Universidad de las Ciencias Informáticas
Facultad 15

Título: Módulo para la gestión de la entrada y salida de la correspondencia y documentación oficial de la Fiscalía General de la República de Cuba

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

Autores
Galia María Suárez Sánchez
Manuel Enrique Gutiérrez Pérez

Tutor
Ing. Alain Hernández López

Ciudad de la Habana, Junio de 2010

"Diseñar una experiencia hombre-máquina no se reduce a crear un mejor escritorio. Se trata de crear mundos imaginarios que tengan una relación especial con los mundos reales en los cuales se pueda extender, amplificar y enriquecer nuestras propias capacidades de pensar, sentir y actuar."

(Laurel, 1993:32-33).

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Galia María Suárez Sánchez

_____

Firma del Autor

Manuel Enrique Gutiérrez Pérez

_____

Firma del Autor

Ing. Alain Hernández López

_____

Firma del tutor

DATOS DE CONTACTO

Tutor: Ing. Alain Hernández López

Correo Electrónico: ahernandezlop@uci.cu

Autora: Galia María Suárez Sánchez
Correo electrónico: gmsuarez@estudiantes.uci.cu

Autor: Manuel Enrique Gutiérrez Pérez
Correo electrónico: megutierrez@estudiantes.uci.cu

AGRADECIMIENTOS

Me siento profundamente agradecida con todas las personas que se han cruzado en mi vida y que me han inspirado, conmovido e iluminado con su presencia.
En primer lugar agradecer a mi familia en general, por ser constante veladora de mis largos años de estudios.
En especial a mis abuelos maternos Noelia y Jorge, por siempre estar a mi lado y ser mi refugio más seguro. Ustedes son las joyas más valiosas de mi vida y con su mera existencia me impregnan de luz cada vez que respiro.
A mi madre, por todo su amor y por su apoyo incondicional. Por su fuerza ante la vida.
A mi tío Michel porque siempre ha estado pendiente de mi como un padre.
A mi amor Alain, por ser amigo, compañero, consejero, quererme, mimarme, ser especial, amarme y estar conmigo.
A todos mis amigos, por ser seres humanos maravillosos.
A la hermosa obra iniciada el 1ro de enero de 1959.
También quisiera expresar mi reconocimiento y mi gratitud a las siguientes personas, por su ayuda personal y profesional en la creación de este trabajo:
A nuestro tutor Alain, por compartir generosamente su sabiduría y su tiempo (y porque nos llegó caído del cielo).
A Manuel, por ser un extraordinario compañero.
A todos aquellos que me ayudaron a escalar la cima de este sueño, muchas gracias.

Galia María Suárez Sánchez

A mi familia en general por el apoyo que siempre me han brindado.
A todos mis amigos en la universidad que han contribuido a mi formación.
A nuestro tutor Alain por el tiempo dedicado, la experiencia y los consejos.
A mi compañera Galia sin la cual este trabajo no hubiera sido posible.

Manuel Enrique Gutiérrez Pérez

DEDICATORIA

A mis abuelos maternos Noelia y Jorge: mis ángeles de la guarda. Que la luz de su amor infinito me ilumine siempre.

A mi valiente y hermosa madre.

A mi tío Michel.

A la memoria de mi padre.

Galia María Suárez Sánchez

A mi abuela Daisy.
A mis padres.

Manuel Enrique Gutiérrez Pérez

RESUMEN

El presente trabajo de diploma tiene como objetivo proponer un sistema de gestión documental para el manejo de la documentación que transita por la Fiscalía General de la República de Cuba, que garantice un mejor control, organización y manipulación de dicha documentación.

Se presenta un estudio de temas importantes relacionados con la gestión documental y se abordan aspectos concernientes a sistemas existentes que gestionan documentación. Además, se explican las metodologías, tecnologías y herramientas utilizadas para el desarrollo de la aplicación. Este documento recoge, de manera general, todo el proceso de diseño, implementación y prueba del software desarrollado.

PALABRAS CLAVES: Gestión Documental, Sistemas de Gestión Documental, Documento, Aplicación Web.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Con el desarrollo de la sociedad moderna, la información se ha convertido en el mediador principal de todo tipo de actividades, por tanto, es válido decir que es un recurso clave para cualquier tipo de organización, ya que estas deben obtener, procesar, usar y crear información en sus diferentes procesos.

La información puede tomar varias formas, en su mayoría adquiere la de documentos de toda índole. Una organización puede llegar a procesar un volumen muy alto de documentos, provocando, de manera general, que la gestión de los mismos se vuelva un proceso de gran complejidad.

El surgimiento y desarrollo de las tecnologías de la Información y las Comunicaciones (TICs) ha tenido un impacto significativo en prácticamente todas las esferas de la sociedad y se han convertido en una herramienta indispensable, brindando la posibilidad de automatizar los procesos de gestión de la documentación en cualquier sector.

La Universidad de las Ciencias Informáticas (UCI) que surge como parte del proyecto de informatización que asume nuestro país, tiene entre sus objetivos principales la formación de ingenieros altamente calificados y el desarrollo de la producción de *software*. Dicha producción se vincula con el estudio y la investigación y se concentra en el desarrollo de programas de informatización para diferentes entidades nacionales e internacionales a través de proyectos productivos que se distribuyen en varias facultades.

Para cumplimentar dicho objetivo, actualmente la facultad 15, en coordinación con la Fiscalía General de la República (FGR, en lo adelante) y en respuesta a la necesidad de automatizar los procesos jurídicos del país, especialmente los de esta Institución, asume la realización del proyecto Sistema de Gestión Fiscal (SGF, en lo adelante). Este proyecto tiene asignada la tarea de desarrollar una aplicación Web con el objetivo de informatizar todos los procesos de la FGR, con el fin de ayudar en la toma de decisiones y permitir que los fiscales realicen su trabajo con mayor eficiencia y calidad.

La FGR, de acuerdo con al artículo 127, capítulo XIII, de la Constitución de la República de Cuba, es el órgano del Estado al que corresponde, como objetivos fundamentales, el control y la preservación de la legalidad, sobre la base de la vigilancia del estricto cumplimiento de la Constitución, las leyes y demás disposiciones legales, por los organismos del Estado, entidades económicas y sociales y por los ciudadanos; y la promoción y el ejercicio de la acción penal pública en representación del Estado.

En esta institución se reciben y envían constantemente gran cantidad de documentos de varias índoles, a través de la Recepción o de la Oficina de Correspondencia. Dentro del edificio estos documentos pueden pasar por muchos destinatarios, e incluso pueden permanecer en un destinatario archivado. La FGR gestiona toda la documentación que transita por la Institución, para ello lleva un control estricto de la misma, mediante el registro de una gran cantidad de datos de la documentación que se recibe y de la que es enviada al exterior. Igualmente registra todos los datos concernientes a los movimientos internos de los que puede ser objeto, y de las personas que la poseen en cada momento. Se hace necesario controlar

además, las cantidades totales de los documentos que han sido recibidos, los que se encuentran en la Fiscalía, así como cuáles tiene cada trabajador en su poder. Estos procesos actualmente se realizan de forma completamente manual, lo que provoca que sean muy tediosos, complejos y propensos a errores, derivándose consecuencias tales como:

- Menor control y organización sobre el estado y localización de los documentos.
- La información puede ser modificada libremente.

Para ello se ha solicitado la automatización de estos procesos al proyecto SGF, lo que permitirá llevar un control más preciso y exacto de toda la correspondencia y documentación oficial que circula en la organización, brindando así la posibilidad de conocer su estado y ubicación en todo momento.

Partiendo de esta situación problemática, se identifica el siguiente **problema científico:**

¿Cómo lograr la optimización de los procesos de gestión de la correspondencia y documentación oficial que transita por la FGR?

Para ello se ha definido como **objeto de estudio**:

El proceso de desarrollo de *software*.

El **campo de acción** por su parte, sería:

Sistemas de gestión de correspondencia y documentación.

Con estas condiciones, el **objetivo general** consiste en:

Desarrollar un módulo para el proyecto SGF que permita la gestión de la correspondencia y la documentación oficial que transita por la FGR.

La presente investigación está basada en la siguiente **hipótesis**:

El desarrollo de un módulo para el proyecto SGF que gestione la correspondencia y documentación que transita por la FGR permitirá tener un mayor control, seguimiento y localización sobre dichos documentos.

Las tareas planteadas para dar cumplimiento a estos propósitos consisten en:

- Realización del diseño teórico de la investigación.
- Redacción de la introducción.
    - Estudio del estado del arte.
- Elaboración del capítulo 1.
    - Redacción de la Propuesta a desarrollar.
    - Desarrollo del Modelo de Negocio.
    - Desarrollo del Modelo del sistema.
    - Descripción funcional del sistema.
- Elaboración del capítulo 2.
    - Desarrollo del Modelo de Diseño.
- Elaboración del capítulo 3.

- Desarrollo del Modelo de Implementación.
- Implementación de la solución.
- Desarrollo de la validación de la propuesta.

- Elaboración del capítulo 4.

Se han utilizado los **métodos científicos** siguientes:

Métodos teóricos:

Analítico – sintético: Este método es utilizado para analizar y comprender los procesos relacionados con la gestión de la documentación que transita por la FGR y establecer conclusiones sobre cómo sistematizar dichos procedimientos. Además se realiza un estudio del proyecto SGF en general, lo que posibilitó una mayor comprensión del problema en concreto.

Histórico – lógico: Su utilización posibilitó la realización de un estudio de las tendencias históricas y actuales a nivel nacional e internacional sobre el desarrollo de la gestión documental, así como hacer un análisis de las metodologías, herramientas y lenguajes utilizados en el desarrollo del sistema.

Inductivo – deductivo: A través de la inducción y la deducción se obtienen determinadas conclusiones acerca de la problemática que se plantea, que constituyen puntos de partida para la conformación de objetivos y para la toma de decisiones.

Métodos empíricos:

Entrevista: Se utiliza para obtener información detallada acerca de cómo ocurre el proceso de gestión y control de toda la documentación que transita por la FGR.

Estructuración del contenido con una breve explicación de sus partes:

Capítulo 1: El capítulo **"Fundamentación Teórica"** comprende el estado del arte sobre el tema tratado. Se expone el marco conceptual sobre el que gira la propuesta de solución, explicándose la significación de lo que constituye la Gestión Documental y cómo se comporta en la actualidad. Se realiza un análisis de las metodologías, herramientas y lenguajes a utilizar en el desarrollo del sistema.

Capítulo 2: El capítulo **"Características del sistema"** aborda, como su nombre lo indica, las características principales del sistema a desarrollar. Se explica la manera en que son realizados los diferentes procesos y los que serán automatizados. Incluye la modelación del negocio y la modelación del sistema y por tanto, la especificación de los requisitos funcionales y no funcionales como los elementos indispensables y deseables con que debe cumplir la aplicación.

Capítulo 3: Este capítulo se concentra en el desarrollo del **"Diseño del sistema"**. Comprende fundamentalmente la realización el Modelo del Diseño y la obtención de sus principales artefactos como los diagramas de clases del diseño y la especificación de las operaciones.

Capítulo 4: En este capítulo se realiza la **"Implementación y validación de la propuesta"** a través del método de prueba de caja negra y los casos de pruebas correspondientes.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

La gestión tradicional de la documentación en papel conlleva a las entidades una serie de problemas que repercuten directamente en la actividad diaria de la organización. La gestión documental es una actividad muy importante que debe adecuarse a las condiciones existentes en cada entorno organizacional. La implantación de un sistema de gestión de este tipo es clave para optimizar los procesos y mejorar el servicio ofrecido y la seguridad de la información.

En el desarrollo de este capítulo se muestran aspectos fundamentales relacionados con la Gestión Documental y se realiza un estudio del arte de algunos de los Sistemas de Gestión Documental más utilizados en el mundo. Se presenta además, un análisis detallado de las metodologías, tecnologías, herramientas y lenguajes de programación que serán empleados para el desarrollo de la aplicación.

1.2. Definiciones importantes para el dominio del problema

1.2.1. Documento

Se puede definir como documento a la "información registrada, producida o recibida en el inicio, en el proceso o en la resolución de una actividad de una institución o de un particular, y que consta del contenido, del contexto y de la estructura suficiente para proveer a la actividad de valor probatorio." (Marcos, 1999)

1.2.2. Gestión documental

La gestión documental (GD, en lo adelante) es una actividad muy antigua. Con el decursar del tiempo los hombres han tenido la necesidad de comunicar sus acciones y registrar sus actuaciones. Las formas, los métodos, las herramientas y los soportes han evolucionado, sobre durante el último siglo en que el nivel de información se ha incrementado considerablemente y la necesidad de documentarla se ha hecho un proceso muy difícil.

Varios autores han definido en qué consiste la GD, por ejemplo para Alberch Fugueras, "engloba un conjunto de operaciones comprometidas con (…) el mantenimiento, uso y destino final de los documentos a lo largo de su ciclo de vida." (Fugueras R., 2003)

Para Codina es un "conjunto de normas, técnicas y prácticas usadas para administrar el flujo de documentos de todo tipo en una organización, permitir la recuperación de información desde ellos, determinar el tiempo que los documentos deben guardarse, eliminar los que ya no son útiles y asegurar la

conservación indefinida de los documentos más valiosos, aplicando principios de racionalización y economía". (Codina, 1993)

Si bien cada autor brinda su visión con respecto al tema, se puede establecer como conclusión general que la GD posee una importancia extraordinaria para las diferentes organizaciones, debido entre otras razones, a que permite el establecimiento de procesos para mantener la información en un formato que facilite su acceso oportuno, así como su conservación, uso y eliminación, todo lo que contribuye a que se realice una labor con mayor eficacia, que los empleados sean más eficientes y que se establezcan procesos más ágiles.

1.2.3. Sistemas de Gestión Documental

Con el transcurso de los años se hizo necesario automatizar los procesos que la GD contiene y surgen con este objetivo los Sistemas de Gestión Documental (en lo adelante, SGD), los cuales ofrecen los mecanismos necesarios para el identificación, almacenamiento, seguimiento, recuperación y presentación de los documentos de una manera coherente y ordenada, es decir, pretenden el tratamiento integral, consistente y fiable de los documentos.

Una definición más conceptualizada sería: "Los sistemas de gestión documental son programas de gestión de bases de datos que disponen de una tecnología idónea para el tratamiento de documentos científicos, culturales y técnicos. Estos sistemas difieren en aspectos fundamentales de los de gestión de bases de datos convencionales, o de aplicación general, que se utilizan para la gestión de documentos administrativos." (Codina, 1993)

En la actualidad, existen en el mundo los más diversos SGD, desde el simple registro manual de la correspondencia que entra y sale, hasta los más sofisticados sistemas informáticos que manejan no sólo la documentación administrativa propiamente, sino que también controlan los flujos de trabajo, capturan información desde bases de datos, enlazan con el contenido de archivos, bibliotecas, centros de documentación y permiten realizar búsquedas y recuperar información de cualquier lugar.

1.3. Valoración del estado del arte

1.3.1. **Características de los Sistemas de Gestión Documental** (Sitio Web "La Gestión del Conocimiento")

Entre las principales características de los SGD se encuentran:

- Manejo de grandes volúmenes de documentación.
- Garantía de acceso a la información más actual.
- Mantenimiento coherente de la información procedente de diferentes compañías y organizaciones.

- Definición de flujos de trabajo en el sistema para la gestión de procesos operativos entre departamentos y empresas externas.
- Control de acceso a la información.
- Seguridad ante la posible pérdida de documentación.

1.3.2. Ventajas de los Sistemas de Gestión Documental (Sitio Web "La Gestión del Conocimiento")

De entre las ventajas que aportan estos sistemas a una organización destacan las siguientes:

- Reducción de costes de los procesos empresariales, mediante el rediseño de procesos, sustitución del trabajo administrativo no productivo y reducción del espacio físico de almacenamiento.
- Reducción de los ciclos de trabajo, aumentando la concurrencia de las distintas actividades necesarias.
- Unificación de los procesos empresariales en los distintos ámbitos departamentales y geográficos.
- Aumento de las capacidades de comunicación en toda la organización, mejorando la integridad y seguridad de la información.

1.3.3. Sistemas de Gestión Documental más utilizados en el mundo

En el mundo existen varios SGD y aunque todos persiguen generalmente el mismo objetivo y en funcionalidad hacen casi lo mismo, difieren en cuanto a sus características y ventajas de utilización. A continuación se muestran algunas descripciones de algunos de estos sistemas.

1.3.3.1. **OrfeoGPL** (Sitio Oficial OrfeoGPL)

OrfeoGPL es un SGD que comienza su desarrollo en el año 2002 por la Superintendencia de Servicios Públicos Domiciliarios (SSPD), en Colombia, como solución a la necesidad de ordenar sus archivos y tener control en la gestión de sus documentos. Fue licenciado como *software* libre bajo licencia GNU/GPL para compartir el conocimiento y mantener la creación colectiva libre.

Es una herramienta de Gestión Documental y de Procesos, que permite gestionar electrónicamente la producción, el trámite, el almacenamiento digital y la recuperación de documentos, evitando su manejo en papel, garantizando la seguridad de la información y la trazabilidad de cualquier proceso que se implemente mediante su funcionalidad. Por tanto Orfeo no es solamente un Sistema de Gestión de Documentos, sino que además es un sistema de gestión de los procesos de la Empresa o Entidad, ya que permite la incorporación de los procesos propios de las organizaciones optimizando su gestión y control y facilitando si se requiere la certificación de la calidad de los mismos.

Esta herramienta puede instalarse en cualquier Sistema Operativo (GNU/Linux, Unix, Windows), con diferentes bases de datos (PostgreSQL, Oracle, MySqL y Ms-SQL), además maneja múltiples tipos de formatos, logrando así obtener independencia de plataforma tecnológica y reducción de costos en la implantación.

1.3.3.2. KnowlegdeTree

KnowledgeTree es un sistema comercial de Gestión Documental de código abierto, que permite asegurar, compartir, administrar en general los documentos de una organización. Está diseñado para ayudar desde pequeñas a medianas empresas a controlar y administrar la colaboración de documentos, las versiones y la seguridad. Es un gestor de archivos online, que conecta procesos, personas e ideas. (Sitio Oficial de Knowledgetree, 2008)

Está disponible en tres ediciones: *KnowledgeTree Enterprise*, *KnowledgeTreeSMB* y *KnowledgeTree Open Source*, las cuales son compatibles con los sistemas operativos Windows y GNU/Linux. Mantiene una activa e innovadora comunidad de código abierto. Utiliza Apache como servidor Web, MySQL como gestor de Base de Datos, y el lenguaje de programación PHP para el código fuente. (Chalef, 2009)

Centenares de miles de empresas y entidades a lo largo del mundo ya han implementado con éxito KnowledgeTree abarcando sectores tan dispares como centros de investigación médica, servicios financieros, universidades o gobiernos, entre otros. (Calvo, 2007)

1.3.3.3. **Alfresco** (Sitio Oficial de Alfresco)

Es una plataforma para la Gestión de Contenidos en la empresa, que ofrece entre sus funcionalidades un Sistema de Gestión Documental desarrollada para empresas de pequeño y mediano tamaño, que posibilita la búsqueda y obtención de documentos, colaborar en ellos y gestionar su ciclo de vida en un repositorio centralizado. Posee un sistema de ficheros virtuales, categorización del contenido, servicios de librería, colaboración en equipo, flujo de trabajo integrado y seguridad.

