ESCUELA SUPERIOR POLITÉCNICA DEL CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

"PROPUESTA DE UNA GUÍA METODOLÓGICA UTILIZANDO TECNOLOGÍA SPRING. COMO FRAMEWORK DE DESARROLLO PARA APLICACIONES INFORMÁTICASEN LA ESPOCH. CASO PRÁCTICO COMPROTEC. ESPOCH."

"TESIS DE GRADO PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS"

Presentado por:

SANTIAGO ISRAEL PÉREZ AUQUI.

RIOBAMBA – ECUADOR

2014

AGRADECIMIENTO

Como prioridad en mi vida agradezco a mis Padres por ser los mejores,

por haber estado conmigo apoyándome en los momentos difíciles,

por dedicar tiempo y esfuerzo para ser un hombre de bien,

y darme excelentes asesoramientos en mi caminar diario.

A la Escuela Politécnica del Chimborazo, a sus autoridades y profesores,

por abrir sus puertas y darme la confianza necesaria para triunfar en la vida y transmitir sabiduría

para mi formación profesional.

Son muchas las personas que han formado parte de mi vida profesional a las que les agradezco su

amistad, asesoramientos, apoyo,

ánimo y compañía en los momentos más difíciles de mi vida.

Algunas están aquí conmigo y otras en mis recuerdos y en mi corazón,

sin importar en donde estén quiero darles las gracias por formar parte de mí.

Santiago Israel Pérez Auqui

DEDICATORIA

Este trabajo de tesis de grado está dedicado a mis queridos PADRES

quienes con mucho cariño, amor y ejemplo han hecho de mí una persona

con valores para poder desenvolverme como PROFESIONAL.

Y no me puedo ir sin antes decirles que sin ustedes no lo hubiera logrado,

tantas desveladas sirvieron de algo y aquí está el fruto.

Les agradezco a todos ustedes con toda mi alma por ser unos buenos amigos y compartir momentos

agradables y momentos tristes,

pero esos momentos son los que hacen crecer y valorar

a las personas que nos rodean. Se les aprecia colegas. Los llevaré en el corazón.

Santiago Pérez

FIRMAS RESPONSABLE Y NOTAS

NOMBRES	**FIRMAS**	**FECHA**
ING. IVÁN MENES **DECANO DE LA FACULTAD** **DE INFORMÁTICA Y ELECTRÓNICA**	----------	----------
ING. JORGE HUILCA **DIRECTOR DE LA ESCUELA** **DE INGENIERÍA EN SISTEMAS**	----------	----------
ING. GLORIA ARCOS **DIRECTORA DE TESIS**	----------	----------
ING. ROBERTO INSUASTI **MIEMBRO DE TESIS**	----------	----------
DIRECTOR DEL CENTRO DE **DOCUMENTACIÓN**	----------	----------

Nota: ----------

"Yo, Santiago Israel Pérez Auqui, soy responsable del contenido, ideas y resultados planteados en el presente proyecto de tesis, y el patrimonio intelectual de mismo pertenecen a la Escuela Superior Politécnica de Chimborazo".

_____

Santiago Israel Pérez Auqui

ÍNDICE DE ABREVIATURAS

ADF	Application Development Framework
API	Application Programming Interface.
CGI	Common Gateway Interface
CRUD	Create Read Update and Delete
DI	Dependence Injection
ERP	Planificación de Recursos Empresariales
ESPOCH	Escuela Superior Politécnica de Chimborazo.
HQL	Hibernate Query Language
HTML	Hipertext Markup Language
HTTP	Hipertext Transfer Protocol
IBM	International Business Machines
IDE	Integrated Development Environment
IoC	Inversion of Control
ISO	Organización Internacional de Normalización
J2EE	Java Enterprise Edition
J2SE	Java Standard Edition
JDBC	Java Database Connectivity
JDK	Java Development Kit
JDO	Java Data Objects
JMS	Java Message Service
JPA	Java Persistence API
JPOX	Java Persistant Objects

JSF	Java Server Faces
JSP	Java Server Pages
LDAP	Lightweight Directory Access Protocol
MDB	Message Driven Bean
MVC	Modelo Vista Controlador
ODBC	Open Database Connectivity
OJB	Object Relational Bridge
ORM	Object-Relational Mapping
POA	Programación Orientada a Aspectos
POJO	Plain Object Java Old
RMI	Remote Method Invocation
SQL	Estructured Query Language
UI	Interfaz de Usuario
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language
XUL	**XUL** XML-based User-Interface Language

ÍNDICE GENERAL

CAPÍTULO I MARCO REFERENCIAL

CAPÍTULO II MARCO TEÓRICO

ÍNDICE FIGURAS

ÍNDICE TABLAS

INTRODUCCIÓN

La presente investigación se encamina a estudiar y determinar cuál es la herramienta de mejor productividad para realizar aplicaciones web con Spring MVC e Hibernate, estableciendo parámetros de comparación los mismos que serán analizados y probados con el prototipo de JSP y JDBC, en diferentes prototipos lo que permitirá el cumplimiento de los objetivos planteados

Una prueba de rendimiento es una prueba que comprueba la estabilidad de la aplicación web, de un proceso incluido en la aplicación, tanto en las condiciones favorables como en las no favorables. (Las pruebas de rendimiento permiten mejorar el desarrollo de aplicaciones web, ya que se reducen los tiempos corrección de errores de carga, stress.

En el Capítulo I Marco referencial, se detalla los antecedentes, la justificación de la investigación los objetivos a cumplirse y la hipótesis planteada, la misma que será comprobada al final de la investigación.

El Capítulo II Marco Teórico, comprende el estudio y definiciones, así como también las ventajas y desventajas de las herramientas del Framework Spring MVC, que están destinados para el desarrollo de esta investigación.

En el Capítulo III Metodología Propuesta DAWE está desarrollada específicamente para aplicaciones web empresariales complejas, ya que se aplica la utilización de tecnologías JAVA como el que ayudará a realizar las aplicaciones de forma rápida framework Spring MVC, para garantizar la calidad de las mismas y manteniendo una organización.

En el Capítulo IV se detalla el desarrollo del proceso de Gestión de proyectos de la COMPROTEC en la ESPOCH el rendimiento cual se aplicara pruebas de rendimiento, permitirá automatizar la información que se maneja en la misma, también a través de este proporcionar una herramienta de vital importancia en la toma de decisiones de este departamento.

En el Capítulo V Comparación de prototipos, tecnología JSP / JDBC Y tecnología SPRINGMVC / HIBERNATE, presenta la información acerca de la determinación de las herramientas a comparar descripción de las mismas, así también los indicadores de rendimiento, la creación de prototipos, la realización de pruebas de rendimiento, valoración de las herramientas y la comprobación de hipótesis.

CAPÍTULO I MARCO REFERENCIAL

1.1 ANTECEDENTES

En la actualidad existen varias herramientas y lenguajes de programación para poder desarrollar aplicaciones de calidad. El problema es que muchos desarrolladores no tienen el hábito de utilizar patrones de diseñó o estándares de desarrollo, debido a esto existen los problemas de complicación de proceso. Es por eso que se ha adoptado la utilización de frameworks, que es la mejor forma de realizar un análisis, diseñó, codificación de la aplicación a desarrollar.

Para este proyecto de tesis se ha seleccionado un framework de aplicaciones open source llamado "Spring" MVC.

En este caso se ha buscado un problema con su cierto nivel de complejidad, para desarrollar un sistema que cumpla con las necesidades y requerimientos planteados en dicho problema, y así poder reflejar las excelencias funcionales que ofrece el framework Spring.

Para esto se ha encontrado que en la actualidad en COMISIÓN DE PROYECTOS Y TRANSFERENCIA TECNOLÓGICA (COMPROTEC), no se realiza un control a través de un sistema de los proyectos que se desarrollan dentro de la institución, como son proyectos deinvestigación, emprendimiento, desarrollo, inversión, eventos de difusión científica, publicaciones de artículos científicos, patentes y marcas, y tesis de grados.

Para el desarrollo de este sistema se pretende utilizar el framework Spring MVC ya que provee una forma más sencilla y mejor de desarrollar aplicaciones.

El framework Spring ayudará a que todas las características mencionadas anteriormente puedan cumplirse y crear así un sistema flexible, portable que a su vez sea amigable y trasparente. Logrando así una completa interacción en todos los aspectos con el usuario, creando un producto final que será de gran valor y podrá ser utilizado a futuro. Además de tener los conocimientos suficientes y la documentación adecuada para poder construir una aplicación utilizando Spring MVC.

Lo que se pretende realizar es un estudio del nuevo Framework Spring MVC para la plataforma java para la creación de aplicaciones web y así facilitar el desarrollo de aplicaciones J2EE, promoviendo buenas prácticas de diseño y programación, en concreto se trata de manejar patrones de diseño como Factory, Abstract Factory, Builder, Decorator, Service Locator que son ampliamente reconocidos dentro de la industria del desarrollo de software.

Para llevar de una mejor manera la gestión de servicios del departamento de la COMPROTEC se creará Frameworks de trabajo a los cuales se podrá integrar para mejorar la manera en como maneja estos procesos la Escuela Superior Politécnica del Chimborazo.

Como primer punto haremos un estudio a cerca de conceptos preliminares y sus características que permitan conocer más a fondo a cerca de las tecnologías.

Es necesario la aplicación de una guía integrada del Frameworks de trabajo realizando primeramente un análisis de la ventajas del Framework Spring MVC , lo cual permitirá obtener una idea estable para realizar una guía metodológica que contendrá los respectivos procesos para posteriormente implementarla en la Escuela Superior Politécnica del Chimborazo, permitiendo optimizar la manera en cómo se lleva la gestión y control de trabajo y que a su vez minimizar la utilización de papeleo ya que la automatización parte desde cero.

Se implantará un sistema que gestionará las actividades de una manera estable que se llevan a cabo COMPROTEC para gestionar el inventario de proyectos de investigación, proyectos de inversión y/o planes de negocios, proyectos de desarrollo, así como también para la publicación en el sitio web de la ESPOCH para que la información de las investigaciones se accesible por otras instituciones mediante el sitio web www.espoch.edu.ec.

1.2 JUSTIFICACIÓN DEL PROYECTO DE TESIS

Justificación Teórica

La mayoría de instituciones a veces no cuentan con normas, patrones o estándares de desarrollo de software, las entidades que las poseen no saben cómo utilizarlas de forma óptima. La mejor forma es a través de la utilización de frameworks, la mayoría de empresas e instituciones probablemente deban manejar más de un marco de trabajo, es por eso que se pretende utilizar Spring framework MVC ya que esta es de mucha utilidad a la hora de trabajar con otros frameworks de desarrollo.

La utilización de Spring framework MVC permite resolver el problema de la no reutilización de código, ya que este es la principal ventaja de un framework, a través de Spring MVC, se permitirá desarrollar una aplicación eficaz.

El principal objetivo de Spring Framework MVC es el constituirse en una alternativa sencilla y fácil. La simplificación del desarrollo de aplicaciones y de sus respectivas pruebas es una de las

claves del éxito de Spring MVC. Este Framework se sustenta en dos características básicas en su núcleo: Inversión de Control y la Programación Orientada a Aspectos.

Justificación Metodológica

Para dar soporte a la realización de este proyecto, se pretende realizar una guía metodológica con la cual nosotros se puede dar garantía para la generación de futuras aplicaciones, ya que a través de esta guía metodológica se puede especificar un modelo a seguir en la futuro desarrollo de aplicaciones dentro de la ESPOCH, la guía contribuirá al mejoramiento de experiencias en marcha sobre el desarrollo de aplicaciones y para facilitar la misma.

La guía a desarrollar estará en capacidad de sistematizar conceptual y teóricamente la experiencia práctica del objeto de estudio, es una forma de elaboración intelectual, procurando hacer participantes de los hallazgos a los involucrados dentro de la ejecución de este proyecto.

Justificación Práctica

Es de vital importancia realizar una aplicación informática en la (COMPROTEC) ya que se permitirá automatizar la información que se maneja en la misma, también a través de este proporcionar una herramienta de vital importancia en la toma de decisiones de este departamento, y al mismo tiempo permitirá comprobar las fortalezas del framework Spring MVC, y de esta forma adoptar un framework para el desarrollo de las futuras aplicaciones dentro de la ESPOCH.

Se realizaran prototipos de prueba para el análisis de la mejoría que se dará en la gestión laboral de la COMPROTEC, Utilizando la nueva tecnología para la automatización del sistema de gestión.

El presente proyecto de tesis está enfocado a plantear un estándar para el desarrollo de aplicaciones informáticas dentro de la ESPOCH a través del framework open source Spring MVC, ya que por medio de este se establecerá una forma más eficiente de desarrollar aplicaciones, ya que esta herramienta proporciona integración con otros frameworks de desarrollo y así sacar provecho de las

fortalezas de estos, esto se lo puede realizar gracias a que Spring Framework MVC tiene un controlador que proporciona al desarrollador una tarea más centrada.

La aplicación que se desarrollará permitirá compartir información, ya que es función de la COMPROTEC presentar informes a esta organización, también permitirá realizar la gestión de proyectos, publicaciones de artículos científicos y patentes de los proyectos de investigación, que es de vital importancia para la organización.

1.2.1 OBJETIVOS

1.2.1.1 OBJETIVO GENERAL

Realizar una guía metodológica utilizando tecnología Spring MVC, como framework de desarrollo para aplicaciones informáticas en la ESPOCH.

1.2.1.2 OBJETIVOS ESPECÍFICOS

- Analizar la tecnología Spring MVC Framework para determinar las ventajas de su utilización.
- Desarrollar prototipos de prueba para evaluar el rendimiento de Spring MVC Framework.
- Generar una aplicación web para la gestión de servicios, mediante la utilización de herramientas software JAVA Y POSGRESQL, en base a los lineamientos que siga la COMPROTEC.
- Evaluar la aplicación para determinar la mejora en la gestión proyectos de la COMPROTEC.

1.2.2 HIPÓTESIS

La utilización del Framework Spring MVC que permitirá obtener mayor productividad en el desarrollo de aplicaciones web.

CAPÍTULO II MARCO TEÓRICO

2 ANÁLISIS DE LA TECNOLOGÍA SPRING

2.1 Visión general

Spring Framework es una solución ligera y un potencial de ventanilla única para la construcción de sus preparadas para la empresa de aplicaciones. Sin embargo, la Spring es modular, lo que permite utilizar sólo aquellas partes que necesite, sin tener que llevar el resto. Usted puede utilizar el contenedor IoC, con Struts en la parte superior, pero se puede También utilizamos sólo el código de integración de hibernación o la capa de abstracción JDBC.http://static.springsource.org/spring/docs/3.0.0.M3/reference/html/

El Spring Framework apoya la gestión de transacciones declarativa, el acceso remoto a su lógica a través de RMI o servicios web, y diversas opciones para la persistencia de datos.

Ofrece un marco de todas las funciones de MVC, y le permite AOP integrar de forma transparente en su software.[4]

La Spring está diseñada para ser no invasivo, lo que significa que el código de la lógica de dominio general, no tiene dependencias en el propio marco. En su capa de integración (tales como la capa de acceso de datos), algunos dependencias de la tecnología de acceso de datos y las bibliotecas de Spring existirá. Sin embargo, debería ser fácil para aislar las dependencias del resto de su código base.[4]

2.1.1 Introducción

Spring es un framework[1]de aplicación desarrollado para aplicaciones escritas en el lenguaje de programación Java. Fue creado gracias a la colaboración de grandes programadores, entre ellos se encuentran como principales partícipes y líderes de este proyecto Rod Johnson y Jürgen Höller. Estos dos desarrolladores, además de otros colaboradores que juntando toda su experiencia en el desarrollo de aplicaciones J2EE (Java 2 Enterprise Editions), incluyendo EJB (EnterpriseJavaBeans), Servlets[2] y JSP (Java Server Pages), lograron combinar dichas herramientas y otras más en un sólo paquete, para brindar una estructura más sólida y un mejor soporte para este tipo de aplicaciones.[4]

Además se considera a Spring un framework lightweight, es decir liviano o ligero, ya que no es una aplicación que requiera de muchos recursos para su ejecución, además el framework completo

[1]Es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

[2]utilizados para ampliar las capacidades de los servidores que alojan aplicaciones acceder a través de una petición-respuesta modelo de programación.

puede ser distribuido en un archivo .jar de alrededor de 1 MB, lo cual representa muy poco espacio, y para la cantidad de servicios que ofrece es relativamente insignificante su tamaño.[4]

Este framework se encuentra actualmente en su versión 3.0, está adquiriendo gran auge y una gran popularidad. Una de las características que ayuda a este éxito, es que es una aplicación open source, lo cual implica que no tiene ningún costo, ni se necesita una licencia para utilizarlo, por lo tanto da la libertad a muchas empresas y desarrolladores a incursionar en la utilización de esta aplicación. Además de que está disponible todo el código fuente de este framework en el paquete de instalación.[4]

2.1.2 Historia

Los primeros componentes de lo que se ha convertido en Spring Framework fueron escritos por Rod Johnson en el año 2000, mientras trabajaba como consultor independiente para sus clientes en la industria financiera en Londres. Mientras escribía el libro Expert One-on-one J2EE Design And Development (Programmer to programmer), Rod amplió su código para sintetizar su visión acerca de cómo las aplicaciones que trabajan con varias partes de la plataforma J2EE podían llegar a ser más simples y más consistentes que aquellas que los desarrolladores y compañías estaban usando por aquel entonces. En el año 2001 los modelos dominantes de programación para aplicaciones basadas en web eran ofrecidas por el API Java Servlet y los Enterprise JavaBeans, ambas especificaciones creadas por Sun Microsystems en colaboración con otros distribuidores y partes interesadas que disfrutaban de gran popularidad en la comunidad Java. Las aplicaciones que no eran basadas en web, como las aplicaciones basadas en cliente o aplicaciones en batch, podían ser escritas con base en herramientas y proyectos de código abierto o comercial que proveen las características requeridas para aquellos desarrollos. [4]

Se formó un pequeño equipo de desarrolladores que esperaba trabajar en extender el framework y un proyecto fue creado en Sourceforge en febrero de 2003. Después de trabajar en su desarrollo

durante más de un año lanzaron una primera versión (1.0) en marzo de 2004. Después de este lanzamiento Spring ganó mucha popularidad en la comunidad Java, debido en parte al uso de Javadoc y de una documentación de referencia por encima del promedio de un proyecto de código abierto. Sin embargo, Spring Framework también fue duramente criticado en 2004 y sigue siendo el tema de acalorados debates. Al tiempo en que se daba su primer gran lanzamiento muchos desarrolladores y líderes de opinión vieron a Spring como un gran paso con respecto al modelo de programación ágil; esto era especialmente cierto con respecto a Enterprise JavaBeans. Una de las metas de diseño de Spring Framework es su facilidad de integración con los estándares J2EE y herramientas comerciales existentes. Esto quita en parte la necesidad de definir sus características en un documento de especificación elaborado por un comité oficial y que podría ser criticado. Spring Framework hizo que aquellas técnicas que resultaban desconocidas para la mayoría de programadores se volvieran populares en un periodo muy corto de tiempo. El ejemplo más notable es la inversión de control. En el año 2004, Spring disfrutó de unas altísimas tasas de adopción y al ofrecer su propio framework de programación orientada a aspectos (aspect-oriented programming, AOP) consiguió hacer más popular su paradigma de programación en la comunidad Java [cita requerida]. En 2005 Spring superó las tasas de adopción del año anterior como resultado de nuevos lanzamientos y más características fueron añadidas. El foro de la comunidad formada alrededor de Spring Framework (The Spring Forum) que arrancó a finales de 2004 también ayudó a incrementar la popularidad del framework y desde entonces ha crecido hasta llegar a ser la más importante fuente de información y ayuda para sus usuarios.[4]

2.1.3 ¿Qué es Frameworks de Trabajo?

El término framework, o marco de trabajo, se ha popularizado en los últimos años dentro del ambiente de desarrollo de software es común encontrar dicho término en diversas circunstancias: leyendo un libro sobre algún lenguaje de programación, buscando información de interés en Internet sobre una nueva tecnología Web, etc.[4]

Un framework, dentro del ambiente de desarrollo de software, es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.

Típicamente, un framework puede incluir soporte de programas, librerías y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Un framework se considera como una arquitectura de software que modela las relaciones generales de los componentes del proyecto que lo implementa; provee una estructura y manera de trabajo la cual utilizan las aplicaciones del proyecto. La finalidad de los frameworks es facilitar el desarrollo de software, permitiéndoles a diseñadores y programadores concentrarse en los requerimientos del proyecto, reduciendo los posibles problemas con las tecnologías utilizadas, así como facilitando ciertas funcionalidades básicas y comunes.[4]

Otro término también bastante popular hoy en día es el de patrones de diseño, que muchas veces se tiende a emplear en el mismo sentido que framework, pensando que hacen referencia a un mismo concepto; sin embargo, debe quedar claro que tienen significados diferentes, ya que un framework representa código implementado, mientras que un patrón de diseño representa conocimiento y experiencia.[4]

2.1.4 Razones para usar Frameworks de Trabajo.

Uso de frameworks a medio/largo plazo tiene ventajas en el mantenimiento de la aplicación/código, ampliaciones, mejoras, etcétera, a diferencia de no usar ninguno.[4]

Uno de los motivos que más me convence de los frameworks es que suelen basarse en el patrón de diseño MVC (Model-View-Controller). Este patrón de diseño dice que nuestra aplicación, debería tener, al menos, estas tres capas: modelo, vista y controlador, este patrón de diseño lo puede ser implementado sin usar ningún framework. Pues no le falta razón, es cierto, se podría implementar sin usar ningún framework. Pero la diferencia radica en que el framework te guía/obliga a implementar este patrón, lo cual hará que tu aplicación sea más robusta. Por otro lado, si decides implementar el patrón sin usar framework corres el riesgo de introducir código en la capa que no

debes, y todo pueda acabar en el archiconocido spaghetti code, lo cual hará, con el paso del tiempo, que nuestra aplicación se vaya volviendo día tras día más difícil de mantener.[4]

Los frameworks guían en el proceso de desarrollar nuestra aplicación siguiendo el patrón de diseño MVC, pero ¿qué son estas capas? ¿Para qué sirve cada una de ellas? La capa modelo es la que se encarga de trabajar con los datos, habitualmente es la que se encarga de almacenar los datos (una base de datos por ejemplo). Por su parte, la vista es la capa de presentación, cómo presentamos los datos al usuario final, al cliente de nuestra aplicación. Y, por último, el controlador es el mediador entre el modelo y la vista. Normalmente la vista requiere de datos del modelo, pero la vista no tiene comunicación directa con el modelo. Por tanto, la vista pide esos datos al controlador y éste a su vez al modelo, el modelo maneja los datos en la forma que defina nuestra lógica de la aplicación y devuelve la respuesta de nuevo al controlador, quién, por último, entrega esta respuesta a la vista para poder ser rende rizada.[4]

Otras ventajas de los frameworks:

Convention over configuration: los frameworks suelen tener una serie de reglas de convención. Entre estas reglas está por ejemplo la estructuración de directorios. Así, si conocemos estas reglas, cuando en un tiempo necesitemos tocar algo en el código, sabremos rápidamente dónde localizar este código porque tendremos montada una buena jerarquía de directorios mucho mejor de la que podríamos crearnos manualmente si no utilizáramos frameworks. No obstante, muchos frameworks permiten y dan total libertad para saltarte (si es de tu agrado) estas convenciones por defecto, y crear las tuyas propias.[4]

El código de las librerías base que aportan los frameworks está muy testeado. Podrías desarrollar tú una librería con las mismas funcionalidades pero alcanzar un nivel de testeo tan grande como lo han conseguido muchos frameworks no es fácil. Entonces mejor usar un código que está muy probado a reinventar la rueda.[4]

Casi todos los frameworks disponibles hoy en día poseen una comunidad (unas más grandes que otras) de usuarios. En esta comunidad de usuarios, algunos de ellos desarrollan módulos o extensiones para el framework y lo distribuyen gratuitamente. Imagina que requieres de una funcionalidad que de base el framework no te da. Pero has encontrado un módulo desarrollado por alguien de la comunidad de usuarios para tu framework con dicha funcionalidad. En lugar de desarrollarla tú, sería tan fácil como incorporar ese módulo a tu aplicación.[4]

Permite que el trabajo en equipo sea más sencillo. Si todo el equipo conoce el funcionamiento del framework, todos sabrán, por ejemplo, la estructura de directorios de la aplicación y sabrán localizar con facilidad el fichero de código fuente con el que requieran trabajar. Además, si en un futuro incorporas más personal a la plantilla, este personal con que conozcan el framework es suficiente. Y te puedo asegurar que encontrar un perfil con esta condición será mucho más sencillo que encontrar alguien que conozca tu sistema, totalmente sin usar frameworks.[4]

Tabla II.I. Ventajas de un Framework

Ventajas de los frameworks	
Normalización:	Una de las grandes ventajas de un marco es estandarizar el desarrollo. Debido a que hemos definido un conjunto de clases y / o funciones, están "obligados" a trabajar en la herramienta de elección.
Tasa de crecimiento:	Para los módulos genéricos hacer uso del tiempo ahorrado al no tener que "reinventar la rueda" en cada proyecto.
Calidad:	Los Frameworks de los mercados principales son versiones bien probados de alfa, beta y Release Candidate (RC) y son mantenidos por las comunidades y / o negocios experimentado.
Mantenimiento:	Siguiendo unos términos y códigos de alta calidad, se gana en facilidad de mantenimiento, ya que sabemos dónde encontrar lo que necesitamos.
Comunidad:	Excelente ambiente de aprendizaje donde se puede obtener ayuda y adquirir experiencia no sólo con la herramienta y el idioma, sino la cultura y forma de pensar.
Seguridad:	Sin duda es una de las cuestiones que preocupan a mucha gente (y con razón) y que es un muy considerado para el lanzamiento de parches y actualizar los Frameworks más populares.

2.1.5 Principales Frameworks de Trabajo.

Según Larman, un marco es una representación de un conjunto de objetos extensible que tienen funciones relacionadas. La firma de un marco de trabajo proporciona un conjunto ampliable de puntos (núcleo) que puede ser conectado a un sistema nuevo por el programador, estos puntos son diferentes estrategias para la reutilización de componentes, objetos y portlets. Nosotros como ejemplo AWT paquetes y lenguaje Java Swing. Estos Frameworks proporcionar muchas clases e interfaces, donde los desarrolladores pueden extender estos superposición de clases y métodos. [1][2]

2.1.5.1 Hibernate (Hibernate, 2009)

Hibernate es un marco que tiene un alto rendimiento para la persistencia de objetos utilizando el modelo relacional y también servicios de consulta (query). El modelo Hibernate desarrollo permite al programador para desarrollar la capa Persistencia utilizando el paradigma orientado a objetos, incluyendo la membresía, herencia, polimorfismo, composición y colecciones. Hibernate proporciona la el desarrollador de la capacidad de expresar sus propias consultas SQL en extensión(HQL), de la misma manera como comandos SQL, o con un taco los objetos.

A diferencia de otros Frameworks, la propuesta de Hibernate no se esconde el poder de SQL y en lugar de proporcionar una manera fácil de integrar sus aplicaciones orientadas a objetos Java y Net.

Mantenido en el proyecto de código abierto LGPL y está disponible con Versión 3.2.4 del 29 de enero de 2009.[4]

2.1.5.2 ¿Que proporciona Spring?

Una potente gestión de configuración basada en JavaBeans, aplicando los principios de Inversión de Control (IoC). Esto hace que la configuración de aplicaciones sea rápida y sencilla. Ya no es necesario tener singletons ni ficheros de configuración, una aproximación consistente y elegante. Estas definiciones de beans se realizan en lo que se llama el contexto de aplicación. [4][14]

Una capa genérica de abstracción para la gestión de transacciones, permitiendo gestores de transacción añadibles (pluggables), y haciendo sencilla la demarcación de transacciones sin tratarlas a bajo nivel. Se incluyen estrategias genéricas para JTA y un único JDBC DataSource. En contraste con el JTA simple o EJB CMT, el soporte de transacciones de Spring no está atado a entornos J2EE. [4]

Una capa de abstracción JDBC que ofrece una significativa jerarquía de excepciones (evitando la necesidad de obtener de SQLException los códigos que cada gestor de base de datos asigna a los errores), simplifica el manejo de errores, y reduce considerablemente la cantidad de código necesario.[4]

Integración con Hibernate, JDO e iBatis SQL Maps en términos de soporte a implementaciones DAO y estrategias con transacciones. Especial soporte a Hibernate añadiendo convenientes características de IoC, y solucionando muchos de los comunes problemas de integración de Hibernate. Todo ello cumpliendo con las transacciones genéricas de Spring y la jerarquía de excepciones DAO.[4]

Funcionalidad AOP, totalmente integrada en la gestión de configuración de Spring. Se puede aplicar AOP a cualquier objeto gestionado por Spring, añadiendo aspectos como gestión de transacciones declarativa. Con Spring se puede tener gestión de transacciones declarativa sin EJB, incluso sin JTA, si se utiliza una única base de datos en un contenedor Web sin soporte JTA.[4]

Un framework MAC (Model-View-Controller), construido sobre el núcleo de Spring. Este framework es altamente configurable vía interfaces y permite el uso de múltiples tecnologías para la capa vista como pueden ser JSP, Velocity, Tiles, iText o POI. De cualquier manera una capa

modelo realizada con Spring puede ser fácilmente utilizada con una capa web basada en cualquier otro framework MVC, como Struts, WebWork o Tapestry.[4]

Toda esta funcionalidad puede usarse en cualquier servidor J2EE, y la mayoría de ella ni siquiera requiere su uso. El objetivo central de Spring es permitir que objetos de negocio y de acceso a datos sean reutilizables, no atados a servicios J2EE específicos. Estos objetos pueden ser reutilizados tanto en entornos J2EE (Web o EJB), aplicaciones "standalone", entornos de pruebas, sin ningún problema. [4]

La arquitectura en capas de Spring ofrece mucha de flexibilidad. Toda la funcionalidad está construida sobre los niveles inferiores. Por ejemplo se puede utilizar la gestión de configuración basada en JavaBeans sin utilizar el framework MVC o el soporte AOP.[4]

2.1.6 Filosofía de Spring

Filosofía POJO, menos interfaces y excepciones chequeadas que fuerzan a complicar el código cuando integramos elementos diferentes. La prueba del software es esencial. Spring ayuda a implementar código chequeable mediante pruebas unitarias. La aplicación de Spring debe ser placentera.[4]

El código de la aplicación no debe depender de las APIs de Spring. Spring no debe competir con soluciones que ya funcionan, sino permitir su fácil integración en la aplicación (Ejemplo: Hibernate, JDO, etc.)[4]

2.1.7 Arquitectura

Spring es un framework modular que cuenta con una arquitectura divida en siete capas o módulos, como se muestra en la Figura II.2 lo cual permite tomar y ocupar únicamente las partes que interesen para el proyecto y juntarlas con gran libertad.[4]

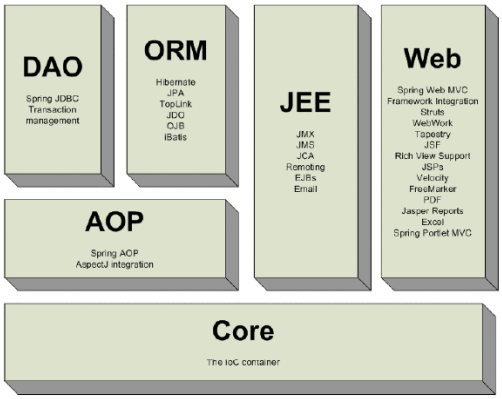


Figura II.1.Arquitectura de Spring Framework

Mediante el análisis de lo referido anteriormente mostramos una tabla con las principales características de Spring Framework.

Tabla II.II. Características de Spring.

Características principales de Spring	
Buenas Prácticas	La inicial motivación era facilitar el desarrollo de aplicaciones J2EE, promoviendo buenas prácticas de diseño y programación. En concreto se trata de manejar patrones de diseño como Factory, Abstract Factory, Builder, Decorator, Service Locator, etc; que son ampliamente reconocidos dentro de la industria del desarrollo de software.
Tipo de Código	Código abierto
Arquitectura	Enfoque en el manejo de objetos de negocio, dentro de una arquitectura en capas.
Módulos	El Core Container o Contenedor de Inversión de Control , Aspect-Oriented Programming Framework, Data Access Framework, Transaction Management Framework, Remote Access framework, Spring Web MVC, Spring Web Flow, Spring Web Services.

2.1.8 Módulos

El Spring Framework se compone de rasgos organizados en cerca de 20 módulos. Estos módulos se agrupan en el Core Container, datos de acceso / integración, Web, AOP (Programación Orientada a Aspectos), Instrumentación, y de prueba, como se muestra en el siguiente diagrama.[4]

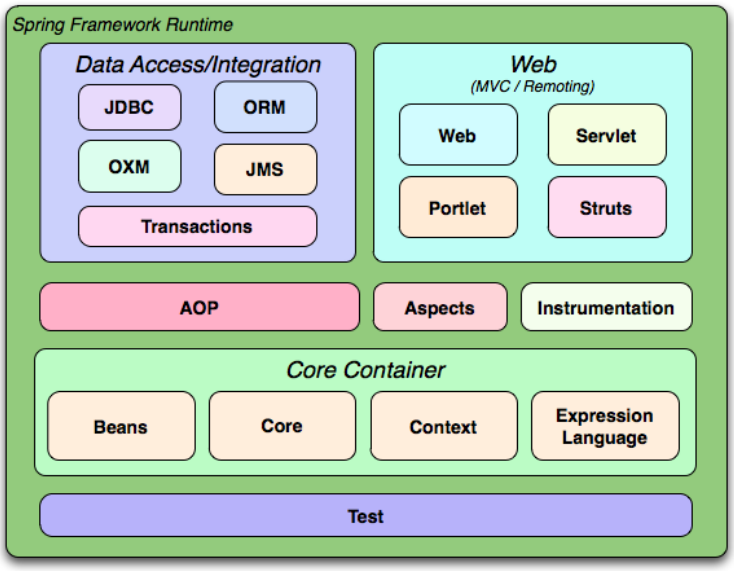


Figura II.2. Módulos Spring Framework Runtime.

2.1.9 Spring Core

Esta parte es la que provee la funcionalidad esencial del framework, está compuesta por el BeanFactory, el cual utiliza el patrón de Inversión de Control (Inversion of Control) y configura los objetos a través de Inyección de Dependencia (Dependency Injection). El núcleo de Spring es el paquete org.springframework.beans el cual está diseñado para trabajar con JavaBeans.[4]

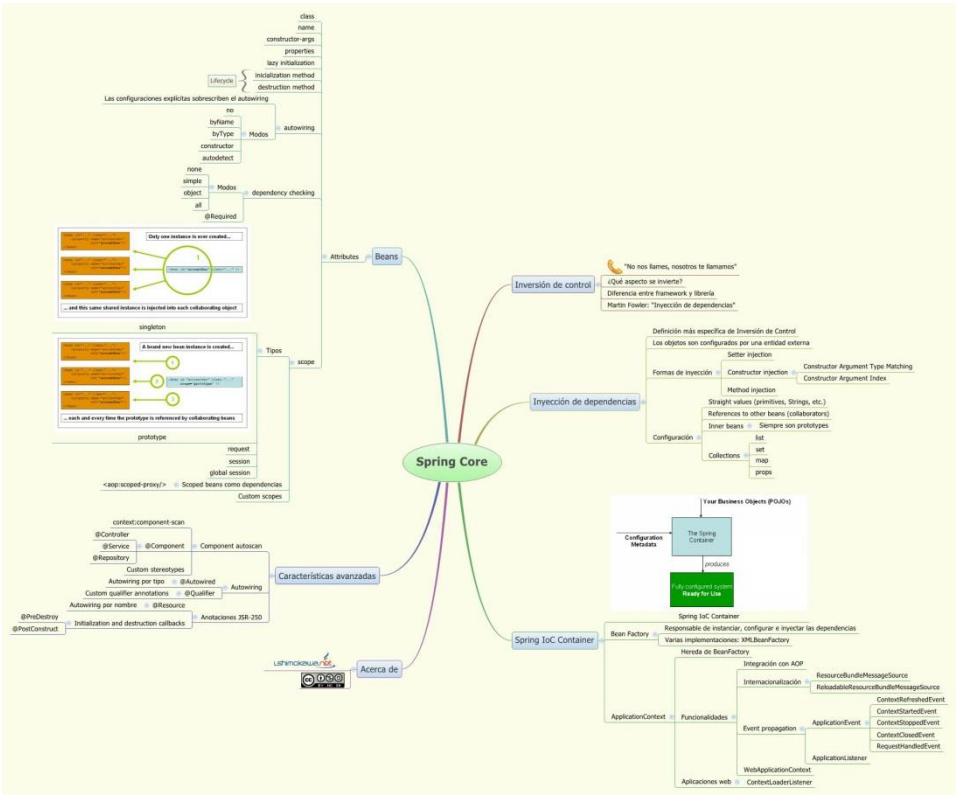


Figura II.3. Funcionalidad del Spring Core.

2.1.10 Bean Factory

Es uno de los componentes principales del núcleo de Spring es una implementación del patrón Factory, pero a diferencia de las demás implementaciones de este patrón, que muchas veces sólo producen un tipo de objeto, BeanFactory es de propósito general, ya que puede crear muchos tipos diferentes de Beans[3]. Los Beans pueden ser llamados por nombre y se encargan de manejar las relaciones entre objetos.[4]

2.1.11 Dependency Injection

Es una forma de Inversión de Control, que está basada en constructores de Java, en vez de usar interfaces específicas del framework.[4][14]

[3] Es un componente software que tiene la particularidad de ser reutilizable y así evitar la tediosa tarea de programar los distintos componentes uno a uno.

Con este principio en lugar de que el código de la aplicación utilicé el API del framework para resolver las dependencias como: parámetros de configuración y objetos colaborativos, las clases de la aplicación exponen o muestran sus dependencias a través de métodos o constructores que el framework puede llamar con el valor apropiado en tiempo de ejecución, basado en la configuración.

A continuación se presenta un gráfico de un modelo simplificado de entidades y servicios con inversión of control y dependency injection.[4]

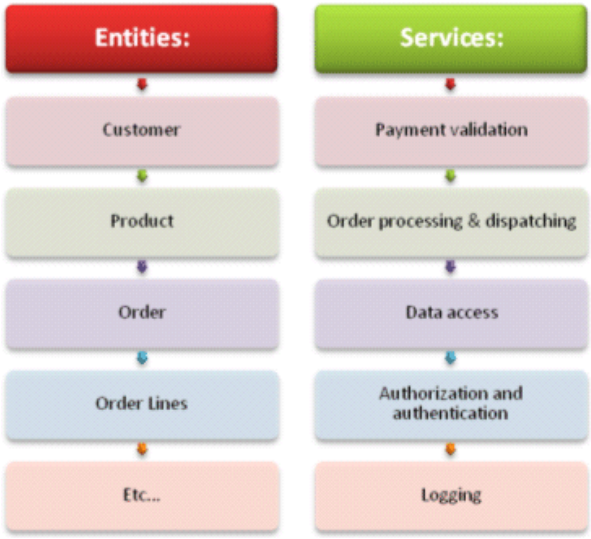


Figura II.4. Modelo de entidades y servicios

Todo esto se puede ver de una forma de push y pop, el contenedor hace un push de las dependencias para ponerlas dentro de los objetos de la aplicación, esto ocurre en tiempo de ejecución. La forma contraria es tipo pull, en donde los objetos de la aplicación jalan las dependencias del ambiente. Además los objetos de la Inyección de Dependencia nunca cargan las propiedades ni la configuración, el framework es totalmente responsable de leer la configuración.[4]

Spring soporta varios tipos de Inyección de Dependencia, pero en si estos son los dos más utilizados:

2.1.11.1Setter Injection

En este tipo la Inyección de Dependencia es aplicada por medio de métodos JavaBeans setters, que a la vez tiene un getter respectivo.[4]

2.1.11.2Constructor Injection

Esta Inyección es a través de los argumentos del constructor.

A continuación se muestra un ejemplo tomado del libro de Java Development with Spring framework del autor Rob Johnson, en el cual se muestra como un objeto es configurado a través de Inyección de Dependencia. Se tiene una interfaz Service y su implementación ServiceImpl. Supóngase que la ServiceImpl tiene dos dependencias: un int que tiene configura un timeout y un DAO (Data Access Object). Con el método SetterInjection se puede configurar ServiceImpl utilizando las propiedades de un JavaBean para satisfacer estas 2 dependencias.[4]

```
public class ServiceImpl implements Service
 {      private int timeout;
        private AccountDao accountDao;
        public void setTimeout(int timeout)
        {       this.timeout = timeout;}
        public void setAccountDao(AccountDao accountDao)
        { this.accountDao = accountDao;}
}
```

Con Constructor Injection se le da las dos propiedades al constructor:

```
public class ServiceImpl implements Service
{
        private int timeout;
        private AccountDao accountDao;
        public ServiceImpl(int timeout, AccountDao accountDao)
        {       this.timeout = timeout;
                this.accountDao = accountDao;}}
```

"La clave de la innovación de la Inyección de Dependencia es que trabaja con sintaxis pura de Java: no es necesaria la dependencia del API del contenedor".

2.1.12 Data Access Integration

El acceso a datos / Integración capa consiste en el JDBC, ORM, OXM, JMS y los módulos de transacción. El módulo de JDBC proporciona una capa de abstracción JDBC que elimina la

necesidad de hacer tediosa codificación JDBCy el análisis de los códigos de proveedor de base de datos de error específicos. El módulo ORM proporciona capas de integración para APIs populares mapeo objeto-relacional, incluyendo JPA, JDO, Hibernate, e iBatis. Usando el paquete ORM puede utilizar todos estos Frameworks de O / R-mapeo en combinación con todas las ofertas de otras características de la Spring, como la transacción declarativa simple función de gestión mencionadas anteriormente.[4][11]

El módulo de OXM proporciona una capa de abstracción que apoya Objeto / XML de mapeo para implementaciones JAXB, Castor, XMLBeans, JiBX y XStream. El Java Messaging Service (JMS) módulo contiene funciones para mensajes de productores y consumidores. El módulo de transacción apoya la gestión de transacciones programáticas y declarativas para las clases que implementar interfaces especiales y para todos sus POJO (Plain Old Java Objects).[4]

2.1.13 Spring Context

En sí Spring Context es un archivo de configuración que provee de información contextual al framework general. Además provee servicios enterprise como JNDI, EJB, email, validación y funcionalidad de agenda.[4]

2.1.14 Spring ORM

En lugar de que Spring proponga su propio módulo ORM (Object-Relational Mapping), para los usuarios que no se sientan confiados en utilizar simplemente JDBC,propone un módulo que soporta los frameworks ORM más populares del mercado, entreellos:[4]

- Hibernate (2.1 y 3.0): es una herramienta de mapeo O/R open source muy popular, que utiliza su propio lenguaje de query llamada HQL.[7]
- iBATIS SQL Maps (1.3 y 2.0). una solución sencilla pero poderosa para hacer externas las declaraciones de SQL en archivos XML.
- Apache OJB (1.0): plataforma de mapeo O/R con múltiples APIs para clientes.

- Entre otros como JDO[4] (1.0 y 2.0) y Oracle TopLink.

Todo esto se puede utilizar en conjunto con las transacciones estándar del framework. Spring e Hibernate es una combinación muy popular. Algunas de las ventajas que brinda Spring al combinarse con alguna herramienta ORM son:[4]

- Manejo de sesión: Spring hace de una forma más eficiente, sencilla y segura la forma en que se manejan las sesiones de cualquier herramienta ORM que se quiera utilizar.
- Manejo de recursos: se puede manejar la localización y configuración de los SessionFactories de Hibernate o las fuentes de datos de JDBC por ejemplo haciendo que estos valores sean más fáciles de modificar.
- Manejo de transacciones integrado: se puede utilizar una plantilla de Spring él para las diferentes transacciones ORM.
- Envolver excepciones: con esta opción se pueden envolver todas las excepciones para evitar las molestas declaraciones y los catch en cada segmento de código necesarios.
- Evita limitarse a un solo producto: Si se desea migrar o actualizar a otra versión de un ORM distinto o del mismo, Spring trata de no crear una dependencia entre la herramienta ORM, el mismo Spring y el código de la aplicación, para que cuando sea necesario migrar a un nuevo ORM no sea necesario realizar tantos cambios.
- Facilidad de prueba: Spring trata de crear pequeños pedazos que se puedan aislar y probar por separado, ya sean sesiones o una fuente de datos (datasource).

[4] Java Data Objects es el proceso de creación de objetos mediante el entorno java.

2.1.15 Spring DAO

El patrón DAO (Data Access Object) es uno de los patrones más importantes y usados en aplicaciones J2EE, y la arquitectura de acceso a los datos de Spring provee un buen soporte para este patrón.[4]

2.1.16 DAO Y JDBC

Existen dos opciones para llevar a cabo el acceso, conexión y manejo de bases de datos: utilizar alguna herramienta ORM o utilizar el template de JDBC (Java DatabaseConnectivity) que brinda Spring. La elección de una de estas dos herramientas es totalmente libre y en lo que se debe basar el desarrollador para elegir es en la complejidad de la aplicación. En caso de ser una aplicación sencilla en la cual únicamente una clase hará dicha conexión, entonces la mejor opción sería el Spring JDBC o en caso contrario que se requiera un mayor soporte y sea más robusta la aplicación se recomienda utilizar una herramienta ORM.[4]

El uso de JDBC muchas veces lleva a repetir el mismo código en distintos lugares, al crear la conexión, buscar información, procesar los resultados y cerrar la conexión. El uso de las dos tecnologías mencionadas anteriormente ayuda a mantener simple este código y evitar que sea tan repetitivo, además minimiza los errores al intentar cerrar la conexión con alguna base datos.

Las clases bases que Spring provee para la utilización de los DAO son abstractas y brindan un fácil acceso a recursos comunes de base de datos. Existen diferentes implementaciones para cada una de las tecnologías de acceso a datos que soporta Spring.[4]

Para JDBC existe la clase JdbcDaoSupport que provee métodos para accesar al DataSource y al template pre-configurado que se mencionó anteriormente: JdbcTemplate.

Únicamente se extiende la clase JdbcDaoSupport y se le da una referencia al DataSource actual.[4]

2.1.17 Spring Web

El módulo web de Spring se encuentra en la parte superior del módulo de contexto, y provee el contexto para las aplicaciones web. Este módulo también provee el soporte necesario para la integración con el framework Struts de Yakarta.[4]

Este módulo también se encarga de diversas operaciones web como por ejemplo: las peticiones multi-parte que puedan ocurrir al realizar cargas de archivos y la relación de los parámetros de las peticiones con los objetos correspondientes (domain objetcs o businessobjects).

Se muestra la típica aplicación web realizada con spring framework.[4]

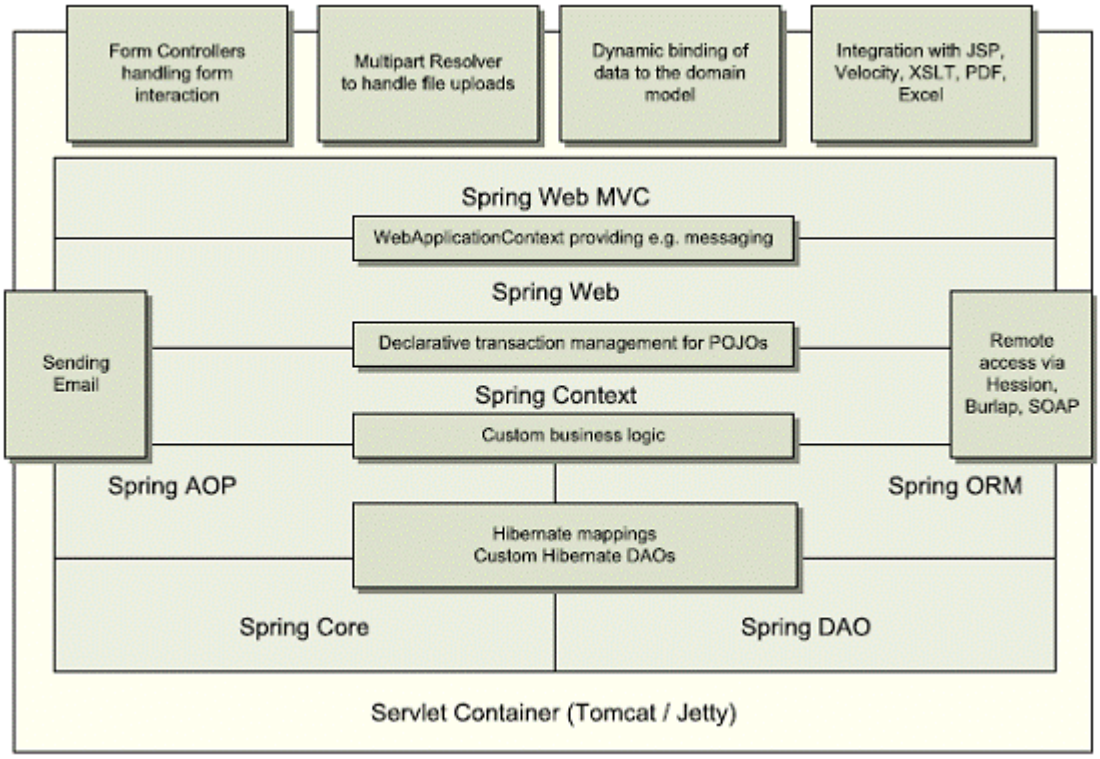


Figura II.5. Modelo de entidades y servicios

2.1.18 Spring Web MVC

Spring brinda un MVC (Model View Controller) para web bastante flexible y altamente configurable, pero esta flexibilidad no le quita sencillez, ya que se pueden desarrollar aplicaciones sencillas sin tener que configurar muchas opciones.[4]

Para esto se puede utilizar muchas tecnologías ya que Spring brinda soporte para JSP, Struts, Velocity, entre otros.[14]

El Web MVC de Spring presenta algunas similitudes con otros frameworks para web que existen en el mercado, pero son algunas características que lo vuelven único:

- Spring hace una clara división entre controladores, modelos de JavaBeans y vistas.
- El MVC de Spring está basado en interfaces y es bastante flexible.
- Provee interceptores (interceptors) al igual que controladores.
- Spring no obliga a utilizar JSP como única tecnología View también se puede utilizar otras.
- Los Controladores son configurados de la misma manera que los demás objetos en Spring, a través de IoC.
- Los web tiers[5]son más sencillos de probar que en otros frameworks.
- El web tiers se vuelve una pequeña capa delgada que se encuentra encima de la capa de business objects.

[5] Nivel de fiabilidad de un centro de datos asociados a cuatro niveles de disponibilidad definidos.

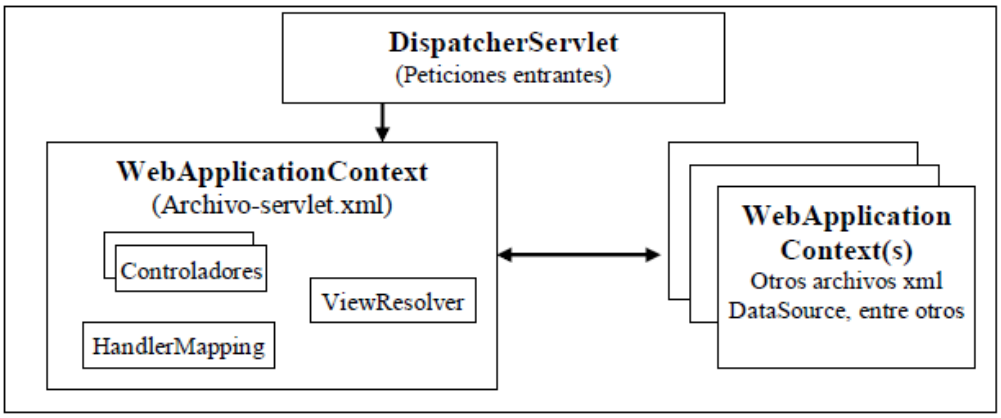


Figura II.6. Arquitectura básica de Spring MVC.

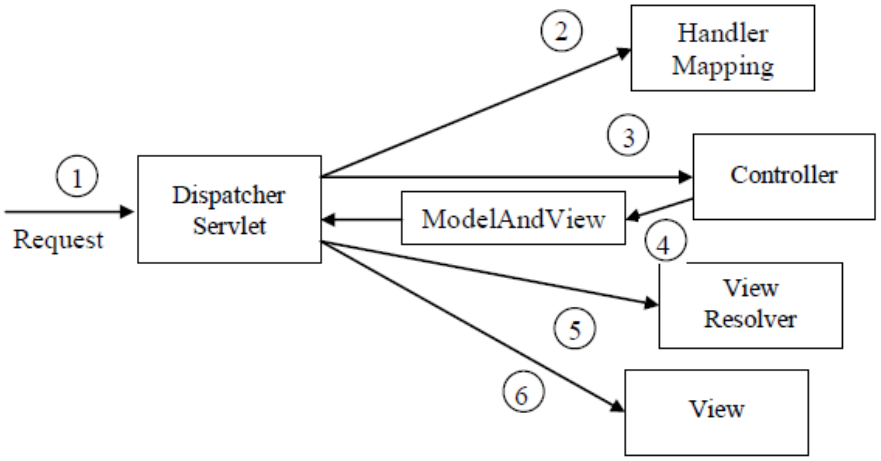


Figura II.7. Ciclo de vida de un Request.

