Universidad de las Ciencias Informáticas

Facultad III

Título

Implementación de los módulos Gestión de Boletas y Gestión de Documentos del Sistema Integral de Documentación e Información Judicial

Trabajo de Diploma

Para optar por el título de
Ingeniero en Ciencias Informáticas

Autores: *Hurshel Norberto Desouza Noguera*
Jorge Luis Pérez Rodríguez

Tutor: ***Ing.*** *Heiler Fabars Corrales*
Co-tutora: ***Ing.*** *Ana Ivette Ferrer Hernández*

La Habana, Cuba
Curso: 2011-2012

“El hombre debe transformarse al mismo tiempo que la producción progresa; no realizaríamos una tarea adecuada si fuéramos tan sólo productores de artículos, de materias primas y no fuéramos al mismo tiempo productores de hombres”.

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo de diploma y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Autores:

_____ _____

Jorge Luis Pérez Rodríguez Hurshel Desouza Noguera

Tutor: Co-Tutor:

_____ _____

Ing. Heiler Fabars Corrales Ing. Ana Ivette Ferrer Hernández

DATOS DE CONTACTO

Síntesis del Tutor:

Tutor: Ing. Heiler Fabars Corrales

Categoría Docente: Instructor.

Síntesis: Graduado en la Universidad de las Ciencias Informáticas. Profesor de Inglés.

Correo electrónico: hfabars@uci.cu

Teléfono: 837 2270

Años de graduado: 3 años de graduado.

Síntesis del Co-Tutor:

Co-Tutora: Ing. Ana Ivette Ferrer Hernández

Categoría Docente: Adiestrada.

Síntesis: Graduado en la Universidad de las Ciencias Informáticas. Imparte clases del curso optativo Complemento de Base Datos de la facultad 3.

Correo electrónico: aiferrer@uci.cu

Teléfono: 837 3460

Años de graduado: 1 año de graduado.

AGRADECIMIENTOS

A mi madre, por ser un ejemplo para mí, por ser madre y padre, y por ser ese motor impulsor que me guió durante todo este largo camino.

A mi padrastro, por ser como un padre para mí, por el apoyo incondicional que siempre me brindaste y por transmitirme ese ejemplo de firmeza en todo momento.

A toda mi familia por darme ese gran apoyo en los momentos más difíciles de la carrera, por ser tan preocupados y atentos conmigo.

A Melisa por tener que soportarme tantos años y darme una niña tan preciosa que es todo para mí.

A todos mis amigos, en especial a Alejandro Casanova por ayudarme esa cantidad de veces que lo fui a molestar, cuando realmente necesité ayuda en la elaboración de este trabajo.

A mi tutor y cotutora por estar siempre ahí, por su atención incondicional y eficiente.

A todos los educadores que contribuyeron a mi formación como profesional en esta universidad.

Jorge Luis

DEDICATORIA

A mi mamá y mi padrastro por todo lo que han sacrificado por su hijo durante toda su vida, por ser excelentes padres y aún mejores personas y por su apoyo incondicional durante estos largos años de sacrificio y separación.

A mi abuelita María Luisa por la paciencia y el amor que me brindó toda la vida, por educarme y quererme como uno más de sus hijos.

A mi niña Katherine, que es el tesoro más grande que me ha dado la vida.

A Melisa, por todo el apoyo que me ha brindado en los momentos difíciles y las alegrías de momentos felices. Por ser mi pareja y madre de la criatura más linda que me inspira a seguir adelante día tras día.

A nuestro Comandante en Jefe Fidel Castro Ruz, por ser nuestro guía y nuestra luz e iniciador de la Universidad de las Ciencias Informáticas.

Jorge Luis

RESUMEN

El presente trabajo tiene como objetivo principal la implementación de dos módulos de vital importancia para el proyecto Sistema Integral de Documentación e Información Judicial (SIDIJ), los cuales son Gestión de Documentos y Gestión de Boletas, el primero referido a la automatización de los procesos que tienen que ver con los fondos documentales del centro y el segundo para el registro de los servicios brindados. Para la correcta elaboración de los mismos se hizo necesario analizar los procesos y requisitos que les corresponden, así como lograr mediante un estudio previo sobre el framework, llegar a dominar y aplicar las tecnologías a utilizar durante el proceso de desarrollo, lo cual, queda explícito en el documento. Todo esto debido a la imperante necesidad de adaptarse a una arquitectura ya definida por el proyecto en cuestión.

Se obtienen además, todos los artefactos correspondientes a los flujos de trabajo de Implementación y Pruebas, basados en los elementos arquitectónicos definidos para la elaboración de los mismos. Estos artefactos son evaluados mediante un conjunto de pruebas, a través de las cuales se valida su funcionamiento y estructura. Una vez concluido el desarrollo de estos módulos se puede afirmar que el Centro Nacional de Documentación e Información Judicial tendrá a su disposición los complementos de una herramienta informática necesaria para una optimización de la gestión de los procesos que en él se desarrollan.

Palabras claves

Gestión, boletas, documentos, framework, módulos.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Los volúmenes de información y datos que rodean al hombre en la actualidad van aumentando con el transcurso del tiempo. Esto provoca que su almacenamiento, búsqueda y gestión se convierta en un proceso complejo. Por estas razones el uso de las Tecnologías de la Información y las Comunicaciones en función del desarrollo constituye una prioridad para cada empresa dentro de la optimización de sus procesos productivos. En este contexto surgen las aplicaciones informáticas como herramientas de desarrollo, a través de las cuales, y con la realización de un adecuado proceso de desarrollo de software, los resultados son cualitativa y cuantitativamente superiores.

Cuba ha destinado parte de sus esfuerzos hacia la informatización de nuestra sociedad, con el propósito de insertarse en el marco tecnológico mundial y alcanzar mayor eficiencia en los procesos que se realizan en todas las esferas del país. En Cuba es necesario un sistema que almacene y gestione la información jurídica, con herramientas de código abierto que permitan su mantenimiento, adaptaciones y versiones posteriores. En consecuencia con esto, el país se ha propuesto llevar a cabo soluciones internas que permitan satisfacer estas necesidades.

Como materialización de esta idea, surge la Universidad de las Ciencias Informáticas (UCI), la cual juega un papel protagónico con todos sus proyectos productivos, la gran mayoría de repercusión nacional, ofreciendo de esta manera soluciones propias a procesos que se realizan manualmente en muchas instituciones a nivel nacional.

La UCI cuenta con varios centros de desarrollo de software con el objetivo de informatizar cada uno de los sectores, empresas y organismos tanto cubanos como extranjeros. Cada uno de los centros se especializa en la creación de software de una rama en específico. El Centro de Gobierno Electrónico (CEGEL), encargado de la esfera jurídica en conjunto con el Centro Nacional de Documentación e Información Judicial (CENDIJ), entidad perteneciente al Tribunal Supremo Popular (TSP), determinaron la necesidad de construir un sistema integral con el objetivo de mejorar la prestación de servicios de búsqueda y gestión de información judicial, surgiendo así el Sistema Integral de Documentación e Información Judicial (SIDIJ).

Para lograr una adecuada organización de los procesos que se realizan en el centro, el sistema está formado por los siguientes módulos:

- Administración.
- Gestión de boletas.
- Gestión de documentos.
- Desarrollo de colecciones.
- Reportes.

La no culminación de los flujos de Implementación y Prueba correspondientes a los módulos Gestión de Boletas y Gestión de Documentos, ha generado un atraso en las fechas de entrega de los artefactos del proyecto, lo que ha provocado que el despliegue de una versión funcional se haya visto limitado. Esto trae consigo que en el centro, los servicios de búsqueda, préstamo, gestión de documentos y gestión de boletas se tornen lentos y engorrosos, dependiendo totalmente de un procesamiento manual de la información. En muchos casos, estos procesos no garantizan una respuesta inmediata a los clientes, lo cual representa una deficiencia significativa en la entrega a tiempo de los documentos solicitados.
Por las deficiencias que provocaría la no implementación de cada uno de estos módulos es que surge el siguiente **problema a resolver**: ¿Cuáles son las especificaciones de los artefactos necesarios para la implementación de los requisitos de software de los módulos Gestión de Boletas y Gestión de Documentos del SIDIJ?

Por lo tanto el **objeto de estudio** se enmarca en el Proceso de Desarrollo del Software y para dar solución al problema planteado se traza como **objetivo general** implementar los módulos Gestión de Boletas y Gestión de Documentos para el Sistema Integral de Documentación e Información Judicial (SIDIJ) delimitando como **campo de acción** los flujos de Implementación y Prueba en el proceso de desarrollo de software para aplicaciones web. Se define como **idea a defender** que: Desarrollando la implementación de los módulos Gestión de Boletas y de Gestión Documentos se obtendrán las especificaciones de los artefactos necesarios que contribuirán al proceso de despliegue de una primera versión funcional del sistema en el CENDIJ.

Objetivos específicos:

- Elaborar el marco teórico de la investigación.
- Realizar un estudio de los algoritmos para la búsqueda de documentos referenciales.
- Realizar la implementación de los módulos Gestión de Boletas y Gestión de Documentos a partir de los artefactos de entrada.
- Validar la solución obtenida.

Para dar cumplimiento a los objetivos planteados se han trazado las siguientes **tareas de investigación**:

- Estudio y análisis de funcionalidades desarrolladas anteriormente en sistemas jurídicos existentes que se correspondan con las que se desean implementar.
- Estudio acerca de los sistemas de recuperación de información.
- Estudio y aplicación de las potencialidades del framework [1] que tributen a la correcta implementación de las funcionalidades.
- Análisis del Modelo de Diseño y el Modelo de Datos correspondientes a los módulos a implementar.
- Análisis y aplicación de los distintos algoritmos de búsqueda de documentos.
- Implementación de los módulos Gestión de Boletas y Gestión de Documentos.
- Aplicación de técnicas de validación de Caja Negra a los módulos obtenidos.
- Validación de los módulos resultantes mediante técnicas de Caja Blanca utilizando la librería de pruebas Lime que incluye Symfony.

La realización de las tareas estuvo sustentada por un conjunto de métodos teóricos de investigación:

Método Analítico-Sintético

Su utilización posibilitará la realización de un estudio de los servicios que se prestan en el CENDIJ, la utilización de la documentación existente de cada uno de estos servicios y la manera de llevar a cabo la simulación de los mismos en un ambiente digital.

[1] Marco de trabajo, denota la infraestructura sobre la cual se reúnen un conjunto de lenguajes, herramientas y servicios que simplifican el desarrollo de aplicaciones en entornos de ejecución distribuidos. Ver Glosario de términos para más detalles.

Método Modelación

La Modelación es una reproducción simplificada de la realidad. Es justamente el proceso mediante el cual se crean modelos con vistas a investigar la realidad. Este método se utilizará fundamentalmente para modelar los artefactos que se generan en el flujo de trabajo de implementación como los diagramas de componentes y diagramas de despliegue.

Posibles resultados

Dos módulos implementados con tecnología web, que se integran al Sistema Integral de Documentación e Información Judicial contribuyendo al despliegue del mismo en el CENDIJ.

El presente trabajo consta de 3 capítulos que abordan los temas fundamentales distribuidos de la siguiente manera:

- **Capítulo 1:"Fundamentación teórica de la investigación".** Se describen los conceptos asociados al dominio del problema. Se realiza el estudio del estado del arte de los distintos sistemas de gestión relacionados con actividades fiscales o jurídicas tanto a nivel nacional como internacional, con un análisis crítico de sus ventajas y desventajas. Además se caracterizan las tecnologías y herramientas utilizadas en la solución al problema.

- **Capítulo 2: "Propuesta de solución".** En este capítulo se realiza la descripción y el análisis de la solución obtenida con el apoyo de los artefactos del análisis y diseño propuestos por los analistas en fases anteriores, como son los estándares de código a utilizar, el modelo de la base de datos, los diagramas de clases del diseño, los diagramas de colaboración y secuencia, así como la descripción de las funcionalidades a automatizar.

- **Capítulo 3:"Validación de la solución obtenida".** En este capítulo se realiza la validación de los resultados obtenidos a través de pruebas de software que garantizan la calidad de la solución propuesta.

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se realiza un estudio del estado del arte de las soluciones informáticas nacionales e internacionales con el objetivo de analizar sus características y funcionalidades, y así orientar el trabajo a la necesidad de construir dos módulos para la Gestión de Documentos y Gestión de Boletas que puedan ser integrados posteriormente al SIDIJ. También se detallan aspectos importantes de la arquitectura, de los patrones de diseño, herramientas, lenguajes y tecnologías utilizadas en el proyecto. Se estudian además, algunos conceptos asociados a los métodos de recuperación de la información, que pueden resultar de vital importancia para la selección de un algoritmo adecuado que sirva para llevar a cabo las búsquedas de documentos referenciales en los módulos del sistema.

1.1 Software existente vinculado a sistemas de documentación e información judicial

En el mundo el desarrollo de las tecnologías marcha a la par con el incremento de la cantidad de sistemas para la informatización de los procesos en las distintas esferas. La necesidad de comprender y analizar la trascendencia de los sistemas informáticos relacionados con la gestión documental en el ámbito nacional e internacional constituye un factor clave para la realización de los módulos en cuestión.

1.1.1 Sistemas internacionales

Infolex: Es un sistema español dedicado en exclusivo al mundo jurídico y puede ser utilizado en cualquier despacho independientemente de su tamaño o especialidad.

Entre sus principales características se encuentran:

- Control de Expedientes. Permite la gestión de cualquier tipo de Expedientes: Judiciales, Extrajudiciales, e Igualas. Teniendo diferentes carátulas en función de sus necesidades.

- Seguimiento del asunto. Tener un control exhaustivo del Expediente. Por cada asunto se permiten registrar todas las actuaciones que sean de su interés, como pueden ser autos, providencias, llamadas, notas, etc.
- Gestión documental. Se permite generar todo tipo de escritos derivados de la tramitación, como pueden ser demandas, escritos de trámite, informes, etc.
- Digitalización de la documentación y accesible con un simple clic.
- Fusión de documentos, escritos y plantillas, teniendo siempre toda la información disponible en el propio expediente. (Reuters, Thomson, 2012)

Lex-Doctor 9.1: Es un software argentino que permite opcionalmente gestionar las bases de datos a través de Internet y cambiar el motor de datos que gestiona la información. Este software ha sido desarrollado por "**Sistemas Jurídicos S.R.L.**", una empresa que desarrolla y comercializa exclusivamente sistemas de computación para ser aplicados a la actividad jurídica y está ubicada como líder en el mercado latinoamericano en la producción de dichos software. Es utilizado principalmente en el manejo de la gestión integral de Receptorías de Expedientes, Juzgados, Tribunales y dependencias similares de los Poderes Judiciales.
(Lex Doctor, 2012)

1.1.2 Sistemas nacionales

Software para el Control de Contratos: DESOFT ha trabajado por ir informatizando todos sus procesos internos del país. En Ciego de Ávila existe una aplicación para el control de los contratos a la que se le hizo una primera mejora. En esa aplicación los abogados son los encargados de registrar los clientes e insertar los contratos. El número del contrato es invariable, pero el software permite incorporar detalles como son: pagos, plazos de entrega y detalles sobre la venta. Los especialistas de ventas y los subgerentes tienen su sesión para hacer las consultas que necesiten. También existe la posibilidad de hacer reportes y exportar a una tabla en Excel. (Silva, 2010)

1.1.3 Análisis de los sistemas existentes a nivel nacional e internacional

Luego del estudio de las características recopiladas de estos sistemas, se pasa a realizar un análisis (*Ver tabla* 1) de los aspectos positivos y negativos que puedan facilitar la integración de sus módulos al sistema SIDIJ.

Herramientas	Base de Datos	Idioma	Tipo de software	Gestión de documentos	Gestión de Boletas	Interoperabilidad	Acceso libre a los módulos
Infolex	*SQL Server 2005*	*Español*	*Software propietario*	*Parcial*	*NO*	*NO*	*NO*
Lex-Doctor	*Oracle*	*Español*	*Software propietario*	*SI*	*NO*	*NO*	*NO*
Software para el control de contratos	*PostgreSQL 8.3*	*Español*	*Software libre*	*NO*	*NO*	*NO*	*NO*

Tabla 1 Análisis comparativo de los sistemas.

En el caso de las herramientas Infolex y Lex-Doctor, cumplen parcialmente con el parámetro de tener un módulo para la Gestión de documentos, pero la interoperabilidad de los mismos no es posible ya que son sistemas propietarios, utilizan diferentes gestores de bases de datos y el acceso libre a dichos módulos está restringido.

Después de haber estudiado y analizado estos sistemas que se relacionan con la recuperación y gestión de la información jurídica, se concluyó que no es posible la adaptación de sus módulos al SIDIJ, debido a que ninguno de estos utiliza el manejo de boletas en sus funcionalidades. Además todos mantienen un acceso privativo al trabajo con documentos jurídicos, lo que impide que se pueda profundizar más en el estudio acerca de cómo se gestionan los mismos.

1.1.4 Propuesta de solución

Teniendo en cuenta lo antes mencionado se hace necesario el desarrollo de dos módulos que permitan la gestión de documentos y la gestión de boletas, que utilicen un conjunto de herramientas libres para su elaboración y que cumplan con los requisitos necesarios para el despliegue del sistema. Todas estas características antes expuestas constituyen las bases para el desarrollo de los módulos “Gestión de Documentos y Gestión de Boletas” y su integración al sistema SIDIJ.

