Universidad de las Ciencias Informáticas

Facultad 3

Implementación del módulo Misiones para el Sistema de Gestión para el Convenio Integral de Cooperación Cuba-Venezuela.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor: Javier Domínguez Pérez

Tutora: Ing. Yaniet Piñeiro Pérez

Asesor: Ing. Ariel Morales Malpica.

Julio 2011

Seguid hambrientos, seguid alocados

Steve Jobs

DECLARACIÓN DE AUTORÍA

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

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

Autor: Javier Domínguez Pérez

Tutora: Ing. Yaniet Piñeiro Pérez

Asesor: Ing. Ariel Morales Malpica.

DATOS DE CONTACTO

Ing. Yaniet Piñeiro Pérez
E-mail: ypineiro@uci.cu

Ing. Ariel Morales Malpica
E-mail: aemorales@uci.cu

A mi padre,
A mi madre,
A mi hermana,
A Dayamí.

RESUMEN

Con el surgimiento del Convenio Integral de Cooperación entre Cuba y Venezuela se han creado muchos proyectos que han beneficiado a ambas partes. La gran cantidad de información que se manejaba de los proyectos se hacía de forma manual lo que provocaba un atraso en el proceso, en el control de la cantidad de proyectos y el monto que representaban. Debido a esto se decidió crear una aplicación para gestionar la información de los proyectos de manera más eficiente. Esta aplicación presentó problemas de rendimiento por lo que se hizo un estudio para detectar las causas de este problema y se decidió implementar una nueva versión que solucionara esto.

En el presente trabajo de diploma se expone la implementación del módulo Misiones para el Sistema de Gestión para el Convenio Integral de Cooperación Cuba-Venezuela. El desarrollo de este módulo tiene el objetivo de aumentar el rendimiento en la gestión de las solicitudes de Misiones con respecto al sistema anterior.

PALABRAS CLAVE

rendimiento, solicitud de Misiones, convenio

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Gracias al Convenio Integral de Cooperación entre la República de Cuba y la República Bolivariana de Venezuela, ambas partes se han beneficiado por una gran cantidad de proyectos y el número de estos sigue en aumento en el marco de la Alternativa bolivariana para las Américas. En las Comisiones Mixtas Cuba-Venezuela se conciben una serie de proyectos bilaterales, los cuales deben seguir un conjunto de pasos hasta su contratación y firma. Inicialmente y durante algún tiempo la elaboración, aprobación y financiamiento de las Fichas Técnicas en el marco de las Mixtas se realizaba de forma manual lo que provocaba una demora significativa en el proceso y se hacía difícil el control total de la cantidad de proyectos propuestos y del monto que representaban. Por esta razón se decidió la creación del Sistema de Gestión para el Seguimiento de los Proyectos del Convenio Integral de Cooperación Cuba-Venezuela. Este debía brindar un mejor seguimiento a los proyectos que se ejecutaban, facilitar mayor variedad en la obtención de los reportes para la toma de decisiones y facilitar el almacenamiento de la información relacionada con los proyectos de forma íntegra y segura.

La primera versión del sistema fue desarrollada en tiempo de acuerdo al cronograma de desarrollo propuesto y una vez desplegado jugó un papel importante en la IX Mixta Cuba-Venezuela. A pesar de esto el sistema presentó problemas de rendimiento. El tiempo de respuesta del sistema era alto y la interfaz de usuario demoraba en cargarse.

A partir de lo planteado anteriormente se define el siguiente **problema de investigación**: La poca utilización de los patrones de diseño, el tamaño de las interfaces y la sobrecarga de clases del módulo Misiones afectan su rendimiento.

El **objeto de estudio** lo constituyen los procesos de desarrollo de software.

El **campo de acción** es la implementación de aplicaciones web de gestión.

El **objetivo general** de este trabajo es implementar el módulo Misiones del Sistema de Gestión para el Convenio Integral de Cooperación Cuba-Venezuela para aumentar su rendimiento.

Para llevar a cabo este objetivo general se plantean los siguientes **objetivos específicos**:

- ✓ Elaborar el marco teórico de la investigación.
- ✓ Implementar el módulo Misiones del Sistema de Gestión para el Convenio Integral de Cooperación Cuba-Venezuela para lograr un mayor rendimiento, respecto a la versión anterior.
- ✓ Validar la solución propuesta respecto al cumplimiento de los requisitos y su rendimiento.

De acuerdo a los objetivos específicos se definen las siguientes **tareas de investigación:**

- ✓ Consultar bibliografía relacionada con las Metodologías de Desarrollo de Software y tecnologías usadas en el desarrollo de aplicaciones web.
- ✓ Determinar la Metodología de Desarrollo de Software y las tecnologías para el desarrollo del módulo.
- ✓ Implementar el módulo de Misiones.
- ✓ Realizar pruebas unitarias.
- ✓ Aplicar métricas para validar la calidad de la solución.
- ✓ Evaluar el módulo según el rendimiento del sistema.

Como **idea a defender** se tiene:

Con la implementación del Módulo Misiones del Sistema de Gestión para el Convenio Integral de Cooperación Cuba-Venezuela se logrará aumentar el rendimiento en la gestión de las solicitudes de Misiones.

Los métodos usados en la investigación son:

Métodos Teóricos:

- ✓ Analítico – sintético: Para analizar la bibliografía consultada en el estudio del estado del arte.
- ✓ Histórico – lógico: Para el estudio del estado del arte de las herramientas propuestas para el desarrollo del módulo y para el análisis evolutivo de dichas herramientas.

Métodos empíricos:

- ✓ Experimento: Para evaluar el rendimiento del sistema.

El presente trabajo se encuentra estructurado en tres capítulos, resumidos de la siguiente forma:

Capítulo 1: Se realiza la fundamentación teórica del trabajo, incluyendo un estudio del estado del arte de metodologías de desarrollo de software, lenguajes de programación y tecnologías como frameworks de desarrollo, gestores de bases de datos, entornos de desarrollo integrado, servidores web, frameworks para pruebas unitarias y herramientas para crear reportes.

Capítulo 2: Brinda una descripción de la solución propuesta. Incluye los artefactos relacionados con el flujo de implementación y convenciones de codificación. Además se muestran las medidas tomadas en la implementación con vistas a lograr un rendimiento superior a la primera versión del módulo Misiones.

Capítulo 3: Se valida la solución propuesta a través de pruebas de caja blanca, caja negra y de rendimiento.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

En este capítulo se analizan metodologías de desarrollo de software, lenguajes de programación y tecnologías como frameworks de desarrollo, gestores de bases de datos, entornos integrados de desarrollo, servidores web, frameworks para pruebas unitarias y herramientas para crear reportes usados a nivel mundial en el desarrollo de aplicaciones web. Además se hace un análisis justificando la selección de la metodología de desarrollo de software y las tecnologías a utilizar en el desarrollo del módulo Misiones.

1.2 Metodologías de desarrollo de software.

Las metodologías de desarrollo de software representan un grupo de políticas, procesos y procedimientos usados para el desarrollo de sistemas de información. Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Actualmente se pueden clasificar las metodologías en dos grandes grupos: las metodologías ágiles y las tradicionales.

Las metodologías ágiles se basan en dos aspectos puntuales, el retrasar las decisiones y la planificación adaptativa; permitiendo potenciar aún más el desarrollo de software a gran escala. Ya que tienen un desarrollo de software incremental, es decir, se van entregando pequeñas porciones de software en períodos de tiempo cortos. Además de que los desarrolladores y clientes trabajan juntos, permitiendo de esta forma una cercana comunicación entre ellos, logrando controlar de una forma efectiva cualquier cambio que se produzca en el proceso de desarrollo del software. Y por sobre todo es muy sencillo pues es muy fácil de aprender, de modificar y muy bien documentado. Dentro de estas metodologías se encuentran el Extreme Programing (XP) y SCRUM.

En las metodologías tradicionales, 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. Se caracterizan por comenzar con la obtención y análisis de los requerimientos solicitados por el usuario, luego de una intensa interacción con los usuarios y clientes, se definen los requerimientos funcionales y no funcionales del futuro sistema. Estas metodologías están basadas en la producción de proyectos de larga duración, lo que permite estructurar un amplio equipo de trabajo en roles que cumplirían diferentes funciones dentro de la producción. Entre las principales metodologías tradicionales existen los tan conocidos RUP y MSF.

Se decidió no utilizar una metodología ágil para desarrollar la solución porque al estar los clientes en Venezuela no era posible que formaran parte del equipo de desarrollo. Por lo tanto se decidió utilizar una metodología tradicional, que mediante la obtención de los requisitos solicitados por el usuario y una gran documentación permitieran un mejor entendimiento de lo que se va a hacer al equipo de desarrollo.

1.2.1 MSF.

El marco de desarrollo de soluciones de Microsoft (MSF) es una serie de conceptos, modelos y prácticas de uso que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. Se centra en los modelos de procesos y equipos. MSF propone 5 fases de desarrollo: Visión y alcance, planificación, desarrollo, estabilización y despliegue. MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación. El modelo de proceso combina los mejores principios del modelo en cascada y del modelo en espiral. Combina la claridad que planea el modelo en cascada y las ventajas de los puntos de transición del modelo en espiral.

MSF tiene las siguientes características:

- ✓ Adaptable: es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
- ✓ Escalable: puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.
- ✓ Flexible: es utilizada en el ambiente de desarrollo de cualquier cliente.
- ✓ Tecnología Agnóstica: porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

1.2.2 RUP.

El Proceso Unificado Racional, Rational Unified Process en inglés, y sus siglas RUP, es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP no es un sistema con pasos firmemente establecidos, sino que trata de un conjunto de metodologías adaptables al contexto y necesidades de cada organización, donde el software es organizado como una colección de unidades atómicas llamados objetos, constituidos por datos y funciones, que interactúan entre sí.

RUP tiene tres características fundamentales: dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental (1).

Los casos de uso no son sólo una herramienta para especificar los requisitos de un sistema, también guían su diseño, su implementación y prueba, es decir, guían el proceso de desarrollo basándose en el modelo de casos de uso. Los casos de uso se especifican, se diseñan y son la fuente a partir de la cual los ingenieros de pruebas construyen sus casos de pruebas (1).

La arquitectura surge de las necesidades de la empresa, como las perciben los usuarios y los inversores, y se refleja en los casos de uso. También se ve influida por varios factores, como la plataforma en la que tiene que funcionar el software (arquitectura del hardware, sistema operativo, sistema de gestión de base de datos, protocolos para comunicaciones de red), los bloques de construcción reutilizables de que se dispone por ejemplo un marco de trabajo. La arquitectura es una vista del diseño completo con las características más importantes resaltadas (1).

El desarrollo de un producto software comercial supone un gran esfuerzo que puede durar entre varios meses hasta posiblemente un año o más. Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo, los incrementos y al crecimiento del producto. Para una efectividad máxima, las iteraciones deben estar controladas, deben seleccionarse y ejecutarse de una forma planificada. (1)

RUP se divide en 4 fases, dentro de las cuales se realizan varias iteraciones según el proyecto y en las que se hace mayor o menos esfuerzo en las distintas actividades. En las iteraciones de cada fase se hacen diferentes esfuerzos en diferentes actividades:

- ✓ **Fase de inicio**: Se hace un plan de fases, donde se identifican los principales casos de uso y se identifican los riesgos. Se concreta la idea, la visión del producto, como se enmarca en el negocio y el alcance del proyecto.
- ✓ **Fase de elaboración**: se realiza el plan de proyecto, donde se completan los casos de uso y se mitigan los riesgos. Planificar las actividades necesarias y los recursos requeridos, especificando las características y el diseño de la arquitectura.
- ✓ **Fase de construcción**: se basa en la elaboración de un producto totalmente operativo y en la elaboración del manual de usuario. Construir el producto, la arquitectura y los planes, hasta que el producto está listo para ser enviado a la comunidad de usuarios.

- ✓ **Fase de transición**: se realiza la instalación del producto en el cliente y se procede al entrenamiento de los usuarios. Realizar la transición del producto a los usuarios, lo cual incluye: manufactura, envío, entrenamiento, soporte y mantenimiento del producto, hasta que el cliente quede satisfecho, por tanto en esta fase suelen ocurrir cambios.

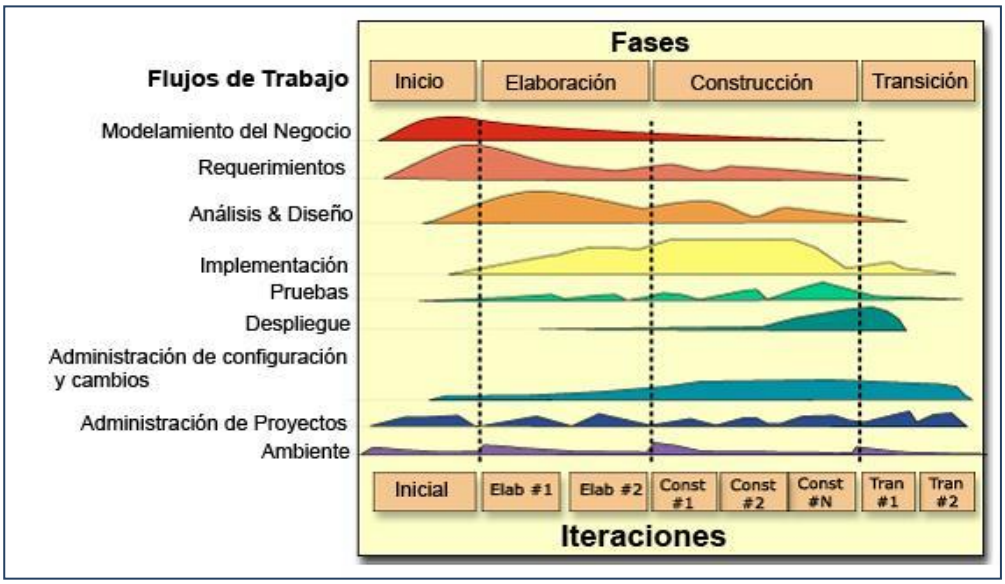


Fig. 1 Fases y flujos de trabajo de RUP

Modelamiento del negocio:

Con este flujo de trabajo pretendemos llegar a un mejor entendimiento de la organización donde vamos a implantar nuestro producto. Los principales motivos para esto son los siguientes (2):

- ✓ Asegurarnos de que el producto será algo útil, no un obstáculo.
- ✓ Conseguir que encaje de la mejor forma posible en la organización.
- ✓ Tener un marco común para los desarrolladores, los clientes y los usuarios finales.

Requerimientos:

Este es uno de los flujos de trabajo más importantes, porque en él se establece QUÉ es lo que tiene que hacer exactamente el sistema que construyamos. En esta línea los requisitos son el contrato que debemos cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que especifiquemos (2).

Análisis y diseño:

El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describe cómo implementar el sistema. El análisis consiste en obtener una visión del sistema que se preocupa de ver

QUÉ hace, de modo que sólo se interesa por los requisitos funcionales. Por otro lado el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva CÓMO cumple el sistema sus objetivos (2).

Implementación:

En este flujo de trabajo se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y demás. Además se deben hacerlas pruebas de unidad: cada implementador es responsable de testear las unidades que produzca. El resultado final de este flujo de trabajo es un sistema ejecutable (2).

Pruebas:

Este flujo de trabajo es el encargado de evaluar la calidad del producto que estamos desarrollando, pero no para aceptar o rechazar el producto al final del proceso de desarrollo, sino que debe ir integrado en todo el ciclo de vida (2).

Despliegue:

El objetivo de este flujo de trabajo es producir con éxito distribuciones del producto y distribuirlo a los usuarios. Las actividades implicadas incluyen:

- ✓ Testear el producto en su entorno de ejecución final.
- ✓ Empaquetar el software para su distribución.
- ✓ Distribuir el software.
- ✓ Instalar el software.
- ✓ Proveer asistencia y ayuda a los usuarios.
- ✓ Formar a los usuarios y al cuerpo de ventas.
- ✓ Migrar el software existente o convertir bases de datos (2).

Administración de configuración y cambios:

La finalidad de este flujo de trabajo es mantener la integridad de todos los artefactos que se crean en el proceso, así como de mantener información del proceso evolutivo que han seguido (2).

Administración de proyectos:

El objetivo de la administración de un proyecto es conseguir equilibrar el completar los objetivos, administrar el riesgo y superar las restricciones para desarrollar un producto que sea acorde a los requisitos de los usuarios (2).

Ambiente:

La finalidad de este flujo de trabajo es dar soporte al proyecto con las adecuadas herramientas, procesos y métodos. Es decir tener a punto las herramientas que se vayan a necesitar en cada momento, así como definir la instancia concreta de proceso unificado que se va a seguir. En concreto las responsabilidades de este flujo de trabajo incluyen (2):

- ✓ Selección y adquisición de herramientas.
- ✓ Establecer y configurar las herramientas para que se ajusten a la organización.
- ✓ Configuración del proceso.
- ✓ Mejora del proceso.
- ✓ Servicios técnicos.

1.2.3 Selección de la metodología de desarrollo de software.

Se seleccionó RUP para el desarrollo de la solución pues es una de las metodologías más abarcadoras para el análisis, implementación y documentación de sistemas orientados a objetos. Se puede ajustar a las necesidades del proyecto, ya que brinda facilidades relacionadas a la organización y genera gran documentación, contribuyendo a un mejor entendimiento del equipo de trabajo. Además es fundamental el conocimiento de los integrantes del equipo de desarrollo sobre esta metodología, ahorrando consigo tiempo de estudio e investigación.

1.3 Lenguajes de programación.

Un lenguaje de programación, es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente.

1.3.1 Java.

Java es toda una tecnología orientada al desarrollo de software con la cual se puede realizar cualquier tipo de programa. Hoy en día, la tecnología Java ha cobrado mucha importancia en el ámbito de Internet gracias a su plataforma J2EE. También en la industria para dispositivos móviles hay una gran acogida para este lenguaje.