- El navegador manda un request y lo recibe un DispatcherServlet.
- Se debe escoger que Controller manejará el request, para esto el HandlerMapping mapea los diferentes patrones de URL hacia los controladores, y se le regresa al DispatcherServlet el Controller elegido.
- El Controller elegido toma el request y ejecuta la tarea.
- El Controller regresa un ModelAndView al DispatcherServlet.

- Si el ModelAndView contiene un nombre lógico de un View se tiene que utilizar un ViewResolver para buscar ese objeto View que representará el request modificado.
- Finalmente el DispatcherServlet despacha el request al View.

Spring cuenta con una gran cantidad de controladores de los cuales se puede elegir dependiendo de la tarea, entre los más populares se encuentra: Controller y AbstractController para tareas sencillas; el SimpleFormController ayuda a controlar formularios y él envió de los mismos, MultiActionController ayuda a tener varios métodos dentro un solo controlador a través del cual se podrán mapear las diferentes peticiones a cada uno de los métodos correspondientes.[4]

2.1.19 Dispatcher Servlet

Para configurar el DispatcherServlet como el servlet central, se tiene que hacer como cualquier servlet normal de una aplicación web, en el archivo de configuración web.xml (Deployment Descriptor).[4]

```
<servlet>
<servlet-name>training</servlet-name>
<servlet-class>
org.springframework.web.servlet.DispatcherServlet
</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>ejemplo</servlet-name>
<url-pattern>*.htm</url-pattern>
</servlet-mapping>
```

Entonces el DispatcherServlet buscará como está indicado por el tag <servletname> el contexto de aplicación (application Context) correspondiente con el nombre que se haya puesto dentro de ese tag[6]acompañado de la terminación –servlet.xml, en este caso buscará el archivo ejemplo-servlet.xml. En donde se pondrán las definiciones y como su nombre lo indica el contexto de la aplicación dentro de diferentes beans, con sus correspondientes propiedades y atributos, para el caso

[6] Es una marca con tipo que delimita una región en los lenguajes basados en XML.

del Web MVC, se pondrán los diferentes ViewResolver a utilizar, los controladores y sus propiedades, el HandlerMapping, así como los diferentes beans que sean necesarios.[4]

A continuación se muestra como el dispatcher se relaciona con las peticiones del cliente.

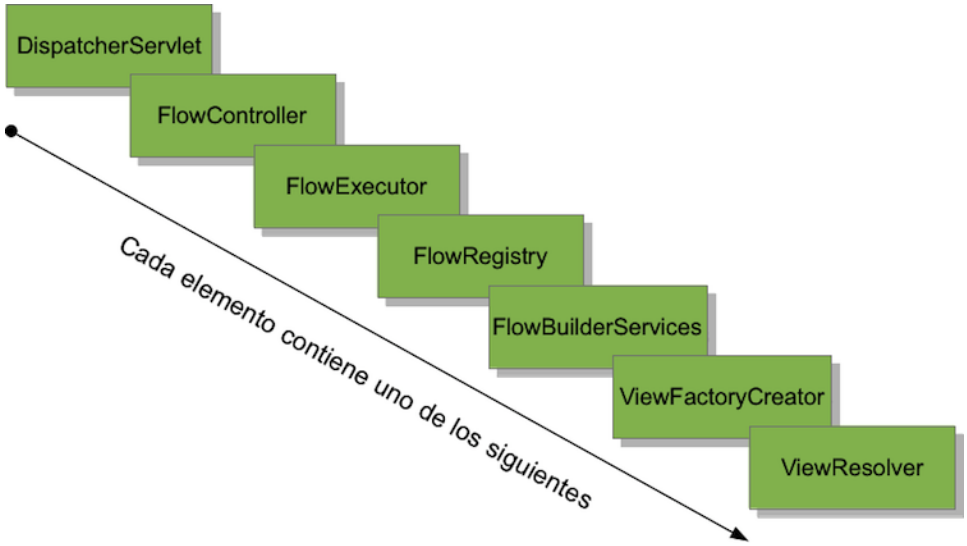


Figura II.8.Petición de clientes con DispatcherServlet.

2.1.20 View Resolvers

En el Spring MVC una vista o View es un bean que transforma los resultados para que sean visibles para el usuario y los pueda interpretar de una mejor forma. En sí un ViewResolver es cualquier bean que implemente la interfaz org.springframework.web.servlet.ViewResolver. Esto quiere decir que un View Resolver es el encargado de resolver el nombre lógico que regresa un controlador en un objeto ModelAndView, a un nombre de archivo físico que el navegador podrá desplegarle al usuario junto con los resultados procesados.[4]

Spring MVC cuenta con cuatro View Resolvers diferentes:

1. InternalResourceViewResolver: Resuelve los nombres lógicos en un archivo tipo View que es convertido utilizando una plantilla de archivos como JSP, JSTL o Velocity.
2. BeanNameViewResolver: Resuelve los nombres lógicos de las vistas en beans de tipo View en el applicationContext del DispatcherServlet.
3. ResourceBundleViewResolver: Resuelve los nombres lógicos de las vistas en objetos de tipo View contenidos en un ResourceBundle o un archivo con extensión .properties.
4. XMLViewResolver: Resuelve los nombres los lógicos de las vistas que se encuentran en un archivo XML separado.

2.1.21 Controladores

Existen varios controladores cada uno especializado para una tarea en particular, claro que como en todos los casos existen algunos que son de uso general. Para poder crear un controlador basta con implementar la interfaz del Controlador deseado, sobrescribir los métodos que sean necesarios para procesar la petición. Esta situación ayuda a poder modularizar la aplicación ya que con la combinación de diversos controladores que sean enfocados a una tarea en particular se puede concentrarse en cada parte de aplicación aislada y tener una mejor estructura que ayudará a detectar más fácilmente las fallas y errores que puedan surgir.[4]

Existe una variedad de controladores, como se muestra en laFigura II.9, los cuales poseen una jerarquía. Spring brinda libertad al desarrollador de escoger que tipo de controlador desea implementar, no lo limita como en algunos otros frameworks.[4]

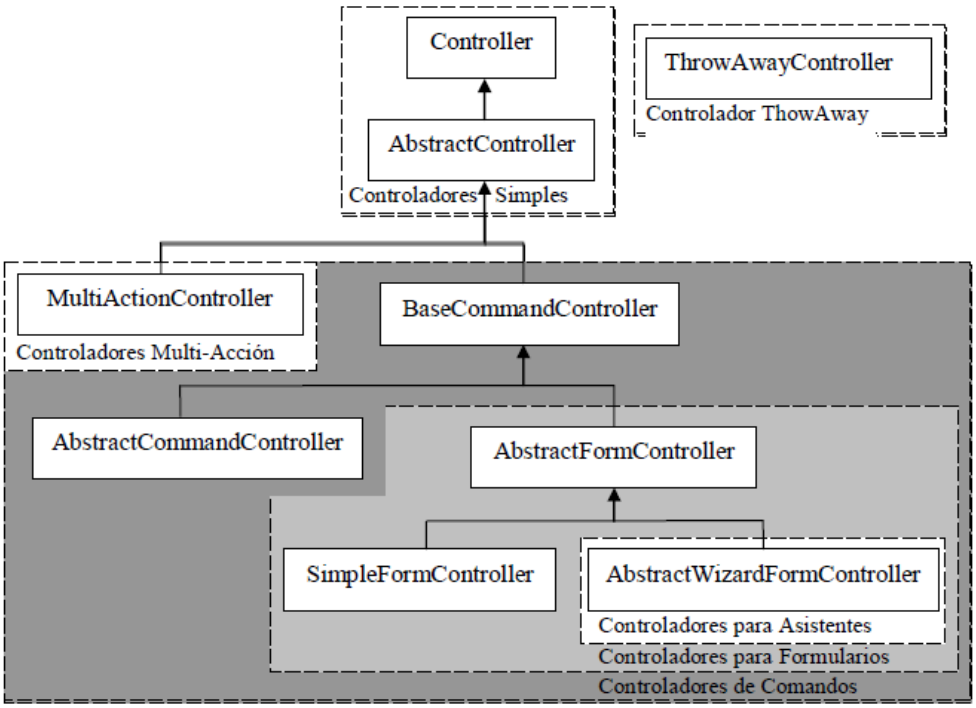


Figura II.9. Controladores de Spring.

La manera en que trabaja cada uno de dichos controladores es similar, cada uno tiene su método específico para procesar las peticiones que haga el usuario, pero todos regresan un objeto tipo ModelAndView, que es como su nombre lo dice el modelo y la vista, ya que se está compuesto de 2 o más atributos. [4]

El principal es el nombre de la vista a la que se va a regresar el modelo para que sea desplegado, y los demás atributos pueden ser parámetros que se le agregan por medio del método .addObject("nombre_parámetro",valor_parámetro). Además el Modelo puede estar formado por un solo objeto o por un Map de Objetos, los cuales se identificarán en la vista con el mismo nombre que se haya mandado dentro del Modelo que se le dio al objeto ModelAndView que se esté regresando para ser procesado.[4]

Tabla II.III. Funcionalidades de Spring Framework.

Función	**Spring Framework**
Manejo de Transacciones	JTA es una de las alternativas disponibles. Pueden usarse diferentes ORM como Hibernate, JDO, JDBC, ODBC
Oportunidad de Transacciones	Soporta atributos de transacción las transacciones anidadas son soportadas sólo si el manejado de transacciones las implementa.
Persistencia de Entidades	Usa implementaciones ORM de terceros como Hibernate, IBATIS, JDO, OJB.
Programación Orientada a Aspectos	Provee servicios de aplicación en forma declarativa, aspectos personalizados pueden ser definidos.
Configuración de la aplicación	En forma primaria se usan archivos XML de configuración, posibilidad de usar Jakarta Commons Attributes a las anotaciones J2SE estándares
Seguridad	Provee integración con la solución Open Source Acegi Security Framework, el mismo que soporta seguridad declarativa basada en el uso de IoC y AOP.
Flexibilidad de Servicios	Cualquier servicio puede ser ensamblado usando un archivo XML de configuración
Integración de Servicios	Spring Framework es desarrollado en forma separada de un servidor de aplicaciones y resulta más dificultoso optimizar su integración. No se aplica si no se usa un servidor de aplicaciones.
Funcionalidad Adicional	Provee oportunidades de integración con varios productos Open-Source, Spring MVC
Testeo	Todos los componentes pueden ser testeados fuera del contenedor
Madurez de la Tecnología y Soporte	La tecnología de Open-Source es soportada por Interface21. Es relativamente madura (2 años desde la liberación 1.0), pero no es un estándar.
Precio	Productos Open-Source son gratuitos.
Documentación	La documentación en formato Javadoc no contiene todos los detalles técnicos, pocos ejemplos detallados.

2.1.22 Ventajas y desventajas

Tabla II.IV. Ventajas de Spring Framework.

Ventajas de Spring Framework
Mayor Velocidad de Desarrollo
Desarrollar más rápido, de forma más organizada y más eficiente. Facilita el desarrollo de funcionalidades específicas y hace que la curva de aprendizaje sea favorable para el desarrollador.
La tendencia en Java
Se ha popularizado en la comunidad de programadores en Java al considerársele una alternativa y sustituta del modelo de Enterprise JavaBean.
Mejor Integración y Flexibilidad
Facilidad de integración con los estándares JEE y herramientas comerciales existentes. Pueden emplearse en cualquier aplicación hecha en Java no solo Web. No obliga a usar un modelo de programación en particular lo que conlleva a una mejor integración con otras herramientas.
Mejor Diseño de Aplicaciones
El framework integra mecanismos basados en las mejores prácticas de programación. Provee de soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria. Spring recomienda el uso de Interfaces, si se decide no utilizar Spring, basta con re-implementar las interfaces.Spring se pueden ejecutar dentro de un contenedor Web o fuera de él en una aplicación Swing.
Aplicaciones más Robustas
Proporciona mayor cantidad de extensiones y mejoras para construir aplicaciones basadas en web que Java Enterprise Edition (JEE). No obliga a usar un modelo de programación en particular lo que conlleva a una mejor integración con otras herramientas.
Manipulación
Facilita la manipulación de nuestros objetos se usen EJBs o no.
Proliferación
Reduce la proliferación de Singletons[7], elimina la necesidad de usar distintos y variados tipos de ficheros de configuración.
Programación
Mejora la práctica de programación, permite el uso o no de EJBs, realizando el mismo tipo de funciones sin ellos.

[7] Está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto.

Tabla II.V. Desventajas de Spring Framework.

Desventajas de Spring Framework
Configuración
La configuración de Spring es compleja ya que para cada servicio que se tenga se ha configurarlo en un XML de configuración. Aunque hay otras formas de configuración de Spring aparte del XML puro: programando por medio de la API, mediante un estándar JSR y con un mínimo XML y anotaciones. La primera dice que la configuración de Spring está inflada y que si se tienen 100 acciones que trabajan con 100 servicios hace falta configurar cada uno de ellos.
Tipado
Perdida de las ventajas del tipado fuerte, ya que al inyectar objetos los fallos sólo pueden detectarse en tiempo de ejecución.
Container
El contenedor de Spring no es ligero (si se usan todos los módulos disponibles), no es recomendable su uso en aplicaciones de tiempo real o en aplicaciones para móviles.
Software
Spring promete un software poco acoplado, pero en realidad, no se preocupa mucho por ello y cree que una vez introducido en la aplicación ésta no puede vivir sin él.

2.1.23 ¿Qué hay de nuevo en Spring 3?

El Spring Framework está basado en Java 5 y Java 6 es totalmente compatible. Además, el soporte es compatible con J2EE 1,4 y Java EE 5, mientras que al mismo tiempo, introducir un apoyo temprano para Java EE 6.[4]

2.1.23.1La inversión del contenedor control (IOC)

Algunas de las características fundamentales del proyecto JavaConfig se han añadido a la infraestructura Spring ahora. Esta significa que las anotaciones siguientes están directamente soportadas:[7][4]

@Configuration	@Bean
@DependsOnG	@Primary
@Lazy	@Import
@ImportResource	@Value

He aquí un ejemplo de una clase Java proporciona la configuración básica utilizando las características JavaConfig nuevos:[4]

```
packageorg.example.config;
@Configuration
public class AppConfig {
private@Value("#{jdbcProperties.url}") String jdbcUrl;
private@Value("#{jdbcProperties.username}") String username;
private@Value("#{jdbcProperties.password}") String password;
@Bean
publicFooService fooService() {
return new FooServiceImpl(fooRepository());
}
@Bean
publicFooRepository fooRepository() {
return new HibernateFooRepository(sessionFactory());
}
@Bean
publicSessionFactory sessionFactory() {
// wire up a session factory
AnnotationSessionFactoryBean asFactoryBean =new AnnotationSessionFactoryBean();
asFactoryBean.setDataSource(dataSource());
// additional config
returnasFactoryBean.getObject();
}
@Bean
publicDataSource dataSource() {
return new DriverManagerDataSource(jdbcUrl, username, password);
}
}
```

Para conseguir que esto funcione es necesario agregar la entrada siguiente componente de exploración en su aplicación mínima contexto XML archivo.[4]

```
<context:component-scan base-package="org.example.config"/>
<util:propertiesid="jdbcProperties"location="classpath:org/example/config/jdbc.properties"/>
```

O bien, puede arrancar una clase de configuración directamente mediante @ AnnotationConfigApplicationContext:[4]

```
public static void main(String[] args) {
ApplicationContext ctx = new AnnotationConfigApplicationContext(AppConfig.class);
FooService fooService = ctx.getBean(FooService.class);
fooService.doStuff();}
```

2.1.23.2¿Qué es IoC?

Este apartado se trata sobre la implementación de la inversión de control (IoC) de spring framework. IoC también se conoce como inyección de dependencias (DI). Se trata de un proceso por el cual se define los objetos de dependencias, es decir, los objetos con los que trabajamos, sólo a través de argumentos del constructor.[4][14]

Los paquetes org.springframework.beans y org.springframework.context son la base para el contenedor Spring Framework COI. La interfaz BeanFactory proporciona una avanzada configuración capaz de gestionar cualquier tipo de objeto mecanismo. ApplicationContext es un sub-interfaz de BeanFactory. Se añade una más fácil integración con Spring AOP características; mensaje manejo de recursos (para su uso en internacionalización), publicación de eventos, y la capa de aplicación específica contextos como el WebApplicationContext para uso en aplicaciones web.[4]

En resumen, la BeanFactory proporciona el marco de configuración y la funcionalidad básica, y el ApplicationContext añade más funcionalidad específica de la empresa. El ApplicationContext es un súper conjunto completo de la BeanFactor.[4]

En Spring, los objetos que forman la columna vertebral de su aplicación y que son gestionados por el IoC Spring recipiente se denominan beans. Un bean es un objeto que se crea una instancia, ensamblado, y de otra gestionada por un contenedor de IoC Spring. De lo contrario, un bean es simplemente uno de los muchos objetos de la aplicación.[4]

El contenedor creará los objetos, cablear juntos, configurarlos y gestionar su ciclo de vida completo, desde la creación hasta su destrucción. El contenedor de Spring utiliza inyección de dependencias (DI) para gestionar los componentes que conforman una aplicación. Estos objetos se llaman Spring Bean.[4]

IoC es también conocida como la inyección de dependencia (DI). Inversión de Control es una técnica que permite la configuración objeto a ser movido fuera de código y en un archivo de

configuración. Con el Ioc de Spring, esto se suele hacer con un archivo XML. En términos técnicos IoC es un modelo de diseño de software y un conjunto de técnicas de programación asociados en los que se invierte el flujo de control de un sistema en comparación con el modo de interacción ágil. En la IoC, en lugar de una aplicación que llama a la estructura, que es el marco en el que llama a los componentes especificados por la aplicación. Este patrón es similar a la que los agentes adoptan con sus clientes en películas de Hollywood. Al navegar 'No me llames, "Yo te que llamo ". Esta es la razón por la IoC es también conocido como el patrón de Hollywood.[4]

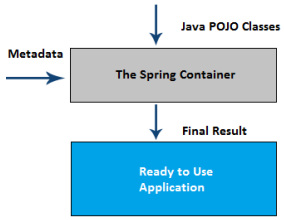


Figura II.10. Inversión de Control

Hay tres formas de inyección de dependencias.

- Constructor de inyección
- Inyección de Setter
- Interfaz de inyección

2.1.23.3 Inyección de Constructor:

En este enfoque, un contenedor IoC utiliza el constructor para inyectar la dependencia. Todas las dependencias (simples o referencias) se declaran en el constructor. Una de las ventajas de la inyección de constructor es que todas las dependencias se declaran en un solo paso. Esto también ayuda en la comprensión de si la clase depende de demasiados servicios.[4][14]

ConstructorInjection.java

```
package net.roseindia;
public class ConstructorInjection
{
        private String message = null;
        public ConstructorInjection(String message)
        { this.message = message;}
        public String getMessage()
         {    return message;  }
        public void setMessage(String message)
        {    this.message = message;  }
}
```

2.1.23.4 Inyección de Setter:

En este enfoque utiliza inyección de dependencias Setters para inyectar los recursos necesarios o dependencias. Cada uno de los objetos que la clase depende tendrá un colocador y el contenedor IoC utilizará los emisores para proporcionar el recurso en tiempo de ejecución.[4]

SetterInjection.java

```
package net.roseindia;
public class SetterInjection
{       private String message = null;
        public String getMessage() {return message;  }
        public void setMessage(String message) {  this.message = message;  }
}
```

2.1.23.5Inyección de interfaz:

Son las implementaciones concretas de una interfaz para el objeto dependiente de acuerdo a la configuración. La diferencia principal entre la inyección y la interfaz de los dos anteriores es que en la interfaz de inyección, cualquiera de las implementaciones de la interfaz puede ser inyectada, mientras que con los otros dos, el objeto de la clase especificada se inyecta. Spring no ofrece soporte directo para inyección Interface.[4]

2.1.23.6Visión General del Contenedor

El org.springframework.context.ApplicationContext interfaz representa la Spring IoC contenedor y es responsable de instanciar, configuración y montaje de los beans antes mencionados.

El recipiente recibe las instrucciones sobre qué objetos para crear una instancia, configurar y ensamblar con la lectura configuración de metadatos. Los metadatos de configuración se representan en XML, anotaciones Java o Java código. Te permite expresar los objetos que componen la aplicación y las interdependencias ricas, Spring Framework entre tales objetos.[4]

Varias implementaciones de la interfaz ApplicationContext se suministran fuera de la caja con Spring. En las aplicaciones independientes, es común para crear una instancia de ClassPathXmlApplicationContext o FileSystemXmlApplicationContext. Mientras que XML ha sido el formato ágil para la definición de metadatos de configuración puede indicar al contenedor utilizar las anotaciones de Java o el código como el formato de metadatos proporcionando una pequeña cantidad de configuración XML para mediante declaración habilitar el soporte para estos formatos de metadatos adicionales.[4]El siguiente diagrama es una vista de alto nivel de cómo funciona la Spring. Sus clases de la aplicación se combinan con metadatos de la configuración de modo que después de la ApplicationContext se crea y se inicializa, se tener un sistema completamente configurado y ejecutable o aplicación.[4]

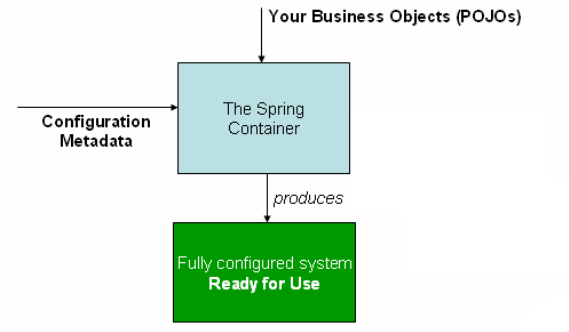


Figura. 2.1.23.6-1. Visión general del contenedor.

Composición de metadatos basado en la configuración XML

Puede ser útil tener definiciones de bean abarcar varios archivos XML. A menudo, cada archivo XML de configuración representa una capa lógica o módulo en su arquitectura.

Usted puede utilizar el application contex de la aplicación para cargar los bean de todos estos fragmentos XML.[4]

Como alternativa, utilice una o más ocurrencias del elemento <import/> para cargar las definiciones de bean otro archivo o archivos. Por ejemplo:

```
<beans>
<import resource="services.xml"/>
<import resource="resources/messageSource.xml"/>
<import resource="/resources/themeSource.xml"/>
<bean id="bean1" class="..."/>
<bean id="bean2" class="..."/>
</beans>
```

En el ejemplo anterior, las definiciones externas bean se cargan a partir de tres archivos, services.xml, messageSource.xml, y themeSource.xml. Todas las rutas son relativas a la ubicación del archivo de definición haciendo la importación, así services.xml deben estar en el mismo directorio o ubicación classpath el archivo haciendo el importador, mientras que messageSource.xml y themeSource.xml debe estar en una recursos de ubicación por debajo de la ubicación del archivo de importación.

El ApplicationContext es la interfaz para una fábrica de avanzada capaz de mantener un registro de diferentes beans y sus dependencias. Usando el método getBean se puede recuperar instancias de los beans.

El ApplicationContext le permite leer las definiciones bean y acceder a ellos de la siguiente manera:

```
// create and configure beans
ApplicationContext context =
new ClassPathXmlApplicationContext(new String[] {"services.xml", "daos.xml"});
// retrieve configured instance
```

```
PetStoreServiceImpl service = context.getBean("petStore", PetStoreServiceImpl.class);
// use configured instance
List userList service.getUsernameList();
```

Utilice getBean () para recuperar instancias de los beans. La interfaz tiene un ApplicationContext algunos otros métodos para la recuperación de frijoles, pero lo ideal es el código de aplicación nunca debe utilizarlos. En efecto, el código de aplicación no debe tener llamadas al método getBean () en absoluto, y por lo tanto ninguna dependencia en Spring APIs en absoluto. Por ejemplo, la integración de Spring con frameworks web proporciona a la dependencia inyección para diferentes clases de framework de desarrollo web como controladores y los frijoles JSF administrados.

2.1.23.7ApplicationContext

Context package agrega la interfaz ApplicationContext, lo que mejora la funcionalidad BeanFactory en un estilo más orientado al Framework. Muchos usuarios utilizan ApplicationContext de forma totalmente declarativa, ni siquiera tener que crear manualmente, pero en lugar de confiar en las clases de apoyo, como ContextLoader automáticamente para crear instancias de un ApplicationContext como parte del proceso normal de inicio de una aplicación Web J2EE. [4]

La base para el paquete de ApplicationContext, está situado en el paquete org.springframework.context derivado del BeanFactory, proporciona toda la funcionalidad de BeanFactory. Para permitir trabajar de una manera más orientada, utilizando capas y contextos jerárquicos, el paquete contexto también proporciona las siguientes funcionalidades:

1. MessageSource, facilitando el acceso a los mensajes de i18n-estilo.
2. El acceso a los recursos, tales como URLs y archivos.
3. Evento propagación a los beans que implementan la interfaz ApplicationListener.
4. Carga de múltiples contextos (jerárquica), permitiendo que cada uno se centra en una capa determinada, por ejemplo la capa de banda de una aplicación.

Tabla II.VI.Características BeanFactory vs ApplicationContex.

Características	BeanFactory	ApplicationContext
Bean instantiation/wiring	Yes	Yes
Automatic BeanPostProcessor registration	No	Yes
Automatic BeanFactoryPostProcessor registration	No	Yes
Convenient MessageSource access (for i18n)	No	Yes
ApplicationEvent publication	No	Yes

2.1.23.8Anotaciones Basadas en la configuración del contenedor (IoC)

Spring introduce la posibilidad de aplicar las propiedades necesarias con el @Required anotación[8]. Con Spring es posible seguir ese mismo enfoque general para conducir inyección Spring dependencia.[4]

En esencia,@Autowired, @Resource , @PostConstruct y @PreDestroy . El uso de estas anotaciones también requiere que ciertos BeanPostProcessors estar registrados en el contenedor Spring. Como siempre, estos pueden ser registrados como definiciones beans individuales, pero también pueden ser registradas implícitamente al incluir la siguiente etiqueta en una configuración de Spring basado en XML:[4]

```
<?xml version="1.0" encoding="UTF-8"?>
<beansxmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-3.0.xsd">
<context:annotation-config/>
</beans>
```

[8]Tenga en cuenta que <context:annotation-config/> sólo busca anotaciones de Bean en el contexto de la aplicación misma se define In, Esto significa que, si se pone <context:annotation-config/> en un WebApplicationContext un DispatcherServlet , sólo chequea por el bean @Autowired en sus controladores, y no sus servicios.

2.1.23.8.1 @Requiered

La @Required anotación se aplica a los métodos setter de las propiedad de un bean, como en el ejemplo siguiente:

```
público SimpleMovieLister clase {

    MovieFinder MovieFinder privado;

@ Required
public void setMovieFinder (MovieFinder MovieFinder) {
este MovieFinder = MovieFinder.;
    }
}
```

Esta anotación indica simplemente que la propiedad de bean afectada deben estar ocupados en el tiempo de configuración: o bien a través de un valor de la propiedad en una definición explícita o a través de bean autowiring. El contenedor se producirá una excepción si la propiedad de bean afectada no ha sido poblada, lo que permite un fracaso ansioso y explícito, evitando NullPointerException s o similares en el futuro.[4]

2.1.23.9@Autowired

Permite reducir el esfuerzo de escribir propiedades o argumentos de constructor y como era de esperar, el @Autowired anotación se puede aplicar:

A métodos setter, a los métodos con nombres arbitrarios y / o argumentos múltiples, a los constructores, a ApplicationContext mediante la adición de la anotación a un campo o método que espera un conjunto de ese tipo.[4]

Tabla II.VII. Métodos Autowired.

Modo	Explicación
no	Es por defecto que definir no autowiring.
byName	Autowiring se realiza por nombre de propiedad.
byType	Autowiring se realiza haciendo coincidir el tipo de datos de nombre de propiedad.
constructor	Autowiring se realiza haciendo coincidir el tipo de datos de nombre de la propiedad con el argumento de constructor inmobiliario.
autodetectar	Cuando constructor por defecto sin argumentos, se auto-hilos según el tipo de datos o auto-wire por constructor.

2.1.23.10 Validaciones

Spring cuenta con un Validator de interfaz que se puede utilizar para validar los objetos.

Vamos a proporcionar un comportamiento de validación de la Calse Person mediante la aplicación de los dos métodos siguientes de la org.springframework.validation.Validator interfaz:

soportes (Class) - ¿Puede esta Validator validar las instancias de la clase suministradas?

validar (Object, org.springframework.validation.Errors) - valida el objeto dado y en caso de errores de validación, Teniendo en cuenta estos registros de errores con el objeto.[4]

La implementación de un Validator es bastante sencillo, sobre todo cuando se sabe de la ValidationUtils clase de ayuda que el Spring Framework también proporciona.[15]

```
publicclass PersonValidator implements Validator {
/**
   * This Validator validates just Person instances
   */
publicboolean supports(Class clazz) {
return Person.class.equals(clazz);
   }
publicvoid validate(Object obj, Errors e) {
ValidationUtils.rejectIfEmpty(e, "name", "name.empty");
       Person p = (Person) obj;
if (p.getAge() < 0) {
e.rejectValue("age", "negativevalue");
       } elseif (p.getAge() > 110) {
e.rejectValue("age", "too.darn.old");       }    }}
```

2.1.24 Acceso a Datos

Esta parte de la documentación de referencia está relacionada con el acceso a datos y la interacción entre la capa de acceso de datos y de negocios o la capa de servicio. Integral de apoyo del resorte de gestión de la transacción está cubierta con algún detalle, seguido de una cobertura completa de los marcos de acceso diferentes de datos y tecnologías de que el marco se integra con Spring.[4]

2.1.24.1Hibernate

Vamos a empezar con una cobertura de Hibernate[9] en un ambiente de Spring, utilizando para demostrar el enfoque que lleva hacia la integración de Spring de O / R mappers. Esta sección cubre muchos temas en detalle y mostrar diferentes variaciones de las implementaciones de DAO y la demarcación de la transacción. La mayor parte de estos patrones puede ser directamente traducida a todas las herramientas ORM otros compatibles.[4][7]

2.1.24.1.1 HibernateTemplate

El modelo de programación básica para plantillas es como sigue, para los métodos que pueden formar parte de cualquier objeto de acceso a datos personalizado o servicio de negocio. No hay restricciones sobre la ejecución del objeto que rodea a todos, sólo tiene que proporcionar un Hibernate SessionFactory . Se puede obtener el último desde cualquier lugar, pero preferiblemente como referencia del bean de un contenedor Spring IoC a través de un simple setSessionFactory. [4]

Los siguientes fragmentos muestran una definición DAO en un contenedor Spring.[4]

[9]A partir de Spring 2.5, requiere Spring Hibernate 3.1 o superior.Hibernate 3,0ni Hibernate 2.0 y 2,1 no reciben soporte.

```
<beans>
<beanid="myProductDao"class="product.ProductDaoImpl">
<propertyname="sessionFactory"ref="mySessionFactory"/>
</bean>
</beans>
publicclass ProductDaoImpl implements ProductDao {
private HibernateTemplate hibernateTemplate;
publicvoid setSessionFactory(SessionFactory sessionFactory) {
this.hibernateTemplate = newHibernateTemplate(sessionFactory);
   }
public Collection loadProductsByCategory(String category) throws DataAccessException {
returnthis.hibernateTemplate.find("from test.Product product whereproduct.category=?",
category);
}
}
```

HibernateTemplate se asegurará de que sesión de instancia se abre y se cierra correctamente, y automáticamente participar en transacciones. HibernateTemplate ofrece métodos alternativos de conveniencia que pueden reemplazar dichas implementaciones de una devolución de llamada de línea. Además, Spring proporciona un conveniente HibernateDaoSupport clase base que proporciona un setSessionFactory método para recibir una SessionFactory, y getSessionFactory() y getHibernateTemplate() para su uso por las subclases. En combinación, esto permite implementaciones DAO muy simples para los requisitos típicos:[4]

```
publicclass ProductDaoImpl extends HibernateDaoSupport implements ProductDao {

public Collection loadProductsByCategory(String category) throws DataAccessException {
returnthis.getHibernateTemplate().find(
"from test.Product product where product.category=?", category);
}
}
```

2.1.24.1.2 Ejecución DAOs basado en API Hibernate 3

La característica "Sesiones contextuales", donde Hibernar maneja una sesión actual por transacción. Esto es aproximadamente equivalente a la sincronización de Spring Hibernate de una sesión por transacción.[4]

```
publicclass ProductDaoImpl implements ProductDao {
private SessionFactory sessionFactory;
publicvoid setSessionFactory(SessionFactory sessionFactory) {
this.sessionFactory = sessionFactory;
   }
public Collection loadProductsByCategory(String category) {
returnthis.sessionFactory.getCurrentSession()
            .createQuery("from test.Product product where product.category=?")
.setParameter(0, category)
            .list();    }}
```

El código anterior sigue el patrón DAO inyección de dependencia: se adapta muy bien en un contenedor Spring IoC, al igual que lo haría si codifica con Spring HibernateTemplate . También se puede configurar de manera clara con Java, solo tiene que crear una instancia y llamar setSessionFactory. Como definición en un Bean Spring, se vería de la siguiente manera:[4]

```
<beans>
<beanid="myProductDao"class="product.ProductDaoImpl">
<propertyname="sessionFactory"ref="mySessionFactory"/>
</bean>
</beans>
```

La principal ventaja de este estilo DAO es que depende del API Hibernate, sin importar cualquier clase Spring se requiere.[4]

Spring LocalSessionFactoryBean soporta Hibernate SessionFactory.getCurrentSession() método para cualquier estrategia de transacción Spring, devolviendo la sesión actual, incluso con HibernateTransactionManager. [4]

2.1.25 Spring Framework Web

2.1.25.1Características de Spring Web MVC

Módulo de Spring web ofrece una gran cantidad de características de apoyo, entre ellas tenemos las mencionadas a continuación:[4]

Una clara separación de roles - controlador, validador, objeto de comando, objeto de formulario, objeto modelo, DispatcherServlet , asignación de controlador, resolución de vista, etc.[4]

Adaptabilidad, no intrusivo. Utilice lo que usted necesita controlador subclase (normal, orden, forma, multi-acción) para un escenario determinado en lugar de derivar de un único controlador para todo.[4]

Código de negocio reutilizable - sin necesidad de duplicación. Puede utilizar los objetos existentes de negocio como objetos de comando o formulario en lugar de reflejar ellas para extender una clase marco básico particular.[4]

Modelo flexible de transferencia.-Transferencia a través de un modelo de nombre / valor Map soporta una fácil integración con cualquier tecnología de visión.[4]

Tema de resolución, soporte para JSP con o sin biblioteca de etiquetas de primavera, el apoyo a JSTL, el apoyo a la velocidad sin necesidad de puentes adicionales, etc.[4]

Una simple pero potente biblioteca de etiquetas JSP conocida como la biblioteca de etiquetas de primavera que ofrece soporte para funciones tales como la unión de datos y temas. Las etiquetas personalizadas permiten una gran flexibilidad en términos de código marcado. [4]

CAPÍTULO III GUÍA METODOLÓGICA PARA LA IMPLEMENTACIÓN DE UN SISTEMA DE GESTIÓN EN LA COMPROTEC

3.1 NECESIDAD DE LA GUÍA METODOLÓGICA.

En los últimos años las aplicaciones web han sufrido un gran auge gracias en gran parte al internet y la proliferación de sitios web, sobre todo con el fin de fomentar el comercio electrónico.

Su facilidad de administración centralizada las hace ideales tanto para su despliegue en internet como en intranets corporativas.

A través de la guía metodológica se propone promover la apropiación y la socialización de las experiencias exitosas del desarrollo de software. Se dará aseguramiento de la calidad en el que se estructura la guía, ofreciendo un consistente marco referencial que permite a toda experiencia formativa del desarrollo de software, tomar conocimiento del estado de situación en la que se encuentra, así como plantear procesos de mejora a la medida de dichas experiencias, y no exigir ninguna condición en particular.

3.1.1 CARACTERÍSTICAS DE LA GUÍA METODOLÓGICA.

La metodología DAWE está desarrollada específicamente para aplicaciones web empresariales complejas, ya que se aplica la utilización de tecnologías JAVA como el framework Spring MVC, para así estructurar una guía que servirá a los desarrolladores de aplicaciones web garantizar la calidad de las mismas y manteniendo una organización que ayudará a realizar las aplicaciones de forma rápida.

3.1.2 FASES DE LA METODOLOGÍA DAWE

La metodología DAWE exige un enfoque sistemático y secuencial del desarrollo del software. Este paradigma abarca las actividades representadas en la Figura III.1.

Figura III.1. Fases de la Metodología.

3.1.2.1 FASE DE PLANIFICACIÓN

Determinación de los principales requerimientos por parte de los usuarios así como también definir las tecnologías a utilizar durante todo el proyecto.

3.1.2.2 FASE DE DESARROLLO DE CAPAS

Establecer las principales capas a utilizar durante el proyecto, e implementación de las interfaces que brindaran calidad y mejor manejo a la aplicación web.

3.1.2.3 FASE DE INTEGRACIÓN DEL FRAMEWORK SPRING MVC

Desarrollo de todos los beans a utilizar en nuestro contenedor de objetos así como también definir los controladores a utilizar.[13]

3.1.2.4 FASE DE INTEGRACIÓN DE LA CAPA DE PRESENTACIÓN

Generar todas las vistas presentadas al usuario así como el desarrollo de todas las validaciones respectivas y la implementación de spring security para el proceso de administración de permisos.[15]

3.1.2.5 FASE DE PRUEBAS

Realizar una implementación de pruebas para la utilización de los usuarios finales y mostrar errores existentes para su respectiva mejora.

3.1.2.6 FASE DE INSTALACIÓN

Instalación final de la aplicación en el servidor y la respectiva configuración de todas las conexiones.

3.2 FASE DE PLANIFICACIÓN

3.2.1 ESTUDIO DE REQUERIMIENTOS

Esta fase es la clave principal de todo desarrollo de software, ya que en esta fase se debe realizar un estudio minucioso de las necesidades fundamentales del usuario, con ayuda del usuario definiremos los requerimientos, debe tener ya bien definido y documentado todos los requerimientos a implantar en la aplicación web.

Definir una propuesta adecuada que se adapte a la complejidad del problema, para así garantizar la calidad del producto y la satisfacción del cliente.

Con nuestro equipo de trabajo se debe de realizar reuniones específicas con los usuarios, ya que esto conllevará a un futuro buen desempeño del sistema y evitar futuros problemas de integridad y necesidades futuras.

Al momento de definir los requerimientos a su vez, seguir identificando un modelo para el futuro desarrollo de la base de datos, mediante la cual se define las restricciones y las integridades de la solución.

Entre las principales tareas de esta fase tenemos las siguientes:

- Recolección de requerimientos funcionales.
- Recolección de requerimientos no funcionales.
- Definición de una propuesta de solución tecnológica.

La identificación de requerimientos funcionales se debe definir los procesos de gestión de todas las entidades (inserción, actualización, eliminación, consultas) así como la visualización de los roles de usuario para la implantación respectiva de la seguridad.

A través de la identificación de los requerimientos funcionales se puede detectar las posibles eventualidades del sistema en los casos que puedan reflejar posibles errores.

La recolección de requerimientos no funcionales se debe definir las propiedades del sistema como la disponibilidad, rapidez, seguridad, escalabilidad. De la misma forma a través de estos requerimientos se puede seguir evaluando las necesidades tecnológicas de adaptación para el sistema, ya que se puede definir el hardware y software necesario para que nuestra aplicación cumpla con todas las necesidades establecidas en los requerimientos no funcionales.

3.2.2 DISEÑO DE LA BASE DE DATOS

Una vez identificados los requerimientos en la etapa anterior se debe proceder a esquematizar la base de datos, fase en la cual vamos a definir el comportamiento del sistema a través de la creación de las entidades y sus relaciones.

Esta etapa solo se definirá la estructura del sistema como las entidades, atributos, relaciones y restricciones. En lo que se refiere al manejo de procedimientos almacenados y funciones es innecesario realizar ya que se dará soporte a estas transacciones a través de un mapeador que más adelante definiremos.

Esta fase de desarrollo de la base de datos se debe de realizar un análisis a fondo porque de esta parte de proceso dependerá mucho el desarrollo ágil de nuestro sistema, es recomendable definir una reunión con el equipo de trabajo y los usuarios para exponer las ideas expuestas en el esquema de base de datos y así verificar la correcto bosquejo del mismo.

3.2.3 ARQUITECTURA DE SOFTWARE

El fin de esta etapa es identificar las herramientas necesarias a utilizar para el desarrollo del sistema, como IDE de desarrollo, herramientas CASE, lenguaje de programación, servidor de aplicaciones a utilizar, frameworks, instrumentos que ayudarán al desarrollo ágil de nuestra aplicación.

A través de este estudio se puede determinar la factibilidad así como el tiempo estimado de desarrollo, ya que a través de estas herramientas se puede dar garantía de un desarrollo ágil y sin complicaciones, estandarizando todos los procesos a ejecutar durante toda la realización del proyecto.

Esta etapa solo se realizaráentre el equipo de desarrollo del sistema, ya que no es de mucha relevancia exponerlo ante los usuarios finales, por cuanto no podrán tomar decisiones sobre su manejo, ya que a través de la experiencia de los desarrolladores se precisara con exactitud toda la arquitectura adecuada para entregar una aplicación web de calidad.

3.2.4 ETAPA DE DOCUMENTACIÓN

La actividad a ejecutar es documentar todos los procesos anteriores mencionados como la etapa de estudio de requerimientos, diseño de la base de datos y arquitectura de software para posteriormente entregar una propuesta firme de cómo se ejecutara el desarrollo de todo el proyecto y también que servirá de constancia del trabajo realizado en la primera fase, es de vital importancia entregar un documento final al cliente, con las especificaciones te todo lo realizado en esta fase, para que este documento de la garantía a ambas partes tanto desarrolladores como clientes de que la aplicación cumplirá con todo lo establecido.

3.3 FASE DE DESARROLLO DE CAPAS

3.3.1 MAPEO DE BASE DE DATOS

Una vez realizado el proceso de creación de la base de datos se debe a realizar la creación del proyecto ya establecido las herramientas establecidas en el proceso de arquitectura de software, para nuestro caso se empleara el IDE NetBeans tal como se muestra en la figura siguiente:[13]

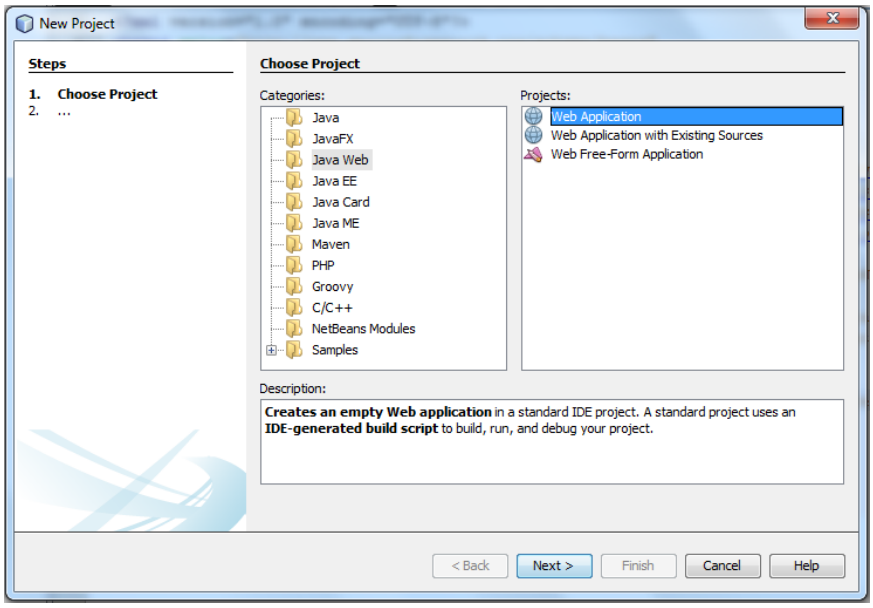


Figura III.2. Creación de la aplicación

Se debe especificar también los frameworks a utilizar como este caso se decide utilizar Spring Web MVC e Hibernate tal como se muestra en la figura siguiente:[7]

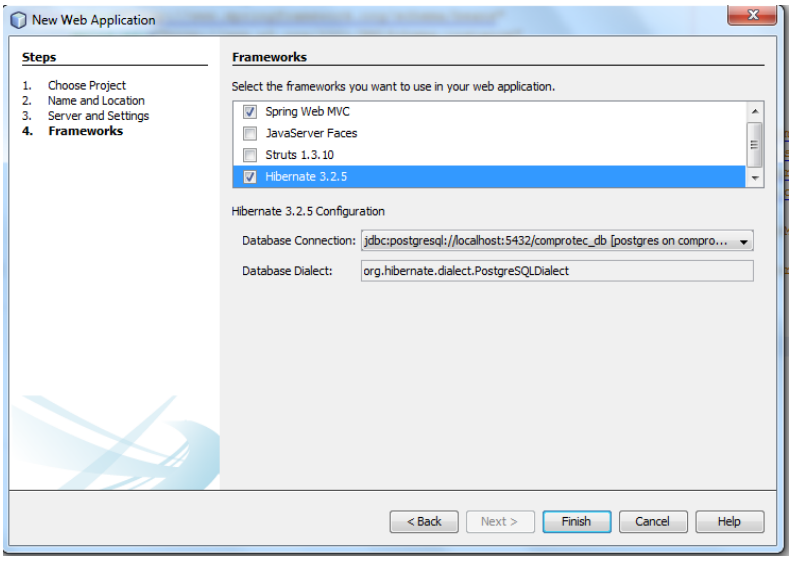


Figura III.3. Framework utilizado.

Para proceder con la creación del mapeo de datos se debe crear un paquete y agregar un archivo de configuración del mapeador Hibernate como se muestra en la figura.

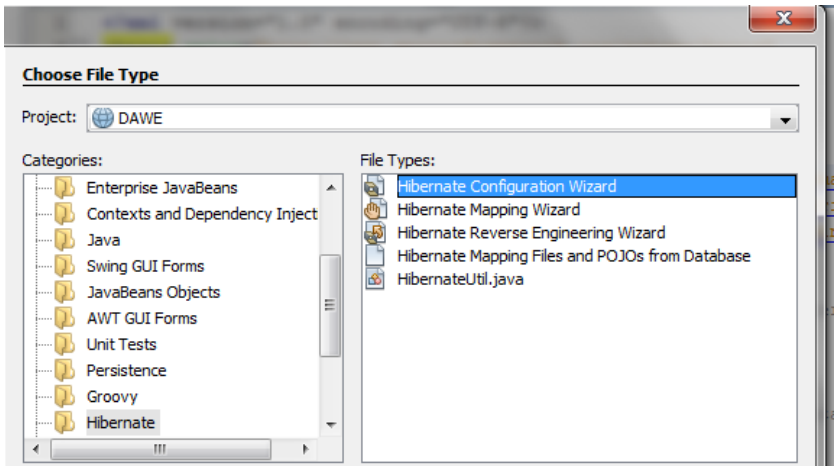


Figura III.4. Hibernate Configuration Wizard

Luego se debe proceder un archivo de ingeniería en reversa de Hibernate como se muestra en la figura:

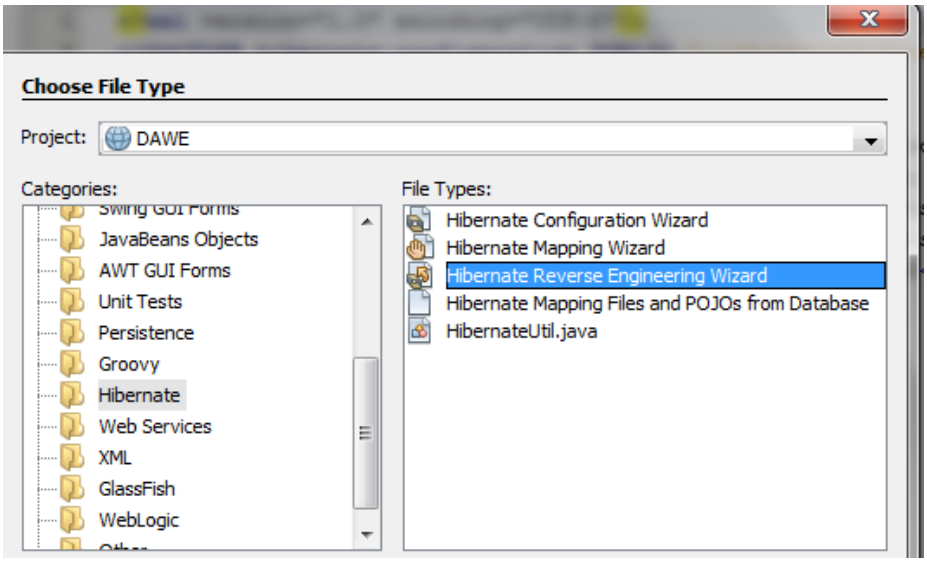


Figura III.5. Reverge Engineering Wizard.

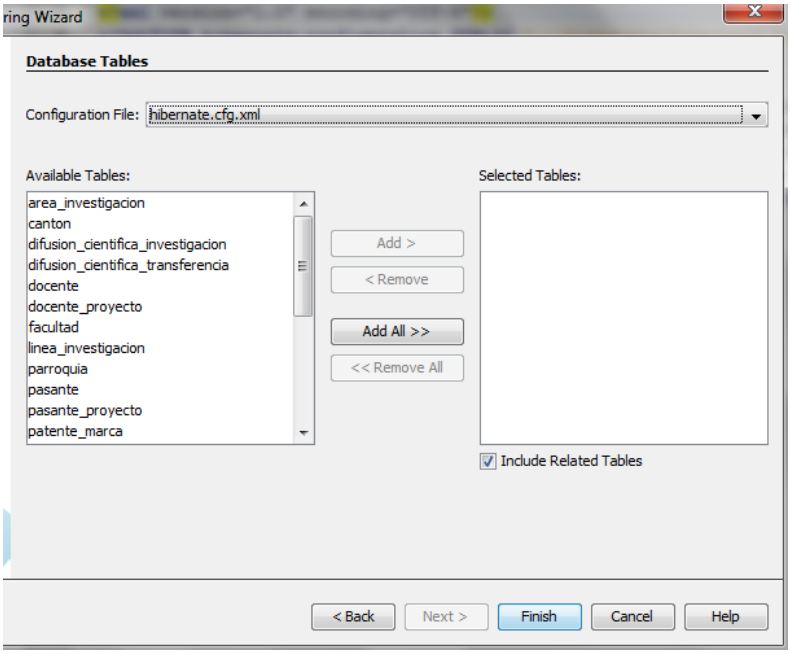


Figura III.6. Configuración del Archivo Hibernate.cfg.

Finalmente se creaun archivo POJO:

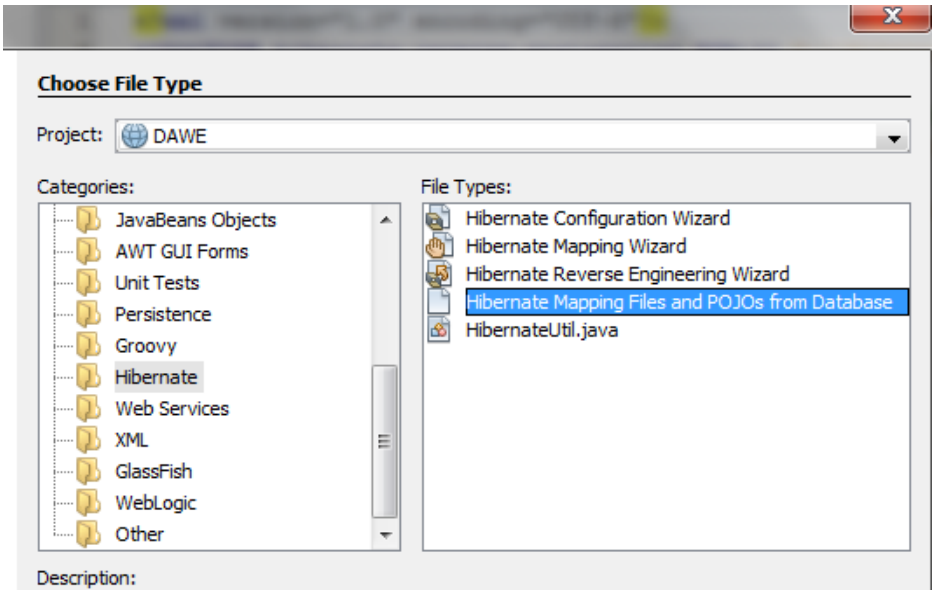


Figura III.7. Creación del Archivo POJOS.