1.2 Métodos de Recuperación de Información

La recuperación se hace mediante consultas a la base de datos, donde se guarda la información organizada usando un lenguaje de consulta adecuado. Son varios los métodos utilizados en todo el mundo para la recuperación de la información entre los que se encuentran: Álgebra de Boole, Ponderación, Relevancia y Truncamientos. La efectividad de estos métodos depende del contexto en el que se utilice por lo que es este el principal criterio a tener en cuenta en su selección. Además ninguno de ellos se utiliza en su forma pura, debido a que lo ideal es la complementariedad de sistemas, que permitan plantear diferentes perspectivas, y determinar las deficiencias de cada uno de ellos en particular. (Codina, 1995 pp. 18-22)

A continuación se describe brevemente el funcionamiento de cada uno de estos métodos:

Ponderación: Consiste en la ponderación o asignación de un peso a los términos de indexación, de manera que a la hora de buscar un documento se puede solicitar sólo aquellos que tienen un índice más elevado, o que aparezcan en los términos empleados como palabras clave. Es común que unos criterios en la búsqueda tengan más valor que otros, por tanto la ponderación pretende darle un valor adecuado a la búsqueda dependiendo de los intereses del usuario. Los documentos recuperados se encuentran en función del valor obtenido en la ponderación. El valor depende de los términos pertinentes que contengan el documento y la frecuencia con que se repita. De forma que, el documento más pertinente de búsqueda sería aquel que tenga representado todos los términos de búsqueda y además el que más valor tenga repetidos más veces, independientemente de donde se localice en el documento (Molina, 2011).

La desventaja más notable de este algoritmo si se evalúa en el contexto del SIDIJ, radica en que las respuestas de las búsqueda solo arrojarían resultados en caso de que:

- El índice del término de búsqueda posee un valor elevado.
- El término representa una palabra clave del documento.

Lo antes expuesto garantiza que el algoritmo no concuerde con el ámbito de búsquedas de documentos referenciales del sistema integral.

Relevancia: La búsqueda por relevancia se basa en recuperar aquellos documentos en que aparece el término o términos de búsqueda con mayor frecuencia, y eliminar aquellos otros en los que no aparece o aparece muy pocas veces. Se basa en métodos probabilísticos, y los resultados se muestran al usuario ordenados por la frecuencia de aparición de los términos de búsqueda. Así

aquellos documentos en que los términos de búsqueda aparecen más veces, se mostrarán en los primeros puestos de la lista. Ello conlleva que el usuario ha de ser capaz de predecir y usar los términos de su pregunta en aquellas palabras y frases que aparecen en la mayor parte de los documentos relevantes. (Olvera Lobo, 1999 pp. 4-14)

Este algoritmo es muy utilizado cuando se realizan búsquedas sobre documentos a texto completo y con carácter general, pues las búsquedas son por aproximación y no por coincidencia. Por estas razones el algoritmo de búsqueda por relevancia no se ajusta al contexto del trabajo con documentos referenciales, pues los resultados esperados deben coincidir con lo solicitado por el usuario.

Truncamientos: La búsqueda por truncamiento brinda la posibilidad de buscar la información a partir de la raíz de un término. Esta generalmente se representa con un "*", de manera que si se efectúa una búsqueda por "Biblio*" se recogerían todos los documentos que contengan esa raíz: Bibliotecas, Bibliotecarios, Biblioteconomía…, etc. El truncamiento puede llevarse a cabo tanto por la derecha, como por la izquierda del término de búsqueda; si se busca "*oxido" serán seleccionados los documentos que llevan el término: Hidróxido, Bióxido. (Palmer, 2012)

En general este algoritmo es muy utilizado para la construcción de sistemas que funcionan como directorios, con el cual se asegura un conjunto de respuestas relacionadas con un término a medio escribir. Por lo general estas respuestas tienden a ser excesivas en información y a veces no cumplen con el resultado esperado, debido a que el contexto que abarca es tan amplio como la información que contenga la base de datos donde se realicen las consultas. Por tales motivos, se puede asegurar que este algoritmo no se adapta a los requerimientos de búsquedas del SIDIJ.

Álgebra de Boole

Internet es una inmensa base de datos. Como tal sus contenidos deben buscarse de acuerdo con las reglas establecidas para realizar búsquedas en bases de datos. Gran parte de estas búsquedas, se apoyan en los principios de la lógica booleana. Estos principios hacen referencia a las relaciones lógicas existentes entre los términos de búsqueda, a las cuales se les dio el nombre del matemático británico George Boole. (Bernie, 2011)

Las búsquedas que usan este algoritmo funcionan por la combinación de dos o varios elementos característicos del contenido de un documento, ya sea por **Inclusión** (Y) por **Ampliación** (O) o por **Exclusión** (NO) (*Ver figura* 1).

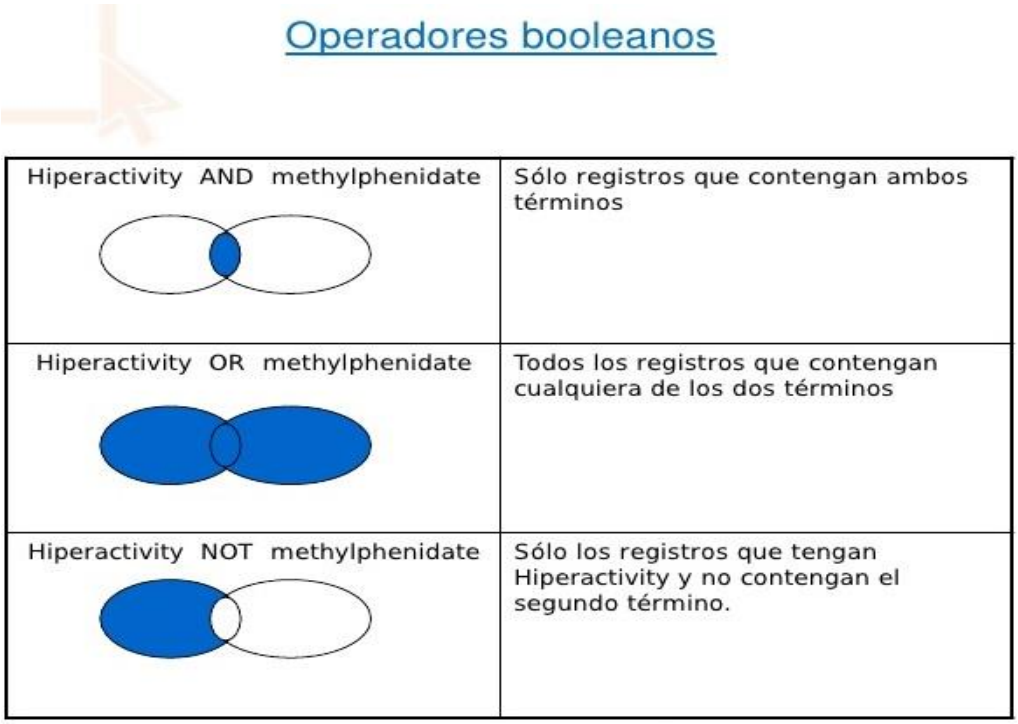

Figura 1 Uso de los operadores booleanos

”La búsqueda por **Inclusión** se realiza cuando se utilizan dos términos, y se desea recuperar solo aquella información que contenga los dos términos juntos. No tendrá valor la información que sólo contenga uno de los términos de manera aislada. Se representa por el operador lógico Y (AND en Inglés).

La búsqueda por **Ampliación** se representa a través del operador “O” (OR en Inglés), y sirve para hacer una búsqueda por cualquiera de los dos términos referenciados, independientemente de que se encuentren ambos términos juntos en el mismo documento o no. El resultado será la selección de los documentos en que aparezcan los dos términos o uno de ellos individualmente.

La búsqueda por **Exclusión** se aplicaría cuando el interés de la búsqueda recaiga en los documentos que trataran un tema, pero sin que interese un aspecto del mismo. El operador lógico utilizado para este caso es “NO” (NOT en Inglés)”. (Martínez Méndez , 2004)

La búsqueda Booleana se utiliza para recuperar información referencial en bases de datos de áreas especializadas del conocimiento. Requiere una cierta familiarización con la operatividad del sistema, pero puede definir búsquedas muy precisas, y también proporciona la posibilidad de ampliar o restringir la focalización de la misma, a un contexto similar al de los resultados obtenidos en la búsqueda inicial. Estas ventajas convierten al Álgebra de Boole en el algoritmo ideal para implementar las búsquedas de documentos referenciales en los módulos del sistema integral.

1.3 Metodología de desarrollo de software

Una metodología de desarrollo de software es un conjunto de procedimientos, técnicas, herramientas y un soporte documental para el desarrollo de productos software, puede seguir uno o varios modelos de ciclo de vida, los cuales indican qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo. La metodología indica cómo hay que alcanzar los distintos productos parciales y finales. Establece un conjunto de reglas en las que se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben jugar. Además detalla la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. (James Rumbaugh, 2000)

1.3.1 Proceso Unificado de Desarrollo (RUP)

El objetivo principal que se persigue con esta metodología es crear software de alta calidad, o sea, que cumpla con los requerimientos de los usuarios dentro de una planificación y presupuesto establecidos. Dicha metodología concibió desde sus inicios el uso de UML como lenguaje de modelado, es un proceso dirigido por casos de uso, avanza a través de una serie de flujos de trabajo (requisitos, análisis, diseño, implementación, prueba) que parten de los casos de uso; está centrado en la arquitectura y es iterativo e incremental. Además, cubre todo el ciclo de vida de desarrollo de un proyecto y toma en cuenta las mejores prácticas a utilizar en el modelo de desarrollo de software. (James Rumbaugh, 2000)
Dentro de sus características principales podemos citar:

- Forma disciplinada de asignar tareas y responsabilidades.
- Desarrollo iterativo, lo que permite ir evolucionando al producto final.
- Administración de requisitos.
- Uso de arquitectura basada en componentes, lo que facilita un alto nivel de reutilización de componentes e interfaces permitiendo así ahorrar recursos.
- Control de cambios.
- Modelado visual del software.
- Verificación de la calidad del software.

RUP genera gran cantidad de documentación que es exigida por los clientes y requerida por la dinámica del proyecto. Esta metodología robusta se adapta muy bien a la construcción de cada uno de los módulos del SIDIJ. Además los desarrolladores están familiarizados con las herramientas que

se utilizan para elaborar cada uno de los artefactos resultantes de la utilización de esta metodología. El desarrollo de los módulos Gestión de Documentos y Gestión de Boletas se enmarca en la etapa de construcción perteneciente a la misma.

1.4 Arquitectura del software

La arquitectura del SIDIJ está basada en el estilo Modelo-Vista-Controlador (MVC) que separa el código del programa en tres capas:

- La lógica relacionada con los datos se incluye en el modelo.
- El código de la presentación en la vista.
- La lógica de la aplicación en el controlador.

Para cumplir con los objetivos fundamentales del proyecto, la dirección del mismo tomó la decisión de emplear un framework (Symfony) que posibilite un desarrollo seguro, eficiente y controlado de los requerimientos identificados que debe tener y cumplir el producto final.

Mediante el uso de Symfony se toma lo mejor del estilo MVC y se implementa de forma que el desarrollo de la aplicación sea rápido y sencillo. En la arquitectura del SIDIJ, se generan las clases de la capa del modelo en función de la estructura de datos de la aplicación; la librería Propel se encarga de esta generación automática, creando la estructura básica de las clases y generando el código necesario. “Cuando Propel encuentra restricciones de claves foráneas (o externas) o cuando encuentra datos de tipo fecha, crea métodos especiales para acceder y modificar esos datos, por lo que la manipulación de los datos es muy sencilla. La abstracción de la base de datos es completamente transparente para el programador, al realizarse mediante PHP Data Objects (PDO)”. (Potencier, et al., 2010)
De esta manera, si se cambia el gestor de bases de datos en cualquier momento, no se debe volver a escribir ni una línea de código, solo se modificaría un parámetro en un archivo de configuración y el sistema continúa funcionando. La aplicación del patrón MVC a esta aplicación le resulta bastante útil debido a que le brinda un determinado nivel de independencia al software mediante la separación de las capas, así como promueve la reutilización del código fuente para la construcción de sistemas similares. Además facilita el mantenimiento del sistema en caso de posibles errores.

1.5 Patrones del diseño

Un **patrón** es una solución a un problema determinado, al que se le da un nombre, y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias

circunstancias. Durante la etapa de diseño de un software se conciben un grupo de patrones que de acuerdo a la solución que brindan, cubren una necesidad determinada en el diseño.

Existen dos grupos de patrones de diseño principales, los GRASP (Patrones Generales de Software para Asignación de Responsabilidades, siglas de General Responsibility Assignment Software Patterns) y los patrones GOF (siglas de Gang of Four que significan Pandilla de los Cuatro) que es el nombre con el que se conoce por lo general a los autores del libro Design Patterns (Larman, 1999).
Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Este grupo contiene patrones como:

- Experto
- Creador
- Bajo Acoplamiento
- Alta Cohesión
- Controlador
- Polimorfismo
- Fabricación Pura
- No Hables con Extraños
- Indirección

Los patrones de diseño GOF son 23 y se clasifican según su propósito en Creacionales, Estructurales y de Comportamiento; y según su ámbito en Objeto y de Clase. En el capítulo 2 se representan los patrones de diseño que fueron utilizados en el modelo de desarrollo del SIDIJ, así como una descripción de cómo y dónde se evidencian los mismos durante la construcción de los módulos.

1.6 Tecnologías para el desarrollo de los módulos Gestión de Boletas y Gestión de Documentos

En la confección de los módulos Gestión de Boletas y Gestión de Documentos se utilizan las siguientes herramientas:

- PHP 5.2.4 como lenguaje de programación.
- Symfony 1.4 como framework de desarrollo.
- Visual Paradigm 6.1 como herramienta case para el modelado.
- Apache 2.2.6 como servidor Web.

- ExtJS 3.0 como framework de interfaz de usuario.
- PostgreSQL 8.4 como Sistema Gestor de Base de Datos.
- Netbeans7.0.1 como IDE de desarrollo.

Todas estas tecnologías en conjunto, fueron definidas en el modelo de desarrollo por la directiva del proyecto de informatización del CENDIJ. Un profundo estudio de las mismas, el análisis y puesta en práctica de sus ventajas es de vital importancia para obtener buenos resultados.

1.6.1 PHP

Es un lenguaje de programación interpretado, ideado originalmente para la construcción de páginas web dinámicas, su acrónimo (Hypertext Preprocessor) significa preprocesador de hipertexto. El gran parecido que posee con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones. Su funcionamiento es muy básico, cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP, este procesa el script solicitado que generará el contenido de manera dinámica y el resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente. (The PHP Group, 2012)

Entre las principales ventajas que ofrece al desarrollo de los módulos del SIDIJ se encuentran:

- Es un lenguaje orientado al desarrollo de aplicaciones web [2] dinámicas con acceso a información almacenada en una base de datos.
- Su código fuente es invisible al navegador web y al cliente, producto de que el servidor se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los gestores de base de datos que se utilizan en la actualidad, destaca su conectividad con el PostgreSQL que es utilizado en la base de datos del SIDIJ.
- Capacidad de expandir su potencial utilizando módulos (llamados extensiones).

[2] "Las aplicaciones web son aquellas en que los usuarios acceden a ellas en un servidor Web a través de Internet o de una intranet" (Pereira, 2006).

- Es libre, por lo que se presenta como una alternativa para la comercialización de aplicaciones web.
- Permite aplicar técnicas de Programación Orientada a Objetos.
- Contiene una biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones, introducido a partir del PHP5 para tratar cualquier error que pueda suceder en la entrada de datos al sistema integral.
- Su utilización en el desarrollo no necesita grandes capacidades de hardware. Teniendo en cuenta los servidores, una aplicación en PHP no demanda tanto volumen de memoria RAM[3].
- Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda. Esta ayuda le sirvió a los desarrolladores en su preparación a la hora de implementar la solución.

Todas estas ventajas que ofrece PHP al programador, hace que cualquier desventaja del mismo sea intrascendente comparada con las potencialidades que ofrece su utilización.

1.6.2 Symfony

Es un framework, diseñado para optimizar el desarrollo de aplicaciones web similares al SIDIJ. Entre sus características más significativas que inciden en el desarrollo de los módulos se encuentran:

- Usa PHP5 como lenguaje de programación base.
- Utiliza el ORM (Object relational mapping) **Propel** que facilita la labor de desarrollo de aplicaciones web, gracias a la capa que transforma el tratamiento de la base de datos mediante objetos, con la que se puede recuperar, insertar y modificar datos. Con su utilización se logra rapidez en el desarrollo y abstracción de la base de datos ya que integra un lenguaje propio para realizar las consultas, lo que facilita que los usuarios dejen de utilizar las sentencias SQL y utilicen el lenguaje propio de la herramienta orientado a objetos.
- Posee soporte de diferentes ambientes (desarrollo, pruebas, producción).
- Manejo de la memoria caché de forma eficiente.

[3] Ver Glosario de términos.

- Usa YAML: Lenguaje que describe los datos con estructura similar a XML, pero con sintaxis más sencilla y ordenada.
- Brinda soporte para Ajax[4] (López, 2007).

Symfony también ofrece un conjunto de ventajas al desarrollador que son aplicadas en el sistema integral como:

- Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja.
- Automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.
- Es compatible con el gestor de base de datos PostgreSQL utilizado para gestionar la base de datos del SIDIJ.
- Genera todas las clases del modelo mediante el uso de Propel, sin necesidad que el desarrollador tenga que programarlas.
- Es independiente de la base de datos, cambiar la base de datos implica solo cambiar una línea en un archivo de configuración "**databases.yml**" y todo sigue funcionando. Esto demuestra su flexibilidad frente a los cambios que puedan ocurrir durante el desarrollo de cada módulo.
- Tiene integrados numerosos mecanismos de seguridad y la gestión de usuarios se puede llevar a cabo a través de un complemento llamado "**sfGuard"**.
- Genera de forma automática los módulos de administración de la aplicación con solo configurar un archivo yml y ejecutar el comando "**generate: module"**.
- Tiene un mecanismo de enrutamiento que integra componentes de seguridad que permiten modificar las direcciones URL para que sean más amigables, así como obtener la información necesaria de la propia ruta e integrarla con el ORM.

