructurado es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

innecesarios para lograr mayor rapidez en la renderización de las páginas.

Symfony 2.1.7:

Symfony2 ha sido ideado para exprimir al límite todas las nuevas características de PHP 5.3 y por eso es uno de los marcos de trabajo de PHP con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en tu proyecto. Symfony2 también es el marco de trabajo que más ideas incorpora del resto de los marcos de trabajo, incluso de aquellos que no están programados con PHP (19).

Además, posee una gran seguridad interna, utilizando técnicas para evitar inyección SQL[3], XSS[4] e incluso CSRF[5], así como la integración de PHP-Unit para la realización de pruebas unitarias. Es un marco de trabajo muy documentado, publicado bajo licencia MIT22, con la que se puede desarrollar aplicaciones web comerciales, gratuitas y/o de software libre (20).

Symfony2 es seleccionado por la necesidad que se tiene de utilizar un marco de trabajo que soporte el lenguaje PHP y permita a los desarrolladores reutilizar otros componentes o partes de otra aplicación. Además, es soportado por el IDE de desarrollo escogido y el equipo de desarrollo ha trabajado con versiones anteriores, por lo que ya conocen su lógica de trabajo y no es necesaria una capacitación profunda para empezar la implementación, permitiendo el ahorro de tiempo en esta tarea.

Doctrine 2.3.2

Doctrine es una herramienta que proporciona persistencia transparente de objetos php, el cual se sitúa en la parte superior de una poderosa capa de abstracción de base de datos (21). Doctrine es una librería muy completa, muy configurable viene integrada por defecto con Symfony2 y presenta entre sus características principales las siguientes:

1. Generación automática del modelo.
2. Posibilidad de trabajar con YAML[6].

---

[3] **Inyección SQL:** es un método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar consultas a una base de datos.

[4] **Cross-site scripting:** es un tipo de agujero de seguridad que permite a una tercera parte inyectar en páginas web vistas por el usuario código en lenguaje script evitando medidas de control.

[5] **Cross-site request forgery:** es un tipo de exploit malicioso de un sitio web en el que comandos no autorizados son transmitidos por un usuario en el cual el sitio web confía.

[6] **YAML:** es un formato de serialización de datos legible. Sus siglas significan en el español "no es otro lenguaje de marcado".

3. Buscadores mágicos.
4. Relaciones entre entidades.
5. Lenguaje DQL[7].

Doctrine es seleccionado por la necesidad de utilizar un marco de trabajo que permita trabajar con los objetos de la base de datos. Doctrine viene integrado por defecto al marco de trabajo Symfony2, además de que permite la generación automática del modelo, logrando agilizar el trabajo.

1.6.2. Lenguajes

Lenguaje Unificado de Modelado (UML)

UML es un lenguaje que permite modelar, construir y documentar los elementos que forman un producto de software que responde a un enfoque orientado a objetos. Es un estándar internacional para definir, organizar y visualizar los elementos que configuran la arquitectura de una aplicación orientada a objetos. UML puede considerarse como un lenguaje de modelado visual que permite una abstracción del sistema y sus componentes (22). Con este lenguaje, se pretende unificar las experiencias acumuladas sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar.

Se selecciona UML ya que es un lenguaje para el modelado enfocado a objeto y es soportado por la herramienta para el diseño seleccionada.

PHP 5.4

PHP es un lenguaje interpretado de alto nivel embebido en páginas html y ejecutado en el servidor. El uso de PHP es extremadamente simple para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales. PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, soporta la mayoría de los servidores web de hoy en día y ofrece soporte para varios gestores de bases de datos entre los que se destaca PostgreSQL. Su característica de ser software libre trae como consecuencia que implique menos costes y servidores más baratos (23).

Es además muy rápido, contiene una biblioteca nativa de funciones sumamente amplia e incluida, no requiere definición de tipos de variables ni manejo detallado del bajo nivel, presenta mejoras de

[7]DQL: Doctrine Query Language es un lenguaje creado para ayudar a los programadores a extraer objetos de la base de datos.

rendimiento, es un lenguaje multiplataforma que funciona en todas las plataformas que soporten Apache. No es un lenguaje de marcas y su principal meta es permitir rápidamente a los desarrolladores la generación dinámica de páginas. Soporta el uso de otros servicios que usen protocolos como IMAP, SNMP, NNTP, POP3, HTTP y derivados. También se pueden abrir sockets de red directos e interactuar con otros protocolos. Al ser un lenguaje libre dispone de una gran cantidad de características que lo convierten en la herramienta ideal para la creación de páginas web dinámicas. Permite la integración con varias bibliotecas externas, generar documentos en formato PDF (documentos de Acrobat Reader) y analizar código XML (23).

PHP es seleccionado por ser un lenguaje de programación del lado del servidor que posee disímiles características para facilitar el desarrollo de una aplicación web, es fácilmente integrable con el marco de trabajo escogido y permite la programación orientada a objeto.

Twig 1.12.1

Twig es un motor y lenguaje de plantillas, integrado con Symfony2 para crear las plantillas de la aplicación, las cuales tienen una sintaxis concisa, fáciles de leer y de escribir. Twig se caracteriza por ser **rápido** debido a que compila las plantillas en código php optimizado, **seguro** ya que tiene un modo de recinto de seguridad para evaluar el código de plantilla que no es confiable, esto permite utilizar Twig como un lenguaje de plantillas para aplicaciones donde los usuarios pueden modificar el diseño de la plantilla y es **flexible** ya que es alimentado por un analizador léxico y un parseador flexible que permite al desarrollador definir sus propias etiquetas y filtros personalizados, y crear su propio DSL[8]. La característica más importante de Twig es que su sistema esta implementado con herencia entre plantillas, lo que permite crear un “esqueleto” de plantilla base que contenga todos los elementos comunes de tu sitio y define los bloques que las plantillas descendientes pueden sustituir, ahorrando así código y tiempo en el trabajo (24).

Twig es seleccionado por la integración que posee con el marco de trabajo escogido. Además, permite agilizar la construcción de plantillas de la vista, brindándoles estructuración y velocidad de ejecución del lado del cliente. Además el uso de Twig limita o minimiza el uso extremo de código a los desarrolladores en muchas ocasiones en las que las vistas de las plantillas estén muy cargadas de contenido. En caso de

[8]DSL (Domain Specific Language): el lenguaje específico de dominio es un lenguaje de programación o especificación de un lenguaje dedicado a resolver un problema en particular, representar un problema específico y proveer una técnica para solucionar una situación particular.

no haberse seleccionado Twig hubiera sido muy conveniente la utilización de otros lenguajes de plantilla basados en texto como lo son Smarty, Django o Jinja. Estos lenguajes son igual de útiles y mediante la combinación de los dos últimos dieron origen a Twig.

CSS 3

CSS es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos (25).

La novedad más importante que aporta CSS 3, de cara a los desarrolladores de webs, consiste en la incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas, algunos de estos mecanismos son el soporte de más color y una gama más amplia de las definiciones del color, bordes con degradados, bordes con imágenes, esquinas redondeadas o bordes redondeados, cajas con sombra, multiples imágenes de fondo, nuevos selectores, entre otras caracteristicas adicionales (26).

CSS3 es seleccionado por las ventajas y facilidades que aporta a la hora de trabajar las plantillas que se van a mostrar en la aplicación, permitiendo ahorro de código y tiempo a los desarrolladores en la elaboración de varios trucos para lograr confeccionar las interfaces como son requeridas por el cliente.

HTML 5

HTML (HyperText Markup Language) es “un lenguaje abstracto usado para representar documentos (se les llama documentos a instancias completas, como lo son las páginas web), y que puede ser transmitido fácilmente por algún medio, como lo es internet” (27). Añade etiquetas para manejar la web semántica (web 3.0): header, footer, article, nav, time (fecha del contenido). Estas etiquetas permiten describir cual es el significado del contenido. Además incorpora mejoras en los formularios con existencia de nuevos tipos de datos (email, number, url, datetime) y facilidades para validar el contenido (28).

Puede incluir un script (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML. Es usado para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML (29).

JavaScript 1.6

JavaScript es un lenguaje de programación utilizado para crear pequeños programas encargados de realizar acciones dentro del ámbito de una página web. Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. El código Javascript es embebido directamente en el código html, haciendo fácil la creación de páginas web con contenido dinámico. Está diseñado para controlar la apariencia y manipular los eventos dentro de la ventana del navegador web y es soportado por la gran mayoría de los navegadores, lo que lo coloca como el lenguaje de programación web del lado del cliente más utilizado (30).

Con JavaScript se pueden crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones JavaScript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador. Es un lenguaje de programación bastante sencillo y pensado para trabajar con rapidez, a veces con ligereza. Es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones y estructuras de datos complejas (30).

JavaScript es seleccionado por la necesidad de contar con un lenguaje para validar el contenido del lado del cliente, así como para manejar algunos datos y eventos que son necesarios y requeridos en las funcionalidades a desarrollar en la aplicación.

JQuery

JQuery es una biblioteca o marco de trabajo de JavaScript que permite simplificar la manera de interactuar con los documentos html, permitiendo manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web. JQuery, al igual que otras librerías, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código. Es decir, con las funciones propias de esta librería se logran grandes resultados en menos tiempo y espacio (31).

JQuery se selecciona ya que es compatible con JavaScript, permite manejar eventos y varias funcionalidades necesarias para el desarrollo de la aplicación sin hacer uso de código en exceso.

1.6.3. Requisitos

Requisito: un requisito es una condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado (32).

Requisitos funcionales: son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares (33).

Requisitos no funcionales: los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad, especifican o restringen las propiedades emergentes del sistema (33).

Técnicas para la captura de requisitos

Entrevista: "es una técnica para obtener datos que consisten en un diálogo entre dos personas: El entrevistador "investigador" y el entrevistado; se realiza con el fin de obtener información de parte de este, que es, por lo general, una persona entendida en la materia de la investigación. La entrevista constituye una técnica indispensable porque permite obtener datos que de otro modo serían muy difíciles conseguir**"** (34).

Tormenta de ideas: "es una técnica de trabajo grupal que consiste en reunir a un grupo de personas con el fin de generar de manera espontánea la mayor cantidad de ideas posible, y así poder, de entre todas éstas, elegir una o varias que permitan tomar una decisión o resolver un problema" (35).

Técnicas para la validación de requisitos

Prototipos no funcionales: el objetivo fundamental de esta técnica, es mostrar un modelo ejecutable del sistema a los usuarios finales y a los clientes. Los cuales podrán experimentar con este modelo para ver si cumple con sus necesidades reales (36).

1.6.4. Diseño

Diseño: según una de las acepciones de la definición que aparece en el Diccionario de la Lengua Española de la R.A.E, el diseño es "concepción original de un objeto u obra destinados a la producción en serie" (37). "El diseño es una representación significativa de ingeniería de algo que se va a construir. En el contexto de la ingeniería del software, el diseño se centra en 4 aéreas importantes de interés: datos, arquitectura, interfaces y componentes" (38).

Diagramas de clases: son los diagramas más comunes en el modelado de sistemas orientados a objetos, usándose en el modelo estático para ver un sistema. Los diagramas de clase son importantes

para la visualización, especificación y documentación del modelo estructural, pero también para la construcción de sistemas ejecutables (39).

1.6.5. Patrones

En términos generales, un patrón se define como la descripción de un problema particular y recurrente de diseño, que aparece en contextos de diseño específico, y presenta un esquema genérico demostrado con éxito para su solución (40).

Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada varias veces sin hacerlo de la misma forma (41).

Patrón de arquitectura

Un patrón de arquitectura de software es un esquema genérico probado para solucionar un problema particular recurrente que surge en un cierto contexto. Este esquema se especifica describiendo los componentes, con sus responsabilidades, relaciones, y las formas en que colaboran (40).

Modelo-Vista-Controlador (MVC)

Este patrón fue descrito por primera vez en 1979 para el lenguaje de programación Smalltalk. Es un patrón de arquitectura de las aplicaciones software, que separa la lógica de negocio de la interfaz de usuario, facilitando la evolución por separado de ambos aspectos. Incrementa la reutilización y flexibilidad del sistema en que se utilice. Mediante el empleo del patrón MVC las vistas y los controladores suelen estar muy relacionados, los controladores tratan los eventos que se producen en la vista **(42)**.

El propósito esencial de MVC es cerrar la brecha entre el modelo mental del usuario humano y el modelo digital que existe en el equipo. MVC soporta la ilusión del usuario de ver y manipular la información de dominio directamente. La estructura es útil si el usuario necesita ver el mismo elemento del modelo de forma simultánea en diferentes contextos y/o desde diferentes puntos de vista (43).

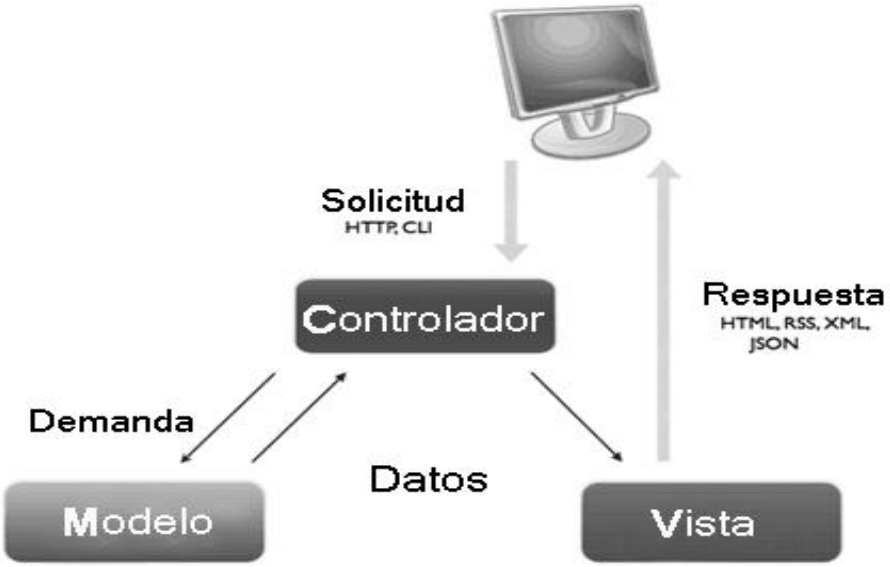


Figura 4: patrón Modelo-Vista-Controlador

Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares (40). “Los patrones de diseño ofrecen las soluciones e idiomas más práctico de que los expertos en el diseño orientado a objetos se sirven para crear sistemas” (44).

Patrones Grasp

Los patrones GRASP comunican los principios fundamentales de la asignación de responsabilidades en el diseño orientado a objetos. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable (40). En capítulos posteriores se reflejará el uso de dichos patrones en el diseño de la solución del problema a resolver.

Patrones GoF

Los patrones GoF (Gang of Four, formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides) se clasifican en 3 categorías basadas en su propósito:

- ✓ **Patrones de comportamiento:** más que describir objetos o clases, describen la comunicación entre ellos.
- ✓ **Patrones estructurales:** separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
- ✓ **Patrones creacionales:** se encargan de la inicialización y configuración de objetos (44).

El marco de trabajo Symfony2 con el cual se implementará el sistema a desarrollar ya trae consigo implementados muchos de estos patrones en su sistema interno, dándole solución a muchos problemas que se presentan continuamente e implementando así numerosas funcionalidades para facilitar el trabajo de los desarrolladores.

1.6.6. Métricas

El IEEE (en español Instituto de Ingenieros Eléctricos y Electrónicos) define métrica como “una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado”. Las métricas son la maduración de una disciplina, que, según Pressman van a ayudar a la evaluación de los modelos de análisis y de diseño, en donde proporcionarán una indicación de la complejidad de diseños procedimentales y de código fuente, y ayudaran en el diseño de pruebas más efectivas (28).

Métricas para requisitos

En este trabajo se utilizará para la validación de los requisitos la métrica “Estabilidad de los requisitos”. La validación de los requisitos con esta métrica brindará información sobre la comprensibilidad y validez de los mismos.