Y se configura el archivo de la siguiente manera como se presenta en la figura.

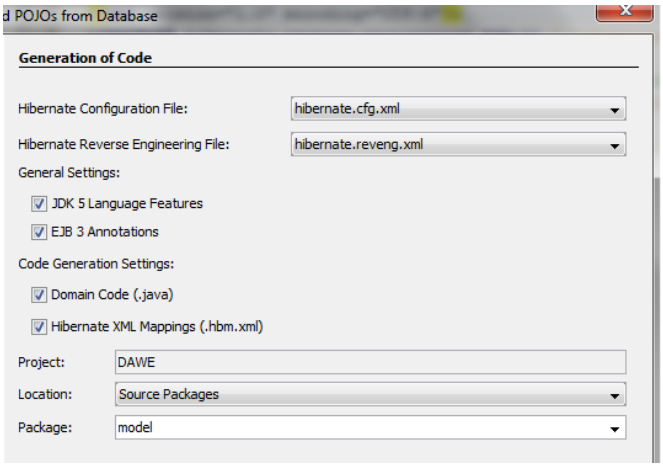


Figura III.8. Configuración POJOS.

Finalmente de la configuración generada se tiene el siguiente mapeo de clases de la base de datos en la siguiente gráfica.

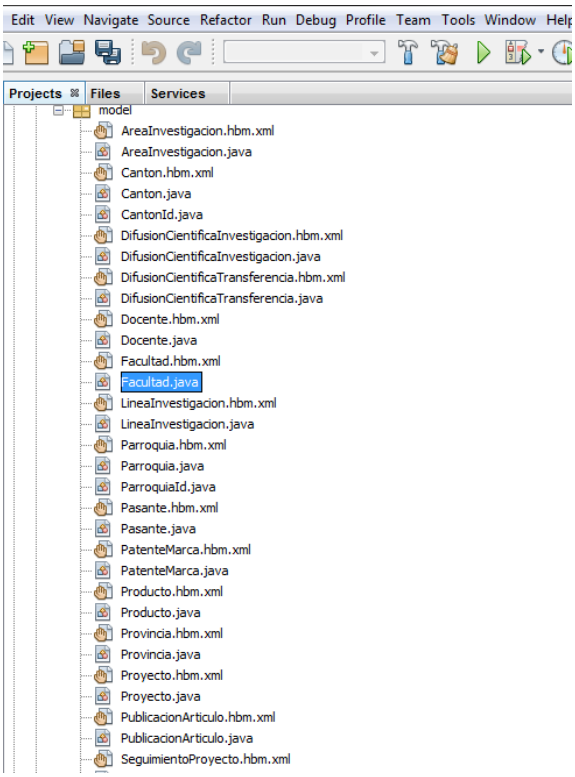


Figura III.9. Código Generado en el paquete model.

3.3.2 CONSTRUCCIÓN DE PAQUETES

Una vez mapeado nuestras entidades de la base de datos, se debe crear los distintos paquetes de la lógica de negocio así como también los paquetes que contendrán las interfaces a implementar.

Los paquetes que se debe crear son los siguientes:

- Paquete de modelo de datos (mapeo de datos).
- Paquete de interfaces DAO (Data Access Object).
- Paquete de Implementación DAO.
- Paquete de Interfaces de Servicio.
- Paquete de Implementación de Servicios.

El paquete de modelo de datos hacer referencia al mapeo de las entidades de la base de datos, como se lo realiza en la fase anterior.

3.3.3 GENERACIÓN DE LA CAPA DE INTERFACES

Para la utilización de esta metodología es importante que se defina en el desarrollo de **interfaces** ya que el desarrollo del proyecto está enfocado a la tecnología Spring MVC la cual es robusta trabajando con interfaces.

Para el desarrollo del paquete de interfaces **DAO**se debe definir solamente los nombres de los métodos que se implementa, como son el manejo de **insertar, actualizar, eliminar y listar** como se ejemplo se muestra en la figura siguiente:

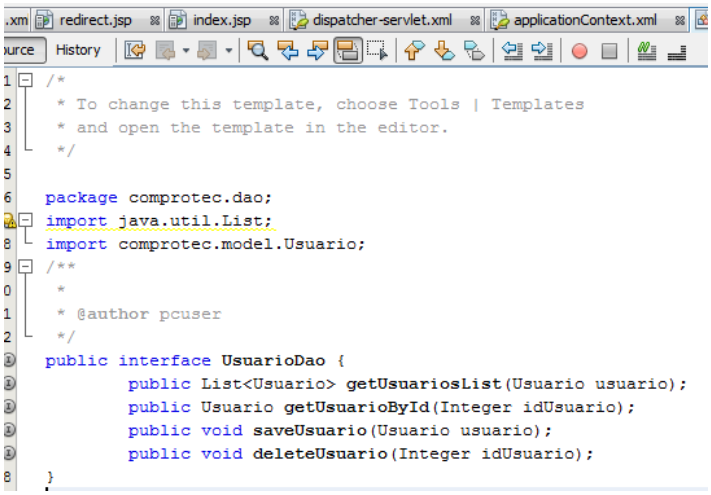


Figura III.10. Definición de métodos CRUD DAO.

El paquete de interfaces de servicio se debe trabajar con la anotación @Service la cual indica que las clases marcadas con esta anotación está en una capa de servicios o de lógica de negocios tal como se indica en la figura siguiente:

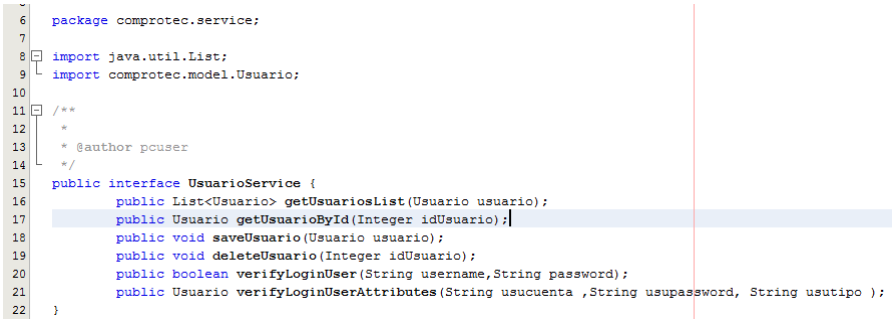


Figura III.11. Definición de métodos CRUD SERVICE.

3.3.4 GENERACIÓN DE LA CAPA DE PERSISTENCIA DE DATOS (DAO)

El proceso de implementación **DAO**se debe codificar los métodos definidos en la fase anterior trabajando con la anotación @Repository que indica que las clases marcadas con esta anotación están relacionada de alguna forma con una capa de persistencia de datos tal como se muestra en la figura siguiente:

```
21  @Repository
22  public class UsuarioDaoImpl implements UsuarioDao {
23
24      private HibernateTemplate hibernateTemplate;
25      private JdbcTemplate jdbcTemplate;
26
27      @Autowired
28      public UsuarioDaoImpl(SessionFactory sessionFactory) {
29          this.hibernateTemplate = new HibernateTemplate(sessionFactory);
30          this.jdbcTemplate = new JdbcTemplate(SessionFactoryUtils.getDataSource(sessionFactory));
31      }
32
33      public List<Usuario> getUsuariosList(Usuario usuario) {
34          StringBuilder query = new StringBuilder("from Usuario ");
35
36          if (usuario != null && usuario.getUsuFiltro() != null
37                  && usuario.getUsuFiltro().length() > 0) {
38
39              query.append("where upper(usuNombre) like '%").append(usuario.getUsuFiltro().
40                      toUpperCase()).append("%' ").append("or upper(usuEmail) like'%").append(usuario.getUsuFiltro
41                      toUpperCase()).append("%' ").append("or upper(usuCedula) like'%").append(usuario.getUsuFiltr
42                      toUpperCase()).append("%' ").append("or upper(usuCuenta) like'%").append(usuario.getUsuFiltr
43                      toUpperCase()).append("%' ").append("or upper(usuApellido) like'%").append(usuario.getUsuFil
44                      toUpperCase()).append("%' ").append("or upper(usuTipo) like'%").append(usuario.getUsuFiltro(
45                      toUpperCase()).append("%' ");
46          }
47          List<Usuario> list = (List<Usuario>) hibernateTemplate.find(query.append("order by usuApellido").toStrin
48          return list;
49      }
```

Figura III.12.Capa de persistencia.

```
54      public List<Usuario> getUsuario(Usuario usuario) {
55          StringBuilder query = new StringBuilder("from Usuario ");
56          if (usuario != null && usuario.getUsuNombre() != null
57                  && usuario.getUsuNombre().length() > 0) {
58              query.append("where upper(usuCuenta) like '").append(usuario.getUsuCuenta().
59                      toUpperCase()).append("and upper(usuPassword) like'").append(usuario.getUsuPassword().toUppe
60          }
61
62          List<Usuario> list = (List<Usuario>) hibernateTemplate.find(query.toString());
63          return list;
64      }
65
67      public Usuario getUsuarioById(Integer idUsuario) {
68          Usuario usuario = (Usuario) hibernateTemplate.get(Usuario.class, idUsuario);
69          return usuario;
70      }
72      public void saveUsuario(Usuario usuario) {
73          hibernateTemplate.saveOrUpdate(usuario);
74      }
76      public void deleteUsuario(Integer idUsuario) {
77          Usuario usuario = getUsuarioById(idUsuario);
78          hibernateTemplate.delete(usuario);
79      }
  }
```

Figura III.13. Capa de persistencia.

3.3.5 GENERACIÓN DE LA CAPA DE LÓGICA DE NEGOCIO

A continuación se debe codificar las interfaces desarrolladas en la fase anterior tal como se muestra en la figura siguiente:

```
5   package comprotec.service.implementation;
6
7   /**
8    *
9    * @author pcuser
10   */
11  import comprotec.dao.UsuarioDao;
12  import comprotec.model.Usuario;
13  import comprotec.service.UsuarioService;
14  import java.util.List;
15  import org.springframework.beans.factory.annotation.Autowired;
16  import org.springframework.stereotype.Service;
17
18  @Service
19  public class UsuarioServiceImpl implements UsuarioService {
20
21      @Autowired
22      private UsuarioDao usuarioDao;
23
25      public void deleteUsuario(Integer idUsuario) {
26          getUsuarioDao().deleteUsuario(idUsuario);
27      }
29      public Usuario getUsuarioById(Integer idUsuario) {
30          return getUsuarioDao().getUsuarioById(idUsuario);
31      }
33      public List<Usuario> getUsuariosList(Usuario usuario) {
34          return getUsuarioDao().getUsuariosList(usuario);
    }
```

Figura III.14. Código de Generación de la Capa de Negocio.

3.3.6 ETAPA DE DOCUMENTACIÓN.

Es importante una vez desarrolladas las fases previas documentarlas haciendo referencia a la importancia de la implementación de cada paquete y a su codificación, esta etapa de documentación servirá para tener ya un esquema de cómo será el tratamiento de los datos por capas y para estandarizar la forma en la cual se nombra a los paquetes así como las clases codificadas en cada una de ellas.

3.4 FASE DE INTEGRACIÓN DE FRAMEWORK SPRING MVC

3.4.1 CONFIGURACIÓN DE CONTENEDOR DE OBJETOS APPLICATION-CONTEXT.XML

Esta fase es la más importante de todo el desarrollo del sistema ya que aquí se define inyección de dependencia proceso en el cual los objetos definen sus dependencias (o sea, los otros objetos con los que trabajan) solo a través de los argumentos de su constructor, argumentos a un método de factory, o métodos setter que son invocados después de que el objeto se ha construido. En este caso en vez de ser el mismo objeto quien se encargue de instanciar, o localizar, las dependencias con las que trabaja (usando directamente su constructor o un localizador de servicios), es el contenedor el que inyecta estas dependencias cuando crea al bean. [3][14]

El archivo contiene únicamente el elemento raíz **"<beans>"** dentro del cual declararemos cada uno de nuestros **beans** usando un elemento **"<bean>"** para cada uno.

En el archivo que debamos configurar procederemos a realizar la inyección de dependencia con los objetos que deseemos trabajar, tal como se muestra en la figura siguiente:

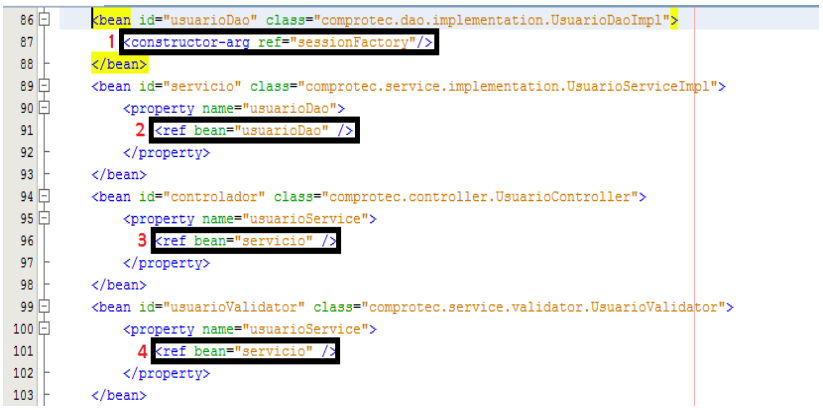


Figura III.15. Configuración del APPLICATION-CONTEXT.XML

- Se hace referencia a la inyección de dependencia por constructor, se inyecta la dependencia de la conexión de la base de datos.
- Inyección de dependencia por setter, como se puede observar se inyecta el **bean** creado de la implementación de servicio.
- Inyección de dependencia por setter, se inyecta el **bean** controlador.
- Inyección de dependencia por setter, se inyecta el **bean** validador.

3.4.2 CONFIGURACIÓN DEL DISPATCHER-SERVLET.

La actividad a desarrollar es configurar el archivo **dispatcher-servlet**, en el cual definiremos la referencia url para el manejo de nuestras respectivas vistas o paginas htm, como se muestra en la figura:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:p="http://www.springframework.org/schema/p"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/bea
       http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
       http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
       http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-contex

        <bean class="org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping"/>
        <context:component-scan base-package="comprotec.controller"/>
        <bean id="messageSource" class="org.springframework.context.support.ReloadableResourceBundleMessageSourc
               <property name="basename" value="/WEB-INF/config/messages/validation" />
        </bean>
        <bean id="viewResolver"
          class="org.springframework.web.servlet.view.InternalResourceViewResolver"
          p:prefix="/WEB-INF/jsp/"
          p:suffix=".jsp" />
        <!-- Returns messages based on a resource bundle -->

</beans>
```

Figura III.16. Configuración del Dispatcher-Servlet

3.4.3 GENERACIÓN DE LA CAPA DE CONTROLADORES.

En esta fase se debe codificar los controladores de cada uno de nuestras vistas a desarrollar, trabajaremos con la anotación @Controller, indica que las clases marcadas con esta anotación son el controlador de una aplicación web.

Utilizaremos la anotación @Autowired permite marcar un constructor, campo, setter, o método de configuración para ser auto-cableado por el proceso de inyección de dependencia de Spring.

Y finalmente la anotación @RequestMapping, la cual hace referencia al comportamiento url de la petición solicitada por la vista, la codificación se muestra en la figura siguiente:

```
Source  History
26  @Controller
27  public class TesisGradoController {
28
29      @Autowired
30      private TesisGradoService tesisGradoService;
31      @Autowired
32      private TesisGradoValidator tesisGradoValidator;
33
34      public void setTesisGradoService(TesisGradoService tesisGradoService) {
35          this.tesisGradoService = tesisGradoService;
36      }
37
38      @RequestMapping("tesisgrado/tesisGradoList.htm")
39      public void patenteMarcaList(Model model,
40              @ModelAttribute("tesisGrado") TesisGrado tesisGrado) {
41          List<TesisGrado> tesisGrados = tesisGradoService.getTesisGradoList(tesisGrado);
42          model.addAttribute("tesisGrados", tesisGrados);
43          model.addAttribute("tesisGrado", tesisGrado);
44      }
45
46      @RequestMapping(value = "tesisgrado/tesisGradoView.htm", method = RequestMethod.GET)
47      public @ModelAttribute("tesisGrado")
48      TesisGrado tesisGradoView(@RequestParam(value = "tgCodigo", required = false) Integer tgCodigo) {
49          if (tgCodigo != null) {
50              TesisGrado tesisGrado = tesisGradoService.getTesisGradoById(tgCodigo);
51              return tesisGrado;
52          }
53          return new TesisGrado();
54      }
```

Figura III.17. Capa del controladores

3.4.4 ETAPA DE DOCUMENTACIÓN.

En esta fase de documentación se debe especificar, los diferentes tipos de inyección de dependencia a utilizar, ya que es de vital importancia indicar como está compuesto nuestro contenedor de objetos a través de los **beans**.

Se debe documentar los controladores creados, haciendo énfasis en las diversas peticiones urls a través de la anotación @RequestMapping, para poder tener especificado las acciones a realizar por cada petición que realice el cliente a través de las urls.

3.5 FASE DE INTEGRACIÓN DE CAPA DE PRESENTACIÓN

3.5.1 GENERACIÓN DE LA CAPA DE VISTAS.

Se debe desarrollar las interfaces o páginas de iteración con el usuario final, se debe trabajar con las librerías propias de Spring MVC para relacionar cada atributo con las páginas a desarrollar, tal como se muestra en la figura siguiente:

Figura III.18.Etiquetas del framework Spring MVC

3.5.2 GENERACIÓN DE LA CAPA DE VALIDACIÓN.

Una vez desarrolladas las vistas, se debe codificar la capa de validaciones, la cual servirá para controlar las restricciones de la lógica de negocio así también como sus respectivas validaciones, trabajando con la anotación @Component se debe desarrollar una clase de validación por cada entidad de ingreso que se realizar en nuestra aplicación como se observa en la figura siguiente:[12][15]

```java
package comprotec.service.validator;

import comprotec.model.Usuario;
import comprotec.service.UsuarioService;
import org.hibernate.tool.hbm2x.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.springframework.validation.Errors;
import org.springframework.validation.Validator;

@Component
public class UsuarioValidator implements Validator {

    @Autowired
    private UsuarioService usuarioService;

    @SuppressWarnings("rawtypes")
    public boolean supports(Class clazz) {
        return Usuario.class.isAssignableFrom(clazz);
    }