[4] **Ajax**: acrónimo de **Asynchronous JavaScript And XML** (JavaScript asíncrono y XML). Es una técnica de desarrollo web para crear aplicaciones interactivas

- Posee una característica llamada "configuración en cascada", la cual permite, a partir de archivos yml, configurar ya sea el proyecto completo, la aplicación o el módulo respectivamente.

- Symfony incluye clases que le permiten al programador gestionar los temas de usuario como la clase "**sfUser"**, y el control de peticiones, como las clases "**sfRequest"** y "**sfWebRequest"** de forma sencilla sin tener que programarlas desde cero.

- Trae organizadas por defecto las carpetas para colocar las clases JavaScript, los estilos CSS[5] y las imágenes. Si se usan estas carpetas es mucho más fácil trabajar con esos elementos porque la ruta es trasparente al programador, el framework maneja su ubicación y nada más se le debe poner el nombre del archivo.

- Brinda la posibilidad de cargar la base de datos en el sistema desde un archivo de configuración "**database.yml"**. Esto ahorra muchísimo tiempo de desarrollo porque se tienen todos los datos para trabajar, todo esto con un simple comando "**generate:database"** (Potencier, et al., 2010).

1.6.3 ExtJS

Para la implementación de la interfaz de usuario de los módulos Gestión de Boletas y Gestión de Documentos, los desarrolladores utilizan ExtJS producto de que es una librería JavaScript que permite construir aplicaciones web complejas y dinámicas. La misma provee un grupo de componentes con un acabado visual extraordinario y es muy potente en los temas de validación, debido a que brinda la opción de crear validadores de datos propios para los campos de los formularios, lo que es útil si se tiene validaciones que se repiten a lo largo de la aplicación.

Entre sus características principales se encuentran:

- Posee componentes de interfaces de usuario con un alto rendimiento y fáciles de personalizar.
- Brinda un modelo de componentes extensibles.
- Posee una API[6] fácil de utilizar (Frederick, 2009).
- Su licencia puede ser de código abierto o comercial.

[5] Ver Glosario de términos.
[6] API es acrónimo de Application Programming Interface. Ver Glosario de términos para otros detalles.

Entre las ventajas más significativas que brinda ExtJS a los desarrolladores del SIDIJ se encuentran**:**

- Tiene incluidos la mayoría de los controles de los formularios web, con tablas para mostrar datos y elementos semejantes a las aplicaciones de escritorio. Un ejemplo de ellos son los formularios, paneles, barras de herramientas, menús y muchos otros.

- Dentro de su librería incluye componentes para el manejo de datos, lectura de XML[7], lectura de datos JSON[8] e implementaciones basadas en AJAX.

- Permite crear aplicaciones complejas utilizando componentes predefinidos, así como un manejador de layouts. Por esto provee una experiencia consistente sobre cualquier navegador. (Cobas Díaz, 2011)

Utilizar esta librería para la construcción de las interfaces gráficas de cada módulo permitió tener relevantes beneficios como:

- Existe un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.
- Comunicación asíncrona. En este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta (Frederick, 2009).
- Eficiencia de la red. El tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico (Frederick, 2009).

1.6.4 PostgreSQL

PostgreSQL es un sistema gestor de base de datos objeto-relacional, bajo licencia BSD[9]. Este se ha convertido en el sistema de gestión de bases de datos de código abierto más potente del mercado, constando ya con una arquitectura suficientemente probada, que se ha ganado una sólida reputación de confiabilidad, integridad de datos y corrección.

[7] XML (eXtensible Markup Language): Es un meta-lenguaje que nos permite definir lenguajes de marcado adecuados a usos determinados.
[8] JavaScript Object Notation, Notación de Objetos de JavaScript.
[9] Ver Glosario de términos.

PostgreSQL es multiplataforma, brinda soporte para los lenguajes más populares, incluyendo PHP que es el utilizado en la construcción de los módulos del SIDIJ. Este utiliza un modelo cliente/servidor y usa multiprocesos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto, de esta manera el sistema continúa funcionando. Es un sistema muy seguro que funciona muy bien con grandes cantidades de datos (PostgreSQL, 2008).

A continuación se enumeran algunos de los beneficios que proporciona este gestor de bases de datos en la implementación de los módulos del SIDIJ:

- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, etc. También permite la creación de tipos propios.
- Incorpora una estructura de datos de tipo arreglo.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias.
- Soporta el uso de índices, reglas y vistas.
- Permite implementar una gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- Soporta casi toda la sintaxis SQL (incluyendo sub-consultas, transacciones, tipos y funciones definidas por el usuario).
- Es altamente escalable. Tanto en la cantidad bruta de datos que puede manejar como en el número de usuarios concurrentes que puede atender, realizando esto de forma más eficiente que muchos otros gestores.

Los datos son gestionados por una aplicación gráfica llamada PGAdmin III que es una de las más completas y populares con licencia Open Source. Está escrita en C++ y utiliza una librería gráfica multiplataforma, permitiendo que se pueda usar en sistemas operativos como: GNU/Linux y Windows. Es capaz de gestionar versiones y posee una interfaz gráfica que soporta todas las características de PostgreSQL y facilita enormemente la administración.

1.6.5 NetBeans

NetBeans IDE es un producto libre y gratuito sin restricciones de uso, de código abierto, escrito completamente en Java, teniendo el código fuente disponible para su reutilización bajo las licencias CDDL y la GPL. Es multiplataforma, presenta una interfaz muy amigable e intuitiva, soporta disímiles lenguajes, además en sus versiones más recientes se le han ido acoplando nuevas funcionalidades que resultan de gran utilidad para el desarrollo web, facilitando la creación de proyectos en PHP. (Duarte , 2008)

En su versión 7.0.1 presenta un sistema para examinar todos los directorios de cada proyecto, haciendo reconocimiento y carga de clases, métodos y objetos. Todo esto para acelerar el desarrollo, proponiendo un esqueleto para organizar el código fuente. El editor conjuntamente integra los lenguajes como HTML, JavaScript y CSS, permitiendo la refactorización y búsqueda de usos para CSS y lenguajes de tipo HTML, y el completamiento de links para atributos de CSS. Por otra parte el editor de PHP es muy ágil y robusto, siendo eficiente en el reconocimiento de sintaxis y presentando un potente depurador[10] que facilita la programación (Oracle, 2012).

En el modelo de desarrollo del proyecto SIDIJ, la aplicación de NetBeans como entorno de trabajo aprovecha muchas potencialidades a la hora de programarse en sus módulos. Entre las ventajas que ofrece esta herramienta a los desarrolladores se pueden encontrar:

- Se le pueden añadir un conjunto de complementos que ayudan al desarrollo de la aplicación web.
- Posee un completamiento de código acorde con PHP, que agiliza el trabajo con las funcionalidades que ya traen implementadas tanto el framework como el NetBeans.
- Permite crear aplicaciones web con PHP 5 y además se integra desde su versión 6.8 con Symfony, framework utilizado a lo largo del modelo de desarrollo que permite desarrollar aplicaciones de forma más sencilla y productiva. Además Netbeans puede ejecutar todas las tareas de Symfony, incluso pasándole argumentos y opciones, visualizando el resultado sin necesidad de utilizar una consola de comandos externa (Potencier, 2009).

[10] Un **depurador** (en inglés, **debugger**), es un programa usado para probar y depurar (eliminar los errores) de otros programas (el programa "objetivo").

1.6.6 Servidor Web Apache

Apache es un servidor web gratuito, potente y que ofrece un servicio estable y sencillo de mantener y configurar. Es indiscutiblemente uno de los mayores logros del Software Libre. Alguna de sus características:

- Es multiplataforma, aunque idealmente está preparado para funcionar bajo Linux.
- Muy sencillo de configurar.
- Es un software de código abierto [11](Open-source).
- Contiene amplias librerías de PHP a disposición de los programadores.
- Posee diversos módulos que permiten incorporarle nuevas funcionalidades, estos son muy simples de cargar (Apache.org, 2012).

Las principales ventajas que ofrece a los desarrolladores del SIDIJ a la hora de programar, se encuentran reflejadas a continuación:

- Es un servidor altamente configurable de diseño modular. Resulta muy sencillo ampliar sus capacidades producto de que existen muchos módulos para Apache que son adaptables a éste.
- Es compatible con el lenguaje que se utilizó (PHP) y otros lenguajes de script al poseer todo el soporte que se necesita para tener páginas dinámicas.
- Es posible configurarlo para que ejecute un determinado script cuando ocurra un error en concreto.
- “Posee un alto nivel de configuración en la creación y gestión de trazas. Permite la creación de ficheros de traza a medida del administrador, de éste modo logra un mayor control sobre lo que sucede en el servidor”. (Identi, 2011)

En la arquitectura básica del SIDIJ, el uso de Apache le brinda seguridad y confiabilidad al sistema, ya es uno de los servidores más utilizados en el mundo actualmente. Su uso puede resultar tan potente como flexible, y puede decidir cuáles de los módulos del proyecto serán ejecutados en el servidor.

[11] Ver Glosario de términos.

1.6.7 Visual Paradigm

Para el modelado de los artefactos correspondientes a la implementación de los módulos se utiliza Visual Paradigm debido a que es una potente herramienta CASE para visualizar y diseñar elementos de software. Utiliza UML[12] como lenguaje de modelado, y soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, implementación, pruebas y despliegue. (León, 2008)

Sus principales características son:

- Está disponible en múltiples plataformas (Windows, Linux).
- Hace que el diseño permanezca centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Soporta aplicaciones Web.
- Posee una generación de código: Modelo a código, diagrama a código.
- Contiene un editor de figuras.
- Interoperabilidad con modelos UML2.
- Permite elaborar diagramas de flujo de datos.
- Posee una extensa documentación que sirve de guía a la hora de usar sus componentes para modelar.

El soporte para la aplicación Web y su integración con el NetBeans representan las ventajas más notables que aprovechan los desarrolladores. Esta herramienta se ha utilizado para modelar todos los artefactos correspondientes a la etapa de implementación y pruebas de los módulos del SIDIJ, ya que gracias a la licencia que posee la UCI, se puede hacer uso de Visual Paradigm y explotar al máximo sus potencialidades.

[12] UML (Lenguaje Unificado de Modelado): Es un lenguaje gráfico para visualizar, especificar, construir, documentar y comunicar los artefactos de un sistema de software.

1.7 Conclusiones parciales

En este capítulo se mostró una panorámica sobre la forma en que se desarrollan aplicaciones de software jurídicas en el mundo, mostrando algunas de las tendencias predominantes en esta actividad, las cuales aportaron importantes ideas sobre el desarrollo de módulos de búsqueda y gestión. Es preciso destacar que a pesar de la importante teoría incorporada, relacionada con el estudio mencionado, ninguna de estas tecnologías satisface los requerimientos específicos del negocio del CENDIJ. Por esto se decide llevar a cabo el desarrollo, atendiendo a algunas de estas características y otras específicas, de forma tal que cumpla con todas las necesidades de los clientes. Además en este capítulo se abordaron conceptos sobre la metodología utilizada y los patrones de diseño, así como las herramientas y tecnologías usadas para el desarrollo de los módulos teniendo en cuenta características fundamentales que se ajustan al entorno de trabajo. A partir del estudio realizado, resultó que Symfony en conjunto con las otras tecnologías conforman el entorno para llevar a cabo la construcción de los módulos. La mayoría de estas herramientas empleadas por los desarrolladores son libres, cumpliendo con la idea de independencia tecnológica que se sigue como principio de desarrollo en el país.

.

CAPÍTULO 2

PROPUESTA DE SOLUCIÓN

Introducción

En el presente capítulo se tratan los elementos y características relacionadas con la implementación de los módulos Gestión de Boletas y Gestión de Documentos. Se comienza haciendo una descripción de cómo está concebida la arquitectura y algunos rasgos esenciales en el funcionamiento del framework, con el objetivo de brindar una mayor comprensión del funcionamiento de los módulos implementados. También se muestran los patrones empleados durante el desarrollo, así como la implementación que se siguió para la elaboración de los mismos. Se expone además una valoración crítica de los artefactos propuestos en las fases Análisis y Diseño que anteceden a esta investigación.

Se hace una descripción de las clases y operaciones utilizadas, además, se describen los estándares de codificación aplicados para una mejor legibilidad del código. Por último se muestra el modelo de implementación que pone en práctica el diseño de la solución realizado en etapas anteriores, y se especifican los artefactos generados durante esta etapa que constituyen el flujo de salida en conjunto al código fuente de los módulos representando la solución en general.

2.1 Arquitectura del Sistema Integral de Documentación e Información Judicial

Como se menciona en el capítulo anterior, el desarrollo del SIDIJ responde a una arquitectura en tres capas, la cual representa al patrón arquitectónico Modelo–Vista–Controlador (MVC). Este es de los más utilizados en aplicaciones web, debido a que permite desarrollar sistemas fácilmente escalables y sencillos de mantener. Dicho patrón divide la aplicación en tres capas o niveles de abstracción las cuales son: Modelo, Vista y Controlador.

La estructura que define el framework Symfony para la separación de estas capas deja bien delimitado donde quedan las clases del modelo, las de la vista y las controladoras, que para los módulos Gestión de Boletas y Gestión de Documentos resulta como se muestra a continuación:

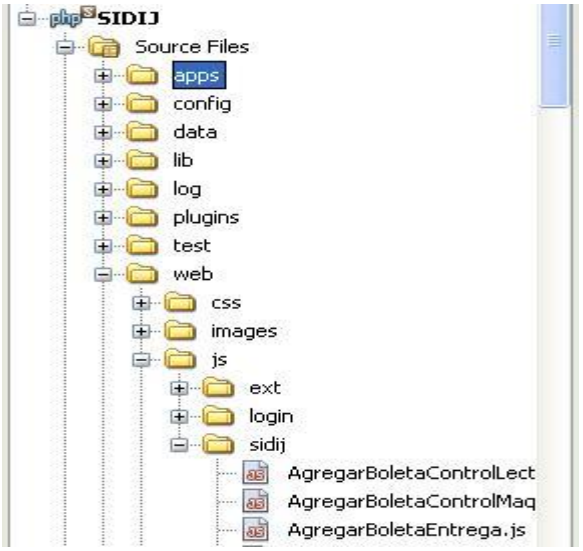

Figura 2 Capa de la vista paquete web.

Las clases que se encuentran dentro del paquete "**SIDIJ/Source files/web/js/sidij**" (*Ver figura* 2), en conjunto con las que importan que se encuentran en "**SIDIJ/Source files/web/js/ext**" son las que responden a la capa de la vista, que son las que transforman el modelo en una página web que le permite al usuario interactuar con ella, para esto se apoya en la utilización de la librería ExtJS, que facilita la construcción de interfaces con un buen acabado y agradables a la vista del usuario.

Figura 3 Capa del controlador paquete apps.

Las clases controladoras son las que se encuentran dentro del paquete "**SIDIJ/Source files/apps/sidij/modules/sidij/actions**" (*Ver figura* 3), estas son las encargadas de procesar las

interacciones del usuario, recibir peticiones de la vista y gestionar la misma. Además si es necesario, solicitan funcionalidades del modelo y realizan cambios apropiados en la capa del modelo o la vista según sea el caso.

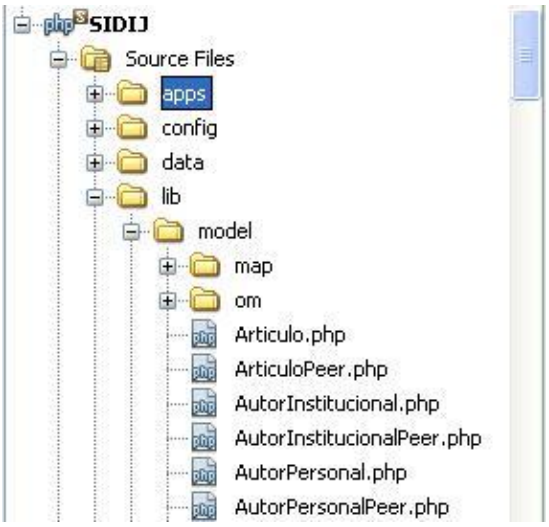

Figura 4 Capa del modelo de los módulos.

Las clases que se encuentran en el paquete "**SIDIJ/Source files/lib/model**" (*Ver figura* 4), van a ser las que responden al modelo, que son las que representan la información con la que trabaja la aplicación. Dentro del paquete "**model**" se encuentran las clases con sufijo "***Peer***" que son las encargadas de insertar, modificar o eliminar objetos de las clases entidades del modelo de datos, dentro del paquete "**om**" se encuentran las clases "***base***[13]***"*** que no son más que las entidades de cada uno de los módulos.

2.2 Patrones GRASP utilizados

Patrón Experto: Este es uno de los patrones que se utilizan al trabajar con el framework de desarrollo Symfony, un ejemplo de esto es la inclusión de Propel como ORM en la construcción de los módulos (*Ver figura* 3). Este genera las clases para la gestión de las entidades con las responsabilidades debidamente asignadas según el patrón Experto, pues cada una de estas clases cuenta con un conjunto de funcionalidades relacionadas directamente con la entidad que representan.