Estabilidad de los requisitos

El objetivo de esta métrica es medir la estabilidad de los requisitos para asegurar su adecuación antes de pasar al próximo flujo de trabajo. Se considera que los requisitos son estables cuando no existen adiciones o supresiones en ellos que impliquen modificaciones en las funcionalidades principales de la aplicación (45).

La estabilidad de los requisitos se calcula como: **ETR = [(RT – RM) / RT] * 100**

- ✓ ETR: valor de la estabilidad de los requisitos.
- ✓ RT: total de requisitos definidos.
- ✓ RM: número de requisitos modificados, que se obtienen como la sumatoria de los requisitos insertados, modificados y eliminados (45).

Métricas de diseño

Las métricas que se utilizarán para medir de forma cuantitativa la calidad de los atributos internos del software permitirán evaluar la calidad del proceso, dichas métricas a emplear son:

Tamaño operacional de clase (TOC)

El tamaño de la clase se determina con la suma del número de atributos (tanto atributos heredados como atributos privados de la instancia) que están encapsulados en la clase y el número total de operaciones (tanto operaciones heredadas como operaciones privadas de la instancia) que están encapsuladas dentro de la clase (46). Los umbrales para medir el TOC son:

Tabla 1: clasificación del TOC de las clases

Clasificación	Valores de los umbrales
Pequeño	<=20
Medio	>20 <=30
Grande	>30

Acoplamiento entre clases objeto (CBO-Coupling between Object Classes)

El CBO es el número de clases asociadas a una clase. Se da dependencia entre dos clases cuando una clase usa métodos o variables de la otra clase. Las clases relacionadas por herencia no se tienen en cuenta. Cuanto más acoplamiento se da en una clase, más difícil será reutilizarla. Además, se dificulta la comprensión y se hace más difícil el mantenimiento por lo que será necesario un mayor esfuerzo en la implementación y en las pruebas (45).

1.6.7. Pruebas

Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. Una vez que se ha generado el código comienzan las pruebas del programa. El proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales; es decir, realizar las pruebas para la detección de errores y asegurar que la entrada definida produce resultados reales de acuerdo con los resultados requeridos (38).

Pruebas de caja blanca

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; ejecuten todos los bucles en sus límites y con sus límites operacionales; y ejerciten las estructuras internas de datos para asegurar su validez (38).

Algunas de las técnicas de las pruebas de caja blanca son las siguientes:

- ✓ **La prueba del camino básico:** el método del camino básico permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida corno guía para la definición de un conjunto básico de caminos de ejecución.
- ✓ **La prueba de condición:** es un método de diseño de casos de prueba que ejercita las condiciones lógicas de condiciones contenidas en el módulo de un programa.
- ✓ **Prueba del flujo de datos:** selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.
- ✓ **Prueba de bucles:** se centra exclusivamente en la validez de las construcciones de bucles (38).

Pruebas de caja negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software, o sea, permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Las pruebas de caja negra ignoran intencionalmente la estructura de control, centra su atención en el campo de la

información y pretenden encontrar errores tales como de funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación (38).

Pruebas de usuario

Las pruebas realizadas por el usuario son pruebas de aceptación que se centran en el cumplimiento de los requisitos definidos para el sistema una vez que este sea terminado. Por lo tanto, constituyen la validación de la aplicación por parte del usuario final y de las cuales Pressman plantea lo siguiente: "La validación del software se consigue mediante una serie de pruebas de caja negra que demuestran la conformidad con los requisitos. Un plan de prueba traza la clase de pruebas que se han de llevar a cabo, y un procedimiento de prueba define los casos de prueba específicos en un intento por descubrir errores de acuerdo con los requisitos. Tanto el plan como el procedimiento estarán diseñados para asegurar que se satisfacen todos los requisitos funcionales, que se alcanzan todos los requisitos de rendimiento, que la documentación es correcta e inteligible y que se alcanzan otros requisitos (por ejemplo, portabilidad, compatibilidad, recuperación de errores, facilidad de mantenimiento)" (47).

Estas pruebas permitirán garantizar que el software realizado cumple con las necesidades del cliente al igual que comprobará que fue construido siguiendo normas de desarrollo de software óptimas y que posee un código fuente seguro y fiable. Además estas validarán que el presente trabajo de investigación cumpla con su objetivo principal.

1.7. Conclusiones

En este capítulo se realizó un estudio de cómo se llevan a cabo los procesos referentes a las revisiones laborales en Cuba y algunas aplicaciones existentes en el mundo que abarcan procesos similares. También se realizó un amplio análisis de las herramientas y lenguajes seleccionados por el equipo de arquitectos del proyecto SIGEFII, teniendo en cuenta la forma en que se iba a manifestar el desarrollo de la solución propuesta con los distintos patrones, así como la utilización de métricas para validar la calidad de los resultados obtenidos en cada fase del proceso de desarrollo de software que se quiere realizar.

Después de un análisis se llega a la conclusión siguiente:

- Las soluciones informáticas desarrolladas en el mundo por las características que poseen y las funcionalidades que cumplen no son adaptables a las necesidades de la FGR.

- Todas las herramientas y lenguajes seleccionados darán un gran impulso en el desarrollo de la solución propuesta en cuanto a rapidez y calidad.
- El uso de patrones en las diferentes fases permitirá el desarrollo de una solución eficiente.
- El uso de las métricas garantizará la calidad requerida de los artefactos generados por cada etapa del desarrollo.

2. Capítulo II. Requisitos y Diseño

2.1. Introducción

En el siguiente capítulo se identifican todos los requisitos funcionales como los no funcionales del sistema, así como la validación mediante las métricas estudiadas de los requisitos funcionales identificados. También se muestra el diseño realizado que abarca el diagrama de clases, diagramas de secuencia y los prototipos no funcionales, que son las interfaces o vistas de todos los procesos que se desarrollan en el módulo de Revisiones Laborales para lograr una representación de la arquitectura. El capitulo concluye con la validación del diseño mediante las métricas reflejadas en el capitulo anterior.

2.2. Requisitos

Mediante el empleo de las técnicas **Entrevista con el cliente** y **Tormenta de ideas** para la captura de requisitos se realizó la recopilación de la información necesaria y se definieron los requisitos que tenía que cumplir el sistema. Llegando a un consenso entre analistas y clientes se obtienen los requisitos, se analizan, se documentan y se validan los mismos, dejando como resultados los requisitos funcionales y no funcionales, al igual que se genera el documento de especificación de requisitos. La mayoría de estos resultados son reflejados a continuación en este epígrafe.

2.2.1. Requisitos funcionales

Se identificaron al final un total de 35 requisitos funcionales, entre los que se tienen 20 independientes del módulo y 15 requisitos que son comunes para todos los otros módulos. Entre estos requisitos identificados se obtuvieron 13 con alta prioridad para el cliente, 15 de prioridad media y 7 de prioridad baja. A continuación se muestran los 20 requisitos propios del módulo:

- ✓ RF_PDC_RL[9]_1 Adicionar rollo.
- ✓ RF_ PDC_RL_2 Actualizar rollo.
- ✓ RF_ PDC_RL_3 Adicionar dictamen.
- ✓ RF_ PDC_RL_4 Actualizar dictamen.

[9]RF_PDC_RL: Estructura empleada para denotar mediante siglas los requisitos funcionales (RF) del subsistema Protección de los Derechos Ciudadanos (PDC) del módulo Revisiones Laborales (RL).

- ✓ RF_ PDC_RL_5 Adicionar ampliación del dictamen.
- ✓ RF_ PDC_RL_6 Actualizar ampliación del dictamen.
- ✓ RF_ PDC_RL_7 Registrar sentencia del TSP.
- ✓ RF_ PDC_RL_8 Actualizar sentencia del TSP.
- ✓ RF_ PDC_RL_9 Registrar auto del TSP.
- ✓ RF_ PDC_RL_10 Actualizar auto del TSP.
- ✓ RF_PDC_RL_11 Buscar rollo.
- ✓ RF_PDC_RL_12 Visualizar detalles del rollo.
- ✓ RF_ PDC_RL_13 Adicionar devolución de expedientes al TSP.
- ✓ RF_ PDC_RL_14 Actualizar devolución de expedientes al TSP.
- ✓ RF_PDC_RL_15 Mostrar vista previa del dictamen.
- ✓ RF_PDC_RL_16 Mostrar vista previa de la ampliación del dictamen.
- ✓ RF_PDC_RL_17 Mostrar vista previa de la carátula del rollo.
- ✓ RF_PDC_RL_18 Mostrar vista previa de la devolución de expedientes al TSP.
- ✓ RF_ PDC_RL_19 Generar reporte de Revisiones Laborales.
- ✓ RF_PDC_RL_20 Mostrar vista previa del reporte de Revisiones Laborales.

2.2.2. Requisitos no funcionales

Los requisitos no funcionales que debe cumplir el sistema son un total de 19, son genéricos para todo el proyecto y se encuentran englobados en 3 requisitos de usabilidad, 5 de confidencialidad, 2 de eficiencia, 2 de soporte, 3 de restricciones de diseño, 3 de interfaz y el último de documentación. Estos se muestran recogidos en el documento **CEGEL_SIGEFII_0113_ERS_EC, sección 3.2** que se encuentra en el expediente de proyecto.

2.2.3. Especificación de requisitos

En esta etapa se generó el artefacto **Especificación de requisitos** que propone el modelo de desarrollo de software empleado para la elaboración de esta investigación. A continuación se hace referencia a uno de los procesos que recoge dicho documento, en este caso el requisito **adicionar rollo**. Los demás requisitos se encuentran recogidos en el documento **CEGEL_SIGEFII_0113_ERS_PDC_RL.doc** en el expediente de proyecto.

Tabla 2: especificación de requisitos para el requisito adicionar rollo

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF_PDC_RL_1	Adicionar rollo	Permite adicionar los datos iniciales del rollo de Revisión Laboral.	Baja	Alta
	Prototipo			

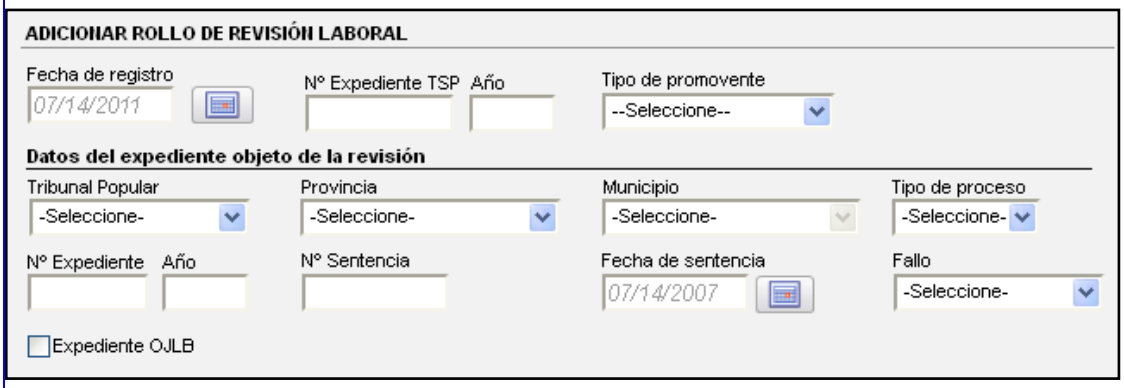


Figura 5 Adicionar rollo de Revisión Laboral.

Campos	Tipos de Datos	Reglas o Restricciones
Fecha de registro	Fecha	Obligatorio. Igual o menor que la fecha actual.
Nº Expediente TSP	Número entero	Obligatorio. Mayor que 0.
Año	Número entero	Obligatorio. Compuesto por cuatro dígitos y mayor

		que 1900. Igual o menor que el año de la fecha de registro.
Tipo de promovente	Cadena	Obligatorio. Los valores de este campo son: • Persona Natural • Persona Jurídica
Datos del expediente objeto de la revisión		
Tribunal Popular	Cadena	Obligatorio. Los valores de este campo son: • Municipal • Provincial
Provincia	Cadena	Obligatorio. Los valores de este campo se encuentran definidos en el nomenclador Provincia.
Municipio	Cadena	Obligatorio. Los valores de este campo se encuentran definidos en el nomenclador Municipio. Se activa si el tribunal es municipal.
Tipo de proceso	Cadena	Obligatorio. Los valores de este campo son: • Derecho • Disciplina
Nº Expediente	Número entero	Obligatorio. Mayor que 0.
Año	Número entero	Obligatorio. Compuesto por cuatro dígitos y mayor que 1900. Igual o menor que el año del expediente del TSP.
Nº Sentencia	Número entero	Obligatorio. Mayor que 0.
Fecha de sentencia	Fecha	Obligatorio. Igual o mayor que el año del expediente del TPP o TPM y menor o igual que el año del expediente del TSP.
Fallo	Cadena	Obligatorio. Los valores de este campo son:

		• Con Lugar • Con Lugar en Parte • Sin Lugar
Expediente OJLB	Booleana	Opcional.
Observaciones	Los nomencladores se evidencian en el documento: Modelos, Reportes y Nomencladores del SIGEF II.	

2.2.4. Validación de los requisitos

Validación mediante los prototipos

La validación de los requisitos con los clientes fue mediante los prototipos de interfaz no funcionales, dándole una visión al cliente de las funcionalidades del sistema y como va a interactuar con él en un futuro. Se revisaron cada uno de los prototipos con los fiscales correspondientes con los procesos laborales y se corrigieron los errores. Al final de este proceso quedo como resultado el acta de aceptación firmada por el cliente.

Validación mediante la métrica Estabilidad de los Requisitos

Para el cálculo de la estabilidad de los requisitos se emplea la siguiente fórmula

ETR = [(RT – RM) / RT] * 100

ETR: valor de la estabilidad de los requisitos.

RT: total de requisitos definidos.

RM: número de requisitos modificados, que se obtienen como la sumatoria de los requisitos insertados, modificados y eliminados.

Entonces teniendo en cuenta que se identificaron en un principio un total de 39 requisitos (sumando los requisitos funcionales comunes) y fueron modificados un total de 8 la estabilidad quedaría de la siguiente manera:

ETR = [(39 - 8) / 39] * 100

ETR = 79,49

El porciento de estabilidad no fue muy aceptable en cuanto a lo esperado, por lo que se realizó una segunda revisión con el cliente, arrojando valores positivos en cuanto a la estabilidad o cambios de los requisitos durante el desarrollo con aproximadamente un 98 %, siendo los resultados aceptables en este proceso.

2.3. Diseño

"El diseño es, inevitablemente, el primer paso que se debe dar cuando se quiere llevar a cabo un proyecto, y abarca desde la planificación previa de este hasta la definición del aspecto final que tendrá. Asimismo es muy importante definir la forma final del producto y dotarlo de un aspecto visual atractivo, extremo que sin el trabajo de un diseñador es francamente complicado de conseguir. Debe ser capaz de facilitar las mejoras del software, tiene que ser entendible por otros profesionales de la especialidad de manera que permita la comprobación del sistema fácilmente" (38). En el presente epígrafe se muestran los resultados arrojados en esta fase del desarrollo de software y la validación de los mismos.

2.3.1. Patrones empleados

Patrón de arquitectura

El patrón arquitectónico empleado es el Modelo-Vista-Controlador, facilitado por el marco de trabajo seleccionado anteriormente. En la aplicación se emplean las 3 capas de la manera siguiente:

- El modelo representa la información o las clases con las que trabaja la aplicación. En Symfony2 son todas las clases Repository y las clases entidades que se encuentran almacenadas en directorio **Entity/** de cada Bundle.
- La vista es con lo que el usuario interactúa y muestra un aspecto del modelo. En Symfony2 son todas las vistas gestionadas por el motor de plantilla Twig que se almacenan en el directorio **Resources/views/**.
- El controlador es la pieza de código que llama al Modelo para obtener algunos datos que le pasa a la vista para la presentación al cliente. En Symfony2 todas las solicitudes son gestionadas por un controlador frontal (app.php, app_dev.php). Estos controladores frontales delegaran la verdadera labor a las acciones, estas acciones son agrupadas en controladores dentro de la carpeta **Controller/**.