    public void validate(Object object, Errors errors) {

        Usuario usuario = (Usuario) object;
        validateNombre(usuario, errors);
        validateApellido(usuario, errors);
        validateCedula(usuario, errors);
        validatePassword(usuario, errors);
        validateTipo(usuario, errors);
        validateEmail(usuario, errors);
```

Figura III.19. Generación de la capa de validación.

3.5.3 INTEGRACIÓN DE SPRING SECURITY.

Una vez desarrollada las etapas anteriores, se debe tomar en consideración implementar la seguridad para nuestra aplicación, en este caso lo vamos a realizar a través de la librería **spring-security,** se trata de un módulo no intrusivo que permite de forma sencilla añadir funcionalidades de control de acceso y autenticación a nuestra aplicación web. Por simplificarlo al máximo se trata de un conjunto de filtros que se ejecutan en cadena antes de acceder a los recursos de nuestra aplicación con el objetivo de dar una buena seguridad.[12]

Se debe agregar la librería **spring-security-core** para poder utilizar las etiquetas de spring security en nuestro archivo de configuración como se muestra en la figura:[4]

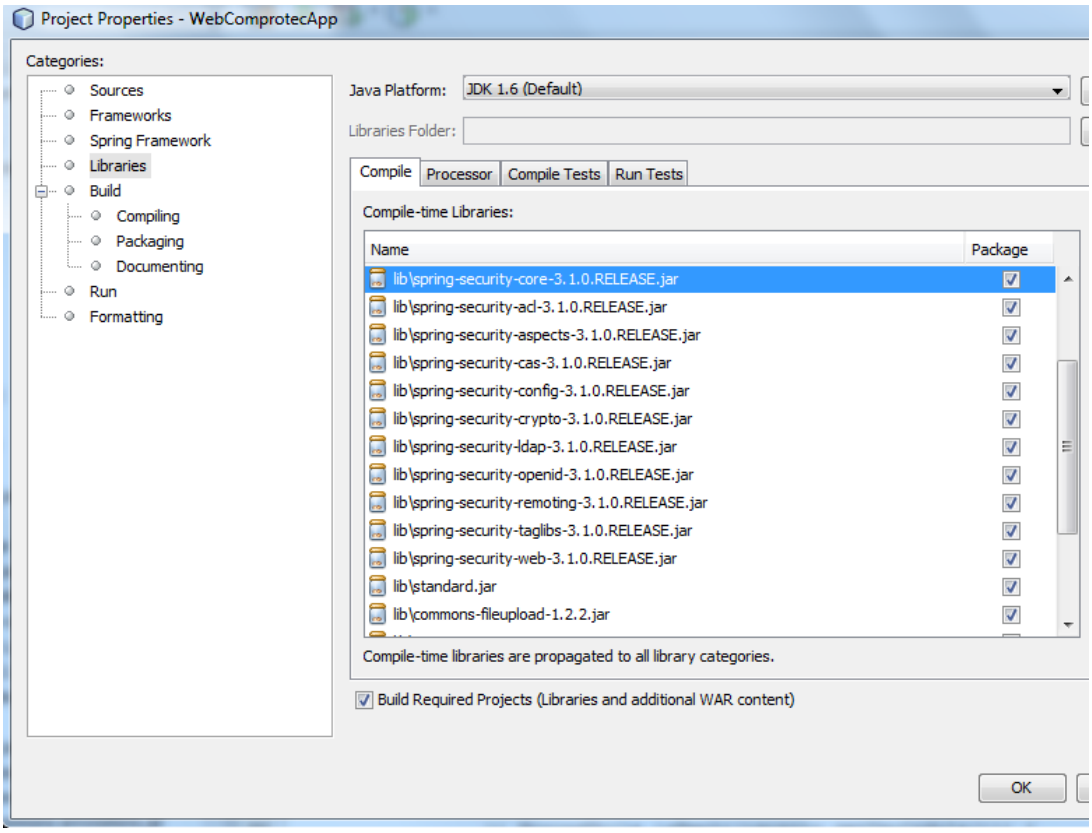


Figura III.20. Integración de Spring Security.

Ahora se debe crear un archivo de configuración de spring, en donde se procede a configurar los distintos accesos y permisos de la aplicación web.

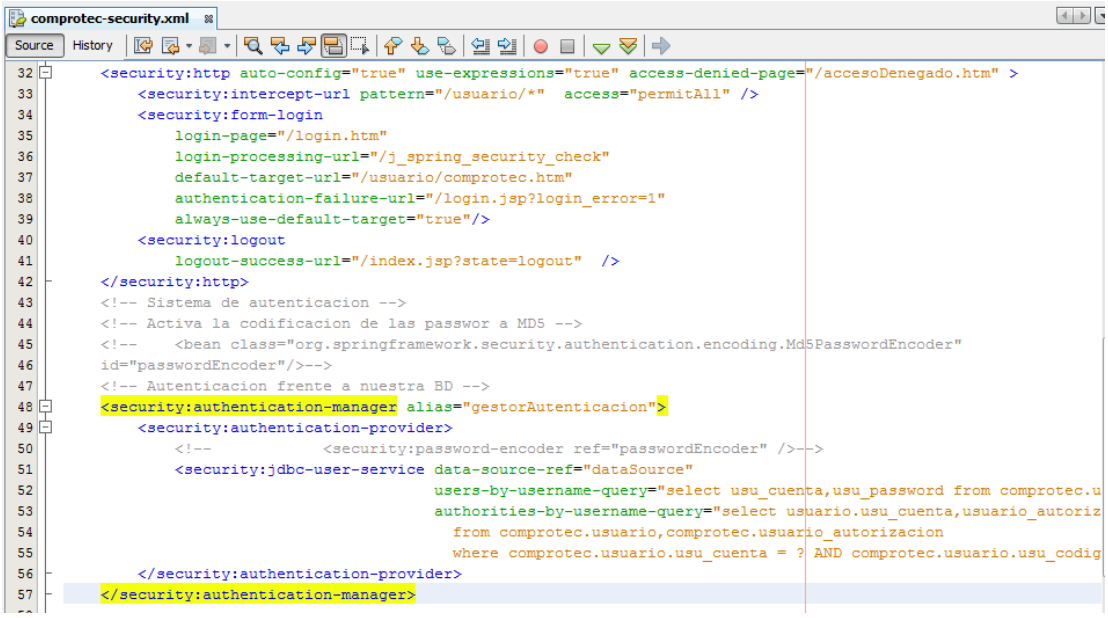


Figura III.21. Configuración de roles de acceso.

3.5.3.1 ETAPA DE DOCUMENTACIÓN

En esta fase de la metodología DAWE se debe registrar todas las páginas desarrolladas, las validaciones realizadas así como también los accesos permitidos a los usuarios, ya que esto ayudará a especificar a los usuarios finales el trabajo que cada uno podrá realizar.

Es importante que una vez terminado esta fase de documentación, se realice una exposición de lo documentado a los usuarios que utilizarán la aplicación, esto de se lo realiza en base a planes de mejoramiento.[15]

3.6 FASE DE PRUEBAS

3.6.1 PRUEBAS DE IMPLEMENTACIÓN CON USUARIOS FINALES.

Se debe efectuar pruebas de ejecución con los usuarios finales, lo cual ayudará para comprobar la fortaleza del sistema y para detectar errores que se deberá corregir.

3.6.2 CORRECCIÓN

La etapa de corrección se debe tomar en consideración las especificaciones realizadas por los usuarios, efectuando las debidas correcciones del sistema y pasar a efectuar una presentación final para comprobar el arreglo de los posibles errores encontrados.

3.6.3 FASE DE INSTALACIÓN

3.6.3.1 IMPLEMENTACIÓN DEL SERVIDOR DE APLICACIONES.

Se realizará la instalación del servidor de aplicaciones, el servidor será GLASSFISH, este servidor proporcionará el alojamiento para que nuestra aplicación pueda ejecutarse con normalidad.

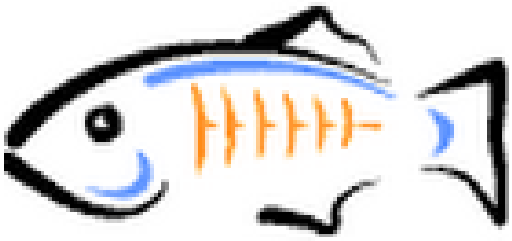


Figura III.22. Servidor Glassfish

CAPÍTULO IV SISTEMA DE GESTIÓN DE LA COMPROTEC

4.1 INTRODUCCIÓN

El departamento COMPROTEC de la ESPOCH está viviendo una nueva etapa dentro de su gestión, donde se requiere mejorar las actividades laborales, tomando en cuenta que actualmente la institución no tiene un sistema de automatización. Un nuevo sistema permitirá ser eficientes en el sentido de atención al cliente de forma rápida en la tramitación de la documentación.

Es importante señalar que la gestión de la administración actual está dirigida no solamente a mejorar los servicios que presta, sino a promover el desarrollo productivo de la COMPROTEC.

Este Capítulo permite definir los datos de la organización, así como la especificación del sistema tanto actual como del propuesto, los requerimientos del sistema, la planificación del proyecto, el estudio de la factibilidad, el análisis orientado a objetos y el diseño del sistema propuesto.

Para el desarrollo del software de gestión de la COMPROTEC, se aplicó la metodología propuesta denominada DAWE Web, la cual consta de las siguientes fases y actividades.

4.2 FASE DE PLANIFICACIÓN

4.2.1 Estudio de la institución

4.2.1.1 Misión

Promover la investigación científica y tecnológica a través de la realización de proyectos y contratos de prestación de servicios, consultaría y asesoría.

4.2.1.2 Estructura orgánica Funcional de la Comprotec

La estructura orgánica de la Comisión de Proyectos y Transferencia de Tecnología se encuentra integrada por:

4.2.1.2.1 NIVEL DIRECTIVO:

El Vicerrector de Investigación y Desarrollo, quien preside.

El Director de la Comisión de Proyectos y Transferencia Tecnológica.

Los Vicedecanos.

4.2.1.2.2 NIVEL OPERATIVO:

Dos investigadores, evaluadores de proyectos.

4.2.1.2.3 NIVEL DE APOYO:

El personal de apoyo estará integrado por una secretaria y un conserje.

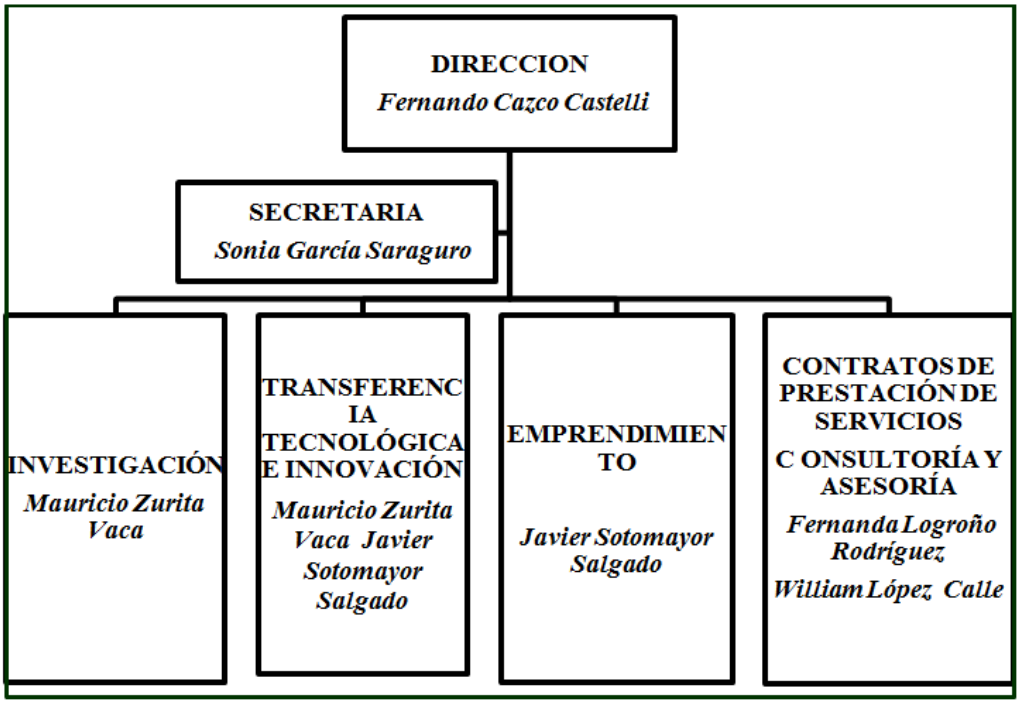


Figura IV.1.Estructura Orgánica Comprotec.

4.2.2 Planificación

Esta es la primera etapa del proyecto en donde interviene la interacción del cliente y los desarrolladores para descubrir los requerimientos del sistema. Además de establecer las historias de usuarios, número de iteraciones, velocidad del proyecto.

La planificación del proyecto está distribuida de la siguiente manera:

- Investigación de las herramientas de desarrollo.
- Investigación de la institución, aquí se obtuvo toda la información referente a la Institución.
- Estructuración del sitio.
- Programación de los diferentes formularios y páginas que interactúan en el sitio Web.
- Pruebas.

4.2.2.1 ANÁLISIS DE REQUERIMIENTOS

El SRS (Especificación de requerimientos Software) es la base para realizar la validación del sistema, además de que ayuda a entender a los desarrolladores lo que quiere el cliente.

El SRS es aplicable a cualquier producto de software es por ello que se aplicara al desarrollo del proyecto Sistema de Gestión de la Comprotec

En esta actividad se desarrolló la Especificación de Requerimientos de Software (SRS) para determinar los requerimientos de usuario que debe cumplir el software creado.

La Especificación de Requerimientos de Software SRS es una de las piezas más importantes en un proyecto de Desarrollo de Software. Es necesario entonces establecer las condiciones o necesidades del usuario para resolver un problema, de esta manera satisfacer un contrato a través de un documento formal.

Los requerimientos que se establecen en el presente entregable son funcionales los cuales definen las funciones que el sistema será capaz de realizar. Además describen las transformaciones que el sistema realiza sobre las entradas para producir salidas y los no funcionales que tiene que ver con características que de una u otra forma pueden limitar el sistema.

4.2.2.2 OBJETIVOS DEL SRS

Los objetivos del SRS son los siguientes:

- Determinar los requerimientos funcionales y no funcionales para el sistema SGC.
- Describir de la forma más clara posible cada uno de los requerimientos del usuario, actividades, métodos, procesos, etc.
- Realizar algoritmos de seguimiento para el buen uso y manejo del sistema por parte de los usuarios, propuestas de interfaces, etc.
- Presentar un prototipo de las interfaces de usuario que contendrá el Sistema.

4.2.2.3 Ámbito

El Sistema SGCOM servirá de ayuda para las pequeñas y medianas empresas que realicen actividades de gestión de proyectos, el mismo que al momento no tiene ningún proceso automatizado, con lo cual se pretende mejorar la gestión de proyectos en la COMPROTEC, Así se le permitirá llevar un mejor control de las actividades que se hacen por parte de sus investigadores, Por tal razón este sistema aporta a la efectividad y rapidez para un mejor control de proyectos.

4.2.2.3.1 Objetivos

Implementar un sistema informático automatizado de acuerdo a las necesidades con el propósito de mejorar gestión de proyectos desarrollados en la comprotec y la productividad de la misma.

4.2.2.3.2 Beneficios

4.2.2.3.2.1 Beneficios Tangibles.

Reducir el error en el registro de tiempos de uso al 90%.

Tiempo de atención más corto 75%.

Ahorro de dinero por utilizar menos insumos de oficina y personal 90%.

Reduce muy notablemente los errores, ya que el programa valida todos los datos ingresados 100%.

Facilita el trabajo laboral de los trabajadores de la institución 80%.

4.2.2.3.2.2 Beneficios Intangibles.

Mejor atención al cliente

Calidad del servicio que atrae a más clientes

Mejor ambiente de trabajo para el personal.

El sistema mejorará la imagen de la empresa.

Permitirá al nivel estratégico tomar decisiones.La información se encuentra estructurada y con fácil acceso.

4.2.2.4 Definiciones, Acrónimos y Abreviaturas

Definiciones

Empleado / Cajero: Persona que se dispone a realizar una venta en la empresa.

Administradores: Persona encargada de la administración del sistema

Acrónimos

SGCOM Sistema de Gestión de la Comprotec.

CASE Ingeniería de Software Asistida por Computador.

SRS Software Requeriments Specifications

IEEE Instituto de Ingeniería Eléctrica y Electrónica

Abreviaturas

HW Hardware

SW Software

4.2.2.5 Referencias

La documentación de especificación de requerimiento se lo realizo aplicando los siguientes puntos especificados aquí:

1. Software Requirements Specifications basado en el estandar 830 IEEE.
2. Documentos de estatutos, acuerdos, de contratación de personal.

4.2.2.6 Visión general

El documento que se presenta está estructurado en tres partes: la introducción, descripción general y los requerimientos específicos.

El documento de SRS describe los requerimientos del software a construir. Se incluyen los requerimientos funcionales y no funcionales, así como también diseño de las interfaces de SW.

Este documento pretende lograr el acuerdo entre usuarios, administradores y director de la COMPROTEC sobre los requisitos que el sistema a desarrollar presentará. Además de explicar la verdadera funcionalidad de las características del software para su funcionamiento posterior.

El SGCOM permitirá acelerar la gestión de trámites para impulso de proyectos desarrollados en la COMPROTEC.

4.2.2.7 DESCRIPCIÓN GENERAL

Se pretende automatizar el departamento de gestión de proyectos de la COMPROTEC ubicado en el Cantón Riobamba, Panamericana Sur km 1 1/2, Teléfono: +593 (03) 2 998-200 | Telefax: +593 (03) 2 317-001 Código Postal: EC060155.

Es una aplicación web que tiene como objetivo la difusión de datos de los proyectos en ejecución, así como también la publicación de artículos y de las últimas noticias de los avances de la gestión de proyectos.

Sistema de Gestión de la Comprotec (SGCOM) permitirá automatizar la información que se maneja en la misma, proporcionar una herramienta de vital importancia en la toma de decisiones de este departamento.

4.2.2.7.1 Perspectiva del producto

El Sistema **SGCOM** cubre los parámetros de procedimientos adjuntando elementos de ingresos, salidas, reportes de información procesada, asignación, modificación, registros, exceptuando la eliminación de datos por motivo de auditorías ya que la información será histórica.

El Software es construido bajo la Plataforma JAVA NetBeans 7.1 con motor de bases de datos PostgreSQL 9.1 lo que garantiza la confiabilidad del sistema, está desarrollado y probado bajo lineamientos propios de la COMPROTEC.

4.2.2.7.2 Funciones del producto

A continuación se detalla las funciones del producto la cual se encuentra organizada en módulos.

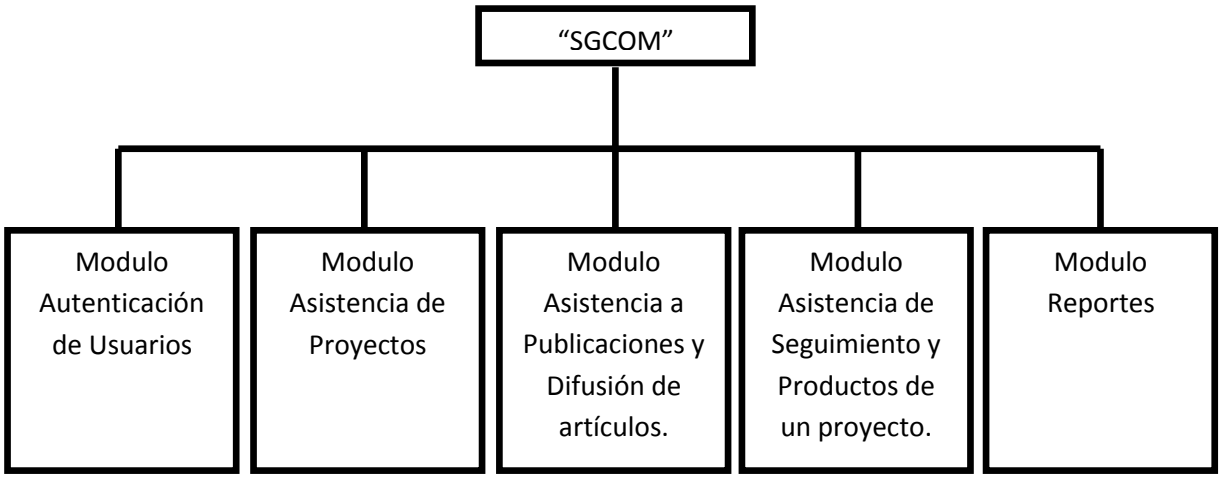


Figura IV.2. Módulos de la aplicación SGCOM.

4.2.2.7.2.1 **Módulo autenticación de usuarios**

Los usuarios, al realizar determinado proceso en el SGCOM deben contar con ciertos permisos, de modo que ningún individuo no involucrado con el mismo pueda realizar estos procesos.Para esto el Sistema contará con un servicio de autenticación y autorización de usuarios.

4.2.2.7.2.2 **Modulo Asistencia de Proyectos**

En este módulo se gestiona todo cuanto concierne a un proyecto donde se registra un proyecto con sus respectivos investigadores, docente, pasante, difusión entre otros indicadores.

4.2.2.7.2.3 **Módulo asistencia a publicación y difusión de artículos**

En este caso el administrador registra la o lasdifusiones científicas de investigación que genera un proyecto determinado, así como también las difusiones científicas de transferencia y publicación de un artículo generado internamente dentro del departamento.

4.2.2.7.2.4 Módulo reportes

Aquí se facilitará la publicación de proyectos con su respectivo estado y descripción, publicaciones de artículos, entre otros contenidos que son de acceso público.

4.2.2.7.3 Características del usuario

Aquí incluiremos las personas que interactúan con el sistema durante la fase de operación y mantenimiento.

Tabla IV.I. Característica del usuario.

FUNCIÓN	NIVEL EDUCACIONAL	CONOCIMIENTOS TÉCNICOS
Desarrollador	Superior	Ing. Sistemas Desarrollador de Sistemas Analista de Sistemas
Personal	Superior	Ofimática, Windows Xp mínimo, Herramienta **SGCOM**.
Administrador	Superior	Visual Studio 2008, Sql Server 2005 0 2008, Redes LAN

4.2.2.7.4 Limitaciones generales

Necesita un sistema operativo (Windows XP, Vista, Seven, Centos 6.3).

Sistema programado para procesadores multi-núcleo

El sistema no podrá funcionar al máximo de su capacidad en equipos que tengan menos de 256 MB de memoria RAM.

El sistema no funciona en máquinas que no tengan acceso a internet.

La aplicación va a ser desarrollada utilizando el Motor de Base de Datos de PostgreSQL 9.1, utilizando el lenguaje de programación de Java Spring Framework, y utilizando el servidor web denominado GlassFish.

Para el correcto funcionamiento del sistema se debe implantar en una arquitectura cliente-servidor en su totalidad.

4.2.2.7.5 Limitaciones De Software

La ESPOCH cuenta con el respectivo Software para la implementación e implantación, dentro de los requerimientos planteamos los siguientes:

4.2.2.7.5.1 Sistema Operativo

Servidor: Linux o Windows

Cliente: Microsoft Windows y Linux

4.2.2.7.5.2 Desarrollador

Programación Java NetBeans 7.1

Spring MVC.

Servidor web GlassFish 3.0.

4.2.2.7.5.3 Sistema de administración de base de datos

DBMS PostgreSQL 9.1

4.2.2.7.5.4 Limitaciones de hardware

El entorno hardware de la ESPOCH no posee limitantes debido a que en su infraestructura posee todos los equipos requeridos para la implementación e implantación del sistema a desarrollarse.

4.2.2.7.6 Supuestos y dependencias

Si el DBMS sufre algún desperfecto este debería ser cambiado por otro de idénticas características.

Los cambios en los requerimientos en algún instante de la construcción del software producirán retraso en el desarrollo e incluso puede causar una reprogramación total del producto de software.

El cambio de algún miembro del equipo de trabajo de desarrollo podría afectar en gran manera el desarrollo del producto.

4.2.2.8 DEFINICIÓN DEL DIAGRAMA DE PROCESO DE LA COMPROTEC

Aquí se realiza un estudio de procesos el objetivo de:

- Mejora continua de las actividades desarrolladas,
- Reducir la variabilidad innecesaria,
- Eliminar las ineficiencias asociadas a la repetitividad de las actividades,
- Optimizar el empleo de los recursos.

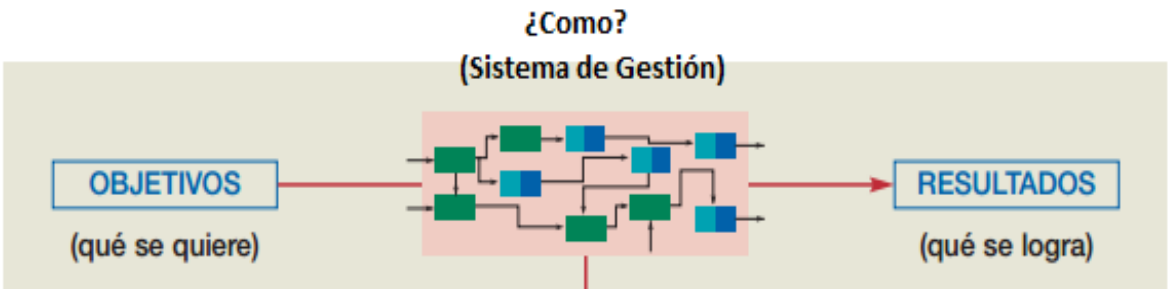


Figura IV.3. Gestión de Procesos.

En la Figura IV.3 se presenta el análisis de las actividades de la Comprotec, lo que no permite analiza las siguientes preguntas, que se quiere, como, sistema de gestión y que se desea lograr.

4.2.2.9 MODELO DE GESTIÓN

4.2.2.10Procesos Planificación Sectorial

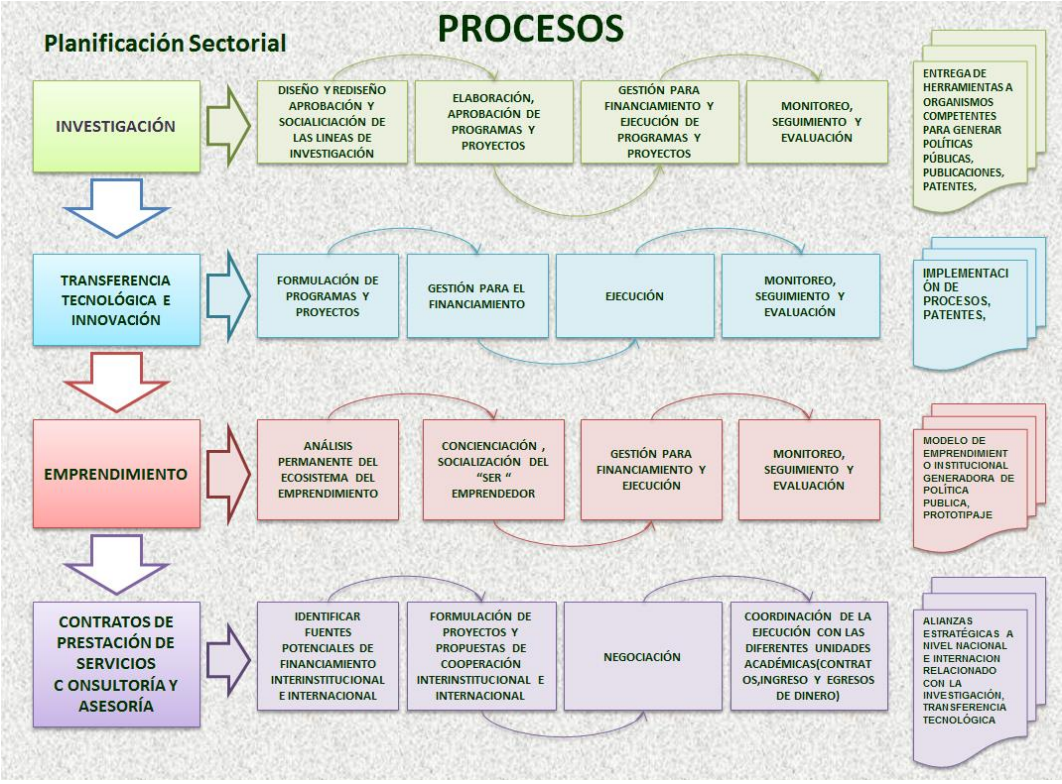


Figura IV.4. Planificación Sectorial.

En la Figura IV.4 se puede observar los diferentes procesos de planificación sectorial, como son investigación, transferencia tecnología e innovación, emprendimiento, contratos de prestación de servicios consultoría y asesoría, en las se basa para el análisis de procesos de la Comprotec.

A través de un análisis lógico y global del negocio se identifica el proceso relacionado con los usuarios críticos para el éxito del proceso de registro de un proyecto para la COMPROTEC.

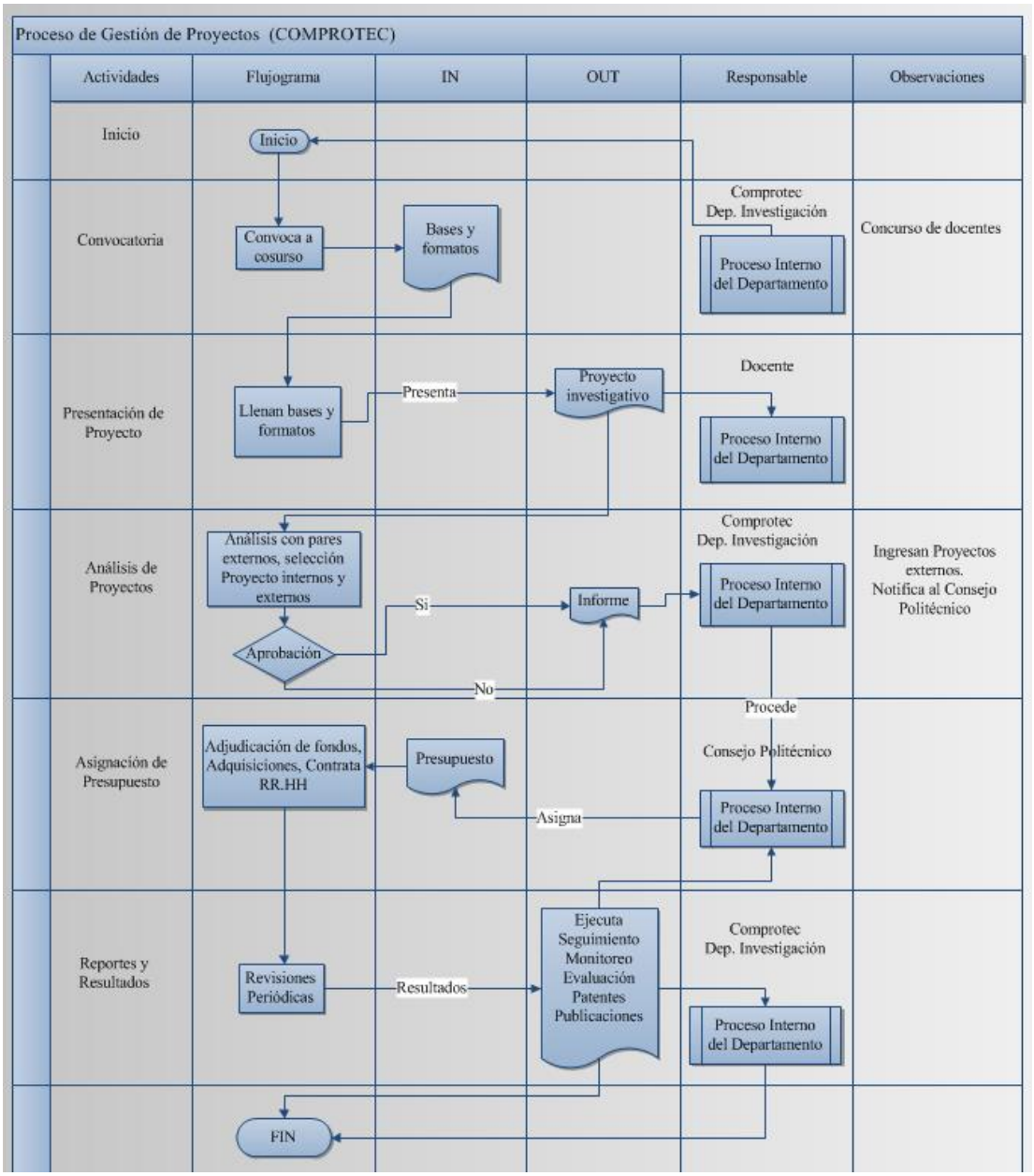


Figura IV.5. Proceso de usuario de la Gestión de Proyectos

En laFigura IV.5 Se definen las actividades que conforman el proceso así como las interconexiones de los flujos de entrada y de salida de las actividades.

4.2.2.11REQUERIMIENTOS FUNCIONALES ESPECÍFICOS

En la tabla siguiente se aprecia una descripción de los requisitos que el sistema deberá cumplir, de acuerdo al desarrollo del sistema y con los requerimientos funcionales o principales que enunciamos en las siguientes páginas donde se especifica los casos de uso extendido es decir los de mayor importancia, que cumplen las principales funciones del sistema lo que conlleva a que sea eficiente para su ejecución en el campo laboral.

1 Instalar Herramientas de Software.

2 Como Director quiero el sistema permita el control de autenticación de los miembros de trabajo de la Comprotec.

3 Como Director quiero el sistema permita el control de gestión de las entidades tales como, el ingreso, modificación, actualización de los datos de cada Proyecto, Docente, Pasante, Seguimiento de proyecto, Productos de proyecto, Patente, Difusión científica de investigación, Difusión científica de transferencia, Publicación de Artículos, Tesis de grado.

4 Como Director quiero el sistema permita realizar búsquedas de las entidades por código, nombre etc.

5 Como Director quiero que permita realizar el seguimiento de los diferentes proyectos existentes.

6 Como Director quiero que permita registrar los productos que genera un proyecto con su respectiva patente en caso que lo tenga.

7 Como Director quiero que el sistema permita registrar docentes con sus respetivas tesis de estudiantes en dirección.

8 Como Director quiero que el sistema permita la gestión de, Difusión científica de investigación, Difusión científica de transferencia, Publicación de Artículos.

9 Como Director quiero que la información pública sea visualizada a través de la web para que nuestros clientes puedan acceder a nuestros servicios desde cualquier parte.

10 Como Director quiero que la información publicada en las cuentas de Twitter, y Facebook se integre para su visualización en el portal web.

11 Como Director quiero el sistema permita generar reportes de los proyectos de investigación en ejecución para que sean visualizados en la web.

4.2.2.11.1 Requerimiento funcional 1

Instalar Herramientas de Software.

1.1 Introducción

Permite a los usuarios ingresar a la pantalla de inicio del portal SGCOM. Para la construcción del mismo se utilizará las herramientas Joomla y Spring Web MVC 3.0 y PostgreSQL, para la gestión de contenido del portal para cumplir con el requerimiento. Este requisito se instala las herramientas como el servidor Glassfish, Apache, Jdk, Tomcat, NetBeans 7.1. Las salidas van dirigidas a los usuarios sin ninguna restricción. [8]

1.2 Entradas

Tabla IV.II. Entradas requerimiento funcional 1.

Fuente:	Herramientas de Software
Cantidad:	Relacionado con las necesidades de la empresa.
Frecuencia:	Depende de los requerimientos del usuario.
Rango de entradas válidas:	Disponible todo el tiempo

1.3 Proceso

El proceso de instalación de herramientas de software falle, tomando en cuenta que en la granja de servidores de la ESPOCH contiene ya preparado la mayor parte de software para implementar la aplicación.

Desarrolladores revisan el software requerido en caso de faltar alguno.

Se informa del faltante al administrador de la granja de servidores para proceder a la instalación del software faltante.

1.4 Salidas

Finalmente con todo el software requerido instalado se procede a la implantación de la aplicación.

1.5 Interfaz de usuario

N/A

4.2.2.11.2 Requerimiento funcional 2

Como Director quiero el sistema permita el control de autenticación de los miembros de trabajo de la Comprotec.

1.1 Introducción

La aplicación permitirá que un empleado acceda al sistema, dado sus datos personales de acceso, ingresar su código asignado para poder autentificarse en el sistema.

1.2 Entradas

Tabla IV.III. Entradas requerimiento funcional 2.

Fuente:	Aplicación web.
Cantidad:	Relacionado con las necesidades del director.
Frecuencia:	Depende de los requerimientos del director.
Rango de entradas válidas:	Disponible todo el tiempo

- Rango (R)
- Cuenta(R)
- Password(R)

1.3 Proceso

- El empleado accede a la aplicación SGCOM.
- El sistema pide verifica el cargo, cuenta, password y permite el acceso, mostrando el ambiente de bienvenida para su gestión.

1.4 Salidas

Introducción del código inválido mostrara nuevamente el screen de Autenticación un mensaje de error.

1.5 Interfaz de usuario

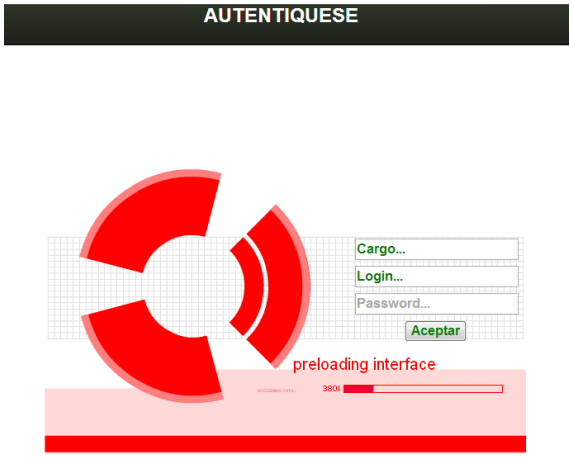


Figura IV.6. Autenticación de usuario.

4.2.2.11.3 Requerimiento funcional 3

Como Director quiero el sistema permita el control de gestión de las entidades tales como, el ingreso, modificación, actualización de los datos de cada Proyecto, Docente, Pasante, Seguimiento de proyecto, Productos de proyecto, Patente, Difusión científica de investigación, Difusión científica de transferencia, Publicación de Artículos, Tesis de grado.

1.1 Introducción

El sistema debe permitir la gestión (CRUD)[10] de las entidades que conforman la base de datos del sistema a desarrollar.

[10] Create Retrive Update Delete

1.2 Entradas

Permitirá la gestión de las diferentes entidades donde sus entradas dependerán de la entidad en gestión, en caso general se ha escogido la entidad principal del sistema.

Tabla IV.IV. Entradas requerimiento funcional 3.

Fuente:	Base de datos
Cantidad:	Uno por usuario.
Frecuencia:	Demanda alta.
Rango de entradas válidas:	Disponible todo el tiempo

- Código provincia (R)
- Código cantón (R)
- Código parroquia (R)
- Código unidad responsable (R)
- Código línea investigación (R)
- Código área investigación (R)
- Código tipo impacto(R)
- Nombre (R)
- Tipo (R)
- Tipo investigación (R)
- Inv Adicionales (R)
- Plan Buen Vivir (R)
- Estado (R)
- Fecha Inicio (R)
- Fecha Fin (R)
- Entidades Auspiciantes (R)
- Presupuesto (R)
- Anexo(R)

1.3 Proceso

- Seleccionar la entidad a gestionar el caso de un proyecto de investigación.
- Seleccionar lo que desea realizar en esta entidad, ya sea agregar un nuevo registro, modificarlo o eliminarlo.
- Si selecciona agregar, insertar la entidad.
- Si selecciona modificar, editar la entidad.

1.4 Salidas

- El sistema verifica que los datos sean válidos.
- Si son datos erróneos presenta un mensaje de error.

- El sistema registra el proyecto.

1.5 Interfaz de usuario

Figura IV.7. Interfaz principal de gestión de proyectos.

4.2.2.11.4 Requerimiento funcional 4

Como Director quiero el sistema debe permita realizar búsquedas de las entidades por código, nombre etc.

1.1 Introducción

El administrador del sistema puede realizar búsquedas de los registros dado un campo de filtro ya sea código o por su nombre.

1.2 Entradas

Tabla IV.V. Entradas requerimiento funcional 4.

Fuente:	Base de datos
Cantidad:	Uno por usuario.
Frecuencia:	Demanda alta.
Rango de entradas válidas:	Disponible todo el tiempo

- Código (o)

- Nombre (o)

1.3 Proceso

El administrador realiza una búsqueda de un registro dado un parámetro de ingreso.

El sistema valido en ingreso en caso de ser erróneo emite un mensaje advertencias.

1.4 Salidas

Finamente ejecuta la búsqueda, visualizado los datos del registro obtenido.

1.5 Interfaz de usuario

Figura IV.8. Interfaz de listado de una entidad.

4.2.2.11.5 Requerimiento funcional 5

Como Director quiero que permita realizar el seguimiento de los diferentes proyectos existentes.

1.1 Introducción

Cada proyecto registrado en el sistema SGCOM tiene diferentes estados por los que deber cursar hasta su culminación.

1.2 Entradas

Tabla IV.VI. Entradas requerimiento funcional 5.

Fuente:	Base de datos
Cantidad:	Uno a la vez.
Frecuencia:	Demanda alta.

Rango de entradas válidas:	Disponible todo el tiempo

- Código provincia (R)
- Objetivos Esperados (R)
- Responsable Principal (R)
- Causas Desviación (R)
- Presupuesto Utilizado (R)
- Fecha Evaluación (R)
- Etapa (R)
- Objetivos Logrados (R)
- Porcentaje Cumplimiento(R)
- Acciones Correctivas (R)
- Presupuesto Disponible (R)
- Fechas Máxima Cumplimiento (R)

1.3 Proceso

El usuario realiza el seguimiento de los proyectos filtrando por su estado.

Verifica el los datos de cada proyecto

Las observaciones generadas se registran en los campos de objetivos según sea el caso.

1.4 Salidas

Objetivos deseados

Objetivos logrados

1.5 Interfaz de usuario

Figura IV.9. Ingreso de seguimientos de proyectos.

4.2.2.11.6 Requerimiento funcional 6

Como Director quiero que permita registrar los diferentes productos que han generado en los diferentes proyectos existentes con su respectiva patente en caso que lo tenga.

1.1 Introducción

Aquí se registrara los diferente productos generados por las investigaciones con su respectiva marca o patente en el caso de que tenga un certificado de valides.

1.2 Entradas

Tabla IV.VII. Entradas requerimiento funcional 6.

Fuente:	Base de datos
Cantidad:	Uno a la vez.
Frecuencia:	Demanda baja.
Rango de entradas válidas:	Disponible todo el tiempo

- Código proyecto
- Nombre
- Presupuesto Invertido
- Descripción
- Fecha Registro

1.3 Proceso

El Investigador procede a informar según el avance de proyecto que se ha culminado con la producción de un producto.

El administrador procede a registrar los datos de dicho producto

El sistema valida los datos ingresados en caso de error emite un mensaje de advertencia.

1.4 Salidas

Producto registrado

1.5 Interfaz de usuario

Figura IV.10. Ingreso de un producto.

4.2.2.11.7 Requerimiento funcional 7

Como Director quiero que el sistema permita registrar Docentes con sus respetivas tesis de estudiantes en dirección.

1.1 Introducción

El sistema permitirá registrar el ingreso de docentes con su respetivo proyecto y tesis en dirección.

1.2 Entradas

Tabla IV.VIII. Entradas requerimiento funcional 7.

Fuente:	Base de datos
Cantidad:	Uno a la vez.
Frecuencia:	Demanda alta.
Rango de entradas válidas:	Disponible todo el tiempo

- Código facultad
- Título Tercer Nivel
- Periodo Académico
- Cedula
- Títulos Posgrado
- Correo Electrónico
- Apellidos Nombres
- Unidad Académica

1.3 Proceso

El Docente se acerca al departamento a registrar su investigación en curso,

El administrador procede a registrar al docente,

El sistema valida los datos ingresados,

El administrador registra el proyecto,

El sistema valida los datos ingresados.

1.4 Salidas

Finalmente queda registrado el docente

1.5 Interfaz de usuario

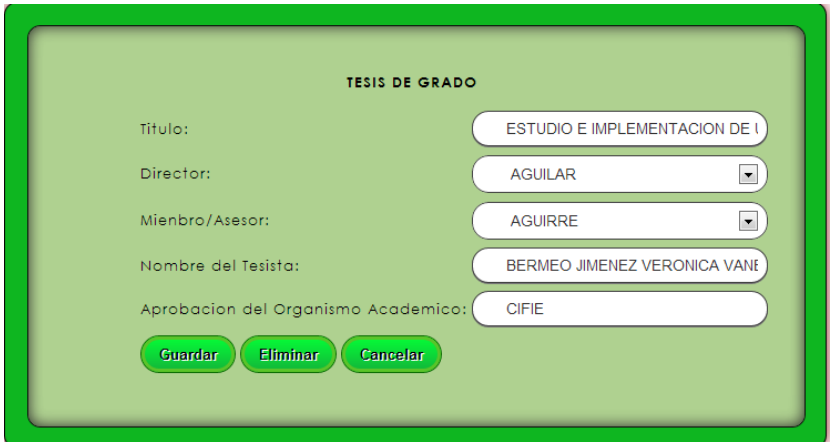


Figura IV.11. Ingreso tesis de grado.

4.2.2.11.8 Requerimiento funcional 8

Como Director quiero que el sistema permita la gestión de Difusión científica de investigación, Difusión científica de transferencia, Publicación de Artículos.

1.1 Introducción

El sistema debe registra y publicar artículos como: Difusión científica de investigación, Difusión científica de transferencia y Publicación de Artículos.

1.2 Entradas

Tabla IV.IX. Entradas requerimiento funcional 8.

Fuente:	Base de datos
Cantidad:	Uno a la vez.
Frecuencia:	Demanda alta.
Rango de entradas válidas:	Disponible todo el tiempo

dci_codigo
pro_codigo
td_codigo
dci_nombre
dci_fecha
dci_numEjemplares
dci_editorial

part_nombreProyecto
part_titulo
part_tipoRevista
part_nombreRevista
part_impresa
part_digital
part_fecha
part_pasi
part_aprobacionOrgAcad

dct_codigo
te_codigo
dct_nombre
dct_lugar
dct_mujeres
dct_hombres
dct_total
dct_fechaInicio
dct_fechaFin
dct_duracionHoras
dct_beneficiarios

1.3 Proceso

El administrador procede a registrar los datos de los principales artículos que serán manejados a gusto de la institución.

1.4 Salidas

Publicación de los siguientes artículos: Difusión científica de investigación, Difusión científica de transferencia y Publicación de Artículos, para poner al alcance del usuario.

1.5 Interfaz de usuario

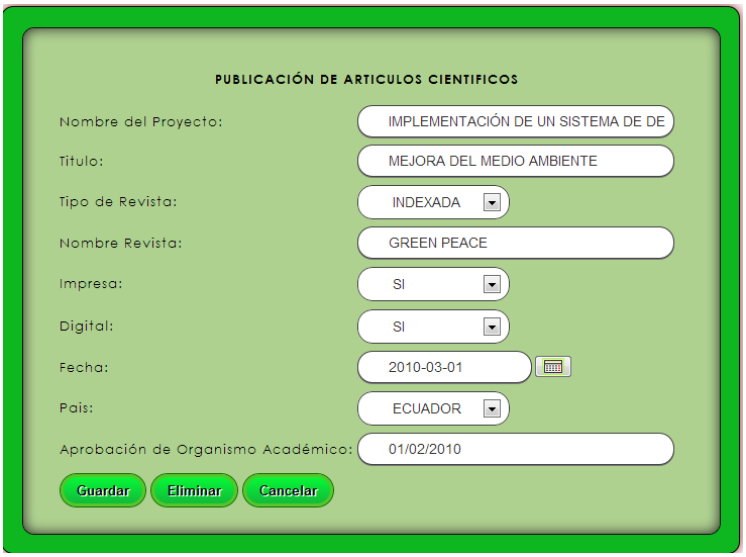


Figura IV.12. Ingreso de publicación de artículos científicos.

Figura IV.13. Ingreso discusión científica de investigación.

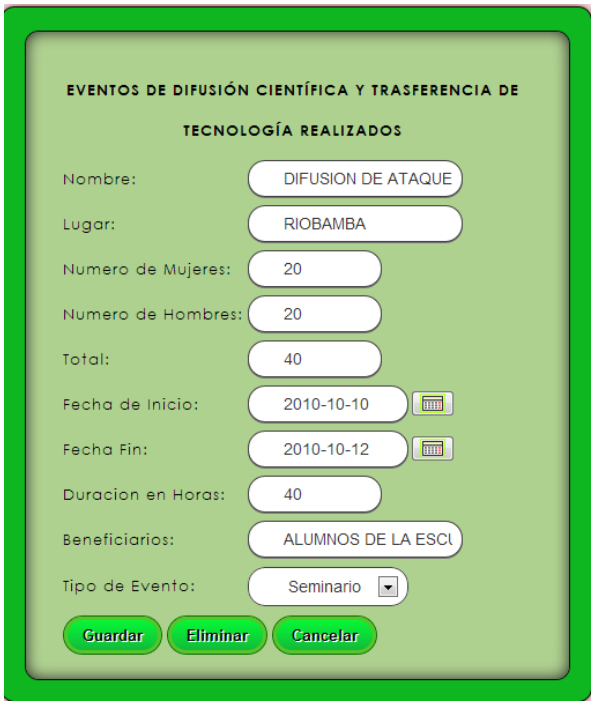


Figura IV.14. Ingreso eventos de difusión científica y transparencia de tecnologías realizados

4.2.2.11.9 Requerimiento funcional 9

Como Director quiero que la información pública sea visualizada a través de la web para que nuestros clientes puedan acceder a nuestros servicios desde cualquier parte.

1.1 Introducción

Toda la información que puede ser accedida por cualquier usuario será pública en el portal.

1.2 Entradas

Tabla IV.X. Entradas requerimiento funcional 9.

Fuente:	Web
Cantidad:	Uno a la vez.
Frecuencia:	Demanda alta.
Rango de entradas válidas:	Disponible todo el tiempo

1.3 Proceso

El administrador está apto para gestión un portal de contenidos desarrollado con Joomla que el permita la publicación de artículos que desee.

1.4 Salidas

Información pública como reportes, artículos, noticia, avisos, para los usuarios externos.

1.5 Interfaz de usuario

Figura IV.15. Publicación de reportes.

4.2.2.11.10 Requerimiento funcional 10

Como Director quiero que la información publicada en las cuentas de Twitter, y Facebook se integre para su visualización en el portal web

1.1 Introducción

El portal para manejo de contenidos deberá visualizar las publicaciones del twitter y Facebook.

1.2 Entradas

Tabla IV.XI. Entradas requerimiento funcional 10.

Fuente:	Web
Cantidad:	Todo.
Frecuencia:	Demanda alta.
Rango de entradas válidas:	Disponible todo el tiempo

1.3 Proceso

El administrado deberá gestionar su cuenta de twitter y Facebook publicar notas.

El administrador deberá ingresar al portal de gestión de contenidos con constatar la visualización de las notas publicadas en las redes sociales nombradas anteriormente.

1.4 Salidas

Notas públicas en Facebook y Twitter.

1.5 Interfaz de usuario

Figura IV.16. Publicación de información accesible para el usuario.

4.2.2.12Requerimientos de Interfaces Externas

4.2.2.13Interfaces Hardware

Conectores RJ45.

Computadora.

Impresora.

4.2.2.14Interfaces Software.

SO (XP, VISTA, SEVEN, Linux,etc.).

Servidor de BD PostgreSQL.

Servidor GlassFish 3.0

4.2.2.15Interfaces de Comunicación

TCP/IP.

4.2.2.16Requerimientos de Rendimiento

4.2.2.17Número de Usuarios simultáneos:

Existen múltiples usuarios que van a manejar el sistema.

4.2.2.18Estaciones de trabajo:

Existen siete estaciones de trabajo.

4.2.2.19Limitaciones de Diseño

4.2.2.20Obediencia a Estándares

SRS IEEE830.

ISO 9300.

4.2.2.21Atributos.

Seguridad

El sistema proporcionará la integridad y la seguridad de toda la información que se maneje debido a la manera de implementación que se utilizará. Siendo el sistema eficaz, confiable y muy seguro.

Mantenibilidad

El sistema debido a su implementación será fácil de darle mantenimiento, al mismo tiempo su mantenimiento será económico.

Escalabilidad

El sistema es flexible ya que se pueden incrementar nuevas funciones de acuerdo a los requerimientos que puedan presentarse.

Interfaz amigable

El sistema cuenta con una interfaz de fácil manejo para el usuario final.

Disponibilidad

El sistema deberá estar disponible durante todo el día.

4.2.2.22Otros requerimientos

4.2.2.22.1 Base de datos

En esta etapa se va a definir un bosquejo de la base de datos de la aplicación que se va a desarrollar.

[1,1]

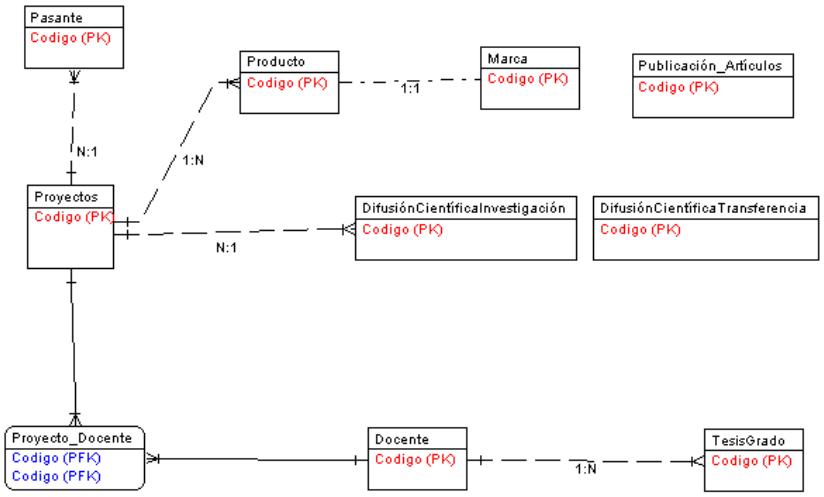


Figura IV.17. Bosquejo de la BD.

4.2.2.22.2 Operaciones

Que exista el control previo al desarrollo cada una de las etapas o requisitos a cumplirse para la gestión de proyectos impulsados por la COMPROTEC.

4.2.2.22.3 Adaptación del sitio

Instalación Eléctrica en la que se sugiere una toma general de corriente para todo el equipo y una individual para cada dispositivo.

Se requieren adaptaciones a nivel de infraestructura física, espacio y cambios necesarios para la adecuación de la red.

4.2.3 DESARROLLO DE BASE DE DATOS

4.2.3.1 Diseño de base de datos

A aquí se presenta el diseño final de la base de datos de la COMPROTEC.

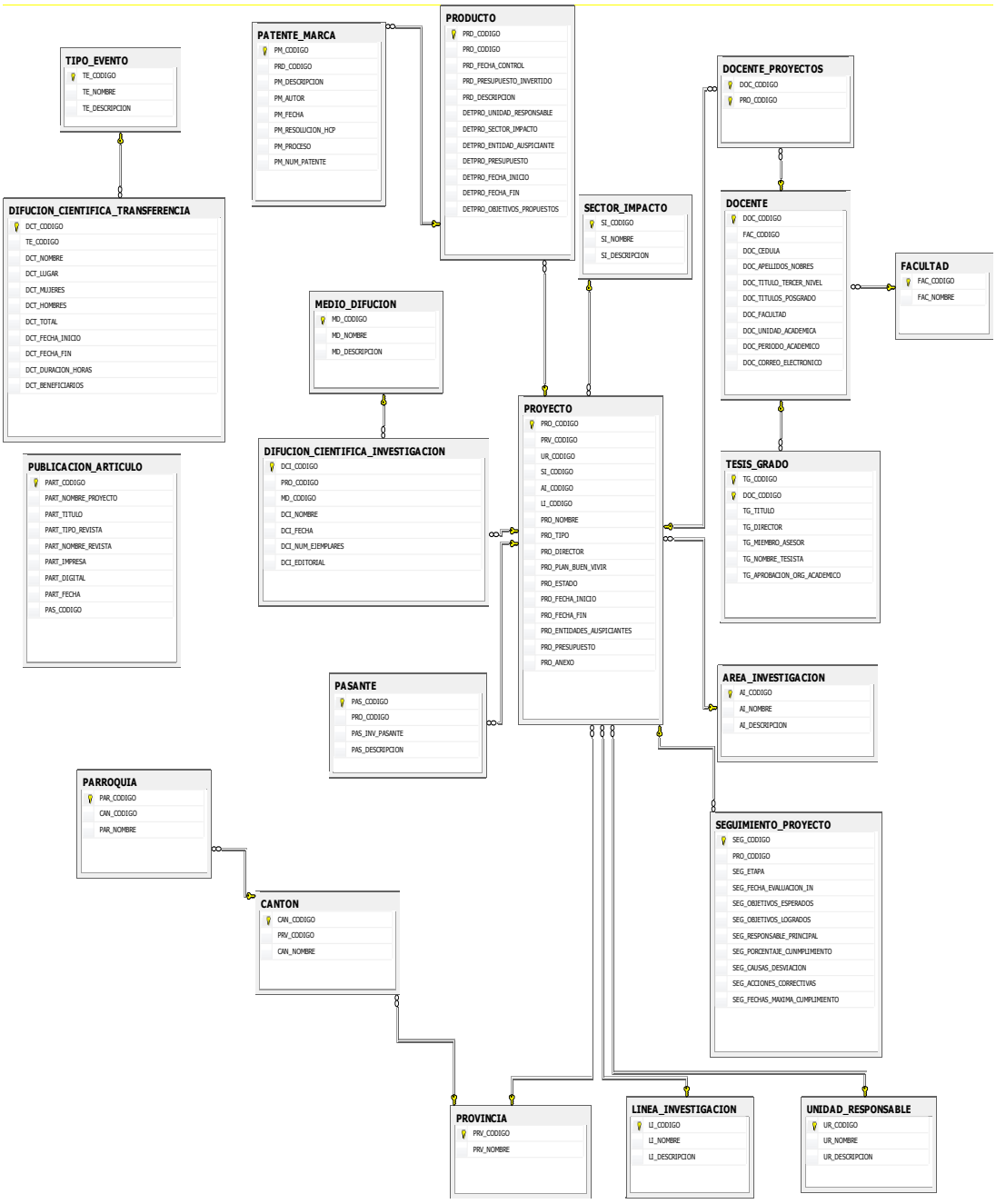


Figura IV.18. Diseño inicial de BD.

4.2.3.2 Presentación de prototipo

Durante esta presentación se establecieron algunos cambios por parte del cliente en cuanto a la base de datos, es decir, se hizo la inclusión de una nueva tabla denominada Login, se agregaron nuevos atributos, se establecieron relaciones correctas entre las tablas y se crearon algunos procedimientos almacenados.

4.2.3.3 Diseño final de base de datos

De acuerdo a estas observaciones se realizó el diseño final de la base de datos, quedando totalmente estructurada.

4.2.3.4 Diagrama de base de datos

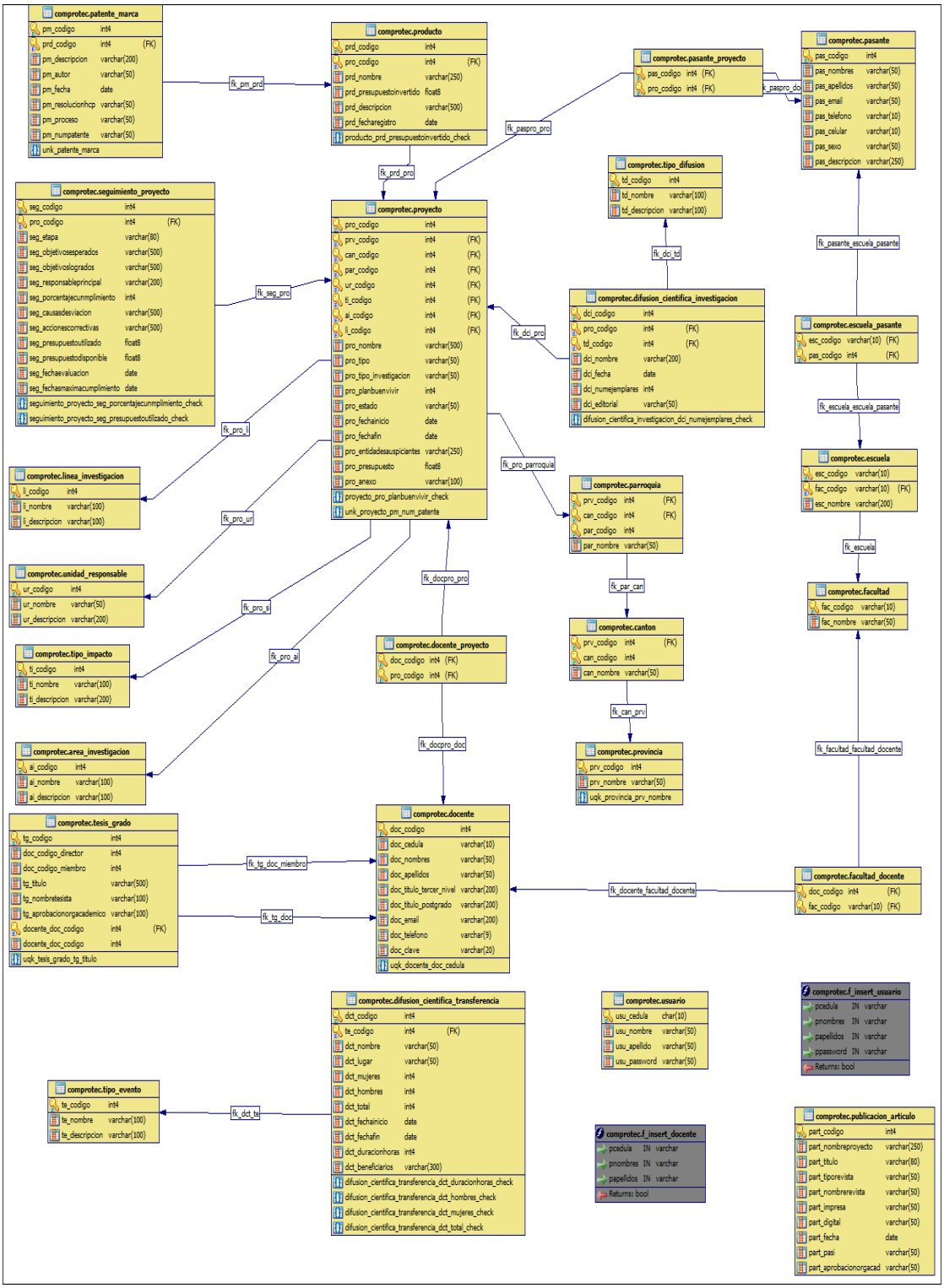


Figura IV.19. Diseño final de la Base de Datos.

4.2.3.5 ARQUITECTURA DE SOFTWARE

Analizando los requerimientos del sistema y en base a la experiencia de los desarrolladores se ha tomado la decisión de utilizar las siguientes herramientas de desarrollo.

- Java como lenguaje de programación ya que este lenguaje no proporcionara un ágil desarrollo de la aplicación a través de la utilización del framework Spring MVC.
- Servidor de aplicaciones Glassfish, ya que brinda una estabilidad al realizar peticiones de muchos usuarios del sistema.
- IDE de desarrollo de Netbeans 7.2.1, se eligió esta herramienta ya que se adapta al lenguaje de programación elegido anteriormente y además ya es conocido por los desarrolladores. Ventaja que llevará a un desarrollo rápido de la aplicación.

En cuanto se refiere a la arquitectura del sistema, se ha tomado la decisión de utilizar el patrón Modelo Vista Controlado (MVC) ya que es la principal funcionalidad del framework a utilizar.[10]

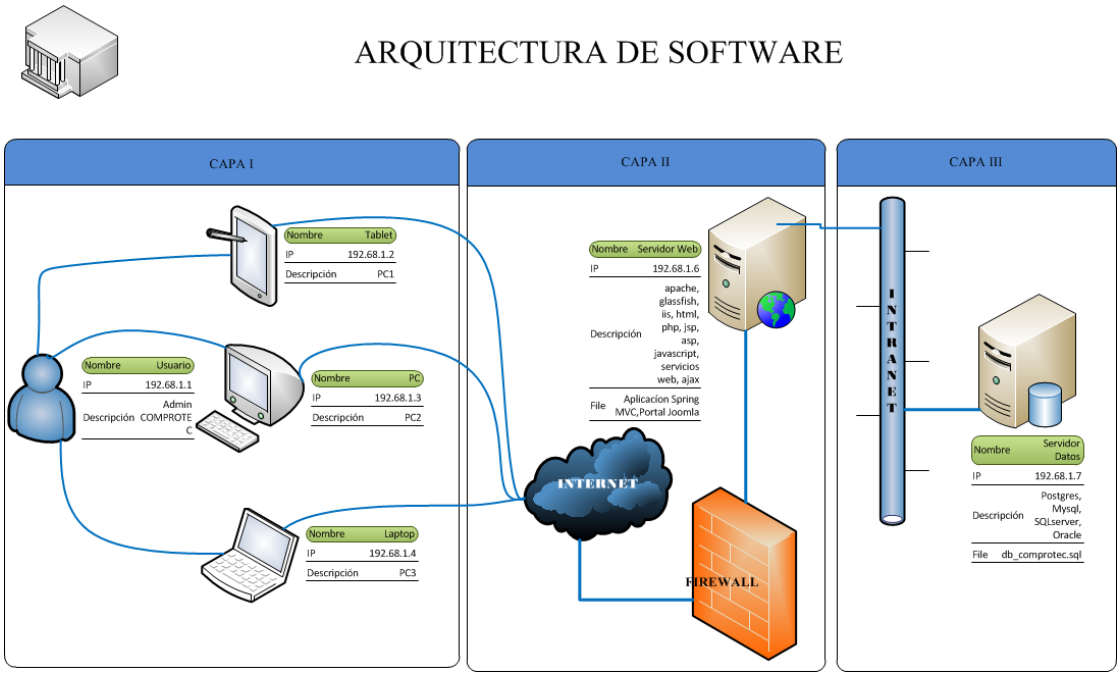


Figura IV.20 Arquitectura de Software

En la Figura IV.20 se presenta la arquitectura detallada del sistema con el que se maneja el sistema de automatización de las gestiones de la COMPROTEC.

4.2.4 FASE DE DESARROLLO DE CAPAS

4.2.4.1 Mapeo de la Base de Datos

Tal como indica la metodología DAWE, se debe de realizar el mapeo de los datos para representar las clases en nuestra aplicación.

Para realizar esta acción se crea el paquete **comprotec.model y comprotec.mapeo,** en el primero se encontraran todas las clases mapeadas de la base de datos y en el segundo paquete se encontraran los archivos de configuración del mapeo con hibernate.[3]

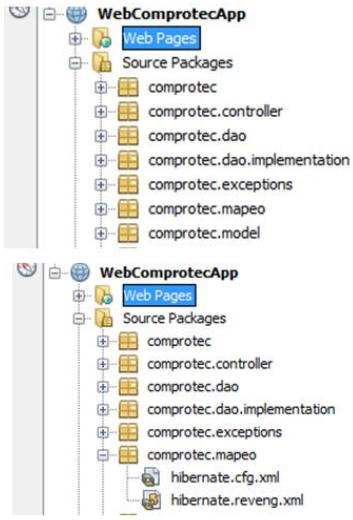


Figura IV.21. Estructura de paquetes.

Tal como indica la metodología, se utiliza el ORM Hibernate para realizar el mapeo.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hib
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
    <property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
    <property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/comprotec_db</property>
    <property name="hibernate.connection.username">postgres</property>
    <property name="hibernate.connection.password">EIS4180</property>
    <property name="hibernate.show_sql">true</property>
    <property name="hibernate.current_session_context_class">thread</property>
    <mapping resource="comprotec/model/Proyecto.hbm.xml"/>
    <mapping resource="comprotec/model/LineaInvestigacion.hbm.xml"/>
    <mapping resource="comprotec/model/TipoDifusion.hbm.xml"/>
    <mapping resource="comprotec/model/DifusionCientificaInvestigacion.hbm.xml"/>
    <mapping resource="comprotec/model/Facultad.hbm.xml"/>
    <mapping resource="comprotec/model/UsuarioAutorizacion.hbm.xml"/>
    <mapping resource="comprotec/model/TipoImpacto.hbm.xml"/>
    <mapping resource="comprotec/model/Producto.hbm.xml"/>
    <mapping resource="comprotec/model/TipoEvento.hbm.xml"/>
    <mapping resource="comprotec/model/SeguimientoProyecto.hbm.xml"/>
    <mapping resource="comprotec/model/PublicacionArticulo.hbm.xml"/>
    <mapping resource="comprotec/model/AreaInvestigacion.hbm.xml"/>
    <mapping resource="comprotec/model/TesisGrado.hbm.xml"/>
    <mapping resource="comprotec/model/Provincia.hbm.xml"/>
    <mapping resource="comprotec/model/Usuario.hbm.xml"/>
    <mapping resource="comprotec/model/Pasante.hbm.xml"/>
    <mapping resource="comprotec/model/Canton.hbm.xml"/>
    <mapping resource="comprotec/model/PatenteMarca.hbm.xml"/>
```

Figura IV.22. Archivo de Configuración hibernate.cfg.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse E
<hibernate-reverse-engineering>
  <schema-selection match-catalog="comprotec_db" match-schema="comprotec"/>
  <table-filter match-name="facultad"/>
  <table-filter match-name="tipo_impacto"/>
  <table-filter match-name="seguimiento_proyecto"/>
  <table-filter match-name="usuario"/>
  <table-filter match-name="difusion_cientifica_investigacion"/>
  <table-filter match-name="provincia"/>
  <table-filter match-name="area_investigacion"/>
  <table-filter match-name="docente"/>
  <table-filter match-name="usuario_autorizacion"/>
  <table-filter match-name="tipo_evento"/>
  <table-filter match-name="canton"/>
  <table-filter match-name="difusion_cientifica_transferencia"/>
  <table-filter match-name="tesis_grado"/>
  <table-filter match-name="pasante_proyecto"/>
  <table-filter match-name="proyecto"/>
  <table-filter match-name="producto"/>
  <table-filter match-name="unidad_responsable"/>
  <table-filter match-name="parroquia"/>
  <table-filter match-name="patente_marca"/>
  <table-filter match-name="pasante"/>
  <table-filter match-name="tipo_difusion"/>
  <table-filter match-name="publicacion_articulo"/>
  <table-filter match-name="docente_proyecto"/>
  <table-filter match-name="linea_investigacion"/>
</hibernate-reverse-engineering>
```

Figura IV.23. Archivo de Configuración hibernate.reveng.xml

A través de los archivos presentados se puede realizar el mapeo para obtener las distintas clases en el proyecto.

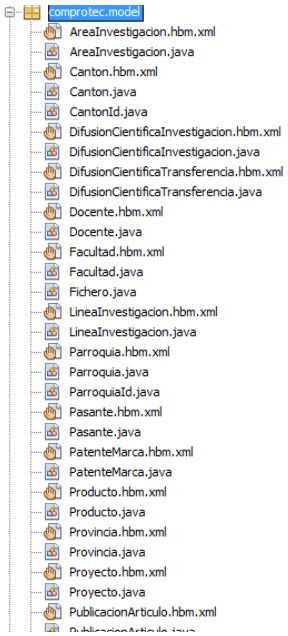


Figura IV.24. Mapeo de la Base de datos.

4.2.4.2 Construcción de Paquetes

El siguiente paso es crear distintos paquetes del proyecto en caso se debe crear los paquetes siguientes:

- comprotec.dao.
- comptotec.dao.implementation.
- comprotec.service.
- comprotec.service.implementation.

A través de todos estos paquetes se define los accesos a la base de datos así como también los distintos controles.

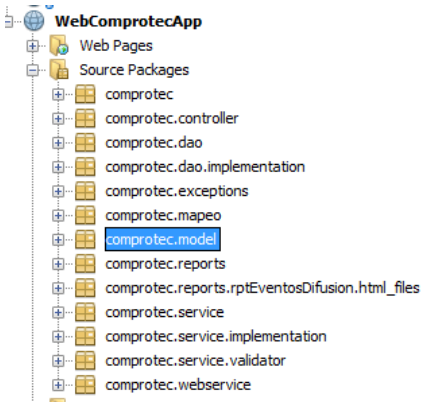


Figura IV.25. Construcción de paquetes

4.2.4.3 Generación de da Capa de Interfaces

Ahora se va a desarrollar las distintas clases que van dentro de cada paquete de interfaz, esto se lo desarrolla ya que el framework SPRING MVC es robusto trabajando con interfaces.

Para nuestro proyecto se define las siguientes interfaces para el paquete **comprotec.dao**

AreaInvestigacionDao.
DifusionCientificaTransferenciaDao.
LineaInvestigacionDao.
PatenteMarcaDao.
ProyectoDao.
SeguimientoProyectoDao.
TipoDifusionDao.
TipoImpactoDao.
UsuarioDao.
DifusionInvestigacionDao.
DocenteDao.
PasanteDao.
ProductoDao.
PublicacionArticuloDao.
TesisGradoDao.
TipoEventoDao.
UnidadResponsableDao.

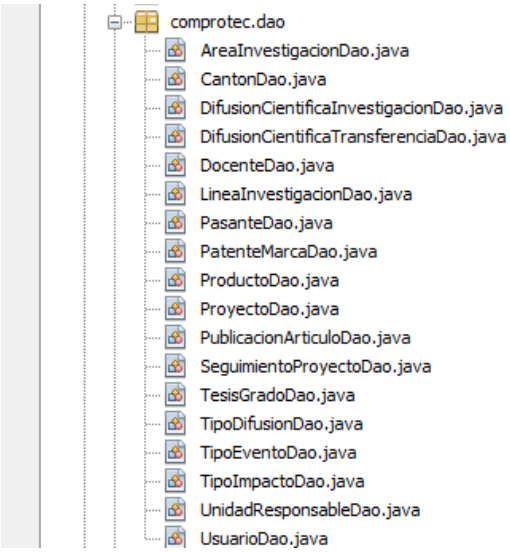


Figura IV.26. Estructura de la capa de interfaces.

Tal como indica la metodología se debe de solo declarar los nombres de los métodos en cada clase **DAO** desarrollada.

```
5
6   package comprotec.dao;
7
8   import comprotec.model.Facultad;
9   import comprotec.model.Pasante;
10  import java.util.List;
11
12  /**
13   *
14   * @author pcuser
15   */
16  public interface PasanteDao {
17          public List<Pasante> getPasantesList(Pasante pasante);
18          public Pasante getPasanteById(Integer idPasante);
19          public void savePasante(Pasante pasante);
20          public void deletePasante(Integer idPasante);
21
22          public List<Facultad> getFacultadList();
23  }
24
```

Figura IV.27. Implementación de una interfaz DAO.

4.2.4.4 Generación de la Capa de Persistencia de Datos (Dao)

Ahora se va a proceder desarrollar los métodos definidos en las interfaces previas, se utiliza las interfaces definidas previamente y tal como indica la metodología para todas las clases se aplica la anotación **@Repository** para indicar que es una clase de acceso a datos.

Para el proyecto se define las siguientes interfaces para el paquete **comprotec.dao**

AreaInvestigacionDaoImpl.
DifusionCientificaTransferenciaDaoImp
DifusionInvestigacionDaoImpl.
DocenteDaoImpl.
LineaInvestigacionDaoImpl.
PatenteMarcaDaoImpl.
ProyectoDaoImpl.
SeguimientoProyectoDaoImpl.
TipoDifusionDaoImpl.
TipoImpactoDaoImpl.
UsuarioDaoImpl.
PasanteDaoImpl.
ProductoDaoImpl.
PublicacionArticuloDaoImpl.
TesisGradoDaoImpl.
TipoEventoDaoImpl.
UnidadResponsableDaoImpl.

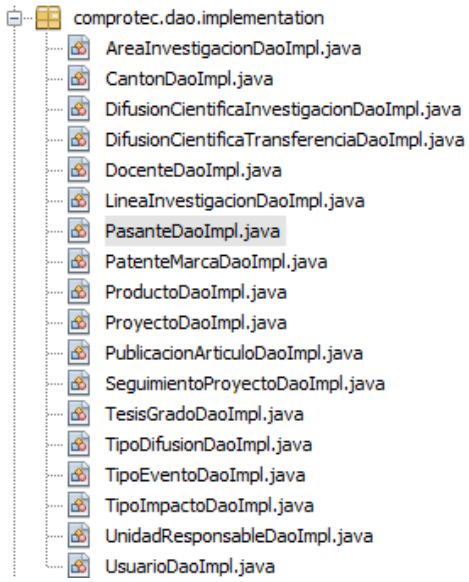


Figura IV.28. Generación de la Capa de Persistencia.

```
@Repository
public class PasanteDaoImpl implements PasanteDao {

    private HibernateTemplate hibernateTemplate;
    private JdbcTemplate jdbcTemplate;

    @Autowired
    public PasanteDaoImpl(SessionFactory sessionFactory) {
        this.hibernateTemplate = new HibernateTemplate(sessionFactory);
        this.jdbcTemplate = new JdbcTemplate(SessionFactoryUtils.getDataSource(sessionFactory));
    }

    public List<Pasante> getPasantesList(Pasante pasante) {
        StringBuilder query = new StringBuilder("from Pasante ");

        if (pasante != null && pasante.getPasFiltro() != null
                && pasante.getPasFiltro().length() > 0) {

            query.append("where upper(pasNombres) like '%").append(pasante.getPasFiltro().
                    toUpperCase()).append("%' ").append("or upper(pasApellidos) like'%").append(pasante.get
                    toUpperCase()).append("%' ").append("or upper(pasEmail) like'%").append(pasante.getPasF
                    toUpperCase()).append("%' ").append("or upper(pasTelefono) like'%").append(pasante.getP
                    toUpperCase()).append("%' ").append("or upper(pasCelular) like'%").append(pasante.getPa
                    toUpperCase()).append("%' ").append("or upper(pasDescripcion) like'%").append(pasante.g
                    toUpperCase()).append("%' ");
        }

        List<Pasante> list = (List<Pasante>) hibernateTemplate.find(query.append("order by pasApellidos").t
        return list;
    }
```

Figura IV.29. Implementación de la capa de persistencia.

4.2.4.5 Generación de la Capa de Lógica de Negocio

Comenzamos a desarrollar las clases del paquete **comprotec.service** , en estas clases se emplea la anotación @Service para representar una clase de lógica de negocio, en este paquete se define las clases siguientes:

AreaInvestigacionServiceImpl.
DifusionCientificaTransferenciaServiceImpl.
LineaInvestigacionServiceImpl.
PatenteMarcaServiceImpl.
ProyectoServiceImpl.
SeguimientoProyectoServiceImpl.
TipoDifusionServiceImpl.
TipoImpactoServiceImpl
UsuarioServiceImpl.
DifusionInvestigacionServiceImpl.
DocenteServiceImpl.
PasanteServiceImpl.
ProductoServiceImpl.
PublicacionArticuloServiceImpl
TesisGradoServiceImpl.
TipoEventoServiceImpl.
UnidadResponsableServiceImpl.

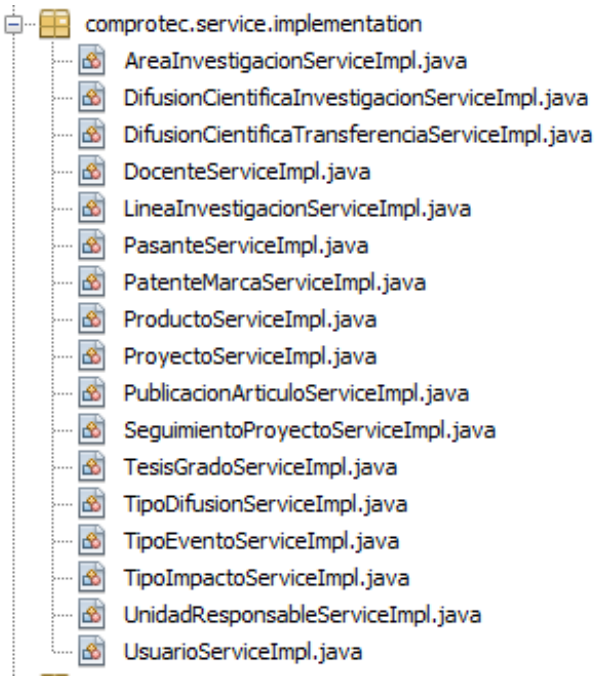


Figura IV.30. Generación de la Lógica de Negocio.

```
20  @Service
21  public class PasanteServiceImpl implements PasanteService{
22
23      @Autowired
24      private PasanteDao pasanteDao;
25
26      public PasanteDao getPasanteDao() {
27          return pasanteDao;
28      }
29
30      public void setPasanteDao(PasanteDao pasanteDao) {
31          this.pasanteDao = pasanteDao;
32      }
33
34
35
37      public List<Pasante> getPasantesList(Pasante pasante) {
38          return getPasanteDao().getPasantesList(pasante);
39      }
41      public Pasante getPasanteById(Integer idPasante) {
42          return getPasanteDao().getPasanteById(idPasante);
43      }
45      public void savePasante(Pasante pasante) {
46          getPasanteDao().savePasante(pasante);
      }
```

Figura IV.31. Implementación de la Lógica de Negocio.

4.2.5 FASE DE INTEGRACIÓN DE FRAMEWORK SPRING MVC

4.2.5.1 Configuración de Contenedor de Objetos application-context.xml

Ahora se configura la inyección de dependencias en el archivo XML las configuraciones que se realizan en este archivo son las siguientes:[3][11][14]

- Conexión a la base de datos.
- Objetos que se deben crear.

```
<context:component-scan base-package="comprotec.model" />

<!-- Activa las anotaciones que controlan las transacciones en la BD -->
<bean id="propertyConfigurer"
      class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"
      p:location="/WEB-INF/config/database/jdbc.properties" />

<!-- CONFIGURACION ESTATICA DE LA BASE DE DATOS -->
<!-- Datos de la conexion a la BD, son insertados desde el jdbc.properties -->
<bean id="dataSource"
      class="org.springframework.jdbc.datasource.DriverManagerDataSource"
      p:driverClassName="${jdbc.driverClassName}"
      p:url="${jdbc.url}"
      p:username="${jdbc.username}"
      p:password="${jdbc.password}"/>

<!-- Para permitir la ejecucion de transacciones SQL -->
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager"
      p:dataSource-ref="dataSource" />


<bean id="sessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</prop>
            <prop key="hibernate.show_sql">true</prop>
            <prop key="hibernate.current_session_context_class">thread</prop>
        </props>
```

Figura IV.32. Configuración del archivo application-context.xml

```
<bean id="usuarioDao" class="comprotec.dao.implementation.UsuarioDaoImpl">
    <constructor-arg ref="sessionFactory"/>
</bean>
<bean id="servicio" class="comprotec.service.implementation.UsuarioServiceImpl">
    <property name="usuarioDao">
        <ref bean="usuarioDao" />
    </property>
</bean>
<bean id="controlador" class="comprotec.controller.UsuarioController">
    <property name="usuarioService">
        <ref bean="servicio" />
    </property>
</bean>
<bean id="usuarioValidator" class="comprotec.service.validator.UsuarioValidator">
    <property name="usuarioService">
        <ref bean="servicio" />
    </property>
</bean>
```

Figura IV.33. Inyección de dependencia del objeto usuario.

4.2.5.2 Configuración del dispatcher-servlet.

La configuración de este archivo es para trabajar con la extensión de las páginas web que en este caso se trabaja con ***.htm.[6]**

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:p="http://www.springframework.org/schema/p"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schem
       http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.x
       http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
       http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-c

        <bean class="org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping"/>
        <context:component-scan base-package="comprotec.controller"/>
        <bean id="messageSource" class="org.springframework.context.support.ReloadableResourceBundleMessage
               <property name="basename" value="/WEB-INF/config/messages/validation" />
        </bean>
        <bean id="viewResolver"
          class="org.springframework.web.servlet.view.InternalResourceViewResolver"
          p:prefix="/WEB-INF/jsp/"
          p:suffix=".jsp" />
        <!-- Returns messages based on a resource bundle -->

</beans>
```

Figura IV.34. Configuración de Dispatcher-servlet.

4.2.5.3 Generación de la Capa de Controladores

Definición de los controladores a crear que son los siguientes:

AreaInvestigacionController.
DifusionCientificaTransferenciaController.
LineaInvestigacionController.
PatenteMarcaController.
ProyectoController.
SeguimientoProyectoController.
TipoDifusionController.
TipoImpactoController.
UsuarioController.
DifusionInvestigacionController.
DocenteController.
PasanteController.
ProductoController.
PublicacionArticuloController.
TesisGradoController.
TipoEventoController.
UnidadResponsableController.

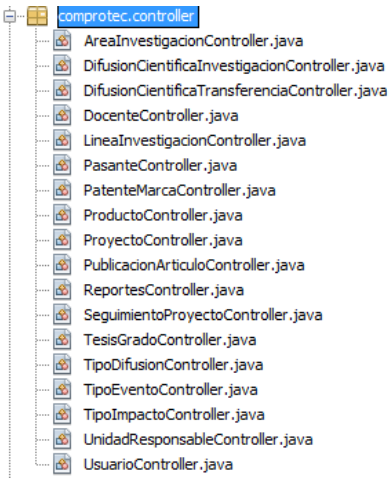


Figura IV.35. Generación de controladores.

Se trabaja con la anotación **@Controller** que definirá la clase como si fuera un controlador.

```
@Controller
public class PasanteController {
    @Autowired
    private PasanteService pasanteService;

    @Autowired
    private PasanteValidator pasanteValidator;

    public void setPasanteService(PasanteService pasanteService) {
        this.pasanteService = pasanteService;
    }



    @RequestMapping("pasante/pasanteList.htm")
    public void pasanteList(Model model,
                    @ModelAttribute("pasante") Pasante pasante) {
            List<Pasante> pasantes = pasanteService.getPasantesList(pasante);
            model.addAttribute("pasantes", pasantes);
            model.addAttribute("pasante", pasante);
    }
    @RequestMapping(value = "pasante/pasanteView.htm", method = RequestMethod.GET)
        public @ModelAttribute("pasante")Pasante pasanteView
                        (@RequestParam(value = "pasCodigo", required = false) Integer pasCodigo) {
                if (pasCodigo != null) {
                        Pasante pasante = pasanteService.getPasanteById(pasCodigo);
                        return pasante;
                }
```

Figura IV.36. Definición de los controladores.

4.2.6 FASE DE INTEGRACIÓN DE CAPA DE PRESENTACIÓN

4.2.6.1 Generación de la capa de vistas.

Se proceder a crear las vistas de presentación, además se crea una carpeta por cada vista generalizando las clases.

areainvestigacion.
difusioncientificatransferencia.
lineainvestigacion.
patentemarca.
proyecto.
seguimientoproyecto.
tipodifusion.
tipoimpacto.
usuariocontroller.
difusioninvestigacion.
docente.
pasante.
producto.
publicacionarticulo.
tesisgrado.
tipoevento.
unidadresponsable.

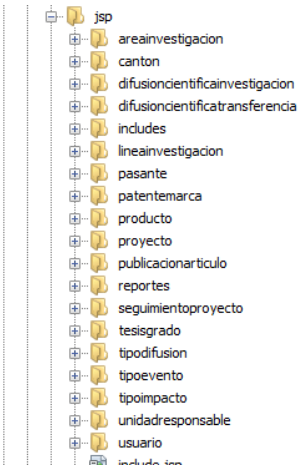


Figura IV.37. Estructura de vistas.