[13]Son las clases que genera el ORM Propel, que contienen los mismos atributos y relaciones que existen en el modelo de datos.

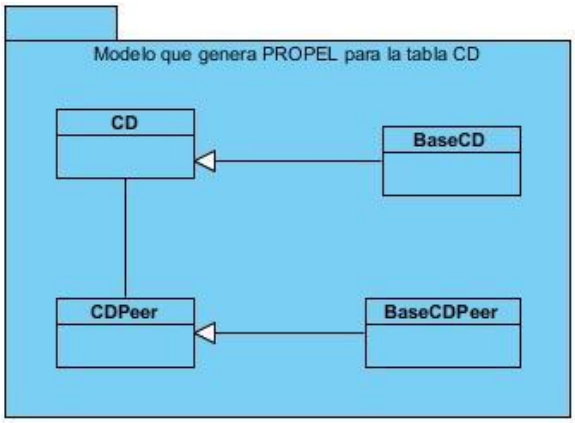

Figura 5 Paquete de modelado para la tabla CD usando PROPEL.

Patrón Creador: En la clase controladora **Actions.php** (*Ver figura* 6) se crean objetos de las clases que representan las entidades, evidenciando de este modo que la clase controladora es creador de dichas entidades.

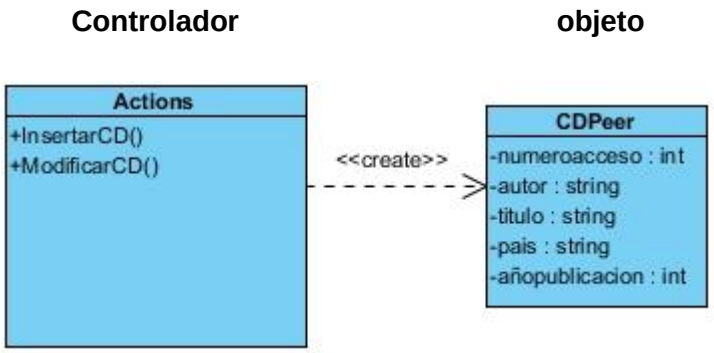

Figura 6 Aplicación del patrón Creador en la clase Actions.

Ejemplo de código perteneciente a la clase Actions.php:

```
public function executeInsertarCD() {
$objetoCD = new Cd (); //Se crea una instancia de CD
$objetoCD->getNumeroAccesoCD();
}
```

Patrón Bajo acoplamiento: Asigna una responsabilidad para mantener bajo acoplamiento. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de

muchas otras. Este patrón está evidenciado en el framework Symfony debido a que dentro de la capa modelo, las clases de abstracción de datos (*Ver figura* 7) son las más reutilizables y no tienen asociaciones con las clases de la capa de la vista ni con el controlador.

Ejemplo:

```
class CDPeer  extends BaseCDPeer {

public function AgregarCD ($nroacceso, $titulocd, $id_documento,….) {

$cd = newCd();

        $cd->setNumeroAccesoCd($nroacceso);

        $cd->setTitulo($titulocd);

        $cd->setIdDocumento($id_documento); …

}
```

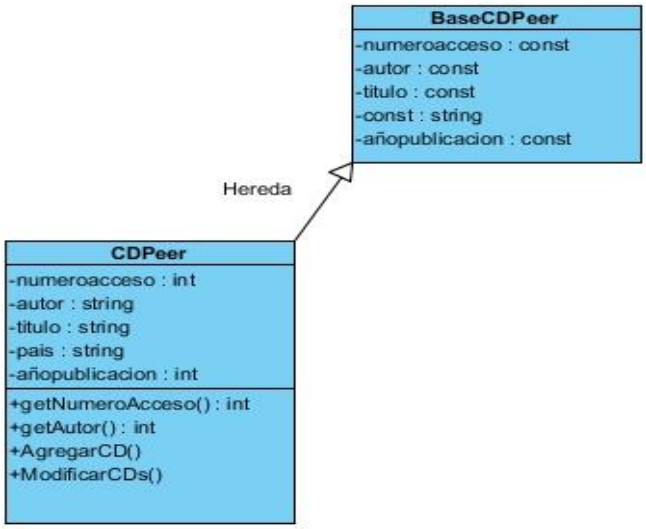

Figura 7 Aplicación del patrón Bajo Acoplamiento.

Patrón Alta cohesión: Una de las características principales de Symfony es la organización del trabajo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases con una alta cohesión. Por ejemplo, las clases con sufijo **actions** en su nombre contienen varias funcionalidades estrechamente relacionadas entre ellas, teniendo un sentido común y un propósito único, siendo las encargadas de controlar las acciones de las plantillas y por lo tanto pertenecen a la capa del

Controlador dentro de la arquitectura MVC. Esto hace posible que el software sea flexible a cambios sustanciales con efecto mínimo.

Patrón Controlador: Symfony es un framework basado en el patrón arquitectónico MVC, que define una capa específica para los controladores, que son el núcleo del mismo. Este patrón se puede observar en las clases "**sfFrontController**", "**sfWebFrontController**", "**sfContex**" propias de Symfony. También tiene una estructura bien organizada de sus controladores que parte desde la página "**indexSuccess.php**" y se cumplimenta en la clase "**actions**". Cada clase de esta capa tiene su responsabilidad y es única, por ejemplo, hay controladores que se encargan de la seguridad del sistema trabajando con ficheros "**yml**".

Ejemplo:

```
protected function getController($moduleName, $controllerName, $extension) {

//Código fuente….

}
```

2.3 Patrones GOF aplicados.

Patrón Solitario: Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En el controlador frontal de los módulos hay una llamada a **sfContext**::**createInstance**(). El objeto **sfContext** (*Ver figura* 8) contiene referencias a todos los objetos internos de Symfony como la petición, la respuesta, el usuario, entre otros. Debido a que **sfContext** actúa como instancia única, se puede hacer uso de la instrucción **sfContext::getInstance**() para obtenerlo en cualquier punto de la aplicación y por tanto, para tener acceso a los objetos internos de Symfony (Potencier, 2009).

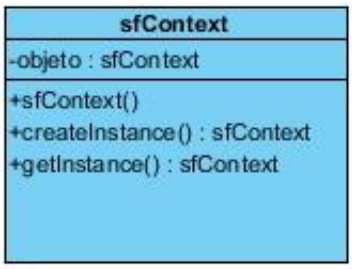

Figura 8 Funcionamiento del patrón Solitario.

Patrón Decorador: En este método de la clase abstracta "**sfView"** padre de todas las vistas, tiene cada una un decorador para permitir añadir funcionalidades a las vistas dinámicamente. Este patrón se observa en el archivo denominado "**Layout.php"** que contiene el layout[14] (plantilla global) de todas las páginas del registro. El mismo almacena el código HTML que es común a todas las páginas del registro, para no tener que repetirlo en cada página, por lo que el layout decora la plantilla.

En la *figura* 9 se muestra una imagen donde se puede apreciar la aplicación del patrón Decorador en el SIDIJ.

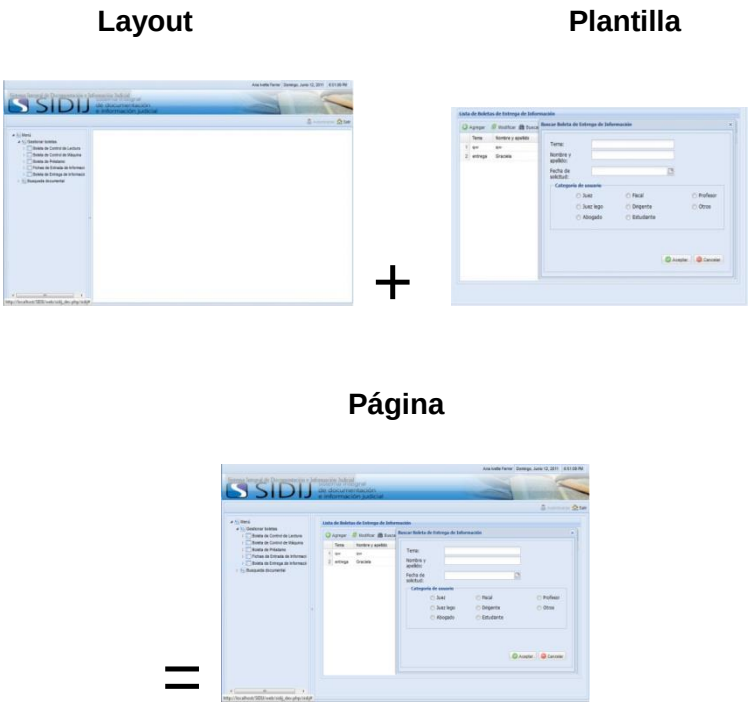

Figura 9 Funcionamiento del Layout.

2.4 Valoración de los artefactos de entrada

Como consecuencia de la aplicación de una metodología de desarrollo de software se tiene conformado un modelo de desarrollo del sistema que consta de 25 requisitos que engloban (7 casos de usos de gestión y 8 casos de uso de búsquedas referenciales). Se cuenta con todos los artefactos necesarios que constituyen las entradas para comenzar el flujo de trabajo de implementación de los módulos Gestión de Boletas y Gestión de Documentos, los cuales son:

[14] Ver glosario de términos.

- Casos de uso correspondientes a los módulos Gestión de Boletas y Gestión de Documentos.
- Requisitos funcionales correspondientes a cada caso de uso antes definido.
- Especificación de los requisitos funcionales.
- Modelo de Datos.

Uno de los artefactos generados que constituyen entradas para esta etapa es la "**Especificación de Requisitos"** en su versión 1.1, propuesto por los analistas del sistema para describir los servicios que se deben ofrecer en el software y las limitaciones asociadas a su funcionamiento. En dicho artefacto se detallaron los requisitos funcionales y no funcionales obtenidos a partir de la extracción de información de las actividades que se desean informatizar en el CENDIJ.
Los artefactos recibidos reúnen las condiciones necesarias, lo cual facilita la información necesaria para establecer la funcionalidad a alcanzar con cada módulo y reflejarse todas las necesidades de los clientes.

2.5 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en los requisitos funcionales y restricciones importantes. Este modelo tiene como entrada esencial el resultado del análisis, o sea el modelo de análisis y define una abstracción del funcionamiento del sistema, siendo utilizado como entrada fundamental de las actividades de implementación, puesto que es considerado como un plano para estas. Este modelo lo componen varios artefactos, de los cuales se abordan el diagrama de casos de uso del sistema de los módulos, los diagramas de clases del diseño y los diagramas de interacción.

2.5.1 Diagrama de casos de uso del sistema de los módulos Gestión de Boletas y Gestión de Documentos.

En el documento de casos de uso del sistema en su versión 1.2 se aprecia la descripción de los módulos Gestión de Boletas y Gestión de Documentos en términos de casos de uso y su interacción con los actores del sistema. Este documento se hace imprescindible para la implementación de las funcionalidades al mostrarse en él de una manera más específica las características que van a ser implementadas. El diagrama que se muestra a continuación (*Ver figura* 10) utiliza una serie de estereotipos para la representación de los actores del sistema, así como de los casos de uso y las relaciones entre ellos. Las relaciones vienen dadas a partir de la existencia de un actor general que inicializa los casos de uso Autenticar usuario, Visualizar listado de términos y Buscar información

referencial. De este usuario general heredan un conjunto de actores como el Catalogador, el cual inicializa el caso de uso Gestionar Documento que extiende del caso de uso “Buscar información referencial” antes mencionado. El Especialista en servicios y la Recepcionista son los otros actores que heredan del usuario general, los cuales se encargan de inicializar cada uno de los casos de uso referentes a la Gestión de Boletas, y extienden del caso de uso “Buscar boleta”. De esta manera quedan plasmados los actores del sistema y casos de uso relacionados con los módulos en cuestión.

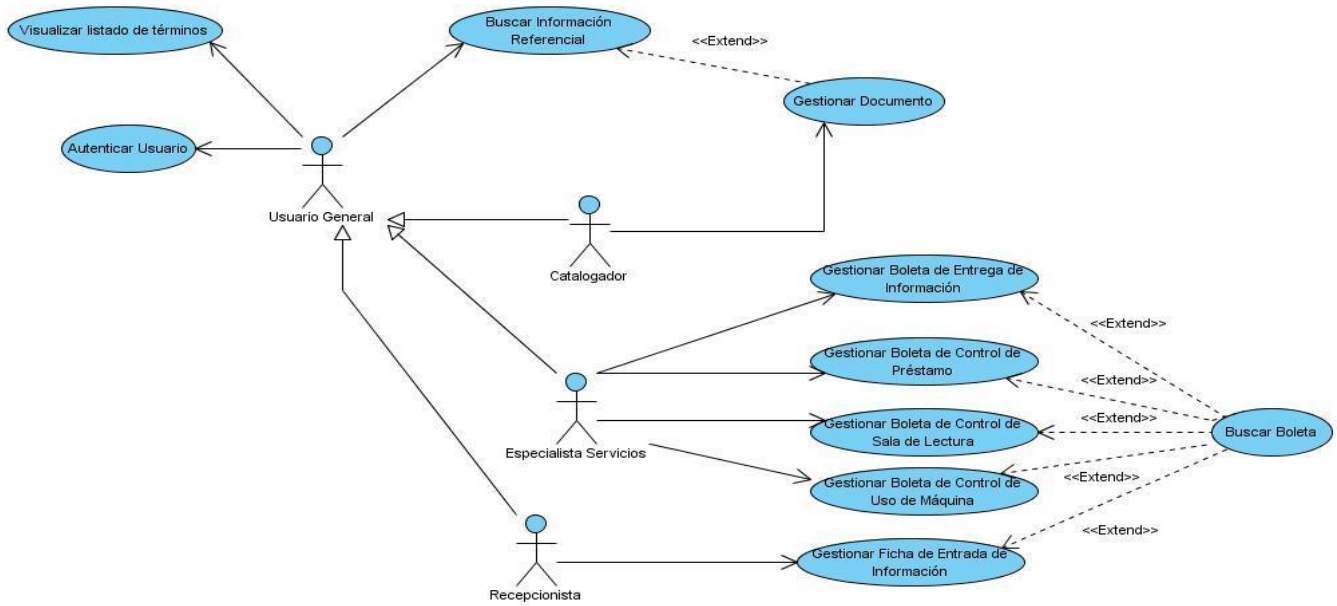

Figura 10 Diagrama de casos de uso del sistema para los módulos.

Este diagrama permite a los desarrolladores tener una visión más general de la estructura y función de los módulos que se van a implementar.

2.5.2 Diagramas de clases del diseño

Estos diagrama son una representación estática de los elementos de la solución del sistema, mostrando la especificación para las clases software de una aplicación, presentando como notación de los elementos que lo forman (clases e Interfaces) y las relaciones que existen entre los mismos (asociaciones). Las clases se representan mediante una caja subdividida en tres partes, en la parte superior se muestra el nombre, en el medio los atributos y debajo las operaciones (funcionalidades) (Desarrollo orientado a objetos con UML, 2004).

Se utilizaron además estereotipos web para la representación de los formularios, los archivos JavaScript, los archivos CSS, las páginas clientes y las páginas servidoras. A continuación se muestra la representación del diagrama de clases del diseño del caso de uso **Gestionar Boleta _Sala_Lectura** (*Ver figura* 11)**.** Este diagrama le permitió a los desarrolladores tener una visión de cada una de las relaciones entre las clases pertenecientes al caso de uso representado. En combinación con el diagrama de secuencia representan las bases para que cualquier otro desarrollador pueda darle continuidad al trabajo, sin tener un conocimiento previo de los requisitos funcionales. Por todo esto, representa un importante artefacto de entrada para el flujo de implementación, donde los que desarrollan el software deben seguir al pie de la letra cada una de las especificaciones de estos diagramas para asegurar el correcto funcionamiento del software.