Patrones Grasp

- **Alta Cohesión:** es utilizado principalmente en las clases Controller, las cuales son las únicas encargadas de definir y realizar todas las acciones que son solicitadas, es decir, está formada por diferentes funcionalidades que están relacionadas entre sí, teniendo un sentido común y un propósito único, proporcionando flexibilidad en el sistema.

- **Bajo Acoplamiento:** se emplea principalmente en las clases Controller, ya que todas heredan de la clase ProcesoController, logrando así un bajo acoplamiento entre las clases.

- **Experto:** es uno de los patrones que más se utiliza cuando se trabaja con Symfony2, con la inclusión de la librería Doctrine para mapear la base de datos. Symfony2 utiliza esta librería para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades.

- **Creador:** se emplea en las clases RolloRevLabController y RelacionEntregaController. En estas clases se encuentran las acciones definidas para el módulo y se ejecutan cada una de ellas. En dichas acciones se crean los objetos de las clases que representan las entidades, por lo que la clase RolloRevLabController y RelacionEntregaController son "creadoras" de todas las entidades del módulo.

- **Controlador:** en Symfony2 todas las peticiones web son manipuladas por un controlador frontal que es el punto de entrada único de toda la aplicación en un entorno determinado (app.php, app_dev.php). Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

Patrones GoF

Patrones de Comportamiento

- **Observer (Observador):** se utiliza en el despachador de eventos. El núcleo y el manejador de peticiones están basados en dicho despachador de eventos, haciendo que esto sea una de las mejores maneras de desacoplar el código y hacerlo más flexible.

Patrones Creacionales:

- **Abstract Factory (Fábrica Abstracta):** se emplea mucho para la creación de los objetos entidades,

potenciando el encapsulamiento, ya que aísla a los clientes de las implementaciones y se obtiene un incremento de la flexibilidad del diseño.

Patrones Estructurales:

- **Decorator (Decorador):** añade funcionalidad a una clase dinámicamente. El sistema de plantillas Twig, está provisto de un mecanismo de herencia gracias al cual la decoración de plantillas resulta de una flexibilidad y versatilidad total. Esto permite fragmentar la vista en distintas plantillas organizadas por criterios funcionales, y combinarlas para producir la vista completa. Con Twig la herencia se implementa mediante el concepto de bloque, empezando desde la plantilla base de la aplicación que se encuentra en el directorio **app/Resources/view/base.html.twig.**
- **Composite (Compuesto):** permite tratar objetos compuestos igual que los simples. Se utiliza en la construcción de objetos complejos a partir de otros más simples e iguales, simplificando así el tratamiento de los objetos creados, esto mediante la utilización de la composición recursiva y una estructura de árbol.

2.3.2. Estándares de codificación

A continuación se muestran los estándares de codificación empleados por el proyecto para la fase de diseño, en función de lograr unanimidad a lo largo de la construcción de la aplicación:

Clases:

1. Los nombres deben ser descriptivos y concisos. No usar ni grandes frases ni pequeñas abreviaciones.
2. Los nombres de las páginas twig serán en minúscula siempre y usar underscore para palabras compuestas. Eje: adicionar_solicitud.
3. Los nombres de las clases en mayúscula. Eje: RolloRevLab
4. Los atributos siempre en minúscula y usar underscore para palabras compuestas. Eje: persona_proceso
5. Las funciones deben empezar con minúscula y continúan en mayúscula. Eje: imprimirDatos().

Métodos del controlador:

1. Siempre usar el nombre **SalvarActualizar** para el caso de las páginas que contienen formulario. Se utiliza el mismo método del controlador para mostrar la página (get) que para persistir (post).
2. Los que no tengan formularios, un nombre sugerente con la información que se está mostrando. Eje: para el caso de las páginas principales de los procesos, se pone el nombre del proceso en plural. Eje: quejasAction, revisionesLaboralesAction, etc.

Demás métodos:

1. Usar al comienzo del nombre **cargar** para los métodos que cargan por id. Eje: cargarRolloporId().
2. Usar al comienzo del nombre **listar** para los métodos que llenan la información de los grid. Eje: listarRollosporPasos().

2.3.3. Artefactos generados

En el proyecto SIGEF por las facilidades que brinda el marco de trabajo seleccionado no se realiza el modelado del diseño lógico de datos, sino que se exporta el diagrama de clases desde la herramienta Visual Paradigm a clases .**php** directo al marco de trabajo, y desde el marco de trabajo se exportan las clases hacia la base de datos.

A continuación se muestran los diferentes diagramas de clases del módulo Revisiones Laborales, el cual no refleja todas las clases con que se trabajan en dicho módulo ya que se encuentran recogidas en el diagrama de clases **Arquitectura base del proyecto**, el cual contiene todas las clases que son comunes para la aplicación, permitiendo su reutilización en otros módulos. Las clases que se encuentran en color verde son algunas de estas clases comunes que son utilizadas en la mayoría de los módulos del sistema.

Diagrama de clases persistentes

El diagrama de clases persistentes mostrado a continuación permite reflejar las clases con las cuales se van a trabajar, ya que recogen toda la información necesaria, y por lo tanto van a persistir en la base de datos. Refleja todas las relaciones entre las clases, así como sus atributos, operaciones que contienen y su cardinalidad. La clase principal de este diagrama es RolloRevLab, la cual hereda de proceso que es una clase común para la aplicación que contiene la mayoría de los atributos genéricos permitiendo reutilización y facilidades a la hora de la implementación.

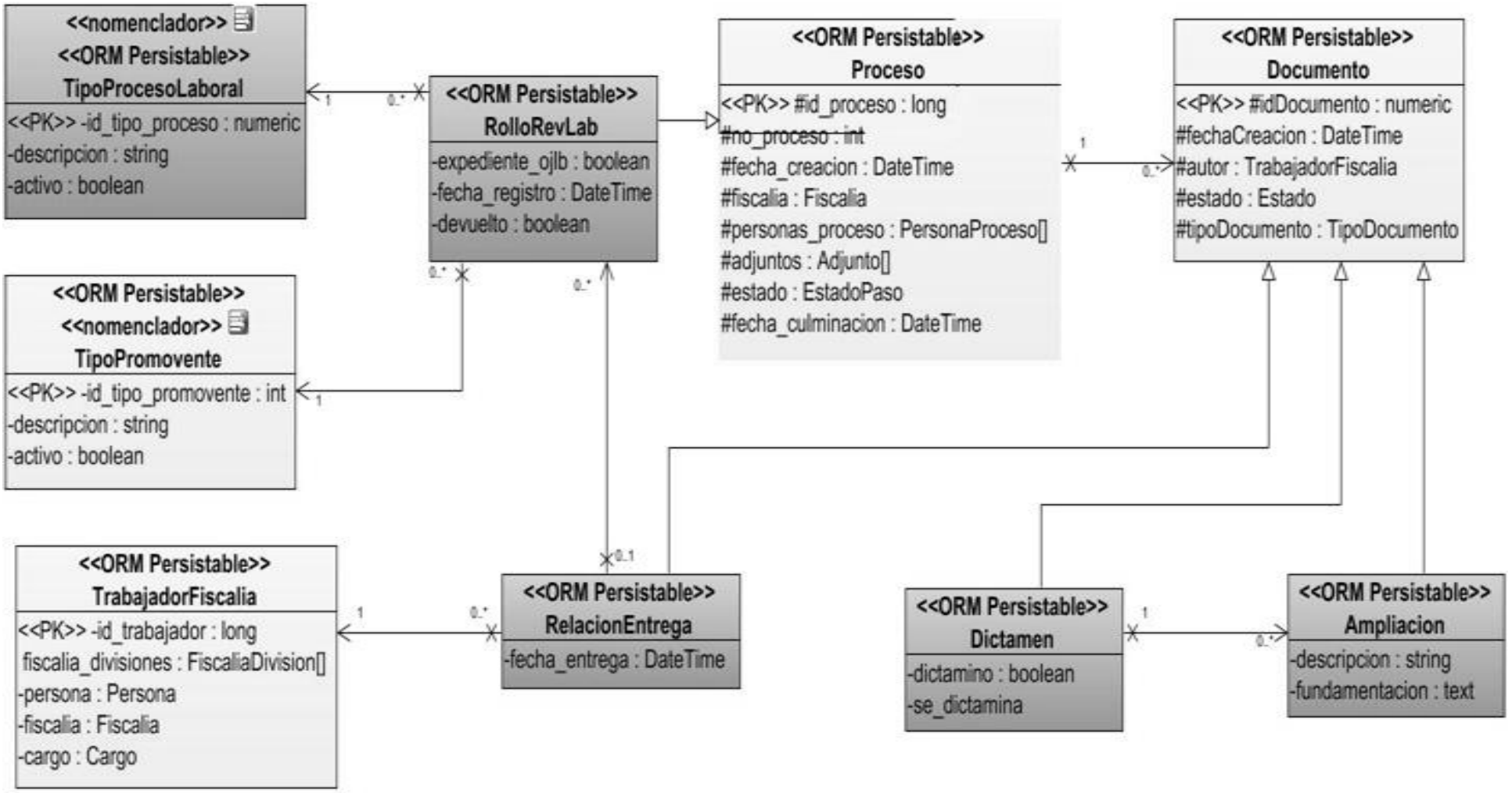


Figura 5: diagrama de clases persistentes

Diagrama de clases controladoras

El siguiente diagrama de controladoras permite mostrar las clases controladoras las cuales son las encargadas de las acciones o funcionalidades del sistema, coordinando así el trabajo que debe realizar la aplicación. También se muestran las clases gestoras que son las intermediarias entre las clases controladoras y las clases repositorio, y se encargan de toda la lógica del negocio. Estas clases encapsulan todos los métodos necesarios para el correcto funcionamiento de la aplicación.

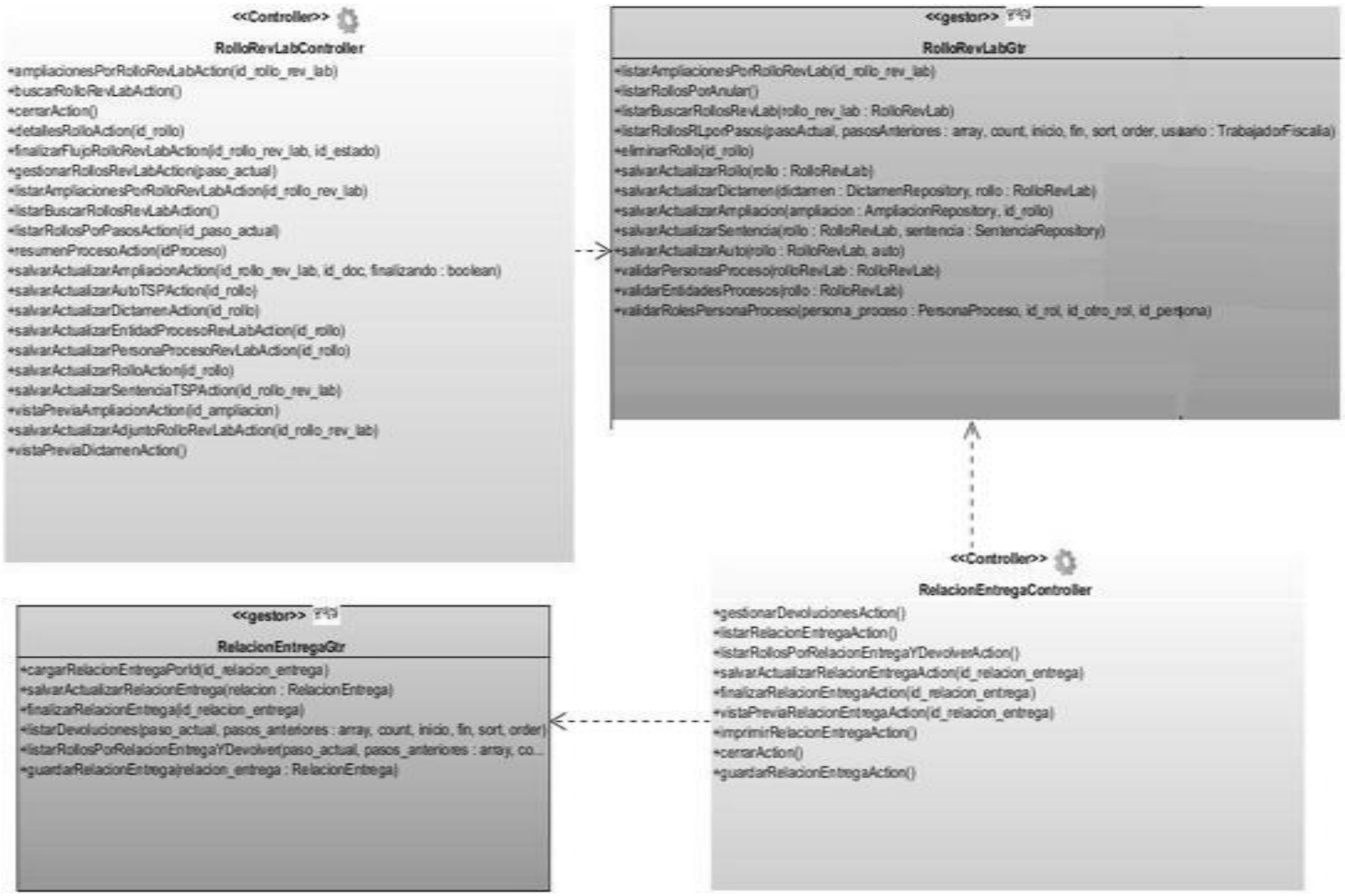


Figura 6: diagrama de clases controladoras y gestoras

Diagrama de clases de repositorio

El siguiente diagrama de clases del repositorio permite mostrar las clases encargadas del acceso a los datos o la información que necesita la aplicación. Contienen todos los métodos necesarios para acceder y obtener a la información que requiera la aplicación para su ejecución, ejemplo de ellos son find(), findAll(), findBy() y findOneBy(), al igual que otros métodos declarados anteriormente para su posterior utilización.