En cada carpeta se define las páginas de listar y mostrar vistas que servirán para realizar la gestión de cada módulo.

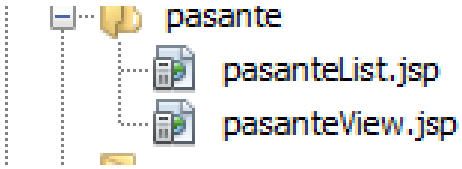


Figura IV.38. Estructura de vistas de una entidad.

Como se observa las vistas están creadas como jsp pero su tratamiento se lo realiza como htm.

4.2.6.2 Generación de la Capa de Validación.

Se debe de desarrollar la capa **comprotec.service.validator,** esta capa sirvió para definir las restricciones de la base de datos así como validar los tipos de datos y formatos de los datos a ingresar.[15]

En esta capa se define las siguientes clases:

AreaInvestigacionValidator.
DifusionCientificaTransferenciaValidator.
LineaInvestigacionValidator.
PatenteMarcaValidator.
ProyectoValidator.
SeguimientoProyectoValidator.
TipoDifusionValidator.
TipoImpactoValidator.
UsuarioValidator.
DifusionInvestigacionValidator.
DocenteValidator.
PasanteValidator.
ProductoValidator.
PublicacionArticuloValidator.
TesisGradoValidator.
TipoEventoValidator.
UnidadResponsableValidator

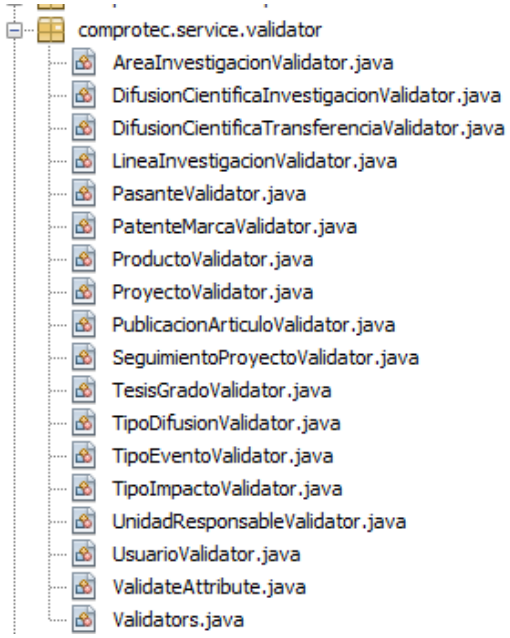
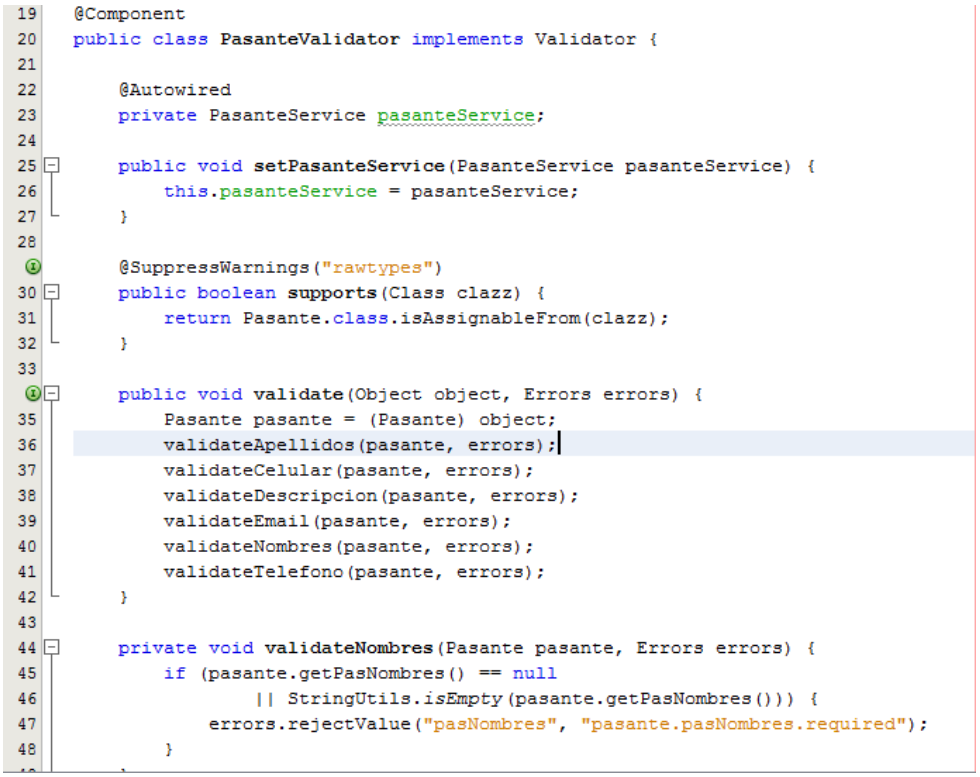


Figura IV.39. Generación de validadores en Spring.

Ahora se trabaja con la anotación **@Component** para especificar que es una clase perteneciente a validaciones.



```
19  @Component
20  public class PasanteValidator implements Validator {
21
22      @Autowired
23      private PasanteService pasanteService;
24
25      public void setPasanteService(PasanteService pasanteService) {
26          this.pasanteService = pasanteService;
27      }
28
29      @SuppressWarnings("rawtypes")
30      public boolean supports(Class clazz) {
31          return Pasante.class.isAssignableFrom(clazz);
32      }
33
34      public void validate(Object object, Errors errors) {
35          Pasante pasante = (Pasante) object;
36          validateApellidos(pasante, errors);
37          validateCelular(pasante, errors);
38          validateDescripcion(pasante, errors);
39          validateEmail(pasante, errors);
40          validateNombres(pasante, errors);
41          validateTelefono(pasante, errors);
42      }
43
44      private void validateNombres(Pasante pasante, Errors errors) {
45          if (pasante.getPasNombres() == null
46                  || StringUtils.isEmpty(pasante.getPasNombres())) {
47              errors.rejectValue("pasNombres", "pasante.pasNombres.required");
48          }
```

Figura IV.40. Implementación del validador de una entidad.

4.2.7 FASE DE PRUEBAS

4.2.7.1 Pruebas de Implementación con Usuarios Finales.

Esta fase de la metodología involucra a todos los usuarios, se realizó una presentación previa con los usuarios finales para detectar errores, para constancia de esta actividad se desarrolla un documento en el cual queda de constancia el desarrollo de la actividad con sus respectivas sugerencias.

4.2.7.2 Análisis de rendimiento

Las pruebas de rendimiento servirán para demostrar que el sistema cumple los criterios de rendimiento. Se pueden medir que partes del sistema o de carga de trabajo provocan que el conjunto rinda mal. Para su diagnóstico, los ingenieros de software utilizan herramientas como pueden ser

monitorizaciones que midan qué partes de un dispositivo o software contribuyen más al mal rendimiento o para establecer niveles del mismo que mantenga un tiempo de respuesta aceptable.

4.2.7.3 Evaluación de Rendimiento

4.2.7.3.1 Actividad 1. Identificar el entorno de prueba.

Con el entorno con el que contamos es con una aplicación en Framework Spring MVC 3.0 en Java al mismo que se le ejecutará las pruebas de rendimiento.

Las herramientas a utilizar son las siguientes:

SOFTWARE:

Tabla IV.XII Herramientas para realizar pruebas de rendimiento

Herramientas	Objetivo
1. JMeter	Apache JMeter una navegación real sobre una aplicación web, herramienta de prueba de carga para analizar y medir el desempeño de una variedad de servicios, y posteriormente poder reproducirla con distintos número de usuarios concurrentes con el fin de obtener información de cómo se comporta nuestra aplicación bajo esas situaciones.
2. AppPerfect	Soporta un amplio conjunto de pruebas y seguimiento de los productos de software, se utilizan para analizar pruebas de stress, carga, web.
3. Monitor de recursos de Windows	El monitor de recursos, es una especie de administrador de tareas avanzado, presenta mucha más información sobre la actividad del CPU, del Disco Duro, de la Red y de la Memoria, muy útil para saber qué proceso es el que está deteniendo el rendimiento de nuestro equipo cuando una aplicación parece colgarse.
4. LoadUI Pro	Consta de una interfaz de arrastrar y soltar visual, LoadUI le permite crear, configurar y redistribuir sus pruebas de carga interactiva y en tiempo real. En un entorno de prueba única, LoadUI ofrece cobertura de la prueba completa y compatible con todos los protocolos y tecnologías estándar. Y es tan poderoso, que genera carga escalable, de alto volumen y de la vida real de cualquier número de equipos locales y remotos. Conocé LoadUI, el futuro de los servicios de pruebas de carga web.
5. Firebug	Extensión de Firefox creada y diseñada especialmente para desarrolladores y programadores web. Es un paquete de utilidades con el que se puede analizar (revisar velocidad de carga, estructura DOM), editar, monitorizar y depurar el código fuente, CSS, HTML y JavaScript de una página web de manera instantánea e inline.
6. Badboy	Badboy integra con JMeter, ya que permite guardar las secuencias de

Herramientas	Objetivo
	comandos en el formato de archivo JMeter para que pueda abrirlos y ejecutarlos en JMeter. Este apoyo significa que se puede utilizar Badboy para grabar las secuencias de comandos para realizar pruebas funcionales y, a continuación, guarde la misma secuencia de comandos en un archivo JMeter.
7. LoadImpact	Load Impact es un servicio online que te permite testear, y por lo canto conocer, la capacidad que tiene tu sitio para soportar picos de demanda mucho más altos que lo habitual.

HARDWARE:

Equipo que se utilizó para realizar las pruebas de rendimiento.

Tabla IV.XIII. Características PC

CARACTERÍSTICAS	DESCRIPCIÓN
Modelo	Hp 520-1160la
Procesador	Intel Core i5-2400S CPU 2.50 GHZ
Memoria RAM	8GB
Sistema Operativo	64 bits

4.2.7.3.2 Actividad 2. Identificar los criterios de rendimiento de aceptación.

Tabla IV.XIV. Parámetros de rendimiento.

PARÁMETRO CONCEPTO	DESCRIPCIÓN
1Infraestructura de hardware	Esta es una prueba que mide la cantidad de recursos utilizado por la aplicación como la memoria, ancho de banda, latencia, etc. El porcentaje del tiempo de procesador consumido. Cuando se compara el rendimiento bajo carga estándar con una instantánea.
2Pruebas de carga transaccional alta	Una prueba de carga se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperada. Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga. Esta prueba puede mostrar los tiempos de respuesta de todas las transacciones importantes de la aplicación.
3 Pruebas de estrés	Esta prueba se utiliza normalmente para romper la aplicación. Se va doblando el número de usuarios que se agregan a la aplicación y se ejecuta una prueba de carga hasta que se rompe. Este tipo de prueba se realiza para determinar la solidez de la aplicación en los momentos de carga extrema y ayuda a los administradores para determinar si la aplicación

PARÁMETRO CONCEPTO	DESCRIPCIÓN
	rendirá lo suficiente en caso de que la carga real supere a la carga esperada.

Tabla IV.XV. Índices de los parámetros de rendimiento.

Parámetro	Índice
Infraestructura de hardware	CPU %
	Memoria %
	Ancho de banda
	Latencia
	Disco Duro
	Red
Pruebas de carga transaccional alta	Ingreso de carga transaccional
	Duración de carga de la aplicación en la web
	Prueba de Carga General
Pruebas de estrés	Solidez de concurrencia

4.2.7.3.3 Actividad 3. Pruebas del Plan y de diseño.

4.2.7.3.3.1 Infraestructura de hardware

Tabla IV.XVI. Parámetro 1 de análisis.

Parámetro	Índice	Descripción
Infraestructura de hardware	Índice 1CPU %	El porcentaje del tiempo de uso del procesador consumido por Spring MVC.
	Índice 2 Memoria %	La cantidad de memoria que se usa activamente por Spring MVC
	Índice 3 Ancho de banda	Describe el total de los datos de tasa de transferencia (en el motor de carga).
	Índice 4 Latencia	Esta variable proporciona la información de la cantidad de un usuario espera respuesta de la página antes de tomar la siguiente acción que podría ser el abandono de aplicación, recarga de la página, etc. Hoy en promedio un usuario espera durante 3 segundos para una página web se cargue antes de tomar cualquier acción.
	Índice 5 Interacción de velocidad	Velocidad de interacción representa la velocidad de la interacción del usuario con las aplicaciones. Esta variable representa la rapidez con que un usuario realizar acciones en una página web y navega por las páginas diferentes.

4.2.7.3.3.2 Pruebas de carga transaccional alta

Describe el tiempo de las cargas transaccionales altas, dependiendo del número de cargas.

Tabla IV.XVII. Parámetro 2 de análisis.

Parámetro	Índice	Descripción
Pruebas de carga transaccional alta	Índice 1 Ingreso de carga transaccional	Cargas transaccional altas por usuario de prueba.
	Índice 2 Duración de carga de la aplicación en la web.	Identifica el tiempo en que muestra o tarda en cargarse la aplicación en el explorador.
	Índice 3 Prueba de Carga General	Evalúa la carga de toda la aplicación web.

4.2.7.3.3.3 Pruebas de estrés

Tabla IV.XVIII. Parámetro 3 de análisis.

Parámetro	Índice	Descripción
Pruebas de estrés	Índice 1 Solidez de concurrencia	Se refiere a la resistencia de la aplicación de soportar una carga determinada de usuarios virtuales.

4.2.7.3.4 Actividad 4. Configurar el entorno de prueba.

En este entorno de pruebas se utiliza recursos como un computador de mesa y herramientas software descritas anteriormente para la simulación de la pruebas de rendimiento de la aplicación.

4.2.7.3.5 Actividad 5. Implementar el Diseño de los ensayos

4.2.7.3.5.1 Infraestructura de hardware

Este parámetro se medirá los recursos consumidos por la aplicación en el que se encuentra hospedada la aplicación, los instrumentos de medición son aplicaciones de software tales como JMeter, AppPerfect, Monitor de recursos de Windows, LoadUI Pro, Firebug, Badboy, LoadImpact, los mismo que nos brindar diferentes interpretaciones del rendimiento de aplicaciones web.

4.2.7.3.5.2 Pruebas de cargas transaccionales

Este parámetro se medirá las cargas transaccionales altas como son (insertar / editar / búsqueda). Así como también se encarga de evaluar la carga de la aplicación completa donde se podrán derivar diferentes resultados para cada una de las páginas de la aplicación.

4.2.7.3.5.3 Pruebas de estrés

Este parámetro permite medir la máxima carga de usuarios que soporta la aplicación lo que simula la eficiencia de la aplicación.

4.2.7.3.6 Actividad 6. Ejecutar la prueba.

4.2.7.3.6.1 Parametro1 Infraestructura de hardware

Los cálculos se han obtenido con las diferentes herramientas de software.

4.2.7.3.6.1.1 Análisis Índice1, Índice 2, Índice 3, Índice 4

Admistrador de recursos de windows

Aquí se captura los siguentes recursos cpu, memoria, disco, red.

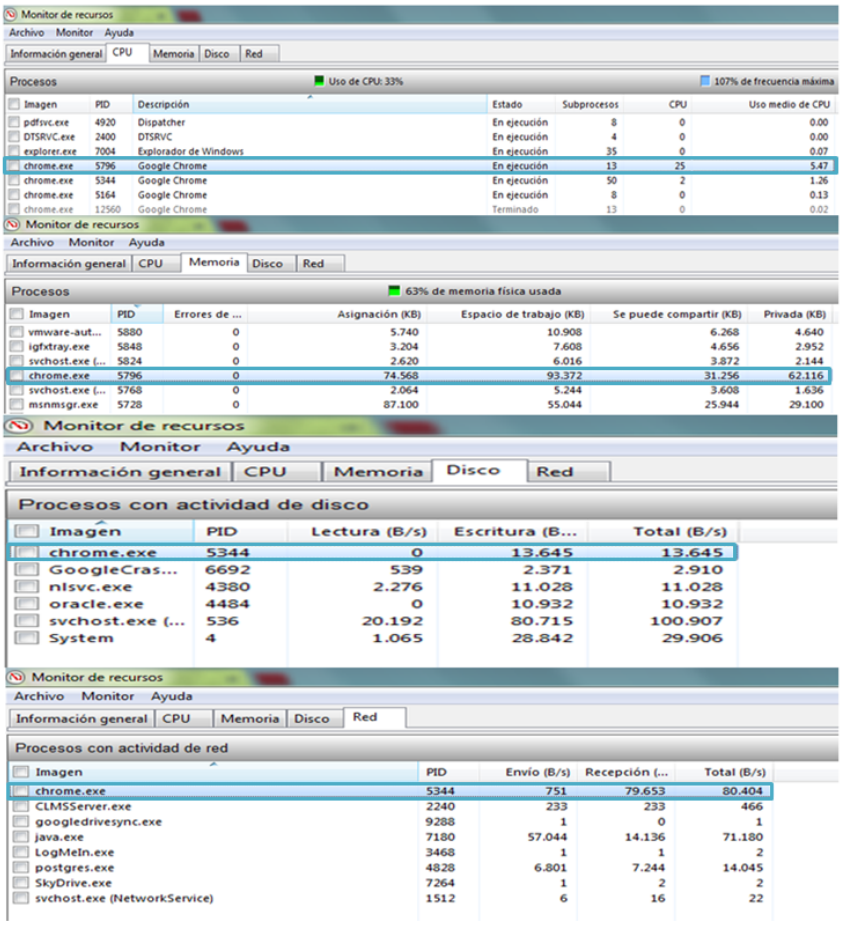


Figura IV.41. Monitor de recursos utilizados por cada una de las aplicaciones.

El la Figura IV.41 Se aprecia el resumen que arroja el monitor de recursos de Windows con parámetros mucho más confiables como se puede observar en la figura que facilita el análisis y la toma de sesiones, para la toma de decisiones, a su vez estos parámetros se resumen en la siguiente tabla.

Tabla IV.XIX. Resumen de resultados del Monitor de Recursos de Windows

Resumen de Recursos Consumidos	
Memoria	74.568 kb
CPU	2% - 5%
Disco	13.645 B/s
Red	Ex 751 B/s
	Rx 97.653 B/s

Como se muestra en la Tabla V.XVIIIse puede captar los datos reales que nos brinda el monitor de recursos como se presenta en la tabla.

4.2.7.3.6.1.2 Análisis Índice 5

ADSL ZONE

Nos permite conocer la latencia que se genera y es medida en milisegundos y se denomina como tiempo de ping. Representa el retraso entre al solicitar algo desde Internet y el punto en el cual lo recibe, en la Figura IV.42 se peude observar el resultado arrojado por la aplicación.

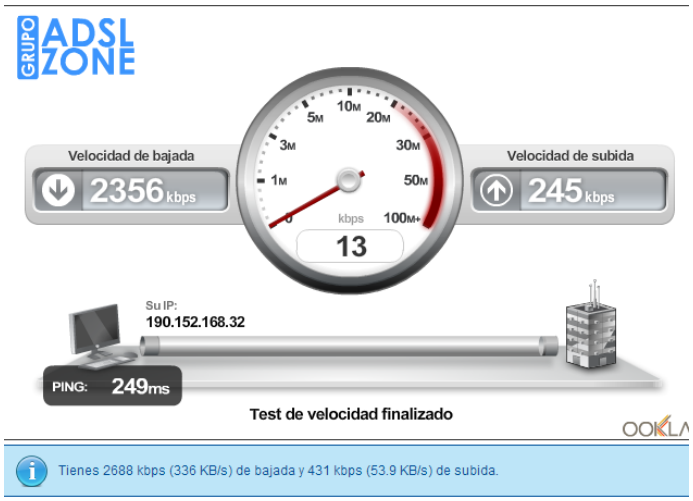


Figura IV.42. Latencia de red

Aquí se presenta en resumen los datos capturados en las herramientas, para definir los índices, resumiendo a datos reales de infraestructura.

Tabla IV.XX.Resumen datos capturados de infraestructura.

Índices		**Monitor de recursos**	**ADSL ZONE**	**Resumen Final**
CPU %		2% - 5%	-	**2% - 5%**
Memoria		74.568 kb	-	**74.568 kb**
Ancho de banda		1655.2 Kbit/s	-	**1655.2 Kbit/s**
Latencia de red		-	249 ms	**249 ms**
Disco		13.645 B/s	-	**13.645 B/s**
Red	Envió	751 B/s	-	**751 B/s**
	Recepción	97.653 B/s	-	**97.653 B/s**

Como puede observar el Tabla IV.XX en resumen de todos los datos que se puede evidenciar en las herramientas utilizadas, se finaliza con conjunto de datos como resumen, como es en CPU se puede verificar que la aplicación utiliza un 5%, memoria 74.568 kb, ancho de banda 1655.2 Kbit/s, en si la latencia en un valor menor 249 ms, en disco utiliza un pequeño espacio de 13.645 B/s, en RED se tiene para él envió 751 B/s y para la recepción 97.653 B/s.

4.2.7.3.6.2 Parámetro 2 Pruebas de cargas transaccionales

4.2.7.3.6.2.1 Índice 1 Pruebas de cargas transaccionales altas

Test de carga de una de las entidades de la aplicación, el objetivo es obtener los tiempos y analizar si son los ideales.

Datos capturados desde la base de datos postgresql para analizar la relación con los tiempos ejecutados atreves de la aplicación.[5]

4.2.7.3.6.2.1.1 PostgreSQL

Insert Usuario

Query returned successfully: 1 row affected, 62 ms execution time.

Consulta Usuario

Total query runtime: 12 ms. 4 rows retrieved

4.2.7.3.6.2.1.2 Hibernate

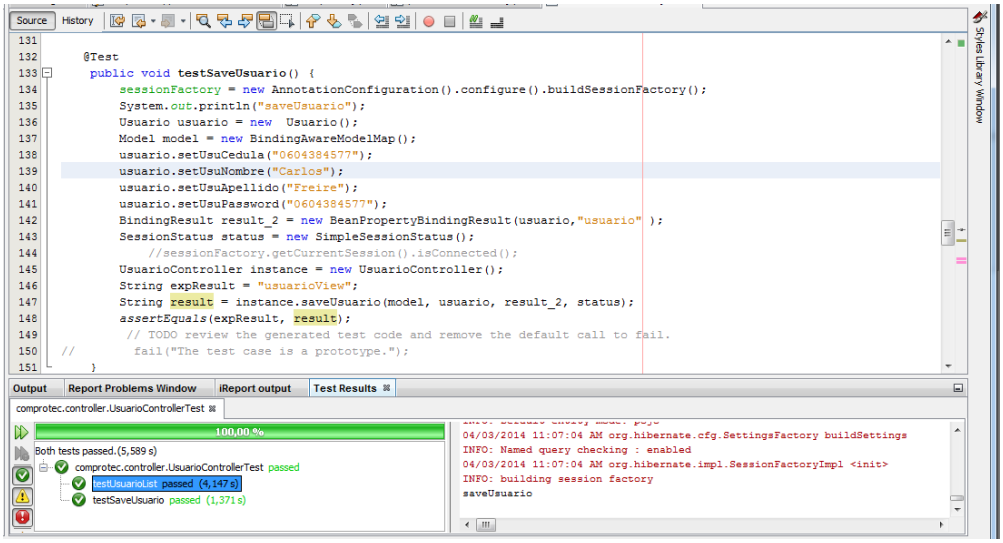


Figura IV.43. Test de carga de ingreso de un Usuario

En la Figura IV.43 se puede observar claramente los datos arrojados en la prueba de carga de una inserción de un nuevo usuario, resultando así un tiempo de 1.371 segundos lo cual es un excelente tiempo, y la una consulta del listado de los usuario con un tiempo de 4,147 segundos, como podemos ambos test han pasado la prueba.

4.2.7.3.6.2.1.3 Interpretación de resultados

Se procede analizar los datos obtenidos del Postgresql e Hibernate y tabular, donde se presenta en resumen los totales de tiempos transcurridos en la ejecución de las transacciones en sus diferentes ambientes de ejecución.

Tabla IV.XXI.Resumen cargas altas.

Parámetros	Save (s)	List (s)	Total (s)
Postgresql	0.062	0.012	0.074
Hibernate	1.371	4.147	5.518

En la Tabla IV.XXI se observa que la ejecución directa de los métodos de cargas en postresql consta de un tiempo bajo lo cual es lógico, ya que los métodos son ejecutados en mismo ambiente

de trabajo, es asíarrojando Save 0.062 s y List 0.012 s y un tiempo de 0.074 s el total para los dos métodos, para el ambiente desde la aplicación mediante Hibernate, cabe indicar que se lo realiza de una página web los resultados tiene un incremento considerable en tiempo de ejecución, es así que arroja para el Save 1.371 s y para el List 4.147 s con total de ejecución de los dos métodos de 5.518 s, como se puede observar los tiempos obtenidos en el ambiente Hibernate son notablemente considerados como adecuados, cabe indicar que los resultados se han obtenido desde un servidor de prueba cuyos resultados prestarán mejores resultados de un ambiente de producción de la aplicación.

4.2.7.3.6.2.2 Índice 2 Duración de carga de la aplicación en la web.

Tabla IV.XXII. Nivel de aceptación de la carga de una página web

Cuantitativa	0-3 s	3-5 s	5-8 s
Cualitativa	Alta	Medio	Bajo
Porcentajes	100%	70%	30%

Firebug

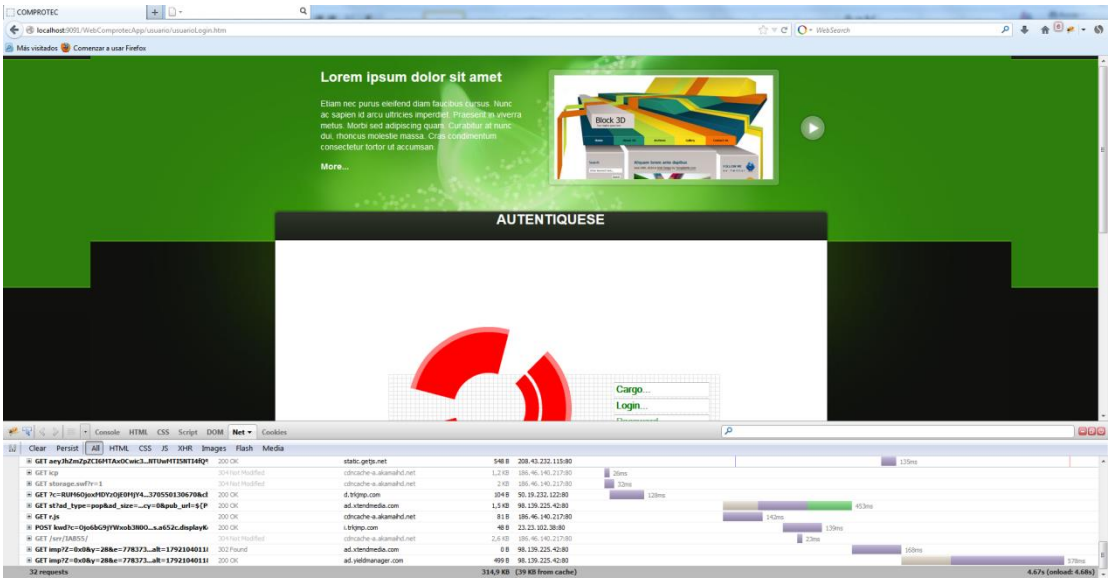


Figura IV.44. Medición de carga con Firebug.

Como se puede observar en la Figura IV.44. Medición de carga con Firebug. Tamaño de cada uno de los recursos cargados el tiempo que conlleva la carga de cada uno de los recursos.Recursos cargados 32 requests o peticiones. Tamaño 314,9 kb presenta el espacio utilizado en disco utilizado en la carga.

Tiempo 4,67 se presenta el tiempo que se tarda para realizar la carga de los archivos para visualizar la aplicación, que se encuentra dentro de los valores de aceptación medio.

4.2.7.3.6.2.3 Índice 3 Pruebas de carga general

4.2.7.3.6.2.3.1 LoadUI PRO

Entorno configurado para un límite de 8000 peticiones, con un splitter para simular una carga real, con unos retardos entre las páginas 10000, además está configurado para que escanea 100 fallas o conocido como assertion (Afirmaciones) y finalice la carga.

Además se puede ver otros resultados por cada uno de los Web Page Runner como son los siguientes parámetros que sea menciona en la siguiente tabla.

Tabla IV.XXIII. Significado de los parámetros de LoadUI.

Opción	Descripción
AVG	El tiempo promedio de la etapa ha tomado (en milisegundos).
MIN	El tiempo más corto de la etapa ha tomado (en milisegundos).
MAX	El tiempo más largo de la etapa ha tomado (en milisegundos).
STD-DEV	Muestra la desviación estándar de las peticiones ejecutadas.
CNT	La etapa de prueba se ha ejecutado el número de veces.
ERR	El número de errores de aserción de la etapa de prueba.
RATIO	Relación de solicitudes con error (el porcentaje de solicitudes que no se ha ejecutado).

En la Tabla IV.XXIII presenta los parámetros que permite medir en el LoadUI para facilitar la toma de decisiones en el análisis.

Web page runner

Representa una página dado su url.

Agregación de un Splitter

Dado que esto no es realista (la mayoría de los usuarios no realizan dos actividades paralelas al mismo tiempo) que deberías añadir un componente divisor Eficaz el primer Página web

Agregación de un Delay

El componente divisor envía los mensajes entrantes a las salidas configuradas de forma secuencial o al azar, por lo que ahora nuestro escenario aleatoriamente va a la izquierda y a la derecha del flujo de páginas efectivas a la primera página.

Se dirige a la parte superior de esta añadiendo un retardo intermediarios últimos dos corredores página web (ya que los usuarios suelen pasar algún tiempo en una página web antes de hacer clic en adelante).

Aquí se ha establecido un retardo de 10 000 ms mediante la distribución de Gauss, que da una simulación realista del comportamiento de los usuarios (el retraso probablemente será más largo).

Fixed rate

Permite configurar una carga fija de peticiones por segundo.

4.2.7.3.6.2.3.1.1 Primer escenario

Tasa Fija de 35 peticiones /segundo

Summary for Load

TIME	REQUESTS	ASSERTION FAILURES	STATUS
00:01:21	3136	100	Failed

Execution Data

Duration	00:01:21
Start Time	15:57:57
End Time	15:59:18
Total number of requests	3136
Total number of failed requests	100
Total number of assertions	5
Total number of failed assertions	0

Execution Metrics

Assertion Failure Ratio	0%
Request Failure Ratio	3%

Runners

NAME	CNT	MIN	MAX	AVG	STD-DEV	MIN/AVG	MAX/AVG	ERR	RATIO
Web Page Runner	383	400	16191	13296	11672809,70	0,03	1,22	0	0%
Web Page Runner	581	3	9087	5876	5511316,37	0,00	1,55	0	0%
Web Page Runner	569	463	16773	12190	18502856,34	0,04	1,38	0	0%
Web Page Runner	446	164	8751	6641	2418970,99	0,02	1,32	0	0%
Web Page Runner	1257	3	8754	5408	6055051,59	0,00	1,62	100	7%

Assertions

NAME	CONSTRAINT	FAILURES
Assertion 6	Range 0 - 100	0

Figura IV.45. Tabla de resumen para una tasa fija de 35 peticiones por segundo.

Como se puede observar en la Figura IV.45 la carga ha durado 1:21 ya que se ha encontrado las 100 fallas, y ha llegado a tan solo las 3136 peticiones, presentando un estado de fallo en la prueba de carga, además se puede ver un 7% de error generado.

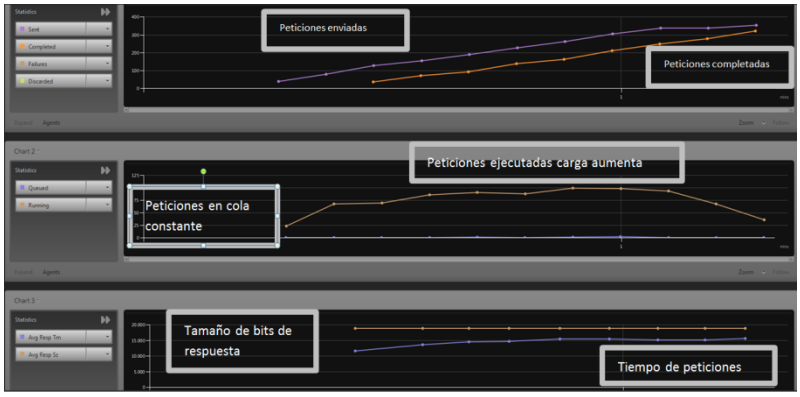


Figura IV.46. Resumen gráfico para una tasa fija de 35 peticiones por segundo.

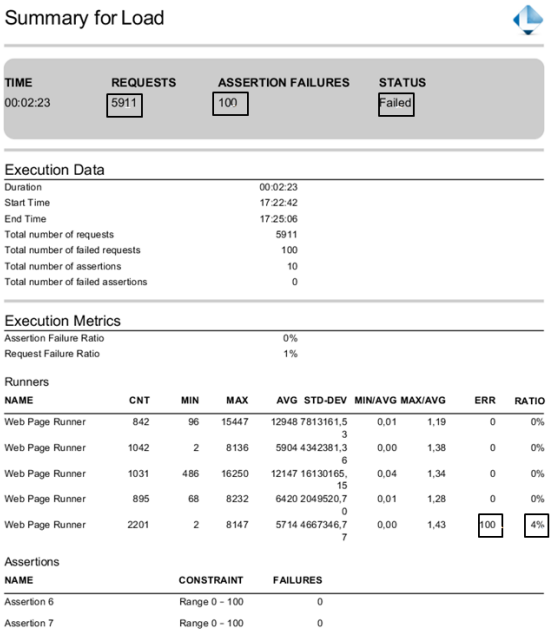
En la Figura IV.46 se puede observar que no se han ejecutado todas las peticiones no sobrepasa ni las 4000 peticiones es lo que es el 50 % de las peticiones indicadas en la preparación del entorno.

Además se puede deducir que las peticiones en cola son constantes, y que mientras la carga aumenta, se aprecia una baja en el rendimiento ya que se va aumentando las peticiones en ejecución.

También se puede observar que el tamaño de bits de respuesta es uniforme mediante toda la ejecución y el tiempo tomado para las peticiones es tiende a subir, a más peticiones.

4.2.7.3.6.2.3.1.2 Segundo Escenario

Tasa Fija de 25 peticiones /segundo



Summary for Load

TIME	REQUESTS	ASSERTION FAILURES	STATUS
00:02:23	5911	100	Failed

Execution Data

Duration	00:02:23
Start Time	17:22:42
End Time	17:25:06
Total number of requests	5911
Total number of failed requests	100
Total number of assertions	10
Total number of failed assertions	0

Execution Metrics

Assertion Failure Ratio	0%
Request Failure Ratio	1%

Runners

NAME	CNT	MIN	MAX	AVG	STD-DEV	MIN/AVG	MAX/AVG	ERR	RATIO
Web Page Runner	842	96	15447	12948	7813161,53	0,01	1,19	0	0%
Web Page Runner	1042	2	8136	5904	4342381,36	0,00	1,38	0	0%
Web Page Runner	1031	486	16250	12147	16130165,15	0,04	1,34	0	0%
Web Page Runner	895	68	8232	6420	2049520,70	0,01	1,28	0	0%
Web Page Runner	2201	2	8147	5714	4667346,77	0,00	1,43	100	4%

Assertions

NAME	CONSTRAINT	FAILURES
Assertion 6	Range 0 - 100	0
Assertion 7	Range 0 - 100	0

Figura IV.47. Tabla de resumen para una tasa fija de 25 peticiones por segundo.

Como se puede observar en la Figura IV.47 la carga ha durado 2:23 ya que se ha encontrado las 100 fallas, y ha llegado a tan solo las 5911 peticiones, presentando un estado de fallo en la prueba de carga, además se puede ver un 4% de error generado.

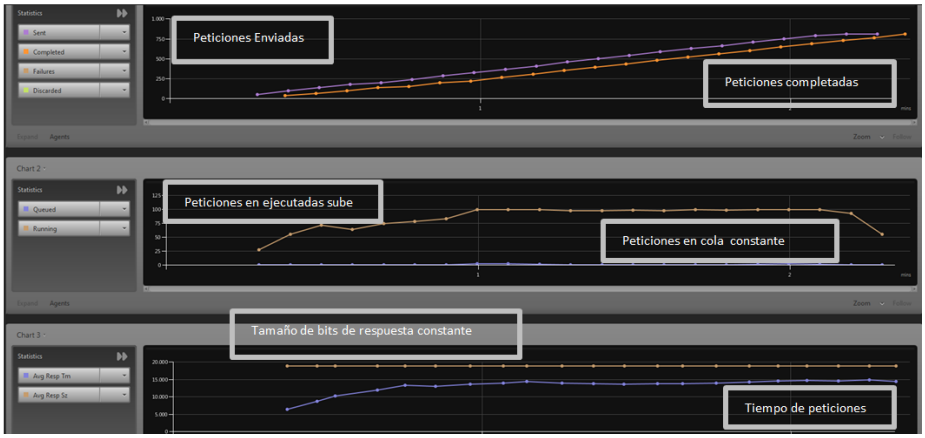


Figura IV.48. Resumen gráfico para una tasa fija de 25 peticiones por segundo.

En la Figura IV.48 se puede observar que no se han ejecutado todas las peticiones indicadas en la preparación del entorno.

Además se puede deducir que las peticiones en cola son constantes, y que mientras la carga aumenta se aprecia una baja en el rendimiento ya que se va aumentando las peticiones en ejecución.

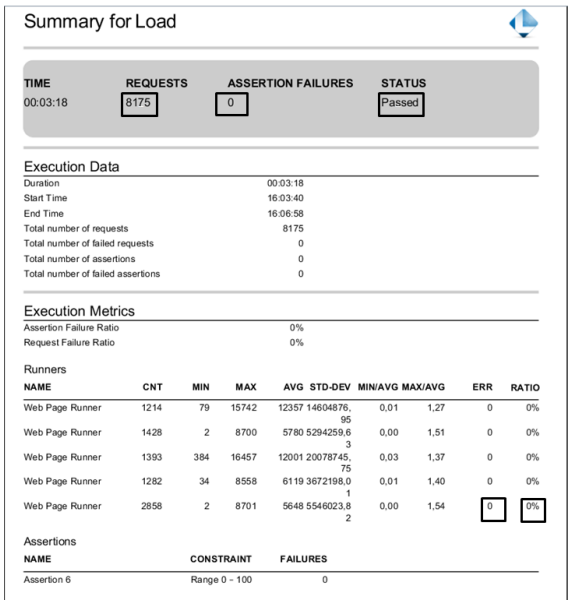
También se observa que el tamaño de bits de respuesta es uniforme mediante toda la ejecución y el tiempo tomado para las peticiones es tiende a subir a más peticiones.

4.2.7.3.6.2.3.1.3 Tercer Escenario

Tasa Fija de 20 peticiones /segundo

Figura IV.49. Interfaz de LoadUI con el diseño de las pruebas.

En la Figura IV.49 presenta el diagrama de pruebas como se observa está compuesta de Web page runner, delay, splitter, fixed rate.



Summary for Load

TIME	REQUESTS	ASSERTION FAILURES	STATUS
00:03:18	8175	0	Passed

Execution Data

Duration	00:03:18
Start Time	16:03:40
End Time	16:06:58
Total number of requests	8175
Total number of failed requests	0
Total number of assertions	0
Total number of failed assertions	0

Execution Metrics

Assertion Failure Ratio	0%
Request Failure Ratio	0%

Runners

NAME	CNT	MIN	MAX	AVG	STD-DEV	MIN/AVG	MAX/AVG	ERR	RATIO
Web Page Runner	1214	79	15742	12357	14604876,95	0,01	1,27	0	0%
Web Page Runner	1428	2	8700	5780	5294259,63	0,00	1,51	0	0%
Web Page Runner	1393	384	16457	12001	20078745,75	0,03	1,37	0	0%
Web Page Runner	1282	34	8558	6119	3672198,01	0,01	1,40	0	0%
Web Page Runner	2858	2	8701	5648	5546023,82	0,00	1,54	0	0%

Assertions

NAME	CONSTRAINT	FAILURES
Assertion 6	Range 0 – 100	0

Figura IV.50. Tabla de resumen para una tasa fija de 20 peticiones por segundo.

Como se puede observar en la Figura IV.50 la carga ha durado 3:18 y no se han encontrado fallas, y se han completado las 8000 peticiones, presentando un estado de ejecutado en la prueba de carga, además se puede ver un 0% de error generado

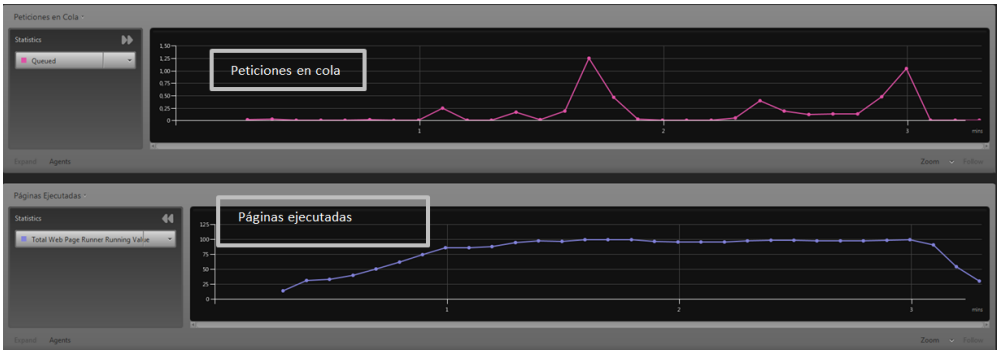


Figura IV.51. Resumen gráfico de cola y páginas ejecutando.

Como se puede visualizar en la Figura IV.51 que las peticiones en cola en su mayor parte son continuas ya hay un poco de picos que sobrecarga el servidor. En cuanto al total de páginas

ejecutadas se aprecia que tiende a subir de acuerdo a las peticiones, luego se mantienes durante la ejecución de todas las peticiones y empieza a bajar finalmente la sobrecarga.

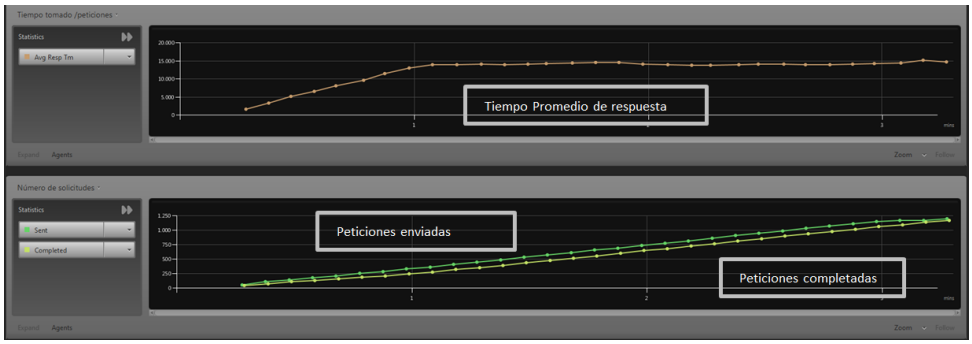


Figura IV.52. Tiempo por segundo y número de solicitudes enviadas y completadas.

Se puede apreciar en la Figura IV.52 que el tiempo promedio de carga de peticiones se mantiene sin sobrecargar el servidor, Y finalmente aquí tenemos que las 8000 peticiones han sido enviadas y completas.

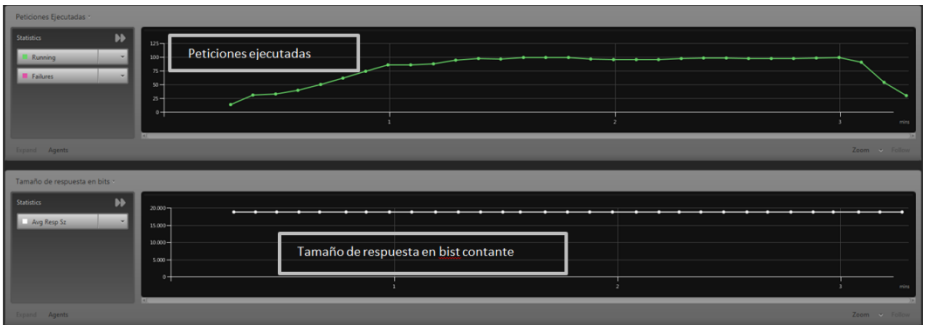


Figura IV.53. Peticiones en ejecución, fallidas y promedio de tamaño de las respuestas en bits

Se puede ver en la Figura IV.53 que no se han obtenido fallas de peticiones, y que al inicio la carga tiende a subir hasta llegar al nivel de mantenerse durante toda la carga y bajar, el promedio de tamaño de respuesta en bits es continuo para toda la carga de peticiones.

4.2.7.3.6.2.3.1.4 Conclusión de la prueba de carga

Tabla IV.XXIV. Tabla de resumen de escenarios en Loadui

Escenario	Tiempo	Fixed rate	Err	Request	Ratio	Status
1	1.21 s	35 p/s	100	3136	7%	Failed
2	2.23 s	25 p/s	100	5911	4%	Failed
3	3.18 s	20 p/s	0	8175	0%	Passed

En conclusión se puede deducir que la carga de la página está dentro del nivel de aceptación ya que se encuentra dentro del rango (3-5) segundos recomendables en la carga de un sitio web.

En cuanto a las pruebas de cargas realizadas con LoadUI se tiene como resultado según la Tabla IV.XXIV para 35 peticiones o usuarios virtuales tomando en cuenta que solo alcanza hasta las 3136 de 8000 con 100 fallas resultando una prueba fallida ya que no supera toda la carga.

Para 25 usuarios virtuales se obtiene 5911 de 800 peticiones con 100 fallas detectadas que la igual no alcanza toda la cargar dando como resultado carga fallida.

Finalmente para 20 usuarios virtuales se obtiene toda la carga de 8175 peticiones con 0 fallas detectadas, lo que indica que ha superado las peticiones de prueba para 20 usuarios, a su vez se puede deducir que la aplicación soporta 20 usuarios, en una carga extrema de 8000 peticiones lo que es justificable par los usuario actuales que tiene la COMPROTEC.

4.2.7.3.6.3 Parámetro 3Pruebas de estrés

El objetivo de probar con esta simulación es obtener el rendimiento de carga de estrés para determinar una carga máxima de usuarios virtuales y así poder tener una perspectiva del rendimiento en la carga de conexiones continuas, herramienta utilizada es Apache JMeter.

4.2.7.3.6.3.1 Índice 1 Solides de concurrencia

Aquí se presenta los cálculos obtenidos con respecto las pruebas de estrés.

Para esta prueba se ha preparado para 100 conexiones virtuales o también conocido como usuarios virtuales, con período de subida de 5 seg, tiempo en que un nuevo usuario ejecuta el escenario, con un contador de 1 que es el número de iteraciones por cada hilo.

Con esta planificación se obtiene que el periodo de carga es igual a la siguiente formula.

$$(\text{Periodo de Carga}) = \frac{(\text{Periodo de Subida})}{\text{Número de hilos}}$$

$$(\text{Periodo de Carga}) = \frac{5}{100}$$

$$(\text{Periodo de Carga}) = 0.05$$

Se puede deducir que cada 0.05 seg se carga un nuevo usuario con lo que se crea una sobre carga de usuarios a la aplicación.

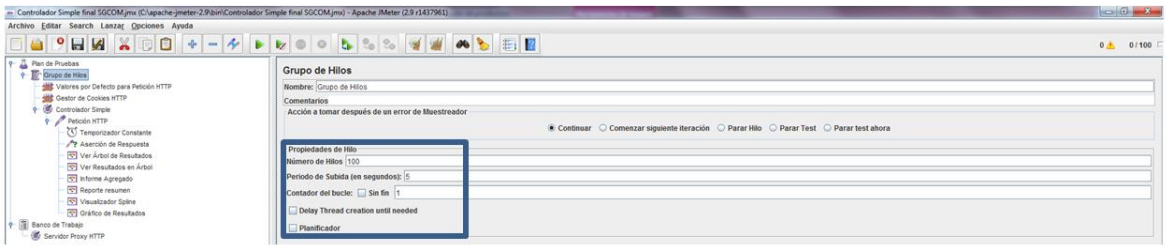


Figura IV.54. Configuración de la carga de estrés

En la Figura IV.54 se muestra la configuración del entorno para la prueba de estrés, con 100 usuarios virtuales, con periodos de subida de 5 seg, contador de bucle de 1.

4.2.7.3.6.3.1.1 Informe agregado

Proporciona información de las estadísticas de la ejecución, número de muestras (solicitudes), tiempos de respuesta de la aplicación medios, mediana, máximo, mínimo, del 90%; errores de solicitudes no resueltas, rendimiento en muestras por segundo, y la cantidad de Kb que el servidor procesa por segundo.

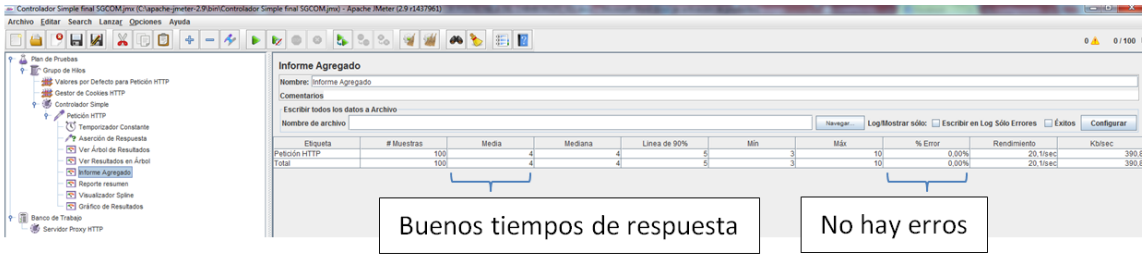


Figura IV.55. Informe agregado

En la Figura IV.55 se puede observar en cuanto a cero errores, con buen tiempo de carga de 4 segundos con un total de 100 peticiones al servidor como se puede observar.

Tiempo utilizado para los 100 hilos:

$$\text{Tiempo total} = \#\text{muestras} * \text{media} = 100 * 4 = 400\text{ms} = 4\text{seg}$$

Hora se puede deducir que el tiempo para cada hilo sería:

$$\text{Tiempo } c_h = \frac{\text{Tiempo total}}{60 * 100} = \frac{400}{60 * 100} = 0.066\text{seg}$$

4.2.7.3.6.3.1.2 Árbol de resultados

Permite observar el resultado de cada uno de los solicitudes realizados por JMeter además de la pantalla (si procede) a la que accedió cada solicitud.

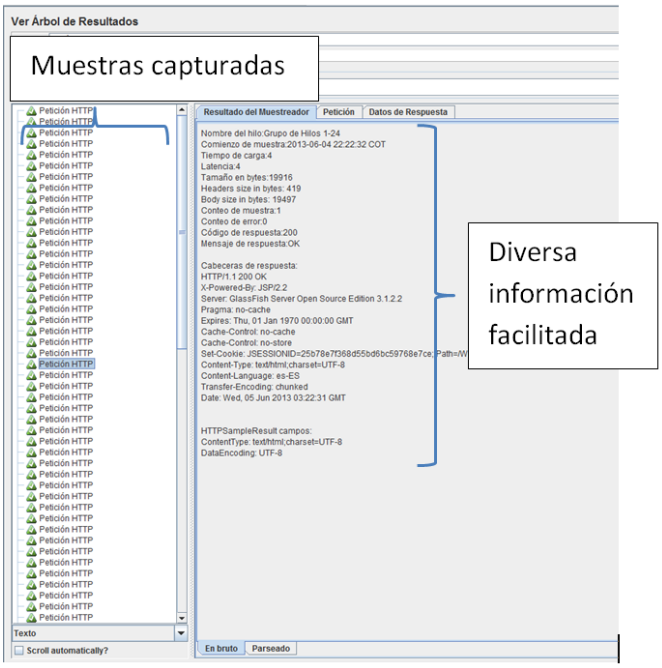


Figura IV.56. Árbol de resultados de rendimiento.

En la Figura IV.56 se puede analizar cada uno de los parámetros de cada petición como se captar fácilmente el tiempo de carga, latencia, tamaño en bytes, mensaje de respuesta entre otros.

4.2.7.3.6.3.1.3 Resultados de árbol

Muestra un árbol con todas las respuestas, indicando muestras, tiempos de respuesta y bytes transmitidos.Aquí presentamos el historial de las 100 conexiones virtuales, donde se puede observar cada una con sus respectivas métricas, como son tiempos, estado satisfactorios en las 100 conexiones, bytes utilizados y latencias generadas.