En un primer nivel, Java es un lenguaje de programación de propósito general, orientado a objetos, que fue introducido por Sun Microsystems en 1995, y diseñado en principio para el ambiente distribuido de Internet. Pero lo que hace de Java un concepto diferente es que, en un segundo nivel, es también un entorno para la ejecución de programas, englobado en la llamada máquina virtual de Java. Este entorno

es un software que permite que las aplicaciones escritas en Java se ejecuten en cualquier ordenador, independientemente del sistema operativo y de la configuración de hardware utilizados (4).
Algunas de sus características son (5):

- ✓ Robustez.
- ✓ Gestión automática de la memoria.
- ✓ Permite programación multihilos.
- ✓ Soporta la arquitectura cliente-servidor.
- ✓ Mecanismos de seguridad incorporados.
- ✓ Herramientas de documentación incorporadas.

1.3.2 PHP.

PHP (acrónimo de PHP: Hypertext Preprocessor) es un lenguaje de código abierto muy popular especialmente adecuado para desarrollo web y que puede ser incrustado en HTML (6).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <title>Example</title>
    </head>
    <body>

        <?php
            echo "Hola, ¡soy un script PHP!";
        ?>

    </body>
</html>
```

Fig. 2 Ejemplo de código PHP

En lugar de usar muchos comandos para mostrar HTML (como en C o Perl), páginas PHP contienen HTML con código PHP incluido en el mismo que en este caso, muestra el texto "Hola ¡soy un script PHP!". El código PHP está en medio de etiquetas **<?php** y **?>** que indican el comienzo y final de este (6).

PHP corre en (casi) cualquier plataforma utilizando el mismo código fuente, pudiendo ser compilado y ejecutado en aproximadamente 25 plataformas, incluyendo diferentes versiones de Unix, Windows y Macs. Como en todos los sistemas se utiliza el mismo código base, los scripts pueden ser

ejecutados de manera independiente al OS. La sintaxis de PHP es similar a la del C, por esto cualquiera con experiencia en lenguajes del estilo C podrá entender rápidamente PHP. Entre los lenguajes del tipo C incluimos a Java y Javascript, de hecho mucha de la funcionalidad del PHP se la debe al C en funciones como fread() o srtlen() (7).

PHP es completamente expandible. Está compuesto de un sistema principal (escrito por Zend), un conjunto de módulos y una variedad de extensiones de código. Muchas interfaces distintas para cada tipo de servidor. PHP actualmente se puede ejecutar bajo Apache, IIS, AOLServer, Roxen y THTTPD. Otra alternativa es configurarlo como módulo CGI. Puede interactuar con muchos motores de bases de datos tales como MySQL, MS SQL, Oracle, Informix, PostgreSQL, y otros muchos. Siempre podrás disponer de ODBC para situaciones que lo requieran. Tiene una gran variedad de módulos cuando un programador PHP necesite una interfaz para una librería en particular, fácilmente podrá crear una API para esta. Algunas de las que ya vienen implementadas permiten manejo de gráficos, archivos PDF, Flash, Cybercash, calendarios, XML, IMAP, POP, etc (7).

PHP generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz. Está completamente escrito en C, así que se ejecuta rápidamente utilizando poca memoria.

PHP es Open Source, lo cual significa que el usuario no depende de una compañía específica para arreglar cosas que no funcionan, además no estás forzado a pagar actualizaciones anuales para tener una versión que funcione. Muchos de nosotros que hemos esperado que Allaire arregle algo apreciamos esto (7).

1.3.3 Selección del lenguaje de programación.

Se decidió utilizar Java para el desarrollo de la solución pues es es un lenguaje libre y existen entornos de desarrollo gratuitos de muy buena calidad para desarrollar con este. Java se caracteriza por su robustez, seguridad y escalabilidad, además el hecho de que sea un lenguaje tipado (hay que declarar el tipo de dato a las variables, a los parámetros de los métodos, etc) permite que sea más fácil entender el código y darle mantenimiento a la aplicación.Estas características hacen que Java sea muy usado en el desarrollo de aplicaciones de gestión.

1.4 Frameworks de desarrollo.

Un Framework es un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación. No son más que arquitecturas definidas para un determinado dominio de la aplicación que contiene un conjunto de componentes implementados y sus interfaces bien definidas, estos

componentes se pueden utilizar, redefinir y crear nuevos componentes. Los objetivos principales que persigue un framework son, entre otros, acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

1.4.1 OpenXava.

OpenXava es un framework para desarrollar aplicaciones JavaEE/J2EE rápida y fácilmente. La filosofía subyacente es definir con anotaciones de Java o con XML y programar con Java, pero cuanto más definimos y menos programamos mejor. El objetivo principal es hacer que las cosas más típicas en una aplicación de gestión sean fáciles de hacer, mientras que ofrecemos la flexibilidad suficiente para desarrollar las funciones más avanzadas y específicas (8).

El funcionamiento de OpenXava está basado en los componentes de negocio, en los controladores y los módulos. Los componentes de negocio son las clases del modelo. OpenXava genera la vista y el acceso a datos mediante anotaciones en los componentes de negocio.

```
1  package Entidades;
2
3  import java.util.*;
6
7  @Entity
8  @Table(name="empresa",schema="public")
9  public class Empresa {
10     @Id
11     @Column(name="idempresa")
12     private int idEmpresa;
13
14     @OneToMany(mappedBy = "empresa", cascade = CascadeType.ALL)
15     @Size(min = 1)
16     private Collection<Componente> componente;
17
18     @OneToMany(mappedBy = "empresa", cascade = CascadeType.ALL)
19     @Size(min = 1)
20     private Collection<Area> area;
21
```

Fig. 3 Clase del modelo con anotaciones.

Los controladores se definen en el fichero controladores.xml del paquete xava. Estos están compuestos de acciones en las que se programan el comportamiento de la aplicación.

```
<controlador nombre="RevisarEntidad">
    <accion nombre="Aceptar" clase="base.acciones.RevisarEntidad"
        confirmar="true" modo="detail">
        <poner propiedad="operacionRealizada" valor="1" />
    </accion>
    <accion nombre="Rechazar" clase="base.acciones.RevisarEntidad"
        modo="detail">
        <poner propiedad="operacionRealizada" valor="2" />
    </accion>
    <accion nombre="Cancelar" modo="detail" clase="base.acciones.CambiarModoAccion">
        <poner propiedad="modo" valor="list" />
    </accion>
</controlador>
```

Fig. 4 Controlador.

Finalmente, lo que relaciona al componente de negocio con los controladores es el módulo que se define en el fichero aplicacion.xml del paquete xava.