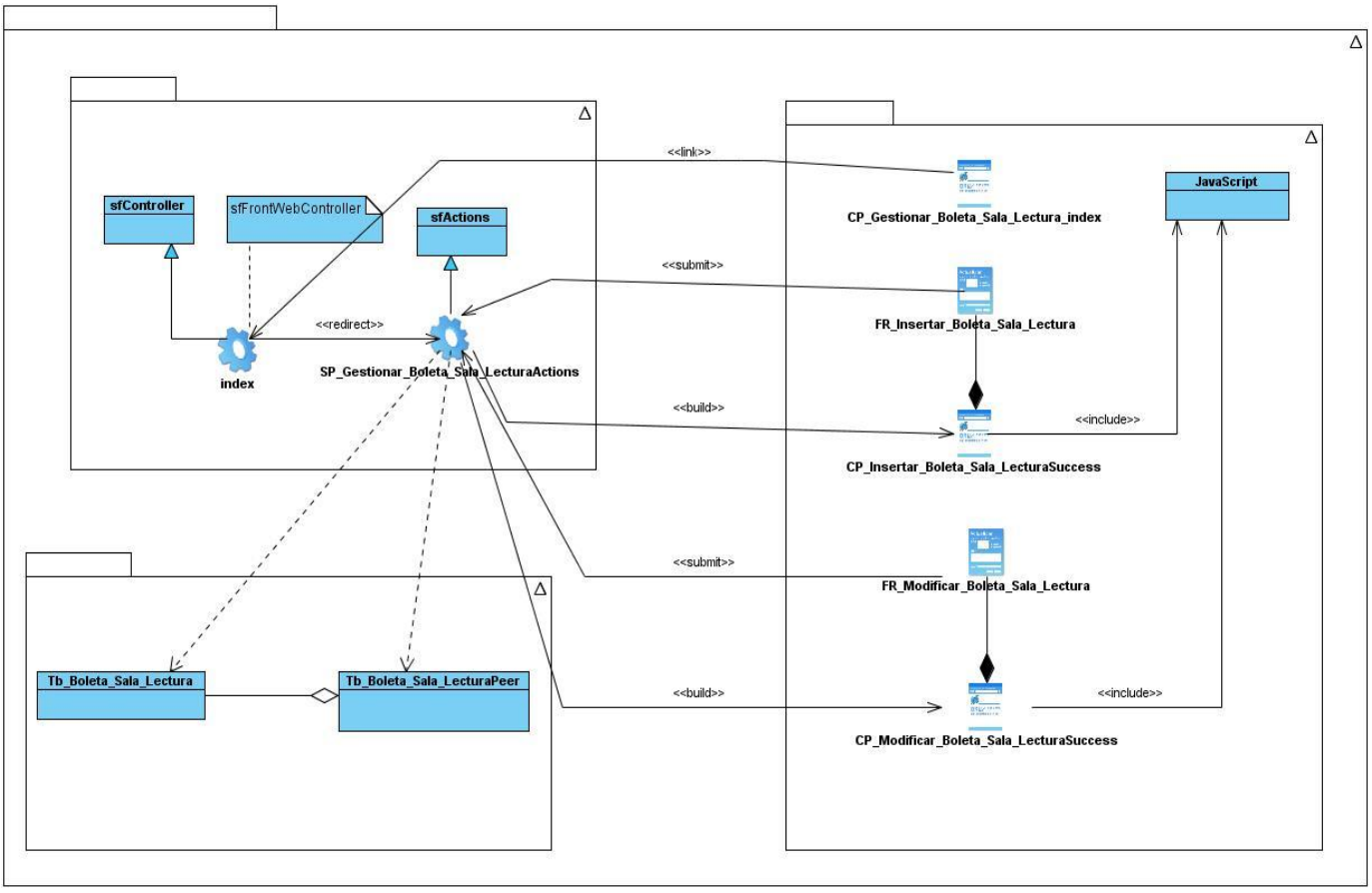

Figura 11 Diagrama de clases del diseño del caso de uso Gestionar Boleta Sala Lectura.

2.5.3 Diagramas de interacción

En la utilización de la metodología RUP existen dos tipos de diagramas de interacción, ambos se basan en la misma información y muestran un patrón de interacción entre objetos, pero cada uno de ellos se enfatiza a un aspecto en particular. Estos diagramas son: el diagrama de colaboración y el diagrama de secuencia.

2.5.4 Diagramas de secuencia

En la investigación solo se abarcan diagramas de secuencia (*Ver figura* 12) para comprender el funcionamiento de cada flujo de eventos a la hora de llevar a cabo el desarrollo de los módulos. Los mismos muestran una iteración ordenada según la secuencia temporal de los eventos, es decir muestran los objetos que interactúan y los mensajes que se intercambian ordenados según la secuencia de tiempo en que se realiza cada uno de ellos. De esta manera queda establecido para el eje vertical la representación del tiempo y en el horizontal es donde se colocan los objetos y actores participantes en la interacción. Estos últimos no tienen que tener un orden prefijado pues cada uno tiene una línea vertical hacia donde se establecen los mensajes que se muestran mediante flechas, estableciendo el tiempo de arriba hacia abajo (Desarrollo orientado a objetos con UML, 2004).

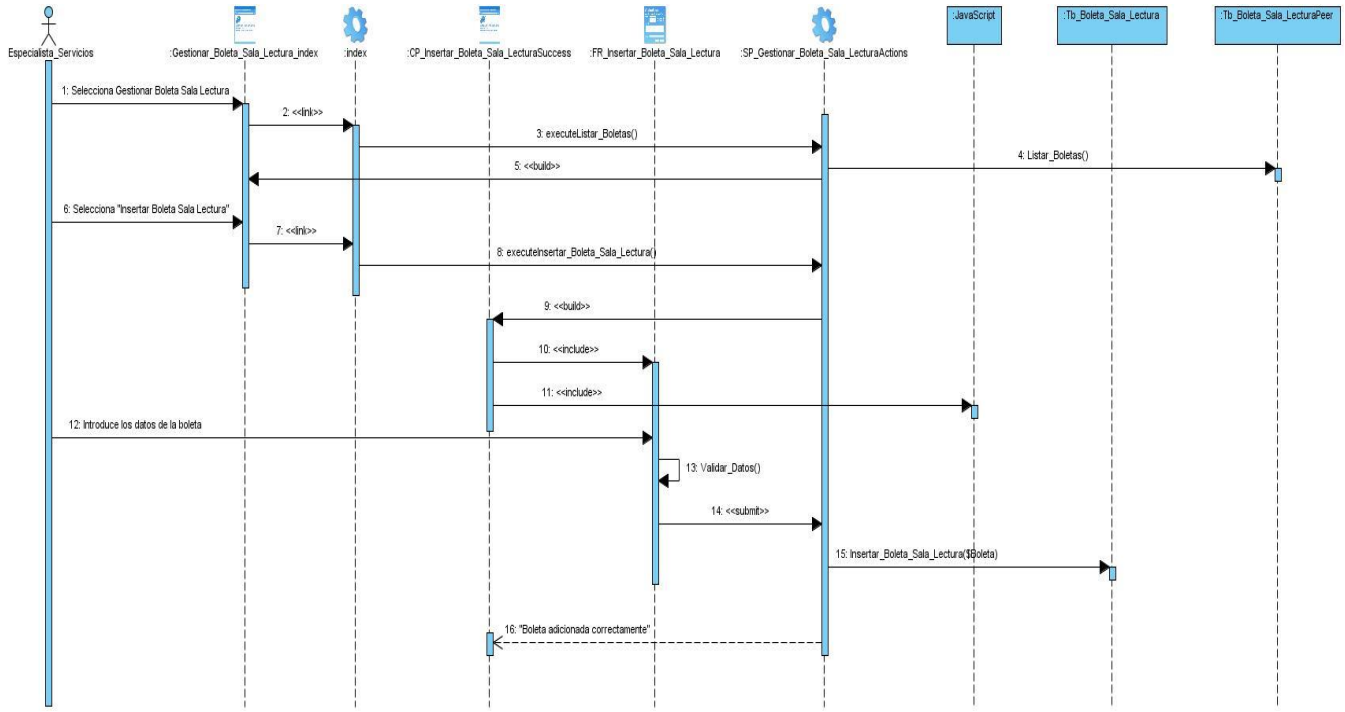

Figura 12 Diagrama de secuencia del escenario Gestionar Boleta Control de Lectura.

2.6 Modelo de datos

El modelo de datos es el proceso que implica crear una representación que tienen los usuarios de los datos. Básicamente está formado por tres elementos fundamentales: los objetos, que son todas las entidades que manipulan los datos a persistir; los atributos, que son las características básicas de los objetos antes mencionados; y las relaciones, que son las que enlazan a dichos objetos entre sí.

El modelo de datos del SIDIJ está constituido por un total de 93 tablas incluyendo las tablas generadas por relaciones de muchos a muchos y 393 atributos. Específicamente los módulos Gestión de Boletas y Gestión de Documentos hacen uso de un total de 39 tablas incluyendo las que representan relaciones de muchos a muchos lo cual representa aproximadamente el 42 % del total. Para la construcción se tuvieron en cuenta todos los datos que se necesitan persistan en el sistema. La nomenclatura de dichas entidades es de la siguiente forma: para las tablas de datos tb<Nombre> y para las que almacenan relaciones entre entidades tb<Nombre1><Nombre2>.

Se trabajó principalmente con las tablas relacionadas a descriptores así como con boletas para la gestión de las mismas. Además se trabajó con documentos de tipo referencial para la inserción y búsquedas específicamente, siendo las tablas más usadas. A continuación se muestra un ejemplo de la tabla "**publicaciones seriadas**" (*Ver figura* 13) una de las más complejas por la cantidad de relaciones y atributos que posee.

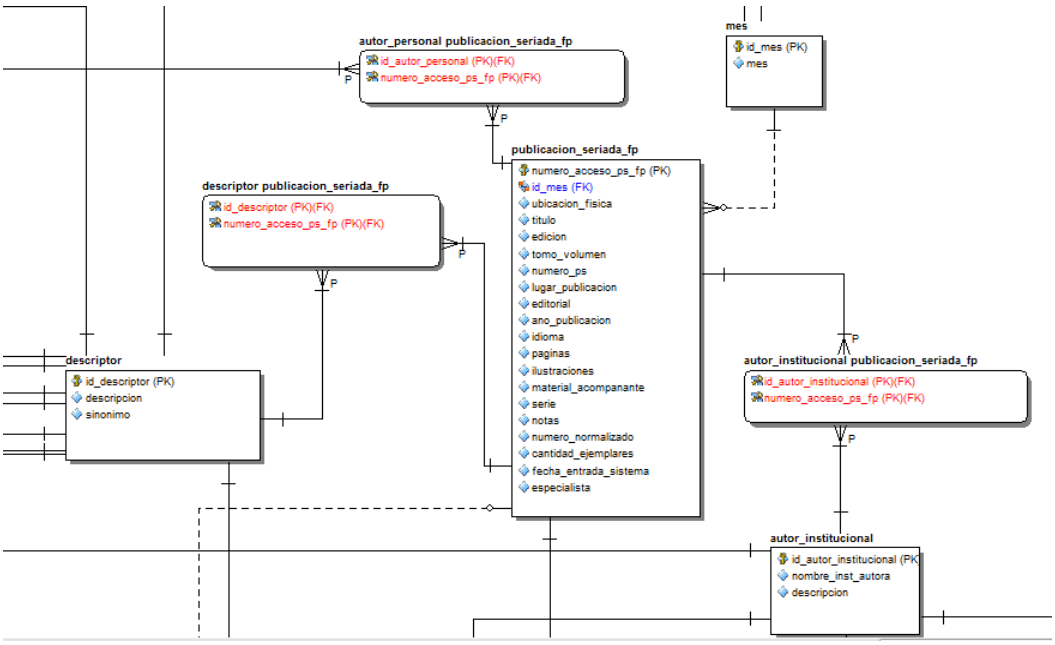

Figura 13 Modelo de datos para Publicaciones seriadas.

2.7 Algoritmo de búsqueda utilizado

Para conformar el buscador del SIDIJ, fue utilizado el algoritmo Álgebra de Boole porque el mismo se ajusta más al contexto del trabajo con manipulación de descriptores. Además es un algoritmo apropiado para la búsqueda de documentos referenciales que arroja resultados bastante ajustados a lo que el usuario solicita. Es necesario tener en cuenta los elementos claves que permiten realizar la búsqueda, proporcionando un alto grado de precisión ya sean índices, palabras claves o descriptores.

A continuación se muestra una breve explicación del funcionamiento de dicho algoritmo en el método de búsquedas Libro y Folletos y además los ejemplos del código fuente de dicha funcionalidad perteneciente al módulo Gestión de Documentos. Se muestra además la interfaz correspondiente en la que se puede apreciar su utilización.

2.7.1 Descripción de la funcionalidad de búsqueda de libros y folletos

1. El método recibe todos los parámetros (*Ver figura* 14) posibles por los que se puede realizar una búsqueda de cualquier libro o folleto existente en la base de datos.

```
public static function BusquedaLibroFolletos($anno_public, $autorinstitu, $autorpersonal,
$ilustraciones, $material, $mencion, $editorial, $no_acceso, $lugarpublicacion, $nota, $serie,
$titulo,$edicionLF,$ubicacion, $pais,$idioma,$tomoVol,$paginas, $descritor1, $descritor2,
$descritor3, $descritor4, $combinacion1, $combinacion2, $combinacion3, $numero_normalizado,
$cantidad_ejemplares, $fecha_entrada, $especialista) {
```

Figura 14 Declaración de la funcionalidad BusquedaLibroFolletos.

2. Se crea un objeto de tipo **criteria** vacío (*Ver figura* 15) el cual se utiliza para devolver todos los objetos de la clase.

```
$criteria = new Criteria();
```

Figura 15 Creación de un objeto tipo criteria.

3. Se valida que cada uno de los campos no estén vacíos para realizar la consulta (*Ver figura* 16) a la base de datos por ese parámetro.

```
if (!empty($anno_public)) {
    $criteria->add(self::ANO_PUBLICACION, $anno_public);
}
```

Figura 16 Consulta a través del parámetro año de publicación.

4. Se valida que cada uno de los 3 descriptores para el caso que se encuentren distintos de un valor vacío entonces se comprueban el tipo de combinaciones posibles (*Ver figura* 17) en dependencia de lo que se solicite, si la combinación entre descriptor 1 y descriptor 2 se hace por el operador lógico NOT, entonces el resultado serán todos los documentos que tengan que ver con el descriptor 1 y se excluyen los que contienen al 1 y al 2 a la vez. En caso de que la combinación se haga por el operador OR entonces el resultado de la búsqueda arrojaría todos los libros y folletos que hable de 1 y de 2. Por último en caso de que aparezcan relacionados por el operador AND entonces el resultado de la búsqueda se limitaría a mostrar solo los libros y folletos que contengan a los dos descriptores juntos.

```
337         if (!empty($descritor1)) {
338             $criteria->add(DescriptorLibrosYFolletosFpPeer::ID_DESCRIPTOR, $descritor1);
339         }
340         if (!empty($combinacion1)) {
341             if ($combinacion1 == 'NOT') {
342                 if (!empty($descritor2)) {
343                     $criteria->add(DescriptorLibrosYFolletosFpPeer::ID_DESCRIPTOR, $descritor2, Cri
344                 }else
345                     return null;
346                 }
347             if ($combinacion1 == 'AND') {
348                 if (!empty($descritor1) && !empty($descritor2)) {
349                     $criterion = $criteria->getNewCriterion(DescriptorLibrosYFolletosFpPeer::ID_DES
350                     $criteria->addAnd($criterion);
351                 }else
352                     return null;
353                 }
354             if ($combinacion1 == 'OR') {
355                 if (!empty($descritor1) && !empty($descritor2)) {
356
357                     $criteria->addOr(DescriptorLibrosYFolletosFpPeer::ID_DESCRIPTOR, $descritor2);
358                 }else
359                     return null;
360                 }
361         }
```

Figura 17 Ejemplo del uso de los operadores booleanos.

5. Por último se refleja una particularidad en la que a través de una consulta que lleva como parámetro la llave primaria de la tabla Libros y Folletos se retorna el resultado de la

búsqueda (*Ver figura* 18) por cada índice que haya sido seleccionado a la hora de buscar, además se valida que si todos los elementos son vacíos pues la respuesta en nula.

```
    $criteria->setDistinct();
    $criteria->addJoin(DescriptorLibrosYFolletosFpPeer::NUM_ACCESO_LIBROS_FOLLETOS_FP, self::NUM_ACCESO_LIBROS_FOLLETOS_FP);
    $librofolletos = self::doSelect($criteria);
    if (!empty($librofolletos))
        return $librofolletos;
    return null;
}
```

Figura 18 Código que retorna el resultado de la búsqueda.

El resultado de una ecuación de búsqueda booleana es un conjunto de documentos relevantes en el caso de que los haya. Estos son seleccionados siguiendo la lógica booleana, la cual plantea que el documento es verdadero cuando contiene al menos un término de la pregunta (si el operador es un **OR**), cuando contiene todos los términos de la pregunta (si el operador es un **AND**) o cuando no contiene ninguno de los términos de la pregunta (cuando es un **NOT**).

2.7.2 Opciones de búsqueda

En general, las opciones de búsqueda en estos módulos son las siguientes:

- **Búsqueda directa**: Se teclea directamente una o varias palabras en el espacio reservado para ello en la interfaz.
- **Búsqueda a través de índices**: En lugar de teclear un término, el usuario hace su consulta a través de índices o listas alfabéticas con todas las entradas ya sea de uno o diferentes campos concretos.
- **Búsqueda a través de códigos**: En algunos campos la búsqueda no se realiza a través de texto sino de código alfanumérico o numérico.

2.8 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje de programación utilizado y cómo dependen los componentes unos de otros (James

Rumbaugh, 2000). Como parte del Modelo de Implementación se encuentran los Diagramas de Componentes y los Diagramas de Despliegue.

2.8.1 Diagramas de componentes

Los diagramas de componentes describen los elementos físicos de un sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binarios y ejecutables. Los componentes representan todos los tipos de elementos software que entran en la fabricación de la aplicación informática. Pueden ser simples archivos, paquetes, bibliotecas cargadas dinámicamente, etc.

A continuación se enuncian las ventajas que brinda la utilización de diagramas de componentes en el desarrollo de los módulos:

- Muestra como el sistema está dividido en componentes y las dependencias entre ellos.
- Proveen una vista arquitectónica de alto nivel del sistema.
- Ayuda a los desarrolladores a visualizar el camino de la implementación.