Alfresco fue fundado como empresa en el 2005 por John Newton. Está enteramente desarrollado con tecnología J2EE y la forma de acceso básica es a través de su cliente Web, lo que lo convierte en un *software* multiplataforma casi sin trabajo de desarrollo adicional. Corre sobre los sistemas operativos: GNU/Linux, MacOS, Unix y Windows. Soporta varios Sistemas Gestores de Bases de Datos como Oracle, MySQL y PostgreSQL. Como servidores de aplicaciones puede usar JBoss, Apache Tomcat y J2SE 5.0.

1.3.3.4. DocManager

DocManager es un excelente sistema de administración de documentos y archivador de los mismos. Es una herramienta libre con licencia GPL, que fue creada por el desarrollador Eric Lawman.

"La aplicación como sistema de información permite mantener, controlar, consultar y administrar la documentación de la empresa; realizar evaluaciones sobre elementos del sistema; programación y ejecución de auditorías, control de no conformidades, seguimiento a los planes de acción correctivos y preventivos, generando oportunidades de mejora. Permite actualizar y consultar los indicadores de los procesos, seleccionar y evaluar a los proveedores; evaluar las competencias y desempeño del personal de la empresa y administrar las quejas y reclamos de los clientes.

Está desarrollado bajo ambiente Web, lo cual facilita el acceso a la información desde cualquier lugar, realiza notificaciones al correo electrónico de cualquier tipo de plataforma, permitiendo que se incorpore a las actividades diarias del personal de la empresa así como un manejo de agenda de actividades pendientes por realizar. DocManager permite un adecuado control sobre la información, gracias al manejo de acceso sobre usuarios, garantizando seguridad y protección, con la divulgación deseada. " (Catálogo de Software, 2010)

Completamente desarrollado en PHP puede utilizarse en los sistemas operativos Windows XP o W2003 y como Sistema Gestor de Base de Datos soporta Microsoft SQL Server.

1.4. Proyecto Sistema de Gestión Fiscal

Sistema de Gestión Fiscal es un proyecto de I+D (Investigación + Desarrollo) que consiste en una aplicación Web a través de la cual, el fiscal, desde la esfera de trabajo a la que pertenece, puede realizar todas sus actividades o procesos pertinentes, los cuales se encuentran digitalizados en una secuencia lógica de proceder según lo establecido y asistidos por las leyes que en cada caso correspondan. Este sistema constituye una completa ayuda a la toma de decisiones de los fiscales, viabiliza y humaniza el trabajo de los mismos, permitiendo reducir al máximo el margen de errores. El mismo proporcionará entre otras funcionalidades la automatización de los procesos fundamentales en los que interviene la FGR.

1.4.1. Necesidad del proyecto SGF de contar con un Sistema de Gestión Documental

Dentro de los procesos que debe automatizar el proyecto SGF para la FGR se encuentra la gestión de toda la documentación que transita por la FGR y controlar en todo momento el estado de esta documentación. Aunque los SGD analizados cuentan con numerosas ventajas, ninguno de ellos permite su adaptabilidad a los requerimientos de la FGR, debido a que:

- Los SGD estudiados están diseñados para manejar grandes flujos de documentos en formato digital y la FGR no está preparada para ese nivel de compromiso con el sistema, se desea específicamente una aplicación que gestione la localización y el estado de los documentos.

- Están creados principalmente para gestionar documentación y procesos que se generan dentro de las propias empresas u organizaciones como resultado de flujos de trabajo, aspecto que tampoco cumple con los requerimientos precisados.
- Estos sistemas analizados son desarrollados por entidades y colaboradores independientes con intereses comunes y las necesidades definidas por la FGR no serían atendidas con el esmero y la importancia requerida.

Es por estas razones que se precisa la construcción de una aplicación que sea desarrollada específicamente sobre la base de las necesidades de la FGR. Para cumplir con este objetivo se hace necesario determinar el tipo de aplicación a desarrollar, así como las metodologías, tecnologías, lenguajes y herramientas a emplear.

1.5. Aplicaciones Web

Una aplicación Web es un sistema Web (servidor Web, red, protocolo, navegador) donde la entrada del usuario (entrada de datos y navegación) afecta el estado del negocio. Su arquitectura general es la de un sistema cliente - servidor. El protocolo principal de comunicación que utilizan es HTTP (*HyperText Transfer Protocol*, Protocolo de Transferencia de Hipertexto), el cual funciona normalmente desconectado, es decir, el cliente hace una petición al servidor, este la procesa y le devuelve el resultado, terminando la comunicación entre estos.

Dentro de las principales ventajas que brinda una aplicación Web se encuentran: (Graham, 2001)

- **Compatibilidad multiplataforma:** Las aplicaciones Web tienen un camino mucho más sencillo para la compatibilidad multiplataforma que las aplicaciones de *software* descargables.
- **Inmediatez de acceso:** No necesitan ser descargadas, instaladas y configuradas.
- **Menos requerimientos de memoria:** Al residir y correr en los servidores del proveedor, las aplicaciones basadas en Web usan en muchos casos la memoria de esas computadoras, dejando más espacio para correr múltiples aplicaciones al mismo tiempo sin incurrir en frustrantes deterioros en el rendimiento.
- **Múltiples usuarios concurrentes**: Pueden ser utilizadas por múltiples usuarios al mismo tiempo.

1.5.1. Arquitectura cliente – servidor

Las aplicaciones Web se encuadran dentro de las arquitecturas cliente/servidor. En este tipo de arquitectura “un programa, el cliente, pide a otro programa, el servidor, que le preste un servicio. El cliente pide un servicio (por ejemplo, que le envíe una página Web) y el servidor cumple con el pedido. En una red, el modelo cliente/servidor ofrece una manera muy útil de interconectar programas dispersos en

diferentes lugares. El modelo cliente/servidor se ha convertido en una de las ideas centrales de la computación en red". (Sitio Web Directorio Electrónico de Guatemala)

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores. La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores Web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Las funciones que lleva a cabo el proceso cliente se pueden resumir en los siguientes puntos:

- Administrar la interfaz de usuario.
- Interactuar con el usuario.
- Procesar la lógica de la aplicación y hacer validaciones locales.
- Generar requerimientos de bases de datos.
- Recibir resultados del servidor.
- Formatear resultados.

Algunas de las funciones que lleva a cabo el proceso servidor son:

- Aceptar los requerimientos de bases de datos que hacen los clientes.
- Procesar requerimientos de bases de datos.
- Formatear datos para trasmitirlos a los clientes.
- Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

1.6. Metodologías de desarrollo de software

La industria del *software* ha evolucionado en los últimos tiempos de tal manera, que ha sido necesario desarrollar y optimizar a la par modelos y metodologías para sostener la demanda de producción de sistemas cada vez mayores en complejidad y tamaño, logrando su construcción de forma óptima y eficiente. Una metodología se encarga de elaborar estrategias de desarrollo de *software* que promuevan prácticas adaptativas en vez de predictivas, centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa del cliente.

Hoy día, existen dos categorías dentro de los cuales se pueden clasificar las metodologías. Por una parte existen las metodologías tradicionales, en las cuales lo principal es el control del proceso a través de una planificación exhaustiva, donde se controlan las actividades que se realizarán, los artefactos que se generarán, además de las herramientas y notaciones que serán usadas. Las metodologías tradicionales están basadas en la producción de proyectos de larga duración, lo que permite estructurar un amplio equipo de trabajo en roles que cumplen diferentes funciones dentro del proceso de desarrollo. Dentro de

estas se encuentra el *Rational Unified Process* (RUP), y la metodología MSF (*Microsoft Solution Framework*).

El otro gran grupo en que se clasifican las metodologías es el de las ágiles, las cuales surgidas por la incapacidad que tienen las tradicionales de operar en condiciones volátiles, se caracterizan por una gran reducción de los tiempos de desarrollo del *software*, planteando que es más importante la producción de un *software* funcional que procesar una documentación excesiva, lo que dota a estas metodologías de un gran dinamismo y de una amplia capacidad para responder ante los posibles cambios, ya que no sigue un plan estricto para desarrollar el trabajo. Las metodologías ágiles han presentado éxito en proyectos de poco alcance, donde el equipo de desarrollo no necesita ser amplio y se requiere la entrega inmediata del producto. Dos de las metodología claramente visibles en este grupo son Programación extrema (*eXtreme Programming*: XP) y Scrum.

En un proyecto de desarrollo de *software* la metodología define *Quién* debe hacer *Qué*, *Cuándo* y *Cómo* debe hacerlo, de ahí la importancia que tiene hacer una correcta elección de la metodología a emplear. Hoy día no existe una metodología que sea enteramente universal por eso para su selección hay que tener en cuenta las características de cada proyecto. RUP es la metodología seleccionada para desarrollar el sistema.

1.6.1. Proceso Unificado Racional (RUP)

El Proceso Unificado Racional (RUP: *Rational Unified Process,* en inglés) es un conjunto de metodologías que forman un marco de trabajo unificado, cohesivo y exhaustivo para el desarrollo de *software*. (The Menlo Institute, 2002)

Es una metodología de tipo tradicional muy utilizada en la actualidad pues goza de excelente prestigio dentro del mundo informático por los éxitos alcanzados. RUP se caracteriza por dividir el ciclo de vida de la producción del *software* en 4 fases:

- **Inicio:** Es donde se determina la visión del proyecto, o sea, se comprende el entorno y se determina el alcance del producto.
- **Elaboración:** En esta etapa se determinan los cimientos de la arquitectura y se analiza el dominio del problema.
- **Construcción:** En esta fase se obtiene la capacidad operacional inicial del producto.
- **Transición:** Se obtiene el *release* o liberación del producto y se pone en manos de los usuarios finales.

Además de esto cuenta con 9 flujos de trabajo, 6 principales y 3 de soporte los cuales son: Modelamiento del negocio, Requisitos, Análisis y Diseño, Implementación, Prueba, Despliegue, Gestión de configuración

y cambios, Gestión de Proyectos y Entorno, donde los 3 últimos constituyen los flujos de soporte. **(Ver Figura 1 "Fases y flujos de trabajo de RUP")**

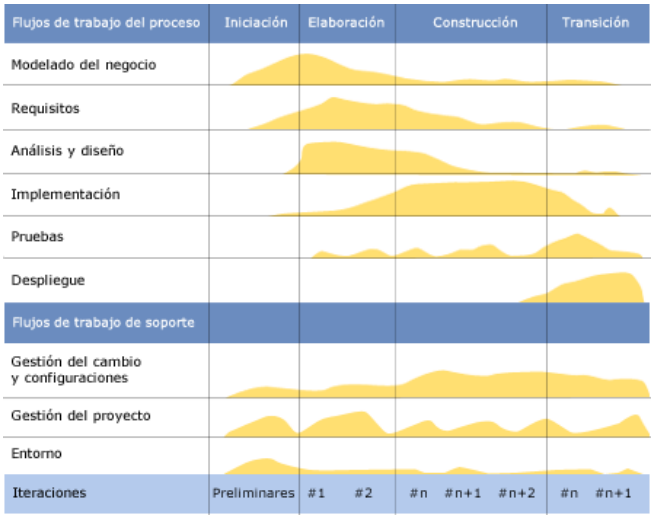


Figura 1: Fases y flujos de trabajo de RUP (Pressman, 1998)

Los elementos característicos del RUP son:

- **Actividades:** Son los procesos que se llegan a realizar en cada iteración.
- **Trabajadores:** Son las personas o entidades involucradas en cada proceso.
- **Artefactos:** Un artefacto puede ser un modelo, un elemento dentro del modelo, un documento, en fin, todo lo que puede ser generado en el proceso.

Sus características principales son:

- **Dirigido por casos de uso**: Los casos de uso guían el proceso de desarrollo pues los modelos que se obtienen representan la realización de los mismos.
- **Centrado en arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo.
- **Iterativo e Incremental:** Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. En el caso de una iteración de la fase elaboración, se centra la atención en el análisis y diseño, a la vez que se refinan los requerimientos y se obtiene un producto con un determinado nivel, que irá creciendo incrementalmente en cada iteración.

Algunas de las ventajas que presenta RUP son las siguientes:

- Evaluación de lo realizado en cada fase que permite cambios de objetivos.
- Funciona bien en proyectos de innovación.
- Realiza un seguimiento detallado en cada una de las fases.

1.7. Herramientas, lenguajes y tecnologías de desarrollo

1.7.1. Herramientas de modelado CASE

Una herramienta CASE (*Computer Aided Software Engineering* o Ingeniería de *Software* Asistida por Computadora), es cualquier herramienta informática utilizada para la planificación, desarrollo y evolución del *software*. (Lamb, 1998)

La tecnología CASE supone la automatización del desarrollo del *software*, de la documentación, la generación de código, el chequeo de errores y la gestión de proyecto, contribuyendo a mejorar la calidad y la productividad en el desarrollo de sistemas de información. Entre las ventajas de su utilización se encuentra que permiten aumentar la portabilidad y la reutilización del *software*, así como la estandarización de la documentación. Es por estas razones que resulta de gran importancia el escoger correctamente esta herramienta para asegurar el cumplimiento de los objetivos propuestos.

Existen numerosas herramientas CASE, desde las que abarcan el ciclo completo de desarrollo de *software*, las orientadas a la automatización y soporte de las actividades en las primeras fases de desarrollo: análisis y diseño, así como las dirigidas a las últimas fases de construcción e implantación. Algunos ejemplos son: ERwin, EasyCase y PowerDesign. Las mismas son utilizadas específicamente para el diseño de bases de datos; para la generación de esquemas de bases de datos e ingeniería inversa, así como la automatización de las fases de análisis y diseño; y para el análisis, diseño inteligente y construcción sólida de una base de datos, respectivamente. Otras herramientas muy utilizadas que soportan el ciclo de vida completo son Rational Rose y Visual Paradigm. Esta última es la que se utilizará en el desarrollo de la aplicación propuesta.

1.7.1.1. Visual Paradigm para UML

Visual Paradigm para UML es una herramienta de modelado CASE creada por la compañía Visual Paradigm, con sede en Hong Kong, China.

Ventajas:

- Es una herramienta con un diseño centrado en casos de uso enfocados al negocio.
- Posee capacidades de ingeniería directa e inversa, modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Está disponible en múltiples versiones y plataformas.
- Permite dibujar 13 tipos de diagramas diferentes a través de un intuitivo modelado visual.
- Permite generar código a partir de los diagramas creados.
- Admite la importación y exportación de XML e imágenes, la administración de requerimientos y la creación de esquemas de clases a partir de una base de datos y viceversa.

1.7.2. Lenguaje de modelado de software

Los lenguajes de modelado de *software* ayudan a los ingenieros a "visualizar" el sistema a construir. Aunque existen varios lenguajes de este tipo como BPML (*Business Process Modelling Language*, por sus siglas en inglés) o Lenguaje para modelar Procesos de Negocio, utilizado como su nombre lo indica para el modelado de procesos de negocio, UML (*Unified Modelling Language*, por sus siglas en inglés) o el Lenguaje Unificado de Modelado, es el lenguaje de modelado de sistemas de *software* más conocido y utilizado en la actualidad, ya que se ha convertido en "un *estándar de la industria,* pero no sólo de la industria del *software* sino, en general, de cualquier industria que requiera la construcción de modelos como condición previa para el diseño y posterior construcción de prototipos." (Rumbaugh, y otros, 1998)

1.7.2.1. Lenguaje Unificado de Modelado (UML)

El Lenguaje Unificado de Modelado fue creado por Jim Rumbaugh, Ivar Jacobson y Grady Booch. Es un lenguaje de modelado visual multi-propósito usado para especificar, visualizar, construir y documentar los artefactos de un sistema de *software*. Captura las decisiones y la comprensión de los sistemas a construir, por lo que comprende, diseña, analiza, configura, mantiene y controla información de tales sistemas.

Algunas propiedades de UML son las siguientes:

- Es un lenguaje distribuido y adecuado a las necesidades de conectividades actuales y futuras.
- Reemplaza a decenas de notaciones empleadas con otros lenguajes.
- Modela estructuras complejas.
- Las estructuras más importantes que soportan tienen su fundamento en las tecnologías orientadas a objetos, tales como objetos, clase, componentes y nodos.
- Emplea operaciones abstractas como guía para variaciones futuras, añadiendo variables si es necesario.
- Modela el comportamiento completo del sistema: casos de uso, diagramas de secuencia y de colaboraciones, que sirven para evaluar el estado de las máquinas. (Rumbaugh, y otros, 1998)

UML se basa en la representación del sistema a través de diferentes vistas compuestas por una serie de diagramas, los cuales son especificados en la siguiente tabla: **(Tabla 1: Vistas y diagramas de UML)**

Área	*Vista*	*Diagramas*	*Conceptos Principales*
estructural	vista estática	diagrama de clases	clase, asociación, generalización, dependencia, realización, interfaz
	vista de casos de uso	diagrama de casos de uso	caso de uso, actor, asociación, extensión, inclusión, generalización de casos de uso
	vista de implementación	diagrama de componentes	componente, interfaz, dependencia, realización
	vista de despliegue	diagrama de despliegue	nodo, componente, dependencia, localización
dinámica	vista de máquina de estados	diagrama de estados	estado, evento, transición, acción
	vista de actividad	diagrama de actividad	estado, actividad, transición de terminación, división, unión
	vista de interacción	diagrama de secuencia	interacción, objeto, mensaje, activación
		diagrama de colaboración	colaboración, interacción, rol de colaboración, mensaje
gestión del modelo	vista de gestión del modelo	diagrama de clases	paquete, subsistema, modelo
extensión de UML	todas	todos	restricción, estereotipo, valores etiquetados

Tabla 1: Vistas y diagramas de UML (Rumbaugh, y otros, 1998)

1.7.3. Lenguajes de programación Web

Los lenguajes de programación son herramientas que permiten crear programas, representan en forma simbólica y en manera de un texto los códigos que podrán ser interpretados por una persona. Actualmente existen diferentes lenguajes de programación para desarrollar en la web, del lado del cliente existen varias tecnologías que pueden ser utilizadas, como ActiveX y *applets,* pero no están tan estandarizadas como las que se contemplan en el epígrafe correspondiente: XHTML y JavaScript.

Los lenguajes del lado del servidor son aquellos que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían en un formato comprensible para el cliente. Algunos de los más utilizados son Perl (*Practical Extracting*), ASP (*Active Server Pages*), y PHP (*Hypertext Pre-processor*). Este último es gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. El mismo será el lenguaje utilizado para la programación del sistema.

1.7.3.1. Programación del lado del cliente

1.7.3.1.1. XHTML

XHTML (*eXtensible Hipertext Markup Language*, Lenguaje de Marcado de Hipertexto extensible) es una versión modernizada de HTML (*HyperText Markup Language,* Lenguaje de Marcas de Hipertexto). XHTML 1.0 es una adaptación de HTML 4.01 al lenguaje XML (*eXtensible Markup Language*, Lenguaje de

Marcado Extensible), manteniendo así sus características pero añadiendo algunas restricciones y elementos propios de XML.