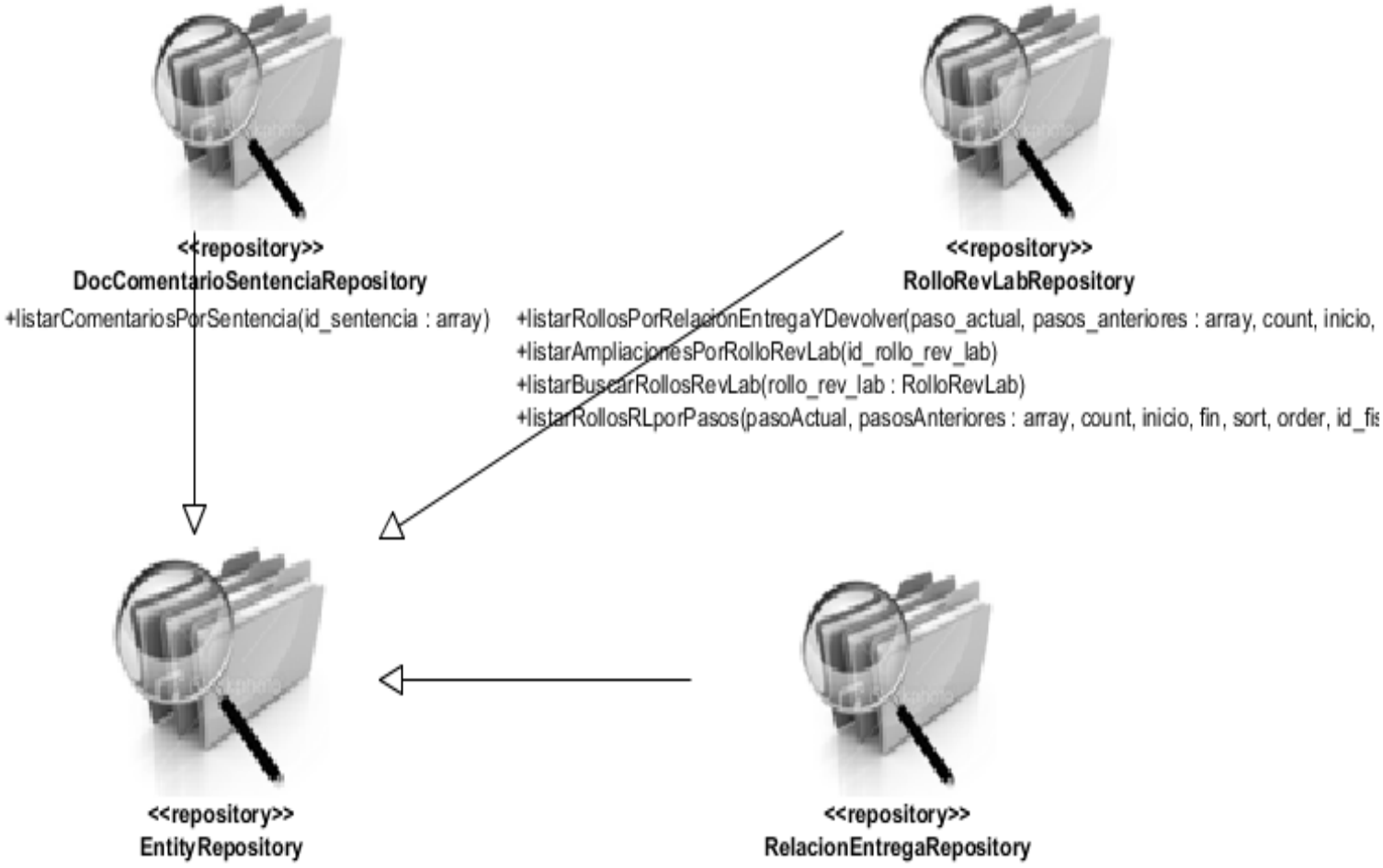


Figura 7: diagrama de clases de repositorio

Diagrama de clases de la vista

El diagrama de clases de la vista permitirá a los desarrolladores una mejor abstracción y comprensión del sistema a la hora de su implementación. Dicho diagrama refleja todas las páginas de la aplicación, incluyendo las vistas previas, los formularios y las vistas generadas o trabajadas con JavaScript. A través de este diagrama se puede conocer la navegación básica por la que debe atravesar el sistema, empezando por la página principal gestionar_rollo. El mismo está dividido por los diferentes pasos del proceso que son iniciar, dictaminar, ampliar, registrar sentencia y auto, y devolución de expedientes. El diagrama mostrado a continuación contiene solo las vistas de la funcionalidad de iniciar rolllo.

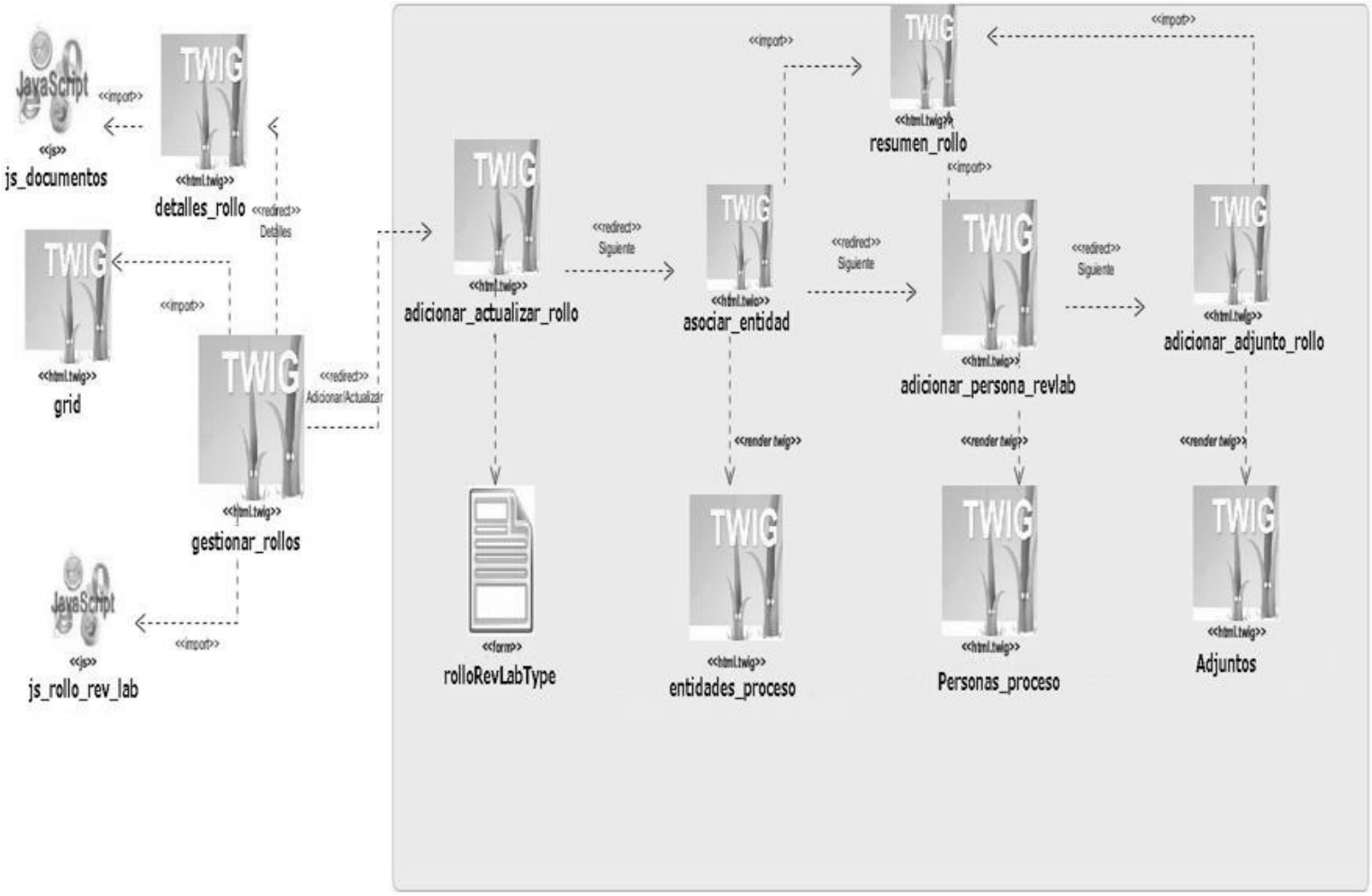


Figura 8: diagrama de clases de la vista

Diagramas de secuencia

Los diagramas de secuencia permiten guiar a los desarrolladores a lo largo de la implementación del sistema, representando la interacción entre los objetos del sistema. A continuación se muestra el diagrama de secuencia de la funcionalidad adicionar rollo, que según el criterio del autor recoge una similitud de la lógica del negocio de la mayoría de los diagramas de secuencia elaborados. Este diagrama de secuencia muestra el flujo de pasos que debe realizar el sistema para poder adicionar un rollo. Los demás diagramas de secuencia se encuentran reflejados en los anexos del documento.

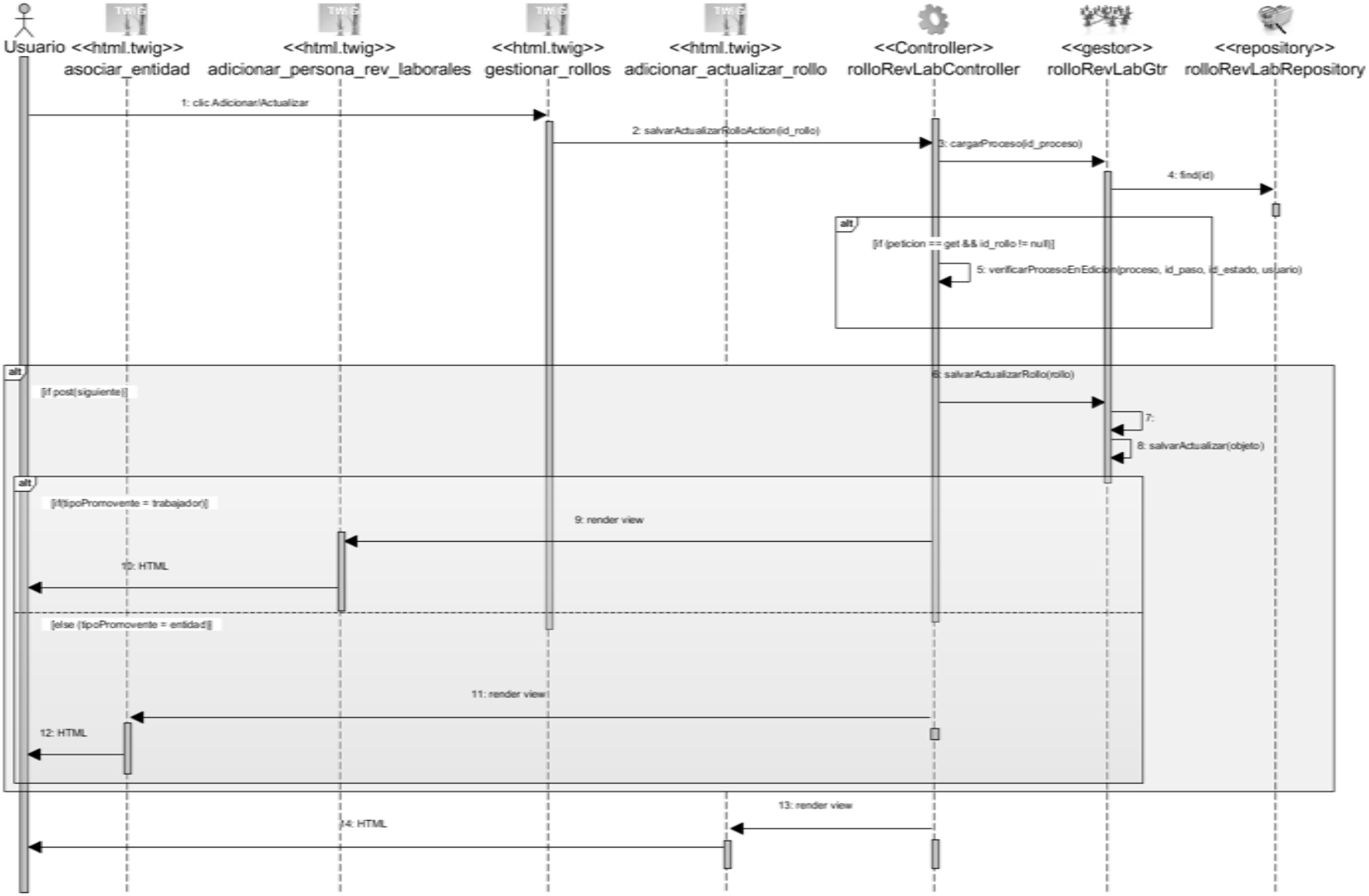


Figura 9: diagrama de secuencia de la funcionalidad adicionar rollo de Revisiones Laborales

2.3.4. Validación del diseño

En este epígrafe se valida el diseño realizado mediante la aplicación de las métricas reflejadas en el capítulo 1.

Tamaño operacional de las clases (TOC)

Tabla 3: representación del tamaño de las clases del diseño.

No.	Clases	Cantidad de atributos	Cantidad de operaciones	Tamaño
1	RolloRevLab	11	23	Grande
2	TipoProcesoLaboral	3	6	Pequeño
3	TipoPromovente	3	6	Pequeño
4	Proceso	8	16	Medio
5	Documento	5	10	Pequeño
6	Dictamen	7	4	Pequeño
7	Ampliacion	7	4	Pequeño
8	RelacionEntrega	5	0	Pequeño
9	TrabajadorFiscalia	7	14	Medio

Se trabajó con un total de 9 clases, de las cuales se obtuvo un promedio de 6,2 atributos y 9,2 operaciones por clases.

Luego de aplicar esta métrica, se llega a la conclusión que la mayoría de las clases con que se trabajan son pequeñas (6 de 9), por lo que no asumen grandes responsabilidades (56% de baja responsabilidad), el sistema tendrá baja dificultad en su implementación (56%), y contará con una alta reutilización de sus clases. Por lo que se puede afirmar que los resultados obtenidos de la métrica empleada son positivos.

Acoplamiento entre clases objeto (ACO)

Utilizando esta métrica descrita en el capítulo 1 se lograron obtener los resultados del acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas del diseño utilizando los criterios siguientes:

Tabla 4: criterios de evaluación de la métrica ACO

Atributo	Categoría	Criterio
Acoplamiento.	Ninguno.	0
	Bajo.	1
	Medio.	2
	Alto.	>2
Complejidad de mantenimiento.	Baja.	<=Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	>2*Promedio
Reutilización.	Baja.	>2*Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	<=Promedio

A continuación se muestran los resultados obtenidos en cada uno de estos atributos gráficamente:

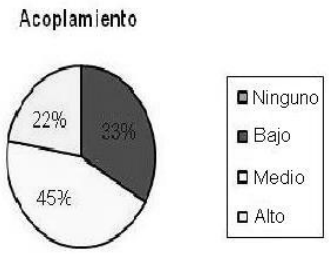


Figura 10: resultado en % del acoplamiento

Figura 11: resultado en % de la complejidad de mantenimiento

Figura 12: resultado en % de la reutilización

Estos valores arrojados por la métrica empleada permiten determinar y concluir que las clases del diseño presentan un acoplamiento medio entre ellas, por lo que arrastra valores medios igualmente para los atributos de complejidad de mantenimiento y la reutilización de las mismas.

2.4. Conclusiones

En este capítulo se realizó la identificación de requisitos y el diseño del módulo Revisiones Laborales para el SIGEFII, así como la validación de los resultados de ambas fases mediante métricas, obteniendo en cada una de ellas resultados satisfactorios. Mediante la etapa de requisitos se pudo llegar a identificar los requisitos funcionales y no funcionales que debe cumplir la aplicación que se quiere desarrollar, así como una descripción de las funcionalidades del sistema en el documento Especificación de requisitos. Mediante la etapa de diseño queda reflejada como resultado una visión más profunda de la aplicación. En esta fase se generaron los diagramas de clases persistentes, de repositorio, de controladoras, de la vista y los diagramas de secuencia, facilitando una guía a seguir para la posterior implementación de la aplicación.

3. Capítulo III. Implementación y Prueba.

3.1. Introducción

En el siguiente capítulo se desarrolla la implementación de la aplicación, luego de estar terminado el diseño y las validaciones del mismo. La implementación del sistema se obtiene en términos de los componentes necesarios, es decir, ficheros de código fuente y ejecutables. Este capítulo contiene el modelo de implementación, que está compuesto por los diagramas de componentes y de despliegue. También se validan estos resultados con las pruebas anteriormente mencionadas en el capítulo 1, al igual que se realiza la validación de las variables de la investigación.

3.2. Implementación

La implementación del sistema se comienza una vez que se tenga el diseño del mismo y quedándole claro a los desarrolladores que es lo que deben hacer y cómo se debe comportar el software a construir. A través de los resultados del capítulo anterior se obtienen los artefactos que se muestran en este capítulo.

3.2.1. Artefactos

Diagrama de despliegue

En el modelo de despliegue se describe como se organizarán los componentes de acuerdo a la estructura de la FGR, así como la relación que debe existir entre estos para un correcto funcionamiento del sistema. Los procesos laborales solo se realizan en la FGR por lo que solo se despliega en dicha institución.

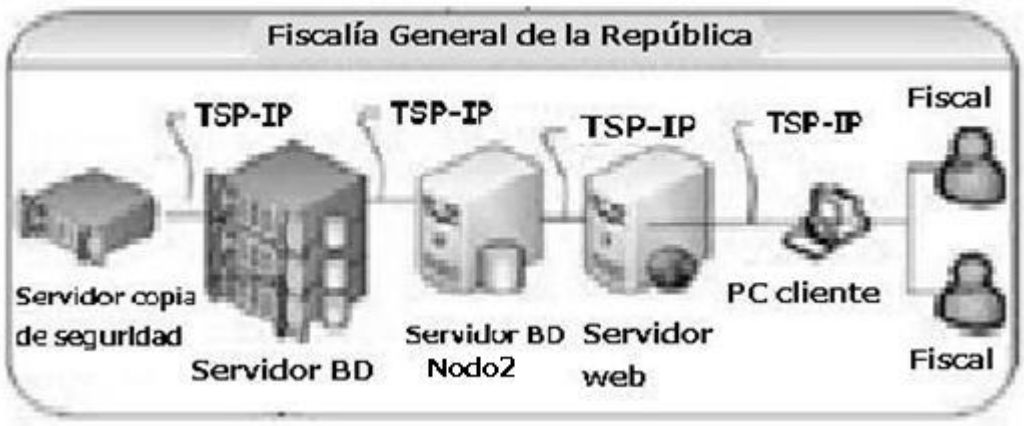


Figura 13: diagrama de despliegue

Diagrama de componentes

El diagrama de componentes representa la separación de un sistema de software en componentes físicos (por ejemplo archivos, cabeceras, módulos, paquetes, etc.), mostrando las dependencias entre estos componentes y la forma organizativa de la aplicación. En el siguiente diagrama se muestra como está estructurada la aplicación a partir del uso del marco de trabajo Symfony.