En el siguiente diagrama (*Ver figura* 19) se representan los componentes de los módulos Gestión de Boletas y Gestión de Documentos de modo general, donde se evidencian las relaciones entre los componentes, desde el acceso a través del controlador frontal, la clase Actions.php, cada una de las interfaces de la aplicación y las clases del negocio. Todos estos componentes aparecen agrupados en paquetes que representan las capas del patrón de arquitectura Modelo-Vista-Controlador, así como otro paquete de configuraciones que representa cada uno de los archivos configurables en el marco de trabajo para el desarrollo de los módulos del sistema integral. El último estereotipo representa a la base de datos del sistema que se relaciona directamente con clases de acceso a datos y las de abstracción de datos que en conjunto con el ORM Propel, conforman la capa del modelo. La capa del controlador está constituida por el controlador frontal y las clases actions y la capa de la vista contiene todas las plantillas que se relacionan con la página principal a través de una clase interfaz llamada Viewport.js

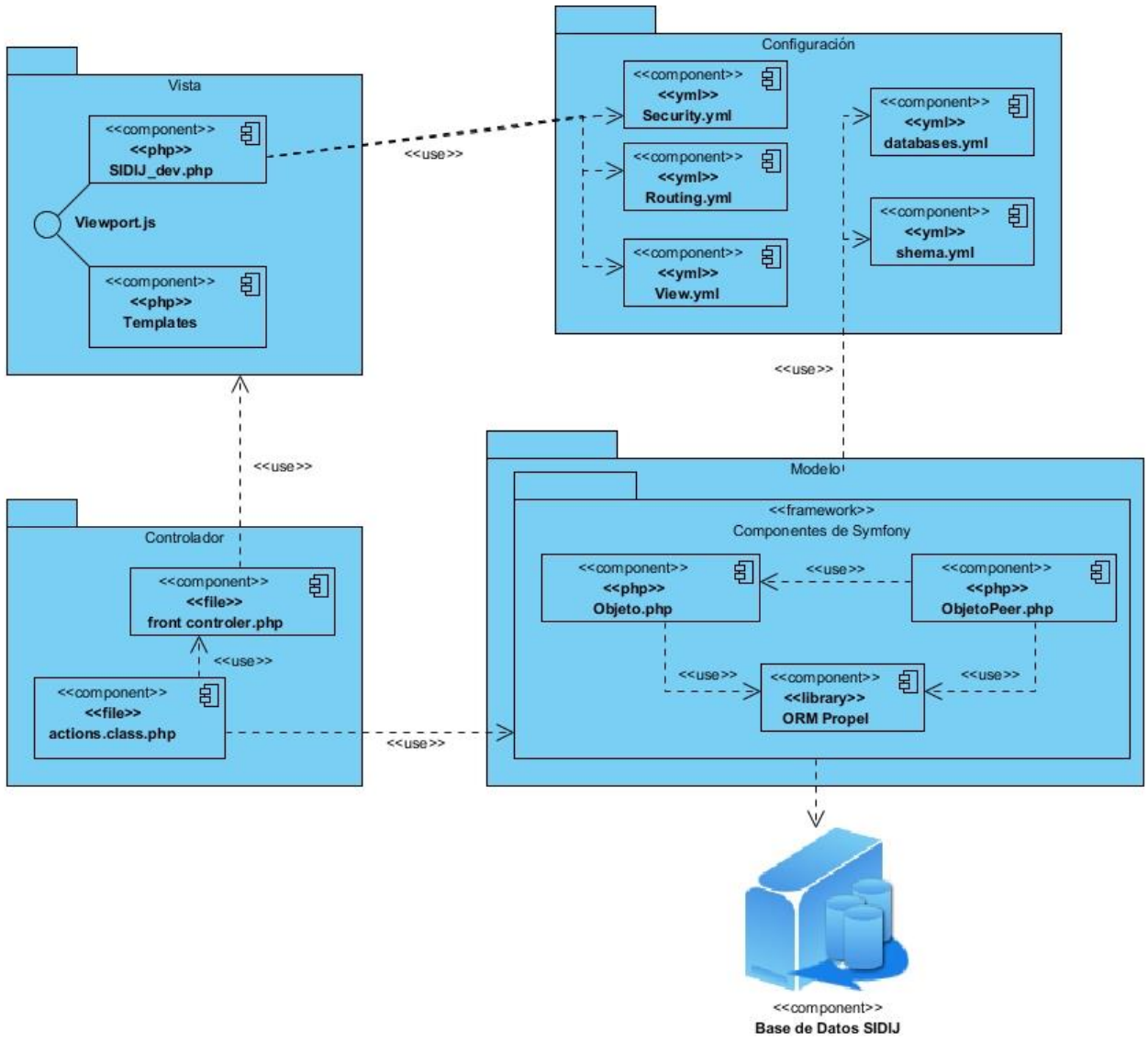

Figura 19 Diagrama de componentes de los módulos.

A continuación se muestra el diagrama de componentes (*Ver figura* 20) para el caso de las búsquedas referenciales en los módulos, donde aparecen representadas cada una de las clases interfaz como componentes que se relacionan a través del componente de interfaz Viewport.js que a su vez tiene una relación estrecha con la librería ExtJS y su uso garantiza la aplicación de los estilos a cada una de las plantillas de la presentación. Todos estos elementos aparecen agrupados en un paquete que representa la vista y a la vez este se comunica con la capa controladora que contiene el controlador frontal y la clase **Actions.php**. Esta es la encargada de controlar las clases del modelo para cada módulo representado mediante componentes que interactúan con el ORM, el cual se relaciona directamente con la base de datos del SIDIJ.

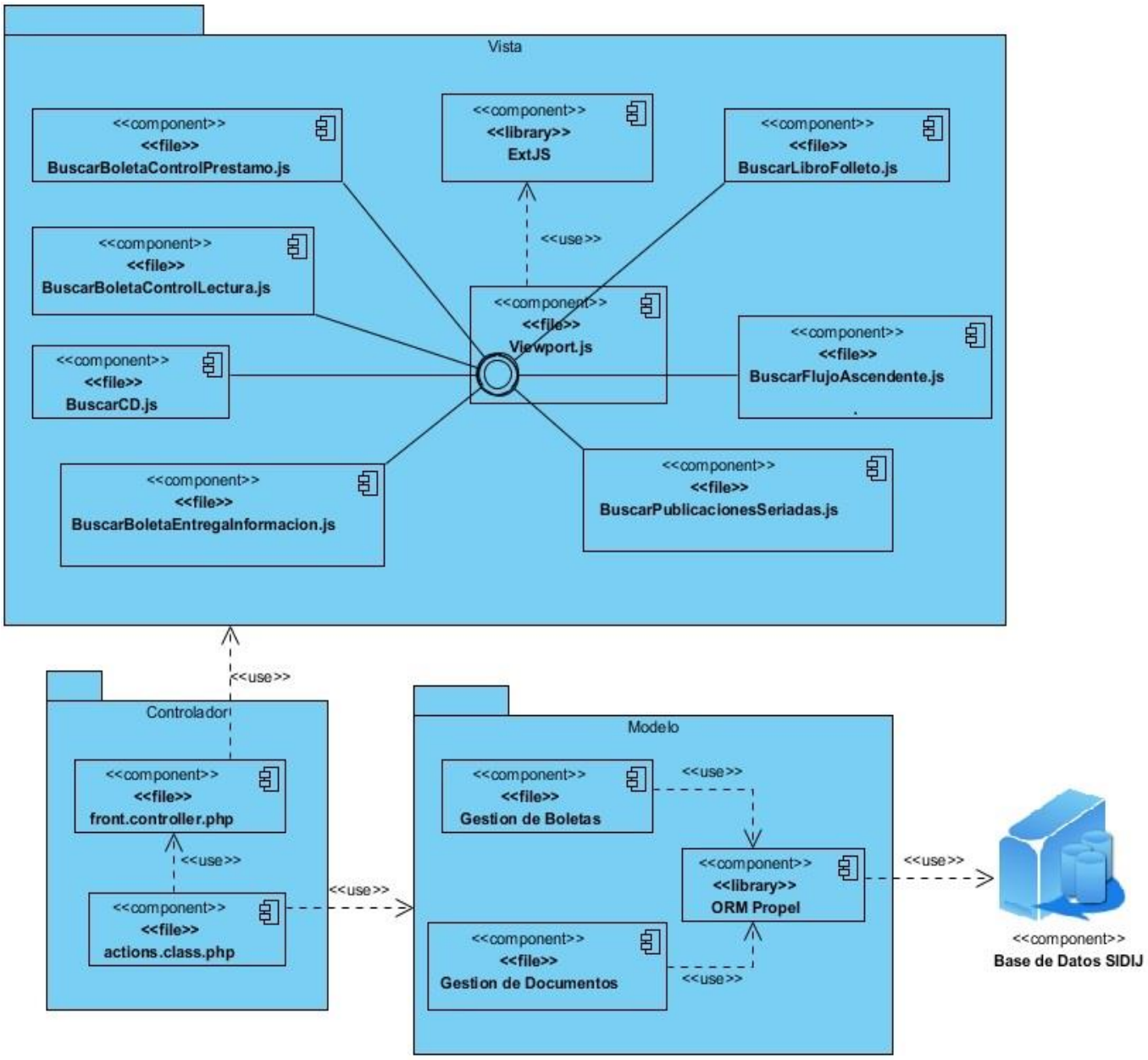

Figura 20 Diagrama de componentes para las búsquedas en los módulos.

2.8.2 Diagrama de despliegue

La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Los estereotipos permiten precisar la naturaleza del equipo.Algunas de sus características son:

- Cada nodo[15] representa un recurso de cómputo, normalmente un procesador o un dispositivo de hardware similar.
- Los nodos poseen relaciones que representan medios de comunicación entre ellos, tales como Internet, intranet, bus y similares.
- El modelo de despliegue puede describir diferentes configuraciones de red, incluidas las configuraciones para pruebas y para simulación. (James Rumbaugh, 2000)

El siguiente diagrama de despliegue (*Ver figura* 21) representa cada uno de los elementos de los que depende la arquitectura de funcionamiento del sistema. Se muestran 4 nodos que representan una PC cliente, una impresora, un servidor web y un servidor de bases de datos, asi como sus respectivos conectores. En las notas aparecen explicadas cada una de estas conexiones con detalles.

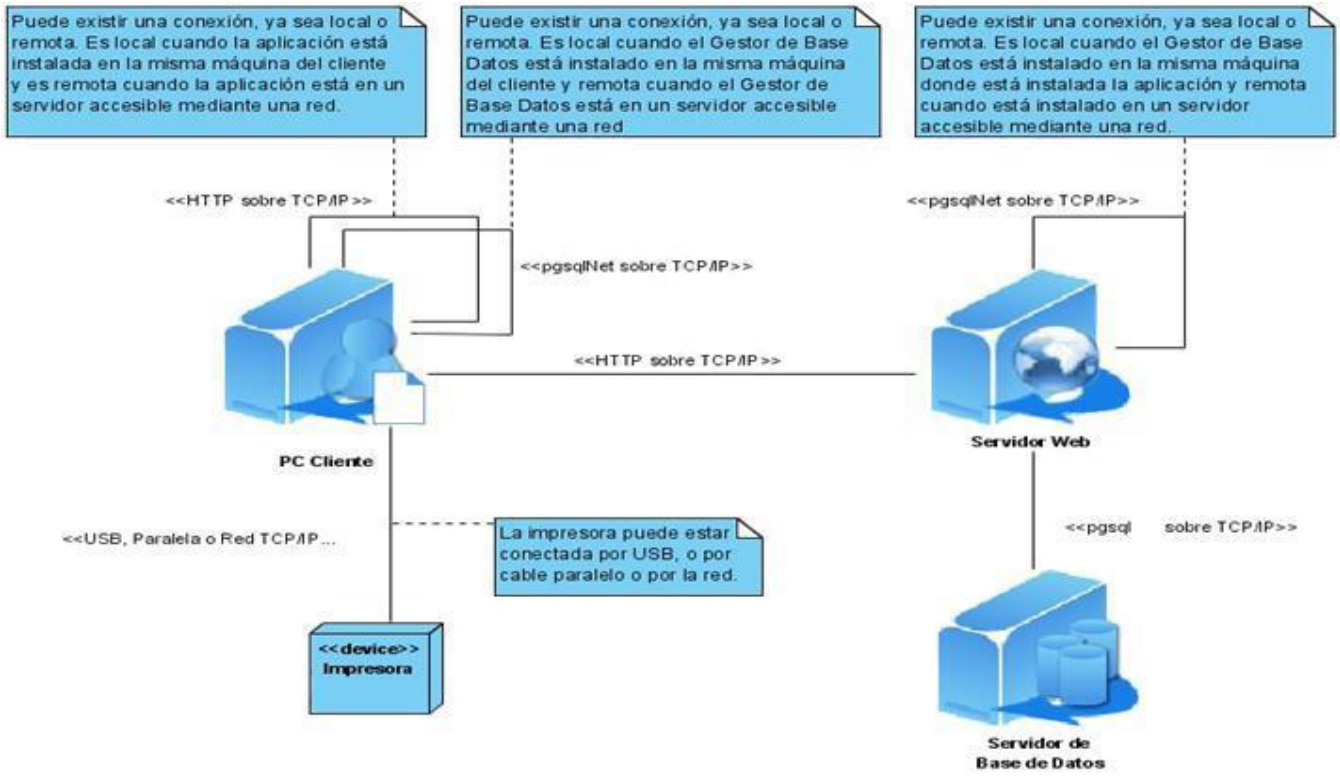

Figura 21 Diagrama de despliegue del SIDIJ.

2.9 Estándares de codificación utilizados

Los estándares de codificación son un conjunto de directrices, normas y reglamentos enfocados a la especificación de cómo debe escribirse el código fuente de la aplicación. Estos incluyen pautas

[15] Ver Glosario de términos.

sobre la nomenclatura de las variables, clases y paquetes; la correcta identación del código, como escribir estructuras de control e incluso las formas de poner los comentarios, en sí todo lo referente a la generación del código.

La correcta elaboración y utilización de los mismos permite que todos los desarrolladores implementen siguiendo las mismas pautas y así puedan entender el código del resto como si estuvieran mirando el de ellos, así la aplicación será más legible, uniforme y fácil de mantener.

Symfony tiene definido en su página principal (http://www.symfony-project.org/) sus propios estándares de codificación, pero en la implementación de los módulos del SIDIJ se utilizaron algunas modificaciones como:

Identación

- En el contenido siempre se identará sin usar las tabulaciones en el código. La sangría se hace por pasos de 4 espacios.

  Ejemplo:

```
public function executeInsetarDescriptor() {
    $nombre = $this->getRequestParameter('nombredescriptor');
```

Clases

- El nombre de las clases controladoras comenzará con letra inicial minúscula basada en el estándar **lowerCamelCase** que dicta que para un nombre compuesto por varias palabras comenzará con minúscula pero todas las palabras internas que lo componen comienzan con mayúscula.

  Ejemplo:

```
class loginActions extends sfActions{
 }
```

- Las declaraciones de clases de abstracción de datos deben contener el sufijo "**Peer**" al final de las mismas y deben comenzar con letra mayúscula.

  Por ejemplo: **public class CDPeer extends BaseCDPeer{ }**

Nomenclatura General

- No poner espacios después de un paréntesis de apertura y antes de un cierre.

  Por ejemplo:

```
return $this->renderText(json_encode($json));// correcto
return $this->renderText( json_encode($json) );// incorrecto
```

- En todo momento se utilizarán nombres que sean claros, concretos y libres de ambigüedades. Ejemplo: "**fechaEntrega**" y no solamente "**fecha**",
- El nombre de todas las variables y métodos comenzarán con letra minúscula y si este está compuesto por varias palabras se utilizará el estilo de escritura **"lowerCamelCase"**.
  Ejemplo: **$idLibroFolleto, $numeroAcceso**

Rutinas y Métodos

- Un buen nombre para una función o método es aquel que describe todo lo que la rutina hace.
- Los nombres de los métodos comienzan con la palabra "**execute"**, seguido del objeto al que afecta. Por ejemplo: **static function executeObtenerListadoDescriptores() { }**.

Nombre de variables

- Los parámetros van siempre en minúsculas.
  Por ejemplo: **$usser = $this->getRequestParameter('usuario');**

- Se procurará evitar dar nombres sin sentido a variables temporales.

Por ejemplo: **$temp, $i, $tmp**.

De esta manera quedan descritos los estándares de codificación utilizados en el desarrollo de los módulos del SIDIJ.

2.10 Descripción de clases y operaciones

Documentar el código de una aplicación es añadir suficiente información como para explicar lo que hace detalladamente, de forma que no sólo los ordenadores sepan qué hacer, sino que además los humanos entiendan qué están haciendo y por qué. La descripción de las clases y sus operaciones constituye una de las tareas dentro de la documentación del código, además constituye una necesidad que sólo se aprecia en su debida magnitud cuando hay errores que reparar o hay que extender el programa con nuevas capacidades o adaptarlo a un nuevo escenario.

Por una u otra razón, todo software que tenga éxito será modificado en el futuro, bien por el programador original, bien por otro programador que le sustituya. Pensando en esta revisión de código es por lo que es importante que el programa se entienda: para poder repararlo y modificarlo.

A continuación se pone un ejemplo de cómo se describen las clases y sus operaciones en el documento Modelo del Diseño 1.2 (Revisar artefactos del proyecto), para esto se tuvo en cuenta el módulo "Gestión de Boletas".

2.10.1 Clases controladoras

En este ejemplo (*Ver tabla* 2) se menciona y describen los métodos usados en la clase "**actions.class"** para lograr gestionar satisfactoriamente una boleta de entrega de información.

Gestionar Boleta Entrega de Información	
Tipo de clase: controladora	
Nombre: actions.class	
Nombre	**Descripción**
executeComboxTecnicos	Se obtienen todos los técnicos mediante la invocación del método UsuarioFuncionPeer::getFuncionTecnicos () y se envían a la vista.
executeComboxTSP	Se obtienen todos los tribunales superiores populares mediante la invocación del método TipoProcendenciaPeer::getTSP () y se envían a la vista.
executeComboxTPP	Se obtienen todos los tribunales provinciales populares mediante la invocación del método TipoProcendenciaPeer::getTPP () y se envían a la vista.
executeComboxTMP	Se obtienen todos los tribunales municipales populares mediante la invocación del método TipoProcendenciaPeer::getTMP () y se envían a la vista.
executeGetBoletaEntrega	Crea una instancia de la clase BoletaEntregaInformación y la envía hacia la vista.
executeInsertarboletaEntrega	En este método se obtienen todos los parámetros que vienen desde la vista y se inserta una nueva Boleta de entrega de información mediante la llamada al método: BoletaEntregaInformaciónPeer::insertarBoletaEntrega($parametros).

executeModificarBoletaEntrega	Crea una instancia de la clase BoletaEntregaInformaciónPeer y a partir de los datos que se modifiquen, se hace una llamada al método ModificarBoletaEntrega enviando como parámetros la instancia modificada.