1.7.3.1.2. JavaScript

Es uno de los lenguajes de programación del lado del cliente más utilizado, gracias a su compatibilidad con la mayoría de los navegadores modernos.

"JavaScript proporciona los medios para:

- Controlar las ventanas del navegador y el contenido que muestran.
- Programar páginas dinámicas simples.
- Evitar depender del servidor Web para cálculos sencillos.
- Capturar los eventos generados por el usuario y responder a ellos sin salir a Internet.
- Comprobar los datos que el usuario introduce en un formulario antes de enviarlos.
- Comunicarse con el usuario mediante diversos métodos". (Sitio Web E.T.S.I.Informática)

1.7.3.1.3. Librería jQuery

jQuery es una librería de JavaScript rápida, concisa y compatible con múltiples navegadores que se especializa en la interacción entre JavaScript y HTML[1]. Fue liberada en enero del 2006 por John Resig. Simplifica el trabajo con documentos HTML, manejo de eventos, animación, e interacciones Ajax para el desarrollo rápido de aplicaciones web. JQuery es *software open source* y está diseñada para cambiar la forma en que se escribe JavaScript. (Sitio Oficial de jQuery)

1.7.3.1.4. **AJAX** (Eguíluz Pérez, 2008)

El término AJAX es un acrónimo de *Asynchronous JavaScript + XML*, que se puede traducir como "JavaScript asíncrono + XML".

AJAX es la combinación de varias tecnologías independientes:

- XHTML y CSS[2], para crear una presentación basada en estándares.
- DOM[3], para la interacción y manipulación dinámica de la presentación.
- XML, XSLT[4] y JSON[5], para el intercambio y la manipulación de información.

[1]HTML: *HyperText Markup Language.*
[2]CSS: *Cascading Style Sheets,* (*Hojas* de estilo en cascada).
[3]DOM: *Document Object Model.*
[4]XSLT: *Extensible Stylesheet Language Transformations*, (Transformación del lenguaje de hojas de estilo extensible).
[5]JSON: *JavaScript Object Notation*

- XMLHttpRequest[6], para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

1.7.3.2. Programación del lado del servidor

1.7.3.2.1. **PHP v5.x** (Vaswani, 2008)

PHP es un acrónimo recursivo que significa *Hypertext Pre-processor* (inicialmente *PHP Tools*, o, *Personal Home Page Tools*) es un lenguaje "*open source*" interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor, que fue escrito originalmente por Rasmus Lerdorf.

Ventajas de PHP:

- **Rendimiento:** El motor de PHP 5 fue rediseñado completamente con un administrador de memoria optimizado para mejorar el rendimiento.
- **Portabilidad:** PHP está disponible para sistemas operativos como UNIX, Microsoft Windows, Mac OS y OS/2. Los programas escritos en este lenguaje son portables entre plataformas.
- **Facilidad de uso:** Su sintaxis es clara y consistente, cuenta además con documentación exhaustiva para todas las funciones de su núcleo.
- **Código Abierto:** El lenguaje es desarrollado por una comunidad de programadores voluntarios que publican su código libremente en la Web y puede ser usado sin el pago de licencias o inversiones en hardware costoso. Esto reduce el costo de la producción del *software* sin afectar la flexibilidad o confiabilidad.
- **Soporte comunitario:** Cuenta con amplio soporte gracias a la numerosa comunidad de programadores que lo usan en todo el mundo.

1.7.3.3. Entorno de desarrollo integrado Eclipse

Un entorno de desarrollo integrado (*integrated development environment*, IDE en inglés) proporciona un marco de trabajo amigable para gran cantidad de lenguajes de programación. Además es posible que un mismo entorno de desarrollo tenga la posibilidad de utilizar varios lenguajes de programación. Netbeans, ZendStudio y Eclipse son entornos de desarrollo integrados que pueden utilizar PHP como lenguaje de programación. Este último es el que se utilizará para el desarrollo de la aplicación propuesta.

[6]XMLHttpRequest: también referida como XMLHTTP, *Extensible Markup Language / Hypertext Transfer Protocol*.

Eclipse es una plataforma de programación utilizada para crear entornos de desarrollo. Sus creadores lo definen como un IDE para todo y nada en particular. La arquitectura de *plugins* de Eclipse permite, además de integrar diversos lenguajes sobre un mismo IDE, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces y ayuda en línea para librerías.

1.8. Sistemas de Gestión de Base de Datos

"El sistema de gestión de la base de datos (SGBD) es una aplicación que permite a los usuarios definir, crear y mantener la base de datos, y proporciona acceso controlado a la misma.**"** (Marqués Andrés)

Para el desarrollo de la aplicación propuesta es necesario contar con un SGBD, ya que se maneja una gran cantidad de información que hay que controlar mediante una base de datos y se necesita hacerlo de manera eficiente. Entre los SGBD más usados en la actualidad se encuentran MySQL, Oracle y PostgreSQL.

El primero de estos es un sistema relacional, multihilo y multiusuario. Es, además, un sistema propietario patrocinado por una empresa privada, que posee el *copyright* de la mayor parte del código.

Oracle es un sistema muy potente para la gestión de base de datos. Es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hace que solamente se vea en empresas muy grandes y multinacionales, por norma general.

PostgreSQL por su parte es un sistema de bases de datos de código abierto muy avanzado. A esta realidad contribuye el hecho de que no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales la cual trabaja en su desarrollo y perfeccionamiento. Este es el SGBD determinado para ser utilizado en la aplicación.

1.8.1. **PostgreSQL 8.4** (Alarcón, 2006)

PostgreSQL es un potente sistema de base de datos relacional libre. Fue desarrollado en la Universidad de California, en el departamento de Ciencias de la Computación. Posee más de 15 años de activo desarrollo y arquitectura probada que se ha ganado una muy buena reputación por su confiabilidad e integridad de datos. Funciona en los principales sistemas operativos, incluyendo las distribuciones de GNU/Linux, las variantes de UNIX y Windows.

Principales características:

- Soporta casi toda la sintaxis SQL, tiene soporte total para llaves foráneas, uniones, vistas, disparadores y procedimientos almacenados.
- Usa una arquitectura cliente/servidor.

- Tiene soporte interno para lenguajes procedurales como Python, Perl, TCL, los cuales permiten agregarle mayor dinamismo al sistema.
- Soporta herencia de tablas.
- Incluye la mayoría de los tipos de datos de SQL92 y SQL99.
- Posee puntos de recuperación a un momento dado.
- Replicación asincrónica.
- Copia de seguridad en línea.

Ventajas:

- Máximo tamaño de base de datos ilimitado.
- Máximo tamaño de tabla 32 TB.
- Máximo tamaño de tupla 1.6 TB.
- Máximo tamaño de campo 1 GB.
- Máximo de tuplas por tabla ilimitado.
- Máximo columnas por tabla 250 - 1600 dependiendo del tipo de columnas.
- Máximo de índices por tabla ilimitado.

1.9. Servidor Web

Un servidor Web "es el programa que, utilizando el protocolo de comunicaciones HTTP, es capaz de recibir peticiones de información de un programa cliente (navegador), recuperar la información solicitada y enviarla al programa cliente para su visualización por el usuario." (Sitio Web Arazandi)

Actualmente, a pesar de que existen cierto número de servidores Web, uno de los más utilizados es Apache. Conforme a un informe publicado en el sitio www.netcraft.com el 7 de junio de este año, según el criterio de cuotas de mercado de los principales servidores, Apache ocupa la primera posición con 53, 84% por encima de servidores como Microsoft (24,08%) y Google (7,04%).

1.9.1. Servidor Web Apache 2.2.4

Entre las características más destacables de Apache están las siguientes:

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Es una tecnología gratuita de código fuente abierto.
- Es un servidor altamente configurable de diseño modular.
- Trabaja con gran cantidad de Perl, PHP y otros lenguajes de script. También trabaja con Java y páginas jsp.
- Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.

- Tiene una alta configurabilidad en la creación y gestión de logs ya que permite la creación de ficheros de log a medida del administrador. (Vargas Del Valle, 2007)

1.10. *Framework*

Aunque los lenguajes de programación Web cuentan con numerosas ventajas, desarrollar hoy en día de la manera tradicional o "a puro código" no es para nada factible. Una de las tendencias más generalizadas es el uso de aplicaciones pre-elaboradas que sirven como base a otras aplicaciones más complejas, las cuales son conocidas como *frameworks* o marcos de trabajo.

"Un *framework* simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un *framework* proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un *framework* facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas". (Zaninotto, y otros)

Por las ventajas antes descritas, la utilización de un *framework* en la implementación constituye una gran ayuda para el desarrollo de la aplicación. Algunos de los ejemplos de los más utilizados son ZendFramework, Codelgniter y Symfony. Aunque cada uno de ellos cuenta con gran cantidad de características deseables y numerosas ventajas, como por ejemplo, que todos poseen código abierto, permiten desarrollar aplicaciones con mayor rapidez y la utilización del lenguaje de programación PHP, teniendo en cuenta las necesidades de la solución se empleará el *framework* Symfony.

1.10.1. **Symfony 1.3** (Zaninotto, y otros)

Symfony es un completo *framework* diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.

Symfony está desarrollado completamente con PHP 5, presenta un código fácil de leer y que permite un mantenimiento muy sencillo. Es además, fácil de extender, ya que posibilita su integración con librerías desarrolladas por terceros. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

1.11. Fundamentación de la metodología, las tecnologías, lenguajes de programación y herramientas utilizados

Se escoge la metodología, las tecnologías, lenguajes de programación y herramientas que se proponen dado que son elementos de punta dentro del conjunto de técnicas más actualizadas que se utilizan para desarrollar *software* y poseen numerosas ventajas y características que se corresponden con las necesidades y exigencias del problema a resolver planteado.

Además, el módulo a desarrollar forma parte del proyecto SGF y este debe acoplar su desarrollo con el sistema principal, lo que conlleva a la utilización de los elementos establecidos en la línea base del proyecto. Influye también el hecho de que en el proyecto se ha generado cierto conocimiento en el uso de estas herramientas y tecnologías y es conveniente su reutilización.

1.12. Patrones

¿Qué es un Patrón?

"Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos sobre cómo aplicarlo en nuevas situaciones, o sea, un patrón es una descripción de un problema bien conocido que suele incluir: descripción, escenario de uso, solución concreta, consecuencias de utilizar el patrón, ejemplos de implementación y lista de patrones relacionados". (Larman, 2008)

Patrones de Casos de Uso

Las técnicas y diseños exitosos que se utilizan una y otra vez en modelos de casos de uso se formalizan como patrones que expresan buenos modelos de casos de uso. Un patrón de caso de uso capta una técnica para hacer el modelo mantenible, reutilizable y comprensible. (Övergaard, y otros, 2004)

Patrón "El nombre revela la intención": Hace un llamado al uso de nombres descriptivos para los casos de usos, que ubiquen expectativas en el lector. Es decir que revelen la intención del caso de uso y refleje el único objetivo que el actor está intentando lograr. Para ello este patrón propone como buena práctica nombrar los casos de uso comenzando con un verbo activo que describa el uso del caso de uso y seguido del verbo con una frase que describa su propósito. Ser conciso pero lo suficientemente descriptivo para capturar la esencia del caso de uso. (Övergaard, y otros, 2004)

Patrón "Completar una única meta": Consiste en escribir cada caso de uso dirigido hacia una completa y bien definida meta.

Patrón "Claro Conjunto de Roles": Plantea identificar los actores y el rol que cada uno juega con respecto al sistema. Describir claramente cada uno de los actores con que el sistema debe actuar recíprocamente. (Övergaard, y otros, 2004)

Patrones arquitectónicos

Un patrón arquitectónico "representa, en términos de componentes y las relaciones entre ellos, posibles soluciones para incorporar en el diseño, aspectos que mejoran la usabilidad del sistema final." (Moreno, y otros)

Dentro de los patrones de arquitectura se puede encontrar el patrón Modelo – Vista - Controlador (MVC) y el Patrón Modelo de Tres Capas.

Patrón "MVC": En el desarrollo de la aplicación informática que propuesta se utilizará el *framework* Symfony, el cual está basado en el patrón clásico MVC, que está formado por tres niveles:

- El Modelo, que representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La Vista, que transforma el modelo en una página web que permite al usuario interactuar con ella.
- El Controlador, que se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos, comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Identifica clases, instancias, roles, colaboraciones y la distribución de responsabilidades. (Gracia, 2005)

Patrones GRASP[7]:

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. (Larman, 2008)

Se pueden destacar 5 patrones GRASP principales, los cuales son aplicados en el *framework* Symfony, utilizado para desarrollar la aplicación:

Patrón "Experto": La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados. Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. Con el uso de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece el hecho de tener sistemas más robustos y de fácil mantenimiento. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases "sencillas" y

[7] GRASP: *General Responsibility Assignment Software Patterns* (Patrones Generales de Software para Asignación de Responsabilidades).

más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

Patrón "Creador": se asigna la responsabilidad de que una clase B cree un objeto de la clase A solamente cuando:

- B contiene A.
- B es una agregación (o composición) de A.
- B almacena A.
- B tiene los datos de inicialización de A.
- B usa A.

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte a bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización.

Patrón "Alta Cohesión": Cada elemento del diseño debe realizar una labor única dentro del sistema. Ejemplos de una baja cohesión son las que hacen demasiadas cosas. Este patrón mejora la claridad y la facilidad con que se entiende el diseño, simplifica el mantenimiento y las mejoras en funcionalidad y a menudo se genera un bajo acoplamiento.

Patrón "Bajo Acoplamiento": Debe haber pocas dependencias entre las clases. Estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento y produzca resultados negativos propios de un alto acoplamiento. Esto contribuye que las clases no se afecten por cambios de otros componentes, sean posibles de entender por separado y fáciles de reutilizar.

Patrón "Controlador": Se encarga de asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades, mayor potencial de los componentes reutilizables y la garantía de que los procesos de dominio sean manejados por la capa de los objetos del dominio y no por la de la interfaz.

Patrones GOF[8]

Existen tres tipos de patrones GOF:

Patrones de Comportamiento: Los patrones de comportamiento están relacionados con algoritmos y asignación de responsabilidades a los objetos. Estos patrones describen no sólo patrones de objetos sino también patrones de comunicación entre ellos. Dentro de ellos se pueden clasificar en función de que

[8]GOF: *Guest Observer Facility*

trabajen con clases (*Template Method, Interpreter*) u objetos (*Chain of Responsibility, Command, Iterator, Mediator, Memento, Observer, State, Strategy, Visitor*). (Albacete, 2002)

Patrones de Estructura: Los patrones estructurales están relacionados con cómo las clases y los objetos se combinan para dar lugar a estructuras más complejas. También se puede hablar de patrones estructurales asociados a clases (*Adapter*) y asociados a objetos (*Bridge, Composite, Decorator, Facade, Flyweight, Proxy*), los primeros utilizan la herencia y los últimos la composición. (Albacete, 2002)

Patrones de Creación: Los patrones de creación proporcionan ayuda a la hora de crear objetos, principalmente cuando esta creación requiere de la toma de decisiones, la cual puede ser dinámica. Estos patrones ayudan a estructurar y encapsular dichas decisiones. En algunas ocasiones existe más de un patrón que se puede aplicar a la misma situación y en otras se pueden combinar múltiples patrones convenientemente. Un patrón de creación asociado a clases usa la herencia para variar la clase que se instancia, mientras que un patrón de creación asociado a objetos delegará la instanciación a otro objeto. Ejemplos de estos patrones son *Singleton, Builder, Prototype*. (Albacete, 2002)

El *framework* que se utiliza para desarrollar la aplicación, Symfony, implementa los siguientes patrones GOF:

En la categoría de **creacionales** Symfony utiliza el patrón:

***Decorator*:** Añade responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades. En Symfony el contenido de la plantilla se integra en el *layout*, o si se mira desde el otro punto de vista, el *layout* decora la plantilla.

***Singleton*:** Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

***Command*:** Encapsula peticiones en forma de objetos permitiendo así parametrizar los clientes utilizando distintas peticiones, encolar las peticiones y ofrecer la posibilidad de deshacer las operaciones. Permite solicitar operaciones sin tener que saber cómo o quien lleva a cabo esas operaciones.

En la categoría de **comportamiento**:

***Observer*:** Para no recurrir a soluciones fuertemente acopladas (que reducen la posibilidad de reutilización), este patrón define una dependencia uno a muchos entre objetos, para que, cuando uno de ellos cambie su estado, todos los que dependan de él sean avisados y puedan actualizarse convenientemente.

Registry: Provee un mecanismo para almacenar datos globales, esto permite la persistencia de estos datos durante la vida del sistema.

Front Controller: Permitir que sólo exista un único punto de entrada en la aplicación, esto ayuda a centralizar las restricciones de seguridad y a delegar las peticiones en otros componentes.

1.13. Conclusiones parciales

Al finalizar este capítulo se arriba a las siguientes conclusiones:

- Los conceptos estudiados resultaron de gran importancia para una mejor comprensión del objetivo de este trabajo.
- El análisis de los SGD permitió concluir que por sus características ninguno es adaptable a las necesidades que presenta la FGR, por lo que se hace necesario el desarrollo de una aplicación que responda a la automatización de las mismas.
- Las metodologías, herramientas, tecnologías y lenguajes de programación que serán utilizados son los idóneos para el desarrollo del sistema y se resumen en las siguientes:
  - RUP, como metodología de desarrollo.
  - UML, como lenguaje de modelado.
  - Visual Paradigm para UML, como herramienta de modelado CASE.
  - PHP, como lenguaje de programación.
  - Eclipse, como entorno de desarrollo.
  - Symfony, como marco de trabajo.
  - PostgreSQL y Apache, como servidores de BD y Web, respectivamente.
- Los patrones de casos de uso y diseño a emplear, permitirán obtener una solución más eficiente y reutilizable.

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

2.1. Introducción

En el siguiente capítulo se describe la solución propuesta. Se hace referencia a las reglas del negocio asociadas al dominio del problema, ofreciendo una visión general del funcionamiento del proceso de gestión documental que se realiza actualmente. Se describen los requisitos funcionales y no funcionales que debe cumplir el sistema que se propone, se realiza su modelación en términos de casos de uso y los mismos son detallados en sus descripciones, todo lo que permite hacer una concepción general de la aplicación a desarrollar.

2.2. Modelación del negocio

Un sistema, por pequeño que sea, generalmente es complicado. Por eso se necesita dividirlo en piezas si se pretende comprenderlo y gestionar su complejidad. Esas piezas se pueden representar a través de modelos que permitan abstraer sus características esenciales. De ahí, que en el campo del *software*, resulte útil la creación de modelos que organicen y presenten los detalles importantes de problemas reales que se vinculan con el sistema informático a desarrollar. Estos modelos deben cumplir una serie de propiedades, entre ellas la de ser coherentes y relacionados. Uno de los modelos útiles previo al desarrollo de un *software* es el modelo del negocio.

2.2.1. ¿Qué es el modelo del negocio?