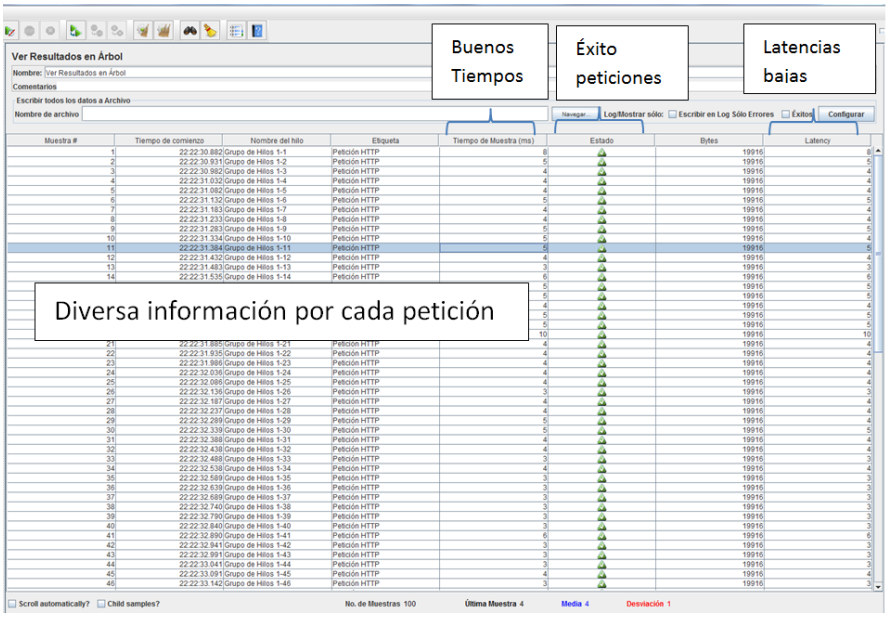


Figura IV.57. Resultado de árbol

En la Figura IV.57 se puede observar que se ha obtenido buenos tiempos en las peticiones que está en el rango de (3-10) ms, con estado satisfactorio de cada petición, como igual se puede observar con latencias bajas.

4.2.7.3.6.3.1.4 Visualizador Spline

Muestra un gráfico de tiempo de respuesta en spline (curva definida en porciones mediante polinomios), indicando el máximo, mínimo y la media.

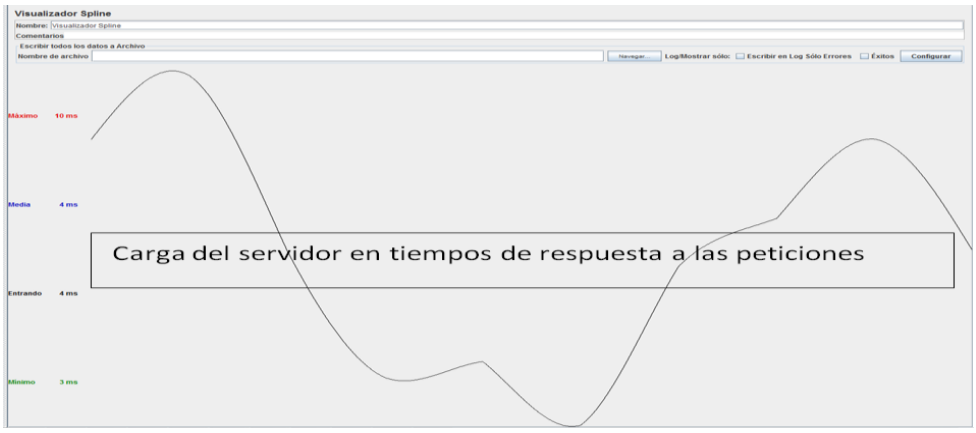


Figura IV.58. Visualizador Spline

En la Figura IV.58 se puede apreciar la muestra que el tiempo de respuesta de solicitudes es errática durante el inicio y, a continuación se establece manteniendo un ritmo medio como se observa en los picos durante todas las peticiones.

4.2.7.3.6.3.1.5 Gráficos de Resultados

Muestra un gráfico con los tiempos de respuesta medios, desviación y mediana, así como el rendimiento en muestras por minuto de las ejecuciones del escenario.

Figura IV.59. Rendimiento del servidor

En la Figura IV.59 se puede visualizar el rendimiento del servidor para las 100 peticiones, con una última muestra de 4 ms, con rendimiento óptimo en como esta en verde de 1205546 minutos, i al igual se tiene una media de 4 ms.

4.2.7.3.6.3.1.6 Interpretación de Resultados

En esta prueba se observa que los tiempos de respuesta de nuestro servidor son muy buenos.

Se resuelven el 100% de las solicitudes, por lo que no se producen errores.

El tiempo promedio de carga de una solicitud es de 0.066 ms.

Se obtuvo el tiempo de carga para las 100 solicitudes es de 4 segundos.

La carga del servidor, como se puede apreciar en la gráfica, es elevada al principio cayendo progresivamente, salvo una pequeña carga al final de las solicitudes que muestra la line a de color verde.

Se puede afirmar que este nivel de carga para este servidor es bajo, Por lo que se resuelven todas las peticiones de cada usuario de forma continua sin entrelazarse las peticiones de otros usuarios.

4.2.7.3.7 Actividad 7. Análisis de resultados, informe y vuelva a probar.

Consolidar y compartir los datos de resultados. Analizar los datos tanto de forma individual como en equipo multifuncional. Repriorize las pruebas restantes y volver a ejecutar cuando sea necesario. Cuando todos los valores de medición se encuentran dentro de los límites aceptados, ninguno de los umbrales fijados han sido violados, y toda la información deseada se ha recogido, usted ha terminado de probar este escenario en particular en esa configuración particular.

4.2.8 FASE DE INSTALACIÓN

4.2.8.1 Implementación del Servidor de Aplicaciones.

Para alojar la aplicación SGCOM, se procedió a instalar el servidor de aplicaciones Glassfish.

Figura IV.60. Instalación del Servidor de Aplicaciones Glassfish

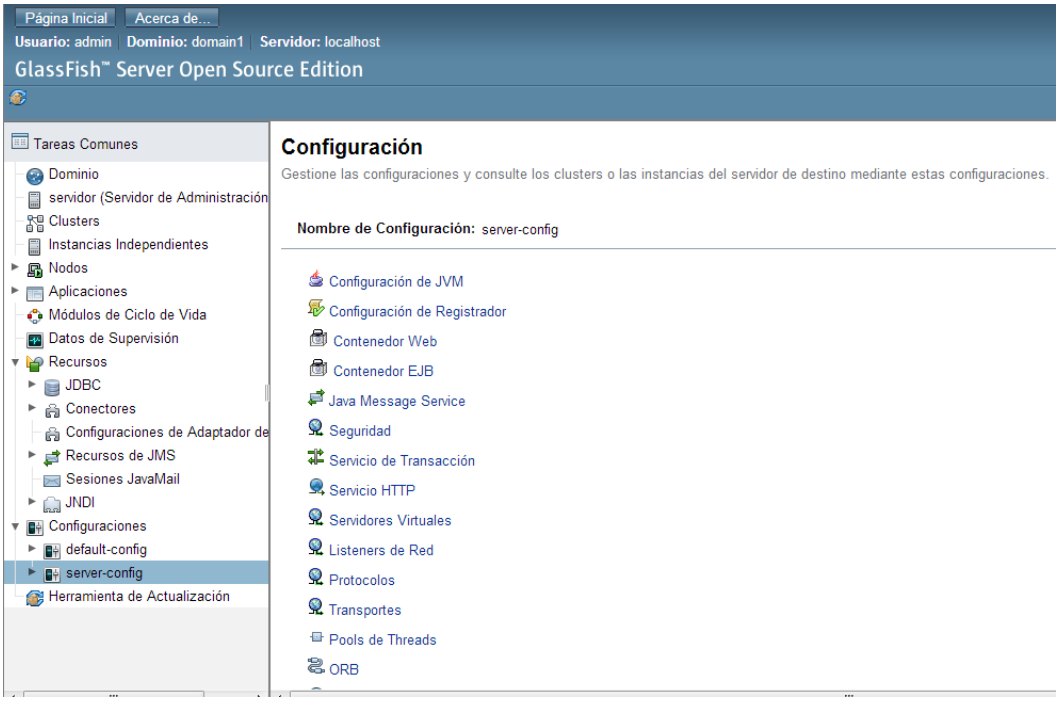


Figura IV.61. Consola de Administración de Glassfish

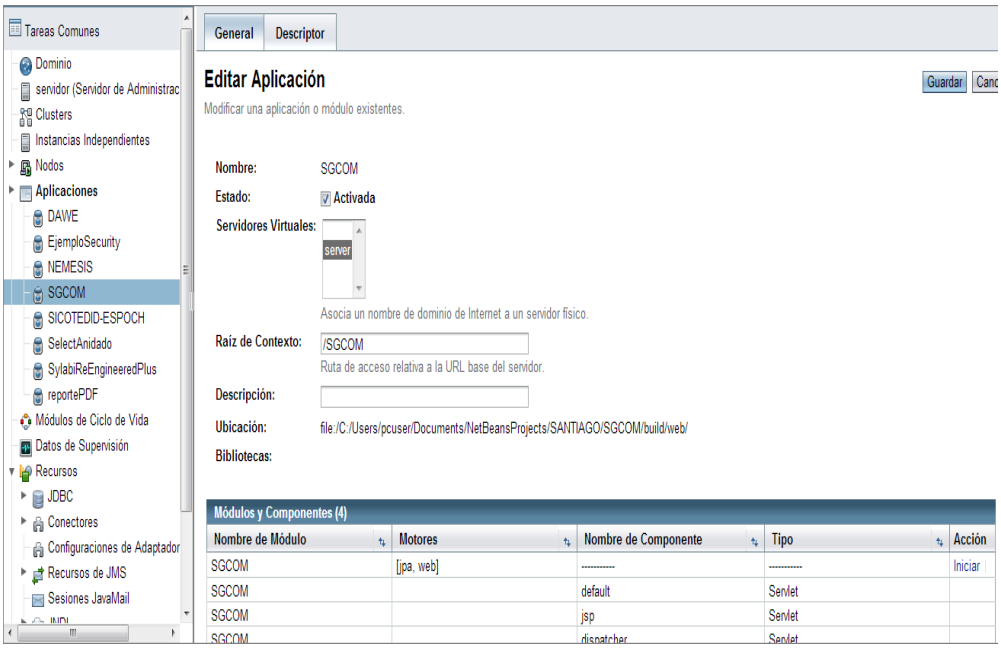


Figura IV.62. Despliegue de la Aplicación Web SGCOM

4.3 DESARROLLO DE MÓDULO JOOMLA

4.3.1 MODULO JOOMLA

La aplicación consta de principalmente con gestión de los siguientes módulos.[8]

- Gestión de artículos.
- Organigrama.
- Calendario de eventos.
- Mapa de sitio.
- Descargas
- Menú multimedia.
- LikeBox Facebook
- Likebox Twitter

4.3.1.1 Portal

El portal está desarrollado para la publicación de contenido de la institución, aquí se presenta la forma de ingreso al módulo de gestión del portal.

Figura IV.63. Login Joomla

Figura IV.64. Menú Principal.

Figura IV.65. Cuerpo del portal.

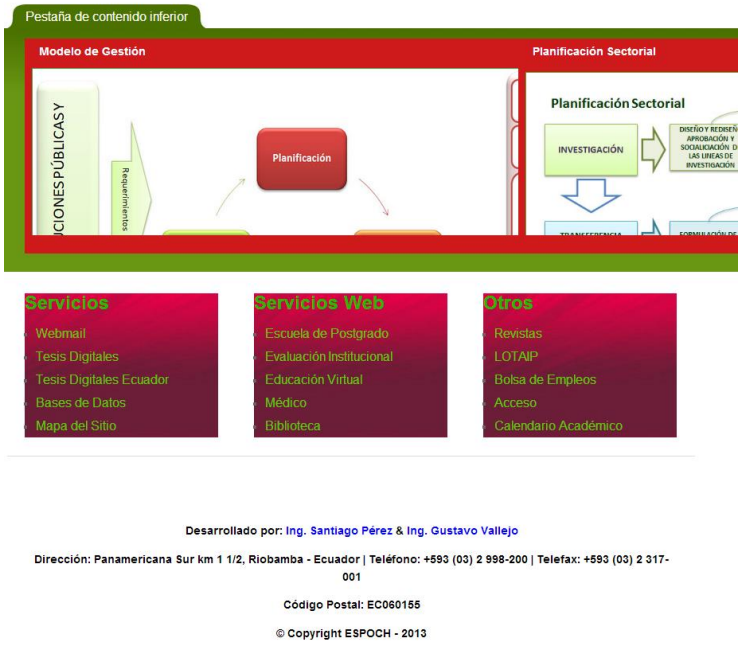


Figura IV.66. Pie del portal.

4.3.1.2 Gestión de artículos.

Este módulo permite al administrador gestionar la publicación de artículos, noticias, etc. con el fin de poner información pública de interés para el usuario. [8]

Ejemplo:

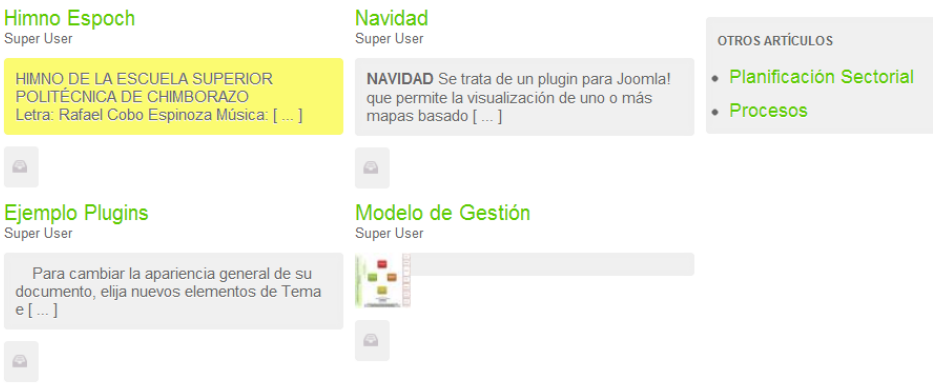


Figura IV.67. Publicación de artículos.

4.3.1.3 Organigrama.

El Organigrama de la institución representa los puestos de trabajo con su respectivo agente.

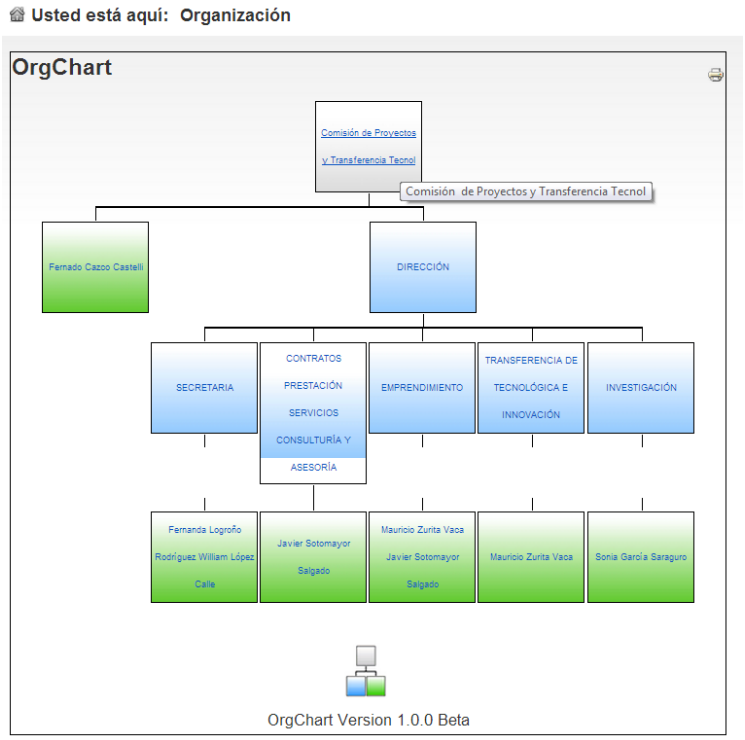


Figura IV.68. Organigrama COMPROTEC.

4.3.1.4 Calendario de eventos.

Administración de eventos para publicar fechas importantes o reuniones etc.

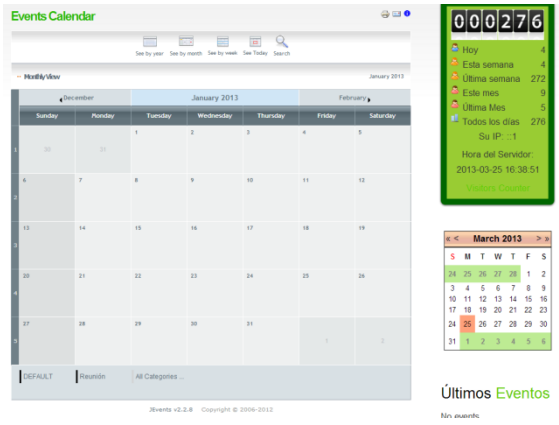


Figura IV.69. Calendario de Eventos.

4.3.1.5 Mapa de sitio.

Es ítem de menú presenta un árbol jerárquico de la estructura del contenido del portal en resumen.

Figura IV.70. Mapa de Sitio.

4.3.1.6 Descargas.

Este ítem permitirá al usuario publicar archivos para descarga de usuarios, tales como formularios donde el usuario puede imprimir o descargarlos, entre otras utilidades.

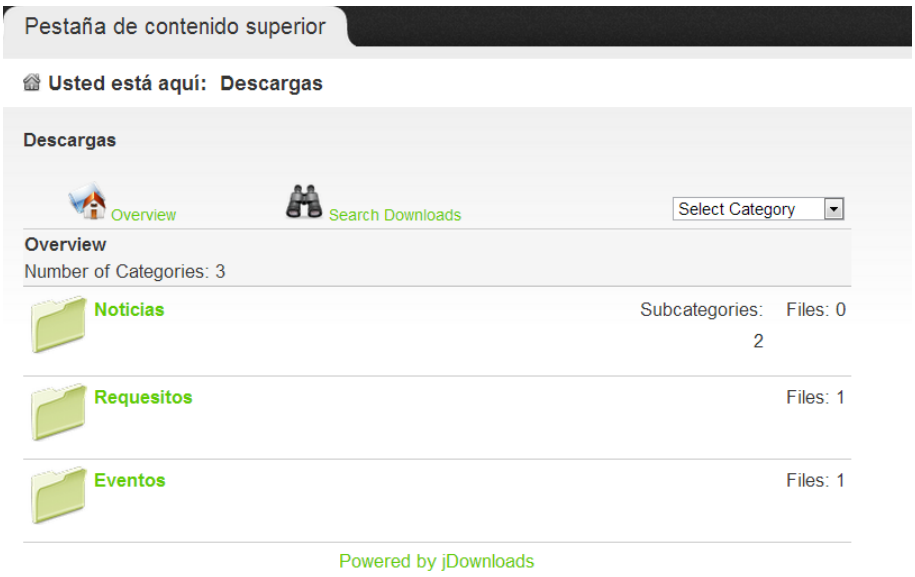


Figura IV.71. Gestión de Archivos.

4.3.1.7 Menú multimedia.

Aquí presenta la galería de video que está vinculado con el canal de videos de http://www.youtube.com/channel/UCYPwVwnQ0_UM51AYyInrknQ de la COMPROTEC .

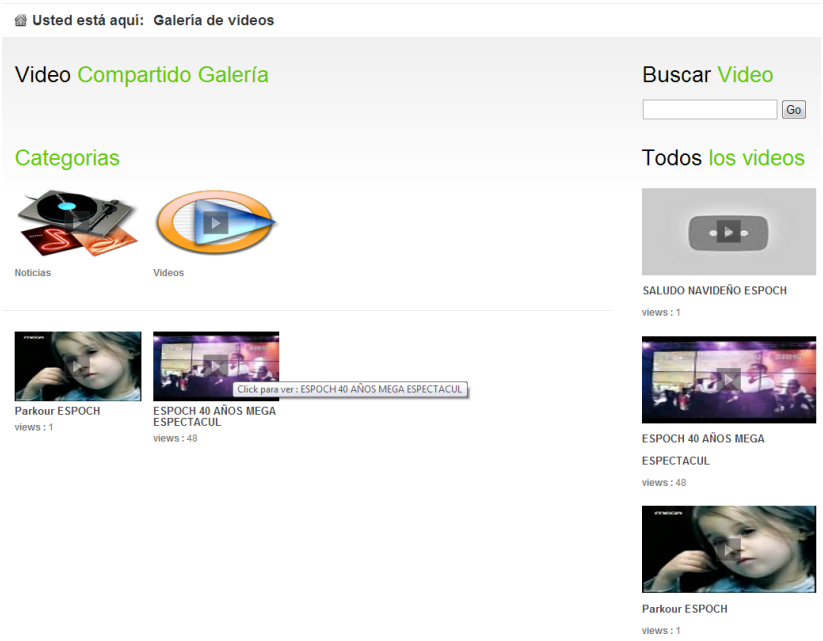


Figura IV.72. Galería de videos.

4.3.1.8 LikeBox Facebook

Permite sincronizar las publicaciones que se tienen en Facebook y su lista de Fans

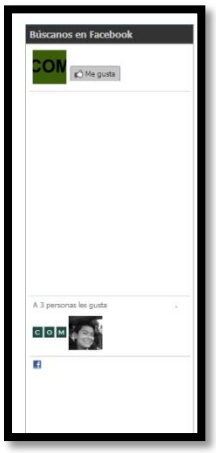


Figura IV.73. LikeBox Facebook

.

4.3.1.9 Likebox Twitter

Permite sincronizar las publicaciones que se tienen en Twitter.

Figura IV.74. LikeBox Twitter

4.4 DESARROLLO DE MÓDULO FRAMEWORK SPRING MVC 3.0

En este apartado se muestra la construcción de la aplicación web para la gestión de la COMPROTEC, la misma que es desarrollada con Java Framework Spring MVC 3.0.

4.4.1 Creación de la aplicación SPRING MVC 3.0

Para crear la aplicación, primeramente se creó una aplicación de tipo Web Application en el IDE NetBeans.[6][7]

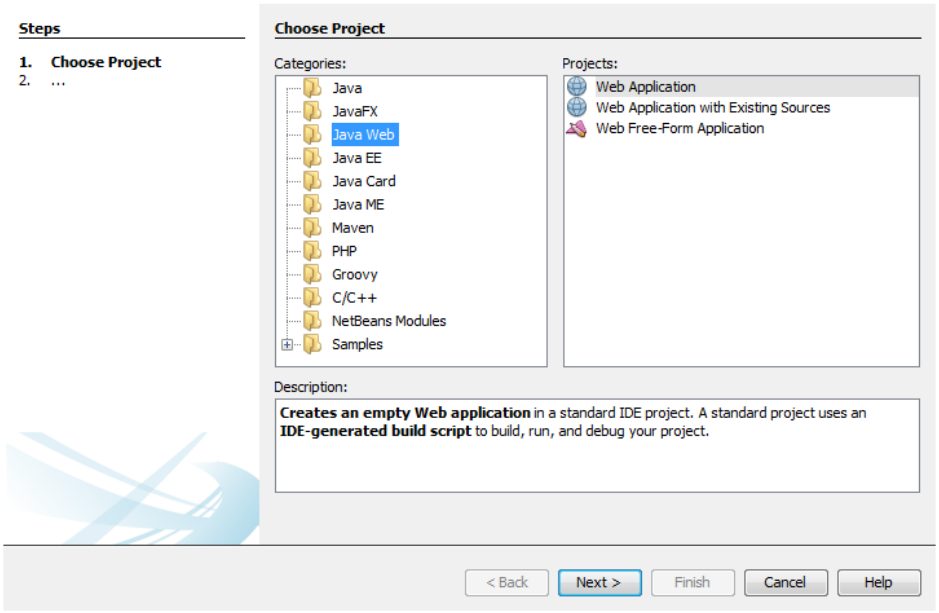


Figura IV.75. Creación de la aplicación web.

Renombrar dicha aplicación en el primer screen, ahora se va al siguiente screen donde se muestras estas configuraciones.

Se escoge el servidor web GlassFish 3.1.1 tomando en cuenta que la instalación del sistemas este servidor puede variar, elegir Java EE 6 Web y finalmente activamos la opción Dependency Injection.

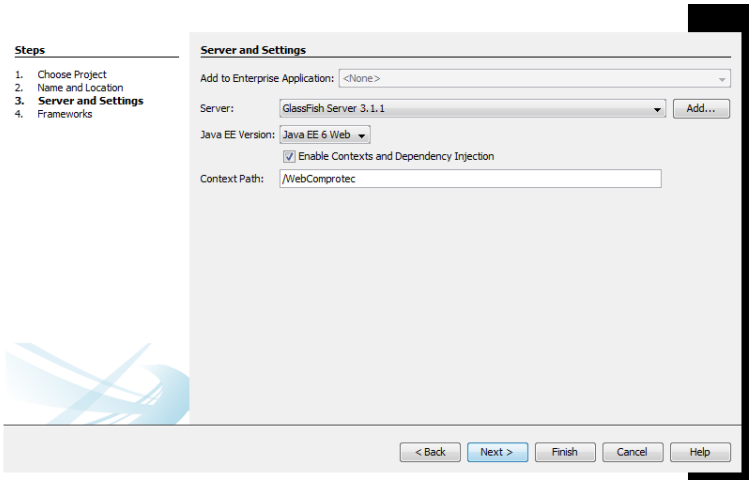


Figura IV.76. Configuración de la aplicación.

Finalmente en la creación de aplicación se elige los Frameworks que vamos a integrar para la creación de la aplicación.Spring Web MVC es el framework principal que se va a utilizar.

Hibernate es el que será el responsable de interactuar con la base de datos PostgreSQL en la creación de la capa Model que corresponde el enlace con el Framework Spring Web MVC 3.0

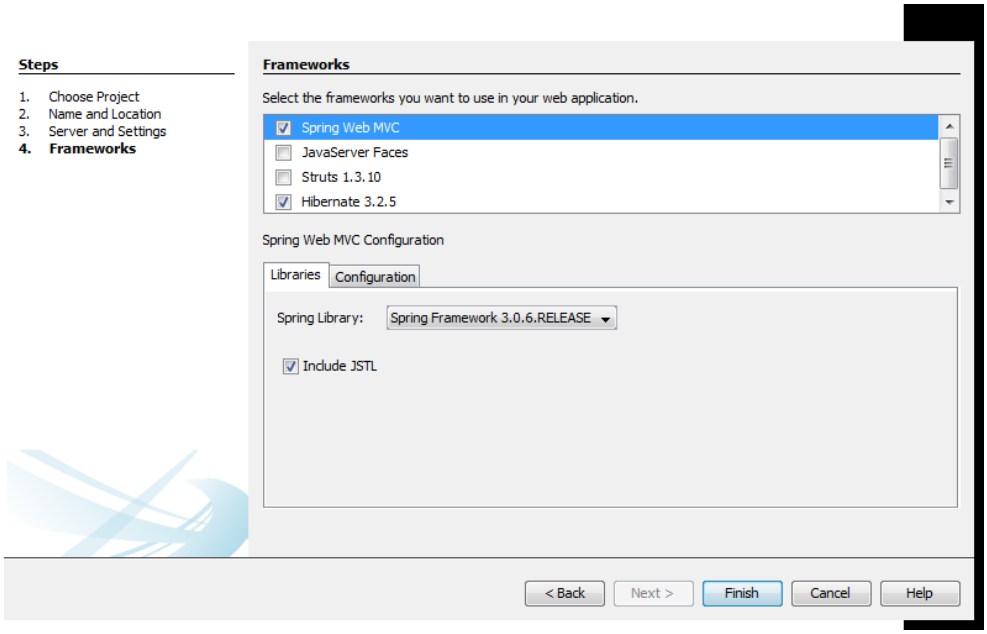


Figura IV.77. Configuración de frameworks de la aplicación.

4.4.2 Acceso a datos

4.4.2.1 HibernateUtil

Aquí se genera la parte del modelo se debe generar la creación del HibernateUtil a que obliga a tener siempre disponible una referencia al objeto SessionFactory para que cualquier clase pueda tener acceso al objeto Session y por lo tanto a todas las funcionalidades de Hibernate, tomando en cuenta que se puede utilizar JDBC, el caso es que ambos permiten el acceso a la base de datos.

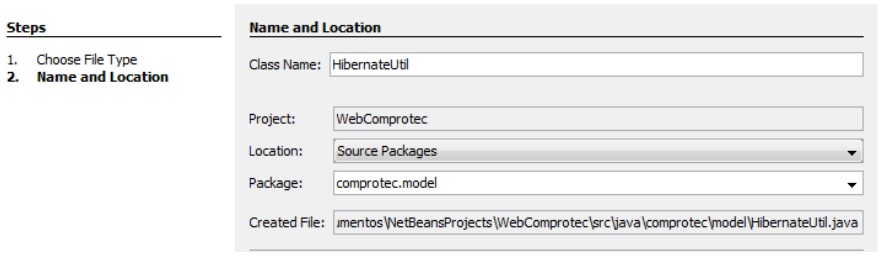
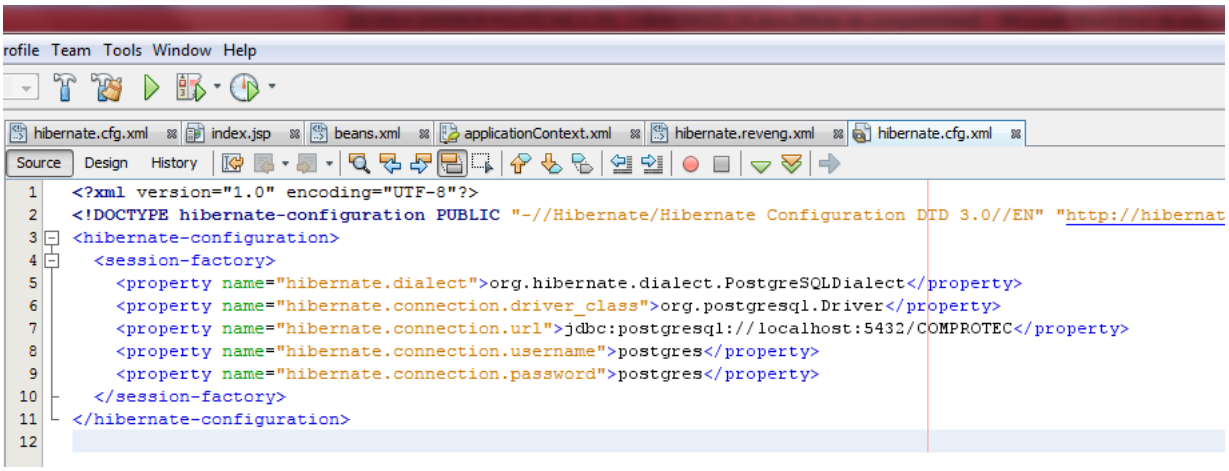


Figura IV.78. Creación del archivo HibernateUtil.

4.4.2.2 Hibernate.cfg.xml

También se necesita un fichero de configuración general de Hibernate.cfg.xml, en el que se almacena información como conectar a la base de datos que se maneja e incluimos todos los ficheros xml que describen el mapeo entre las clases y tablas de la base de datos. Un ejemplo para una base de datos PostgreSQL puede ser el siguiente:



4.4.2.3 Hibernate.reveng.xml

Este archivo permite crea el fichero xml para escoger las tablas que se utiliza en la aplicación.

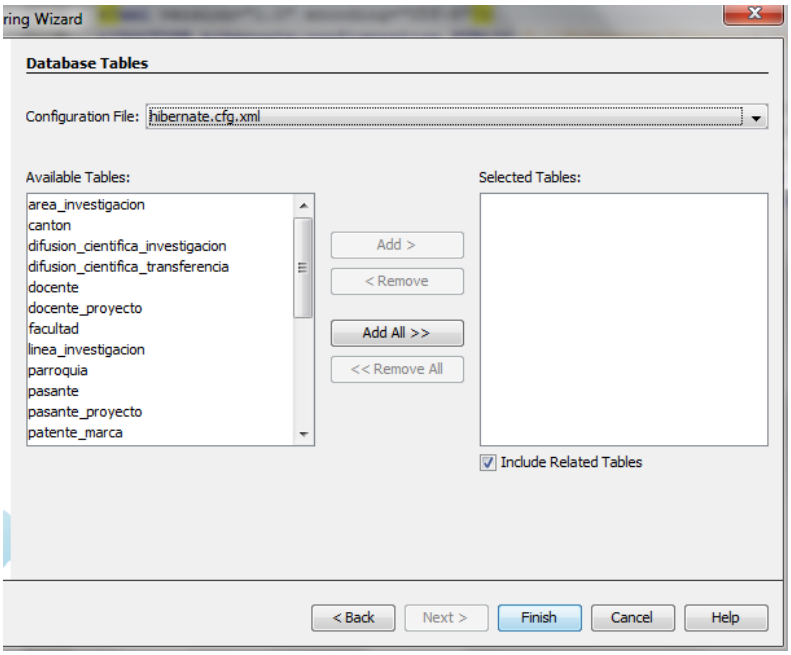


Figura IV.79. Selección de tablas para el Hibernate.cfg

4.4.2.4 Creación de los archivos de mapeo hibernate y pojos

Los archivos de mapeo son archivos XML que contienen datos acerca de cómo las columnas de las tablas se asignan a los campos en los POJOs. Usted necesita tener los archivos hibernate.reveng.xml y hibernate.cfg.xml para utilizar el asistente, con cual se crea la capa MODEL del patrón de Spring Web MVC 3.0.

1. Haga clic en el nodo Source Packages en la ventana de Proyectos y seleccione Nuevo> Otros para abrir el asistente de New File. Seleccionamos Hibernate Mapping Files and POJOs from Database en la categoría de Hibernate. Clic en Siguiente

2. Se debe asegurar de que los archivos hibernate.cfg.xml e hibernate.reveng.xml se seleccionan en las listas desplegables.

3. Verificar que el Código de dominio y las opciones de Hibernate XML Asignaciones están seleccionados. Hibernate POJOS archivos de mapeo con los campos asignados a las columnas

especificadas en hibernate.reveng.xml. El IDE también añade entradas de asignación de hibernate.cfg.xml.

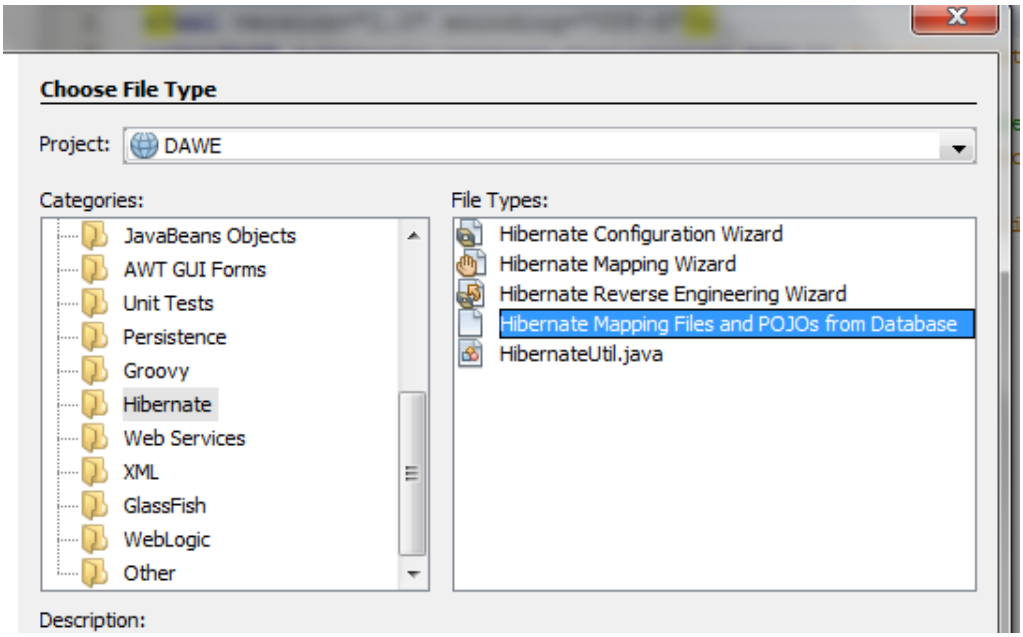


Figura IV.80. Generación del archivo POJOS.

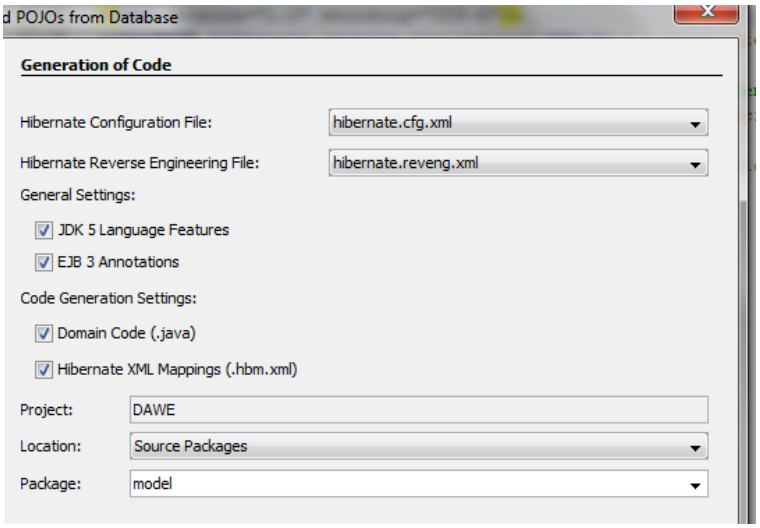


Figura IV.81. Generación del mapeo (POJOS).

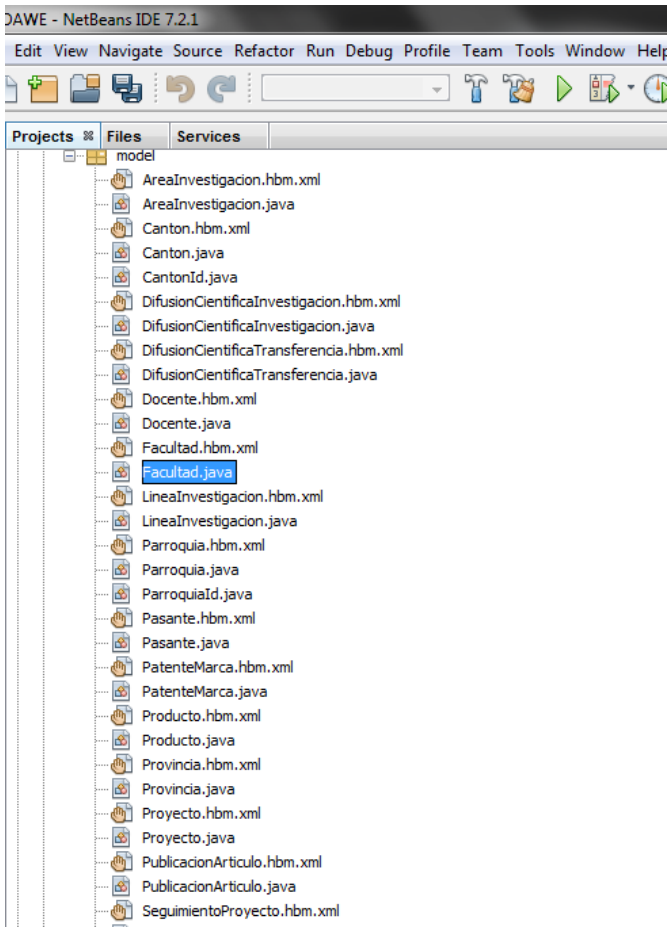


Figura IV.82. Vista del paquete Model Generado.

4.4.3 Paquetes utilizados en la aplicación

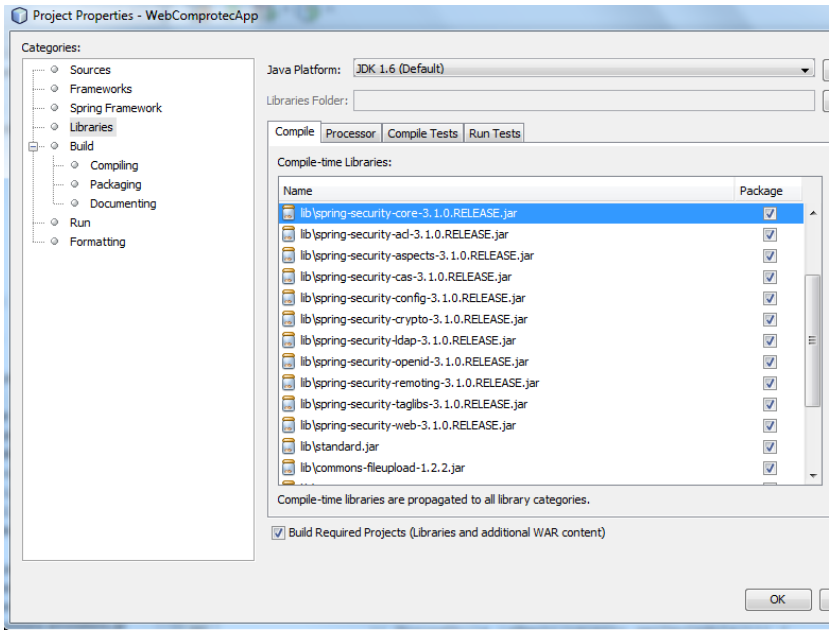


Figura IV.83. Paquetes Utilizados en la Aplicación.

4.4.4 Análisis de mejora en la gestión de proyectos de la COMPROTEC.

El proyecto que realizamos ha contribuido de manera muy importante para identificar y resaltar los puntos que hay que cubrir y considerar para llevar a cabo una implementación exitosa del sistema SGCOM.

En la introducción de este documento hablamos sobre cuáles son los objetivos que se desean lograr con el proyecto u de ellos es minimizar el tiempo de registro de un proyecto encabezado por los docentes, el mismo que minimiza en 80% registrar dicho proyecto.

CAPÍTULO V ANÁLISIS DE RESULTADOS

5.1 Generalidades

Con el fin de realizar la validación de la hipótesis: “La utilización del Framework Spring MVC que permitirá obtener mayor productividad en el desarrollo de aplicaciones web.” se determina que la Chi cuadrado es método a aplicar para la demostración de la hipótesis.

La necesidad de determinar la mejor tecnología para el desarrollo de aplicaciones web en un entorno de programación JAVA,utilizando la tecnología JSP/JDBC y Spring MVC/ Hibernate ,que permita minimizar el tiempo en la construcción y mantenimiento de las aplicaciones web, requiere de un análisis comparativo en base a parámetros quefaciliten la selección de una tecnología adecuada para este fin.

5.1.1 Diseño de Prototipos

Con el propósito de determinar si la tecnología SPRING / HIBERNATE es adecuada en la optimización del tiempo en la construcción y mantenimiento de aplicaciones web, se busca comparar con las tecnologías JSP / JDBC de aplicaciones web, que utilizan tecnología distinta, tales como son:

Para poder analizar y comparar las tecnologías se desarrolló dos prototipos.

Prototipo 1.-Fue implementado con unametodología tradicional, que está diseñada para proyectos de cualquier dimensión; haremos énfasis en el uso de herramientas como son JSP y JDBC, el prototipo de comparación, diseño e implementación del Sistema SIA (Sistema de Información y Acreditación) de la (Unidad Técnica de Evaluación Interna) de la ESPOCH.

Prototipo 2.- Fue implementado con unametodología propuesta, con herramientasSpring MVC e Hibernate, desarrollada en el presente trabajo investigativo.

Para esto, se han determinado parámetros de comparación los cuales permitirán establecer el tiempo en la construcción y mantenimiento de aplicaciones web y obtener un resultado de esta comparación.

5.1.1.1 Definición de los parámetros de comparación

Los parámetros y variables que a continuación se definen para la comparación de las dos tecnologías fueron seleccionados por el autor de esta tesis, conjuntamente con el encargado de la dirección de proyectos en la COPROTEC, en base a las referencias de la información obtenida.

Los parámetros que se consideraron para comparar las dos tecnologías son los siguientes:

Tabla V.I. Definición de Parámetros

PARÁMETRO	DESCRIPCIÓN
Reusabilidad	Poder usar partes o componentes de un software determinado en otro proyecto.
Generación de Código	Capacidad para convertir un programa sintácticamente correcto en una serie de instrucciones a ser interpretadas por una máquina.
Modularidad	Capacidad que tiene un sistema de ser estudiado, visto o estudiado como la unión de varias partes que interactúan entre sí y que trabajan para alcanzar un objetivo común, realizando cada una de ellas una tarea necesaria para la consecución de dicho objetivo.
Mantenibilidad	Facilidad con la que un sistema o componente software puede ser modificado para corregir fallos, mejorar su funcionamiento u otros atributos o adaptarse a cambios en el entorno.
Presentación	Tiene que ver con la facilidad de realizar interfaces amigables para el usuario.

5.1.1.2 Definición de Indicadores

De los parámetros establecidos para la comparación entre los prototipos, se ha seleccionado los siguientes indicadores que permiten medir la el tiempo en la construcción y mantenimiento de aplicaciones web.

Tabla V.II. Definición de los indicadores de Reusabilidad.

REUSABILIDAD	
ÍNDICE	**DESCRIPCIÓN**
Portabilidad	Se define como la característica que posee un software para ejecutarse en diferentes plataformas, el código fuente del software es capaz de reutilizarse en vez de crearse un nuevo código cuando el software pasa de una plataforma a otra.
Independencia de componentes	Capacidad de funcionamiento de un módulo independiente de la plataforma o de otro proyecto.
Facilidad de integración	Capacidad de integrar módulos a otros proyectos fácilmente.

Tabla V.III. Definición de los indicadores de Generación de código.

GENERACIÓN DE CÓDIGO	
ÍNDICE	**DESCRIPCIÓN**
Creación de capa de acceso a datos	Es la creación automática utilizando herramientas de desarrollo de código para la generación de clases que permiten el manejo de las bases de datos.
Creación de capa de lógica de negocio	Es la creación automática utilizando herramientas de desarrollo de código para la generación de clases que permiten el manejo de la lógica de datos.
Líneas de código	Índice que permite medir la cantidad de líneas de código para la realización de una consulta en específico.
Código no utilizable	Código generado o programado que no se utiliza.
Complejidad	Índice que permite medir la facilidad o complejidad de entendimiento del código.

Tabla V.IV. Definición de los indicadores de Modularidad.

MODULARIDAD	
ÍNDICE	**DESCRIPCIÓN**
MVC	Patrón para establecer un modelo de abstracción de desarrollo de software, que separa en componentes Modelo - Vista - Controlador.
Diseño estructurado	Indicador para establecer la estructura y organización de los diferentes objetos en la aplicación web.
Capacidad de descomposición	Indicar que se utiliza para cuantificar el nivel de descomposición de los diferentes componentes web.
Comprensión	Patrón para medir el nivel de facilidad y manejabilidad del código utilizado en la aplicación web.

Tabla V.V. Definición de los indicadores de Mantenibilidad.

MANTENIBILIDAD	
ÍNDICE	**DESCRIPCIÓN**
Escalabilidad	Propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para reaccionar y adaptarse sin perder calidad.
Disponibilidad de Documentación	Cantidad de información disponible en la red o en otros medios, sobre un tema específico.
Código Entendible.	Líneas de código entendible y fácil de manipular por un desarrollador cualquiera con conocimiento del tema.
Adaptabilidad a nuevos requerimientos	Facilidad con la que se adapta o acopla a nuevos requerimientos un software determinado.

Tabla V.VI. Definición de los indicadores de Presentación.

PRESENTACIÓN	
ÍNDICE	**DESCRIPCIÓN**
Templates personalizados	Se refiere al uso de plantillas de interfaz de usuario pre elaboradas, que permitan ahorrar tiempo en el desarrollo de interfaces.
Componentes jQuery y javaScript	Permite la creación de aplicaciones web interactivas, también llamadas aplicaciones RIA.
Validaciones	Proceso de comprobación de si algo satisface un cierto criterio.
Menor uso de java script	JavaScript es un lenguaje interpretado que se embebe en una página web HTML. Al utilizar javaScript para realizar validaciones o para ejecutar menús desplegables, datepickers, etc. esto hace que el desarrollo de la aplicación sea mayor en tiempo; en cuanto tiene que ver en el mantenimiento y desarrollo de la aplicación web.

5.1.1.3 Criterios de evaluación.

A continuación se muestran los valores cualitativos y cuantitativos que se darán a los parámetros a ser analizados en la comparación de las tecnologías, para lo cual se utilizara valores del 0 al 4.

Tabla V.VII. Criterios de Evaluación General.

CRITERIOS DE EVALUACIÓN GENERAL					
Cuantitativa	0	1	2	3	4
Cualitativa	Bajo	Medio Bajo	Medio	Medio Alto	Alto
Porcentajes	0%	25%	50%	75%	100%

5.1.1.4 Análisis de los parámetros de comparación para las tecnologías.

El análisis comparativo será realizado en base a la información que se obtenga de la investigación bibliográfica y a la observación realizada por los autores de la tesis utilizando para esto los dos prototipos implementados anteriormente.

5.1.1.4.1 Reusabilidad

5.1.1.4.1.1 Portabilidad

La portabilidad tiene que ver con el hecho de poder utilizar el código fuente de un software en uno totalmente independiente del anterior. También la portabilidad se refiere a que un sistema puede ser ejecutado en diferentes plataformas sin ningún inconveniente. A mayor portabilidad menor es la dependencia del software con respecto a la plataforma.

El pre-requisito para la portabilidad es la abstracción generalizada entre la aplicación lógica y las interfaces del sistema. Cuando un software se puede compilar en diversas plataformas (x86, IA64, amd64, etc.), se dice que es multiplataforma. Esta característica es importante para el desarrollo de reducción costos, cuando se quiere hacer una misma aplicación. Por lo general, todo software JAVA es multiplataforma, sin embargo, no todo software JAVA puede ser reutilizable.

El prototipo 1 desarrollado usando las tecnologías JSP y JDBC puede ser ejecutado en diferentes plataformas, sin embargo, la reutilización del código fuente es compleja, ya que es necesario acoplar éste código al nuevo proyecto, debido a que este código no funciona de forma independiente a las interfaces de usuario.

Por otro lado el prototipo 2 desarrollado, el cual utiliza la tecnología Spring Web MVC 3.0 e Hibernate manejan este término de portabilidad completamente, ya que si bien es multiplataforma, también el código fuente puede ser reutilizado en otros proyectos.[7]

De acuerdo con este análisis se establecen los siguientes valores:

Tabla V.VIII. Evaluación del Índice de Portabilidad.

Tecnologías / Indicador	**Prototipo 1**	**Prototipo 2**
	JSP / JDBC	**Spring MVC / Hibernate**
Criterio de evaluación	2	4
Porcentaje	50%	100%

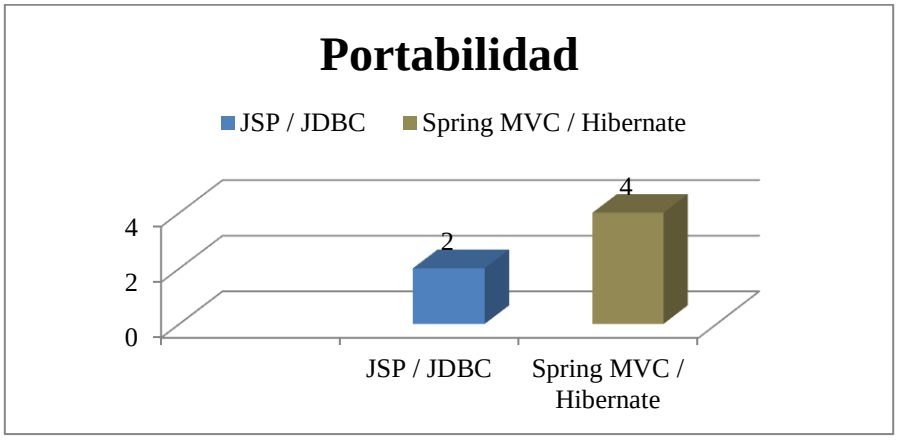


Figura V.1. Índice de comparación de Portabilidad.

Se observa los resultados obtenidos en la Figura V.1 para el índice de Portabilidad, en el cual el prototipo 2, que utiliza la tecnología jsp/jdbc, con el valor de 4 toma ventaja sobre el prototipo 1, el cual utiliza la tecnología spring / hibernate, ya que el prototipo 2 la portabilidad resulta más efectiva que en el prototipo 1.

5.1.1.4.1.2 Independencia de componentes

La independencia de componentes tiene que ver con el hecho de que un módulo pueda funcionar de manera independiente a otros módulos, es decir, un módulo independiente no necesariamente debe estar acoplado a otros para poderse ejecutar.

Mediante el desarrollo del prototipo 1, se establecieron dos módulos: Capa de Acceso a Datos y Lógica de Negocio y Capa de Presentación. Estos dos módulos no pueden actuar de manera independientes, es decir, la capa de acceso a datos y lógica de negocios necesita de la capa de presentación para poderse compilar, si hay un error en cualquiera de estos módulos, no se podrán compilar de forma separada.