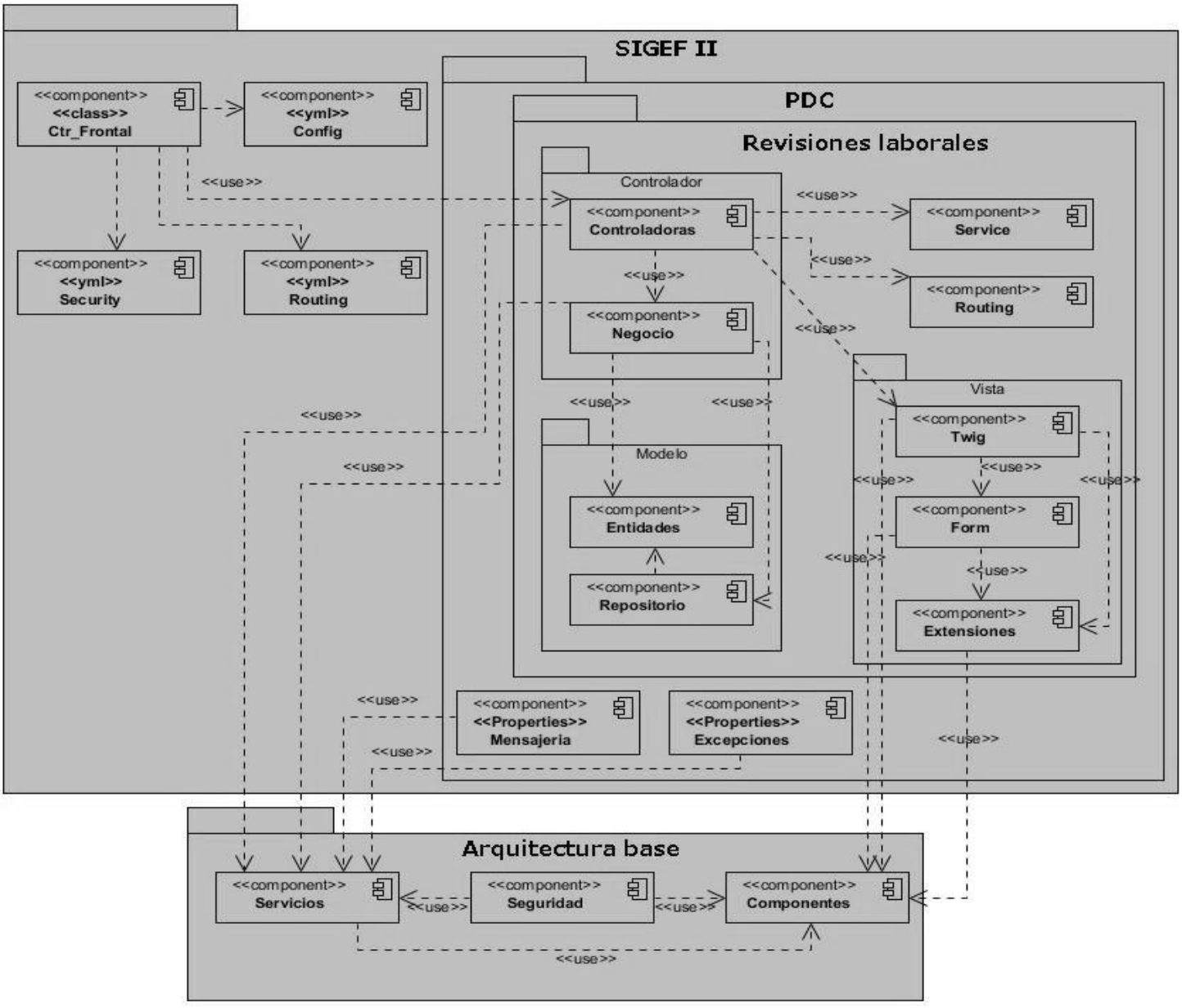


Figura 14: diagrama de componentes

3.2.2. Estándares de codificación

Los estándares de codificación fueron definidos por la dirección del proyecto antes de comenzar el desarrollo, para así lograr unanimidad en el código de todos los módulos del sistema. Dicho estándar de codificación se encuentra registrado en el documento “Estándares de codificación para PHP” del proyecto SIGEF II.

A continuación se muestran algunos ejemplos de los mismos:

Clases:

Los nombres de clases pueden contener sólo caracteres alfanuméricos. Los números están permitidos en los nombres de clase, pero desaconsejados en la mayoría de casos. Las barras bajas (_) están permitidas solo como separador de ruta (el archivo " Entity/Table.php " debe apuntar al nombre de clase " Entity_Table ").

Clases abstractas:

En general, las clases abstractas siguen las mismas convenciones que las clases, con una regla adicional: Los nombres de las clases abstractas deben acabar con el término, "Abstract", y ese término no debe ser precedida por un guión bajo.

Nombres de archivos:

Cualquier archivo que contenga código PHP debe terminar con la extensión " .php ", con la excepción de los scripts de la vista.

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.

3.3. Pruebas

Las pruebas son un factor crítico para garantizar la calidad del software. Una prueba conlleva la intención de descubrir algún error en el sistema desarrollado. El éxito de las pruebas se mide en función de la capacidad de detectar un error que estaba oculto (38). En este epígrafe se recogen las validaciones del sistema realizadas mediante las pruebas de caja negra, caja blanca y del usuario.

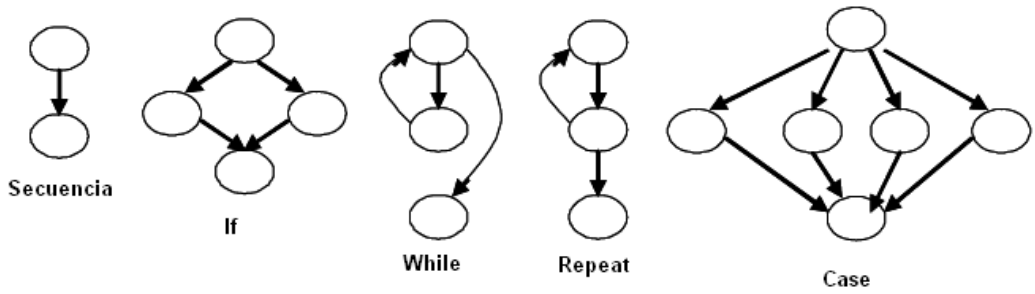
3.3.1. Pruebas de caja blanca

Las pruebas de la caja blanca usan la estructura de control del diseño procedural para derivar los casos de prueba. A continuación se muestra la validación mediante la **prueba del camino básico** en el cual se obtienen un conjunto de caminos independientes, se construye el grafo de flujo asociado y se calcula la complejidad ciclomática del mismo. El código que se muestra para la realización de esta prueba es el de la funcionalidad adicionar/actualizar rollo de revisiones laborales:

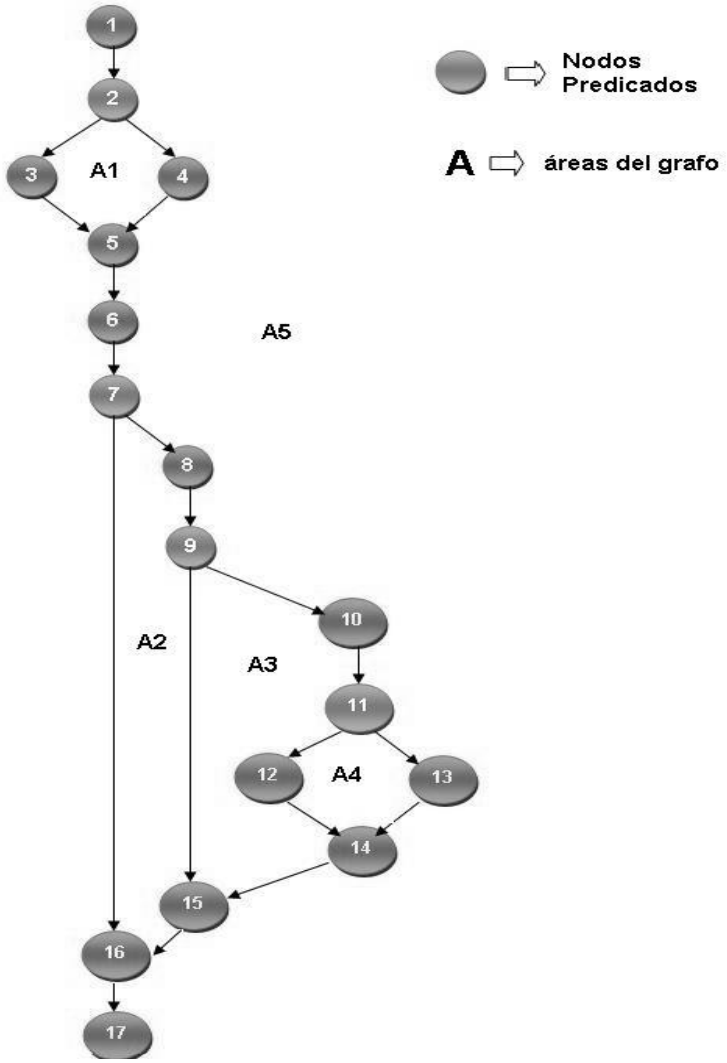
```
public function salvarActualizarRolloAction($paso, $id_proceso) {
    $peticion = $this->getRequest();
    $proceso = $this->getRolloRevLabGtr()->crearRolloRevLab($id_proceso);
    $usuario = $this->getUser();
    if ($paso == ConfigUtil::paso_iniciar_rl) {
      $url = $this->generateUrl('rl_rollo_rev_lab', array('paso' => $paso));
    } else {
       $url = $this->generateUrl('rl_detalles_rollo', array('paso' => $paso, 'id_rollo' => $id_proceso));
    }
  $qb_tipo_tribunal = $this->getRolloRevLabGtr()->qbTipoTribunal(array(ConfigUtil::tipo_tribunal_supremo));
  $qb_fallo_tribunal = $this->getRolloRevLabGtr()->qbFalloTribunal(array(ConfigUtil::tipo_sentencia));
    $frm = $this->createForm(new AdicionarRolloType(), $proceso, array(
             'qb_tipo_tribunal' => $qb_tipo_tribunal,
             'qb_fallo_tribunal' => $qb_fallo_tribunal));
    if ($peticion->getMethod() == 'POST') {
           $frm->bind($peticion);
       if ($frm->isValid()) {
           $proceso = $this->getRolloRevLabGtr()->salvarActualizarRollo($proceso,
                            ConfigUtil::tipo_proceso, $paso, ConfigUtil::estado_pendiente, $usuario);
               if (isset($peticion->get(ConfigUtil::btn_guardar))) {
                    return $this->redirect($this->generateUrl('rl_rollo_rev_lab', array('paso' => $paso)));
                } else {
                    return $this->redirect($this->generateUrl('rl_entidad_proceso_rollo',
                              array('paso' => $paso, 'id_proceso' => $proceso->getIdProceso())));
                }
        }
     }
        return $this->render('RevLaboralBundle:RolloRevLab:adicionar_actualizar_rollo.html.twig', array(
                    'frm' => $frm->createView(),
                    'paso' => $paso,
                    'id_proceso' => $proceso->getIdProceso()));
}
```

Figura 15: código del método perteneciente a la funcionalidad adicionar o actualizar rollo

Una vez tenido el código el primer paso a realizar es dibujar el grafo de flujo asociado según las sentencias declaradas en el código, empleando para eso las instrucciones siguientes:



Por lo que el grafo de flujo queda de la siguiente manera:



El siguiente paso es calcular la complejidad ciclomática del grafo de flujo (**V(G)**) que establece el número de caminos independientes del conjunto básico de un programa. Existen tres formas para esta operación:

- ✓ **V(G) = A** , siendo A el número de áreas independientes del grafo, incluyendo el área exterior del grafo.

  V(G) = 5

- ✓ **V(G) = A – N + 2** , siendo A el número de aristas del grafo y N el número de nodos.

  V(G) = 20 – 17 +2

  V(G) = 5

- ✓ **V(G) = P + 1 ,** siendo P el número de nodos predicado del grafo. Un nodo predicado es aquel nodo del que salen varias aristas.

  V(G) = 4 + 1

  V(G) = 5

Una vez calculada la complejidad ciclomática el siguiente paso es encontrar los caminos independientes del grafo:

Tabla 5: caminos independientes del grafo de flujo

No.	Camino
1	1,2,3,5,6,7,16,17
2	1,2,**4**,5,6,7,16,17
3	1,2,4,5,6,7,**8,9,15**,16,17
4	1,2,4,5,6,7,8,9,**10,11,12,14**,15,16,17
5	1,2,4,5,6,7,8,9,10,11,**13**,14,15,16,17

Una vez obtenidos los caminos básicos del flujo se procede a elaborar los casos de prueba. Debe realizarse al menos un caso de prueba por cada camino independiente obtenido. Se elaboró un caso de prueba por cada camino independiente que se obtuvo del código perteneciente a los 13 requisitos con mayor prioridad para el cliente, de donde se obtuvieron un total de 2 fallos en el código, los cuales fueron analizados y solucionados posteriormente. A continuación se muestran los casos de prueba elaborados para cada uno de los caminos independientes obtenidos anteriormente:

Caso de prueba para el camino #1:

Variable en el código	Valor otorgado
paso	Iniciar_rollo_rl
petición	get

Caso de prueba para el camino #2:

Variable en el código	Valor otorgado
paso	dictaminar_rollo_rl
petición	get

Caso de prueba para el camino #3:

Variable en el código	Valor otorgado
paso	Iniciar_rollo_rl
petición	post
formulario	inválido

Caso de prueba para el camino #4:

Variable en el código	Valor otorgado
paso	Iniciar_rollo_rl
petición	post
formulario	válido
botón	guardar

Caso de prueba para el camino #5:

Variable en el código	Valor otorgado
paso	Iniciar_rollo_rl
petición	post
formulario	válido
botón	siguiente

3.3.2. Pruebas de caja negra

Las pruebas de caja negra realizadas durante el proceso de revisión del software pretenden encontrar a través de los casos de prueba los errores que presenta el sistema desarrollado. Algunos de los tipos de errores a encontrar son:

- Funciones incorrecta o ausente.
- Errores en la interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.

- Errores de inicialización y de terminación.

Esta prueba cuenta de varias técnicas, la utilizada en esta investigación para realizarles las pruebas al sistema es la **Técnica de la partición de equivalencia**, la cual divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Donde una clase de equivalencia representa un conjunto de estados válidos, inválidos o que no aplican, para determinadas condiciones de entrada. Las condiciones de entrada son valores numéricos específicos, un rango de valores, un conjunto de valores relacionados o una condición lógica (38).

El escenario a probar que se muestra es el de adicionar/actualizar rollo de revisiones laborales, que es la funcionalidad principal de módulo.