“El modelado del negocio es una técnica para comprender los procesos del negocio de la organización”. (Jacobson, y otros, 2000)

Entre los principales objetivos de un modelo de negocio se encuentran:

- Comprender la estructura y la dinámica de la organización en la cual se va a implantar un sistema.
- Comprender los problemas actuales de la organización e identificar las mejoras potenciales.
- Asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización.
- Derivar los requerimientos del sistema que va a soportar la organización.

2.2.2. Descripción del negocio actual

La FGR está estructurada de la forma siguiente:

- Fiscalía General de la República.
- Fiscalías provinciales.

- Fiscalías municipales.
- Fiscalía Militar

La Fiscalía General tiene su sede en la capital de la República y tiene entre sus objetivos, además de los fundamentales que le asigna la Constitución, los siguientes:

- Procurar el restablecimiento de la legalidad cuando sea quebrantada por disposiciones o decisiones contrarias a la Constitución y las leyes o por aplicación indebida o incumplimiento de estas.
- Promover la sanción de quienes atenten contra la independencia y la soberanía del Estado, así como contra los intereses políticos, económicos y sociales de este.
- Proteger a los ciudadanos en el ejercicio legítimo de sus derechos e intereses.

La FGR está constituida por el Fiscal General, los vicefiscales generales, las unidades organizativas, los fiscales y el personal auxiliar que determine el Fiscal General.

En esta Institución se reciben a diario documentos de toda índole: reclamaciones civiles, laborales, de familia, penales, quejas sobre cualquier asunto que estime la población que deba intervenir, documentos variados de diferentes Instituciones, expedientes penales, confiscatorios, sobre verificaciones fiscales, etcétera.

Para un mejor entendimiento de este proceso los documentos que se reciben se dividen en las siguientes clasificaciones:

- **Documentos comunes:** Se refiere a los documentos que por lo general son entregados en la Recepción por cualquier persona. (carta personal, queja, reclamación, etc.)
- **Documentos oficiales:** Se refiere a los documentos que por lo general deben ser entregados en la Oficina de Correspondencia. Pueden ser valijas oficiales, documentos procedentes de otras instituciones o fiscalías, etc. Habitualmente son entregados por alguien con la autorización requerida (un trabajador de otra Oficina de Correspondencia).

Todos estos documentos se reciben a través de la Recepción y/o de la Oficina de Correspondencia y son trasladados a sus respectivos destinatarios dentro del edificio. Los mismos pueden permanecer en poder de su destinatario inicial, puede ser reenviado para otro departamento de la Fiscalía o para un destino externo a ella. Cada vez que un documento es recibido o sufre algún movimiento (interno o externo) debe quedar plasmado en los registros respectivos designados para ello. De igual manera se hace necesario controlar las cantidades totales de documentos que se encuentran en la Institución y los que mantiene cada fiscal en su poder.

Este trabajo se efectúa actualmente de forma manual, lo que repercute directamente en que se tenga un menor control y organización, que existan atrasos en los procesos, demora al consultar los estados de los documentos y afecta además, la toma rápida y eficiente de decisiones.

2.2.3. Reglas del negocio

"Las reglas de negocio describen políticas que deben cumplirse o condiciones que deben satisfacerse, por lo que regulan algún aspecto del negocio". (Entorno Virtual de Aprendizaje, 2009 - 2010)

En relación al negocio que se modela se encuentran las siguientes reglas del negocio:

- Los documentos de entrada sólo llegan a la Fiscalía por dos vías: Recepción y Oficina de Correspondencia.
- Los documentos de salida se generan en los departamentos de la Fiscalía.
- Los documentos de salida pueden estar vinculados a otro(s) y pueden tener dos destinos:
  - Movimientos internos: para los departamentos dentro de la misma Fiscalía.
  - Movimientos externos: Sus destinatarios son las Fiscalías provinciales u otros organismos.
- Solamente los usuarios internos (secretarias de los departamentos o los propios fiscales) podrán darle un nuevo destino a los documentos que ellos han recibido.
- Las salidas de documentos para el exterior solamente serán tramitadas por la Oficina de Correspondencia y los mismos deben proceder de un departamento dentro de la Fiscalía.

2.2.4. Actores del negocio

"Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados." (Entorno Virtual de Aprendizaje, 2009 - 2010)

En este caso se cuenta con los siguientes Actores del Negocio:

- Fiscal.
- Portador.
- Solicitante.

2.2.4.1. Descripción de los actores del negocio

A continuación se especifican las descripciones de cada uno de los actores del negocio. **(Tabla 2: Descripción de los actores del negocio)**

Actores del negocio	Justificación
Fiscal	Interactúa con el negocio motivado por la necesidad de enviar o recibir algún

	documento, así como obtener datos del estado de estos.
Portador	Interactúa con el negocio motivado por la necesidad de hacer la entrega de algún documento.
Solicitante	Interactúa con el negocio con el fin de realizar peticiones sobre determinada información de interés, que le permita tomar decisiones o cumplir con ciertas tareas.

Tabla 2: Descripción de los actores del negocio

2.2.5. Trabajadores del negocio

"Un trabajador del negocio es una abstracción de una persona (o grupo de personas), una máquina o un sistema automatizado; que actúa en el negocio realizando una o varias actividades, interactuando con otros trabajadores del negocio y manipulando entidades del negocio. Representa un rol." (Entorno Virtual de Aprendizaje, 2009 - 2010)

En este caso se cuenta con los siguientes trabajadores:

- Recepcionista
- Trabajador de Correspondencia.
- Secretaria.

2.2.5.1. Descripción de los trabajadores del negocio

A continuación se especifican las descripciones de cada uno de los trabajadores del negocio. **(Tabla 3: Descripción de los trabajadores del negocio)**

Trabajadores del negocio	Justificación
Recepcionista	Encargado de recibir y registrar los documentos comunes que llegan a la Fiscalía por esta vía, así como de brindar cualquier información referente a este proceso que le sea solicitada.

Trabajador de Correspondencia	Encargado de recibir y registrar los documentos tanto comunes como oficiales que lleguen a la Fiscalía por esta vía, así como los provenientes de la Recepción y darles un destino dentro del edificio. También se encarga de tramitar los documentos destinados al exterior del edificio, así como de brindar cualquier información referente a este proceso que le sea solicitado.
Secretaria	Encargada de recibir todos los documentos que envíe Correspondencia y darle un nuevo destino dentro de la propia Institución o fuera de ella de ser necesario, así como de brindar cualquier información referente a este proceso que le sea solicitada.

Tabla 3: Descripción de los trabajadores del negocio

2.2.6. Casos de uso del negocio

"Un modelo del negocio describe los procesos de negocio (…) en términos de casos de uso del negocio y actores del negocio que se corresponden con los procesos del negocio y los clientes, respectivamente." (Jacobson, y otros, 2000)

En este caso se tienen los siguientes casos de uso del negocio (CUN, en lo adelante):

CUN 1: Entregar documentos en Recepción.

CUN 2: Entregar documentos en Correspondencia.

CUN 3: Realizar envío interno de documento.

CUN 4: Realizar envío externo de documento.

CUN 5: Solicitar información.

2.2.6.1. Diagrama de casos de uso del negocio

A continuación se muestra el diagrama de casos de uso del negocio, donde se puede observar la interrelación que existe entre los tres actores del negocio identificados y los cinco CUN definidos. **(Figura 2: Diagrama de casos de uso del negocio)**

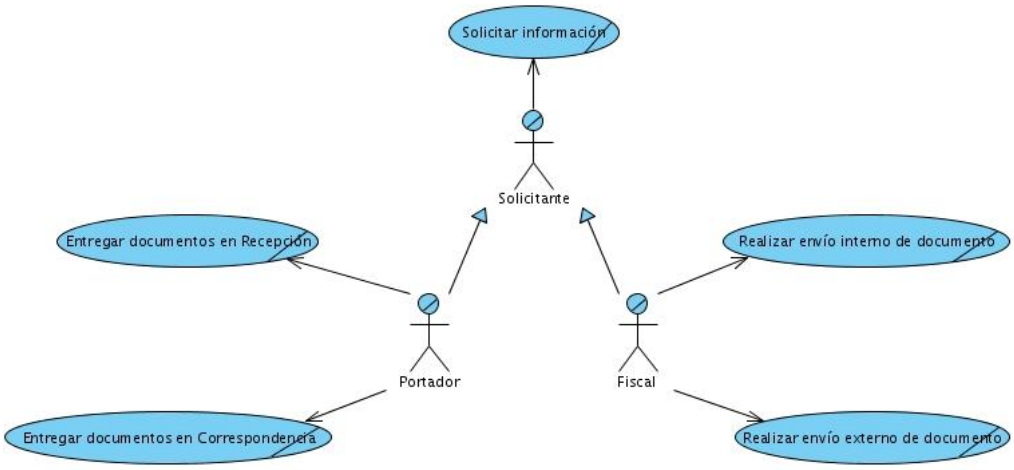


Figura 2: Diagrama de casos de uso del negocio

2.2.6.2. Descripción de los casos de uso del negocio

En las descripciones de los CUN se detallan los procesos que se ejecutan y que son ejemplificados mediante los Diagramas de actividades.

A continuación se presenta la descripción del CUN "Entregar documentos en Recepción". **(Tabla 4: Descripción del CUN "Entregar documentos en Recepción")**

Para acceder a la descripción íntegra de todos los CUN, consultar el documento Modelo del Negocio.

Caso de Uso del Negocio	**Entregar documentos en Recepción**
Actores	Portador
Trabajadores	Recepcionista
Propósito	Recibir toda la documentación que llega a la Fiscalía por la Recepción.
Casos de Uso Asociados	**-**
Resumen El caso de uso de inicia cuando cualquier Portador de documentos se presenta en la Recepción para hacer entrega de los mismos. Esta solicitud es atendida por la Recepcionista, la cual	

procede a insertar los datos requeridos del o los documentos que han sido entregados y finalmente los registra como recibidos, le proporciona al Portador un comprobante y los traslada a la Oficina de Correspondencia.	
Precondiciones	Que se necesite hacer la entrega de algún documento en la Fiscalía.
Flujo normal de los eventos	
Acciones del actor	**Respuesta del proceso de negocio**
1- El Portador llega a la Recepción para hacer entrega de la documentación.	**2-** La Recepcionista solicita el Carné de Identidad del Portador para registrar sus datos personales (nombre, carné de identidad, dirección) y registra, además, los datos del o los documentos que han sido entregados (No. del documento, fecha de recibido, tipo, clasificación, promovente, observaciones de entrada) en el Registro de documentos de Recepción. **Ver Flujo alterno** (si se necesita entregar un documento oficial).
	3- La Recepcionista entrega un Comprobante al Portador con los datos: número del documento, fecha de recibido, nombre del Portador y nombre y apellidos de la Recepcionista (la persona que recibe).
6- El Portador recibe el Comprobante.	**7-** La Recepcionista traslada el o los documentos hacia Correspondencia.
Flujo alterno de los eventos	
Acciones del actor	**Respuesta del proceso de negocio**

	2.1- La Recepcionista le informa que debe dirigirse a la Oficina de Correspondencia.
Poscondiciones	Se genera un nuevo documento donde se registran todos los datos de la información recibida.
Prioridad	**Crítica**

Tabla 4: Descripción textual del CUN "Entregar documentos en Recepción"

2.2.6.3. Diagramas de actividades de los Casos de Uso del Negocio

Un diagrama de actividad describe un proceso que explora el orden de las tareas o actividades que logran los objetivos del negocio. Constituyen un tipo especial de diagrama de estados que muestra el flujo de actividades dentro del negocio, donde se representan los actores, trabajadores y entidades del negocio, así como las actividades realizadas por ellos y sus relaciones. (Entorno Virtual de Aprendizaje, 2009 - 2010)

A continuación se muestra el diagrama de actividad correspondiente al CUN "Entregar documentos en Recepción". **(Figura 3: "Diagrama de actividades del CUN "Entregar documentos en Recepción")**

Para acceder a los restantes diagramas de actividades, consultar el documento Modelo del Negocio.

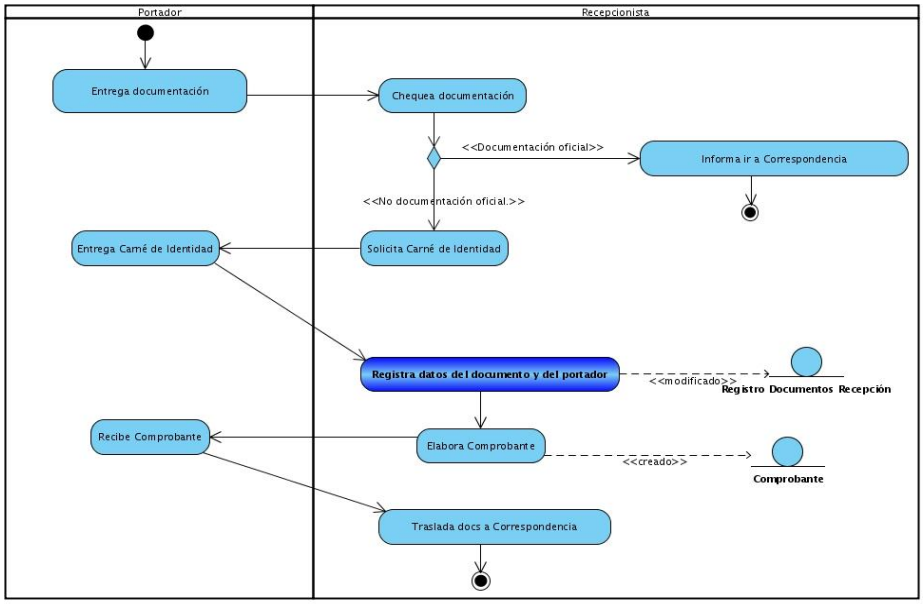


Figura 3: Diagrama de actividades del CUN "Entregar documentos en Recepción"

2.2.7. Modelo de Objetos

"Un modelo de objetos del negocio (…) describe (…) un conjunto de trabajadores que utilizan un conjunto de entidades del negocio (…) Una entidad del negocio representa algo, (…) que los trabajadores toman, inspeccionan, manipulan, producen o utilizan en un caso de uso del negocio." (Jacobson, y otros, 2000)

A continuación se muestra el modelo de objetos elaborado. **(Figura 4: Diagrama del modelo de objetos)**

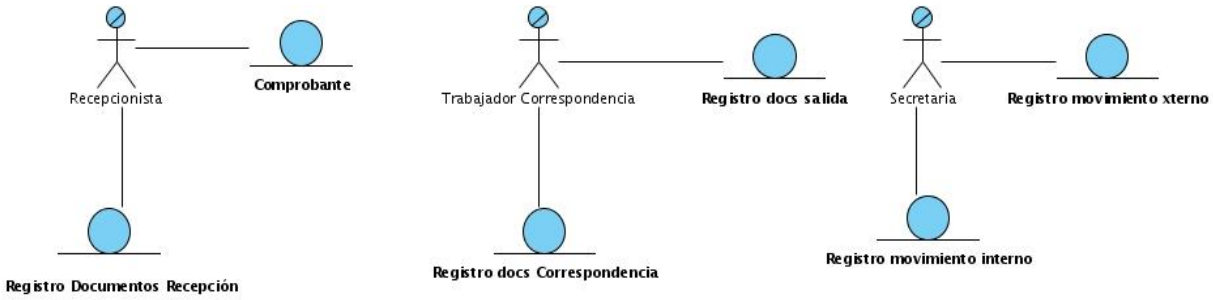


Figura 4: Diagrama del modelo de objetos

2.3. Requisitos del sistema

La captura de requisitos y la modelación del sistema son las actividades fundamentales que se desarrollan en este contenido.

¿Qué es un requerimiento?

La *IEEE Standard Glossary of Software Engineering Terminology* define un requerimiento como:

- Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
- Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
- Una representación documentada de una condición o capacidad como en 1 ó 2.

2.3.1. Requisitos funcionales

Un requisito funcional "especifica una acción que debe ser capaz de realizar el sistema." (Jacobson, y otros, 2000)

A continuación se especifican los requisitos funcionales con que debe cumplir la aplicación:

RF 1:Adicionar documentos.

El sistema debe permitir adicionar los datos de cualquier documento que llegue a la FGR.

RF 2: Enviar documento.

El sistema debe permitir realizar el envío de los datos correspondientes de los documentos que se registren a cualquier dirección, departamento o a la Oficina de Correspondencia.

RF 3: Registrar salida externa de documento.

El sistema debe permitir registrar los datos de los documentos que serán enviados fuera de la Fiscalía.

RF4: Buscar un documento específico.

El sistema debe permitir realizar búsquedas sobre cualquier documento que se encuentre registrado.

RF 5: Mostrar información del usuario.

El sistema debe permitir conocer a cada usuario informaciones como:

- Totalidad de documentos recibidos en el año, en el mes y en el día.
- Totalidad de documentos en el edificio.
- Totalidad de documentos en su poder.
- Los documentos pendientes por recibir.
- Los documentos que han sido enviados sin acuse de recibo.

2.3.2. Requisitos no funcionales

Un requisito no funcional "especifica propiedades del sistema". (Jacobson, y otros, 2000)

Estas propiedades constituyen las características que hacen al producto atractivo, usable, rápido y confiable. Pueden significar la diferencia entre un producto bien aceptado por los clientes y usuarios y otro de poca o ninguna calidad y aceptación. A continuación se especifican los requisitos no funcionales con que debe cumplir la aplicación:

RNF 1: Apariencia o interfaz externa.

- El diseño debe ser sencillo, amigable e intuitivo, permitiendo de esta manera una navegación rápida y fácil por parte del usuario.
- Debe ser adaptable a cualquier tipo de resolución de pantalla.

RNF 2: Usabilidad.

- El diseño de la aplicación debe ser concebido para un número limitado de usuarios, por lo que sólo podrán utilizarla las personas autorizadas.

RNF 3: Soporte.

- El usuario del módulo deberá recibir un adiestramiento previo en la utilización del sistema con el fin de que pueda explotar sus prestaciones sin contratiempos ocasionados por la falta de preparación técnica.

RNF 4: Portabilidad.

- Debe ser posible acceder a la aplicación desde cualquier sistema operativo con un navegador que cumpla los estándares.

RNF 5: Rendimiento.

- El sistema debe contar con un alto nivel de respuesta para los diferentes procesos que se realizan.

RNF 6: *Software*

- Para el desarrollo de la aplicación se utiliza el *framework* Symfony 1.3.
- En la máquina servidor debe estar instalado el sistema operativo GNU/Linux/Debian 4 Etch, Apache.
- En las máquinas clientes debe estar instalado una versión de Windows o de GNU/Linux.
- A la aplicación se podrá acceder desde cualquier navegador Web. Se recomienda Internet Explorer 6.0 o superior y Firefox 2.0 o superior.

RNF 7: Hardware

- Para garantizar el correcto funcionamiento de la aplicación se necesitan como requerimientos mínimos una computadora con un procesador Pentium II o superior, una memoria RAM de 512 MB o más, un disco duro de 10 GB o más y una tarjeta de red a 100 Mbps o más.

RNF 8: Restricciones de diseño e implementación.