Tabla 2 Descripción de la Clase Controladora para Gestionar Boleta Entrega de Información.

2.10.2 Clases del modelo

Las clases entidades pertenecientes al modelo, se encargan de modelar la información del sistema y el comportamiento asociado a una información.

En este ejemplo se mencionan las clases del modelo que tienen relación con la clase "**Descriptor"** (*Ver tabla* 3) y que por tanto se hizo necesario el trabajo con ellas para obtener la información requerida.

Tipo de Clase: Entidad
Nombre
tb_cd
tb_libro_folleto
tb_flujo_ascendente
tb_publicaciones_seriadas

Tabla 3 Descripción de la clase entidad Descriptor.

2.10.3 Clases de la interfaz

Estas clases son las que permiten la interacción del usuario con el sistema mediante un formulario donde se le muestran los datos que deben ser introducidos por el mismo para realizar las diferentes acciones que serán manejadas en el controlador. En la siguiente tabla (*Ver tabla* 4) se describe el funcionamiento de la clase interfaz "InsertarLibrosYFolletosFP".

InsertarLibrosYFolletosFp	
Tipo de clase: Interfaz	
Nombre del fichero	**Descripción**
InsertarLibrosYFolletosFp.js	Crea toda la estructura visual mediante la utilización del ExtJS. Esta vista permite la entrada de los datos necesarios para insertar un libro o un folleto. Se realizan validaciones de campos y se muestran mensajes de error y confirmación al hacer alguna acción que lo requiera.

Tabla 4 Descripción de la clase interfaz InsertarLibrosYFolletosFP.

2.11 Tratamiento de errores y validación en la entrada de datos al sistema

Para lograr el correcto funcionamiento del software se debe tomar en cuenta un adecuado tratamiento de errores, ejemplo de ello es cuando el sistema captura todas aquellas excepciones que son lanzadas y se les realiza un tratamiento para que no deje de funcionar. Esta es la vía mediante la cual se da solución a las problemáticas de los lanzamientos de excepciones que surgen según las peticiones del usuario, al tomar los tipos de errores lanzados y mostrar de manera visible los mensajes de confirmación al usuario (*Ver figura* 22), esto le informará de lo que está erróneo y como corregirlo.

Figura 22 Mensaje de aviso.

A través de la interfaz, el usuario asumirá un papel en la introducción de la información, para lo cual cuenta con cuadros de opción, menú de selección y textos autocompletables lo cual facilitará la

entrada de datos. Si existe un error en los datos le aparecerá un mensaje en pantalla al usuario indicándole el error. Todas estas validaciones en las interfaces de los módulos se pueden llevar a cabo utilizando:

- Campos de tipo fecha
- Listas no editables
- Campos de selección

Estos elementos que brinda la librería ExtJS se utilizan para evitar considerablemente el margen de error en la entrada de datos al sistema. ExtJS contiene una serie de expresiones regulares que se utilizan para validar campos que requieran de la entrada de solo datos numéricos, de solo letras, de fecha y hora específicamente, de correos electrónico (*Ver figura* 23), así como para validar la entrada correcta de un año, un carnet de identidad y otros elementos que requieran una X cantidad de caracteres. Permite ponerle un límite de hasta 255 caracteres en un campo de texto para que no ocurran errores de sobrecargas de campos, lo mismo sucede con la entrada de valores numéricos y de caracteres extraños.

Figura 23 Mensaje de error en la entrada de un correo electrónico.

En general el tratamiento de errores en cualquier aplicación que utiliza la librería ExtJS puede asegurar buenos resultados, pues cuenta con elementos que ayudan a reducir la ocurrencia de los mismos en la entrada de información por parte del usuario. Estos representan una ventaja para el programador debido a que solo tiene que hacer uso de sus expresiones regulares, las cuales vienen detalladas en la ayuda de ExtJS. (Frederick, 2009)

Para el tratamiento de excepciones en la capa del controlador, los desarrolladores utilizan las clásicas sentencias de validación **try** y **catch**. Para solicitar la ejecución de cualquier funcionalidad, esta se pone dentro de la sentencia **try**. En caso que la función contenga un error en su ejecución, este es capturado en la sentencia **catch**, la cual envía una respuesta hacia la interfaz, en la mayoría de los casos un mensaje describiendo el error cometido en la entrada de datos.

Por ejemplo:

```
try { if (BoletaSalaLecturaPeer::modificarBoletaSalaLectura($nombre, $procedencia, $fecha, $categoríausuario, $nombre_técnico,$externo, $idboleta ) == 1) {
    return $this->renderText("{success: true}") } else
    return $this->renderText("{success: false}");
}
catch (Exception $e) {
  return new Response( "{success: false, message: 'Se ha cometido un error al entrarlos datos!'}" );
 }
```

2.12 Conclusiones parciales

Con la realización de este capítulo, se evidenció, que el diseño propuesto para la implementación de los módulos posibilitó la realización satisfactoria de los mismos, producto de que el diseño tuvo en cuenta las peculiaridades del framework Symfony, con el objetivo de lograr un mayor aprovechamiento de estas para el funcionamiento de los módulos. Se plasmaron además los estándares de codificación que se utilizaron para la implementación del SIDIJ para obtener un código más legible y entendible, ayudando a la comunicación entre desarrolladores. Se obtuvo una propuesta de solución perceptible que permitirá el despliegue del sistema informático en el CENDIJ cumpliéndose de ésta forma el objetivo general.

CAPÍTULO 3

VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

Introducción

"Una de las características típicas del desarrollo de software basado en el ciclo de vida, es la realización de controles periódicos, normalmente coincidiendo con los hitos del proyecto o la terminación de documentos. Estos controles pretenden hacer una evaluación de la calidad de los productos generados (especificación de requisitos, documentos de diseño) para poder detectar posibles defectos de manera inmediata. Sin embargo, todo sistema o aplicación, independientemente de estas revisiones, debe ser probado mediante su ejecución controlada antes de ser entregado al cliente. Estas ejecuciones o ensayos de funcionamiento, posteriores a la terminación del código del software, se denominan habitualmente pruebas". (S. Pressman, 2005)

Con esta cita de Pressman se introduce el presente capítulo en el que se evidencian los resultados de las pruebas y validaciones realizadas a los módulos, que permitieron garantizar el correcto funcionamiento y totalidad de las funcionalidades de los mismos.

3.1 Pruebas de software

El proceso de pruebas constituye un elemento crítico para determinar el status de la calidad de un producto software. En esta etapa se diseñan una serie de casos de prueba que intentan encontrar errores en el software desarrollado. Para la validación de los resultados que se obtendrán después de haber desarrollado los módulos, se ha decidido realizar las pruebas de Caja Blanca y Caja Negra (*Ver figura* 24), asegurando así la completitud de los requisitos funcionales, parte fundamental en el desarrollo de un software.

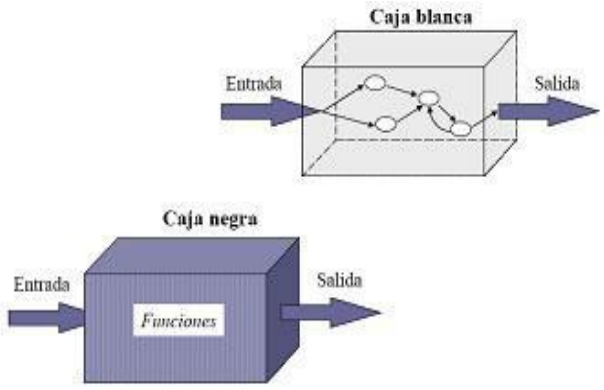

Figura 24 Técnicas de pruebas utilizadas.

3.1.1 Objetivos de las pruebas

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces (Pressman, 2005 pág. 304).

3.2 Pruebas de Caja Blanca

Las pruebas de Caja Blanca son las que se realizan sobre las funciones internas de un módulo, 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, se 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.
- Ejerciten las estructuras internas de datos para asegurar su validez.

Algunas técnicas de prueba de Caja Blanca son:

- Prueba de condición: Ejercita las condiciones lógicas 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.
- La prueba del camino básico: Permite 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 idea es realizar un grupo de casos de prueba que garanticen que se pueda probar por lo menos una vez cada una de las sentencias del programa (Pressman, 2005 p. 286).

En la validación de la solución de los módulos, se realizó la prueba del camino básico a varias funcionalidades, la cual permite definir los diferentes caminos independientes de la función y probar su funcionamiento al menos una vez. A continuación se muestra el proceso desarrollado al método **"obtenerDescriptorCD ()"** de la clase del modelo **"DescriptorCDPeer.php"**.

Para ello se comienza analizando el código y enumerando las instrucciones. (*Ver figura* 25)

```
14    public static function obtenerDescriptoresCD($nroacceso){
15
16        $descriptores = array();//1
17        $criteria = new Criteria();//1
18        $criteria->addSelectColumn(DescriptorPeer::ID_DESCRIPTOR);//1
19        $criteria->addSelectColumn(DescriptorPeer::DESCRIPCION);  //1
20        $criteria->add(self::NUMERO_ACCESO_CD, $nroacceso);//1
21        $criteria->addJoin(self::ID_DESCRIPTOR, DescriptorPeer::ID_DESCRIPTOR);//1
22        $resultado = self::doSelectStmt($criteria);//1
23
24        WHile ($stmt = $resultado->fetch(PDO::FETCH_NUM))//2
25        {
26            $arr['id_descriptor'] = $stmt[0];//3
27            $arr['nombre'] = $stmt[1];//3
28            $arrs[] = $arr;//3
29        }
30
31        if(isset($arrs))//4
32            return $arrs;//5
33        return null;//6
34    }//7
```

Figura 25 Función a la que se le aplica el método del Camino Básico.

Se debe tener en cuenta la notación para las instrucciones (*Ver figura* 26) a la hora de construir el grafo.

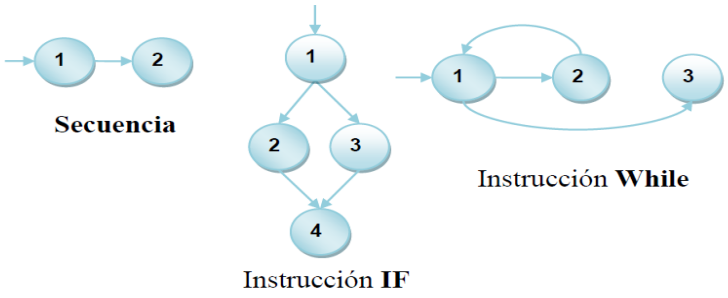

Figura 26 Notación para las instrucciones de secuencia, if y while.

Seguidamente es necesario representar el grafo de flujo (o grafo del programa), que va a representar el flujo de control lógico del código anterior, a través de nodos, aristas y regiones. El grafo queda como se muestra a continuación (*Ver figura* 27).

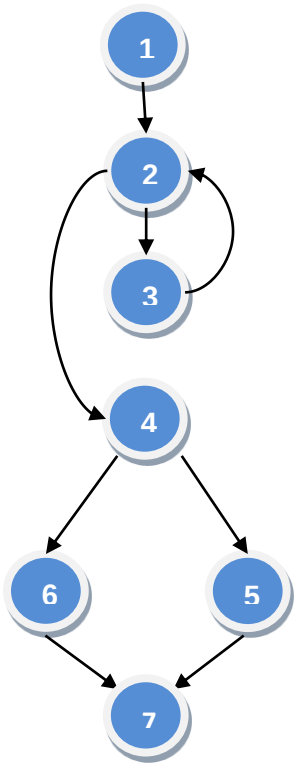

Figura 27 Grafo de flujo del código de la función.

Luego de realizado el grafo es necesario conocer la cantidad de caminos independientes que se deben buscar para probar; y para esto se calcula la complejidad ciclomática, que se puede calcular de tres formas:

1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática. Por lo que V(G) = **3**
2. V (G) = (A – N) + 2 donde "**A"** es el número de aristas del grafo de flujo y "**N**" es el número de nodos del mismo.
   V (G) = 8 – 7 +2 = **3**
3. V (G) =P+ 1 donde "**P**" es el número de nodos predicado contenidos en el grafo.
   Los nodos 2 y 3 son nodos predicados. V (G) = 2+1 = **3**

En cualquiera de las tres formas anteriores la complejidad ciclomática es **3**, por lo que la cantidad de caminos básicos (*Ver tabla* 5) que puede tomar el algoritmo durante su ejecución es **3** y quedan definidos como muestra la siguiente tabla:

Número	Caminos básicos
1	1-2-4-6-7
2	1-2-4-5-7
3	1-2-3-7

Tabla 5 Caminos básicos.

Después de haber extraído los caminos básicos del flujo se procede a ejecutar los casos de pruebas para este procedimiento, es necesario realizar al menos uno por cada camino básico. Para realizarlos es preciso cumplir con las siguientes exigencias:

Descripción: Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.

Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.

Entrada: Se muestran los parámetros que entran al procedimiento.

Resultados Esperados: Se expone el resultado que se espera devuelva el procedimiento.

Una vez determinados cada uno de estos parámetros, se procede a realizar la prueba unitaria de flujo básico haciendo uso de la librería para pruebas que se integra en Symfony llamada Lime, con la cual se verificará la calidad del código fuente.

3.2.1 Lime: librería para pruebas unitarias en Symfony

En PHP existen muchas herramientas para llevar a cabo un conjunto pruebas unitarias y Symfony a su vez incluye una propia librería llamada Lime. La misma proporciona soporte para la realización de pruebas unitarias, alguna de las ventajas que le brinda al desarrollador y al probador son:

- Ejecuta los archivos de prueba en un entorno independiente para evitar interferencias entre las diferentes pruebas.
- Las pruebas son fáciles de leer y sus resultados también lo son.
- Utiliza colores de forma clara e inteligente para distinguir la información más importante.
- Está escrita con PHP, es muy rápida a la hora de dar una respuesta y está bien diseñada internamente.
- Consta únicamente de un archivo, llamado Lime.php, y no tiene ninguna dependencia. (Potencier, et al., 2010)

Se utiliza esta librería para comparar los resultados obtenidos con los esperados por el desarrollador. A continuación se muestran tres escenarios que representan los casos de prueba que forzarán la ejecución de cada camino del conjunto básico, quedando de la siguiente forma:

Descripción: Se debe comprobar si coincide la relación entre un ID y un descriptor para obtener una correcta respuesta en la llamada a la función "**obtenerDescriptorCD()**". Los datos de entrada cumplirán con los siguientes requisitos: El id del descriptor es un número entero y el nombre una cadena de letras.

Esta descripción es común para todos los casos de prueba que se realizarán.

Caso de prueba para el camino básico # 1.

Condición de ejecución: Para esta prueba es necesario que la variable descriptor sea igual 2 y el nombre igual a ABANDONO DE DESTINOO.

Entrada: Las variables reciben los valores $descriptor= 2, $nombre = ABANDONO DE DESTINOO.

Resultados esperados: Se espera que se muestre un mensaje de error en la comprobación debido a la entrada incorrecta en el nombre.

El resultado obtenido fue correcto.

Caso de prueba para el camino básico # 2.

Condición de ejecución: Para esta prueba es necesario que la variable descriptor sea igual 3 y el nombre igual a ABANDONO DE DESTINO.

Entrada: Las variables reciben los valores $descriptor=3, $nombre = ABANDONO DE DESTINO.

Resultados esperados: Se espera que se muestre un mensaje de error en la comprobación debido a la entrada incorrecta en el valor del descriptor.

El resultado obtenido fue correcto.

Caso de prueba para el camino básico # 3.

Condición de ejecución: Para esta prueba es necesario que la variable descriptor sea igual 2 y el nombre igual a ABANDONO DE DESTINO.

Entrada: Las variables reciben los valores $descriptor = 2, $nombre = ABANDONO DE DESTINO.

Resultados esperados: Teniendo en cuenta la condición de ejecución se espera que se muestre un mensaje de éxito en la comprobación.

El resultado obtenido fue correcto.

3.3 Pruebas de Caja Negra

Las pruebas de Caja Negra también conocidas con sus varios nombres como pruebas funcionales, pruebas de caja opaca o pruebas de entrada/salida se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa y su objetivo principal es la detección de errores mediante el uso de casos de prueba. Las pruebas de Caja Negra pretenden encontrar estos tipos de errores:

- Funciones incorrectas o ausentes.
- 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.

3.3.1 Ejecución de las pruebas de Caja Negra

Para aplicar las Pruebas de Caja Negra se toma de ejemplo la descripción del caso de prueba para el caso de uso Agregar Boleta de Control de Lectura y se realiza el flujo varias veces recreando posibles escenarios.

Diseño del Caso de prueba # 1: Caso de Uso Agregar Boleta Control de Lectura. (Primera iteración)

Descripción General: El Caso de Uso inicia cuando se necesita agregar al sistema una nueva boleta de control de lectura.

Condiciones de Ejecución:

- El usuario debe estar autenticado.
- El usuario debe escoger “Agregar” después que se visualice la lista de boletas de control de, lectura e introducir los campos requeridos.
- El usuario debe pulsar el botón “Insertar”.