Tabla 6: caso de prueba de la funcionalidad adicionar o actualizar rollo

Escenario	Descripción	Tipo Promovente	Tribunal Popular	Provincia	Municipio	Tipo Proceso	No. Exp	Año	No. Sentencia	Fallo Tribunal	Exp. OJLB	Respuesta del sistema	Flujo central
Adicionar o Actualizar rollo de revisiones laborales.	Permite adicionar o actualizar un rollo de revisiones laborales, así como los datos del expediente de revisión enviado por el TSP.	V	V	V	V	V	V	V	V	V	V	Adiciona o actualiza el rollo y el expediente de revisión.	Selecciona la opción Iniciar rollo del menú principal del proceso. Selecciona la opción adicionar rollo de la página o selecciona un rollo y el botón actualizar. Muestra una pantalla con los datos a introducir o con los valores a editar. Salva el rollo y asocia el expediente al proceso.
		Persona natural	Municipal	Cienfuegos	Cienfuegos	Disciplina	14	2013	2	Con lugar	No		
		N/A	V	V	V	V	V	V	V	V	V	El sistema muestra un error con el mensaje “El campo tipo de promovente es obligatorio”.	
		Vacío	Provincial	Cienfuegos	No se muestra	Disciplina	14	2013	2	Con lugar	No		
		V	N/A	V	V	V	V	V	V	V	V	El sistema muestra un error con el mensaje “El campo tipo de tribunal es obligatorio”.	
		Persona jurídica	Vacío	No se muestra	No se muestra	Disciplina	14	2013	2	Sin lugar	Si		
		V	V	N/A	V	V	V	V	V	V	V	El sistema muestra un error con el mensaje “El campo provincia es obligatorio”.	
		Persona jurídica	Provincial	Vacío	No se muestra	Disciplina	14	2013	2	Sin lugar	Si		
		V	V	V	N/A	V	V	V	V	V	V	El sistema muestra un error con el mensaje “El campo municipio es obligatorio”.	
		Persona jurídica	Municipal	Matanzas	Vacío	Disciplina	14	2013	2	Sin lugar	Si		
		V	V	V	V	N/A	V	V	V	V	V	El sistema muestra un error con el mensaje “El campo tipo de proceso es obligatorio”.	
		Persona jurídica	Municipal	Matanzas	Colon	Vacío	14	2013	2	Sin lugar	Si		

		V	V	V	V	V	N/A	V	V	V	V	El sistema muestra un error con el mensaje “El campo No de expediente es obligatorio”.	
		Persona jurídica	Provincial	Matanzas	No se muestra	Derecho	Vacío	2013	2	Sin lugar	No		
		V	V	V	V	V	V	N/A	V	V	V	El sistema muestra un error con el mensaje “El campo año es obligatorio”.	
		Persona jurídica	Provincial	Matanzas	No se muestra	Derecho	1	Vacío	2	Con lugar	No		
		V	V	V	V	V	V	V	N/A	V	V	El sistema muestra un error con el mensaje “El campo No de sentencia es obligatorio”.	
		Persona jurídica	Provincial	Matanzas	No se muestra	Derecho	1	2012	Vacío	Con lugar	No		
		V	V	V	V	V	V	V	V	N/A	V	El sistema muestra un error con el mensaje “El campo fallo del tribunal es obligatorio”.	
		Persona jurídica	Provincial	Matanzas	No se muestra	Derecho	1	2012	6	Vacío	Si		

[V indica válido y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.]

Fueron objeto de pruebas de caja negra los 13 requisitos con más prioridad para el cliente, donde se obtuvo como resultado en el registro de defectos y dificultades un total de 3 no conformidades detectadas, lo que constituye aproximadamente un 77% de satisfacción. En una próxima iteración después de haber corregido estas conformidades se obtuvo un 100% de satisfacción en los casos de prueba.

3.3.3. Pruebas de usuario

Las pruebas de usuario fueran realizadas mediante un laboratorio de aceptación una vez terminada la parte de la aplicación perteneciente al módulo, preparado por la dirección del proyecto en conjunto con la FGR. Durante 7 o 8 días se realizaron las pruebas mediante la revisión de todas las funcionalidades a cumplir, viendo las mismas desde la perspectiva de los futuros usuarios de la aplicación, comparando los resultados obtenidos mediante el sistema y mediante la formal manual utilizada hoy en día. Las variables a medir fueron las reflejadas a continuación:

Mejora de la gestión del proceso: se medirá a través del control que se lleve de los procesos y la celeridad en el tiempo en que se realice cada uno de ellos.

Control de los procesos: se medirá con el cumplimiento de términos como las búsquedas de los rollos creados en un período determinado, los reportes elaborados de cada proceso; con la visualización de los rollos creados a diario, los detalles de los mismos; y con la garantía de la persistencia de la información.

Celeridad: se medirá con la disminución del tiempo en la creación de los rollos, en la creación de los documentos y del acceso a los mismos.

Tabla 7: validación de la variable control

Dimensión	Antes	Después
Detalles de los rollos.	Muy difícil conocer los detalles del rollo, teniendo en cuenta la gran cantidad de datos que contiene el mismo (datos del expediente de revisión, del expediente del TPP o TPM, de la sentencia, del dictamen, de las personas y entidades asociadas al proceso, además de los documentos generados en el proceso).	Se puede acceder desde cualquier instancia a los detalles del rollo de manera rápida.

Búsquedas de rollos y documentos.	Se realiza de forma manual, buscando por fecha u orden alfabético, lo que es muy engorroso y contempla cierta cantidad de tiempo para encontrarlos.	Se encuentran fácilmente mediante diversos criterios de búsquedas.
Persistencia de los datos.	Los datos son siempre escritos en el proceso que se esté llevando a cabo, no importa que se conozcan o estén recogidos en un proceso anterior.	Los datos se guardan en la base de datos, permitiendo en determinados casos como asociarle personas o entidades al proceso, seleccionarlas solamente para vincularlas al mismo sin necesidad de adicionarlas nuevamente.

Tabla 8: validación de la variable celeridad

Dimensión	Antes	Después
Tiempo de demora de las búsquedas de rollos o documentos.	De 10 a 30 minutos, dependiendo del tiempo que lleva creado.	De 2 a 10 segundos. El tiempo de respuesta de la aplicación es de hasta 10 segundos para las consultas más pesadas.
Tiempo de elaboración de un rollo o documento.	De 10 a 25 minutos. Los rollos tienden a demorar más puesto que recogen mucha información.	De 10 a 20 minutos.

3.4. Conclusiones

Mediante el desarrollo de este capítulo se muestran los resultados alcanzados a través de la implementación y las pruebas de la solución propuesta en este trabajo de investigación. Se obtuvieron a raíz de la fase de implementación algunos artefactos como el modelo de despliegue, el modelo de componente y los ficheros de código. También se realizaron las pruebas de caja negra mediante la técnica partición de equivalencia y la prueba de caja blanca mediante la prueba del camino básico. Se encontraron algunas no conformidades en dichas pruebas, las cuales fueron corregidas en un pequeño período de tiempo, por lo que en su totalidad arrojaron valores satisfactorios. Mediante los resultados obtenidos con las pruebas de usuario, se llegó a la conclusión que se obtuvieron resultados muy superiores en cuanto a la celeridad y control de los procesos brindada por la aplicación a la obtenida por la forma manual.

Conclusiones Generales

Con el desarrollo de este trabajo de investigación quedan confirmadas la importancia y la necesidad de la construcción de un sistema para informatizar los procesos laborales de la fiscalía, ya que permitió resolver varios problemas que afectaban el cumplimiento de las tareas fiscales en tiempo y forma. Después de haber finalizado este trabajo y lograr el resultado esperado se concluye que:

- ✓ Se estudiaron y analizaron satisfactoriamente los sistemas existentes en el mundo relacionados con los procesos laborales, identificando sus características y ventajas que permitieron poner en función de la solución propuesta.
- ✓ Se hizo un buen uso de las tecnologías, herramientas y lenguajes seleccionados para lograr el desarrollo del trabajo en su totalidad, priorizando no violar las necesidades del país de la migración al software libre.
- ✓ Las técnicas y patrones empleados en las diferentes fases en la construcción del software permitieron llegar a resultados fiables, los cuales se pudieron comprobar en todos los artefactos obtenidos a lo largo del desarrollo.
- ✓ Se tuvo en cuenta además la aplicación de buenas prácticas en el desarrollo de software como son los estándares de codificación, logrando así el propósito de lograr un código legible y organizado para la aplicación.
- ✓ Se realizó un conjunto de pruebas al software final para comprobar la calidad del mismo como fueron: las pruebas de caja blanca y caja negra, arrojando resultados satisfactorios.
- ✓ Las pruebas de usuario realizadas a la aplicación dejaron validadas satisfactoriamente las variables de la investigación, mostrando la superioridad de tiempo empleado en realizar cualquier proceso laboral de forma manual que con la aplicación desarrollada, al igual de como se adquiere un control superior sobre los procesos que ya se encuentran guardados mediante los reportes y búsquedas por diferentes parámetros que posibilita la aplicación.

Una vez culminada la aplicación que gestionará los procesos laborales que se llevan a cabo en la FGR, se recomienda:

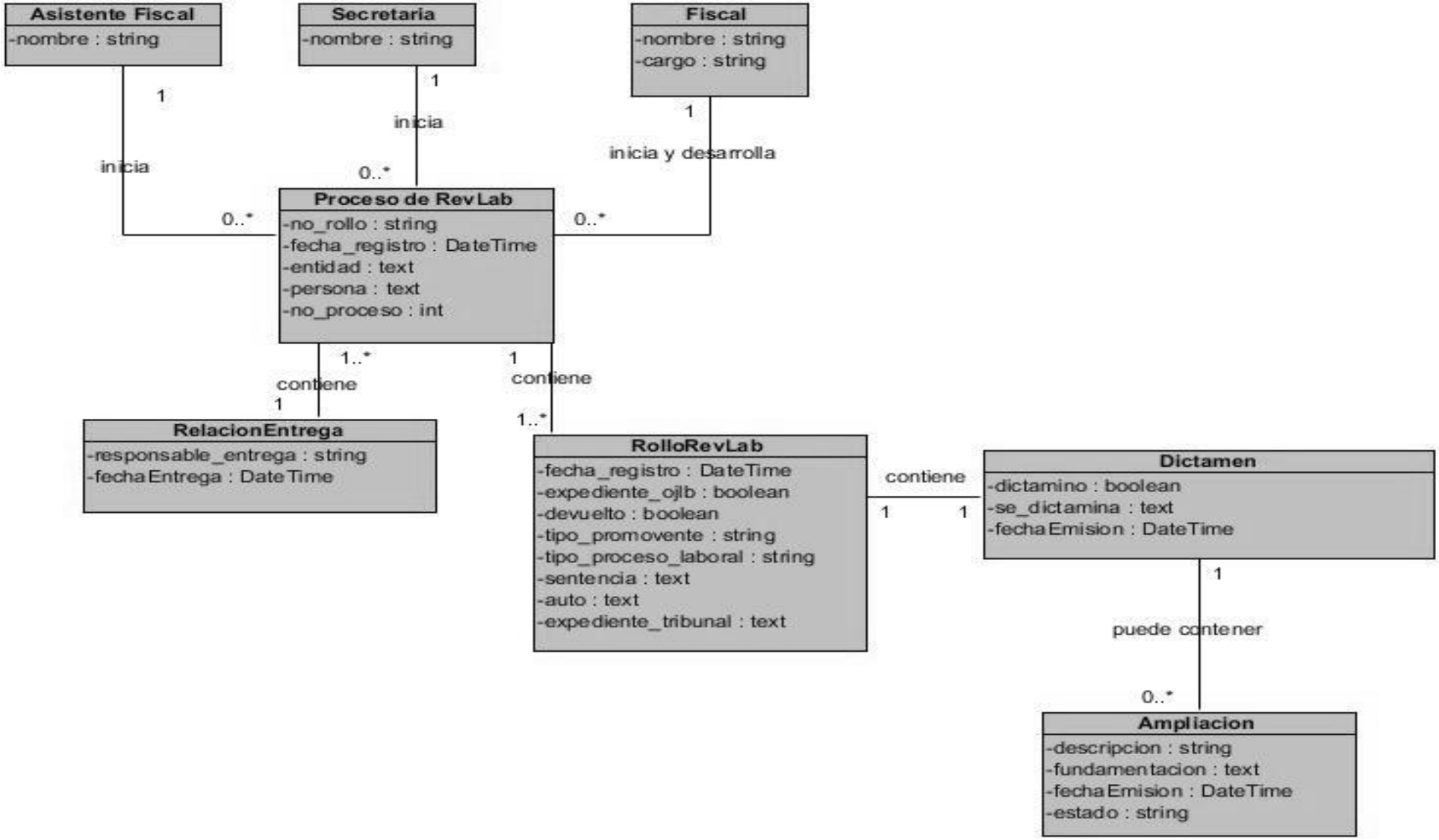
- ✓ Realizar un análisis con mayor profundidad que permita incorporar al software nuevas funcionalidades.
- ✓ Realizar una revisión con el objetivo de incorporar la gestión de nuevos reportes.
- ✓ Luego de poner en funcionamiento el sistema retroalimentarse de las experiencias de los usuarios (fiscales) y mejorar la arquitectura de información realizada.