- La lógica de presentación debe constituir una capa independiente de la lógica del negocio y de la capa de acceso a datos. Estas últimas también deben estar separadas entre sí.
- Para realizar el diseño del sistema debe emplearse la metodología RUP y el lenguaje de modelado UML. Para la representación de este último se emplea la herramienta Visual Paradigm para UML.
- La aplicación debe de estar implementada en el lenguaje de programación PHP.
- Se debe utilizar como Gestor de Base de Datos a PostgreSQL.

RNF 9: Seguridad.

- Identificar al usuario antes de que efectúe cualquier acción.
- Cada usuario contará con diferentes permisos en el sistema en correspondencia con el rol que ocupe.

2.4. Modelación del sistema

2.4.1. Descripción del sistema propuesto

Durante el desarrollo del Capítulo I, se analizaron diferentes SGD muy utilizados actualmente y se concluyó que ninguno de ellos era adaptable a las necesidades que presentaba la FGR, por lo que se hacía necesario el desarrollo de una nueva aplicación capaz de gestionar la entrada y salida de la correspondencia y documentación oficial de la Fiscalía, capaz de permitir mantener un control de esta documentación, así como conocer en todo momento su localización y el estado en que se encuentra.

La solución diseñada se compone de dos (2) partes o módulos fundamentales:

- **Módulo de Información:** Este módulo agrupa todas las acciones que proporcionan información al usuario sobre el estado de los documentos registrados en el sistema. Brinda pormenores concernientes a los documentos que tiene en su poder cada usuario en un instante dado y permite localizar documentos en cualquier parte del edificio.
- **Módulo de Documentos:** En este módulo están contenidas las acciones para insertar documentos en el sistema ya sea desde la Recepción o la Oficina de Correspondencia. Se incluye además la funcionalidad para registrar los datos de los documentos que se enviarán al exterior de la Fiscalía y para enviar documentos entre direcciones, departamentos y al exterior del edificio.

Se realizará además un mapeo de roles para asignar a cada rol participante únicamente las funcionalidades que le corresponden evitando así pérdidas de información e inseguridad de la misma, respetando siempre la disponibilidad de la información que a cada rol concierne. Por ejemplo: las secretarias sólo tendrán el acceso necesario a la información específica que se necesita registrar para realizar el movimiento interno o externo de un documento, y sólo tendrán visibles aquellos documentos que tenga en su poder la dirección o departamento donde trabaja, o los que tenga "pendiente por recibir".

2.4.2. Actores del sistema

"Los actores suelen corresponderse con trabajadores (o actores del negocio) en un negocio". (Jacobson, y otros, 2000).

En este caso se identifican los siguientes actores del sistema:

- Recepcionista.
- Trabajador de Correspondencia.
- Trabajador de departamento.

2.4.2.1. Descripción de los actores del sistema

A continuación se especifican las descripciones de cada uno de los actores del sistema. **(Tabla 5: Descripción de los actores del sistema)**

Actores del sistema	Justificación
Recepcionista	Encargado de recibir y registrar los documentos comunes que llegan a la Fiscalía por esta vía, así como de brindar cualquier información referente a este proceso que le sea solicitada.

Trabajador de Correspondencia	Encargado de recibir y registrar los documentos tanto comunes como oficiales que lleguen a la Fiscalía por esta vía, así como los provenientes de la Recepción y darles un destino dentro del edificio. También se encarga de tramitar los documentos destinados al exterior del edificio, así como de brindar cualquier información referente a este proceso que le sea solicitado.
Trabajador de departamento	Reúne a los actores Fiscal y Secretaria, como encargados de recibir todos los documentos que envíe la Oficina de Correspondencia y darle un nuevo destino dentro de la propia Institución o fuera de ella de ser necesario, así como de brindar cualquier información referente a este proceso que le sea solicitada.

Tabla 5: Descripción de los actores del sistema

2.4.3. Casos de uso del sistema

"Cada forma en que los actores usan el sistema representa un caso de uso. Los casos de uso son "fragmentos" de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. De manera más precisa, un caso de uso especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de la secuencia." (Jacobson, y otros, 2000)

Los casos de uso del sistema (CUS) son los siguientes:

CUS 1: Adicionar documento común.

CUS 2: Adicionar documento oficial.

CUS 3: Enviar documento.

CUS 4: Registrar salida externa de documento.

CUS 5: Mostrar información al usuario.

CUS 6: Buscar información.

CUS 7: Dar entrada a documentos.

A continuación se describe un breve resumen de cada uno de los CUS. **(Tabla 6: Resumen del CUS "Adicionar documento común", Tabla 7: Resumen del CUS "Adicionar documento oficial", Tabla 8: Resumen del CUS "Enviar documento", Tabla 9: Resumen del CUS "Registrar salida externa de documento", Tabla 10: Resumen del CUS "Mostrar información al usuario", Tabla 11: "Resumen del CUS "Buscar información", Tabla 12: Resumen del CUS "Dar entrada a documentos")**

CUS - 1	Adicionar documento común
Actor	Recepcionista, Trabajador de Correspondencia.
Descripción	El CU inicia cuando se necesitan registrar los datos sobre la llegada de un documento común a la FGR. Este CU finaliza cuando el documento se encuentra registrado en la cola "Entrada de Documentos".
Referencias	RF 1

Tabla 6: Resumen del CUS "Adicionar documento común"

CUS - 2	Adicionar documento oficial
Actor	Trabajador de Correspondencia.
Descripción	El CU inicia cuando el usuario Correspondencia necesita registrar los datos sobre la llegada de un documento a la FGR. Este CU finaliza cuando esta operación es realizada y el documento se encuentra registrado en la cola "Entrada de Documentos".
Referencias	RF 1

Tabla 7: Resumen del CUS "Adicionar documento oficial"

CUS - 3	Enviar documento
Actor	Trabajador de departamento
Descripción	El CU inicia cuando necesitan registrar los datos sobre envío interno o externo de un documento. Este CU finaliza cuando esta operación es realizada y los datos del documento son enviados a su nuevo destino.

Referencias	RF 2

Tabla 8: Resumen del CUS "Enviar documento"

CUS - 4	Registrar salida externa de documento
Actor	Trabajador de Correspondencia
Descripción	El CU inicia cuando se necesitan registrar los datos sobre la salida al exterior de la FGR de un documento. Este CU finaliza cuando esta operación es realizada.
Referencias	RF 3

Tabla 9: Resumen del CUS "Registrar salida externa de documento"

CUS - 5	Mostrar información al usuario
Actor	Secretaria, Recepcionista, Trabajador de Correspondencia, Trabajador de departamento.
Descripción	El CU consiste en que los usuarios acceden al módulo y la información es presentada en la interfaz principal.
Referencias	RF 5

Tabla 10: Resumen del CUS "Mostrar información al usuario"

CUS - 6	Buscar información
Actor	Secretaria, Recepcionista, Trabajador de Correspondencia, Trabajador de departamento.
Descripción	El CU inicia cuando algún usuario de los mencionados busca información específica sobre un documento que necesita conocer. Este CU finaliza cuando esta acción es concretada.
Referencias	RF 4

Tabla 11: Resumen del CUS "Buscar información"

CUS - 7	Dar entrada a documentos.

Actor	Trabajador de Correspondencia.
Descripción	El CU inicia cuando el usuario necesita enviar la información sobre los documentos que se han recibido a los distintos departamentos dentro de la Fiscalía. Este CU finaliza cuando la acción es concretada.
Referencias	RF 1, RF 2.

Tabla 12: Resumen del CUS "Dar entrada a documentos"

2.4.3.1. Diagrama de casos de uso del sistema

A continuación se muestra el diagrama de casos de uso del sistema, donde puede observarse la interrelación entre los actores identificados y los CUS definidos". **(Figura 5: "Diagrama de casos de uso del sistema")**

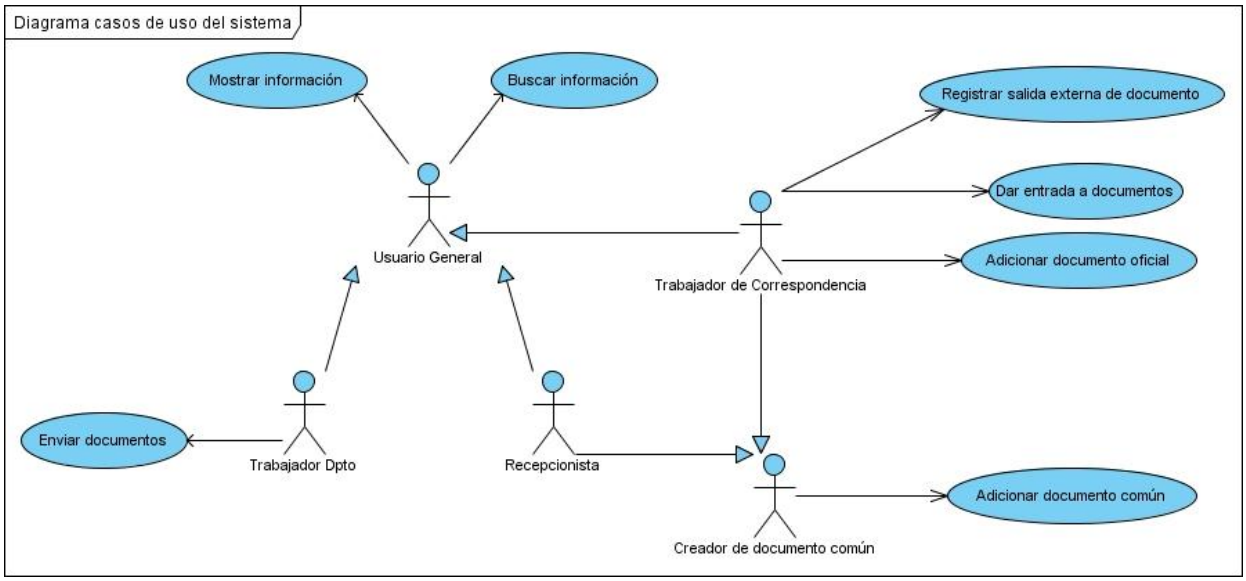


Figura 5: Diagrama de casos de uso del sistema

2.4.3.2. Descripción de los casos de uso del sistema

A continuación se muestra la descripción del CUS "Adicionar documento común". **(Tabla 13: Descripción del CUS "Adicionar documento común")**

Para acceder a la descripción íntegra de todos los CUS, consultar el documento Modelo de casos de uso del sistema.

Caso de Uso del Sistema	**Adicionar documento común**
Actores	**Recepcionista, Trabajador de Correspondencia.**
Propósito	Permitir almacenar todos los datos requeridos referentes a los documentos comunes que lleguen a la FGR.
Resumen	El CU inicia cuando se necesitan registrar los datos sobre la llegada de un documento común a la FGR. Este CU finaliza cuando el documento se encuentra registrado en la cola "Entrada de Documentos".
Precondiciones	El usuario Correspondencia debe autenticarse antes de realizar acciones en el sistema.
Referencias	**RF 1**
Prioridad	Crítico
Flujo normal de eventos	
Acciones del actor	**Respuesta del sistema**
1- Los usuarios acceden al sistema y seleccionan "Nuevo Documento Común".	**2-** El sistema muestra una interfaz con todos los datos que deben ser insertados: Datos a insertar: **Documento:** • No. Referencia. • Observaciones. • Cantidad de hojas. • Provincia. **Portador:** • Nombre. • Primer Apellido.

	• Segundo Apellido. • No. Carnet. • Dirección particular. **Remitente:** • Nombre. • 1er Apellido. • 2do Apellido.
3- El usuario inserta todos los datos requeridos. **4-** Pulsa la opción "Guardar". **Ver Flujo alterno 3.1** (si los datos son incorrectos o están incompletos).	**5-** El sistema guarda los cambios de la acción realizada. **6-** Se muestra un mensaje informando que la operación fue realizada con éxito. *Finaliza el CUS.*

Flujo alterno de eventos	
Acciones del actor	**Respuesta del sistema**
	3.1- El sistema muestra una notificación de que los datos introducidos son incorrectos o están incompletos y da la opción para que vuelvan a ser introducidos mostrando nuevamente los campos vacíos.

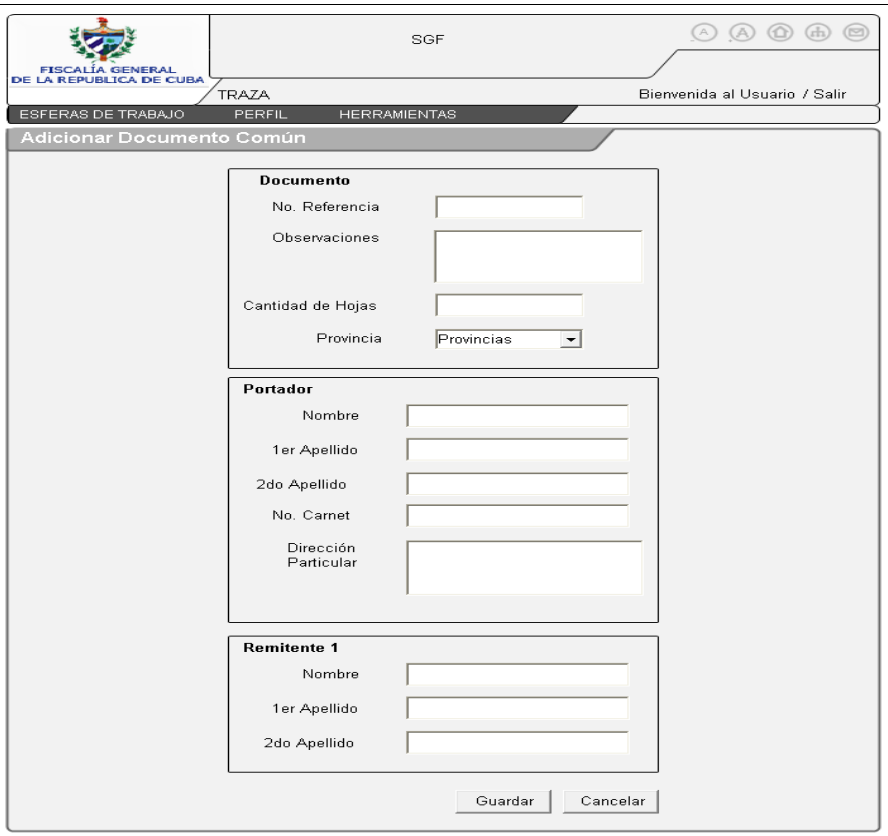
Prototipo de Interfaz de Usuario	
Poscondiciones	El sistema debe permitir guardar los cambios que se realicen con respecto a la gestión de la entrada de documentos.

Tabla 13: Descripción del CUS "Adicionar documento común"

2.5. Conclusiones parciales

Los artefactos generados en este capítulo son de gran importancia, ya que cada uno de ellos representa una parte esencial para alcanzar un entendimiento mejor y más profundo sobre el *software* a desarrollar, en particular:

- El modelo de negocio permitió recoger todas las necesidades de los clientes, lograr un mejor entendimiento de los procesos que son llevados a cabo y ganar en claridad sobre las funcionalidades a automatizar.
- El modelo del sistema garantizó una comprensión exacta de lo que se desea desarrollar, y por tanto, un mayor entendimiento y un acuerdo común entre clientes y desarrolladores.

CAPÍTULO 3. DISEÑO DEL SISTEMA

3.1. Introducción

En el siguiente capítulo se describe el diseño del sistema. Se obtienen consecuentemente los diagramas de clases del diseño web, el de las clases persistentes y el Modelo de datos.

3.2. Flujo de Análisis y Diseño

El objetivo principal de esta disciplina es transformar los requerimientos a una especificación que describa cómo implementar el sistema. El análisis fundamentalmente consiste en obtener una visión que se preocupa de ver qué hace el sistema de *software* a desarrollar, por tal motivo este se interesa en los requerimientos funcionales. Por otro lado, el diseño es un refinamiento que toma en cuenta los requerimientos no funcionales, por lo cual se centra en cómo el sistema cumple sus objetivos.

El análisis se encarga de representar una vista interna del sistema en la cual usando el lenguaje de los desarrolladores se refina los requisitos y se estructuran en base a clases y paquetes. Luego este proceso continúa en el diseño hasta obtener los objetos lógicos que interactúan y darán cumplimiento a los requerimientos funcionales y no funcionales planteados.

3.3. Análisis del sistema

“Flujo de trabajo cuyo propósito es analizar los requisitos descritos en la captura de requisitos, mediante su refinamiento y estructuración. El objetivo de esto es lograr una comprensión más precisa de los requisitos y obtener una descripción de los requisitos que sea fácil de mantener y que nos ayude a dar estructura al sistema en su conjunto.” (Jacobson, y otros, 2000)

3.3.1. Modelo del análisis

El análisis se basa en un modelo de objetos conceptual al cual se le llama Modelo de Análisis. El modelo de análisis ayuda a refinar los requisitos y permite razonar sobre los aspectos internos del sistema. Ayuda también a estructurar los requisitos ya que proporciona una estructura centrada en el mantenimiento, en aspectos tales como la flexibilidad ante los cambios y la reutilización. (Jacobson, y otros, 2000)

El modelo del análisis es un artefacto que usualmente se genera para entender mejor los requisitos y realizar mejor el diseño. Es típicamente un artefacto temporal, que brinda una aproximación al diseño y se realiza normalmente cuando no se tiene una idea clara de los procesos y requerimientos. (Jacobson, y otros, 2000)

Sobre la base de los planteamientos anteriores se decide no realizar el análisis del sistema propuesto, pues los requisitos son claros y se encuentran muy bien especificados.

3.4. Diseño del sistema

"Flujo de trabajo fundamental cuyo propósito fundamental es formular modelos que se centran en los requisitos no funcionales y el dominio de la solución y que prepara para la implementación y pruebas del sistema." (Jacobson, y otros, 2000)

3.4.1. Modelo del diseño

"El modelo de diseño se puede utilizar para visualizar la implementación y para soportar las técnicas de programación gráfica" (…) es un modelo de objetos que describe la realización física de los casos de uso, centrándose en como los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. El modelo de diseño sirve de abstracción a la implementación y se utiliza como una entrada fundamental de las actividades de implementación". (Jacobson, y otros, 2000)

3.4.2. Patrones de arquitectura y diseño

3.4.2.1. Patrón Arquitectónico MVC (Modelo – Vista – Controlador)

Durante el desarrollo del Capítulo 1 se explica que para el desarrollo de la aplicación propuesta se utilizará el *framework* Symfony, "que está basado en un patrón clásico del diseño web conocido como arquitectura MVC, formado por 3 niveles: Modelo, Vista y Controlador. **(Figura 6: Patrón Modelo – Vista – Controlador)**

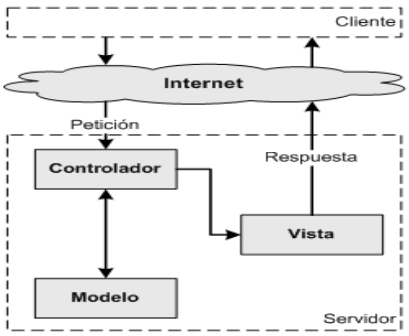


Figura 6: Patrón Modelo - Vista – Controlador

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. El controlador se encarga de aislar al modelo

y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etcétera.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes.