```
<modulo nombre="AdministrarRecursoHumano">
    <var-entorno nombre="XAVA_SEARCH_ACTION" valor="CRUD.searchExecutingOnChange"/>
    <modelo nombre="RecursoHumano"/>
    <vista nombre="administrarRecursoHumano"/>
    <controlador nombre="InitController"/>
    <controlador nombre="CRUD_Simple"/>
    <controlador nombre="ListOnly"/>
    <controlador-modo nombre="Void"/>
</modulo>
```

Fig. 5 Módulo.

Algunas características de OpenXava son:

- ✓ Usado durante años para desarrollar aplicaciones críticas.
- ✓ Alta productividad para aplicaciones de gestión.
- ✓ Curva de aprendizaje corta y sencillez de uso.
- ✓ Suficientemente flexible como para crear aplicaciones sofisticadas.
- ✓ Es posible insertar nuestra propia funcionalidad en cualquier punto.
- ✓ Basado en el concepto de componente de negocio.
- ✓ Aunque la interfaz de usuario es generada automáticamente es posible hacer un ajuste bastante fino de la presentación.
- ✓ Adaptado para trabajar con esquemas de base de datos legados.

- ✓ Genera una aplicación J2EE completa: incluyendo la interfaz de usuario y las clases del modelo (con POJOs o EJB).
- ✓ Soporta cualquier servidor de aplicaciones (Tomcat, JBoss, WebSphere).
- ✓ Soporta JSR-168: Todos los módulos OpenXava también son portlets estándar.
- ✓ Mecanismo de persistencia: EJB3 JPA, Hibernate o EJB2 CMP. Al gusto.
- ✓ Está probado con los portales: JetSpeed 2, WebSphere Portal, Liferay y Stringbeans.
- ✓ Fácil integración de informes hechos con JasperReports.
- ✓ Licencia LGPL. Es posible desarrollar aplicaciones comerciales con OpenXava.
- ✓ Todas las etiquetas y mensajes están en inglés, español, alemán, polaco, indonesio, francés y catalán.

1.4.2 Spring.

Es un contenedor ligero para la construcción de aplicaciones empresariales que proporciona un soporte comprensivo para el desarrollo de sistemas. Spring framework es modular, por lo cual permite utilizar solo lo necesario, además no es intrusivo, es decir se integra fácilmente en los proyectos y promueve el desarrollo con una alta cohesión (Especialización de clases) y bajo acoplamiento (Código menos dependiente).

Spring contiene muchas características que le dan una funcionalidad muy amplia; dichas características están organizadas en grandes módulos como se puede observar en el siguiente diagrama. Seguidamente se describen las características de algunos módulos (9).

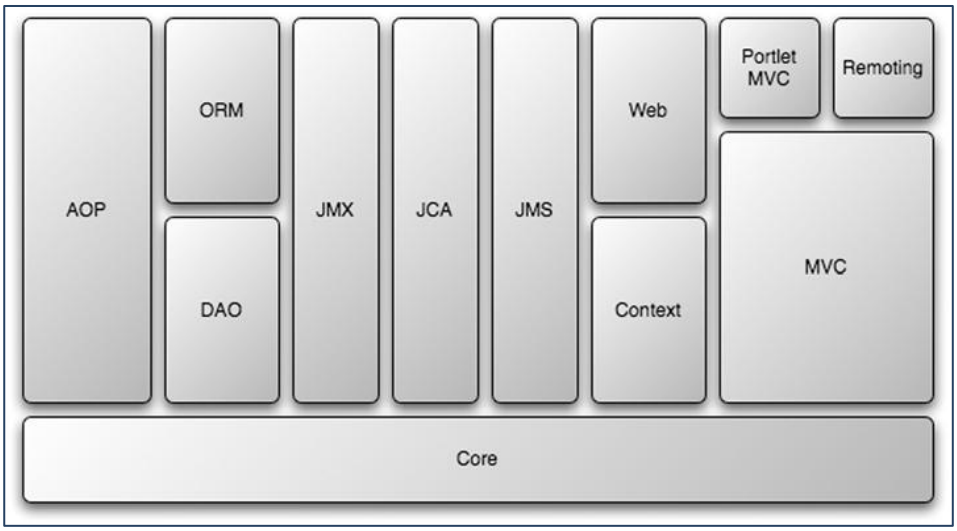


Fig. 6 Módulos de Spring

La parte fundamental del framework es el módulo **Core o "Núcleo",** este provee toda la funcionalidad de Inyección de Dependencias permitiéndole administrar la funcionalidad del contenedor de beans. El concepto básico de este módulo es el BeanFactory, que implementa el patrón de diseño Factory (fábrica) eliminando la necesidad de crear singletons programáticamente permitiéndole desligar la configuración y especificación de las dependencias de la lógica de programación (9).

Encima del módulo core se encuentra el módulo **Context (Contexto),** el cual provee de herramientas para acceder a los beans de una manera elegante, similar a un registro JNDI. El paquete de contexto hereda sus características del paquete de beans y añade soporte para mensajería de texto, como son resourcebundles (para internacionalización), propagación de eventos, carga de recursos y creación transparente de contextos por contenedores (como el contenedor de servlets, por ejemplo) (9).

El paquete **DAO** provee una capa de abstracción de JDBC que elimina la necesidad de teclear código JDBC tedioso y redundante así como el parseo de códigos de error específicos de cada proveedor de base de datos. También, el paquete JDBC provee de una manera de administrar transacciones tanto declarativas como programáticas, no solo para clases que implementen interfaces especiales, pero para todos sus POJOs (por sus siglas en inglés, Viejos y simples objetos java) (9).

El paquete **ORM** provee capas de integración para APIs de mapeo objeto - relacional, incluyendo, JDO, Hibernate e iBatis. Usando el paquete ORM es posible usar esos mapeadores en conjunto con otras características que Spring ofrece, como la administración de transacciones mencionada con anterioridad (9).

El paquete **AOP** provee una implementación de programación orientada a aspectos compatible con AOP Alliance, permitiendo definir pointcuts e interceptores de métodos para desacoplar el código de una manera limpia implementando funcionalidad que por lógica y claridad debería estar separada. Usando metadatos a nivel de código fuente se pueden incorporar diversos tipos de información y comportamiento al código, un poco similar a los atributos de .NET (9).

El paquete **Web** provee características básicas de integración orientadas a la web, como funcionalidad multipartes (para realizar la carga de archivos), inicialización de contextos mediante servletlisteners y un contexto de aplicación orientado a web. Cuando se usa Spring junto con WebWork o Struts, este es el paquete que te permite una integración sencilla (9).

El paquete Web **MVC** provee de una implementación Modelo - Vista - Controlador para las aplicaciones web. La implementación de Spring MVC permite una separación entre código de modelo de

dominio y las formas web y permite el uso de otras características de Spring Framework como lo es la validación (9).

1.4.3 Selección del framework de desarrollo.

Para seleccionar el framework de desarrollo se tuvo en cuenta que se contaba con poco tiempo para desarrollar la solución, con un equipo de trabajo reducido y que se necesitaba disminuir el tamaño de las interfaces para mejorar el rendimiento. A partir de esto se decidió utilizar OpenXava para desarrollar la solución por su facilidad de aprendizaje y de uso. Además tiene una alta productividad en el desarrollo de aplicaciones de gestión pues a partir de anotaciones se genera la vista y el acceso a datos. La vista generada por OpenXava usa pocas imágenes y sólo usa JavaScript para la comunicación AJAX con el servidor, lo que garantiza un tamaño reducido de esta.

1.5 Gestores de bases de datos.

Es un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización.

1.5.1 PostgreSQL.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD (Berkeley Software Distribution) para Bases de Datos y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones tiene prácticamente todo lo que poseen los gestores comerciales.

Algunas características de PostgreSQL son (10):

- ✓ Atomicidad (Indivisible) es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
- ✓ Consistencia es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.
- ✓ Aislamiento es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generarán ningún tipo de error.
- ✓ Durabilidad es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

- ✓ Corre en casi todos los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows, etc.
- ✓ Documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios.
- ✓ Comunidades muy activas, varias comunidades en castellano.
- ✓ Bajo “Costo de Propiedad Total” (TCO) y rápido “Retorno de la Inversión Inicial” (ROI)
- ✓ Altamente adaptable a las necesidades del cliente.
- ✓ Soporte nativo para los lenguajes más populares del medio: PHP, C, C++, Perl, Python, Java, etc.
- ✓ Drivers: Odbc, Jdbc, .Net, etc.
- ✓ Soporte de todas las características de una base de datos profesional (triggers, procedimientos almacenados, funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas, etc.)
- ✓ Soporte de tipos de datos de SQL92 y SQL99.
- ✓ Soporte de protocolo de comunicación encriptado por SSL
- ✓ Extensiones para alta disponibilidad, nuevos tipos de índices, datos espaciales, minería de datos, etc.

1.5.2 MySQL.

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQLAB — desde enero de 2008 una subsidiaria de Sun Microsystems —desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C (11).

A continuación se muestran algunas características de MySQL:

- ✓ El principal objetivo de MySQL es velocidad y robustez.
- ✓ Soporta gran cantidad de tipos de datos para las columnas.
- ✓ Gran portabilidad entre sistemas, puede trabajar en distintas plataformas y sistemas operativos.
- ✓ Cada base de datos cuenta con 3 archivos: Uno de estructura, uno de datos y uno de índice y soporta hasta 32 índices por tabla.
- ✓ Aprovecha la potencia de sistemas multiproceso, gracias a su implementación multihilo.

- ✓ Flexible sistema de contraseñas y gestión de usuarios, con un muy buen nivel de seguridad en los datos.
- ✓ El servidor soporta mensajes de error en distintas lenguas.

1.5.3 Selección del gestor de bases de datos.

PostgreSQL se usará en el desarrollo del módulo pues tiene prácticamente todo lo que poseen los gestores comerciales y tiene soporte nativo para Java. Incluye ordenamientos, en memoria y en disco, rápidos y optimización de consultas sobre datos particionados. Permite a los administradores crear fácilmente una copia para recuperación inmediata de su clúster de bases de datos. Con el uso de este gestor se busca compatibilidad con la primera versión del sistema para la migración de los datos.

1.6 Entornos integrados de desarrollo (IDE).

Un entorno integrado de desarrollo es un programa informático compuesto por un conjunto de herramientas de programación, en su ambiente de trabajo pueden utilizarse uno o varios lenguajes, por lo que son empaquetados en una única aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDE pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Es posible trabajar con varios lenguajes de programación en un mismo IDE como es el caso de Eclipse y NetBeans.

1.6.1 Eclipse.

Eclipse es un Entorno Integrado de Desarrollo, del inglés Integrated Development Environment (IDE), para todo tipo de aplicaciones libres, inicialmente desarrollado por IBM, y actualmente gestionado por la Fundación Eclipse. Es una herramienta para el programador desarrollada principalmente para el desarrollo de aplicaciones Java, facilitando al máximo la gestión de proyectos colaborativos mediante el control de versiones 'cvs', es posible también con subversión, exportar e importar proyectos. Es posible añadir nuevas funcionalidades al editor, a través de nuevos módulos ('plugins'), para programar en otros lenguajes de programación además de Java como C/C++, PHP, Python, Ruby (12). Eclipse tiene una buena integración con OpenXava y la interfaz de usuario está organizada en vistas y perspectivas.

1.6.2 NetBeans.

NetBeans IDE es un entorno de desarrollo - una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Con este IDE puede programarse en varios lenguajes

como Java y PHP. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso (13).

NetBeans es de código abierto y gratuito para uso tanto comercial como no comercial. El código fuente está disponible para su reutilización de acuerdo con la Common Development and Distribution License (CDDL) v1.0 and the GNU General Public License (GPL) v2 (13).

Junto con el soporte para Java EE 6 y GlassFish v3, el NetBeans IDE 6.8 ofrece otras nuevas características y mejoras que incluyen:

- ✓ Soporte PHP Ampliado: Expande el soporte de los lenguajes dinámicos con apoyo para PHP 5.3 y el esquema de Symfony acelera el desarrollo de aplicaciones web PHP.
- ✓ Mejora de C / C + + Profiling: Perfila y sintoniza aplicaciones C / C + + con el nuevo indicador Microstate Accounting, supervisor de uso I/O.
- ✓ JavaFX ™: Código de finalización mejorado, sugerencias y navegación para JavaFX en el editor NetBeans.

1.6.3 Selección del entorno integrado de desarrollo (IDE).

Se decidió utilizar Eclipse porque su interfaz de usuario está organizada en vistas y perspectivas y se puede personalizar. Esto permite un acceso más fácil a las herramientas que se necesitan e incrementa la productividad. A diferencia del NetBeans, Eclipse se integra bien con OpenXava y el equipo de desarrollo ya está familiarizado con el uso de este IDE.

1.7 Servidores web.

Un servidor web es un programa que sirve para atender y responder a las diferentes peticiones de los navegadores, proporcionando los recursos que soliciten usando el protocolo HTTP o el protocolo HTTPS (la versión cifrada y autenticada). Un servidor web básico cuenta con un esquema de funcionamiento muy simple, basado en ejecutar infinitamente el siguiente bucle:

1. Espera peticiones en el puerto TCP indicado (el estándar por defecto para HTTP es el 80).
2. Recibe una petición.
3. Busca el recurso.
4. Envía el recurso utilizando la misma conexión por la que recibió petición.
5. Vuelve al segundo punto.

Un servidor web que siga el esquema anterior cumplirá todos los requisitos básicos de los servidores HTTP, aunque sólo podrá servir ficheros estáticos.

A partir del anterior esquema se han diseñado y desarrollado todos los servidores de HTTP que existen, variando sólo el tipo de peticiones (páginas estáticas, CGIs, Servlets, etc.) que pueden atender, en función de que sean o no sean multi-proceso o multi-hilados, etc.

1.7.1 GlassFish.

GlassFish es un servidor de aplicaciones compatible con Java EE, se distribuye bajo la licencia CDDL y la GNU GPL. GlassFish v3 proporciona una pequeña base con todas las funciones para la implementación de Java EE 6 (14).

La plataforma Java EE 6 mejora significativamente la productividad del desarrollador, presenta el perfil web ligero para aplicaciones web e incluye las últimas versiones de tecnologías como JAX-RS 1.1, Java Server Faces(JSF) 2.0, Enterprise JavaBeans (EJB) 3.1, Java Persistence (App) 2.0, Context and Dependency Injection (CDI) 1.0. GlassFish es un servidor de aplicaciones de código abierto para Java EE (14).

GlassFish usa código de Sun Java Application Server y de la arquitectura TopLink de Oracle. Entre las características más notables de GlassFish cabe destacar su velocidad, alta escalabilidad, manejo centralizado de clusters e instancias, bajo consumo de memoria, interoperabilidad con .NET 3 y un excelente panel de administración.

1.7.2 Apache Tomcat.

Tomcat es un servidor HTTP y un contenedor de servlets. Es software libre (licencia Apache 2.0) gestionado por la fundación Apache. Puede funcionar como servidor HTTP o conectado a otro servidor HTTP como Apache HTTP Server o IIS. Puede ejecutar servicios web mediante Apache Axis (15). Al ser un contenedor web consume menos recursos que un servidor de aplicaciones.

A continuación se muestran algunas características de la versión 6:

- ✓ Implementado de Servlet 2.5 y JSP 2.1.
- ✓ Soporte para Unified Expression Language 2.1.
- ✓ Diseñado para funcionar en Java SE 5.0 y posteriores.
- ✓ Soporte para Comet a través de la interfaz CometProcessor.

1.7.3 Selección del servidor web.

Se escogió Tomcat pues es un contenedor web fácil de administrar y configurar, no es necesario el uso de un servidor de aplicaciones, ya que no se hará uso alguno de los EJB. Consume menos recursos

que un servidor de aplicaciones. Permite lograr escalabilidad mediante clúster. Este contenedor presenta un manejador de reserva de conexiones que se configura de manera que desde la aplicación se acceden a estas conexiones, este mecanismo es más eficiente que el de realizar la conexión en el momento que requiera.

1.8 Frameworks para pruebas unitarias.

Encontrar y corregir errores es una tarea necesaria para mejorar la calidad en el desarrollo de software. Para facilitar esta tarea están los Frameworks para Pruebas Unitarias que proporcionan una manera sencilla, rápida y elegante para escribir pruebas y validarlas automáticamente.

1.8.1 JUnit.

Para averiguar si todo está desarrollado según lo estipulado, JUnit analiza si al ejecutar algún proceso, éste ha devuelto un resultado correcto según los datos que se le han proporcionado. Esto ayuda a encontrar errores y a subsanarlos, algo que es conveniente en desarrollos complicados donde un pequeño error puede resultar realmente difícil de encontrar.

El funcionamiento de JUnit siempre va unido a algún entorno de desarrollo real, ya que este software es tan sólo una librería de la que se pueden leer y ejecutar instrucciones, pero que por sí misma no permite ejecutar nada, aunque incluye varias formas de visualización de los resultados. Algunas características de este framework son:

- ✓ Es una herramienta de código abierto.
- ✓ Multitud de documentación y ejemplos en la web.
- ✓ Se ha convertido en el estándar de hecho para las pruebas unitarias en Java.
- ✓ Soportado por la mayoría de los IDEs como Eclipse o NetBeans.
- ✓ Es una implementación de la arquitectura xUnit para los frameworks de pruebas unitarias.
- ✓ Posee una comunidad mucho mayor que el resto de los frameworks de pruebas en Java.
- ✓ Soporta múltiples tipos de aserciones.
- ✓ Desde la versión 4 utiliza las anotaciones del JDK 1.5 de Java.
- ✓ Posibilidad de crear informes en HTML.
- ✓ Organización de las pruebas en Suites de pruebas.

1.8.2 JTiger.

JTiger es un marco de prueba que provee una solución robusta para realizar pruebas de unidad y no requiere archivos externos de configuración para ejecutar las pruebas. A continuación se muestran algunas características de JTiger:

- ✓ Framework de pruebas unitarias para Java (1.5).
- ✓ Es de código abierto.
- ✓ Capacidad para exportar informes en HTML, XML o texto plano.
- ✓ Es posible ejecutar casos de prueba de JUnit mediante un plugin.
- ✓ Los metadatos de los casos de prueba son especificados como anotaciones del lenguaje Java.
- ✓ El Framework incluye pruebas unitarias sobre sí mismo.

1.8.3 Selección del framework para pruebas unitarias.

Se decidió utilizar JUnit porque es una de las herramientas de pruebas más extendidas y por esta razón es fácil encontrar información sobre su uso, así como ejemplos y tutoriales. Además se puede integrar un plugin para JUnit a Eclipse que simplifica la creación de pruebas y la ejecución de las mismas.

1.9 Herramientas para generar reportes.

Las Herramientas para generar reportes son librerías que a partir de una fuente de datos posibilitan la generación de reportes. Estos son una manera organizada de mostrar la información procedente de una fuente de datos. Los reportes hacen más fácil el análisis de la información y la toma de decisiones.

1.9.1 IText.

IText es una librería Java gratuita para la generación de documentos PDF de forma dinámica. Fue escrita por Bruno Lowagie, Paulo Soares, y otros; está distribuida bajo la Mozilla Public License con la LGPL como licencia alternativa.

Esta librería es utilizada para generar reportes de formatos planos sin campos complejos, aunque tiene herramientas para mejorarlos. Entre sus características tiene la firma digital de documentos PDF y corrección de colores. Además cuenta con un modelo de eventos para el documento, página y tabla que posibilita generar encabezados y pies de página dinámicos, centralizar los estáticos y realizar otras funciones.

1.9.2 JasperReports.

JasperReports es una librería que está diseñada para agregar capacidades de reporte a las aplicaciones Java. La misma permite organizar la información obtenida desde una base de datos relacional, mediante conectores JDBC y organizarla en diseños de reportes previamente definidos en un formato XML.

A continuación se muestran algunas características de JasperReports:

- ✓ Permite una diagramación flexible de los reportes: Los reportes se pueden dividir en secciones opcionales que son: título del reporte, el encabezado de página, una sección para los detalles del reporte, el pie de página y una sección de resumen que aparece al final del reporte.
- ✓ Permite que los desarrolladores le surtan datos en varias formas: esto es que los desarrolladores pueden pasar datos a los reportes por medio del paso de parámetros. Estos parámetros de reportes pueden ser instancia de cualquier clase de Java.
- ✓ Puede generar sub-reportes: JasperReports permite la creación de reportes dentro de reportes lo que facilita bastante el diseño porque es posible usar estos sub-reportes en otros reportes.
- ✓ Los reportes son capaces de presentar los datos de manera textual o a través de gráficos: no sólo son capaces de mostrar los datos que le son pasados sino que pueden generar o calcular con esos datos otros datos de forma dinámica y mostrarlos.
- ✓ Puede generar marcas de agua: JasperReports permite generar textos o imágenes de fondo para utilizarlo como marcas de agua con el propósito de identificar el reporte o simplemente por motivos de seguridad.
- ✓ Se pueden exportar los reportes a una multitud de formatos: Los reportes generados con JasperReports pueden ser exportados a una multitud de formatos como PDF, XLS, RTF, HTML, XML, CVS (valores separados por coma) y texto plano.
- ✓ Cuenta con herramientas para el diseño de los reportes: Se usa comúnmente con iReports, una herramienta gráfica de código abierto para la edición de informes que evita todo el trabajo del diseño de los reportes a nivel de código.

1.9.3 Selección dela Herramienta para generar reportes.

Para la generación de reportes se seleccionó a JasperReports por la facilidad con que se integra a OpenXava. Este framework brinda una acción abstracta para generar los reportes con JapeReports, sólo hay que implementar métodos para obtener los datos y el diseño del reporte.

1.10 Conclusiones parciales.

En este capítulo se realizó el estudio de Metodologías de Desarrollo de Software, Lenguajes de Programación y tecnologías usadas en el desarrollo de aplicaciones web. Esto permitió consolidar el basamento teórico para escoger la metodología y las tecnologías adecuadas según las características del módulo a desarrollar. Sentándose las bases para comenzar la implementación del módulo Misiones.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

2.1 Introducción.

En este capítulo se muestra el uso de patrones de diseño en el desarrollo de la solución así como las convenciones de código y seguridad. Se explican las medidas tomadas para garantizar un mejor rendimiento a partir de un estudio de la versión anterior. Además se muestran los diagramas de componentes del módulo y el modelo de datos que se obtiene a partir de las anotaciones en las clases del modelo. Por último se describe el diagrama de componentes y las clases del modelo.

2.2 Valoración crítica del diseño propuesto por los analistas.

A partir del diseño propuesto por los analistas (ver anexo 1) para la solución del módulo Misiones, se llegó a una mejor comprensión de los requisitos funcionales (ver anexo 3), logrando de esta manera adentrarse en las especificidades de los requisitos no funcionales, las restricciones de los lenguajes de programación y las ventajas que da la multiplataforma. También se logró buscar un punto congruente para iniciar la implementación de las clases.

2.3 Patrones de diseño.

Se utilizaron los patrones GRASP para la asignación de responsabilidades. Esto permite que el sistema sea más robusto, más fácil de mantener, entender, reutilizar y extender.

- ✓ **Experto**

  Un modelo de clase puede definir docenas y hasta cientos de clases de software, y una aplicación tal vez requiera el cumplimiento de cientos o miles de responsabilidades. Durante el diseño orientado a objetos, cuando se definen las interacciones entre los objetos, tomamos decisiones sobre la asignación de responsabilidades a las clases. Si se hacen en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se nos presenta la oportunidad de reutilizar los componentes en futuras aplicaciones (16).

  El patrón experto se basa en asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

  En el módulo se asignó responsabilidades a las clases según la información a la que tienen acceso. Las acciones tienen acceso a la vista por lo tanto según su tipo pueden crear o modificar solicitudes de viaje, realizar búsquedas o filtrar información.

- ✓ **Creador**

  La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia, conviene contar con un principio general para asignar las

responsabilidades concernientes a ella. El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reusabilidad.

El patrón creador se basa en asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos (16):

- B agrega los objetos A.
- B contiene los objetos A.
- B registra las instancias de los objetos A.
- B utiliza específicamente los objetos A.
- B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A).

Se aplica para las acciones que heredan de SaveAction. Estas clases son responsables de la creación de objetos ya que son las que tienen acceso en la vista a la información necesaria para crearlos.

- ✓ **Alta cohesión**

En la perspectiva del diseño orientado a objetos, la cohesión (o, más exactamente, la cohesión funcional) es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. No conviene este tipo de clases pues presentan los siguientes problemas:

- Son difíciles de comprender.
- Son difíciles de reutilizar.
- Son difíciles de conservar.
- Son delicadas: las afectan constantemente los cambios.

Las clases con baja cohesión a menudo representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otros objetos (16).

El patrón alta cohesión se basa en asignar una responsabilidad de modo que la cohesión siga siendo alta.

En el módulo las clases tienen responsabilidades bien definidas. Los calculadores permiten calcular de forma automática algunos campos y los validadores comprueban que los valores de los

campos tengan el formato requerido. Las acciones se apoyan en la clase MapFacade para insertar, modificar y eliminar datos.

- ✓ **Bajo acoplamiento**

  El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras; "muchas otras" depende del contexto, pero no lo estudiaremos aquí por el momento. Una clase con alto (o fuerte) acoplamiento recurre a muchas otras. Este tipo de clases no es conveniente: presentan los siguientes problemas (16):

  - Los cambios de las clases afines ocasionan cambios locales.
  - Son más difíciles de entender cuando están aisladas.
  - Son más difíciles de reutilizar porque se requiere la presencia de otras clases delas que dependen.

  El patrón bajo acoplamiento se basa en asignar una responsabilidad para mantener bajo acoplamiento.

  En el módulo es bajo el número de clases con que una clase se relaciona. Es raro el caso en el que un validador o una acción se relacionen con otra clase además de su clase padre.

- ✓ **Controlador**

  Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación (16).

  El patrón controlador se basa en asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones:

  - El "sistema" global (controlador de fachada).
  - La empresa u organización global (controlador de fachada).
  - Algo en el mundo real que es activo (por ejemplo, el papel de una persona)y que pueda participar en la tarea (controlador de tareas).
  - Un manejador artificial de todos los eventos del sistema de un caso de uso, generalmente denominados "Manejador<NombreCasodeUso>"(controlador de casos de uso).

  En la solución los controladores están compuestos de acciones que son las encargadas de responder ante los eventos del sistema.

```
<controlador nombre="VerEstadosSolicitudesEmbajada">
    <accion nombre="verDetalles" clase="org.openxava.actions.ViewDetailAction"
        oculta="true" modo="list" imagen="images/view.gif">
        <poner propiedad="nextAction" valor="Mode.detail" />
    </accion>
    <accion nombre="buscarSolicitudViajeEstados" oculta="true"
        al-iniciar="true" clase="misiones.acciones.FiltrarSolicitudViajeEstados">
        <poner propiedad="modulo" valor="9" />
    </accion>
    <accion nombre="buscarSolicitudViajeNivel" oculta="true"
        al-iniciar="true" clase="misiones.acciones.FiltrarSolicitudViajeNivel">
    </accion>
    <accion nombre="Cancelar" modo="detail"
        clase="base.acciones.CambiarModoAccion">
        <poner propiedad="modo" valor="list" />
    </accion>
</controlador>
```

Fig. 7 Controlador VerEstadosSolicitudesEmbajada

En la implementación también se usó el patrón estructural **fachada**, perteneciente al grupo de los GOF, cuya intención es proveer una interfaz unificada para un conjunto de clases en un subsistema. Simplifica el acceso a dicho conjunto de clases, ya que el cliente sólo se comunica con ellas a través de una única interfaz.

Este patrón se usa en la gestión de los objetos del modelo con la clase MapFacade. Esta clase brinda un conjunto de métodos que hacen más fácil la gestión de los objetos del modelo en la base de datos. Permite manejar los objetos en el formato de la clase Map que es el formato con que se obtienen los datos de la vista lo que hace más fácil el trabajo.

2.4 Estándares de código.

Un estándar de codificación es un conjunto de directrices, normas y reglamentos sobre la forma de escribir código. Por lo general, un estándar de codificación incluye pautas sobre cómo nombrar variables, la forma de guión, el código, cómo poner entre paréntesis y palabras clave, etc. La idea es ser constante en la programación de modo que, en caso de que haya varias personas trabajando en el mismo código, se hace más fácil para uno entender lo que los demás han hecho. Los estándares utilizados fueron:

2.4.1 Notación CamelCasing

Se utilizó **Upper CamelCasing** para los nombres de las clases. Estos comienzan con mayúscula y si están compuestos por más de una palabra cada una comenzará también con mayúscula, por ejemplo: AceptarEnvioContrato.

Lower CamelCasing se utilizó para los nombres de las variables y los métodos. Esta notación es igual al Upper CamelCasing excepto que la letra inicial es minúscula.

2.4.2 Identación

La identación es usada para tener una mejor visibilidad en el diseño de un programa. La identación muestra las líneas que están subordinadas a otras líneas. Por ejemplo, todas las líneas que forman el cuerpo de un ciclo deberán estar identadas con la instrucción principal del ciclo.

Se deben emplear cuatro espacios como unidad de indentación. La construcción exacta de la indentación (espacios en blanco contra tabuladores) no se especifica. Los tabuladores deben ser exactamente cada 8 espacios.