Sin embargo, en el prototipo 2 que utiliza la tecnología SPRING / HIBERNATE, se establecieron de igual forma dos módulos: Capa de Acceso a Datos y Lógica de Negocios y la Capa de Presentación. La diferencia radica en que el primer módulo el cual es gestionado por Hibernate funciona como un proyecto independiente al segundo módulo gestionado por Spring Web MVC 3.0. Es decir, en un mismo proyecto se tiene dos proyectos ejecutándose de forma conjunta pero no dependiente el uno del otro. Esto permite que el módulo Hibernate pueda ser reutilizado en otros proyectos y acoplarse sin ninguna dificultad a estos, así mismo el módulo Spring Web MVC 3.0.

Conforme al análisis realizado se establecen los siguientes valores:

Tabla V.IX. Evaluación del Índice de Independencia de Componentes.

Tecnologías / Indicador	**Prototipo 1**	**Prototipo 2**
	JSP / JDBC	**Spring MVC / Hibernate**
Criterio de evaluación	1	3
Porcentaje	25%	75%

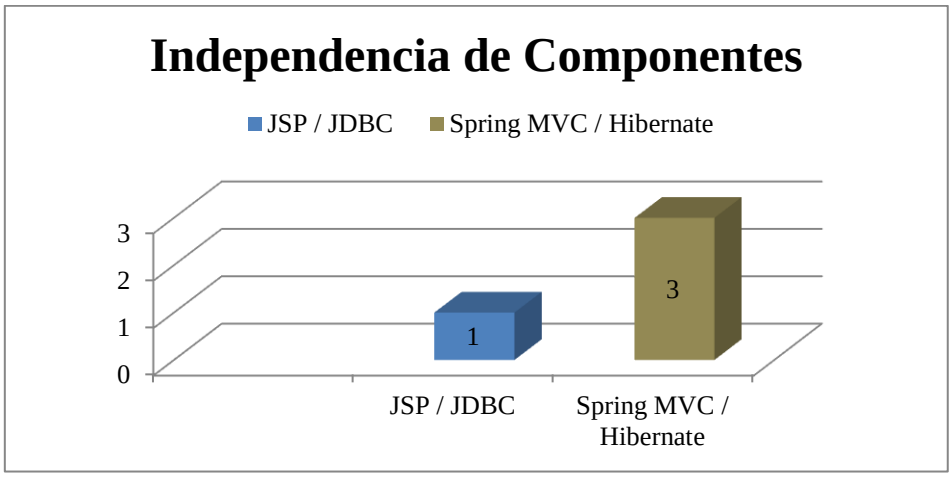


Figura V.2. Índice de comparación de Independencia de Componentes.

En la Figura V.2 se establece los valores correspondientes al prototipo 1 que utiliza la tecnología JSP / JDBC el cual posee un valor de 1 (medio bajo) frente al prototipo 2 que utiliza la tecnología SPRING / HIBERNATE con un valor de 4 (muy alto), determinando así que el prototipo 2 está en ventaja frente al prototipo 1 con respecto a la independencia de sus componentes.

5.1.1.4.1.3 Facilidad de integración

Es muy importante poder realizar la integración de componentes ya existentes a proyectos o aplicaciones nuevas. La integración puede resultar un trabajo complejo o a la vez algo muy sencillo si se utilizan las herramientas tecnológicas.

En el desarrollo del prototipo 1 usando los componentes java, para su realización, la integración no generó mayor complicación.

De igual forma usando la tecnología SPRING / HIBERNATE, la integración a otros genera ninguna dificultad, ya que, como se dijo anteriormente, cada módulo se trata como componentes separados que pueden ser integrados a cualquier proyecto.

Los valores establecidos respecto al análisis hecho son:

Tabla V.X. Evaluación del Índice de Facilidad de Integración.

Tecnologías / Indicador	Prototipo 1	Prototipo 2
	JSP / JDBC	**Spring MVC / Hibernate**
Criterio de evaluación	4	4
Porcentaje	100%	100%

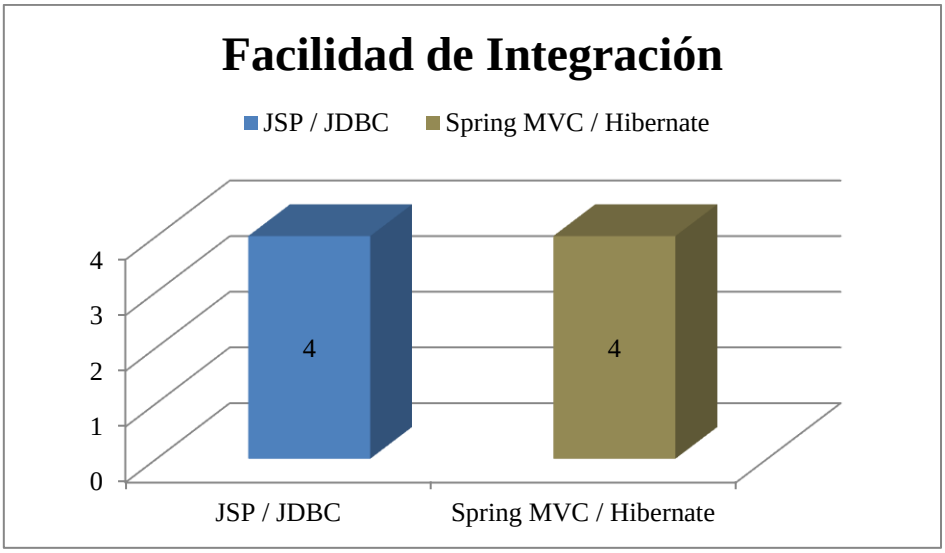


Figura V.3. Índice de comparación de Facilidad de Integración.

En la Figura V.3 se establece que existe una facilidad de integración alta utilizando cualquiera de los componentes ya sea JSP- JDBC o Spring Web MVC 3.0 - Hibernate.

De acuerdo a al análisis de cada índice se obtienen los siguientes valores para medir la reusabilidad que se obtiene desarrollando un proyecto siguiendo la tecnología JSP / JDBC o siguiendo la tecnología SPRING / HIBERNATE.

Tabla V.XI. Resultados globales del índice de Reusabilidad.

Tecnologías / Indicador	JSP / JDBC	Spring MVC / Hibernate	Peso Máximo
Portabilidad	2	4	4
Independencia de componentes	1	3	4
Facilidad de integración	4	4	4
Total	7	11	12
Porcentaje de reusabilidad	58,33%	91,67%	100,00%

Siguiendo la tecnología JSP / JDBC se obtiene una reusabilidad del 58,33%, mientras que siguiendo la tecnología SPRING / HIBERNATE se obtiene una reusabilidad del 91,67%. De modo que, está demostrado que la tecnología SPRING / HIBERNATE abarca muy bien el término de reusabilidad para minimizar el tiempo en la construcción y mantenimiento de aplicaciones web. En la siguiente figura se observan los resultados obtenidos de cada uno de los índices.

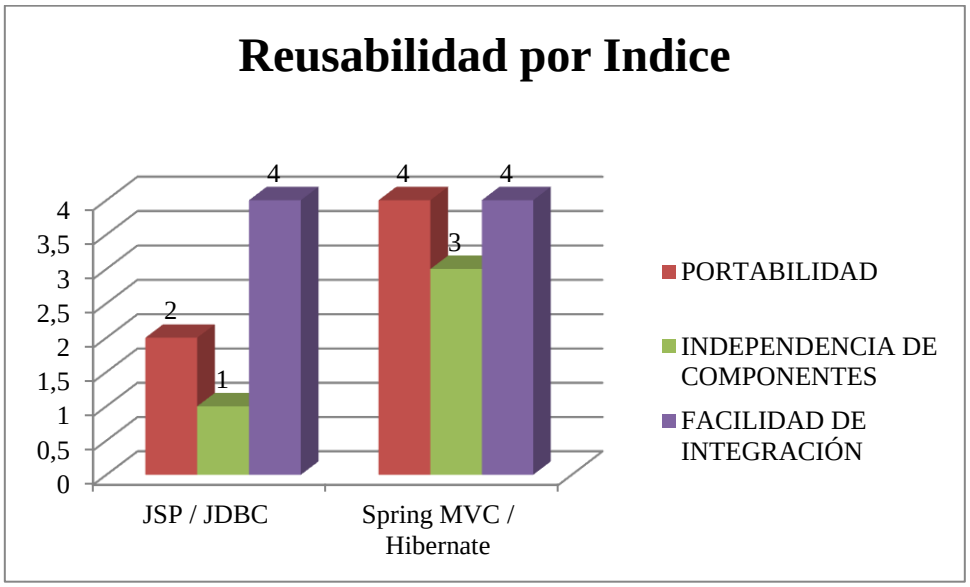


Figura V.4. Resultados del índice de comparación de Reusabilidad

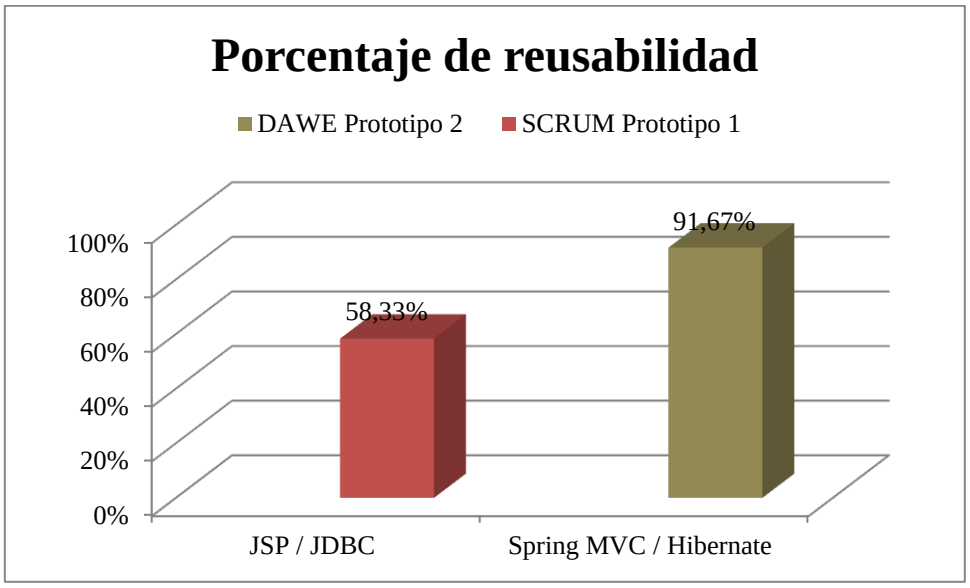


Figura V.5. Porcentajes totales de Reusabilidad.

5.1.1.4.1.4 Interpretación

De acuerdo a la Figura V.5 de porcentajes al utilizar la tecnología JSP / JDBC que hace uso de herramientas como Spring Web MVC 3.0 e Hibernate se maneja muy poco el concepto de Reusabilidad. Si se desea reutilizar componentes de una aplicación realizada en JSP con JDBC en otro proyecto se deberá acoplar cada clase creada al nuevo proyecto, de modo que esto permite que el tiempo en la construcción de un nuevo proyecto sea mayor.

Sin embargo, haciendo uso de las tecnologías Java Spring Web MVC 3.0 e Hibernate la reusabilidad es un concepto que se maneja adecuadamente, debido a que el módulo Hibernate trabaja independientemente y se acopla fácilmente a un nuevo proyecto.

Con todo esto, se puede demostrar que el prototipo desarrollado con JSP y JDBC no se acopla debidamente a nuevos proyectos, mientras que el prototipo desarrollado con Spring Web MVC 3.0 e Hibernate lo hace satisfactoriamente, ya que Hibernate es generado independiente y puede ser exportado a otros proyectos.

5.1.1.4.2 Generación De Código

5.1.1.4.2.1 Creación de capa de acceso a datos (DAO)

La capa de acceso a datos sirve como puente entre la capa controller y el proveedor de datos. Este capa pretende encapsular las especificidades del proveedor de datos tales como (SQL, Oracle, Sybase, archivos XML, texto, hojas electrónicas), a la siguiente capa. Para que si cambia el proveedor de datos solo necesitemos cambiar en una sola capa el proveedor de datos.

Generar el acceso a datos puede ser un proceso muy tedioso y complicado en algunos casos, sin embargo, si se utilizan las herramientas necesarias, se podrá generar esta capa de forma automática.

En los prototipos desarrollados con las diferentes tecnologías, está realizada la capa de acceso a datos. Esta capa fue creada de manera diferente en cada prototipo.

En el prototipo 1 que utiliza tecnología JSP / JDBC, la capa de acceso a datos se realizó con código programable por los desarrolladores, mas no por código generado ya que utilizando las tecnologías JSP y JDBC no permiten la generación de código de la capa de acceso a datos. En cierta manera esto podría ser una ventaja para el desarrollador ya que le permitirá familiarizarse más con el software que se está desarrollando, y como es su propio código conoce donde pueden haber posibles errores de sintaxis o de lógica, sin embargo sería una desventaja en cuanto al tiempo de desarrollo de la aplicación, ya que crear el código adecuado para esta capa es muy tedioso, peor aún si se tiene una base de datos muy grande.

En el prototipo 2, donde se utiliza la tecnología SPRING / HIBERNATE, toda la capa de acceso a datos se genera automáticamente, sin necesidad que se programe, esto permite minimizar el tiempo en la construcción de aplicaciones web, ya que el desarrollador solo se centraría en algunos aspectos de la lógica de negocio y en la capa de presentación. El módulo Hibernate en el cual está establecida la capa de acceso a datos y la lógica de negocios, permite generar sin mayor inconveniente el acceso a datos completamente, una vez que se haya establecido la conexión a la base de datos pertinente.

De acuerdo con esta observación se puede establecer los siguientes valores.

Tabla V.XII. Evaluación del Índice de Creación de capa DAO.

Tecnologías / Indicador	Prototipo 1	Prototipo 2
	JSP / JDBC	Spring MVC / Hibernate
Criterio de evaluación	2	3
Porcentaje	50%	75%

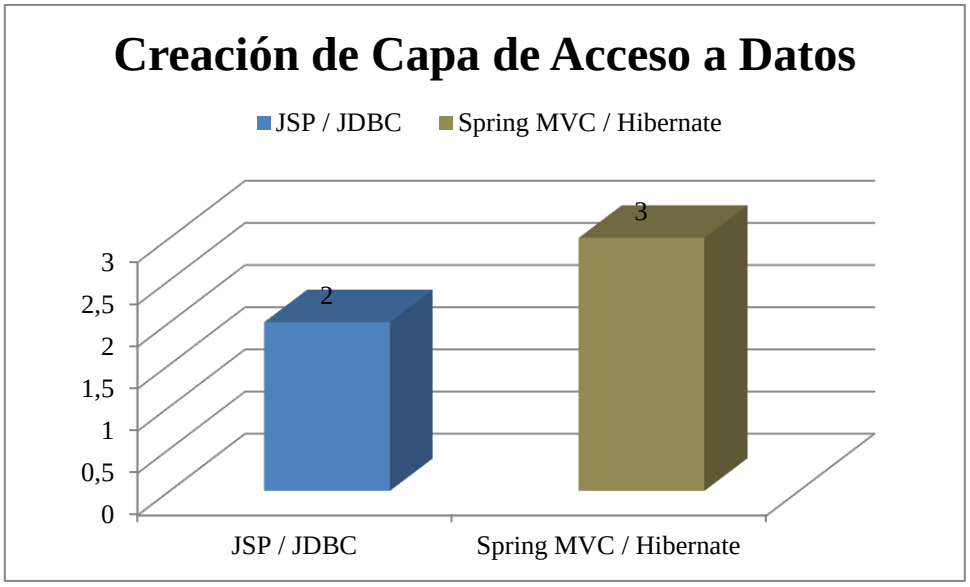


Figura V.6. Índice de comparación de la Capa Acceso a Datos.

Dada la Figura V.6 se observa claramente que siguiendo la tecnología SPRING / HIBERNATE utilizando las tecnologías Spring Web MVC 3.0 - Hibernate se tiene una ventaja con la tecnología JSP / JDBC que utiliza JSP y JDBC, con respecto a la creación de la capa de acceso a datos, ya que en la tecnología SPRING / HIBERNATE, esta capa se genera completamente de forma automática y no hay que programarle, minimizando así el tiempo en la construcción de aplicaciones web.

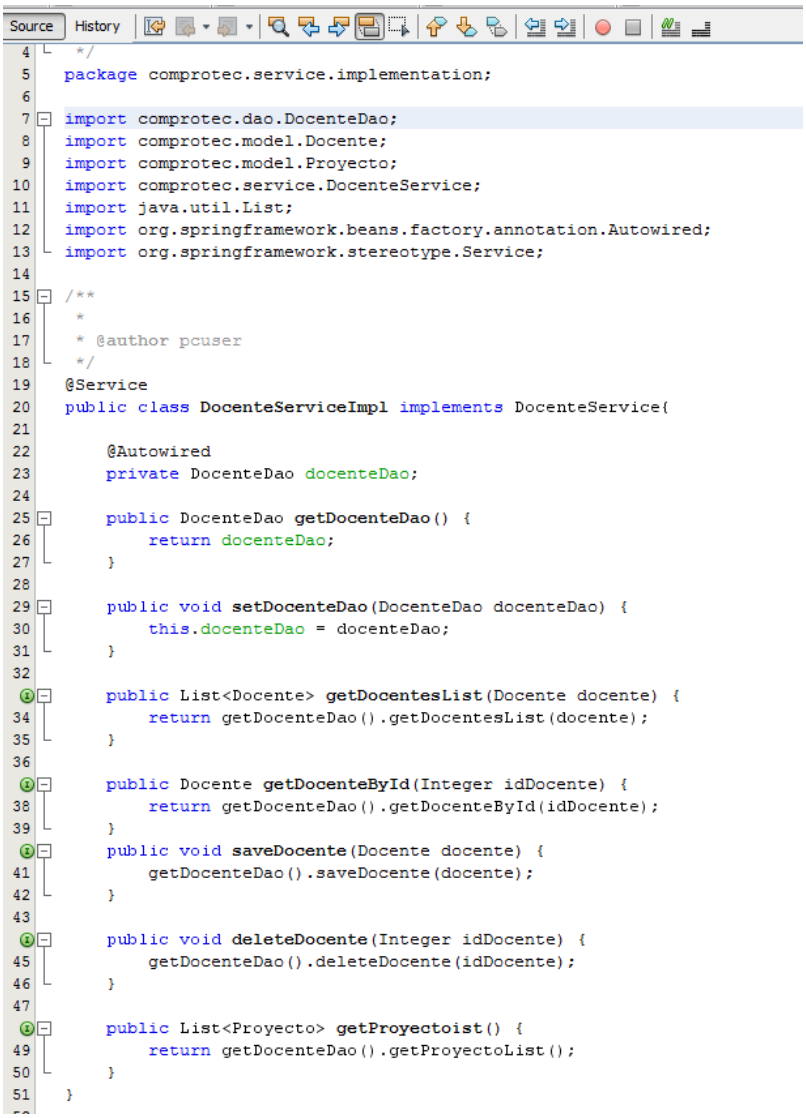
5.1.1.4.2.2 Generación de capa de lógica de negocio (Service/Model)

La capa de lógica de negocio es el corazón de la aplicación. El objetivo de esta capa es que toda la lógica de negocio esté bien localizada y no mezclada con los otros objetos de las otras capas, conllevando así a grandes ventajas. Es en esta capa donde se establecen todas las reglas de negocio

que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.

Utilizando la tecnología JSP / JDBC se debe programar la capa de negocio totalmente conforme a los requerimientos de usuario establecidos ya que las tecnologías utilizadas no generan automáticamente ningún tipo de código para esta capa.

Usando la tecnología SPRING / HIBERNATE, el código que genera con respecto a la lógica de negocio es muy poco. Las tecnologías utilizadas solo generan el CRUD (Create, Retrieve, Update, Delete) de cada entidad, por lo que las demás reglas de negocio deben ser programadas manualmente. Así como se muestra a continuación:



```
Source  History
 4   */
 5  package comprotec.service.implementation;
 6
 7  import comprotec.dao.DocenteDao;
 8  import comprotec.model.Docente;
 9  import comprotec.model.Proyecto;
10  import comprotec.service.DocenteService;
11  import java.util.List;
12  import org.springframework.beans.factory.annotation.Autowired;
13  import org.springframework.stereotype.Service;
14
15  /**
16   *
17   * @author pcuser
18   */
19  @Service
20  public class DocenteServiceImpl implements DocenteService{
21
22      @Autowired
23      private DocenteDao docenteDao;
24
25      public DocenteDao getDocenteDao() {
26          return docenteDao;
27      }
28
29      public void setDocenteDao(DocenteDao docenteDao) {
30          this.docenteDao = docenteDao;
31      }
32
34      public List<Docente> getDocentesList(Docente docente) {
35          return getDocenteDao().getDocentesList(docente);
36
38      public Docente getDocenteById(Integer idDocente) {
39          return getDocenteDao().getDocenteById(idDocente);
41      public void saveDocente(Docente docente) {
42          getDocenteDao().saveDocente(docente);
43
45      public void deleteDocente(Integer idDocente) {
46          getDocenteDao().deleteDocente(idDocente);
47
49      public List<Proyecto> getProyectoist() {
50          return getDocenteDao().getProyectoList();
51  }
```

Figura V.7. Métodos CRUD del Docente.

Dada esta observación se establecerán los siguientes valores con respecto a la generación de la capa de lógica de negocios:

Tabla V.XIII. Evaluación del Índice de la Generación de Capa Service/Model.

Tecnologías / Indicador	Prototipo 1	Prototipo 2
	JSP / JDBC	Spring MVC / Hibernate
Criterio de evaluación	0	1
Porcentaje	0%	25%

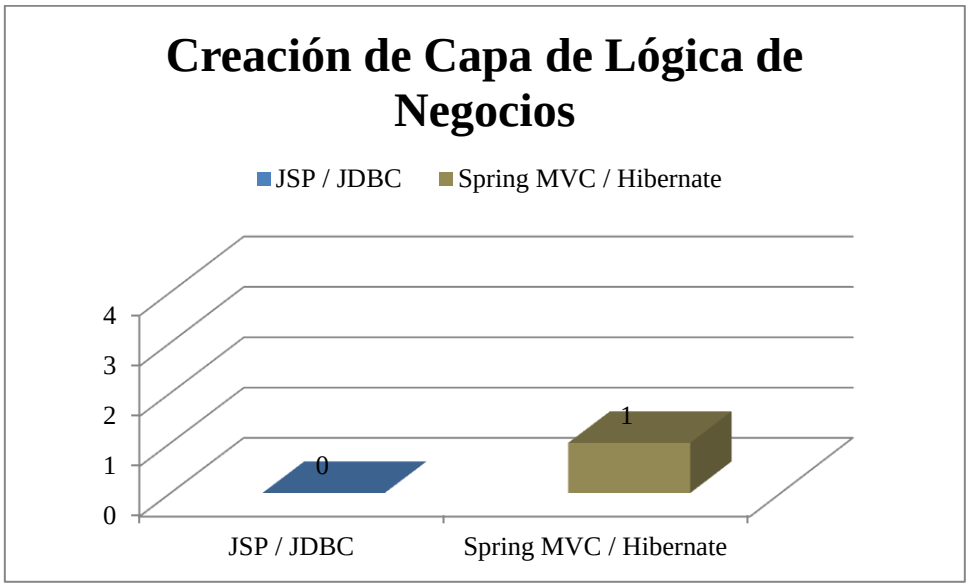


Figura V.8. Índice de comparación de la Capa de Negocio.

En la Figura V.8 se observa que usando la tecnología SPRING / HIBERNATE se ahorra más tiempo al momento de crear la capa de lógica de negocios que usando la tecnología JSP / JDBC, ya que la tecnología SPRING / HIBERNATE genera poco código con respecto a la capa de negocio, sin embargo la tecnología JSP / JDBC no genera ningún tipo de código para esta capa.

5.1.1.4.2.3 Líneas de código

Tabla V.XIV. Criterio de Evaluación para medir las Líneas de Código.

Cuantitativa	**0**	**1**	**2**	**3**	**4**
Cualitativa	**>40**	**30-40**	**20-29**	**10-19**	**<10**
Porcentajes	**0%**	**25%**	**50%**	**75%**	**100%**

El número líneas de código dentro de una aplicación es muy importante para poder mantener de mejor manera determinada aplicación y también minimizar el tiempo en la construcción de estas.

Para verificar el número de líneas de código en cada prototipo, se escogieron procedimientos o tareas puntuales y similares en cada uno de ellos. Vamos a escoger dos tareas en cada prototipo. La primera tarea será la inserción de una determinada entidad y la segunda tarea será la eliminación de esta Entidad.

PROTOTIPO 1

En el prototipo 1 utilizando la tecnología JSP / JDBC el número de líneas de código es el siguiente:

Tarea 1: Inserción de un docente.

```
 */
public byte create(){
    byte bytEstado=0;
    CallableStatement ejecutar;
    try{
        ejecutar = this.conexion.prepareCall("{call SP_INSERT_PROFESOR_INV_PRINC(?,?,?,?,?,?,?,?,?)}");

        ejecutar.setString (1,PRICODIGO);
        ejecutar.setString (2,PRICEDULA);
        ejecutar.setString (3,PRIAPELLIDO_PATERNO);
        ejecutar.setString (4,PRIAPELLIDO_MATERNO);
        ejecutar.setString (5,PRINOMBRES);
        ejecutar.setString (6,PRISEXO);
        ejecutar.setString (7,PRIFECHA_NACIMIENTO);
        ejecutar.setString (8,SEDCODIGO);
        ejecutar.setString (9,NACCODIGO);

        ejecutar.execute();

        ejecutar.close();
        ejecutar = null;

        bytEstado = 1;
    }
    catch(SQLException e){
        this.strError = e.getMessage().toString();
    }
    return bytEstado;
}
/**
```

Figura V.9. Método de Inserción Prototipo 1.

Número de Líneas de Código: 24 líneas de código.

Tarea 2: Eliminación de Técnicos.

```
/**
 * Este metodo elimina un registro de PROFESOR_INVESTIGADOR_PRINCIPA
 * @return no retorna un dato tifo byte
 */
public byte delete(){
    byte bytEstado=0;
    CallableStatement ejecutar;
    try{
        ejecutar = this.conexion.prepareCall("{call SP_DELETE_PROFESOR_INVESTIGADOR_PRINCIPA (?)}");
        ejecutar.setString(1, PRICODIGO);
        ejecutar.execute();

        ejecutar.close();
        ejecutar = null;
        bytEstado = 1;
    }
    catch(SQLException e){
        this.strError = e.getMessage().toString();
    }
    return bytEstado;
}
```

Figura V.10. Método de Eliminación Prototipo 1.

Número de Líneas de Código: 15 líneas de código.

PROTOTIPO 1

En el prototipo 2 utilizando la tecnología SPRING / HIBERNATE el número de líneas de código es el siguiente:

Tarea 1: Inserción de un docente.

```
public void saveProyecto(Proyecto proyecto) {
    hibernateTemplate.saveOrUpdate(proyecto);
}
```

Figura V.11. Método de Inserción Prototipo 2.

Número de Líneas de Código: 3 líneas de código.

Tarea 2: Eliminación de Técnicos.

```
public void deleteProyecto(Integer idProyecto) {
    Proyecto proyecto = getProyectoById(idProyecto);
    hibernateTemplate.delete(proyecto);
}
```

Figura V.12. Método de Eliminación Prototipo 2.

Número de Líneas de Código: 3 líneas de código.

Dada esta observación obtendremos los siguientes valores:

Tabla V.XV. Evaluación del Índice de las Líneas de Código.

Tecnologías / **Indicador**	**Prototipo 1**		**Prototipo 2**	
	JSP / JDBC		**Spring MVC / Hibernate**	
	Tarea 1	Tarea 2	Tarea 1	Tarea 2
Líneas de código	24	15	3	3
Promedio	20		3	
Criterio de evaluación	2		3	
Porcentaje	50%		75%	

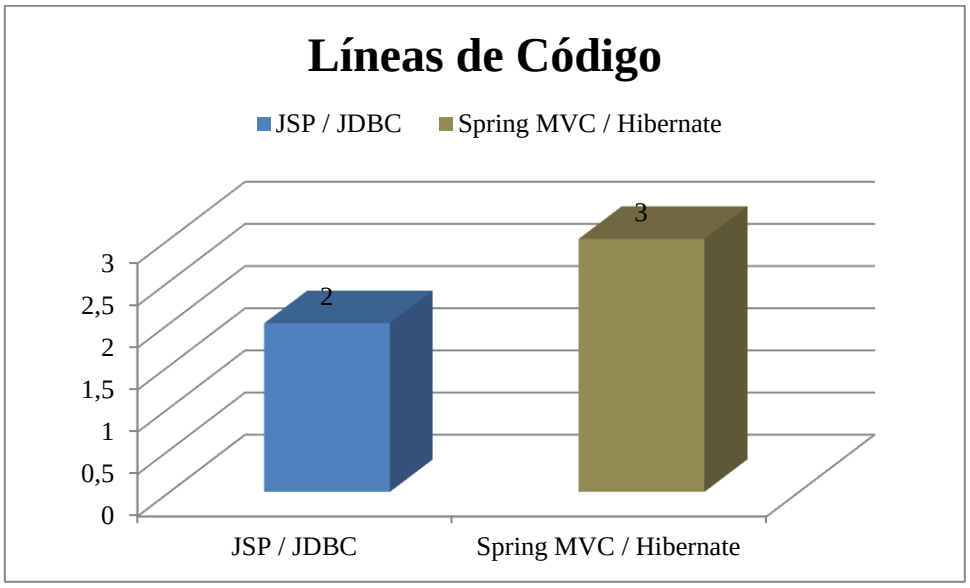


Figura V.13. Índice de comparación de Líneas de Código.

Mediante la Figura V.13 se establece que utilizando la tecnología SPRING / HIBERNATE se ahorra mucho tiempo en la programación de una aplicación web, ya que las líneas de código utilizadas para realizar tareas esenciales que utiliza de 5 a 6 veces más líneas de código que la anterior, para realizar la misma tarea. Esto permite que se optimice el tiempo en la construcción y mantenimiento de aplicaciones web usando la tecnología SPRING / HIBERNATE.

5.1.1.4.2.4 Código no utilizable

Tabla V.XVI. Criterio de Evaluación para medir el Código no Utilizable.

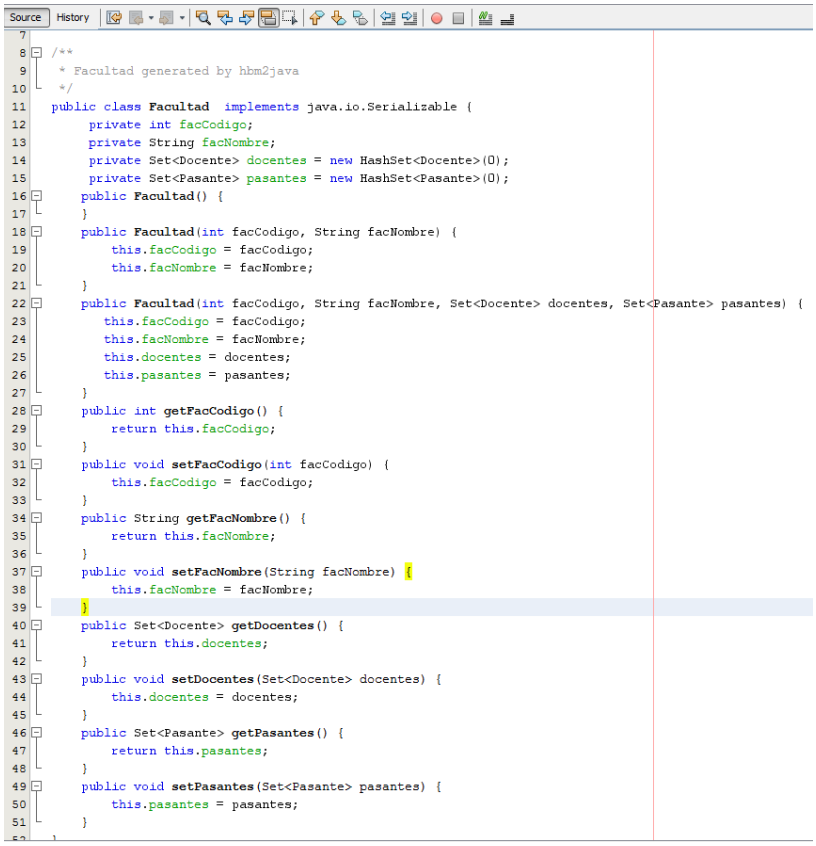
Cuantitativa	**0**	**1**	**2**	**3**	**4**
Cualitativa	**Excesivo**	**Mucho**	**Poco**	**Muy Poco**	**Nada**
Porcentajes	**0%**	**25%**	**50%**	**75%**	**100%**

Tener código no utilizable en una aplicación puede generar problemas al momento de dar un mantenimiento a una determinada aplicación, además que puede volver pesada a la aplicación al momento de su ejecución.

Usando la tecnología JSP / JDBC no se encontrara problemas de código innecesario o no utilizable ya que el código aquí es propiamente programado por el desarrollador, y este código es utilizado en toda la aplicación. Sin embargo, usando la tecnología SPRING / HIBERNATE, como la mayoría del código es generado, se genera código que es innecesario y que no se utiliza en la aplicación, por lo cual esto sería una gran desventaja al utilizar esta tecnología.

Acceso de datos:

El número de líneas de código no utilizable que se genera usando la tecnología SPRING / HIBERNATE es de: 0 líneas por cada entidad en el acceso a datos o en la capa Model del patrón MVC.



```
Source  History
 7
 8  /**
 9   * Facultad generated by hbm2java
10   */
11  public class Facultad  implements java.io.Serializable {
12       private int facCodigo;
13       private String facNombre;
14       private Set<Docente> docentes = new HashSet<Docente>(0);
15       private Set<Pasante> pasantes = new HashSet<Pasante>(0);
16      public Facultad() {
17      }
18      public Facultad(int facCodigo, String facNombre) {
19          this.facCodigo = facCodigo;
20          this.facNombre = facNombre;
21      }
22      public Facultad(int facCodigo, String facNombre, Set<Docente> docentes, Set<Pasante> pasantes) {
23         this.facCodigo = facCodigo;
24         this.facNombre = facNombre;
25         this.docentes = docentes;
26         this.pasantes = pasantes;
27      }
28      public int getFacCodigo() {
29          return this.facCodigo;
30      }
31      public void setFacCodigo(int facCodigo) {
32          this.facCodigo = facCodigo;
33      }
34      public String getFacNombre() {
35          return this.facNombre;
36      }
37      public void setFacNombre(String facNombre) {
38          this.facNombre = facNombre;
39      }
40      public Set<Docente> getDocentes() {
41          return this.docentes;
42      }
43      public void setDocentes(Set<Docente> docentes) {
44          this.docentes = docentes;
45      }
46      public Set<Pasante> getPasantes() {
47          return this.pasantes;
48      }
49      public void setPasantes(Set<Pasante> pasantes) {
50          this.pasantes = pasantes;
51      }
```

Figura V.14. No existe Código no Utilizable en la tecnología spring / hibernate - Capa Acceso a Datos (DAO).

Además genera código no utilizable en la capa DAO con un total de 0 líneas por entidad ya que al igual es intervenida por el usuario para su implementación:

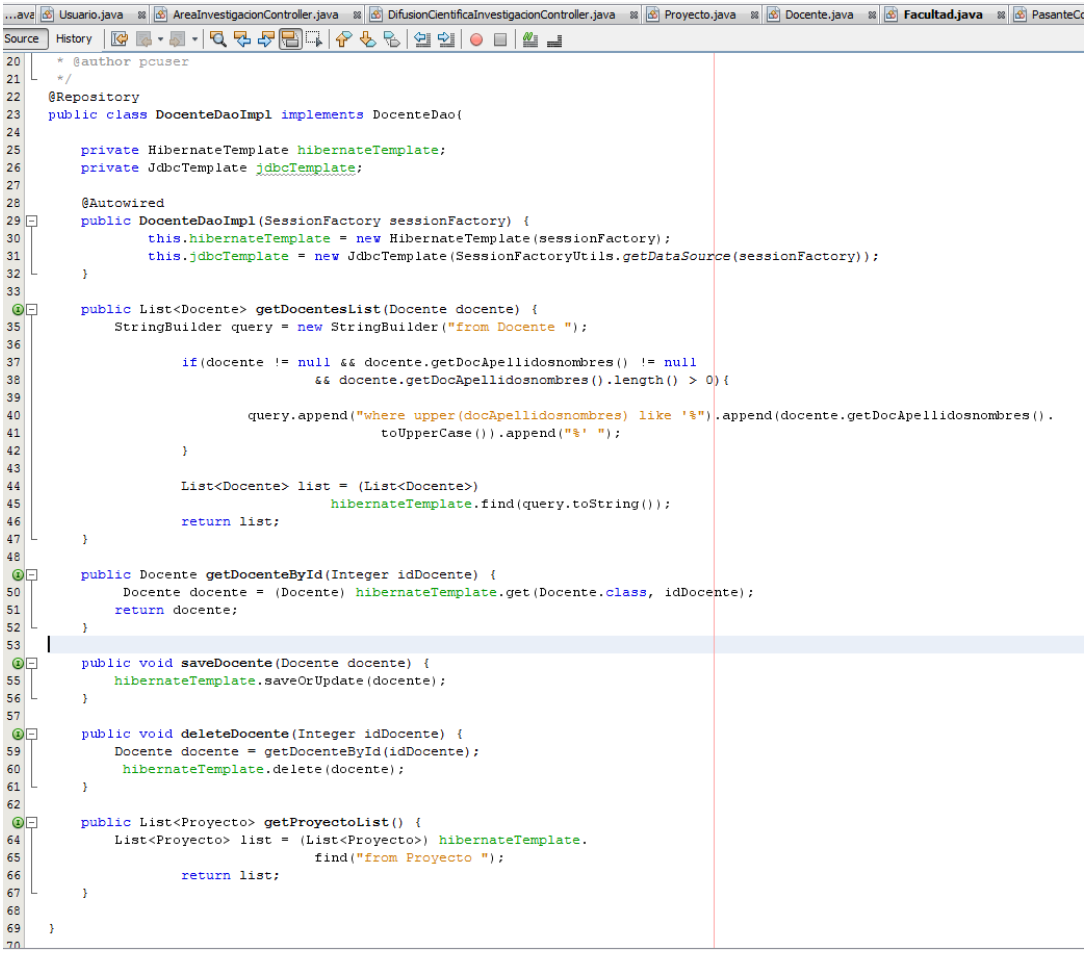


Figura V.15. No existe Código no Utilizable en la tecnología spring / hibernate - Capa de Negocio (Service/Model).

Establecida esta observación se puede establecer los siguientes valores:

Tabla V.XVII. Evaluación del Índice de Código No Utilizable.

Indicador \ Tecnologías	Prototipo 1	Prototipo 2
	JSP / JDBC	Spring MVC / Hibernate
Criterio de evaluación	4	3
Porcentaje	100%	75%

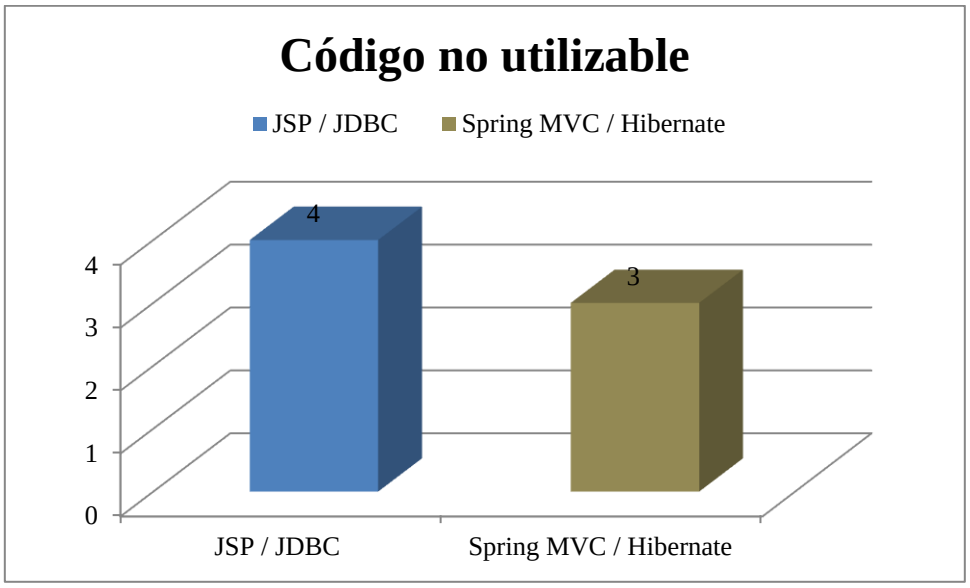


Figura V.16. Índice de comparación de Código no Utilizable.

De acuerdo a la Figura V.16 de código fuente de una aplicación web va a ser aprovechado por completo y no dará lugar a código innecesario que implique dificultad al momento de mantener una aplicación o al momento de construirla. Por lo visto la tecnología SPRING / HIBERNATE no genera código innecesario, para el desarrollo de aplicaciones web.

5.1.1.4.2.5 Complejidad

Tabla V.XVIII. Criterio de Evaluación para medir la Complejidad.

Cuantitativa	**0**	**1**	**2**	**3**	**4**
Cualitativa	**Muy Alta**	**Alta**	**Media**	**Baja**	**Muy Baja**
Porcentajes	**0%**	**25%**	**50%**	**75%**	**100%**

Este índice tiene que ver con cuán entendible o no es el código creado o generado en una aplicación web. Mientras más entendible sea el código menos tiempo se necesitará para construir o mantener una aplicación. Para construir aplicaciones usando la tecnología JSP / JDBC el código creado es entendible por cualquier desarrollador de software, ya que las tecnologías que utiliza se basan a la programación orientada a objetos y es muy fácil de entender por cualquier desarrollador. Sin embargo usando la tecnología SPRING / HIBERNATE, si se desea entender completamente el

código generado o creado se debe tener conocimiento sobre la programación en JAVA, de todos modos, el código aquí es un poco entendible para cualquier desarrollador de software ya que de una u otra manera también se basa en la programación orientada a objetos.

En el prototipo 1, para poder realizar el listado de Docentes de una base de datos se escribe simplemente una consulta SQL:

```
238     /**
239      * Este metodo retorna todos los registros de PROFESOR_INVESTIGADOR_PRINCIPA
240      * @return no retorna un dato tifo byte
241      */
242     public ResultSet retriveAll(){
243        ResultSet rs=null;
           PreparedStatement pst=null;
245        try{
246            pst = conexion.prepareStatement("SELECT * FROM PROFESOR_INVESTIGADOR_PRINCIPA");
247            rs=pst.executeQuery();
248        }
249        catch(SQLException e){
250            this.strError = e.getMessage().toString();
251        }
252        return rs;
253     }
```

Figura V.17. Sentencia SQL entendible.

Este código es entendible para cualquier desarrollador.

Sin embargo, en el prototipo 2, para realizar el listado de Docentes, se tiene el siguiente código:

```
68
69      public List<Docente> getDocenteList() {
70          List<Docente> list = (List<Docente>) hibernateTemplate.
71                                      find("from Docente ");
72                  return list;
73      }
```

Figura V.18. Sentencia SQL Compleja.

Dado estos criterios se obtienen los siguientes valores:

Tabla V.XIX. Evaluación del Índice de Complejidad.

Indicador \ Tecnologías	Prototipo 1	Prototipo 2
	JSP / JDBC	Spring MVC / Hibernate
Criterio de evaluación	3	1
Porcentaje	75%	25%

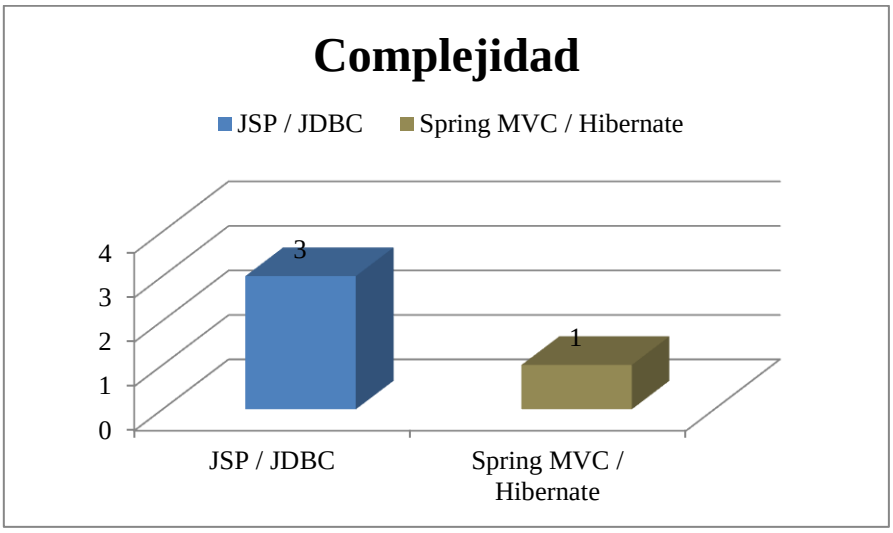


Figura V.19. Índice de comparación de Complejidad.

Analizada la Figura V.19 código que se genera es muy entendible y esto ayuda para un mejor mantenimiento a la aplicación minimizando así el tiempo en el mantenimiento. Por otro lado, se establece una complejidad en el código si se usa la tecnología SPRING / HIBERNATE, afectando de esta manera a un mantenimiento óptimo y eficaz y a maximizar el tiempo en la construcción de nuevas aplicaciones.De acuerdo al análisis de cada índice se obtienen los siguientes valores para medir la generación de código que se obtiene desarrollando un proyecto siguiendo la tecnología JSP / JDBC o siguiendo la tecnología SPRING / HIBERNATE.

Tabla V.XX. Resultados globales del índice de Generación de Código.

Tecnologías / Indicador	**JSP / JDBC**	**Spring MVC / Hibernate**	**Peso Máximo**
Creación de capa Model.	2	3	4
Creación de capa DAO.	0	1	4
Líneas de código.	2	3	4
Código no utilizable.	4	3	4
Complejidad.	3	1	4
Total	11	11	20
Porcentaje de generación de código	55,00%	55,00%	100,00%

Siguiendo la tecnología JSP / JDBC se obtiene una eficacia de generación de código del 55%, de la misma forma siguiendo la tecnología SPRING / HIBERNATE se obtiene una eficacia de generación de código del 70%. De modo que, usando la tecnología SPRING / HIBERNATE tendrá un alto grado de eficacia y el tiempo en la construcción y mantenimiento de aplicaciones web será constante. En la siguiente figura se observan los resultados obtenidos de cada uno de los índices.

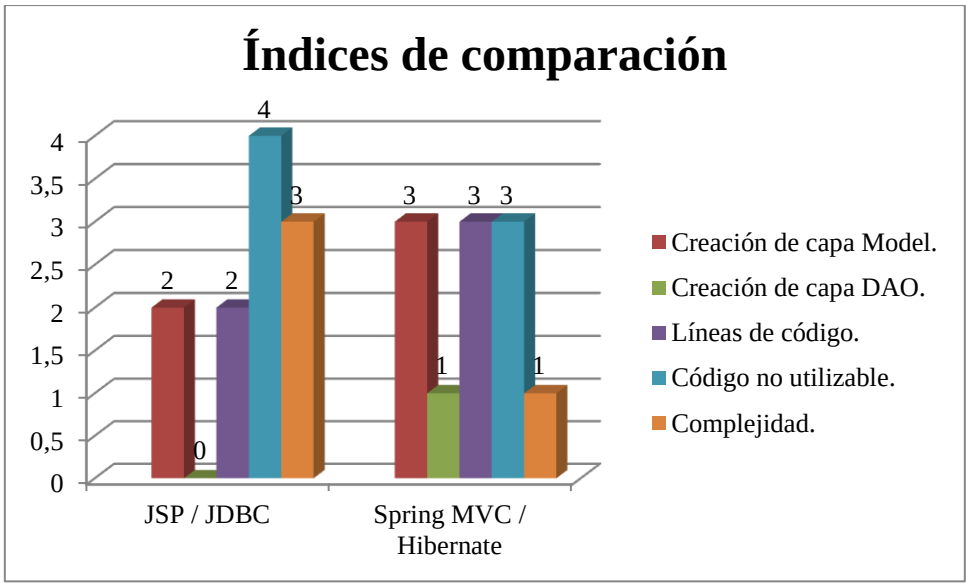


Figura V.20. Resultados del Índice de Comparación de Generación de Código.

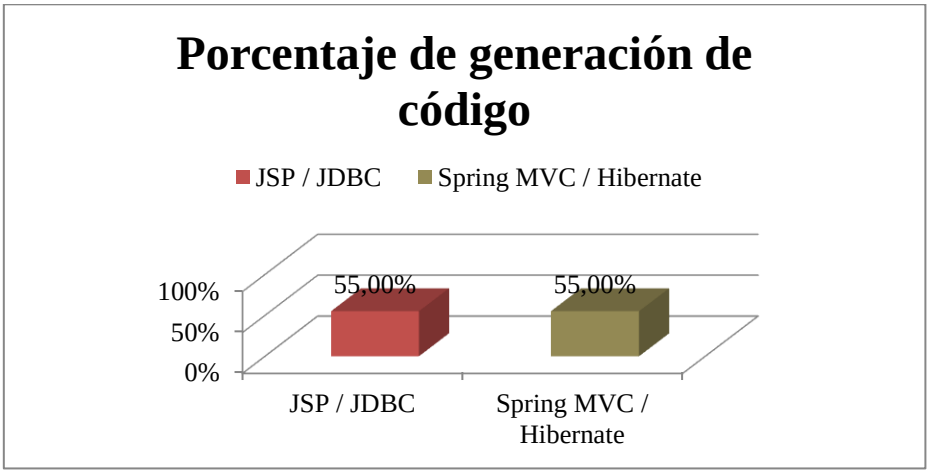


Figura V.21. Porcentajes del parámetro de generación de código.

5.1.1.4.2.6 Interpretación

En cuanto a la generación de código se puede observar en la Figura V.21 que tanto usando una tecnología JSP / JDBC existe un 55% de utilidad del código que se genera, por lo cual, usando la tecnología SPRING / HIBERNATE existe 55% la cual es mejor para minimizar el tiempo en la construcción y/o mantenimiento de aplicaciones web.

5.1.1.4.3 Modularidad

5.1.1.4.3.1 Modelo Vista Controlador (MVC)

Es un patrón o modelo de abstracción de desarrollo de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. El patrón de llamada y retorno MVC (según CMU), se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página.

El prototipo realizado por la tecnología SPRING / HIBERNATE es realizado de acuerdo a este índice, obteniendo ventajas como son:

1) Sencillez para crear distintas representaciones de los mismos datos.
2) Facilidad para la realización de pruebas unitarias de los componentes, así como de aplicar desarrollo guiado por pruebas (TDD).
3) Reutilización de los componentes.
4) Simplicidad en el mantenimiento de los sistemas.
5) Facilidad para desarrollar prototipos rápidos.
6) Los desarrollos suelen ser más escalables.

Tabla V.XXI. Evaluación del Índice de MVC.

Tecnologías / Indicador	Prototipo 1	Prototipo 2
	JSP / JDBC	Spring MVC / Hibernate
Criterio de evaluación	1	4
Porcentaje	25%	100%

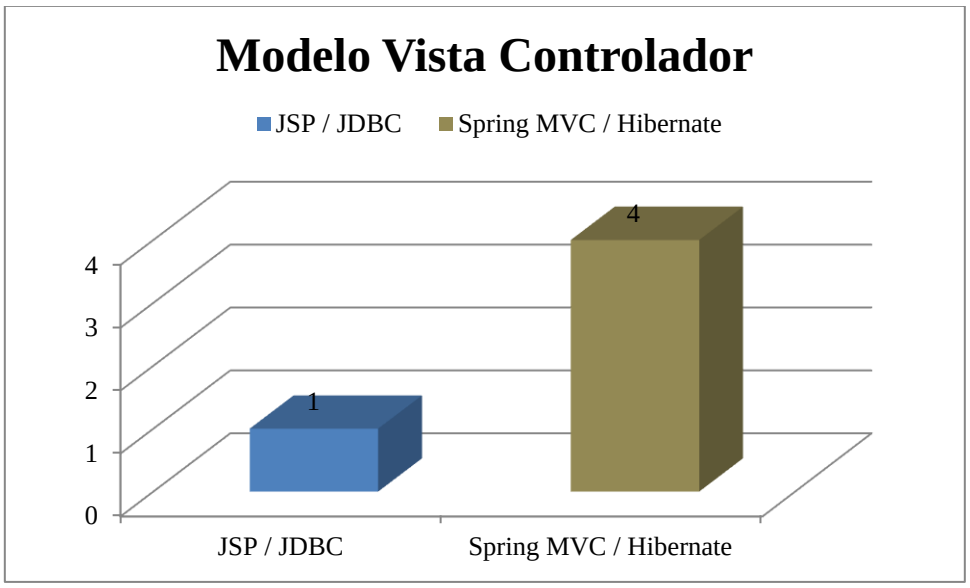


Figura V.22. Índice de comparación de MVC.