El uso de este patrón por Symfony resulta bastante útil además de restrictivo ya que obliga a dividir y organizar el código de acuerdo a las convenciones establecidas por el mismo. El código de la presentación se guarda en la vista, el código de manipulación de datos se guarda en el modelo y la lógica de procesamiento de las peticiones constituye el controlador.

"La implementación que realiza Symfony de la arquitectura MVC incluye varias clases como son:

- **sfController:** es la clase del controlador. Se encarga de decodificar la petición y transferirla a la acción correspondiente.
- **sfRequest:** almacena todos los elementos que forman la petición (parámetros, *cookies*, cabeceras).
- **sfResponse:** contiene las cabeceras de la respuesta y los contenidos. El contenido de este objeto se transforma en la respuesta HTML que se envía al usuario.
- El ***singleton*** de contexto (que se obtiene mediante sfContext::getInstance ()) almacena una referencia a todos los objetos que forman el núcleo de Symfony y puede ser accedido desde cualquier punto de la aplicación.

Symfony toma lo mejor de la arquitectura MVC y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo." (Eguíluz)

El controlador frontal es un componente generado automáticamente por el *framework* que, unido con el *layout* son comunes para todas las páginas de la aplicación. Las clases de la capa del modelo son generadas automáticamente en función de la estructura de datos de la aplicación por la librería Propel y la abstracción de la base de datos es completamente invisible al programador, ya que es realizada mediante un componente llamado Creole. **(Figura 7: Patrón MVC en Symfony)**

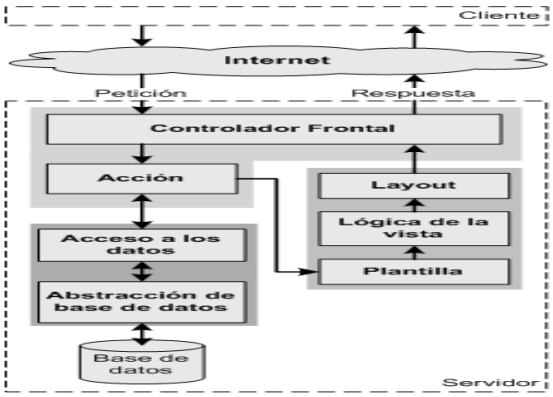


Figura 7: Patrón MVC en Symfony

3.4.2.2. Aplicación de los patrones GRASP en Symfony

En la implementación con Symfony se utilizan numerosos patrones GRASP, seguidamente se mencionan algunos ejemplos, ubicándolos en las capas de Modelo y Controlador que plantea el patrón arquitectónico MVC.

Patrón "Experto": En el modelo de la arquitectura Symfony existen dos tipos de clases que son fundamentales:

- Las encargadas de la abstracción de datos (realizan todas las operaciones con la base de datos).
- Las de acceso a datos (interactúan con las clases de abstracción de datos y devuelven los objetos necesarios por los controladores).

Por cada tabla Symfony genera 4 clases: Clase, ClasePeer, BaseClase y BaseClasePeer. Las clases a las que el *framework* añade el sufijo Peer trabajan directamente con la base de datos y por lo tanto son las encargadas de la abstracción utilizando Propel, en ellas se encuentran los atributos necesarios para este proceso, de ahí la necesidad de que implementen la responsabilidad de efectuar las operaciones con la base de datos, aplicando de esta manera el patrón Experto.

Patrón "Creador": Con Symfony se pueden crear objetos de varias formas, mayormente mediante métodos estáticos.

Patrón "Bajo Acoplamiento": Las clases que implementan la lógica de negocio y de acceso a datos se encuentran en el Modelo, estas clases no tienen asociaciones con las de la Vista o el Controlador por lo que la dependencia en este caso es baja, demostrándose de esta manera el uso de este patrón.

Patrón "Alta Cohesión": El trabajar con Symfony permite la organización del trabajo en cuanto a la estructura del proyecto, esto proporciona crear y trabajar con clases con una alta cohesión. Ejemplo de esto se evidencia en las clases *Actions,* la cual está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas, teniendo un sentido común y un propósito único, siendo las mismas las encargadas de controlar las acciones de las plantillas.

Patrón "Controlador": El controlador se encarga de asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas con las que mantiene un modelo de alta cohesión, esto facilita la centralización de actividades (validaciones, seguridad.). Un ejemplo del patrón controlador se puede ver desde la clase sfFrontController, sfWebFrontController, sfContex, los *actions*, y el index.php del ambiente.

Patrón "*Front Controller*" (Controlador frontal): Symfony implementa este patrón, en el cual existen varias clases controladoras que forman un flujo para atender las peticiones del usuario y de otras clases. La arquitectura del *framework* (MVC) ayuda desde el principio, existiendo una capa específicamente para los controladores, que son el núcleo del mismo, el puesto de mando.

3.4.2.3. Aplicación de ejemplos de patrones Gof en Symfony

Patrón "*Decorator*": Este método permite añadir funcionalidades a las vistas dinámicamente. El archivo llamado layout.php contiene el *layout* de la página, este archivo se le denomina plantilla global, la cual almacena el código HTML que es común a todas las páginas de la aplicación. El contenido de la plantilla se integra en el *layout*. **(Figura 8: "Plantilla decorada con *layout*")**

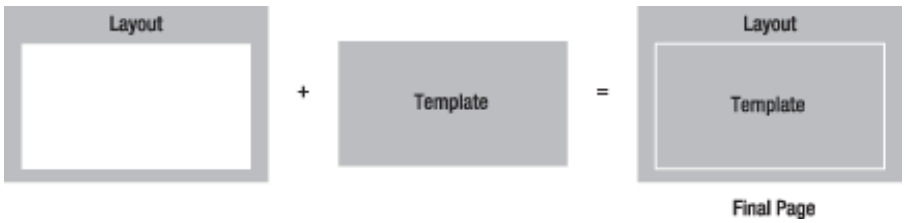


Figura 8: Método "Plantilla decorada con layout"

Patrón "*Singleton*": La clase sfRouting es una de es una de las que se encuentran en la capa Controlador del *framework*. Esta clase es muy utilizada porque es la encargada de enrutar todas las peticiones que se hagan a la aplicación. El ejemplo tiene solamente un punto de creación, el cual es estático, lo que permite entonces la aplicación perfecta del patrón. **(Figura 9: "Método *getInstance* de la clase *sfRouting*")**

```
public static function getInstance ()
  {
    if (!isset (self::$instance))
    {
      self::$instance = new sfRouting();
    }

    return self::$instance;
  }
```

Figura 9: Método "getInstance" de la clase "sfRouting"

Patrón "*Command*": Se evidencia en la clase sfFrontWebController. Esta clase es la encargada de determinar cuál módulo y cuál acción deben responder a las solicitudes de los usuarios. **(Figura 10: "Muestra de la clase sfFrontWebController")**

```
class sfFrontWebController extends sfWebController
{
  /**...
  public function dispatch()
  {
    try
    {
      if (sfConfig::get('sf_logging_enabled'))
      {
        $this->getContext()->getLogger()->info('{sfController} dispatch request
      }

      // reinitialize filters (needed for unit and functional tests)
      sfFilter::$filterCalled = array();

      // determine our module and action
      $request    = $this->getContext()->getRequest();
      $moduleName = $request->getParameter('module');
      $actionName = $request->getParameter('action');

      // make the first request
      $this->forward($moduleName, $actionName);
    }
```

Figura 10: Muestra de la clase "sfFrontWebController"

Patrón "*Registry*": Patrón sumamente útil, es un medio simple y eficiente de compartir datos y objetos sin tener que preocuparse de mantener numerosos parámetros o hacer uso de variables globales. La aplicación de este patrón se observa en la clase sfConfig, clase encargada de almacenar todas las variables de uso global en la aplicación. En el ejemplo se muestra el código de la clase sfConfig donde se crean los métodos globales y cómo el archivo config de la aplicación utiliza los métodos globales definidos en esta. **(Figura 11: "Muestra de la clase sfConfig")**

```
class sfConfig
{
  protected static
    $config = array();

  /**...
  public static function get($name, $default = null)
  {...}

  /**...
  public static function has($name)
  {...}

  /**...
  public static function set($name, $value)
  {...}

  /**...
  public static function add($parameters = array())
  {...}

  /**...
  public static function getAll()
  {...}

  /**...
  public static function clear()
  {...}
}
```

Figura 11: Muestra de la clase "sfConfig"

Patrón "*Front Controller*": La aplicación de este patrón se manifiesta en la capa de Controladores, donde existe una estructura de clases bien definida. Una de sus principales funciones es definir un solo punto de entrada para las peticiones de los usuarios, permitiendo un mejor control del flujo de eventos del sistema. La clase sfController se encarga de decodificar la petición y transferirla a la acción correspondiente.

3.4.3. Clases del diseño

"Una clase del diseño es una abstracción sin costuras de una clase o construcción similar en la implementación del sistema (…) Una clase de diseño y sus objetos participan en varias realizaciones de casos de uso. También puede suceder que algunas operaciones, atributos y asociaciones sobre una clase específica sean solo relevantes para una sola realización de caso de uso. Para manejar todo esto se utilizan los diagramas de clases conectados a una realización de caso de uso, mostrando así sus clases participantes, subsistemas y sus relaciones. (Jacobson, y otros, 2000)

Los diagramas de clases del diseño fueron confeccionados de acuerdo a los módulos diseñados (los cuales fueron explicados en el capítulo 2).

A continuación se muestra el diagrama correspondiente al Módulo de Documentos. **(Figura 12: Diagrama de clases del diseño del "Módulo de Documentos")**

Para examinar el diagrama de clases del diseño correspondiente al Módulo de Información, consultar el documento Modelo del negocio.

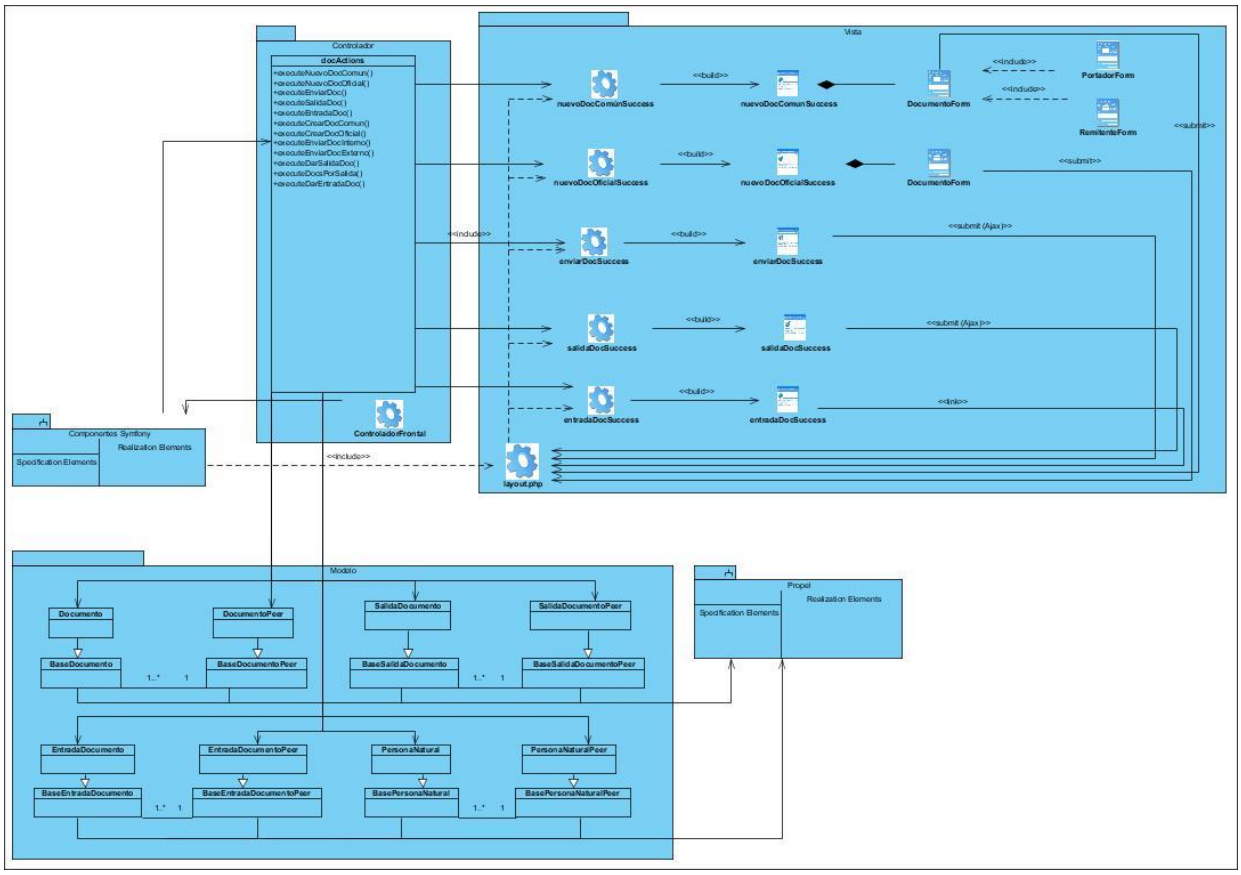


Figura 12: Diagrama de clases del diseño del "Módulo de Documentos"

3.4.3.1. Descripción de las clases del diseño

Debido a que la solución informática que se plantea es una aplicación Web se utilizan los siguientes estereotipos web para la representación de las clases del diseño:

Extensiones para diseño web:

Server Page: Representa la página Web que tiene código que se ejecuta en el servidor.

***Client Page*:** Una instancia de una página Cliente es una página Web con formato HTML.

***Form*:** Colección de elementos de entrada que son parte de una página Cliente.

<<*Build*>>: Representa una asociación especial que relaciona las páginas clientes con las páginas servidoras.

<<*Link*>>: Expresa las asociaciones más comunes entre las páginas, en este caso la de un hipervínculo.

<<*Submit*>>: Es la relación que se crea siempre entre una página servidora y un formulario.

A continuación se describen cada uno de los paquetes y subsistemas de cada uno de los diagramas de clases del diseño del sistema:

Paquete controlador: Una parte importante de su trabajo es común a todos los controladores de la aplicación. Entre las tareas comunes se encuentran el manejo de las peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación y otras tareas similares. Por este motivo, el Controlador se ha dividido en un Controlador frontal, que se encarga de realizar las tareas comunes y las acciones (actions.php), que incluyen el código específico del controlador de cada página. Habrá un actions.php por cada uno de los módulos pero el controlador frontal será el mismo para todo el sistema.

Paquete vista: Las páginas web suelen contener elementos que se muestran de forma idéntica a lo largo de toda la aplicación: cabeceras de la página, el *layout* genérico, el pie de página y la navegación global. En la mayor parte de las veces sólo cambia el interior de la página. Por este motivo, la vista se separa en un *layout* y en una plantilla. El *layout* será global en toda la aplicación o a la mayoría de las páginas. La plantilla sólo se encarga de visualizar las variables definidas en el paquete del controlador.

Paquete del modelo: Solamente contiene las clases encargadas del acceso a los datos almacenados en el gestor de base de datos, las cuales utilizan el ORM (*Object Relational Model*) Propel para el acceso a los mismos.

Las clases y subsistemas que se relacionan a continuación son comunes para cada módulo y debido a que son producto de cada proyecto de Symfony se describen brevemente para una mejor comprensión.

Clases Base: Las clases con nombre Base del directorio lib/model/om/ son las que se generan directamente a partir del esquema. Nunca se deberían modificar esas clases, porque cada vez que se genera el modelo, se borran todas las clases.

Clases de Objetos: Estas clases heredan todos los métodos de la clase Base, pero no les afectan las modificaciones en el esquema. Este mecanismo de clases personalizadas permite empezar a programar

desde el primer momento, sin ni siquiera conocer el modelo relacional definitivo de la base de datos. La estructura de archivos creada permite personalizar y evolucionar el modelo.

Clases Peer: Son las clases que tienen métodos estáticos para trabajar con las tablas de la base de datos. Proporcionan los medios necesarios para obtener los registros de las tablas. Sus métodos devuelven normalmente un objeto o una colección de objetos de la clase objeto relacionada.

Las clases **BaseClase** y **BaseClasePeer**, tienen una relación de asociación sin navegabilidad porque ambas pueden crear instancias una de la otra. La multiplicidad expresa que una instancia de la clase BaseClasePeer puede crear 1 o varias instancias de la clase BaseClase cuando se ejecuta. La combinación de las clases objeto y las clases "*peer*" y las versiones básicas y personalizadas de cada una hace que se generen 4 clases por cada tabla del esquema.

Subsistema Propel: Su función es gestionar el acceso a la base de datos y gestionar el modelo del sistema. Implica que el acceso y la modificación de los datos almacenados en la base de datos se realicen mediante objetos, nunca de forma explícita, permitiendo un alto nivel de abstracción y fácil portabilidad. Propel tiene incluido tareas para generar automáticamente las sentencias SQL necesarias para crear las tablas de la base de datos. La librería Propel se encarga de esta generación automática, ya que crea el esqueleto o estructura básica de las clases y genera automáticamente el código necesario. La abstracción de la base de datos es completamente transparente para el programador, ya que se realiza de forma nativa mediante PDO (*PHP Data Objects*).

Subsistema Componentes Symfony: Representa todas las clases del *framework* Symfony que serán utilizadas durante el funcionamiento del sistema. Dígase validadores de formularios, formularios, plantillas, componentes de seguridad, etc.

Seguidamente se muestra la descripción de la clase del diseño que contiene la lógica del negocio del módulo “Documento”: **(Tabla 14: Descripción de la clase “docActions”)**

Para examinar la descripción de clases del diseño correspondiente al Módulo Información y las restantes descripciones del Módulo Documentos, consultar el documento Modelo del negocio.

Nombre de la clase: docActions
Tipo: Controladora
Descripción: Esta clase contiene los métodos del módulo “Documentos”, permitiendo adicionar, enviar o registrar la salida de un documento.
Nombre del método: executeNuevoDocComun()
Descripción: Este método permite adicionar un nuevo documento con características comunes.
Nombre del método: executeNuevoDocOficial()

Descripción: Este método permite adicionar un nuevo documento con características oficiales.
Nombre del método: executeEnviarDoc()
Descripción: Este método permite realizar el envío de los datos de los documentos que serán enviados entre los departamentos y hacia Correspondencia (si tiene un destino externo).
Nombre del método: executeSalidaDoc()
Descripción: Este método permite registrar los datos de los documentos con un destino exterior a la FGR.
Nombre del método: executeEntradaDoc()
Descripción: Este método permite enviar los datos de los documentos que han sido registrados hacia su destino dentro de la Fiscalía.
Nombre del método: executeCrearDocComun()
Descripción: Acción que permite crear la plantilla para adicionar un nuevo documento común al sistema.
Nombre del método: executeCrearDocOficial()
Descripción: Acción que permite crear la plantilla para adicionar un nuevo documento oficial al sistema.
Nombre del método: executeEnviarDocInterno()
Descripción: Esta acción permite realizar el envío de los datos de los documentos que serán enviados internamente.
Nombre del método: executeEnviarDocExterno()
Descripción: Esta acción permite realizar el envío de los datos de los documentos que serán enviados hacia el exterior.
Nombre del método: executeDarSalidaDoc()
Descripción: Esta acción cambia el estado a un documento por salir, dándole salida y guardando los datos pertinentes al envío.
Nombre del método: executeDocPorSalida()
Descripción: Esta acción muestra una lista con los documentos por salir.
Nombre del método: executeDarEntradaDoc()
Descripción: Esta acción cambia el estado de un documento dándole entrada en el sistema.