```
public String getActionToHide(){
int idestado = getView().getValueInt("estado.idestado");
String[] estados = estados_solicitud_ver_nota_rechazo.split(",");
for (int i = 0; i <estados.length; i++)
if (Integer.toString(idestado).equals(estados[i]))
      return null;
return accion;
}
```

2.5 Seguridad.

Para contribuir a la seguridad de la aplicación se usaron los validadores que proporciona OpenXava para validar los campos de los formularios.

```
@PropertyValidator(value = misiones.validadores.ValidadorReferenciaCarta.class)
```

Además no se concatenó variables a las consultas para evitar las inyecciones SQL. Para pasarle parámetros a las consultas se usó la clase PreparedStatement.

```
con = DataSourceConnectionProvider.getByComponent("Flujo").getConnect
String query = "SELECT estado_siguiente FROM nomencladores.flujo " +
        "WHERE estado_actual = ? and operacion = ? and proces_oactual

PreparedStatement ps = con.prepareStatement(query);
ps.setInt(1, estadoActual);
ps.setInt(2, operacionRealizada);
ps.setInt(3, proceso);
ps.setInt(4, parte);

ResultSet rs = ps.executeQuery();
```

Fig. 8 Ejemplo del uso de la clase PreparedStatement.

2.6 Medidas adoptadas para aumentar el rendimiento del módulo Misiones.

Para la implementación de la solución se tuvo en cuenta aspectos del sistema anterior que influyeron negativamente en su rendimiento y se tomaron algunas medidas para garantizar que el módulo Misiones tuviera un mejor rendimiento. El uso de OpenXava posibilitó la reducción del tamaño de las interfaces de usuario. En la primera versión todo el código JavaScript del módulo estaba en un solo archivo con 4591 líneas de código y un tamaño de 200 kb. Con el uso del FireBug se determinó el tamaño de las principales interfaces de la primera versión y de la solución, lo que demostró una significativa reducción del tamaño de las interfaces de la solución con respecto a las interfaces de la primera versión.

Interfaz	Primera versión	Solución
Administrar solicitud	**880**	**231**
Enviar solicitud	**890**	**225**
Revisar solicitud	**850**	**230**
Procesar solicitud	**855**	**227**
Actualizar registro Misiones	**870**	**226**
Dar entrada al informe de viaje	**850**	**226**
Reporte carta aprobación	**850**	**226**

Tabla 1 Tamaño de las principales interfaces de la primera versión y la solución (kb)

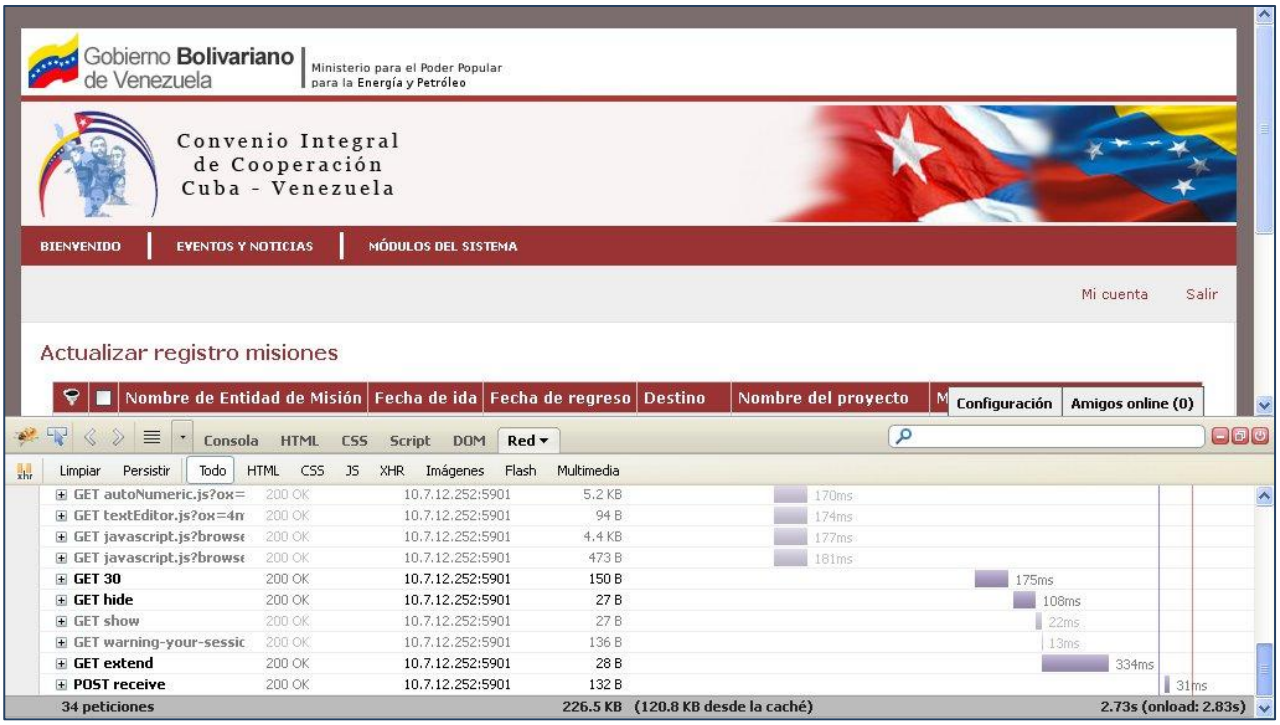


Fig. 9 Interfaz Actualizar registro de Misiones de la solución

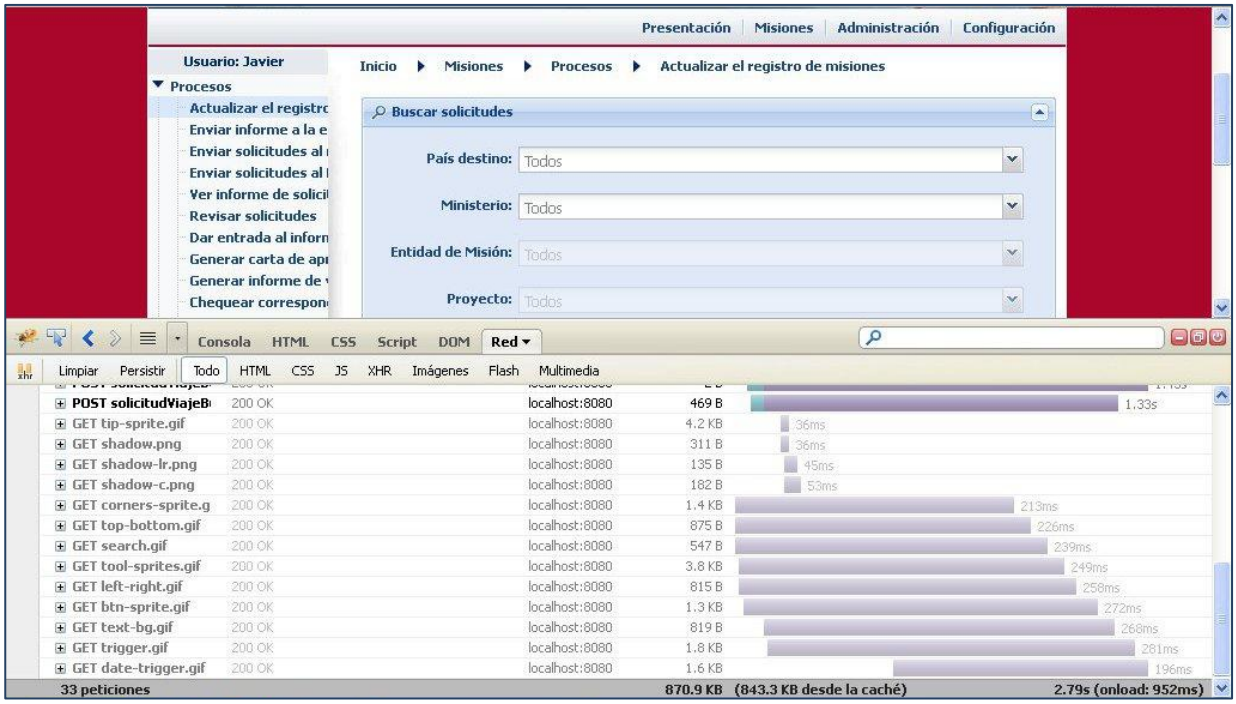


Fig. 10 Interfaz Actualizar registro de Misiones de la primera versión

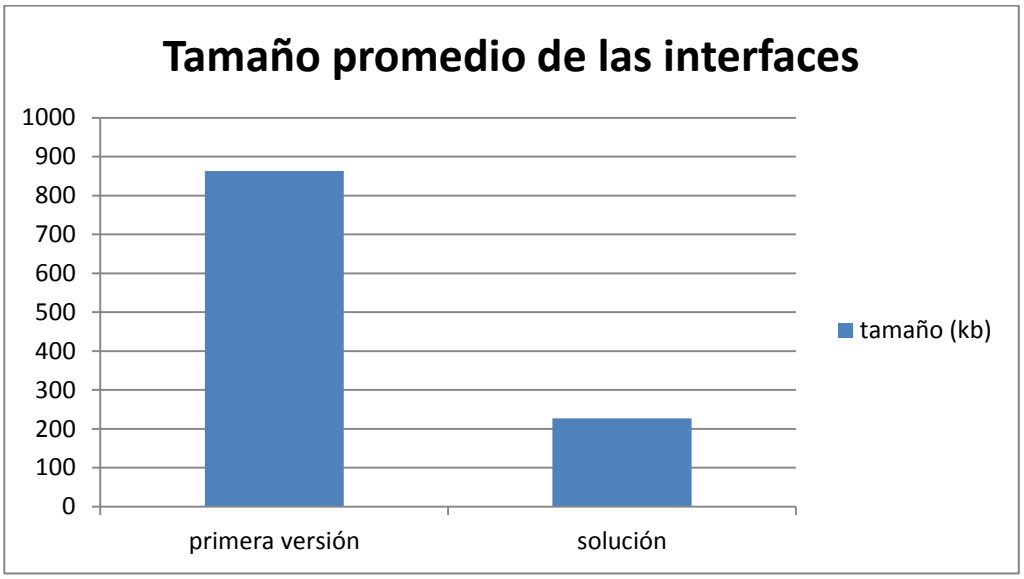


Fig. 11 Gráfica del tamaño promedio de las interfaces de la primera versión y de la solución.

Mediante anotaciones se definió la carga perezosa (Lazy Load) para los objetos de las clases del modelo, que consiste en realizar una carga parcial del objeto, posponiendo la carga de algunos de sus componentes hasta que realmente el programa acceda a ellos. De esta forma se garantiza un menor tiempo de carga de las interfaces ya que los objetos solo se cargan realmente en el momento que se necesitan.

```
@Required
@ManyToOne(fetch=FetchType.LAZY)
@JoinColumn(name="tmision")
@DescriptionsList(descriptionProperties="nombre")
@NoModify
@NoCreate
@ReadOnly(forViews="vista_gestionar_solicitud,vist
private TipoMision tmision;
```

Fig. 12 Ejemplo de un atributo con la anotación para la carga perezosa.

Es muy común en el módulo la presentación de listados de solicitudes de viaje cada vez que un usuario hace una búsqueda. Para evitar que se carguen solicitudes que no se necesitan y reducir el

tiempo de carga de la interfaz se filtran las solicitudes de viaje por estado y por nivel. Como resultado al usuario solo se le muestran las solicitudes que pertenecen a su institución o ministerio y de estas sólo se muestran las que estén un estado que sea de interés en el caso de uso que se está ejecutando.

Se hizo un estudio de la complejidad de los métodos más importantes de la primera versión con el objetivo de no incrementar la complejidad temporal de estos algoritmos en la solución o reducirla en caso de que fuera necesario. Los métodos más significativos de la primera versión no tenían una gran complejidad ($O\ n$ o menor) excepto el método generarInformeViajeEmbajadas con una complejidad $O(nm)$.

```
for (DSolicitud_Viaje solicitud_Viaje : solicitudes) {
    Date fechaIda = solicitud_Viaje.getFechaIda();
    long cantidadDias = Math.abs((solicitud_Viaje.getFechaRegreso().getTime() - fechaIda.getTime()))
    String fechaViaje = formatoFechaCartaSolicitud
            .replace("?D?", diasFormatter.format(fechaIda))
            .replace("?M?", meses.get(Integer.parseInt(mesFormatter.format(fechaIda)) - 1))
            .replace("?A?", annoFormatter.format(fechaIda))
            .replace("?CD?", Long.toString(cantidadDias));
    JSONObject obj = new JSONObject();
    Set<DPersonas_Mision> personas = solicitud_Viaje.getDPersonas_MisionidPersonasMision();
    JSONArray integrantes = new JSONArray();
    Iterator<DPersonas_Mision> persona = personas.iterator();
    String responsable = null;
    for (int i = 0; i < personas.size(); i++) {
        DPersonas_Mision aux = persona.next();
        if (aux.getJefeMision()) {
            responsable = aux.getNombreApellidos();
        }
        else
            integrantes.put(aux.getNombreApellidos());
    }
    obj.put("fechaViaje", fechaViaje);
    obj.put("motivo", solicitud_Viaje.getMotivos());
    obj.put("entidad", solicitud_Viaje.getMinisterio().getNivel().getNombre());
    obj.put("responsable", responsable);
    obj.put("personas", integrantes);
    array.put(obj);
}
```

Fig. 13 Fragmento del método generarInformeViajeEmbajadas

Estos datos se tuvieron en cuenta en el desarrollo de la solución. El método equivalente en la solución a generarInformeViajeEmbajadas es execute de la acción ReporteMisionesInformeViaje, este se implementó con una menor complejidad ($O(n)$). Lo que posibilitó esta reducción de complejidad fue la forma en que se pasaron los datos de las personas al reporte. En la primera versión mediante un ciclo se tomaba los datos de cada una de las personas para escribirlos en un arreglo JSON que luego Spring le

pasaba a JasperReports. En la solución no fue necesario este ciclo porque se le pasaba directamente la lista de personas de la solicitud al reporte. El resto de los métodos equivalentes a los estudiados en la primera versión se implementaron con la complejidad $O(n)$ o menor.

```
public void execute() throws Exception {
    ServletContext application = getRequest().getSession().getServletContext();
    System.setProperty("jasper.reports.compile.class.path",
        application.getRealPath("/WEB-INF/lib/jasperreports.jar") +
        System.getProperty("path.separator") +
        application.getRealPath("/WEB-INF/classes/")
    );

    InputStream xmlDesign = null;
    String jrxml = getJRXML();
    if (isAbsolutePath(jrxml)) {
        xmlDesign = new FileInputStream(jrxml);
    }
    else {
        xmlDesign = JasperReportBaseAction.class.getResourceAsStream("/" + jrxml);
    }
    if (xmlDesign == null) throw new XavaException("design_not_found");
    JasperReport report = JasperCompileManager.compileReport(xmlDesign);
    Map parameters = getParameters();
    JRDataSource ds = getDataSource();
    JasperPrint jprint = null;
    if (ds == null) {
        Connection con = DataSourceConnectionProvider.getByComponent(modelName).getConnection();
        if (!Is.emptyString(XPersistence.getDefaultSchema())) {
            con.setCatalog(XPersistence.getDefaultSchema());
        }
        jprint = JasperFillManager.fillReport(report, parameters, con);
        con.close();
    }
    else {
        jprint = JasperFillManager.fillReport(report, parameters, ds);
    }
    getRequest().getSession().setAttribute("xava.report.jprint", jprint);
    getRequest().getSession().setAttribute("xava.report.format", getFormat());
}
```

Fig. 14 Método execute de la acción ReporteMisionesInformeViaje