Bibliografía

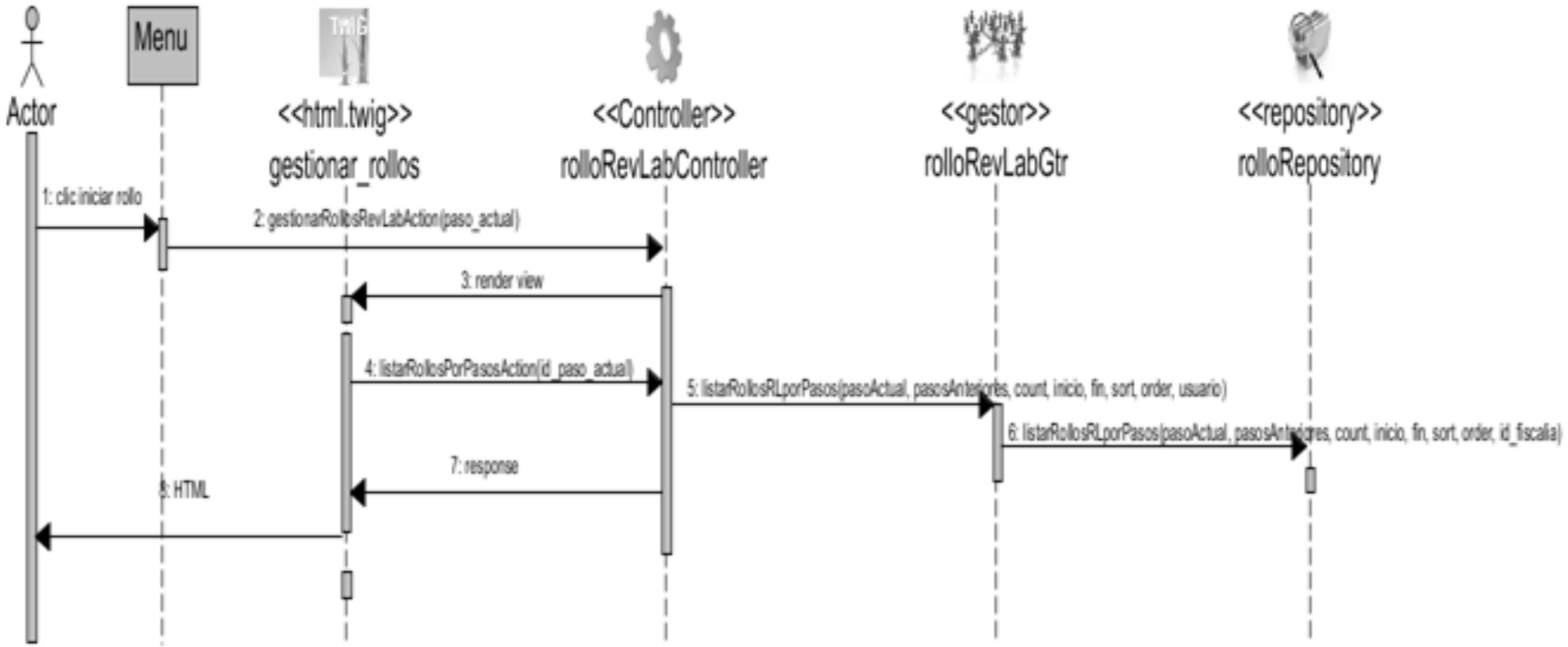
1. Conceptos Básicos de Computación. [Online] [Cited: Enero 26, 2013.] http://dcb.fi-c.unam.mx/users/miguelegc/tutoriales/tutorialcd/mpct_cmpt_scd_1.htm.
2. Gaceta Ofical. [Online] [Cited: Enero 24, 2013.] http://www.gacetaoficial.cu/html/fiscaliageneralrepublica.html..
3. **Figueroa, Yenier.** *Proyecto técnico Sigef II.* UCI, La Habana : s.n.
4. *Ley de la Fiscalía General de la República.* La Habana : s.n., 2007.
5. **Calderío, Blas Roca.** *Ley de Procedimiento Civil, Administrativo y Laboral.* 1977.
6. **Gómez, Lic. Leopoldo Sebastián M.** *Tesis de Magister en Ingeniería del Software"Sistema Informático Jurídicopara la individualización y acuerdos sobre la penaSistema JurídicoSistema penapara pena".*
7. National Criminal Justice Reference Service. [Online] [Cited: Diciembre 5, 2012.] https://www.ncjrs.gov/.
8. Lex Doctor Gestión Jurídica. [Online] [Cited: Diciembre 5, 2012.] www.lex-doctor.com/.
9. **Hector Fuentes Blancos, Yanisleidy Torres Puga, Manuel Enrique Delgado Fernández, Yenier Figueroa Machado.** *Propuesta de arquitectura de software para proyectos de gestión sobre plataformas web.* La Habana : s.n.
10. **Obregón, Ing. William González.** *MODELO DE DESARROLLO DE SOFTWARE.* La Habana : s.n., 2012.
11. Axure RP. [Online] [Cited: Febrero 12, 2013.] http://www.axure.com/features.
12. Visual Paradigm. [Online] http://www.visual-paradigm.com .
13. [Online] [Cited: Marzo 16, 2013.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p.
14. Netbeans. [Online] [Cited: Febrero 11, 2013.] https://netbeans.org/index_es.html.
15. PostgresSQL-es. [Online] [Cited: Diciembre 5, 2012.] http://www.postgresql.org.es/sobre_postgresql.
16. PostgreSQL Cuba. [Online] 2010. [Cited: Marzo 14, 2013.] http://postgresql.uci.cu.
17. PGAdmin guia ubuntu. [Online] [Cited: Diciembre 5, 2012.] http://www.guia-ubuntu.org/index.php?title=PgAdmin_III.
18. Apache Server. [Online] [Cited: Marzo 16, 2013.] http://httpd.apache.org/docs/2.2/es/new_features_2_0.html.
19. **Eguiluz, Javier.** *Desarrollo web ágil con Symfony2.* 2011.
20. Symfony. [Online] [Cited: Mayo 25, 2013.] http://www.symfony-project.org/.
21. Web.Ontuts. [Online] http://web.ontuts.com/tutoriales/utilizando-doctrine-como-orm-en-php/.
22. Eumed.net. [Online] [Cited: Diciembre 5, 2012.] http://www.eumed.net/libros-gratis/2009c/587/Lenguaje%20de%20Modelado%20Unificado.htm.
23. PHP. [Online] [Cited: Marzo 4, 2013.] http://www.php.net.
24. **Pacheco, Nacho.** *Manual de Twig.* 2011.
25. W3C. [Online] [Cited: Enero 17, 2013.] http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo.
26. Circulo de Maquetadores. [Online] [Cited: Febrero 12, 2013.] http://www.circulodemaquetadores.com/css3-impresionantes-caracteristicas/.
27. Internet Básico. [Online] [Cited: Enero 17, 2013.] http://aprenderinternet.about.com/.

28. Desarrollo web. [Online] [Cited: Marzo 11, 2013.] http://www.desarrolloweb.com/articulos-copyleft/articulo-metricas-de-software.html.
29. Lenguajes del lado del servidor o cliente. [Online] [Cited: Abril 26, 2013.] http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html .
30. PublicSpain. [Online] [Cited: Mayo 3, 2013.] http://www.publispain.com/supertutoriales/Javascript/Intro.doc..
31. JQuery. [Online] [Cited: Febrero 6, 2013.] www.jquery.com.
32. Departamento de informatica. [Online] [Cited: Enero 17, 2013.] http://www.infor.uva.es/~mlaguna/is1/apuntes/2-requisitos.pdf.
33. Scribd. [Online] http://es.scribd.com/doc/37187866/Requerimientos-funcionales-y-no-funcionales.
34. **Puente, Wilson.** Portal de Relaciones Publicas. [Online] http://www.rrppnet.com.ar/tecnicasdeinvestigacion.htm.
35. CreceNegocios. [Online] [Cited: Marzo 23, 2013.] http://www.crecenegocios.com/lluvia-de-ideas/.
36. **Sommerville, Ian.** *Ingenieria de Software.* 7ma Edicion.
37. Netvoluciona. [Online] http://www.netvoluciona.es/actualidad/La-importancia-del-diseno-68.
38. **Pressman, Roger S.** *Ingeniería de Software. Un enforque práctico.* 5ta Edición.
39. UNAM. [Online] http://www.mcc.unam.mx/~cursos/Objetos/Cap8/cap8.html.
40. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999. ISBN 970-17-0261-1.
41. **Larman.** 1999.
42. Facultad de informática. [Online] [Cited: Mayo 27, 2013.] http://www.fdi.ucm.es/profesor/jpavon/poo/2.14.mvc.pdf.
43. [Online] [Cited: Mayo 27, 2013.] http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html.
44. MSDN. [Online] http://msdn.microsoft.com/es-es/library/bb972240.aspx.
45. **Pressman, Roger S.** . *Ingeniería de software un enfoque práctico.* . 1998.
46. Calidad del Software y Métricas. Conjunto de métricas OO. [Online] www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r21722.PDF.
47. **Pressman, Roger S.** *Ingeniería de Software. Un enforque práctico.* 2002.
48. **Sánchez Fornaris, Maite y Alcantara Rabí, Dayanis.** Propuesta de una guía de métricas para evaluar el desarrollo de los Sistemas de Información Geográfica. [Online] http://vinculando.org/articulos/sociedad_america_latina/propuesta_guia_de_medidas_para_ev.
49. **S.R., Kemerer Chidamber.** *"A metric suite forsz". s.l. : IEEE Transactions on Software Engineering.* 1994.
50. **Letier, Patricio.** *Pruebas del Software.* Valencia : s.n.
51. MANUAL PRÁCTICO DE HTML. [Online] http://www-app.etsit.upm.es/~alvaro/manual/manual.html#1.
52. Guía Breve de CSS. [Online] http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo.
53. Departamento de Lenguajes y Sistemas Informáticos. [Online] http://www.lsi.us.es/docencia/get.php?id=697 .
54. Diccionario Jurídico. [Online] http://www.diccionariojuridico.mx/index.php.

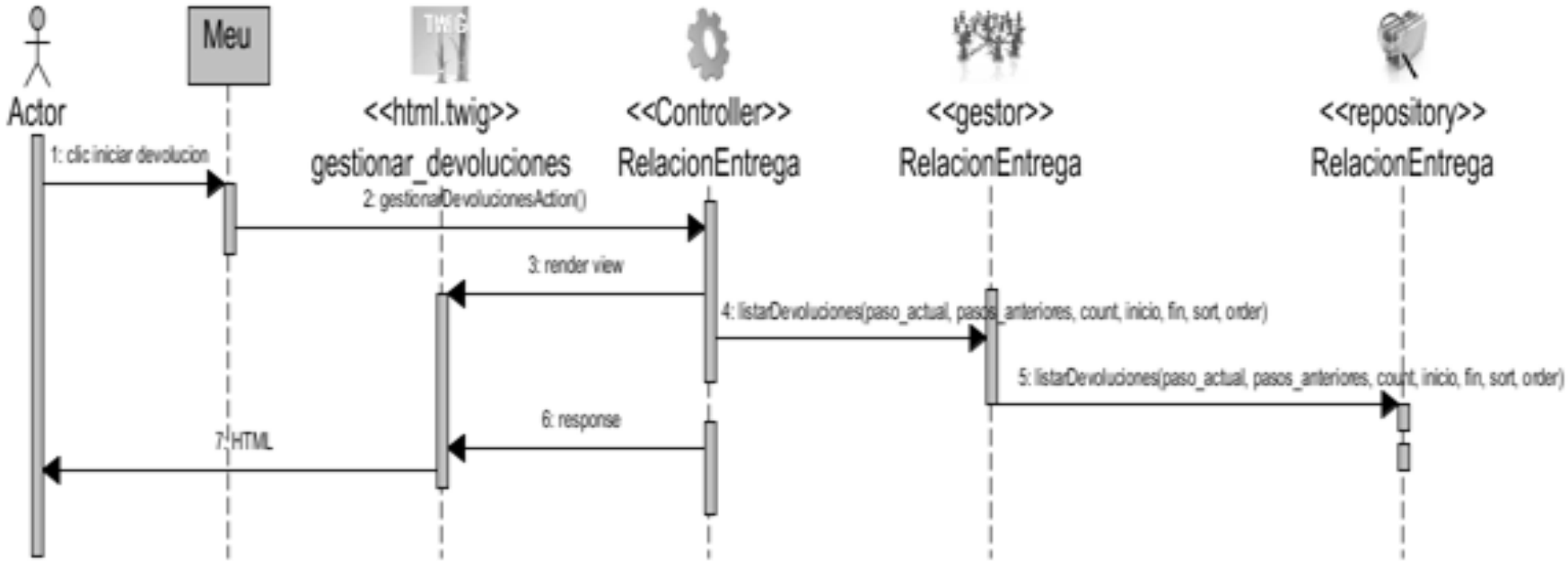
Anexo 1: Modelo Conceptual



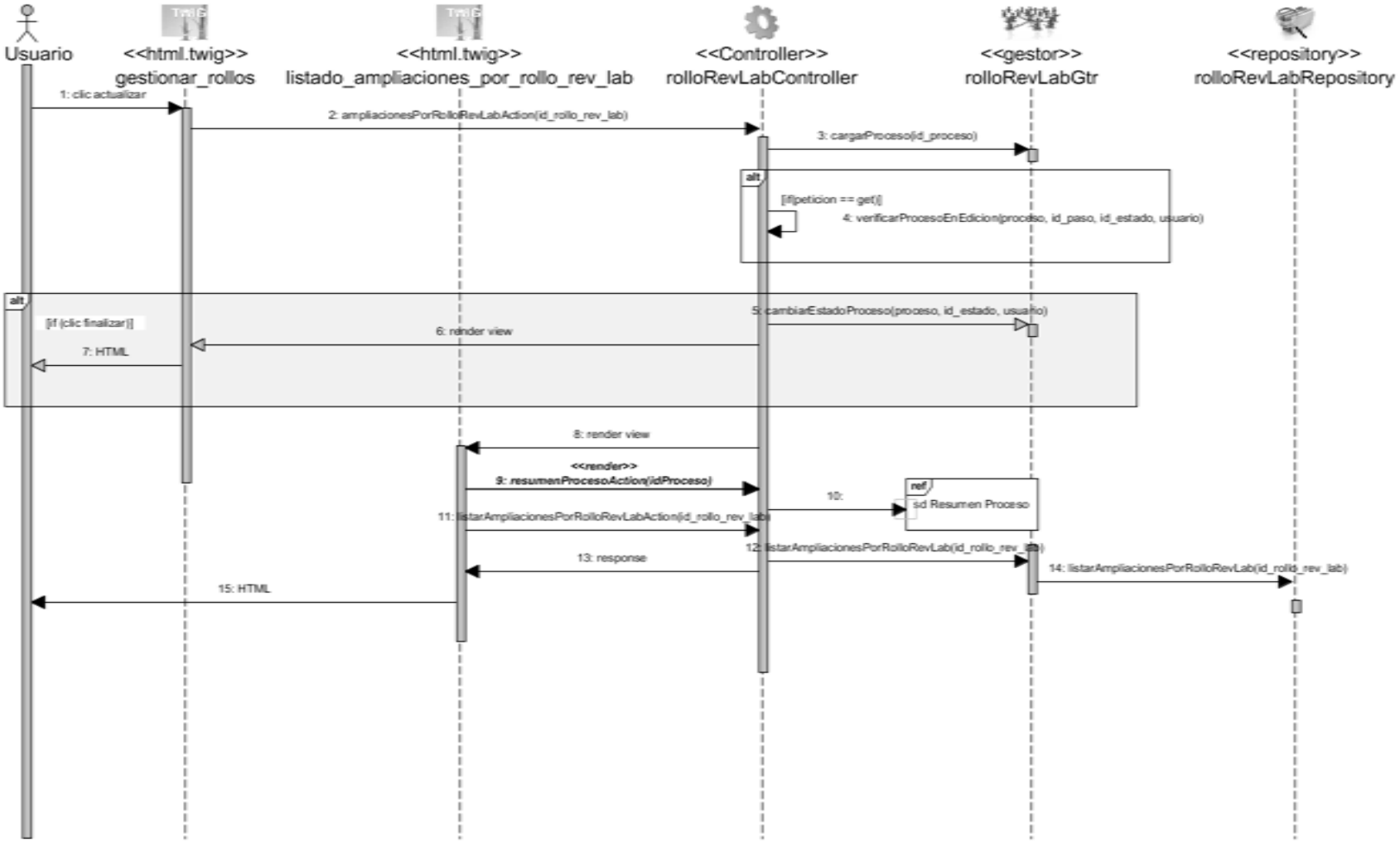
Anexo 2: DS_Listar Rollos por Pasos



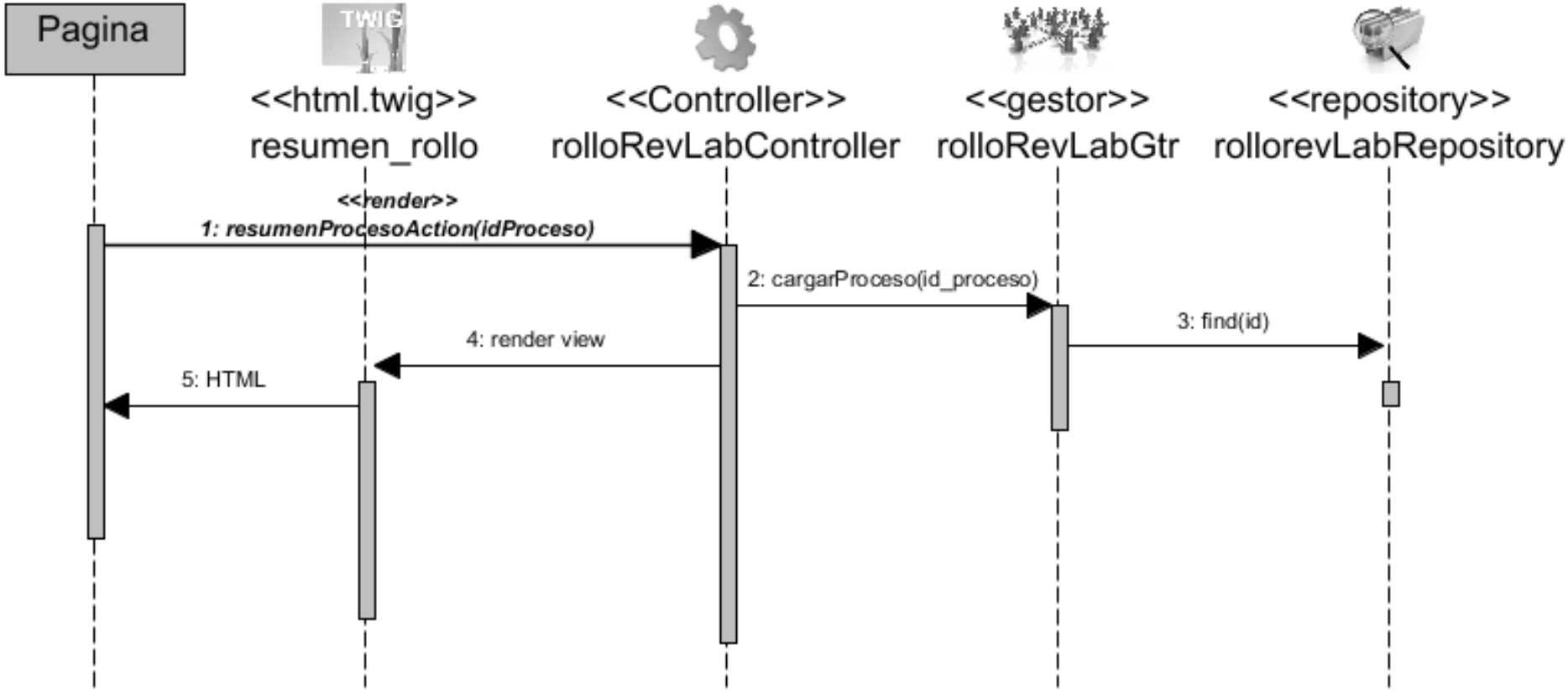
Anexo 3: DS_Listar Devoluciones por Expedientes