Tabla 14: Descripción de la clase "docActions"

3.4.3.2. Clases persistentes

"La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Las clases persistentes referencian directamente las entidades lógicas y sus atributos." (Entorno Virtual de Aprendizaje, 2009 - 2010)

Para definir las clases persistentes del sistema se aplicaron las siguientes reglas:

- Cuando una clase que está formada por otras clases es persistente, automáticamente las clases componentes también son persistentes. Lo contrario no se cumple necesariamente.

- Cuando una clase hija de una jerarquía es persistente, automáticamente son persistentes sus ancestros en el árbol de jerarquía. Lo contrario no se cumple necesariamente.
- Cuando hay herencia múltiple, esta debe ser resuelta antes si el medio de almacenamiento a utilizar no soporta este concepto. La solución más factible es que la clase hija herede de la clase, de la que redefine sus métodos y añada un atributo pasivo del tipo de la otra clase de la que heredaba. Si se redefine comportamiento de más de una clase padre, hay que escoger de cuál se quedaría heredando y añadir un atributo pasivo por cada una de las clases padres de las que no hereda. Los métodos que se redefinen que ya no se reciben por herencia, en su implementación incluirán las relaciones con las clases padres.

3.4.3.2.1. Diagrama de clases persistentes

El siguiente diagrama de clases persistentes es una parte del diagrama de clases del proyecto SGF ya que la aplicación propuesta constituye un módulo de dicho proyecto. **(Figura 13: "Diagrama de clases persistentes")**

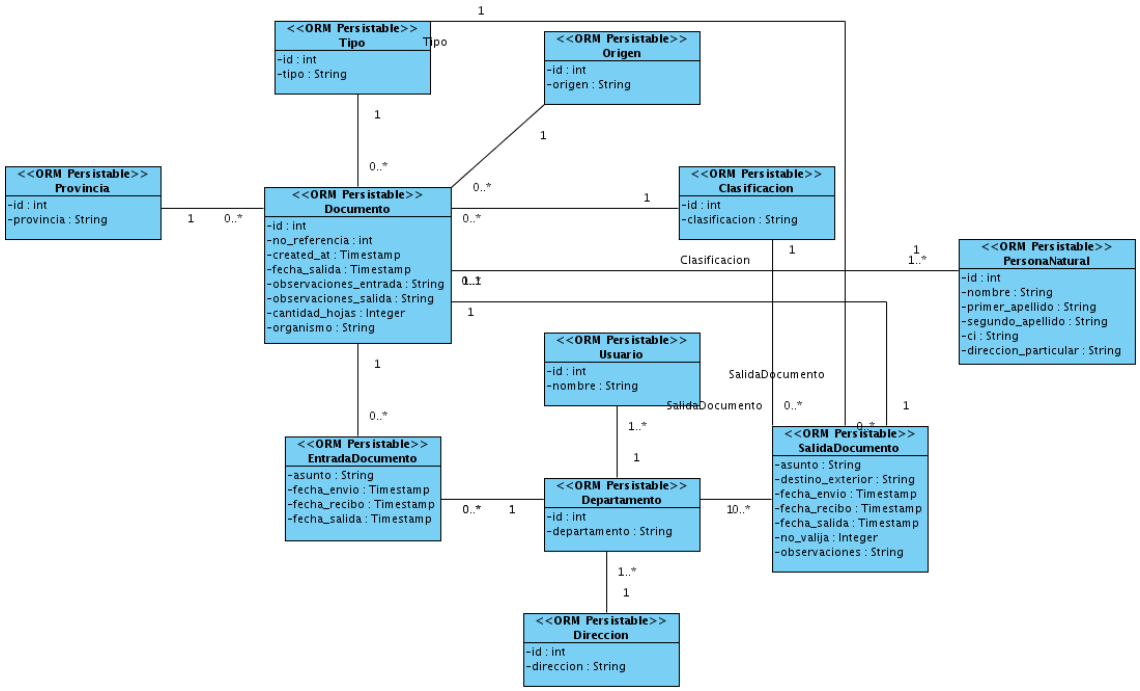


Figura 13: Diagrama de clases persistentes

3.4.3.3. Modelo de datos

"El modelo de datos describe la representación lógica y física de la persistencia de los datos utilizados por la aplicación". (Entorno Virtual de Aprendizaje, 2009 - 2010)

El modelo de datos de la aplicación está compuesto por las entidades que pasarán a ser las tablas de la base de datos que será utilizada por el sistema. **(Figura 14: Modelo de Datos)**

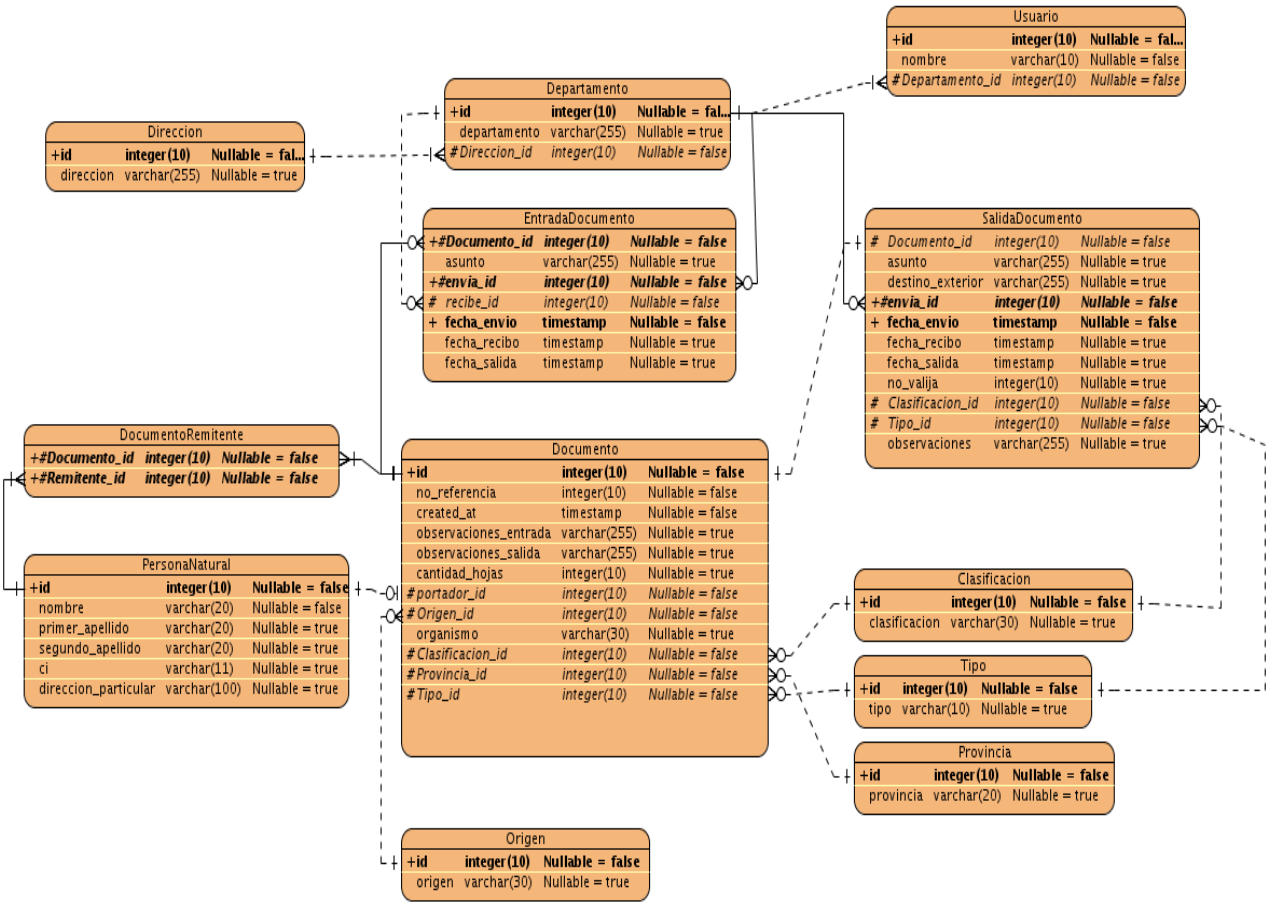


Figura 14: Modelo de datos

3.4.3.3.1. Descripción de las tablas

A continuación se muestran las descripciones de la tabla **tb_persona_natural** y **tb_usuario,** del modelo de datos. **(Tabla 15: Descripción de la tabla "tb_persona_natural", Tabla 16: Descripción de la tabla "tb_usuario")**

Para acceder el resto de las descripciones de las tablas que componen el Modelo de datos, consultar documento Modelo de Datos.

Nombre de la clase: PersonaNatural
Tipo: Entidad
Descripción: Tabla que almacena los datos relacionados con una persona externa.

Atributo	Tipo	Descripción
id	int	Identificador de la tabla “tb_persona_natural”.
nombre	varchar	Nombre de la persona.
primer_apellido	varchar	Primer Apellido.
segundo_apellido	varchar	Segundo Apellido.
ci	varchar	Carnet de Identidad.
dirección_particular	varchar	Dirección particular.

Tabla 15: Descripción de la tabla “tb_persona_natural”

Nombre	tb_usuario	
Descripción	Tabla que almacena los datos de los usuarios del sistema.	
Atributo	**Tipo**	**Descripción**
id	int	Identificador de la tabla.
nombre	varchar	Nombre con que se identifica el usuario.
departamento_id	int	Identificador de la tabla “tb_departamento” (Llave foránea).

Tabla 16: Descripción de la tabla “tb_usuario”

3.5. Conclusiones parciales

Los artefactos obtenidos en este capítulo son muy necesarios y de vital importancia para la construcción del sistema ya que posibilitan definirlo con suficiente detalle como para permitir su interpretación y realización física. En particular:

- El Modelo del diseño permitió materializar con precisión los requerimientos del cliente y sirvió como guía para las actividades de implementación, al producir una representación técnica del *software* que será desarrollado.
- El Modelo de datos permitió definir correctamente la estructura de datos para dar soporte a la información del módulo.

CAPÍTULO 4. IMPLEMENTACIÓN Y PRUEBA

4.1. Introducción

En el siguiente capítulo se obtienen los principales artefactos correspondientes a los flujos de trabajo de Implementación y Prueba. Se obtienen los diagramas de componentes como principal artefacto del Modelo de implementación. Se describen los tipos de prueba a los que será sometido el sistema y se detallan los casos de prueba que se realizarán.

4.2. Flujo de implementación

"En la implementación empezamos con el resultado del diseño e implementamos el sistema en términos de componentes, es decir, ficheros de código fuente, *scripts*, ficheros de código binario, ejecutables y similares." (Jacobson, y otros, 2000)

4.2.1. Modelo de implementación

"El modelo de implementación describe cómo los elementos del diseño, como las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc. (…) 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 o lenguajes de programación utilizados (…) es la entrada principal de las etapas de prueba que siguen a la implementación." (Jacobson, y otros, 2000)

4.2.1.1. Diagramas de componentes

Un diagrama de componentes "muestra un conjunto de componentes y sus relaciones; los diagramas de componentes muestran los componentes de un sistema desde un punto de vista estático." (Jacobson, y otros, 2000)

A continuación se muestran los diagramas de componentes (DC) correspondientes a la aplicación global y al Módulo de Documentos. **(Figura 15: "DC global", Figura 16: "DC del Módulo Documentos", Figura 17: "DC Vista del Módulo Documentos", Figura 18: "DC Controlador Módulo de Documentos", Figura 19: "Modelo Módulo Documentos")**

Para examinar los restantes diagramas de componentes, consultar el documento Modelo de Implementación.

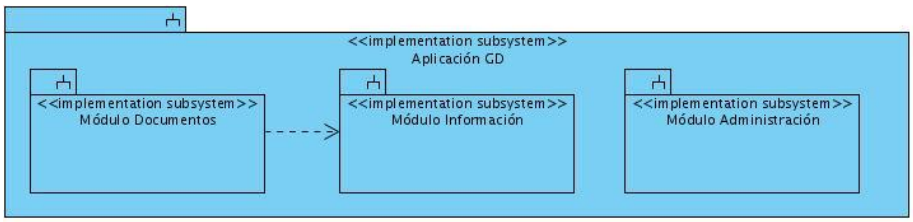


Figura 15: DC global

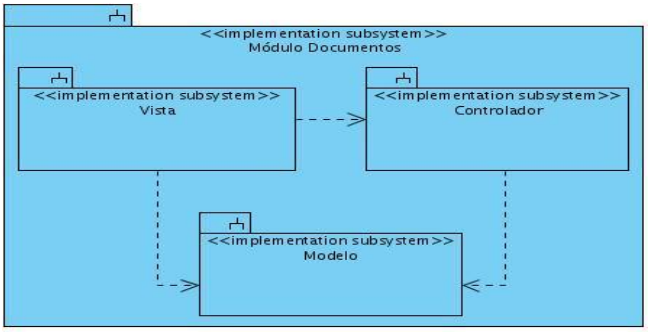


Figura 16: DC del Módulo Documentos

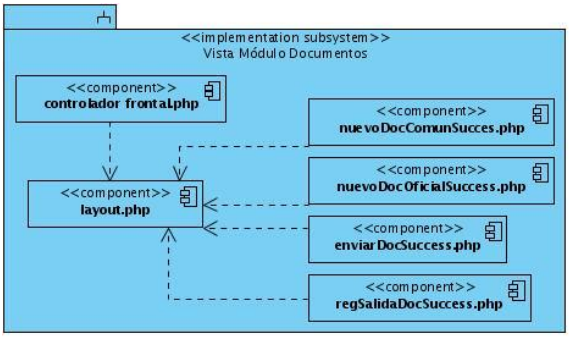


Figura 17: DC Vista del Módulo Documentos

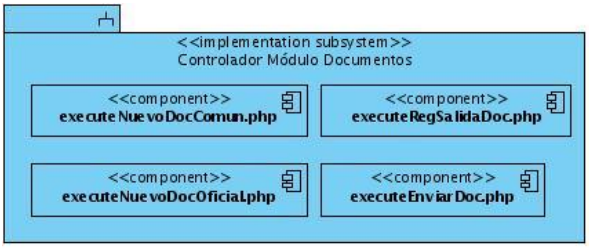


Figura 18: DC Controlador Módulo Documentos

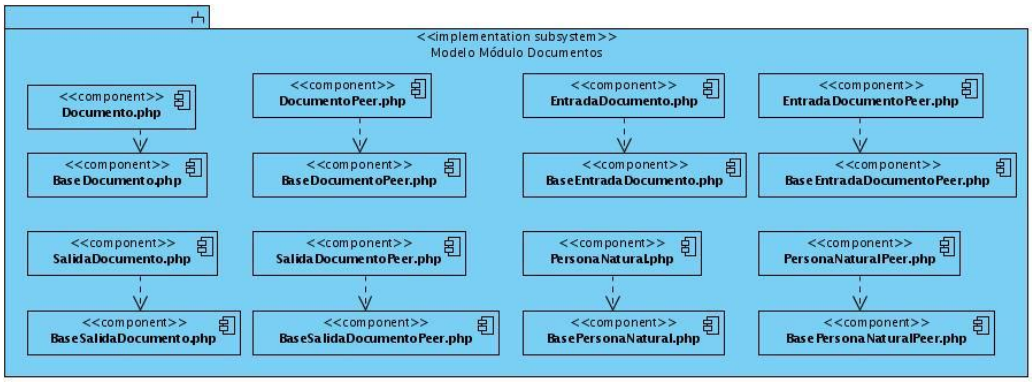


Figura 19: DC Modelo Módulo Documentos

4.2.1.2. Interfaces del sistema

Para la realización de la aplicación se desarrollaron interfaces amigables y seguras, acorde con el personal que trabajará con la misma. A continuación se presenta la interfaz principal de la aplicación. En esta se encuentran los vínculos a todas las funciones con que debe cumplir la misma. **(Figura 20: "Interfaz principal de la aplicación")**

Figura 20: Interfaz principal de la aplicación

4.2.2. Modelo de despliegue

"El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo.

Podemos observar lo siguiente sobre el modelo de despliegue:

- Cada nodo 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." (Jacobson, y otros, 2000)

A continuación se muestra el modelo de despliegue correspondiente a la aplicación. **(Figura 21: "Modelo de despliegue")**

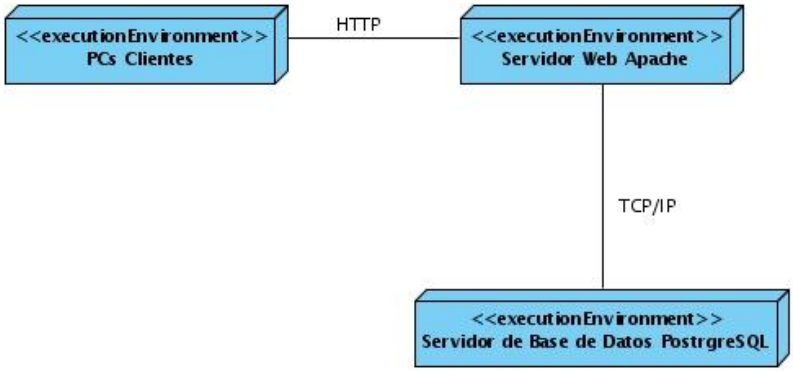


Figura 21: Diagrama de despliegue

4.2.2.1. Descripción de los nodos físicos

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." (Jacobson, y otros, 2000)

Los nodos que componen el modelo de despliegue correspondiente a la aplicación son los siguientes:

- **Nodo PC – Cliente:** se encontrará el sistema operativo Windows o Linux y los navegadores web Mozilla Firefox o Internet Explorer mediante los cuales los clientes tendrán acceso al sistema.
- **Nodo Servidor Web Apache:** se encontrará todo lo relacionado con la aplicación. Estarán agrupados los archivos a través de los cuales el usuario logra acceder al sistema, además se encuentra contenida toda la información específica de cada registro, sus clases; así como almacena además la configuración general del sistema, las clases y librerías externas y los *plugins* de instalación de la aplicación.
- **Nodo Servidor BD PostgreSql:** serán almacenados los datos de la aplicación, se guardará el modelo de objetos del sistema.

4.3. Flujo de prueba

"En el flujo de trabajo de la prueba verificamos el resultado de la implementación probando cada construcción (…), así como las versiones finales del sistema a ser entregadas a terceros." (Jacobson, y otros, 2000)

4.3.1. Modelo de prueba

"El modelo de pruebas describe principalmente cómo se prueban los componentes ejecutables (como las construcciones) en el modelo de implementación con pruebas de integración y de sistema (…) puede describir también cómo han de ser probados aspectos específicos del sistema; por ejemplo, si la interfaz de usuario es utilizable y consistente." (Jacobson, y otros, 2000)

Cualquier producto de ingeniería puede ser probado mediante una de estas técnicas:

- **Pruebas de caja negra:** Conociendo la funcionalidad específica para la cual fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa.
- **Pruebas de caja blanca:** Conociendo el funcionamiento del producto se pueden desarrollar pruebas que aseguren que "todas las piezas encajen", o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada.