```
informe.setNombreMision(solicitud.getNombreProyecto());
informe.setNombre(solicitud.getJefeDelegacion()+" (Jefe de Delegación)");
informe.setDelegacion(new JRBeanCollectionDataSource(solicitud.getPersonas()));
informe.setFechaViaje("Fecha de Viaje: "+Util.getFecha(solicitud.getFechaIda())+" por "+Util.
informe.setMotivo("Motivo: "+solicitud.getMotivo());
collection.add(informe);
```

Fig. 15 Fragmento del método para obtener los datos de la acción ReporteMisionesInformeViaje

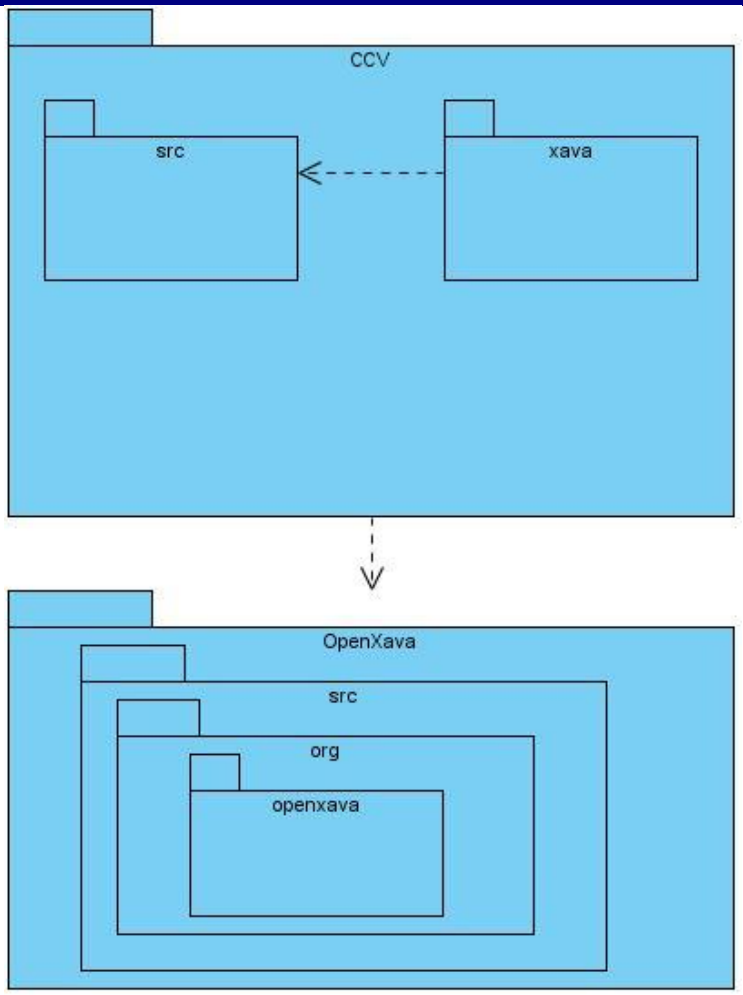
2.7 Modelo de implementación.

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

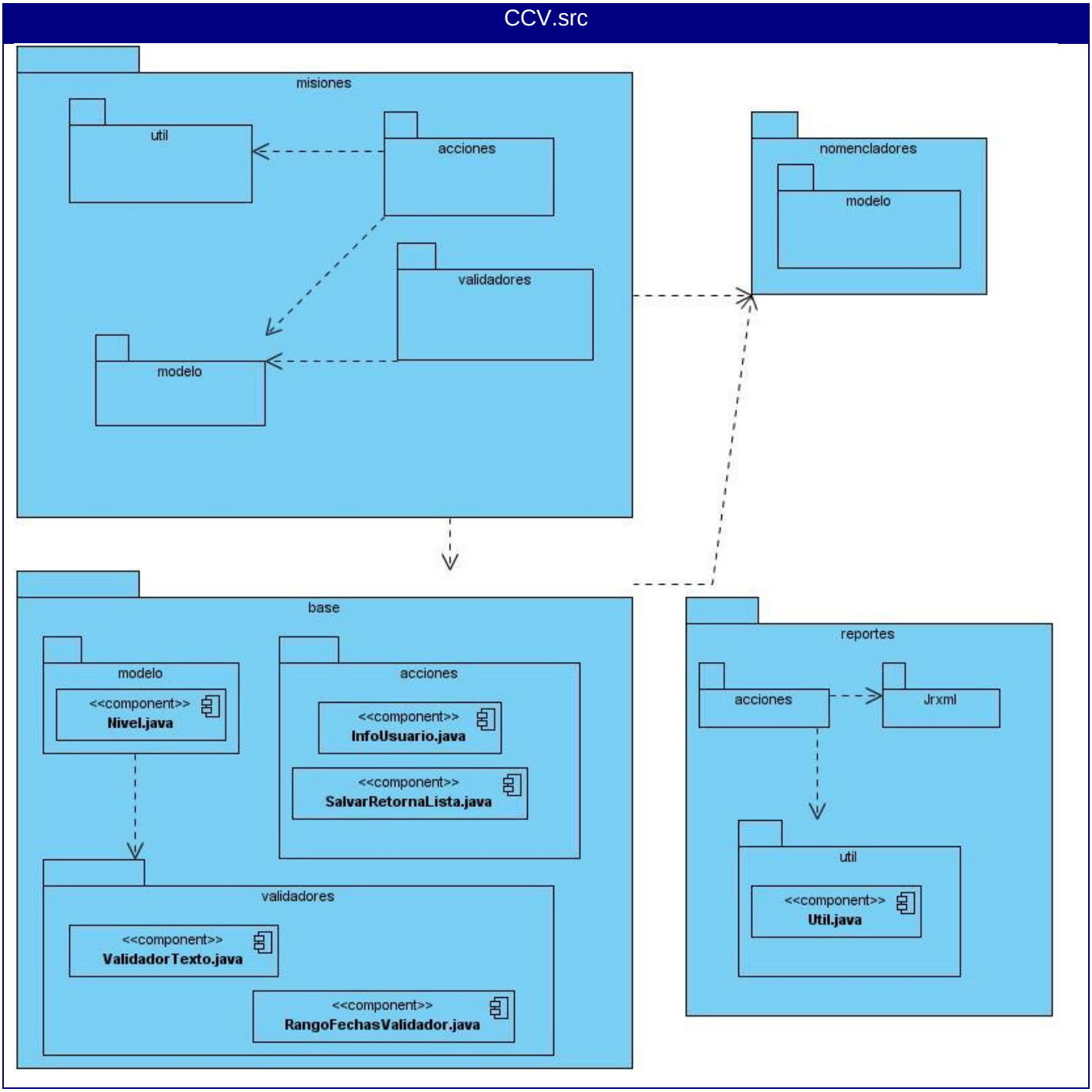
2.7.1 Diagrama de componentes.

Se representa como un grafo de componentes de software unidos por medio de relaciones de dependencia. Es el conjunto de ficheros interrelacionados entre sí para lograr la completa funcionalidad del sistema. A continuación se representa los diagramas de componentes de la aplicación.

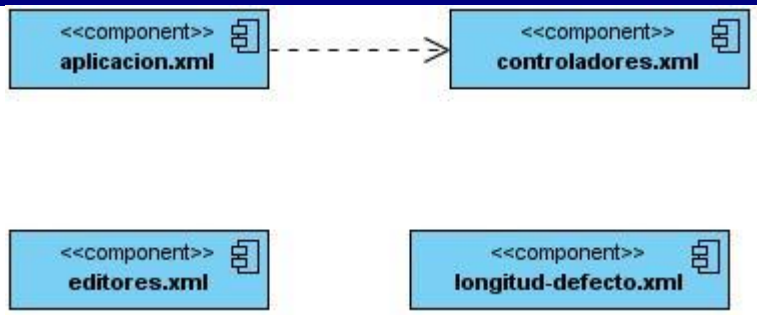
Aplicación



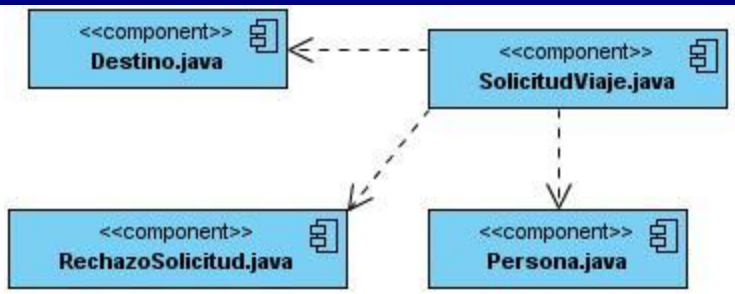
CCV.src


misiones
util
acciones
validadores
modelo
nomencladores
modelo
base
modelo
<<component>>
Nivel.java
acciones
<<component>>
InfoUsuario.java
<<component>>
SalvarRetornaLista.java
validadores
<<component>>
ValidadorTexto.java
<<component>>
RangoFechasValidador.java
reportes
acciones
Jrxml
util
<<component>>
Util.java

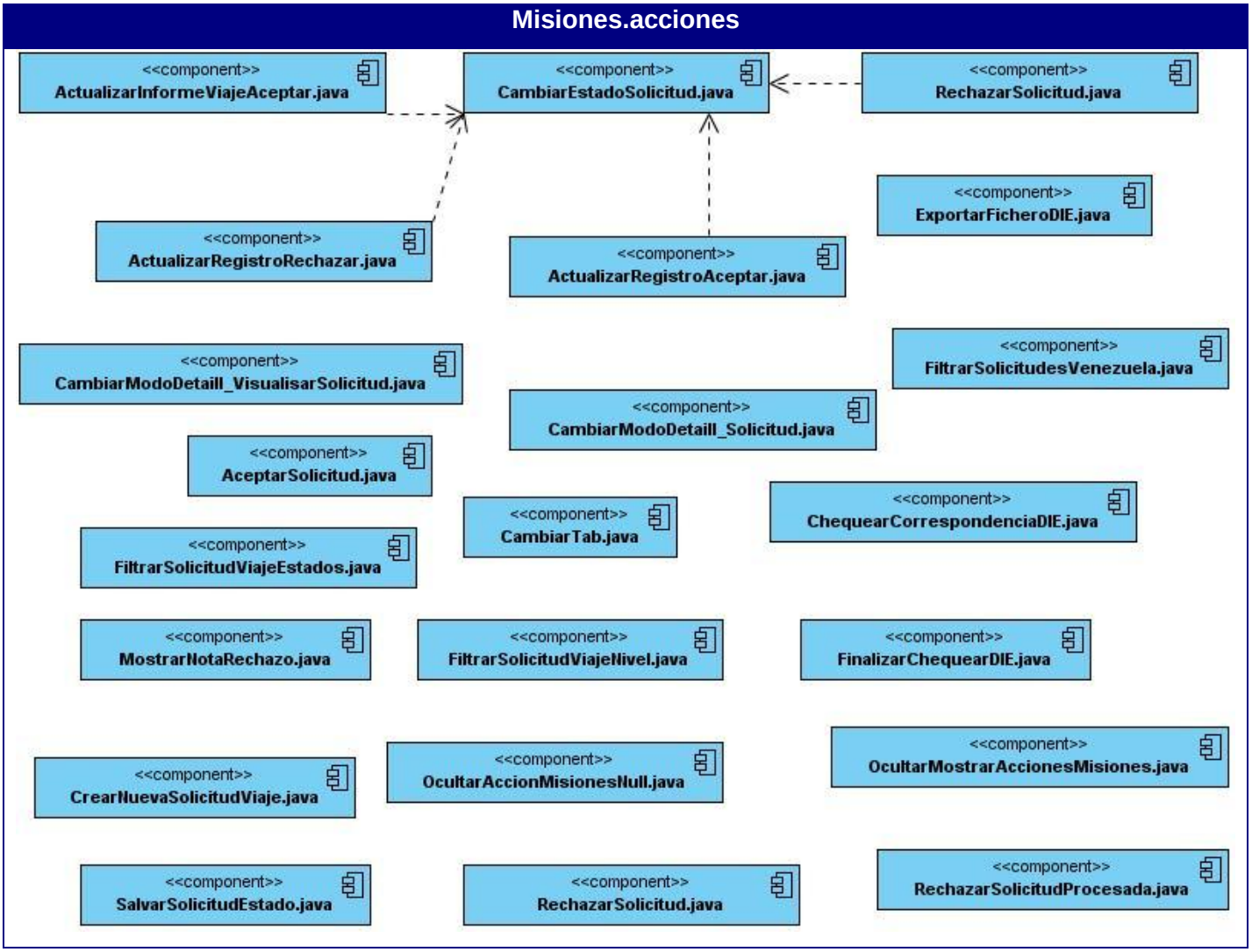
CCV.xava



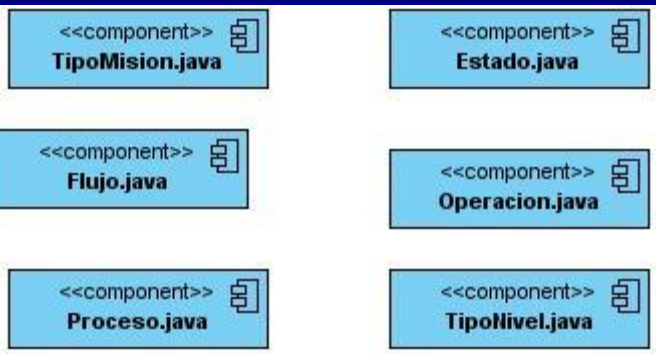
Misiones.modelo



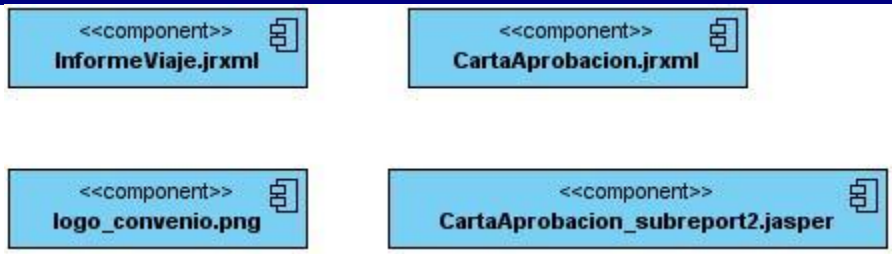
Misiones.validadores


Misiones.acciones
<<component>>
ActualizarInformeViajeAceptar.java
<<component>>
CambiarEstadoSolicitud.java
<<component>>
RechazarSolicitud.java
<<component>>
ExportarFicheroDIE.java
<<component>>
ActualizarRegistroRechazar.java
<<component>>
ActualizarRegistroAceptar.java
<<component>>
CambiarModoDetaill_VisualisarSolicitud.java
<<component>>
FiltrarSolicitudesVenezuela.java
<<component>>
CambiarModoDetaill_Solicitud.java
<<component>>
AceptarSolicitud.java
<<component>>
CambiarTab.java
<<component>>
ChequearCorrespondenciaDIE.java
<<component>>
FiltrarSolicitudViajeEstados.java
<<component>>
MostrarNotaRechazo.java
<<component>>
FiltrarSolicitudViajeNivel.java
<<component>>
FinalizarChequearDIE.java
<<component>>
CrearNuevaSolicitudViaje.java
<<component>>
OcultarAccionMisionesNull.java
<<component>>
OcultarMostrarAccionesMisiones.java
<<component>>
SalvarSolicitudEstado.java
<<component>>
RechazarSolicitud.java
<<component>>
RechazarSolicitudProcesada.java

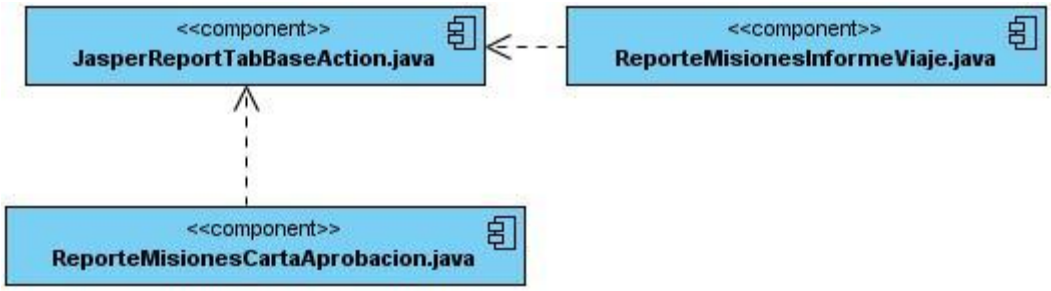
nomencladores.modelo



reportes.Jrxml



reportes.acciones



2.7.2 Diagrama de despliegue.

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación, que describe la distribución, entrega e instalación de las partes que configuran el sistema físico. Muestra las relaciones físicas de los nodos que participan en la ejecución del sistema describiendo la arquitectura física del sistema en términos de: procesadores, dispositivos y componentes de software. A continuación se muestra el diagrama de despliegue de la aplicación y la descripción de sus nodos que estarán en Venezuela y en Cuba.

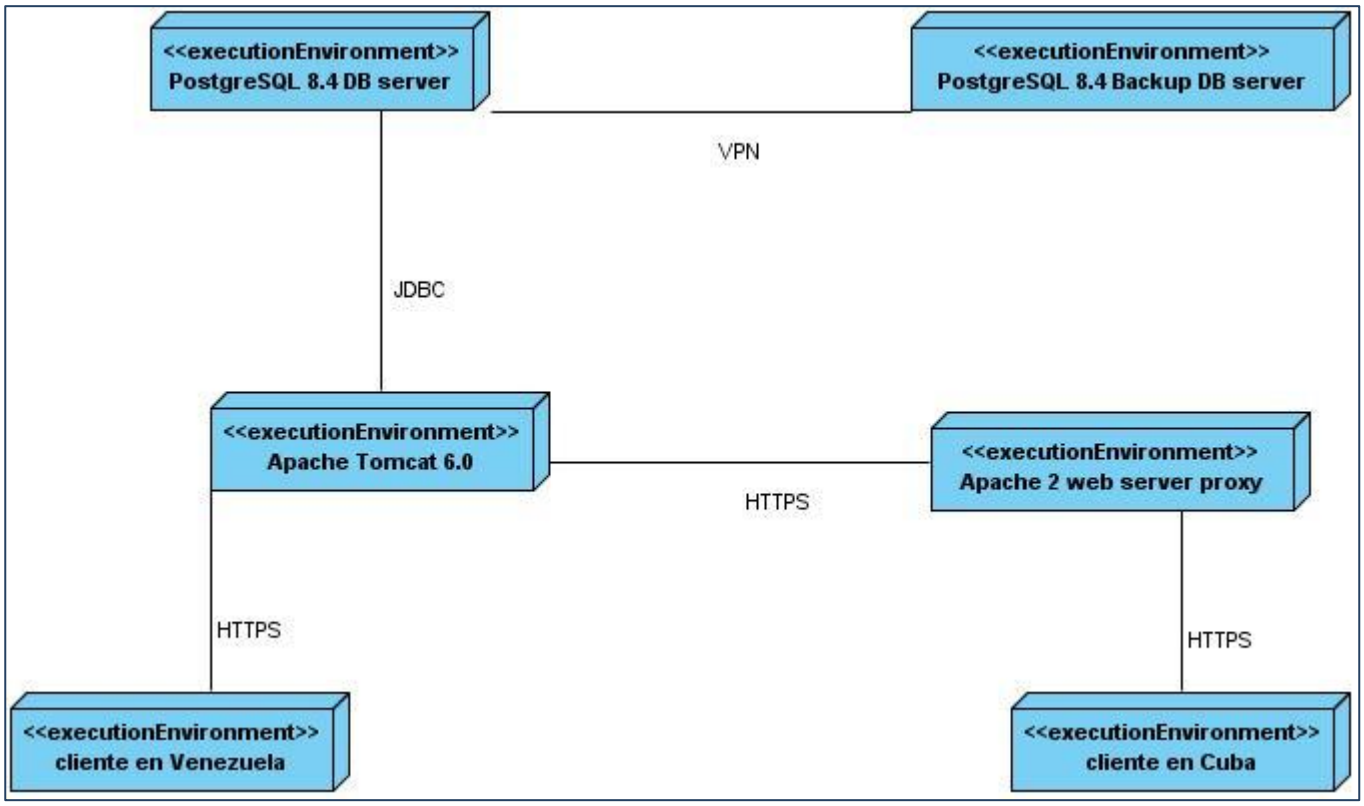


Fig. 16 Diagrama de despliegue de la aplicación.

2.7.2.1 Descripción de los nodos.

Apache Web Server Proxy.

Servidor Web Apache 2 con función de proxy.

Requerimientos de Hardware

- ✓ Procesador Inter Xeon 3.0 Ghz.
- ✓ 4 Gb memoria RAM DDR3.
- ✓ 40 Gb Disco Duro 10000 RPM.

Apache Tomcat 6.0.

Contenedor de Servlets Apache Tomcat 6.0.

Requerimientos de Hardware

- ✓ Procesador Inter Xeon 3.0 Ghz.
- ✓ 8 Gb memoria RAM DDR3.
- ✓ 40 Gb Disco Duro 10200 RPM.

Subsistemas de Implementación.

- ✓ Liferay
- ✓ OpenxavaPortlets

PostgreSQL 8.4 DB Server.

Servidor de Bases de Datos PostgreSQL.

Requerimientos de Hardware

- ✓ Procesador Inter Xeon 3.0 Ghz.
- ✓ 8 Gb memoria RAM DDR3.
- ✓ 40 Gb Disco Duro 10200 RPM.

PostgreSQL 8.4 BackupDb Server.

Servidor de Bases de Datos PostgreSQL para respaldo.

Requerimientos de Hardware

- ✓ Procesador Inter Xeon 3.0 Ghz.
- ✓ 8 Gb memoria RAM DDR3.
- ✓ 40 Gb Disco Duro 10200 RPM.

Cliente en Cuba.

Clientes Parte Cubana.

Requerimientos de Hardware

- ✓ Procesador Intel/AMD 1.6 Ghz.
- ✓ 256 Mb memoria RAM DDR1.
- ✓ 40 Gb Disco Duro.

Clientes en Venezuela.

Clientes Parte Venezolana.

Requerimientos de Hardware

- ✓ Procesador Intel/AMD 1.6 Ghz.
- ✓ 256 Mb memoria RAM DDR1.
- ✓ 40 Gb Disco Duro.

2.8 Modelo de datos.

Es una colección de herramientas conceptuales para describir los datos, las relaciones que existen entre ellos, semántica asociada a los datos y restricciones de consistencia. Se describen las tablas que representan las distintas entidades que pertenecen al dominio del problema y serán almacenadas en la base de datos. A continuación se presenta el modelo de datos utilizado en el módulo que es generado por OpenXava a partir de anotaciones en las clases del modelo.

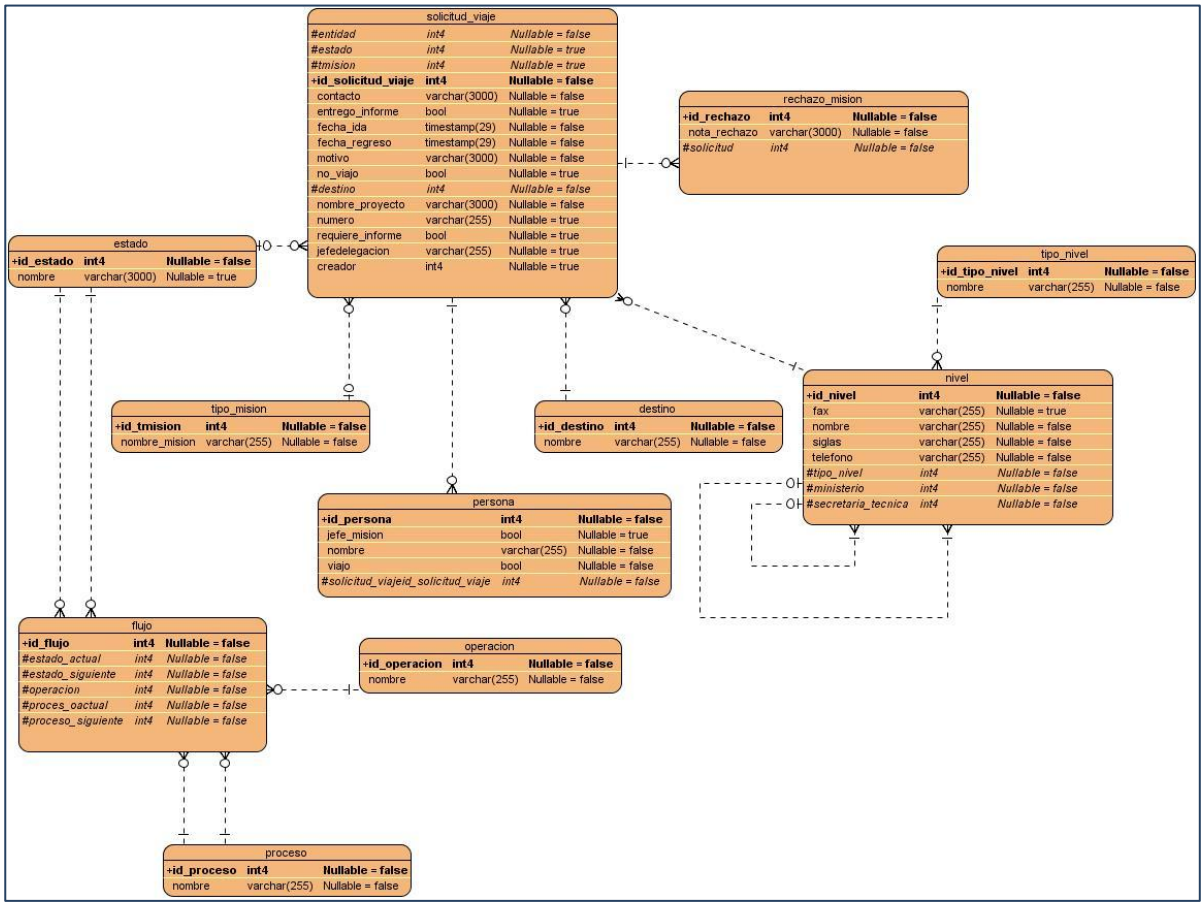


Fig. 17 Modelo de datos del módulo.

2.9 Descripción de las principales clases a utilizar.

En este epígrafe se describen las clases del modelo.

Destino

Atributos

Nombre	Tipo	Descripción
idDestino	int	Identificador del destino
nombre	String	Nombre del destino

Tabla 2 Clase Destino.

Persona

Atributos

Nombre	Tipo	Descripción
idPersona	int	Identificador de la persona
nombrePersona	String	Nombre de la persona
jefeMision	boolean	Indica si es jefe de una misión o no
viajo	boolean	Indica si la persona ya viajó
solicitudViaje	SolicitudViaje	Solicitud de viaje a la que pertenece la persona

Métodos

Nombre	Descripción
personasSinViajar(IntegeridPais)	Devuelve las personas que no han viajado hacia un país específico.
actualizarPersonasViajeDIE(List<Integer>idpersonas)	Actualiza el atributo viajo (true)

Tabla 3 Clase Persona.

RechazoSolicitud

Atributos

Nombre	Tipo	Descripción

idRechazoSolicitud	int	Identificador del rechazo
notaRechazo	String	Texto de la nota de rechazo
solicitud	SolicitudViaje	Solicitud de viaje a la que se le hace el rechazo

Métodos

Nombre	Descripción
RechazoSolicitudrechazoDadoIdEntidad(int identidad)	Devuelve el último rechazo hecho a una solicitud

Tabla 4 Clase RechazoSolicitud.

SolicitudViaje

Atributos

Nombre	Tipo	Descripción
idsolicitudViaje	int	Identificador de la solicitud
nombreProyecto	String	Nombre del proyecto al que pertenecen las personas que hacen la solicitud
fechaIda	Date	Fecha del viaje de ida
fechaRegreso	Date	Fecha del viaje de regreso
motivo	String	Motivo del viaje
contacto	String	Instituciones o personalidades a contactar
requiereInforme	Boolean	Indica si requiere informe de viaje
numeroReferencia	String	Número de referencia de la carta de solicitud de aprobación.
entregoInforme	Boolean	Indica si se entregó el informe de viaje
personas	Collection<Persona>	Personas que solicitan viajar
destino	Destino	Destino del viaje
entidad	Nivel	Ente o ministerio que hace la solicitud
jefeDelegacion	String	Nombre del jefe de la delegación
tmision	TipoMision	Tipo de misión
estado	Estado	Estado de la solicitud

ficheroDIE	Byte[]	Fichero con el listado de todas las personas que realizaron los trámites en el Departamento de Inmigración y Extranjería para salir del país.

Métodos	
Nombre	**Descripción**
eliminarNotasRechazo()	Elimina las notas de rechazo de una solicitud

Tabla 5 Clase SolicitudViaje.

SolicitudViajeActualizarRegistro

Atributos		
Nombre	**Tipo**	**Descripción**
idsolicitudViaje	int	Identificador de la solicitud
numeroReferencia	String	Número de referencia de la carta de solicitud de aprobación.
estado	Estado	Estado de la solicitud

Tabla 6 Clase SolicitudViajeActualizarRegistro.

2.10 Conclusiones parciales.

En este capítulo se obtuvieron los diagramas de componentes, de despliegue y el modelo de datos que permitieron entender cómo se organizarían los componentes del software, sus relaciones de dependencia y la estructura de datos necesaria para la implementación. Al aplicar los estándares de codificación descritos se hizo más legible el código fuente, haciendo más fácil el mantenimiento del módulo. Los patrones de diseño aplicados le aportaron robustez y reusabilidad a la solución. Para asegurar el cumplimiento del objetivo general se describieron las medidas tomadas para lograr que la solución tenga un rendimiento superior a la versión anterior.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción.

En este capítulo se describen las pruebas hechas al software como las pruebas de caja blanca, de caja negra con el propósito de comprobar el correcto funcionamiento del módulo. También se evalúa mediante métricas el diseño propuesto por los analistas y se muestran los resultados de las pruebas de carga y estrés.

3.2 Pruebas de software.

Las pruebas de software son un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación.

La creciente percepción del software como un elemento del sistema y la importancia de los costes asociados a un fallo del propio sistema, están motivando la creación de pruebas minuciosas y bien planificadas. No es raro que una organización de desarrollo de software emplee entre el 30 y el 40 por ciento del esfuerzo total de un proyecto en las pruebas. En casos extremos, las pruebas de software para actividades críticas (por ejemplo, control de tráfico aéreo, control de reactores nucleares) pueden costar ¡de tres a cinco veces más que el resto de los pasos de la ingeniería de software juntos! (17)

Los objetivos de las pruebas son:

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

3.3 Aplicación de pruebas de caja blanca.

A continuación se mostrará un ejemplo de las pruebas de caja blanca realizadas a las principales funcionalidades del módulo. Estas pruebas se hicieron con juegos de datos reales comprobando todos los nodos del árbol.