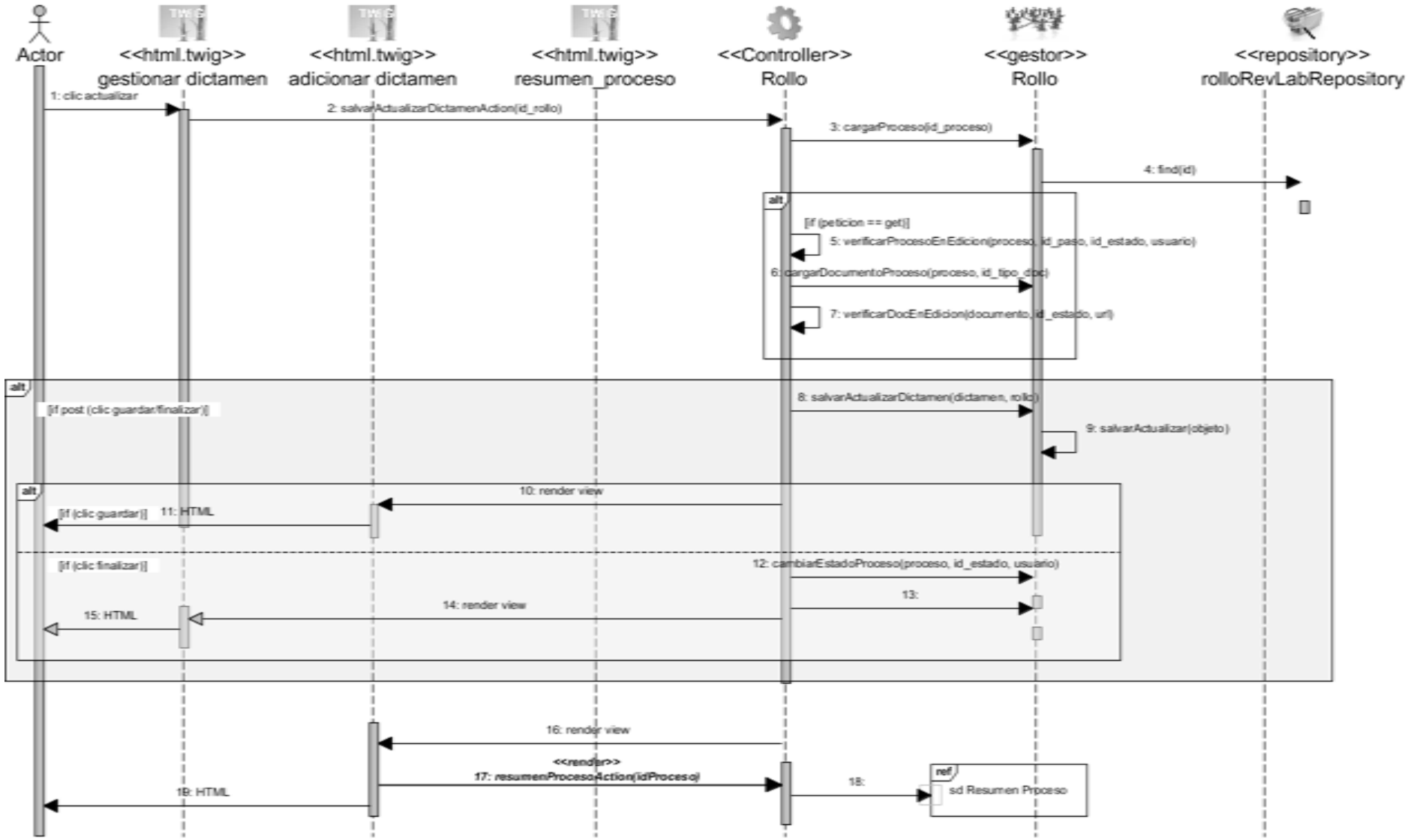
Anexo 4: DS _Listar Ampliaciones por Rollo



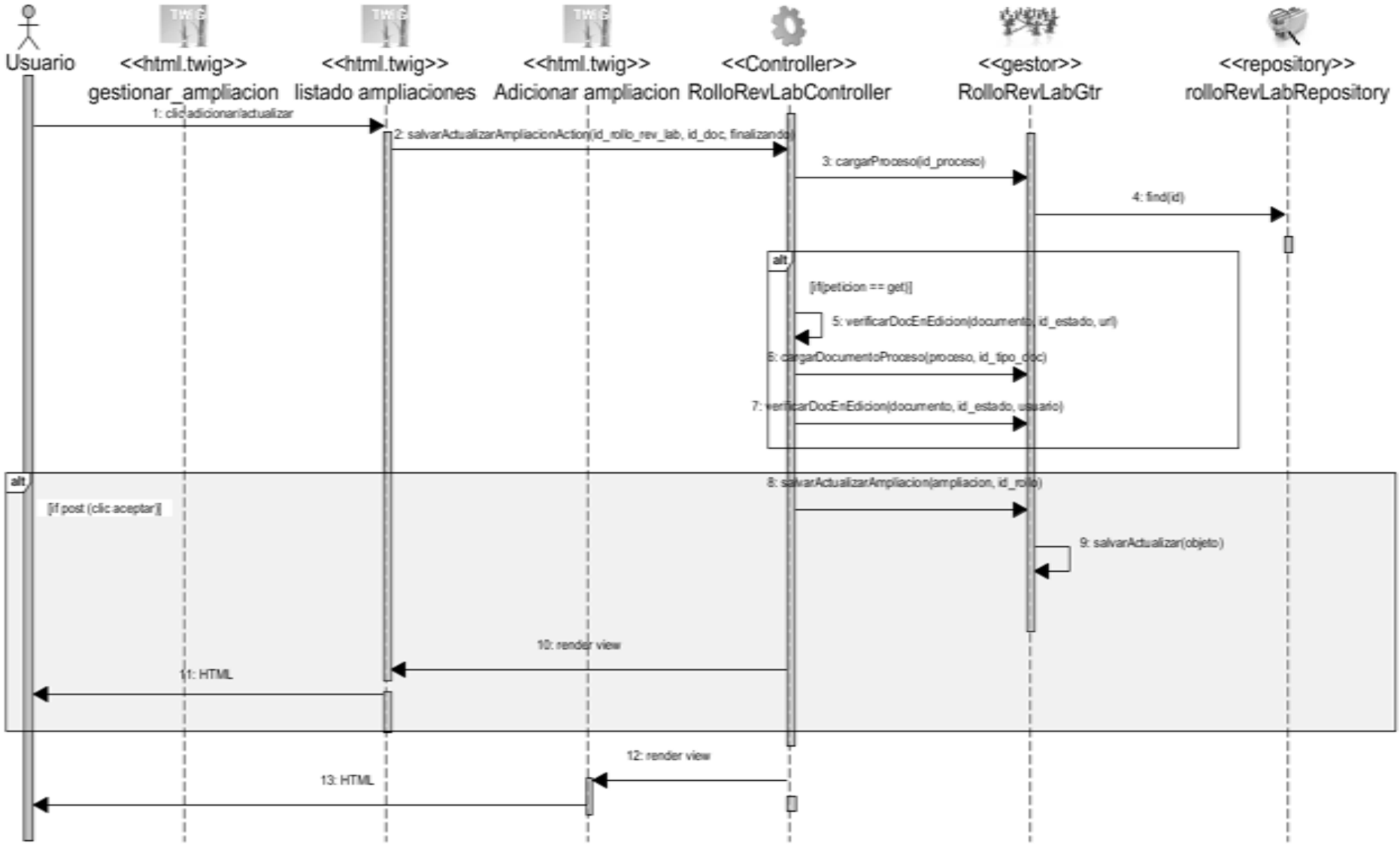
Anexo 5: DS_Resumen Proceso



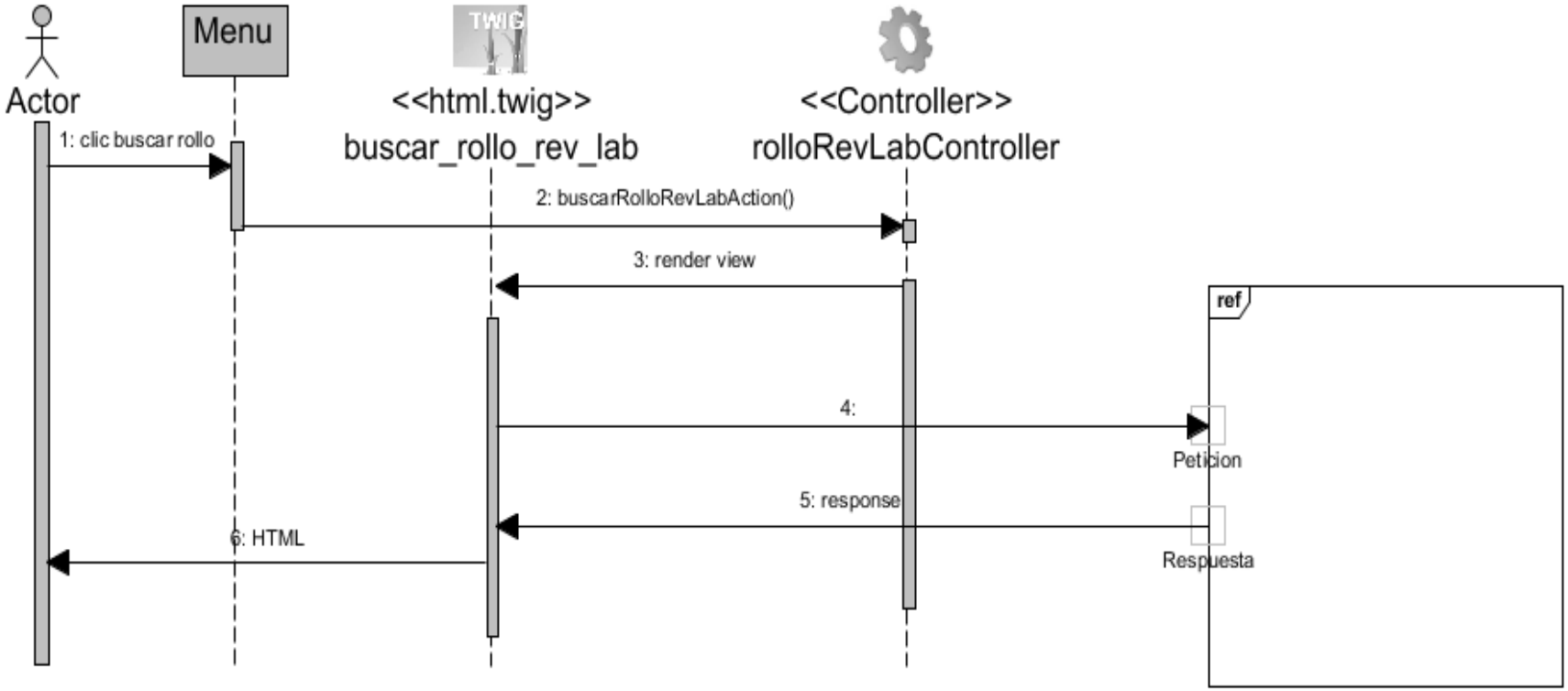
Anexo 6: DS _Adicionar/Actualizar Dictamen



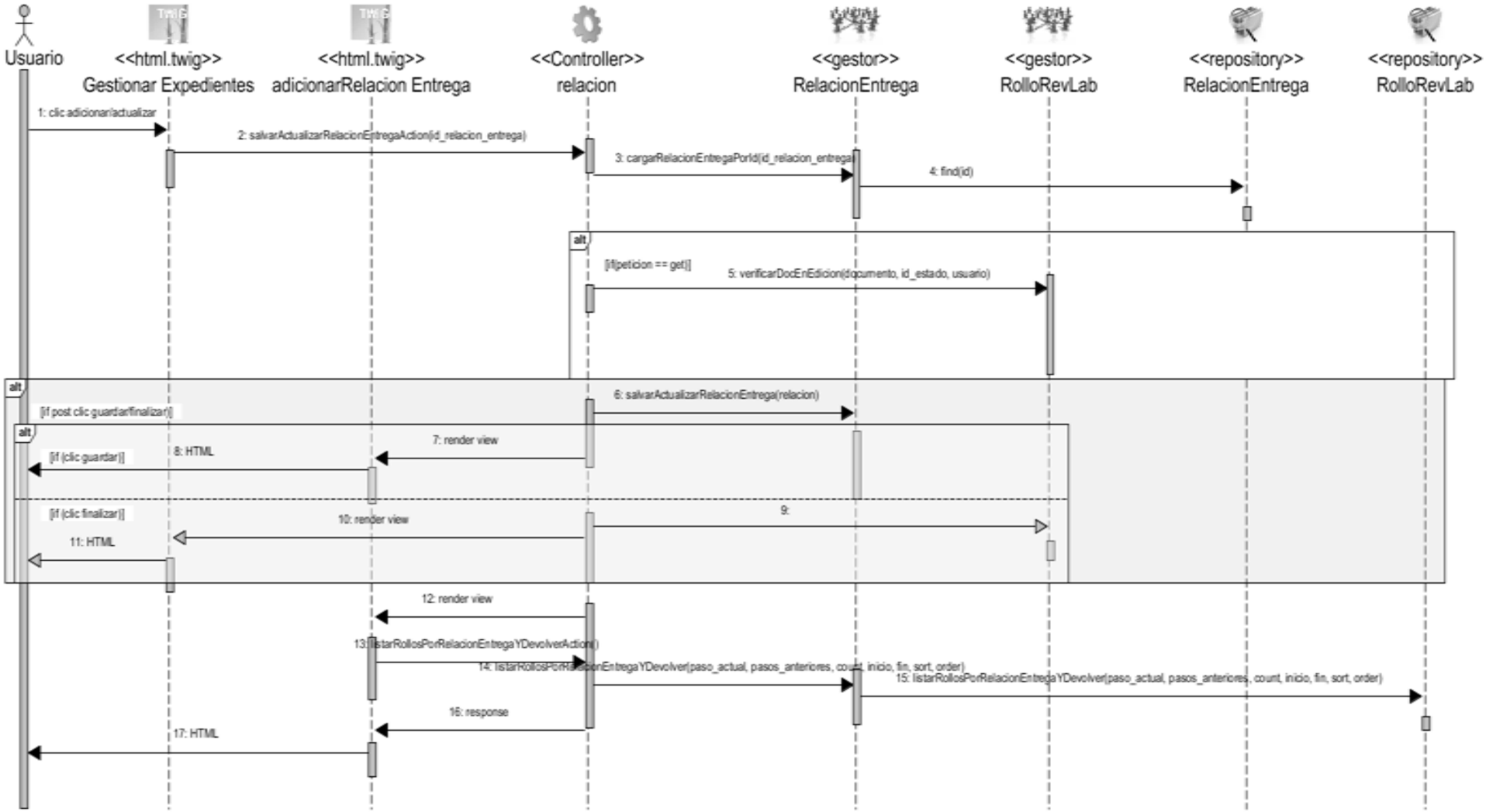
Anexo 7: DS _Adicionar/Actualizar Ampliación



Anexo 8: DS _Buscar Rollo



Anexo 9: DS _Adicionar/Actualizar Relación de Entrega



Anexo 10: DS _Finalizar Flujo