En la siguiente tabla (*Ver tabla* 6) se describen los escenarios correspondientes a cada sección. Un escenario no es más que un flujo de los eventos que ocurran. En este caso son descritos, un total 11 escenarios posibles para las de las 3 secciones pertenecientes al caso de prueba en cuestión.

Nombre de la Sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1. Agregar boleta Control de Lectura	EC 1.1 Agregar una nueva boleta de control de lectura correctamente	En este escenario se agrega una nueva boleta en el sistema, donde todos los campos son introducidos correctamente.
	EC 1.2 Seleccionar Opción Cancelar.	Mediante este escenario se limpian los campos de la ventana.
	EC 1.3 Agregar una nueva boleta con datos incorrectos	Mediante este escenario el sistema muestra un mensaje informando que los datos del formulario no son válidos.
	EC 1.4 Agregar una nueva boleta con datos incompletos	Mediante este escenario el sistema muestra un mensaje de información indicando que los datos introducidos no son válidos.
SC 2. Buscar boleta Control de Lectura	EC 3.1 Buscar una boleta de control de lectura	Se busca una boleta control de lectura ya existente en el sistema

	EC 3.2 Cancelar la búsqueda	Mediante este escenario se cancela la búsqueda de una boleta control de lectura ya existente
	EC 3.3 Buscar una boleta de control de lectura con datos incompletos	No se muestra el resultado de la búsqueda de la boleta en el sistema debido a la existencia de datos incompletos.
SC 3. Modificar Boleta Control de Lectura.	EC 2.1 Modificar una boleta de control de lectura correctamente	Mediante este escenario se modifica una boleta de control de lectura ya existente en el sistema.
	EC 2.2. Seleccionar opción Cancelar	Mediante este escenario se cancela la modificación de una boleta de control de lectura ya existente
	EC 2.3 Modificar una boleta de control de lectura con datos incorrectos	Mediante este escenario el sistema muestra un mensaje informando que los datos del formulario no son válidos.
	EC 2.4 Modificar una boleta de control de lectura con datos incompletos	Mediante este escenario no se modifica la boleta en el sistema debido a la existencia de datos incompletos.

Tabla 6 Secciones a probar en el caso de uso Agregar Boleta Control de Lectura.

A continuación se listan todos los campos asociados a un escenario determinado, describiendo por cada uno de ellos las características que lo definan (*Ver tabla* 7).

No	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Nombre y apellidos	Campo de texto	no	El campo solo permite introducir números.
2	Fecha	Campo de selección	no	El campo sólo permite introducir números respetando el formato AAAA/MM/DD.
3	Categoría	Campo de selección	no	El campo sólo permite seleccionar una categoría de las que aparecen en la lista.
4	Procedencia	Campo de texto	no	El campo sólo permite introducir

				letras.
9	Nombre del técnico	Lista desplegable	no	El campo sólo permite los valores que están en la lista desplegable.

Tabla 7 Descripción de variables SC 1 Agregar Boleta Control de Lectura.

Para documentar los casos de prueba claramente, se utiliza un formato de matriz como en la siguiente tabla (*Ver tabla* 8). La primera columna contiene la identificación del caso de prueba, la segunda tiene una breve descripción del mismo, incluyendo el escenario que es probado y el resto de las columnas menos la última, contienen los elementos de datos que serán utilizados para poner las pruebas en ejecución. La última columna contiene una descripción de la salida prevista del caso de prueba.

Las celdas de la tabla contienen **V, I,** o **N/A**. El valor V indica válido, el I, inválido y N/A indica que no es necesario proporcionar un valor del dato en este caso, debido a que es irrelevante.

Escenario	Respuesta esperada	Nombre y apellidos	Fecha	Categoría	Procedencia	Nombre del técnico	Resultado de la prueba
EC 1.1 Agregar una nueva boleta de control de lectura correctamente	El sistema inserta satisfactoriamente una boleta y realiza la siguiente notificación: Los datos fueron guardados correctamente	V	V	V	V	V	Satisfactorio
EC 1.2 Seleccionar Opción Cancelar	Se limpian los campos de la ventana.	N/A	N/A	N/A	N/A	N/A	Satisfactorio

EC 1.3 Agregar una nueva boleta con datos incorrectos (*)	El sistema muestra un mensaje informando que los elementos del formulario no son válidos.	I	V	V	I	V	Satisfactorio
EC 1.4 Agregar una nueva boleta con datos incompletos(*)	El sistema muestra un mensaje indicando la existencia de datos incompletos	I	V	I	I	V	Satisfactorio

Tabla 8 Matriz de Datos SC 1. Agregar Boleta Control de Lectura.

Nota: El símbolo de (*) significa todas las posibles variantes para esos campos.

En una primera revisión de estas pruebas arrojaron como resultado: 4 errores, 2 consistentes y otros 2 que no fueron significativos. La tabla que se muestra a continuación contiene un registro de las no conformidades encontradas para darles solución en una segunda iteración. (*Ver Tabla* 9)

Aparecen registrados estos errores consistentes, de los cuales uno resultó en la validación del campo llamado "**Externo**" a la hora de realizar la búsqueda de una boleta Control de Lectura. El otro error detectado fue a la hora de agregar una nueva boleta al sistema que no mostraba bien los resultados.

Elemento	Número	No Conformidad	Aspecto Correspondiente	Etapa de Detección	Significativa	No Significativa	Recomendación	Estado NC	Respuesta del Equipo de Desarrollo
Buscar Boleta Control-Lectura	1	No se puede buscar por el campo Externo.	Buscar Boleta Control Lectura	Primera iteración	x	_	Se recomienda revisar la funcionalidad	Resuelta	Se arregló la búsqueda por Externo
Adicionar Boleta Control-Lectura	2	No se muestra el resultado cuando se agrega una boleta	Adicionar Boleta Control Lectura	Primera iteración	x	_	Se recomienda revisar la funcionalidad de la interfaz	Resuelta	Se arregló la interfaz de Gestión de Boleta Control de Lectura

Tabla 9 Registro de defectos y dificultades detectados.

3.3.2 Análisis de los resultados de las pruebas de Caja Negra

Durante el transcurso de la etapa de pruebas a los módulos se diseñaron un total de 7 casos de prueba que incluyen un total de 21 casos de uso (*Ver figura* 28), agrupados en 7 casos de uso de gestión. Cada uno de los errores detectados fue corregido en una segunda revisión donde todas las pruebas arrojaron como resultado el 100% de éxito, demostrando que las funcionalidades de los casos de uso son correctas.

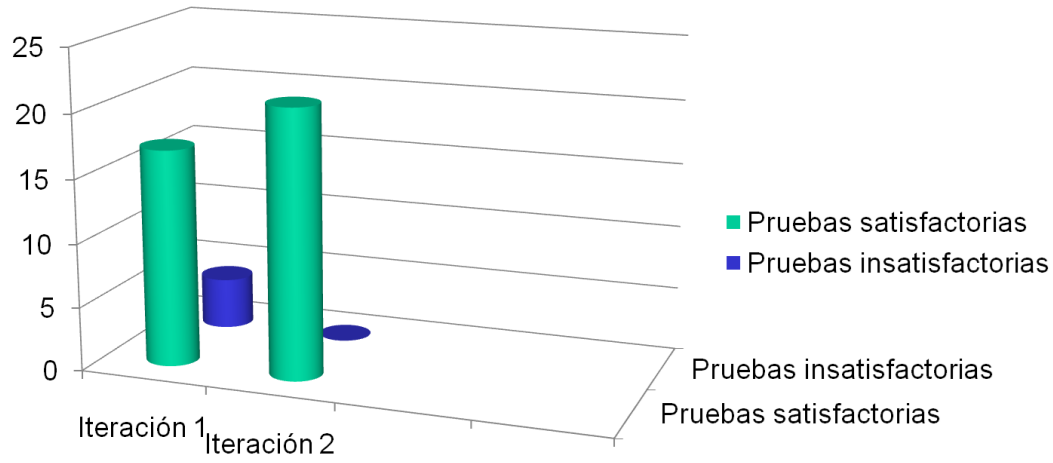

Figura 28 Gráfica de los resultados de la pruebas de Caja Negra.

3.4 Conclusiones parciales

La realización de este capítulo puso de manifiesto la importancia y la necesidad de la realización de diferentes tipos de pruebas de software, pues ayudaron a descubrir errores una vez que se terminó el desarrollo de los módulos. Se aplicaron pruebas de Caja Blanca a las clases del modelo permitiendo evaluar la complejidad ciclomática de sus funcionalidades. El resultado arrojado en el ejemplo abordado, fue de 3 posibles caminos básicos a los que se les hizo un total de 3 casos de pruebas, 2 con valores erróneos y uno correcto mediante los que se confirmó el correcto funcionamiento del código fuente de los módulos. También se hicieron pruebas de Caja Negra, las cuales arrojaron resultados satisfactorios, al encontrar un total de 4 no conformidades entre errores de validación de campos de entrada y errores ortográficos en las interfaces de los módulos, los cuales fueron corregidos eficazmente para lograr la posterior integración de los módulos al sistema integral.

CONCLUSIONES GENERALES

La realización de esta investigación propicia un estudio de las diferentes soluciones informáticas relacionadas con la gestión de documentos y boletas existentes en el mundo, arrojando como resultado que ninguno de sus módulos se ajustan a los requerimientos necesarios para el correcto funcionamiento del SIDIJ necesario en el CENDIJ. Esto que conllevó a la utilización de un modelo de desarrollo sustentado por el framework Symfony para la construcción de los módulos Gestión de Documentos y Gestión de Boletas basándose en la metodología RUP.

La utilización de este modelo anteriormente mencionado, evidenció la completa realización de cada uno de los módulos, logrando alcanzar así el objetivo general propuesto de la presente investigación para contribuir al despliegue del proyecto. Los módulos implementados brindan una alternativa eficiente para el proceso de desarrollo de software, pues el empleo de los mismos posibilita la gestión eficaz y eficiente de todos los documentos y boletas que se manejan a diario en el CENDIJ. Finalmente se realizaron un conjunto de pruebas de Caja Negra y Caja Blanca a la solución propuesta para comprobar la calidad de la solución, las cuales arrojaron resultados satisfactorios.

BIBLIOGRAFÍA

Apache.org. 2012. Apache HTTP Server Project. [En línea] 2012. http://httpd.apache.org/.

Bernie, Dodge. 2011. *Motores de Búsqueda y Álgebra Booleana.* 2011.

Camp de morvedre. 2008. Introducción a los lenguajes web. Programación en PHP y Bases de datos. [En línea] 2008. http://www.iescamp.es/fileadmin/informatica/php_y_mysql.pdf.

Cobas Díaz, Walberto. 2011. Revista Tino. ExtJS (Análisis). [En línea] 2011. http://revista.jovenclub.cu/index.php?option=com_content&task=view&id=634&Itemid=100.

Codina, Luis L. 1995. *Teoría de la recuperación de la información: modelos fundamentales y aplicaciones a la gestión documental.* s.l. : Information World en Español, 1995. págs. 18-22.

Desarrollo orientado a objetos con UML. **Xavier Ferré Grau, María Isabel Sánchez Segura (Facultad de Informática–UPM). 2004.** 2004.

Duarte , Maicol. 2008. Seminario de Programación. NetBeans IDE. [En línea] 2008. http://seminprogramacion.blogspot.com/2008/06/netbeans-ide.html.

Frederick, S., Ramsay, Colin y Blades, Steve 'Cutter'. 2009. *Learning Ext JS.* 2009.

Identi. 2011. ¿Qué es Apache? [En línea] 2011. http://www.identi.es/?topic=37883..

James Rumbaugh, Ivar Jacobson, Grady Booch. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : Addison Wesley, 2000.

Larman, Carig. 1999. *UML y Patrones.Introducción al análisis y diseño orientado a objetos.* s.l. : Prentice Hall, 1999.

León, Eduardo. 2008. Tutorial Visual Paradigm. [En línea] 2008. http://slion2000.blogspot.com/2008/10/tutorial-visual-paradigm.html..

Lex Doctor. 2012. Lex-Doctor. [En línea] 2012. http://www.lex-doctor.com/sobre_lex-doctor.php.

López, Héctor A. Bareiro y Ricardo G. 2007. Comparativa entre Ruby on Rails, Kumbia y Symfony. [En línea] 2007. http://www.sicuma.uma.es/sicuma/independientes/argentina08/Bareiro-Lopez/index.htm.

Martínez Méndez , Francisco Javier . 2004. *RECUPERACIÓN DE INFORMACIÓN:MODELOS, SISTEMAS Y EVALUACIÓN.* 2004.

Molina, Maria Pinto. 2011. BÚSQUEDA Y RECUPERACIÓN DE INFORMACIÓN. [En línea] 13 de 4 de 2011. http://www.mariapinto.es/e-coms/recu_infor.htm.

Olvera Lobo, María Dolores. 1999. *Evaluación de sistemas de recuperación de información: aproximaciones y nuevas tendencias. El profesional de la Información.* Andalucía : s.n., 1999. págs. 4-14. Vol. 8.

Oracle. 2012. Sitio oficial de Netbeans. [En línea] 2012. http://netbeans.org/community/releases/70/.

Palmer, Mª del Carmen Simón. 2012. Truncamiento. [En línea] 2012. http://ble.chadwyck.co.uk/help/noframes/trun.htm.

Pereira, Jorge E. 2006. Aplicaciones Web y ASPs. [En línea] 2006. http://www.mercadeo.com/57_ASPs.htm.

PostgreSQL, E.e.d.d.d. 2008. *Manual del usuario de PostgreSQL.* 2008.

Potencier, Fabien. 2009. *El Tutorial Jobeet.* 2009.

Potencier, Fabien y Zaninotto , Francois . 2010. *La guía definitiva de Symfony.* 2010.

Pressman, Roger S. 2005. *Ingeniería de Software. Un enfoque práctico.* España : Mc Graw Hill, 2005.

Reuters, Thomson. 2012. Infolex. *Aranzadi jurisoft.* [En línea] 2012. http://www.jurisoft.es/ie/infolex.aspx?menu=3.

S. Pressman, Roger. 2005. *Ingeniería de Software. Un enfoque práctico.* España : Mc Graw Hill, 2005.

Silva, Luis Raciel Rodríguez. 2010. Los Sistemas de Gestión de Servicios Legales. Propuesta De Modelación de un Sistema para una Oficina de Asistencia Legal en Cuba. 2010.

The PHP Group. 2012. Sitio oficial de PHP. [En línea] 2012. http://www.php.net/docs.php.

GLOSARIO DE TÉRMINOS

- **IDE (Entornos de desarrollo integrados**): Un entorno de desarrollo integrado, conocido también como IDE por sus siglas en inglés de “Integrated Development Environment”, consiste básicamente en un programa informático que previamente ha sido instalado en la máquina del desarrollador y cuyo principal objetivo es la elaboración de otras aplicaciones, es decir es un entorno de programación que ha sido empaquetado como “Software”, que va a contener un editor de código, un compilador, un depurador y un constructor de interfaz gráfica, que van a facilitar las diferentes tareas necesarias en el desarrollo y mantenimiento de cualquier tipo de aplicación.

- **Módulo**: Cada módulo es una parte del sistema, que se instala y funciona por separado, entrelazándose con otros módulos con los que intercambia información.

- **Framework:** Los frameworks de desarrollo son herramientas construidas con el objetivo de facilitarles a los programadores y diseñadores una mayor organización y estructuración en sus proyectos, estos permiten una estandarización del código y una optimización favorable a cualquier proyecto, debido a que traen un grupo de funcionalidades internas que contribuyen a un desarrollo organizado, estructurado y ágil.

- **Licencia BSD:** La licencia BSD es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution**).**

- **Nodo:** Un nodo es un elemento físico que existe en tiempo de ejecución y que representa un recurso computacional, que en general tiene al menos una memoria y a menudo capacidad de procesamiento.

- **Layout:** Es la ordenación y colocación de todos los elementos que componen una página web, es decir textos, imágenes, tablas y gráficos. También son elementos del layout los colores y el tipo de letra. A la hora de realizar los diseños hay que tener en cuenta que un layout claro permitirá una navegación mucho más fácil.

- **JSON:** Es un formato ligero de intercambio de datos. Está basado en un subconjunto del lenguaje de programación JavaScript. Es un formato de texto que completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl y Python.

- **RAM:** La memoria de acceso aleatorio (en inglés: Random-Access Memory) es la memoria desde donde el procesador recibe las instrucciones y guarda los resultados.

- **Flujo Ascendente:** Investigaciones que no llegan a tener el rótulo de informe científico pero cuyo nivel de profundidad merece que sean tenidas en cuenta. Ejemplo: tesis, actas de congresos, boletines, cuadernos de trabajo, informes técnicos y autobiografías**.**

- **Descriptor:** Es cada uno de los términos o expresiones escogidos entre un conjunto de sinónimos o casi sinónimos para representar generalmente de manera unívoca, un concepto susceptible de aparecer con cierta frecuencia en los documentos con índices, y en las consultas que se realicen.

- **Herramienta CASE**: Las herramientas CASE (Computer Aided Software Engineering: Ingeniería de Software Asistida por Ordenadores) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo del software reduciendo el coste de las mismas en términos de tiempo y dinero.

- **Código abierto**: En inglés Open Source, conlleva como idea más importante la posibilidad de acceder al código fuente, modificarlos y distribuirla como establezca la licencia bajo la que está distribuido.

- **CSS:** Cascade Style Sheet. Conjunto de instrucciones HTML que definen la apariencia de uno o más elementos de un conjunto de páginas Web con el objetivo de uniformizar su diseño.

- **API:** Aplication Programming Interface. Interfaz de Programación de Aplicaciones, es el conjunto de funciones y procedimientos que ofrece cierta librería para ser utilizada por otro software.