```
public void execute() throws Exception
{
    List<Integer> tiposNivel = InfoUsuario.usuarioTiposNivel("SolicitudViaje");
    String condicion = "false";
    String baseCondition="";
    if (tiposNivel.contains(tipo_nivel_st))
    {
        ccndicion_base_solicitud = getTab().getBaseCondition();
        return;
    }

    if (tiposNivel.contains(tipo_nivel_min))
    {
        List<Map<String, Object>> niveles = InfoUsuario.nivelesPorUsuario_TipoNivel("SolicitudViaje", tipo_nivel_min);
        String ministerio = InfoUsuario.getMinisterio("SolicitudViaje", false);
        ccndicion = "misiones.solicitud_viaje.ministerio = ('".concat(ministerio).concat("')");

    }
    else if(tiposNivel.contains(tipo_nivel_ee))
    {

        List<Map<String, Object>> niveles = InfoUsuario.nivelesPorUsuario_TipoNivel("SolicitudViaje", tipo_nivel_ee);
        String idNiveles = "";
        for (int i = 0; i < niveles.size(); i++)
        {
            idNiveles = idNiveles.concat(Integer.toString((Integer)niveles.get(i).get("idnivel")));
            if (i != niveles.size() - 1)
                idNiveles.concat(",");
        }

        condicion = "entidad in (".concat(idNiveles).concat(")");

    }

    baseCondition = (getTab().getBaseCondition() != null && !getTab().getBaseCondition().equals(""))?
            getTab().getBaseCondition().concat(" and ").concat(condicion) : condicion;
    getTab().setBaseCondition(baseCondition);
    condicion_base_solicitud = baseCondition;