Para el sistema desarrollado se le realizaron pruebas de caja negra, la cual se centra fundamentalmente en los requisitos funcionales del *software*.

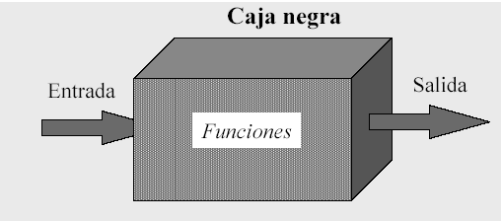


Figura 22: Representación de las pruebas de caja negra

El objetivo de una prueba de caja negra es verificar la función y el comportamiento observable especificado de la unidad sin conocer cómo implementa la función y el comportamiento. Las pruebas de caja negra se centran y se basan en la entrada y la salida de la unidad y permiten encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.

- Errores de rendimiento.
- Errores de inicialización y terminación.

Dentro de las técnicas de prueba de caja negra, se utilizó la Partición de Equivalencia. Esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones de *software*. Mediante la aplicación de esta se examinaron los valores válidos e inválidos de las entradas existentes en el *software*, descubriendo de forma inmediata clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar.

4.3.1.1. Casos de prueba

"Un caso de prueba especifica una forma de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones en las que ha de probarse." (Jacobson, y otros, 2000)

A continuación se ejemplifica el Caso de prueba correspondiente al CU Adicionar documento común. **(Tabla 17: Secciones a probar en el caso de uso, Tabla 18: Descripción de la variable, Tabla 19: Matriz de datos)**

Para examinar los restantes casos de prueba, consultar los documentos Diseño de casos de prueba.

Diseño del caso de prueba 1: CU Adicionar documento común.

Descripción General:

El CU inicia cuando se necesitan registrar los datos sobre la llegada de un documento común a la FGR. Este CU finaliza cuando el documento se encuentra registrado en la cola "Entrada de Documentos".

Condiciones de Ejecución:

- El usuario debe estar autenticado.
- El usuario debe escoger "Nuevo Documento Común" e introducir los campos requeridos.
- El usuario debe pulsar el botón "Guardar".

Secciones a probar en el Caso de Uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad

SC 1: Nuevo Documento Común.	EC 1.1: Nuevo Documento Común.	1.1 El sistema muestra al usuario los parámetros a completar para los nuevos documentos entre los que se encuentran los referentes a: • Documento. • Portador. • Remitente. Se muestra además los botones: • Guardar. • Cancelar
	EC 1.2: Completar los datos para los nuevos documentos.	1.2 El sistema muestra una interfaz para completar la información referente: Para documentación se solicitan los parámetros: • No. Referencia • Observaciones • Cantidad de hojas • Provincia Para Portador se solicitan los parámetros: • Nombre • Primer Apellido • Segundo Apellido • No. Carnet • Dirección Particular Para Remitente se solicitan los parámetros: • Nombre • Primer Apellido • Segundo Apellido Luego de insertados todos los campos de forma correcta, se procede a guardar la información. 1.2.1: El sistema emite un mensaje informando que la operación se realizó con éxito. *Finaliza el caso de uso.*
	EC 1.3: Flujo alterno “Datos incorrectos o incompletos”.	1.3: El sistema muestra una notificación de que los datos introducidos son incorrectos o están

		incompletos y da la opción para que vuelvan a ser introducidos mostrando nuevamente los campos vacíos.

Tabla 17: Secciones a probar en el caso de uso

Descripción de variable:

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	**No. Referencia**	Campo de texto	No	Un número entero mayor que cero con un índice único.
2	**Observaciones**	Campo de texto	Sí	Se introducen observaciones pertinentes al documento.
3	**Cantidad de hojas**	Campo de texto	No	Número entero mayor que cero que indique la cantidad de hojas.
4	**Provincia**	Lista desplegable	No	Se escoge la provincia.
5	**Nombre**	Campo de texto	No	Se introduce el nombre del Portador.
6	**Primer Apellido**	Campo de texto	No	Se introduce el primer apellido del Portador.
7	**Segundo Apellido**	Campo de texto	No	Se introduce el segundo apellido del Portador.
8	**No. Carnet**	Campo de texto	No	Un número entero de longitud once.
9	**Dirección Particular**	Campo de texto	No	Se introduce la dirección del Portador.
10	**Nombre**	Campo de texto	No	Se introduce el nombre del Remitente.
11	**Primer Apellido**	Campo de texto	No	Se introduce el primer apellido del Remitente.
12	**Segundo Apellido**	Campo de texto	No	Se introduce el segundo apellido del Remitente.

Tabla 18: Descripción de la variable

Matriz de Datos

SC 1 Nuevo Documento Común

Escenario	V1	V3	V8	V2	V4	V5	V 6	V7	V9	V 10	V 11	V 12	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
EC 1.1: Nuevo Documento Común.	***V***	***V***	***V***	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	Se adiciona el documento al sistema.	Satisfactorio	**-**El usuario accede a "Nuevo Documento Común". -El usuario inserta los datos solicitados. -El usuario pulsa el botón "Guardar".
	V	***V***	***I***	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	Se emite un mensaje indicando que debe entrar números en ese campo.	Satisfactorio	
	V	***I***	***V***	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	Se emite un mensaje indicando que debe entrar números en ese campo.	Satisfactorio	
	I	***V***	***V***	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	Se emite un mensaje indicando que debe entrar números en ese campo.	Satisfactorio	
	I	***I***	***I***	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	Se emite un mensaje indicando que debe entrar números en ese campo.	Satisfactorio	
	I	***I***	***V***	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	Se emite un mensaje indicando que debe entrar números en ese campo.	Satisfactorio	
	I	***V***	***I***	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	Se emite un mensaje indicando que debe entrar números en ese campo.	Satisfactorio	
	V	***I***	***I***	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	**N/A**	Se emite un mensaje indicando que debe entrar números en ese campo.	Satisfactorio	

Tabla 19: Matriz de datos

Este caso de prueba resultó completamente satisfactorio ya que no se encontraron defectos ni dificultades en su ejecución.

De manera general los resultados obtenidos al realizar las pruebas de caja negra al Módulo para la gestión de la entrada y salida de correspondencia y documentación oficial de la FGR, fueron satisfactorios. Estas pruebas se dividieron en dos etapas. La primera de ellas se fraccionó en dos iteraciones que fueron llevadas a cabo por el equipo de desarrollo. La iteración inicial arrojó como resultados 2 errores, consistentes en errores en los mensajes de validación, los cuales fueron corregidos en la segunda iteración, donde todas las pruebas reflejaron resultados positivos.

La segunda etapa comprende las revisiones realizadas por parte del Grupo de Calidad de la Facultad 15. Luego de haberse efectuado 3 iteraciones de revisiones a los artefactos generados se detectó un promedio de 10 no conformidades, las cuales fueron corregidas. Mediante el desarrollo de los casos de pruebas se identificaron los errores que tenía el sistema los cuales fueron reevaluados a través de la realización de nuevas pruebas en una segunda iteración y donde no se encontraron nuevos errores.

4.4. Conclusiones parciales

En este capítulo se concluye lo siguiente:

- El Modelo de implementación permitió obtener una visión general de los componentes y subsistemas a implementar, facilitando que el proceso de implementación se realizara de forma más simple y con una mayor calidad.
- El Modelo de prueba permitió constatar que las funcionalidades implementadas son correctas, por lo que se corresponden con los requisitos establecidos por el cliente y las normas de calidad establecidas por parte del Grupo de Calidad de la Facultad 15. Se obtuvo, por tanto, la liberación del producto por parte de este último. **(Anexo # 1 “Acta de liberación de calidad”)**

CONCLUSIONES GENERALES

Con la realización del siguiente trabajo se concluye que:

- Se desarrolló el módulo para el proyecto SGF que permite la gestión de la correspondencia y la documentación oficial que transita por la FGR, dándose cumplimiento al objetivo propuesto.
- La elaboración del Modelo del negocio y el Modelo del sistema posibilitó definir las funcionalidades que posee la aplicación y lograr un mayor entendimiento común entre los clientes y desarrolladores.
- La construcción del Modelo del diseño permitió generar los elementos necesarios para desarrollar la implementación del sistema, al concebir una representación técnica del mismo.
- La validación del sistema mediante la aplicación de las pruebas de caja negra arrojó resultados satisfactorios, demostrándose la fiabilidad del *software* desarrollado y obteniéndose la liberación del producto por parte del Grupo de Calidad de la Facultad 15.

RECOMENDACIONES

Como complemento al presente trabajo, se recomienda lo siguiente:

- Desplegar en la FGR la solución aquí propuesta, para validar su uso y adaptación.
- Implementar en futuras versiones del sistema nuevas funcionalidades que aporten comodidad, facilidad y resuelvan nuevas peticiones de los implicados según necesidades o interés de los mismos como son, por ejemplo, la extensión de la aplicación del sistema a todas las Fiscalías del país y que se puedan rastrear la trayectoria de los documentos a través de todas ellas.
- Realizar encuestas periódicas sobre la efectividad del sistema en la organización, el grado de satisfacción de los usuarios con el sistema, así como funcionalidades que se le puede incorporar. Todo esto previendo un futuro perfeccionamiento del sistema.
- La conservación de este documento por parte de la Universidad como consulta y guía para el proyecto SGF y demás personal que proporcionará mantenimiento al sistema.

REFERENCIAS BIBLIOGRÁFICAS

Alarcón, José Manuel. 2006. *Administración de SGBD PostgreSQL.* 2006.

Albacete. 2002. *Diseño y Programación Orientada a Objetos.* [En línea] 2002. [Citado el: 3 de marzo de 2010.] http://www.info-ab.uclm.es/asignaturas/42579/cap4/Creacion.htm.

Calvo, O. 2007. *Catalogo de S.L. para gestores de información.* [En línea] 2007. [Citado el: 26 de febrero de 2010.] http://catalogosl.wordpress.com/2007/06/27/knowledgd-tree-gestor-de-documentos-digitales/.

Catálogo de Software. 2010. *DOCMANAGER, Suite de Aplicaciones de Apoyo al Sistema de Gestión de la Calidad .* [En línea] 28 de febrero de 2010. http://www.catalogodesoftware.com/producto.aspx?pid=340.

Chalef, D. 2009. *CMS_SPAIN.com.* [En línea] 2009. [Citado el: 25 de febrero de 2010.] http://www.ecm-spain.com/noticia.asp?IdItem=7881.

Codina, Lluís. 1993. "Qué es un sistema de gestión documental". [En línea] mayo de 1993. [Citado el: 20 de febrero de 2010.] http://www.elprofesionaldelainformacion.com/.

Codina, Lluis. 1993. El profesional de la Información. [En línea] mayo de 1993. [Citado el: 19 de febrero de 2010.] http://www.elprofesionaldelainformacion.com/contenidos/1993/mayo/qu_es_un_sistema_de_gestin_documental.html.

Eguíluz Pérez, Javier. 2008. *Introducción a Ajax.* 2008.

Eguíluz, Javier. Symfony en pocas palabras. [En línea] [Citado el: 1 de abril de 2010.] http://www.librosweb.es/symfony/capitulo1/symfony_en_pocas_palabras.html.

Entorno Virtual de Aprendizaje. 2009 - 2010. Ingeniería de Software I. *Conferencia 5.* [En línea] 2009 - 2010. [Citado el: 13 de marzo de 2010.] http://eva.uci.cu/mod/resource/view.php?id=21010.

Entorno Virtual de Aprendizaje. 2009 - 2010. Ingeniería de Software II. *Conferencia # 5 "Culminación del flujo de trabajo Análisis y Diseño. Diseño de la Base de Datos".* [En línea] 2009 - 2010. [Citado el: 10 de abril de 2010.] http://eva.uci.cu/mod/resource/view.php?id=29333.

Fugueras R., Alberch i. 2003. *Los archivos entre la memoria histórica y la sociedad del conocimiento.* Barcelona: UOC : s.n., 2003.

Gracia, J. 2005. Ingenieros Software. [En línea] 27 de mayo de 2005. [Citado el: 5 de marzo de 2010.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

Graham. 2001. Diseño yProgramacion de Páginas Web. [En línea] 2001. [Citado el: 26 de febrero de 2010.] www.e-book-tutoriales.blogspot.com.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison - Wesley, 2000.

Lamb, David Alex. 1998. *What's a CASE Tool?* s.l. : Advameg, Inc., 1998.

Larman, Craig. 2008. El Mundo Informatico. [En línea] 2008. [Citado el: 24 de febrero de 2010.] http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/.

Marcos, Mari Carmen. 1999. El Profesional de la Infromación. *Los archivos en la era digital.* [En línea] junio de 1999. [Citado el: 18 de febrero de 2010.] http://www.elprofesionaldelainformacion.com/contenidos/1999/junio/los_archivos_en_la_era_digital.html.

Marqués Andrés, María Mercedes. Sistemas de bases de datos. [En línea] [Citado el: 10 de marzo de 2010.] http://www3.uji.es/~mmarques/f47/apun/node4.html.

Moreno, Ana M. y Sánchez-Segura, Maribel. *Patrones de Usabilidad: Mejora de la Usabilidad del Software desde el momento de Arquitectónico.* [En línea] [Citado el: 10 de marzo de 2010.] http://www.willydev.net/descargas/prev/PatronesUsa.pdf.

Övergaard, Gunnar y Palmkvist, Karin. 2004. *Use Cases Patterns and Blueprints.* s.l. : Addison Wesley Professional, 2004. ISBN 0-13-145134-0.

Pressman, R. S. 1998. *Ingeniería del Software. Un Enfoque Práctico.* 1998.

Rumbaugh, James, Jacobson, Ivar y Booch, Grady. 1998. *The Unified Modeling Language. Reference Manual.* Massachusetts : Addison Wesley, 1998.

Sitio Oficial de Alfresco. Alfresco. [En línea] [Citado el: 26 de febrero de 2010.] http://www.alfresco.com/.

Sitio Oficial de jQuery. Sitio Oficial de jQuery. [En línea] [Citado el: 1 de marzo de 2010.] http://www.jquery.com.

Sitio Oficial de Knowledgetree. 2008. *Knowledgetree.* [En línea] 2008. [Citado el: 25 de febrero de 2010.] http://www.knowledgetree.com/.

Sitio Oficial OrfeoGPL. OrfeoGPL.org. [En línea] http://orfeogpl.org/ata/.

Sitio Web "La Gestión del Conocimiento". La Gestión del Conocimiento. *Gestión Documental.* [En línea] [Citado el: 21 de febrero de 2010.] http://www.gestion-conocimiento.com/contenido/gestiondoc.asp.

Sitio Web Arazandi. Arazandi. *Spam, Glosario de términos sobre Internet y spam.* [En línea] [Citado el: 27 de febrero de 2010.] http://www.aranzadi.es/index.php/informacion-juridica/informacion-interes/glosario-de-terminos-sobre-internet-y-spam.

Sitio Web Directorio Electrónico de Guatemala. Directorio Electrónico de Guatemala. *Glosario.* [En línea] [Citado el: 13 de marzo de 2010.] http://www.deguate.com/infocentros/gerencia/glosario/c.htm.

Sitio Web E.T.S.I.Informática. ¿Qué es JavaScript? [En línea] [Citado el: 22 de febrero de 2010.] http://www.lcc.uma.es/~eat/services/html-js/manual14.html.

The Menlo Institute. 2002. The Menlo Institute. *The Rational Unified Process.* [En línea] 2002. [Citado el: 17 de febrero de 2010.] http://www.menloinnovations.com.

Vargas Del Valle, Ricardo J. 2007. Programación en Capas. [En línea] 2007. [Citado el: 1 de marzo de 2010.] http://www.di-mare.com/adolfo/cursos/2007-2/pp-3capas.pdf.

Vaswani, Vikram. 2008. *PHP. A Begginer's Guide.* New York : Mc Graw Hill, 2008.

Zaninotto, François y Potencier, Fabien. *Symfony 1.2, la guía definitiva.*

BIBLIOGRAFÍA

Álvarez, Miguel Ángel. 2002. DesarrolloWeb. *DesarrolloWeb.* [En línea] 14 de marzo de 2002. [Citado el: 15 de junio de 2010.] http://www.desarrolloweb.com/articulos/714.php.

Álvarez, Miguel Ángel. 2001. DesarrolloWeb. *DesarrolloWeb.* [En línea] 9 de mayo de 2001. [Citado el: 15 de junio de 2010.] http://www.desarrolloweb.com/articulos/392.php .

Álvarez, Miguel Ángel. 2001. DesarrolloWeb. *DesarrolloWeb.* [En línea] 9 de mayo de 2001. [Citado el: 15 de junio de 2010.] http://www.desarrolloweb.com/articulos/393.php .

Dante, P. G. 2005. *Gestión documental, gestión de información y gestión del conocimiento: evolución y sinergias. Comunicación preliminar.* [En línea] 2005. [Citado el: 20 de febrero de 2010.] http://www.cinfo.cu/Userfiles/file/Cinfo/CINFO2005/diciembre2005/Comunicacion.doc.

Gaceta Oficial de la República de Cuba. *Ley de la Fiscalía General de la República.* [En línea] 2008 http://www.gacetaoficial.cu/html/fiscaliageneralrepublica.html

Herramientas Case. **Instituto Nacional de Estadística Informática. 1999.** 22, s.l. : Oficina de Impresiones de la Oficina Técnica de Difusión Estadística y Tecnología Informática del Instituto Nacional de Estadística e Informática (INEI), 1999. 875-99-OI-OTDETI-INEI.

Laurel, B. 1993. *Computer as Theatre.* Nueva York : Addison-Wesley Publishing Company, Inc., 1993.

Sitio Web Netcraft. 2010. Netcraft. [En línea] 7 de junio de 2010. [Citado el: 14 de junio de 2010.] http://news.netcraft.com/archives/2010/01/07/january_2010_web_server_survey.html.

Rational Software Corporation. 2003. *Rational Unified Process 1.0.* [En línea] junio de 2003. [Citado el: 21 de febrero de 2010.]

Anexo # 1 "Acta de liberación de calidad"

UCI

Informáticas

Acta de Liberación de Artefactos, Grupo de Calidad Centro CEGEL de la Facultad 15 de la Universidad de las Ciencias Informáticas.

Martes, 22 de junio de 2010.

Luego de haber efectuado 3 iteraciones de revisiones a los artefactos: Especificación de Requisitos, Modelo de Negocio, Modelo de Sistema y Modelo de diseño del módulo Gestión de la entrada y salida de la correspondencia y documentación oficial de la Fiscalía General de la República de Cuba del proyecto Sistema de Gestión Fiscal del Centro CEGEL de la Facultad 15 y haberse detectado un promedio de 10 No Conformidades, se puede afirmar que se han corregido los defectos encontrados, por lo que queda liberada la aplicación.

Firma del Asesor y Jefe del Grupo de Calidad Centro CEGEL

Ing. Raúl Velázquez Alvarez

UCI Universidad de las Ciencias Informáticas

Decanato Facultad