}
```

Fig. 18 Método execute de la acción FiltrarSolicitudViajeNivel.

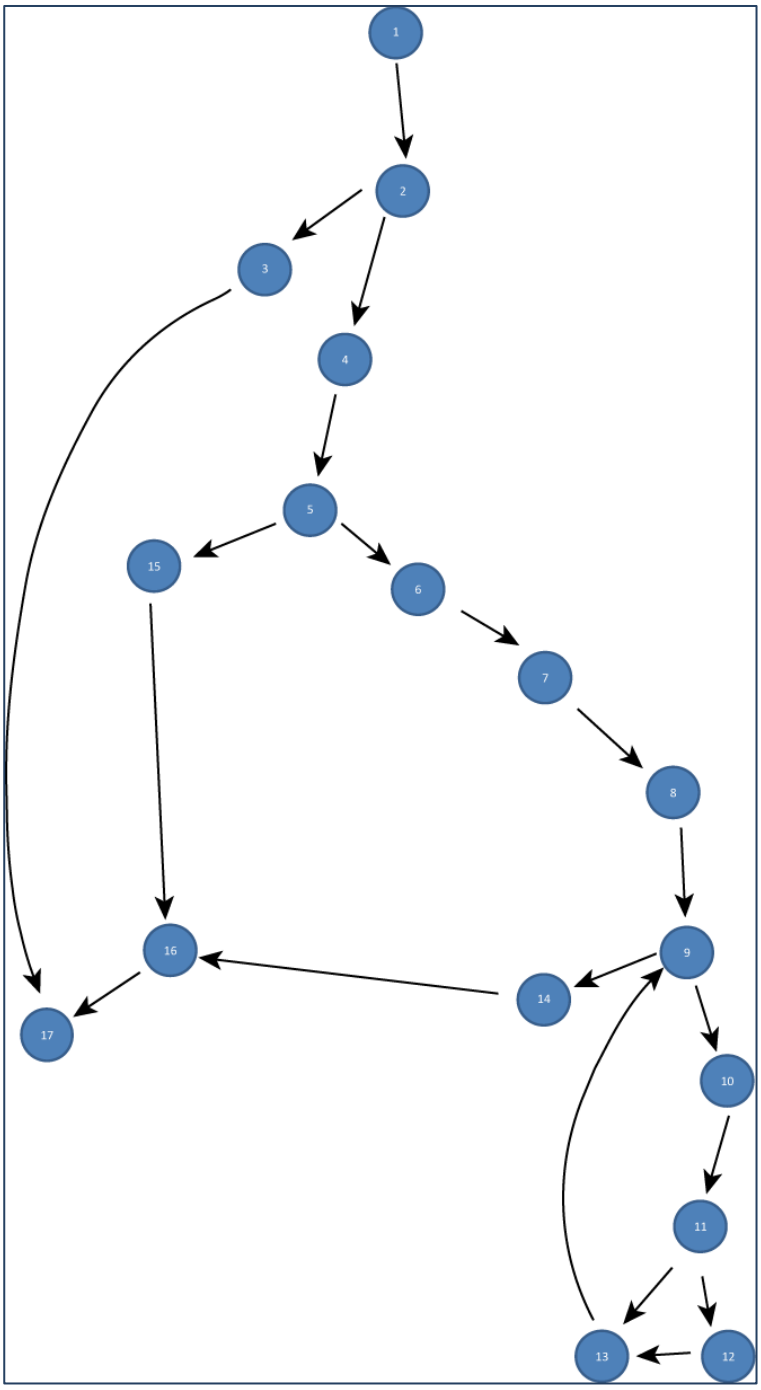


Fig. 19 Grafo asociado al método anterior

V(G) = P + 1

V(G) = 4 + 1

V(G) =5

V(G) = A - N + 2

V(G) = 20 – 17 + 2

V(G) = 5

V(G) = R

V(G) = 5

Caminos básicos	
1	**1-2-3-17**
2	**1-2-4-5-15-16-17**
3	**1-2-4-5-6-7-8-9-14-16-17**
4	**1-2-4-5-6-7-8-9-10-11-13-9-14-16-17**
5	**1-2-4-5-6-7-8-9-10-11-12-13-9-14-16-17**

Tabla 7 Caminos básicos.

A continuación se describen los casos de prueba por cada camino básico:

Caso de prueba para el camino básico 1.

Descripción:

El usuario abrirá el módulo de OpenXava “Administrar solicitud”.

Condición de ejecución:

El id del usuario será 18642.

Entrada:

idUsuario = 18642

Resultados esperados:

Se muestran todas las solicitudes de viaje del sistema.

Caso de prueba para el camino básico 2.

Descripción:

El usuario abrirá el módulo de OpenXava “Administrar solicitud”.

Condición de ejecución:

El id del usuario será 18633.

Entrada:

idUsuario = 18624

Resultados esperados:

Se muestran las solicitudes de viaje pertenecientes al ministerio que representa el usuario.

Caso de prueba para el camino básico 3.

Descripción:

El usuario abrirá el módulo de OpenXava “Administrar solicitud”.

Condición de ejecución:

El id del usuario será 18605.

Entrada:

idUsuario = 18605

Resultados esperados:

Se muestran las solicitudes de viaje pertenecientes al ente ejecutor que representa el usuario.

Caso de prueba para el camino básico 4.

Descripción:

El usuario abrirá el módulo de OpenXava “Administrar solicitud”.

Condición de ejecución:

El id del usuario será 18615.

Entrada:

idUsuario = 18615

Resultados esperados:

Se muestran las solicitudes de viaje pertenecientes al ente ejecutor que representa el usuario.

Caso de prueba para el camino básico 5.

Descripción:

El usuario abrirá el módulo de OpenXava “Administrar solicitud”.

Condición de ejecución:

El id del usuario será 10671.

Entrada:

idUsuario = 10671

Resultados esperados:

Se muestran las solicitudes de viaje pertenecientes al ente ejecutor que representa el usuario.

Para las pruebas de caja blanca se utilizó JUnit para ello se crea un clase que herede de TestCase. Luego se declaran en ella los atributos necesarios para la prueba y se sobrescribe el método setUp que se ejecuta antes de los casos de pruebas y se usa para inicializar las variables que se necesiten. Por último se programan los casos de pruebas, para ello se crea un método cuyo nombre comienza con test seguido por el nombre del caso de prueba que se va a programar en él.

Métodos de JUnit utilizados para realizar las pruebas:

Método	Descripción
assertEquals	**Se acepta el resultado si los objetos son iguales**
asserNotNull	**Se acepta el resultado si el objeto no es null**
assertNull	**Se acepta el resultado si el objeto es null**
assertFalse	**Se acepta el resultado si la condición es falsa**
assertTrue	**Se acepta el resultado si la condición es verdadera**

Tabla 8 Métodos de JUnit.

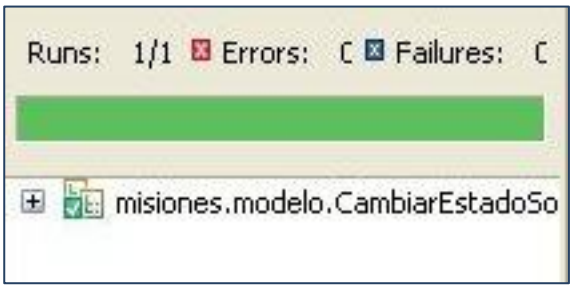


Fig. 20 Prueba satisfactoria

Fig. 21 Prueba insatisfactoria

Estas pruebas se aplicaron a los principales métodos del módulo, para escogerlos se tuvo en cuenta su complejidad y en cuantas funcionalidades eran utilizados. A continuación se presentan ejemplos de pruebas realizadas a lo métodos execute de FiltrarSolicitudViajeEstados y execute de FiltrarSolicitudViajeNivel. Estos métodos tienen complejidades $O(1)$ y $O(n)$ respectivamente, ambos son usados en casi todas las funcionalidades del módulo para filtrar las listas de solicitudes de viaje. Los resultados de pruebas otros métodos se encuentran en el anexo 2.

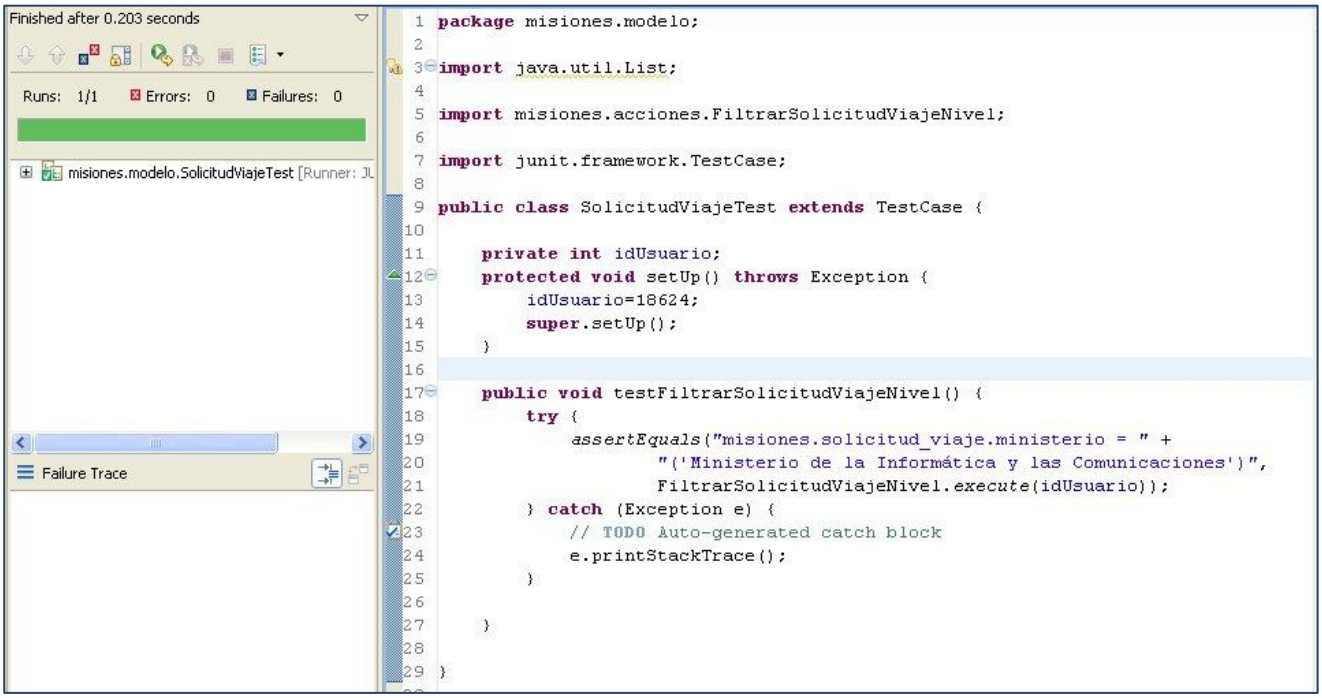


Fig. 22 Prueba realizada al método execute de FiltrarSolicitudViajeNivel

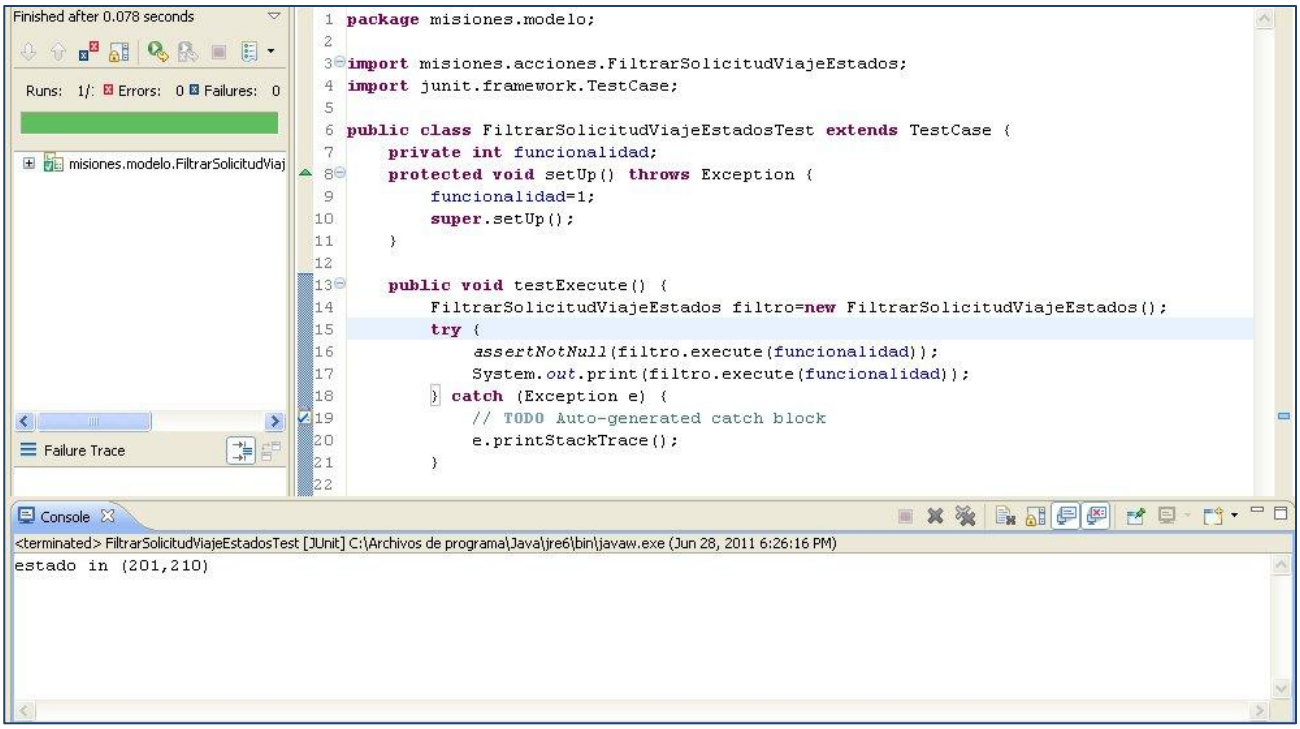


Fig. 23 Prueba realizada al método execute de FiltrarSolicitudViajeEstados

Las pruebas se realizaron en dos iteraciones, en la primera un 13% de las pruebas tuvieron resultados insatisfactorios. Para la segunda iteración se arreglaron los errores detectados en la primera por lo que todas las pruebas tuvieron resultados satisfactorios.

3.4 Aplicación de pruebas de caja negra.

Las pruebas de caja negra están centradas en los requisitos funcionales del software. Mediante un conjunto de condiciones de entrada se ejercitan los requisitos funcionales del sistema. Estas pruebas permiten encontrar:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y terminación.

Para la ejecución de los casos de pruebas se utilizaron un conjunto de datos válidos e inválidos que permitieron ejecutar el sistema en todas sus variantes. A continuación se presentan los resultados del caso de prueba Enviar solicitud al ministerio.

Descripción general.

El caso de uso inicia cuando un coordinador de Ente Ejecutor necesita enviar una solicitud de salida de misión al Ministerio.

Condiciones de ejecución.

El usuario debe estar autenticado con el permiso necesario y deben existir solicitudes en estado 'sin enviar al ministerio.

Tabla 9 Secciones a probar en el caso de uso.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Enviar solicitud	EC 1.1: Seleccionar una solicitud	Se selecciona una solicitud de salida de misión para ser enviada al ministerio.
	EC 1.2: Enviar solicitud al ministerio exitosamente.	Se envía solicitud al ministerio correctamente.
	EC 1.3: Cancelar el envío de la solicitud	Se cancela el envío de la solicitud al ministerio.

Tabla 10 Descripción de variable.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	País destino	**Lista desplegable**	**No**	Pueden ser letras o números ,palabras separadas correctamente
2	Ministerio	**Lista desplegable**	**No**	Pueden ser letras o números , palabras separadas correctamente
3	Entidad de Misión	**Lista desplegable**	**No**	Pueden ser letras o números , palabras separadas correctamente
4	Proyecto	**Lista desplegable**	**No**	Pueden ser letras o números , palabras separadas correctamente
5	Fecha ida	**Campo de texto**	**Si**	Solo números separados por un guión
6	Fecha de Regreso	**Campo de texto**	**Si**	Solo números separados por un guión
7	Nombre y Apellidos	**Campo de texto**	**Si**	Deben ser solo letras y las palabras separadas por un espacio.

Tabla 11 Matriz de Datos

Escenario	País destino	Ministerio	Entidad de Misión	Proyecto	Fecha ida	Fecha de Regreso	Nombre y Apellidos	Respuesta Esperada	Resultado de la Prueba
EC 1.1: Seleccionar una solicitud	NA	NA	NA	NA	NA	NA	NA	El sistema permite seleccionar una solicitud	Satisfactorio.
EC 1.2: Enviar solicitud al ministerio exitosamente	NA	NA	NA	NA	NA	NA	NA	El sistema permite enviar solicitud al ministerio	Satisfactorio.
EC 1.3: Cancelar el envío de la solicitud	NA	NA	NA	NA	NA	NA	NA	El sistema permite cancelar el envío de una solicitud al ministerio	Satisfactorio.

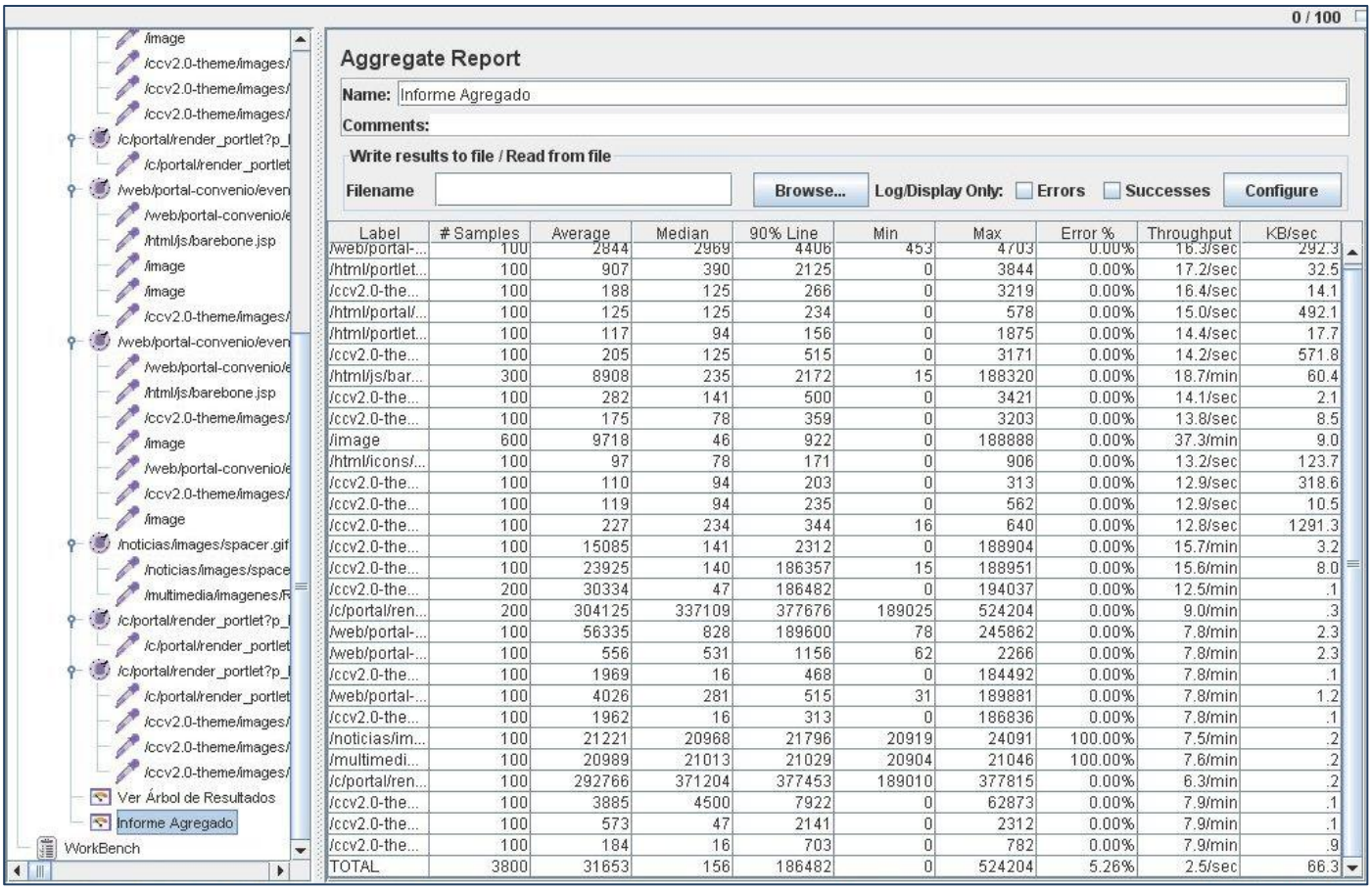
3.5 Evaluación del rendimiento de la aplicación.

Para evaluar el rendimiento de la solución y de la primera versión se hicieron pruebas de carga y estrés con la herramienta Apache JMeter. Las aplicaciones se montaron en un servidor con las siguientes características:

- ✓ Hardware: Core 2 Duo, 2 Gb RAM, SO Ubuntu 10.04, HDD 250 Gb. La aplicación estaba montada en una partición de 10Gb.
- ✓ Software: Gestor de Base de Datos: PostgreSQL 8.4. Servidor de Aplicaciones: Tomcat 6.014. Máquina Virtual: JDK 6.

Se definió un máximo de 100 hilos (cantidad de usuarios que simularon la prueba). Los resultados obtenidos al terminar las pruebas realizadas a la solución, son los siguientes:

- ✓ Muestras: 3800 (Cantidad de páginas que simulan la cantidad de usuarios que están interactuando con el sistema desde la misma URL).
- ✓ Media: 31653 (Media de páginas que se cargaron de manera satisfactoria).
- ✓ Mediana: 156 milisegundos (Tiempo promedio que han tardado en cargarse las páginas).
- ✓ Min: 0 (Tiempo mínimo que ha demorado en cargarse una página).
- ✓ Max: 524204 milisegundos (Tiempo Máximo que ha tardado en cargarse una página).
- ✓ Línea 90 %: 186482 milisegundos (tiempo en que el 90 por ciento de las páginas se cargaron de manera satisfactoria).
- ✓ %Error: 5.26 (Por ciento de error de las páginas que no se llegaron a cargar de manera satisfactoria).
- ✓ Kb/segundos: 66.3/segundos (Velocidad de carga de las páginas).
- ✓ Tiempos de Respuestas: 2.5 segundos (Total del tiempo que demoró en cargarse la cantidad de usuarios de esa prueba).



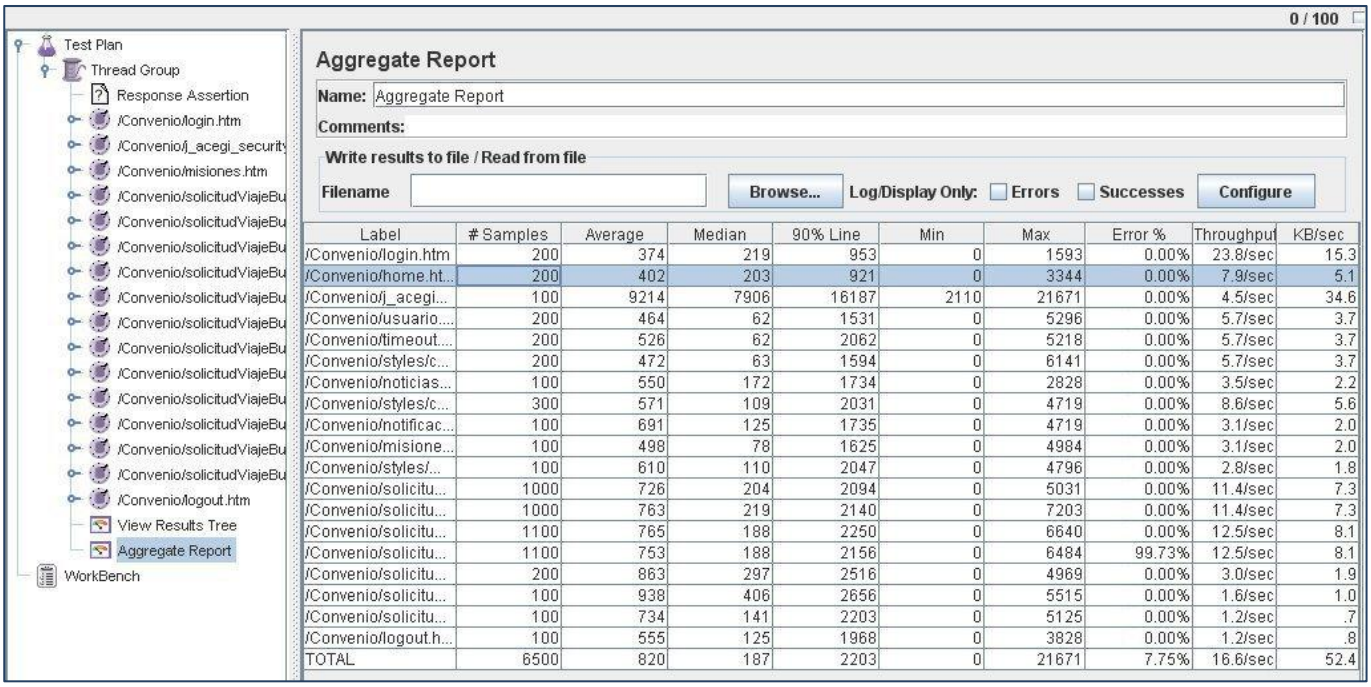
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/web/portal-...	100	2844	2969	4406	453	4703	0.00%	16.3/sec	292.3
/html/portlet...	100	907	390	2125	0	3844	0.00%	17.2/sec	32.5
/ccv2.0-the...	100	188	125	266	0	3219	0.00%	16.4/sec	14.1
/html/portal/...	100	125	125	234	0	578	0.00%	15.0/sec	492.1
/html/portlet...	100	117	94	156	0	1875	0.00%	14.4/sec	17.7
/ccv2.0-the...	100	205	125	515	0	3171	0.00%	14.2/sec	571.8
/html/js/bar...	300	8908	235	2172	15	188320	0.00%	18.7/min	60.4
/ccv2.0-the...	100	282	141	500	0	3421	0.00%	14.1/sec	2.1
/ccv2.0-the...	100	175	78	359	0	3203	0.00%	13.8/sec	8.5
/image	600	9718	46	922	0	188888	0.00%	37.3/min	9.0
/html/icons/...	100	97	78	171	0	906	0.00%	13.2/sec	123.7
/ccv2.0-the...	100	110	94	203	0	313	0.00%	12.9/sec	318.6
/ccv2.0-the...	100	119	94	235	0	562	0.00%	12.9/sec	10.5
/ccv2.0-the...	100	227	234	344	16	640	0.00%	12.8/sec	1291.3
/ccv2.0-the...	100	15085	141	2312	0	188904	0.00%	15.7/min	3.2
/ccv2.0-the...	100	23925	140	186357	15	188951	0.00%	15.6/min	8.0
/ccv2.0-the...	200	30334	47	186482	0	194037	0.00%	12.5/min	.1
/c/portal/ren...	200	304125	337109	377676	189025	524204	0.00%	9.0/min	.3
/web/portal-...	100	56335	828	189600	78	245862	0.00%	7.8/min	2.3
/web/portal-...	100	556	531	1156	62	2266	0.00%	7.8/min	2.3
/ccv2.0-the...	100	1969	16	468	0	184492	0.00%	7.8/min	.1
/web/portal-...	100	4026	281	515	31	189881	0.00%	7.8/min	1.2
/ccv2.0-the...	100	1962	16	313	0	186836	0.00%	7.8/min	.1
/noticias/im...	100	21221	20968	21796	20919	24091	100.00%	7.5/min	.2
/multimedi...	100	20989	21013	21029	20904	21046	100.00%	7.6/min	.2
/c/portal/ren...	100	292766	371204	377453	189010	377815	0.00%	6.3/min	.2
/ccv2.0-the...	100	3885	4500	7922	0	62873	0.00%	7.9/min	.1
/ccv2.0-the...	100	573	47	2141	0	2312	0.00%	7.9/min	.1
/ccv2.0-the...	100	184	16	703	0	782	0.00%	7.9/min	.9
TOTAL	3800	31653	156	186482	0	524204	5.26%	2.5/sec	66.3

Fig. 24 Reporte de JMeter sobre la solución

Estos indicadores permitieron llegar a la conclusión de que las pruebas fueron satisfactorias bajo el entorno de prueba creado. Alcanzando el sistema, a manera de resumen, un Tiempo de Respuesta de 2.5 segundos promedio para una transferencia de 66.3 Kb/segundos.

Durante las pruebas la aplicación no mostró ningún error que pudiera, debido al estrés sufrido, hacer fallar los procesos que se solicitaban.

A partir de los resultados de las pruebas realizadas a la primera versión se pudo observar una significativa reducción del tiempo de respuesta de la solución con respecto a la primera versión.



Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
/Convenio/login.htm	200	374	219	953	0	1593	0.00%	23.8/sec	15.3
/Convenio/home.ht...	200	402	203	921	0	3344	0.00%	7.9/sec	5.1
/Convenio/j_acegi...	100	9214	7906	16187	2110	21671	0.00%	4.5/sec	34.6
/Convenio/usuario....	200	464	62	1531	0	5296	0.00%	5.7/sec	3.7
/Convenio/timeout....	200	526	62	2062	0	5218	0.00%	5.7/sec	3.7
/Convenio/styles/c...	200	472	63	1594	0	6141	0.00%	5.7/sec	3.7
/Convenio/noticias...	100	550	172	1734	0	2828	0.00%	3.5/sec	2.2
/Convenio/styles/c...	300	571	109	2031	0	4719	0.00%	8.6/sec	5.6
/Convenio/notificac...	100	691	125	1735	0	4719	0.00%	3.1/sec	2.0
/Convenio/misione...	100	498	78	1625	0	4984	0.00%	3.1/sec	2.0
/Convenio/styles/...	100	610	110	2047	0	4796	0.00%	2.8/sec	1.8
/Convenio/solicitu...	1000	726	204	2094	0	5031	0.00%	11.4/sec	7.3
/Convenio/solicitu...	1000	763	219	2140	0	7203	0.00%	11.4/sec	7.3
/Convenio/solicitu...	1100	765	188	2250	0	6640	0.00%	12.5/sec	8.1
/Convenio/solicitu...	1100	753	188	2156	0	6484	99.73%	12.5/sec	8.1
/Convenio/solicitu...	200	863	297	2516	0	4969	0.00%	3.0/sec	1.9
/Convenio/solicitu...	100	938	406	2656	0	5515	0.00%	1.6/sec	1.0
/Convenio/solicitu...	100	734	141	2203	0	5125	0.00%	1.2/sec	.7
/Convenio/logout.h...	100	555	125	1968	0	3828	0.00%	1.2/sec	.8
TOTAL	6500	820	187	2203	0	21671	7.75%	16.6/sec	52.4

Fig. 25 Reporte de JMeter sobre la primera versión

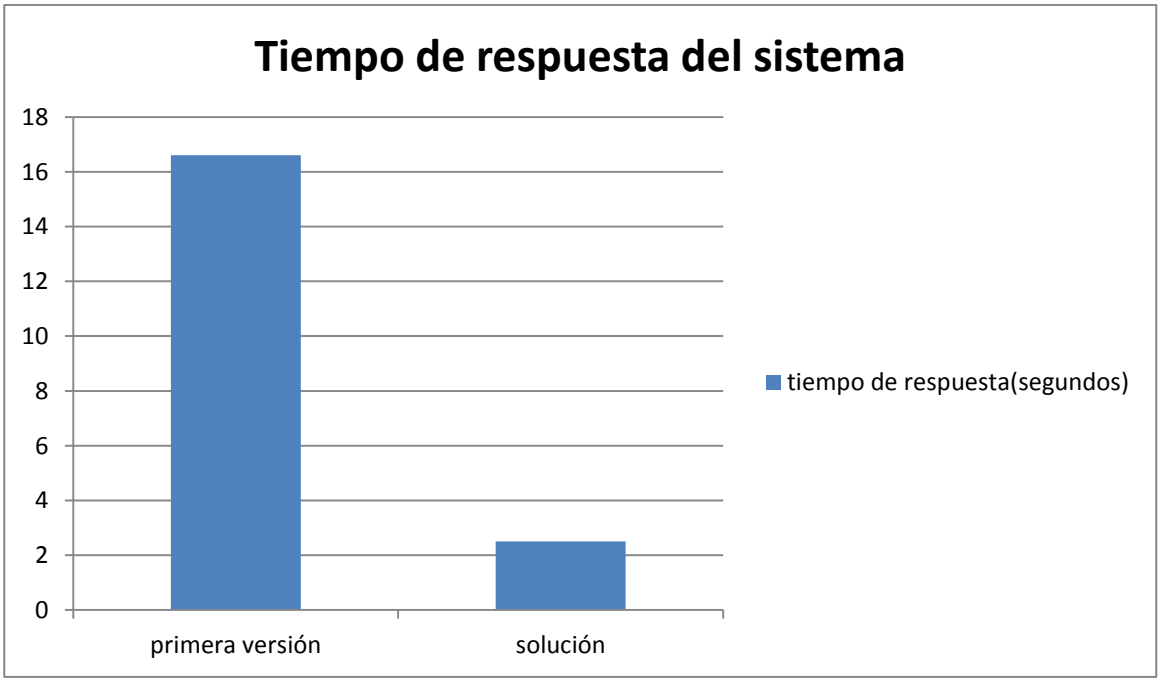


Fig. 26 Gráfica del tiempo de respuesta de la primera versión y de la solución.

3.6 Conclusiones parciales.

En este capítulo se realizó la validación de la solución propuesta. Las pruebas de caja blanca, aplicadas a los principales métodos de la solución, permitieron detectar y corregir errores en su codificación. Con las pruebas de caja negra se comprobó que el funcionamiento del módulo estaba acorde a los requisitos. Además se realizaron pruebas de carga y estrés que permitieron comparar el rendimiento de la solución y la primera versión del módulo, demostrando que la solución desarrollada tiene un rendimiento superior, con respecto a la anterior.

CONCLUSIONES

Al culminar este trabajo se logró cumplir el objetivo propuesto ya que:

- ✓ Se realizó el estudio de metodologías de desarrollo de software, lenguajes de programación y tecnologías usadas en el desarrollo de aplicaciones web. Este estudio permitió determinar la metodología, el lenguaje de programación y las tecnologías que se utilizaron en el desarrollo de la solución.
- ✓ Se describieron aspectos significativos de la solución propuesta como diagramas de componentes, de despliegue y el modelo de datos que permitieron entender cómo se organizarían los componentes del software, sus relaciones de dependencia y la estructura de datos necesaria para la implementación. Se implementó la solución usando los estándares de codificación descritos lo que produjo un código más legible. Además en la implementación se aplicaron medidas para lograr un mejor rendimiento que la primera versión.
- ✓ Se realizó la validación de la solución propuesta, mediante pruebas de caja blanca, caja negra y de carga y estrés que permitieron demostrar que la solución cumple con los requisitos y tiene un rendimiento superior a la primera versión. A partir de estos resultados se obtuvo el acta de liberación por parte de CaliSoft (ver anexo 5).

RECOMENDACIONES

Implementar una funcionalidad para firmar digitalmente los documentos generados por el módulo.

BIBLIOGRAFÍA

1. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* s.l. : Addison Wesley, 2000.

2. **Martínez, Alejandro y Martínez, Raúl.** *sitio de la Universidad de Castilla la Mancha.* [En línea] [Citado el: 12 de Enero de 2011.] http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia%20RUP.pdf.

3. **Rozanec, José.** *sitio de la Universidad Austral.* [En línea] [Citado el: 13 de Enero de 2011.] http://web.austral.edu.ar/descargas/facultad-ingenieria/newsletter/may_01_09/May_03_09/files/extreme.pdf.

4. **Enjolras, Maryvonne.** *sitio de CIEPI.* [En línea] [Citado el: 16 de Enero de 2011.] http://www.ciepi.org/fesabid98/Comunicaciones/m_enjolras.htm.

5. **Alvarez, Miguel Angel.** desarrolloweb.com. [En línea] [Citado el: 25 de Enero de 2011.] http://www.desarrolloweb.com/articulos/497.php.

6. **Achour, Mehdi, y otros, y otros.** *sitio oficial de PHP.* [En línea] [Citado el: 16 de Enero de 2011.] http://www.php.net/manual/es/intro-whatis.php.

7. **Marley, Jimi.** *Programación en castellano.* [En línea] [Citado el: 17 de Enero de 2011.] http://www.programacion.com/articulo/por_que_elegir_php_143.

8. *sitio oficial de OpenXava.* [En línea] [Citado el: 17 de Enero de 2011.] http://openxava.wikispaces.com/overview_es.

9. *Sistemas de Información Cooperativos de la Universidad de Málaga.* [En línea] [Citado el: 20 de Enero de 2011.] http://www.sicuma.uma.es/sicuma/independientes/argentina08/Badaracco/spprincar.htm.

10. **Quiñones, Ernesto .** *PostgreSQL Perú.* [En línea] [Citado el: 20 de Enero de 2011.] www.postgresql.org.pe/articles/introduccion_a_postgresql.pdf.

11. **Díaz Ezcurdia, Juan Pablo.** *Red de Colaboración en Ingenieria de Software y Bases de Datos, Universidad Nacional Autónoma de México.* [En línea] [Citado el: 23 de Enero de 2011.] http://www.redisybd.unam.mx/seminarios_pdf/SunMySql_engine.pdf.

12. *sitio de la universidad de Extremadura.* [En línea] [Citado el: 25 de Enero de 2011.] http://campusvirtual.unex.es/cala/epistemowikia/index.php?title=Eclipse_SDK.

13. *sitio oficial de Netbeans.* [En línea] [Citado el: 25 de Enero de 2011.] http://netbeans.org/index_es.html.

14. *sitio oficial de GlassFish.* [En línea] [Citado el: 27 de Enero de 2011.] http://glassfish.java.net/es/public/getstarted.html.
15. *sitio de la Universidad de Sevilla.* [En línea] [Citado el: 26 de Enero de 2011.] http://www.lsi.us.es/docencia/get.php?id=1923.
16. **Larman, Craig.** *UML y Patrones.* s.l. : Pearson, 2003.
17. **Pressman, Roger.** *Ingeniería del Software.* s.l. : Mc Graw Hill, 2002.
18. **Paniza, Javier.** *OpenXava guía de referencia versión 3.1.*
19. **Lorenz, M y Kidd, J.** *Object Oriented Metrics.* s.l. : Englewood, 1994.
20. **Gamma, Erich.** *Design Patterns. Element of Reusable Object-Oriented Software.* s.l. : Addison-Wesley, 1998.
21. **Beck, Kent.** *Extreme Programming Explained Embrace Change.* s.l. : Addison Wesley, 1999.
22. **Jeffries, R, Anderson, A y Hendrickson, C.** *Extreme Programming Installed.* s.l. : Addison Wesley, 2001.
23. **Beizer, B.** *Software Testing Techniques.* s.l. : Nostrand Reinhold, 1990.
24. **Kroll, Per y Kruchten, Philippe.** *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP.* s.l. : Addison Wesley, 2003.
25. **Hernández Sampieri, Roberto, Fernández Collado, Carlos y Baptista Lucio, Pilar.** *Metodología de la investigación.* s.l. : Mc Graw Hill, 2003.

GLOSARIO

Base de datos: Conjunto de información relacionada que se encuentra agrupada o estructurada.

Código: Cifras, clave. En informática se utiliza para referirse a un conjunto de instrucciones en un lenguaje de programación.

Clase: Declaración o abstracción de un objeto cuando se programa usando el paradigma de orientación a objetos.

DIE: Departamento de Inmigración y Extranjería

Framework: Estructura predefinida para la creación de aplicaciones. Puede estar formado por un conjunto de librerías y clases o por una arquitectura que facilita el desarrollo de software.

IDE: Ambiente integrado de desarrollo (Integrated development environment). Es un conjunto de software que permite el desarrollo de aplicaciones.

Modelo: Representación abstracta de la realidad. Diagramas que representan la estructura de un sistema dado.

Servidor Web: Es un programa que corre sobre el servidor que escucha las peticiones HTTP que le llegan y las satisface. Dependiendo del tipo de la petición, el servidor Web buscará una página Web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al cliente o navegador que realizó la petición.

Software: Término genérico que designa al conjunto de programas que posibilitan realizar una tarea específica en un ordenador.