Se puede observar en la Figura V.22 que el prototipo 2 de la tecnología SPRING / HIBERNATE tiene un valor mayor al prototipo 1, esto debido al patrón Modelo Vista Controlador que adopta la tecnología SPRING / HIBERNATE.

5.1.1.4.3.2 Diseño estructurado

El hecho de que una tecnología no adopte el Modelo Vista desarrollo de software no implica que su diseño estructurado este mal.

Definición:

Diseño estructurado es el proceso de decidir que componentes, y la interconexión entre los mismos, para solucionar un problema bien especificado.

Frente a esto se puede encontrar los siguientes beneficios: Eficiencia, Mantenibilidad, Modificabilidad, Flexibilidad, Generalidad, Utilidad.

En el desarrollo de los prototipos se ha encontrado que tanto la tecnología JSP / JDBC como la tecnología SPRING / HIBERNATE siguen un diseño bien estructurado, razón por la cual tienen el mismo porcentaje de calificación en este índice.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla V.XXII. Evaluación del Índice de Diseño Estructurado.

Tecnologías / Indicador	Prototipo 1	Prototipo 2
	JSP / JDBC	Spring MVC / Hibernate
Criterio de evaluación	4	4
Porcentaje	100%	100%

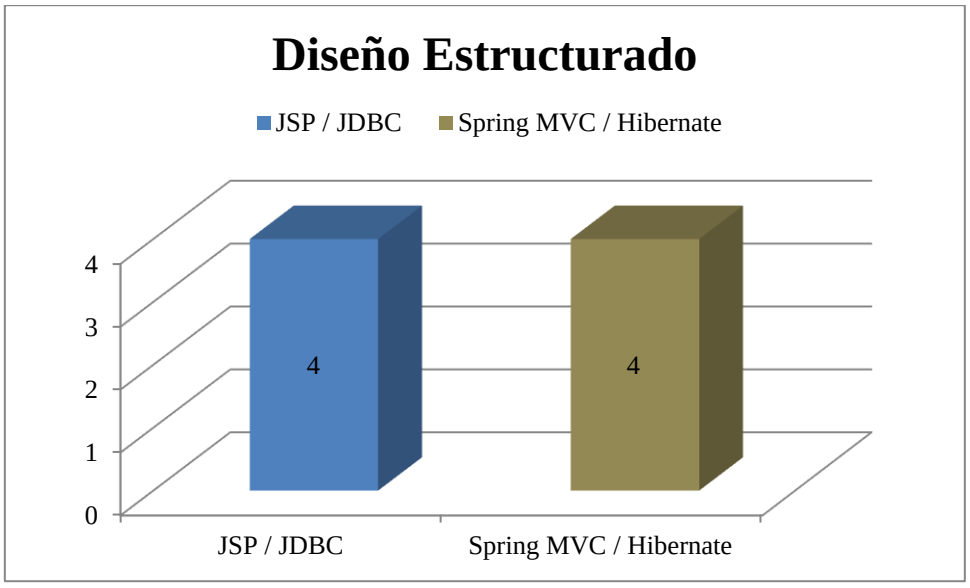


Figura V.23. Índice de comparación de Diseño Estructurado.

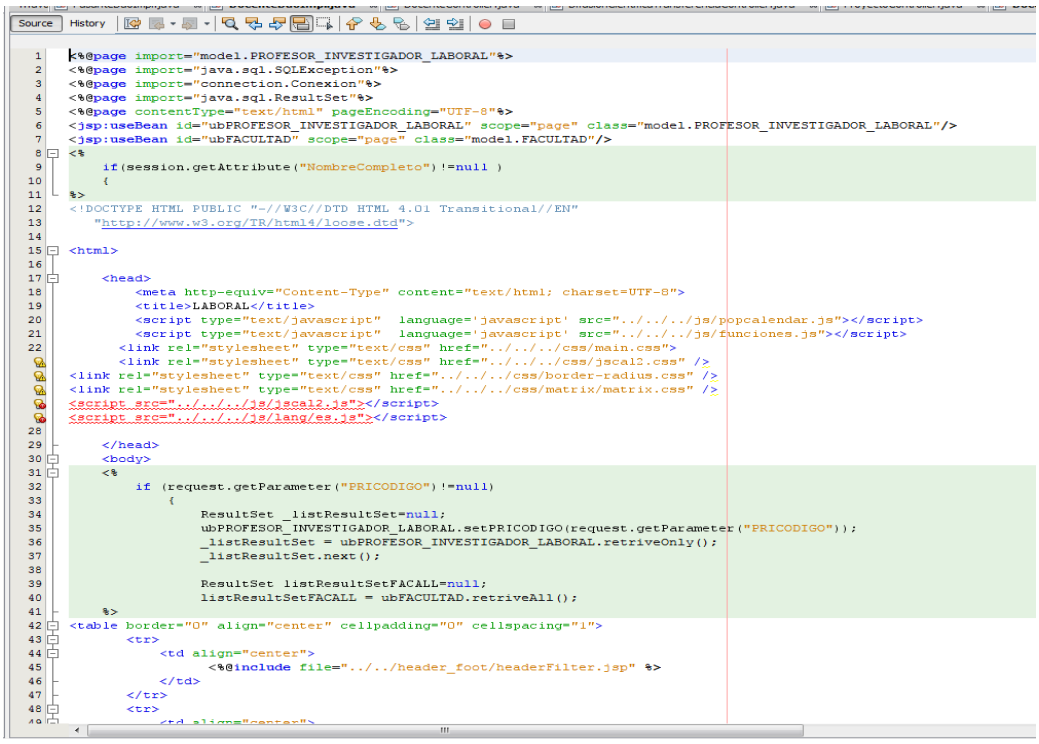
5.1.1.4.3.3 Capacidad de descomposición

Índice que mide el tiempo que se ahorra cuando se hace una aplicación web, mediante la capacidad de descomponer el trabajo en módulos independientes, también se toma en consideración a la hora de unificar los módulos y obtener la aplicación total sin complicaciones.

La aplicación en una tecnología JSP / JDBC utiliza un concepto claro de descomposición, utilizado mediante paquetes, clases, páginas web; sin embargo en las páginas web se mezcla las paginas HTML y paginas JSP, haciendo esto que sea un tanto difícil a la hora de la unificación de módulos.

Por su contraparte la tecnologíaSPRING / HIBERNATE tiene bien marcada la separación de los que son las paginas JSP/XML.

Prototipo 1:



```
<%@page import="model.PROFESOR_INVESTIGADOR_LABORAL"%>
<%@page import="java.sql.SQLException"%>
<%@page import="connection.Conexion"%>
<%@page import="java.sql.ResultSet"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<jsp:useBean id="ubPROFESOR_INVESTIGADOR_LABORAL" scope="page" class="model.PROFESOR_INVESTIGADOR_LABORAL"/>
<jsp:useBean id="ubFACULTAD" scope="page" class="model.FACULTAD"/>
<%
    if(session.getAttribute("NombreCompleto")!=null )
    {
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>

    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>LABORAL</title>
        <script type="text/javascript"  language='javascript' src="../../../js/popcalendar.js"></script>
        <script type="text/javascript"  language='javascript' src="../../../js/funciones.js"></script>
      <link rel="stylesheet" type="text/css" href="../../../css/main.css">
      <link rel="stylesheet" type="text/css" href="../../../css/jscal2.css" />
<link rel="stylesheet" type="text/css" href="../../../css/border-radius.css" />
<link rel="stylesheet" type="text/css" href="../../../css/matrix/matrix.css" />
<script src="../../../js/jscal2.js"></script>
<script src="../../../js/lang/es.js"></script>

    </head>
    <body>
    <%
        if (request.getParameter("PRICODIGO")!=null)
            {
                ResultSet _listResultSet=null;
                ubPROFESOR_INVESTIGADOR_LABORAL.setPRICODIGO(request.getParameter("PRICODIGO"));
                _listResultSet = ubPROFESOR_INVESTIGADOR_LABORAL.retriveOnly();
                _listResultSet.next();

                ResultSet listResultSetFACALL=null;
                listResultSetFACALL = ubFACULTAD.retriveAll();
    %>
<table border="0" align="center" cellpadding="0" cellspacing="1">
        <tr>
            <td align="center">
                <%@include file="../../header_foot/headerFilter.jsp" %>
            </td>
        </tr>
        <tr>
```

Figura V.24. Página HTML Prototipo 1.

Prototipo 2:

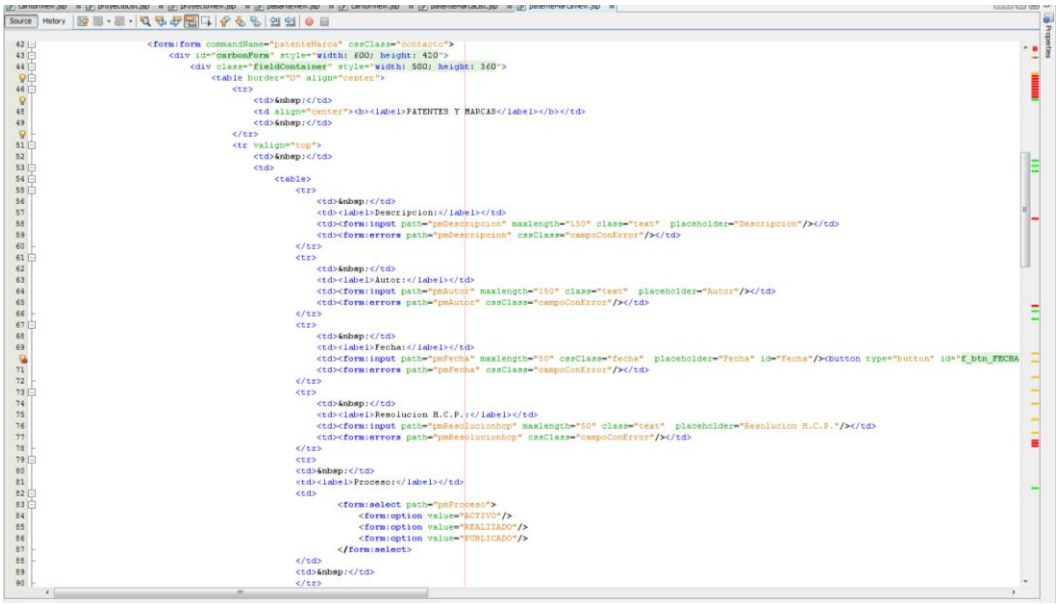


Figura V.25. Página HTML Prototipo 2.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla V.XXIII. Evaluación del Índice de Capacidad de Descomposición.

Indicador \ Tecnologías	Prototipo 1	Prototipo 2
	JSP / JDBC	Spring MVC / Hibernate
Criterio de evaluación	2	4
Porcentaje	50%	100%

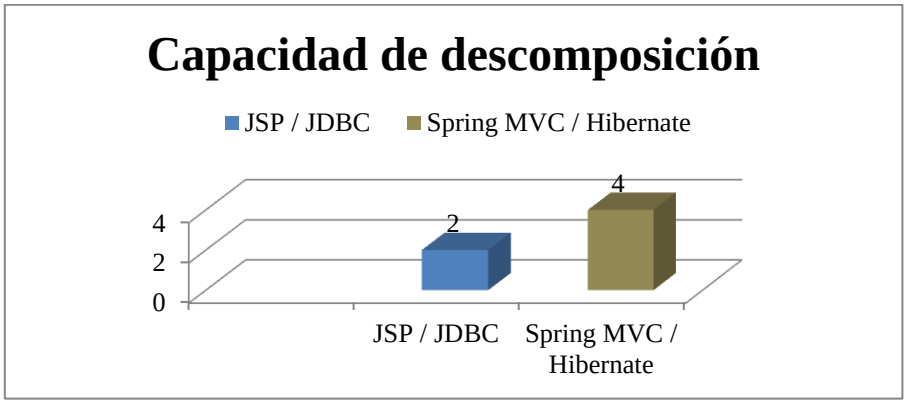


Figura V.26. Índice de comparación de Capacidad de descomposición.

5.1.1.4.3.4 Comprensión

Es mejorar la facilidad de comprensión del código o cambiar su estructura y diseño y eliminar código muerto, para facilitar el mantenimiento en el futuro. Anadir nuevo comportamiento a un programa puede ser difícil con la estructura dada del programa, así que un desarrollador puede reautorizarlo primero para facilitar esta tarea y luego añadir el nuevo comportamiento.

Este índice sirve para medir la optimización del tiempo de desarrollo de una aplicación web especialmente a la hora de mantener la página.

En el desarrollo de la investigación, se ha notado que el prototipo 1 realizado por la tecnología JSP / JDBC en sus páginas de presentación mezclan código HTML con código java, esto hace que la comprensión del funcionamiento sea muy difícil, como se muestra en la figura.

Prototipo 1:

```
<tr>
  <th scope="row"><div align="left">Subárea del conocimiento:</div></th>
  <td><div align="left">
        <select id="Sistema de Nivelación y Admisión" name="SUBCODIGO">
            <option  value="<%=resultSetSubArea.getString("SUBCODIGO")%>" > <%=resultSetSubArea.getString("SUBAREA") %> </option>
           <%
           while (resultSetSubAreaALL.next()) {
           if (resultSetSubArea.getString("SUBCODIGO").equals (resultSetSubAreaALL.getString("SUBCODIGO"))) {
               resultSetSubAreaALL.next();
           }%>
        <option  value="<%=resultSetSubAreaALL.getString("SUBCODIGO")%>"><%=resultSetSubAreaALL.getString("SUBAREA")%></option>
        <%}%>
        </select>
        </div></td>
</tr>
```

Figura V.27. Comprensión de Código Prototipo 1.

Se observa en laFigura V.27como el código java es insertado en páginas HTML, esto hace muy difícil la comprensión de la estructuración de la página HTML.

No así la tecnología SPRING / HIBERNATE en el cual se facilita la comprensión de código, ya que la lógica se la realiza en el bean, como se muestra en la figura.

Prototipo 2:

```
</tr>
 <tr>
    <td> </td>
    <td><label>Numero de Patente:</label></td>
    <td><form:input path="pmNumpatente" maxlength="50" cssClass="numero"  placeholder="Numero Patente"/></td>
    <td><form:errors path="pmNumpatente" cssClass="campoConError"/></td>
</tr>
<tr>
<td> </td>
<td><label>Producto:</label></td>
<td>
        <form:select path="producto.prdCodigo">
                <form:options items="${productosTypes}" itemLabel="prdNombre" itemValue="prdCodigo"/>
        </form:select>
</td>
<td> </td>
</tr>
<table>
<tr>
    <td> </td>
    <td align="center">
        <input type="submit" id="Guardar" name="Guardar" value="Guardar"/>
    </td>
    <td align="center">
        <c:if test="${patenteMarca.pmCodigo!=0 }">
            <input type="button" id="Eliminar" name="Eliminar"
                   onclick="location.href='<c:url value="deletePatenteMarca.htm?dciCodigo=${patenteMarca.pmCodigo}"/>'"
                   value="Eliminar"/>
        </c:if>
        <input type="button" id="Cancelar" name="Cancelar"
               onclick="location.href='<c:url value="patenteMarcaList.htm"/>'"
               value="Cancelar"/>
    </td>
    <td> </td>
</tr>
```

Figura V.28. Compresión de Código del Prototipo 2.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla V.XXIV. Evaluación del Índice de Comprensión.

Tecnologías / Indicador	Prototipo 1	Prototipo 2
	JSP / JDBC	Spring MVC / Hibernate
Criterio de evaluación	2	4
Porcentaje	50%	100%

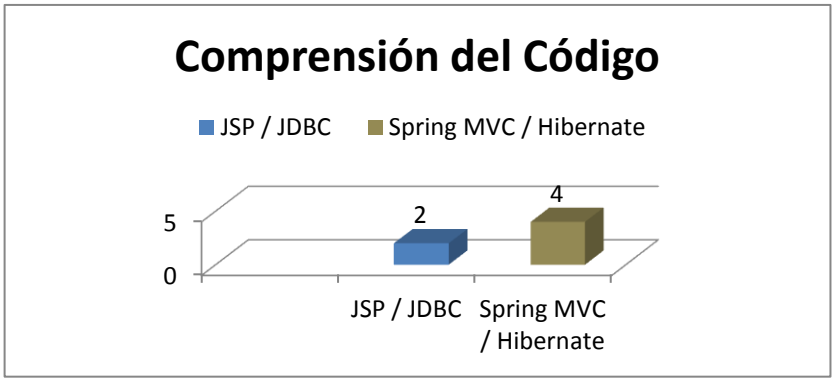


Figura V.29. Índice de comparación de Compresión de Código.

Como se observa en la Figura V.29 se ahorra un 50 % de tiempo más; a la hora de entender el código y realizar el mantenimiento de las aplicaciones web con el uso de la tecnología SPRING / HIBERNATE.

Tabla V.XXV. Resultados globales del índice de Modularidad.

Tecnologías / Indicador	JSP / JDBC	Spring MVC / Hibernate	Peso Máximo
Modelo Vista Controlador	1	4	4
Diseño Estructurado	4	4	4
Capacidad de Descomposición	2	4	4
Comprensión del Código	2	4	4
Total	9	16	16
Porcentaje de generación de código	56,25%	100,00%	100,00%

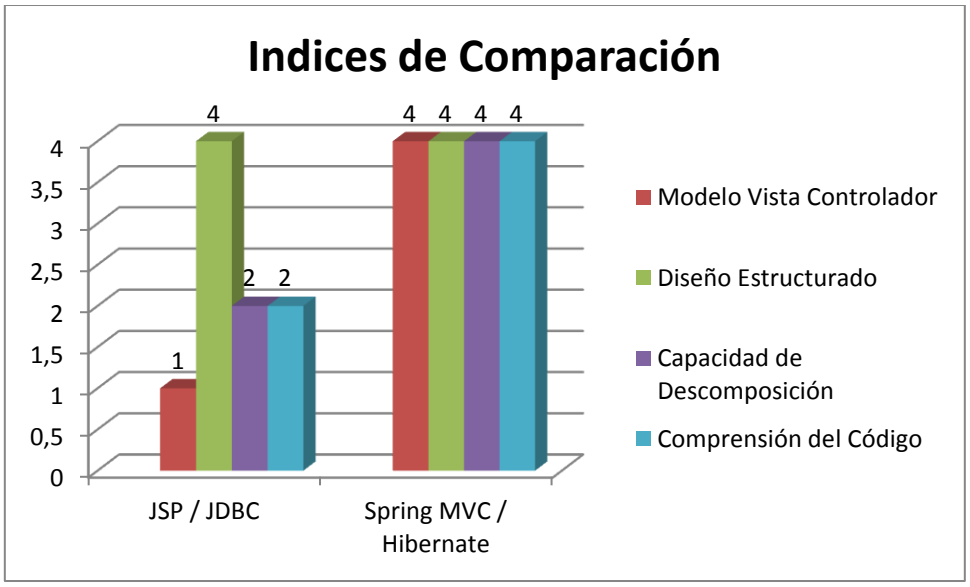


Figura V.30. Resultados del Índice de Comparación de Modularidad.

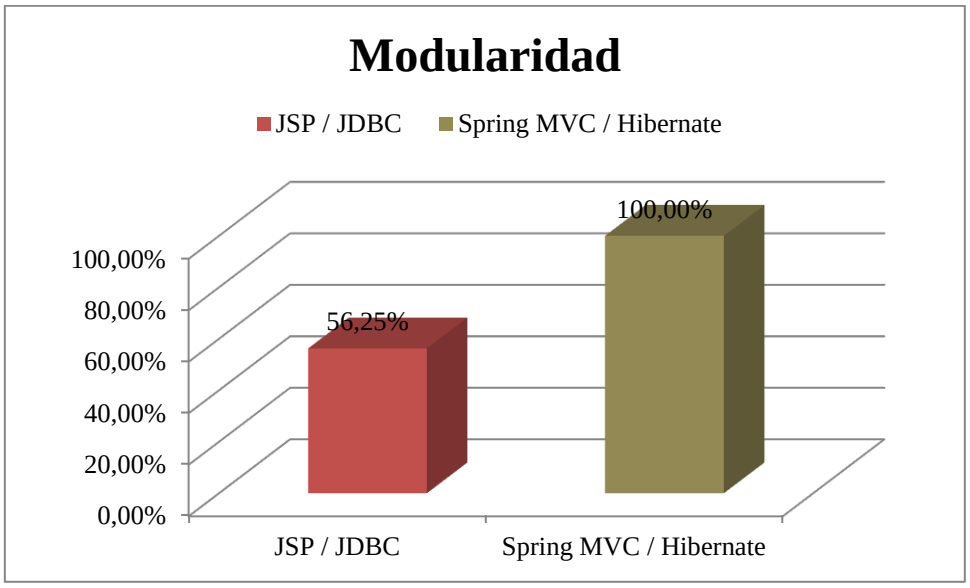


Figura V.31. Porcentajes totales de Modularidad

5.1.1.4.3.5 Interpretación

Se puede observar en la Figura V.31 que el parámetro Modularidad tiene un 100% de optimización de tiempo, usando la tecnología SPRING / HIBERNATE, permitiendo adaptar los beneficios que ofrece este parámetro, no así el prototipo que utiliza la tecnología JSP / JDBC que ha calificado con 56,25% de optimización de tiempo en el parámetro Modularidad.

5.1.1.4.4 Mantenibilidad

5.1.1.4.4.1 Escalabilidad

La escalabilidad es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para reaccionar y adaptarse sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

En el desarrollo de aplicaciones web es común cambios de requerimientos como también adaptación de nuevos requerimientos, es muy importante tener en cuenta este factor ya que si la aplicación es escalable que ahorrar tiempo en la construcción de los nuevos requerimientos.

TecnologíasJSP/JDBC.

La tecnología JSP / JDBC permite la escalabilidad de una manera no óptima ya que no tiene separado claramente la lógica de negocios con la lógica de presentación y lógica de acceso a datos. Para lo cual en la adaptación de nuevos requerimientos se debe realizar una re-estructuración de todo el código.

Tecnologías spring mvc / hibernate.

La tecnología spring / hibernate simplifica las cosas como ya que tiene separada claramente toda la lógica, de manera que es escalable fácilmente.

De acuerdo con este análisis se establecen los siguientes valores.

Se observa los resultados obtenidos para el índice de Escalabilidad, en el cual el prototipo 1, tiene un porcentaje medio de escalabilidad frente al prototipo 2 que es desarrollada por la tecnología SPRING / HIBERNATE.

Tabla V.XXVI. Evaluación del Índice de Escalabilidad.

Tecnologías / Indicador	Prototipo 1	Prototipo 2
	JSP / JDBC	Spring MVC / Hibernate
Criterio de evaluación	2	4
Porcentaje	50%	100%

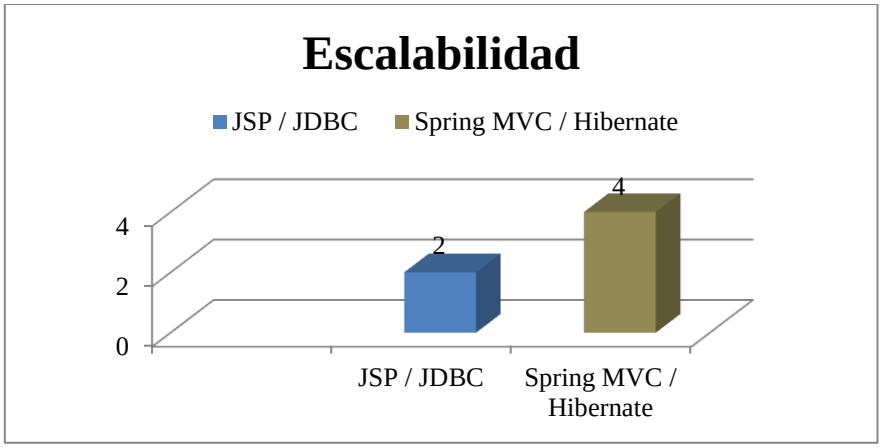


Figura V.32. Índice de comparación de Escalabilidad.

En la Figura V.32 , se observa los resultados obtenidos para el índice de Escalabilidad, en el cual el prototipo 1, tiene porcentaje medio de escalabilidad frente al prototipo 2 que presta mayores ventajas en este índice.

5.1.1.4.4.2 Disponibilidad de documentación

Mediante este indicador se puede saber la cantidad en porcentaje de tiempo ahorrado, a la hora de buscar información ya sea en foros, libros, wikis, etc, para la construcción o mantenibilidad de la aplicación web.

Se puede decir que utilizando la tecnología JSP / JDBC se ahorra mucho tiempo a la hora de buscar información para la construcción de aplicaciones web, ya que esta utiliza técnicas estables y conocidas, lo cual hace que haya mucha disponibilidad de documentación, no así en la tecnología SPRING / HIBERNATE ya que utiliza técnicas nuevas y no se tendrá mucha documentación.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla V.XXVII. Evaluación del Índice de Disponibilidad de documentación

Tecnologías / Indicador	Prototipo 1	Prototipo 2
	JSP / JDBC	Spring MVC / Hibernate
Criterio de evaluación	4	1
Porcentaje	100%	25%

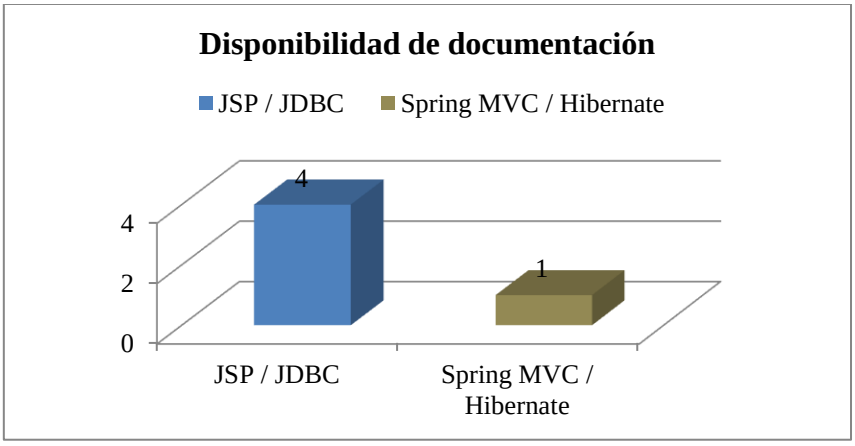


Figura V.33. Índice de comparación de Disponibilidad de documentación.

En la se puede observar en laFigura V.33 que la tecnología SPRING / HIBERNATE no ahorra tiempo a la hora de buscar documentación, a lo contrario de la tecnología JSP / JDBC que ahorra un 100% de tiempo a la hora de buscar documentación para el desarrollo de aplicaciones web.

5.1.1.4.4.3 Código entendible

Es un índice muy importante que se toma en consideración principalmente a la hora de la manteniblidad de las aplicaciones web. Mide que tan entendible es el código que se ha utilizado en la construcción de las aplicaciones web.

Utilizando la tecnología JSP / JDBC los desarrolladores de software están obligados a escribir todo el código razón por la cual es más entendible a la hora de la manteniblidad. Como se muestra en la figura de un ejemplo para llamar un Procedimiento almacenado.

```
151
152     /**
153      * Este metodo insertar un registro de PROFESOR_INVESTIGADOR_PRINCIPA
154      * @return no retorna un dato tifo byte
155      */
156     public byte create(){
157         byte bytEstado=0;
158         CallableStatement ejecutar;
159         try{
160             ejecutar = this.conexion.prepareCall("{call SP_INSERT_PROFESOR_INV_PRINC(?,?,?,?,?,?,?,?,?)}");
161
162             ejecutar.setString (1,PRICODIGO);
163             ejecutar.setString (2,PRICEDULA);
164             ejecutar.setString (3,PRIAPELLIDO_PATERNO);
165             ejecutar.setString (4,PRIAPELLIDO_MATERNO);
166             ejecutar.setString (5,PRINOMBRES);
167             ejecutar.setString (6,PRISEXO);
168             ejecutar.setString (7,PRIFECHA_NACIMIENTO);
169             ejecutar.setString (8,SEDCODIGO);
170             ejecutar.setString (9,NACCODIGO);
171             ejecutar.execute();
172             ejecutar.close();
                ejecutar = null;
174             bytEstado = 1;
175         }
176         catch(SQLException e){
177             this.strError = e.getMessage().toString();
178         }
179         return bytEstado;
180     }
```

Figura V.34. Código de Método Insertar Prototipo 1.

Utilizando la tecnología JSP / JDBC la mayor parte de código es generado por frameworks, razón por la cual existe mayor código no entendible a la hora de la construcción y manteniblidad de aplicaciones Web Ejemplo en el gráfico se muestra código generado por frameworks.

```
53
54      public void saveDocente(Docente docente) {
55          hibernateTemplate.saveOrUpdate(docente);
56      }
57
```

Figura V.35. Código de Método Insertar Prototipo 2.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla V.XXVIII. Evaluación del Índice de Código Entendible

Indicador \ Tecnologías	Prototipo 1	Prototipo 2
	JSP / JDBC	Spring MVC / Hibernate
Criterio de evaluación	4	3
Porcentaje	100%	75%

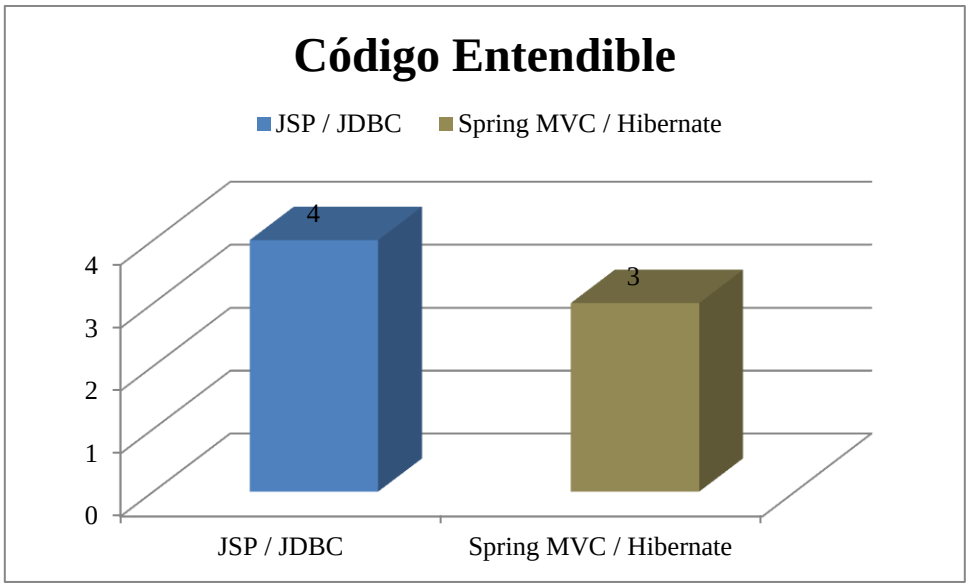


Figura V.36. Índice de comparación de Código Entendible.

En la Figura V.36 se observa que el prototipo realizado con la tecnología JSP / JDBC tiene como porcentaje 100%, esto quiere decir que todo su código es entendible y favorece a la hora de re-estructurar algunos requerimientos, también se observa que el prototipo que utiliza la tecnología SPRING / HIBERNATE tiene 75% pretendiendo decir que hace un poco difícil la comprensión del código.

5.1.1.4.4.4 Adaptabilidad de nuevos requerimientos

Los requerimientos pueden dividirse en requerimientos funcionales y requerimientos no funcionales. Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas.

Los requerimientos no funcionales tienen que ver con características que forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc.

La capacidad de introducir nuevos requerimientos sin la dificultad a la hora de programar, por ende ganar tiempo y recursos.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla V.XXIX. Evaluación del Índice de Adaptabilidad de nuevos Requerimientos.

Tecnologías / Indicador	**Prototipo 1**	**Prototipo 2**
	JSP / JDBC	**Spring MVC / Hibernate**
Criterio de evaluación	2	4
Porcentaje	50%	100%

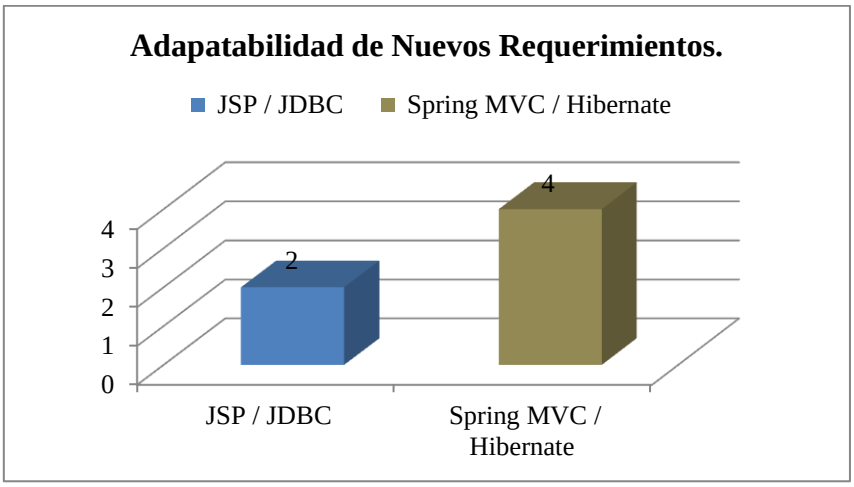


Figura V.37. Índice de comparación de Adaptabilidad de nuevos Requerimientos.

Como se observa en la Figura V.37 el prototipo realizado con la tecnología SPRING / HIBERNATE ofrece el doble de productividad en lo que tiene que ver con el tiempo, en la adaptabilidad de Nuevos Requerimientos frente al prototipo realizado con la tecnología JSP / JDBC.

Tabla V.XXX. Resultados globales del índice de Mantenibilidad.

Tecnologías / Indicador	JSP / JDBC	Spring MVC / Hibernate	Peso Máximo
Escalabilidad	2	4	4
Disponibilidad de Documentación	4	1	4
Código Entendible	4	3	4
Adaptabilidad de nuevos requerimientos	2	4	4
Total	12	12	16
Porcentaje de generación de código	75,00%	75,00%	100,00%

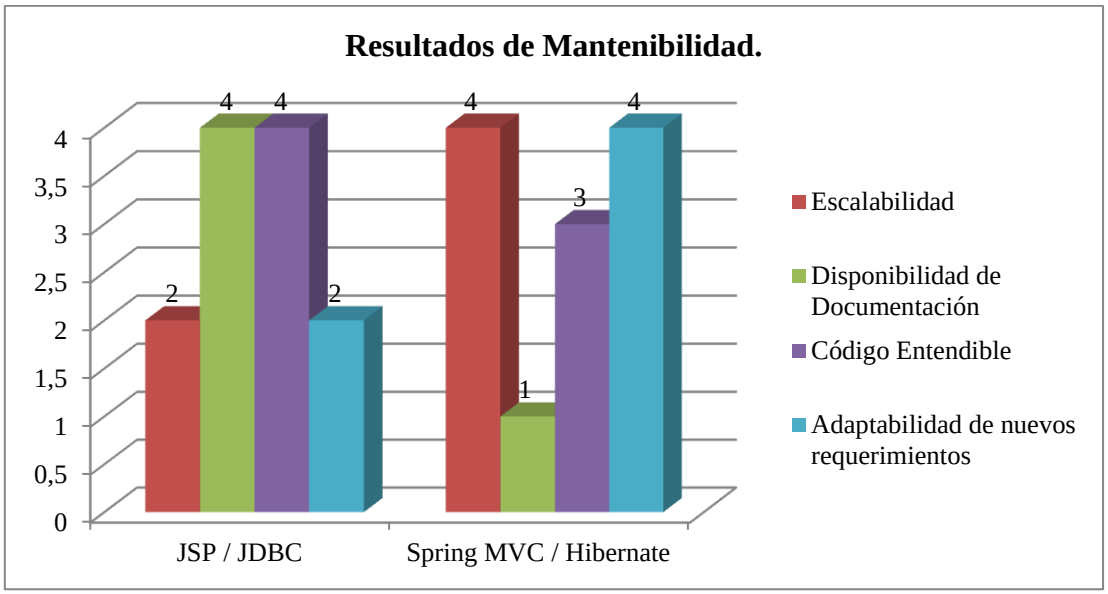


Figura V.38. Resultados del Índice de comparación de Mantenibilidad.

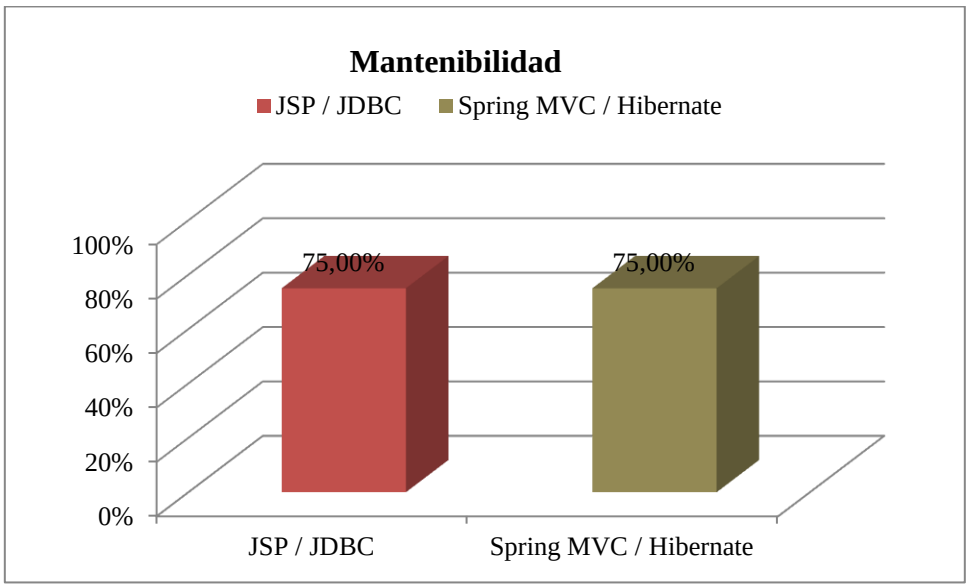


Figura V.39. Porcentajes totales de Mantenibilidad.

5.1.1.4.4.5 Interpretación

En este parámetro como es el de Manteniblidad se puede observar en la Figura V.39 que tanto la tecnología JSP / JDBC como la tecnología SPRING / HIBERNATE tienen un mismo porcentaje de ahorro de tiempo en cuanto tiene que ver con la construcción y adaptación de nuevos requerimientos en una aplicación web.

Debido a los indicadores de: “Documentación Disponible” y “Código entendible”, la tecnología JSP / JDBC ofrece varias ventajas frente a la tecnología SPRING / HIBERNATE, sin embargo los indicadores de: “Escalabilidad” y “Adaptación de Nuevos Requerimientos” trae desventajas frente a la tecnología SPRING / HIBERNATE.

5.1.1.4.5 Presentación

5.1.1.4.5.1 Templates personalizados

El uso de templating trae consigo las ventajas de tener un código ordenado (sobre todo en aplicaciones grandes), fácil de desarrollar y reusar. Se puede crear componentes o templates que abarcan desde simplemente mostrar texto hasta crear componentes que muestren un checkbox, un input o un dropdown dependiendo de la data del backing bean.

En la tecnología JSP / JDBC se puede desarrollar los Templates mediante hojas de estilo, no así en la tecnología SPRING / HIBERNATE que ofrece Templates personalizados que benefician en el tiempo a la hora de la creación de los Templates.

Templates en Tecnología JSP / JDBC

La tecnología JSP / JDBC para el uso de Templates hace uso de un sin número de técnicas tales como son jquery, javascript, frames, etc.

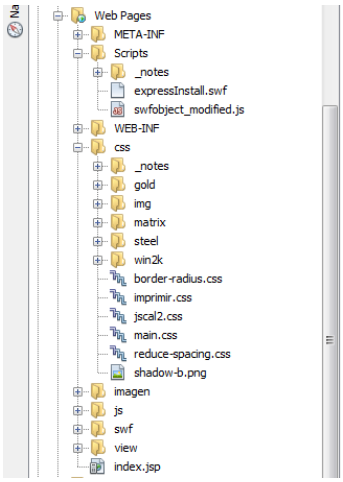


Figura V.40. Hojas de Estilo prototipo 1

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <script type="text/javascript"  language='javascript' src="../../../js/funciones.js"></script>
        <link rel="stylesheet" type="text/css" href="../../../css/main.css">
    </head>
```

Figura V.41. Referencias - Hojas de Estilo prototipo 1.

Templates en Tecnología Spring

La tecnología spring / hibernate simplifica las cosas como se muestra a continuación.

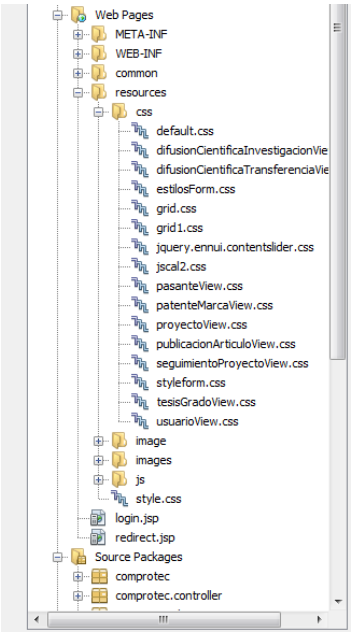


Figura V.42. Hojas de Estilo prototipo 2.

```
<html>
    <head>
        <title>Particle</title>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <link href="<c:url value="/resources/style.css"/>" rel="stylesheet" type="text/css" />
        <link href="<c:url value="/resources/css/jquery.ennui.contentslider.css"/>" rel="stylesheet" type="text/css" media="screen,projection" />
        <script src="<c:url value="/resources/js/jquery.watermark.js"/>" type="text/javascript"></script>
        <link href="<c:url value="/resources/css/grid.css"/>" rel="stylesheet" type="text/css"  />
        <script src="<c:url value="/resources/js/gridjquery.js"/>" type="text/javascript"></script>
        <script src="<c:url value="/resources/js/grid.js"/>" type="text/javascript"></script>
        <script src="<c:url value="/resources/js/js/kendo.grid.min.js"/>" type="text/javascript"></script>
        <script language="javascript" type="text/javascript">
```

Figura V.43. Referencias - Hojas de Estilo prototipo 1.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla V.XXXI. Evaluación del Índice de Presentación

Indicador \ Tecnologías	Prototipo 1	Prototipo 2
	JSP / JDBC	Spring MVC / Hibernate
Criterio de evaluación	3	4
Porcentaje	75%	100%

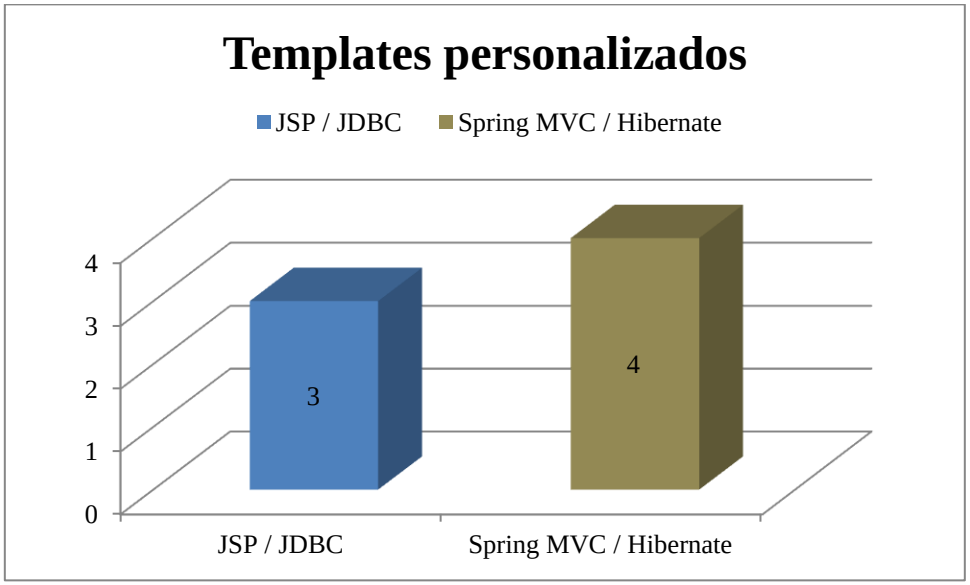


Figura V.44. Índice de comparación de Templates Personalizados

En la Figura V.44, se observa los resultados obtenidos para el índice de Personalizados, en el cual el prototipo 2, que utiliza la tecnología SPRING / HIBERNATE, con el valor de 4 toma ventaja sobre el prototipo 1, el cual utiliza la tecnología JSP / JDBC, esto significa que utilizando los Templates

personalizados que ofrece la tecnología SPRING / HIBERNATE se puede obtener mayor ventaja en el ahorro de tiempo en la construcción de los mismos.

5.1.1.4.5.2 Componentes JQuery y JavaScript

Es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

JavaScript es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

Ajax permite cargar la página en segundo plano sin actualizar la página completa y la presentación. Ajax es un script simple JavaScript que trabajar con XML petición HTTP.

jQuery consiste en un único fichero JavaScript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX. La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX. Para ello utiliza las funciones $() o jQuery().

Tecnología (JSP).

Para realizar componentes jquery, ajax con jsp, se debe implementar un sin número de clases y métodos, haciendo así a la aplicación más demorosa, como se muestra en las figuras siguientes.

```
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Docente</title>
        <script type="text/javascript"  language='javascript' src="../../../js/popcalendar.js"></script>
        <script type="text/javascript"  language='javascript' src="../../../js/funciones.js"></script>
      <link rel="stylesheet" type="text/css" href="../../../css/main.css">
      <link rel="stylesheet" type="text/css" href="../../../css/jscal2.css" />
<link rel="stylesheet" type="text/css" href="../../../css/border-radius.css" />
<link rel="stylesheet" type="text/css" href="../../../css/matrix/matrix.css" />
<script type="text/javascript"  src="../../../js/jscal2.js"></script>
<script type="text/javascript" src="../../../js/lang/es.js"></script>
```

Figura V.45. JavaScript en JSP.

```
600
601  <script type="text/javascript">//<![CDATA[
602
603          var cal = Calendar.setup({
604              onSelect: function(cal) { cal.hide() },
605              showTime: true
606          });
607
608          cal.manageFields("f_btn_ACADFECHA_APROBACION", "Fecha de Aprobación de la Extensión", "%d-%m-%Y");
609  //]]></script>
610      </body>
611  </html>
```

Figura V.46. Función JavaScript.

Tecnología (Spring Web MVC 3.0)

Para implementar Jquery con **Spring Web MVC 3.0**, se hace más difícil ya en spring se debe configurar la ubicación del paquete de recursos y como vemos la llamada utiliza un taglib lo que su ves genera confusión para un aprendiz, como se muestra en las siguientes figuras.

```
Source  History
1   <%@page contentType="text/html" pageEncoding="UTF-8"%>
2   <%@taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
3   <%@taglib prefix="spring" uri="http://www.springframework.org/tags"%>
4   <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5   <%
6       if (session.getAttribute("usuario")!=null)
7   {
8       %>
9       <html>
10  <head>
11  <title>Particle</title>
12  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
13  <link href="<c:url value="/resources/style.css"/>" rel="stylesheet" type="text/css" />
14  <link href="<c:url value="/resources/css/jquery.ennui.contentslider.css"/>" rel="stylesheet" type="text/css" media="screen,projection" />
15  <script src="<c:url value="/resources/js/jquery.watermark.js"/>" type="text/javascript"></script>
```

Figura V.47. Java Script en HTML Prototipo 2.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla V.XXXII. Evaluación del Índice Componentes JQuery y JavaScript.

Indicador \ Tecnologías	Prototipo 1	Prototipo 2
	JSP / JDBC	Spring MVC / Hibernate
Criterio de evaluación	1	3
Porcentaje	25%	75%

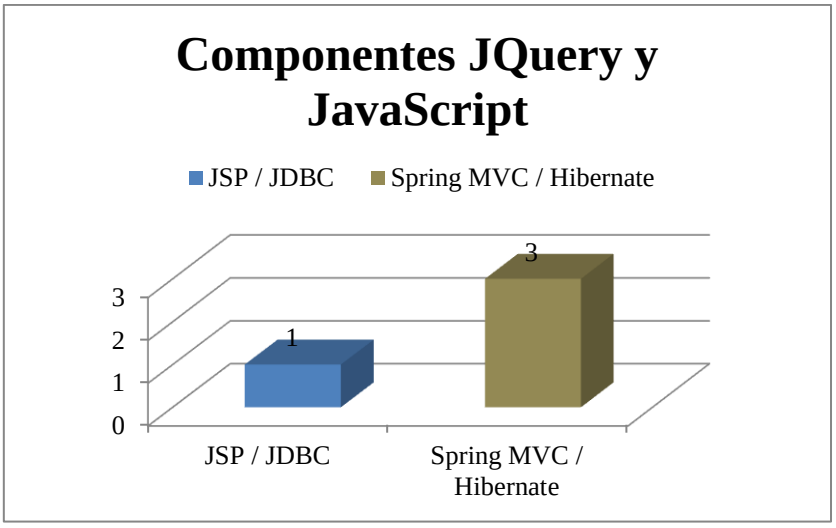


Figura V.48. Índice de comparación de Componentes JQuery y JavaScript

En la Figura V.48, se observa los resultados obtenidos para el índice de JQuery y JavaScript, se puede concluir diciendo que ambas tecnólogas utilizan JQuery y JavaScript, sin embargo, la tecnología JSP / JDBC hace se usó de forma más extensa y difícil, no así la tecnología SPRING / HIBERNATE en la cual su estructura declara un paquete de recursos, tomando en cuenta que hay que configurar dicho paquete.

5.1.1.4.5.3 Validaciones

La validación se define como: El proceso de comprobación de si algo satisface un cierto criterio. Los ejemplos incluyen la comprobación de si una afirmación es verdadera (la validez), si un aparato funciona como está previsto, si un sistema es seguro, o si los datos informáticos son compatibles

con un estándar abierto. La validación implica un documento, puede que la solución o el proceso es correcto o es adecuado para su uso.

La validación de HTML y CSS es un proceso que no es impuesto sobre un diseñador /desarrollador. Sin embargo, se realiza con el fin de asegurar la compatibilidad y la estabilidad de una página web a través de los distintos navegadores. El defensor número de la validación es el W3C (World Wide Web Consortium). La validación es la mejor manera de evitar diversas irregularidades en la prestación de una página web.

Validaciones Tecnología JSP

Se utiliza validaciones usando javascript, no tiene mensajes de error personalizados.

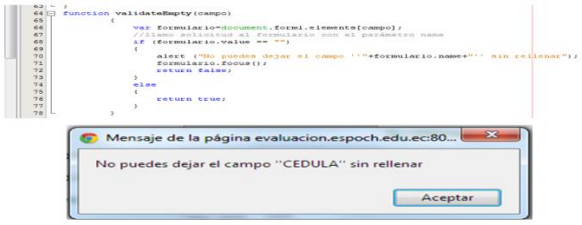


Figura V.49 Validaciones prototipo 1.

Validaciones Tecnología SPRING / HIBERNATE.

Posee validaciones personalizadas y mensajes de error personalizados.

```
<tr>
    <td> </td>
    <td><label>Fecha de Inicio:</label></td>
    <td><form:input path="proFechainicio" maxlength="50" cssClass="fecha"  placeholder="Fech
    <td><form:errors path="proFechainicio" cssClass="campoConError"/></td>
</tr>
<tr>
    <td> </td>
    <td><label>Fecha de Fin:</label></td>
    <td><form:input path="proFechafin" maxlength="50" cssClass="fecha"  placeholder="Fecha"
    <td><form:errors path="proFechafin" cssClass="campoConError"/></td>
</tr>
<tr>
    <td> </td>
    <td><label><pre>Investigadores Adicionales:</pre></label></td>
    <td><form:input path="proInvadicionales" maxlength="50" class="text" placeholder="Invest
    <td><form:errors path="proInvadicionales" cssClass="campoConError"/></td>
</tr>
<tr>
    <td> </td>
    <td><label><pre>Entidades Auspiciantes:</pre></label></td>
    <td><form:input path="proEntidadesauspiciantes" maxlength="50" class="text" placeholder=
    <td><form:errors path="proEntidadesauspiciantes" cssClass="campoConError"/></td>
</tr>
<tr>
    <td> </td>
    <td><label><pre>Presupuesto:</pre></label></td>
    <td><form:input path="proPresupuesto" maxlength="50" class="text" placeholder="Presupue
    <td><form:errors path="proPresupuesto" cssClass="campoConError"/></td>
</tr>
<tr>
    <td> </td>
    <td><label><pre>Plan del Buen Vivir:</pre></label></td>
    <td><form:input path="proPlanbuenvivir" maxlength="50" class="text" placeholder="Presupu
    <td><form:errors path="proPlanbuenvivir" cssClass="campoConError"/></td>
</tr>
<tr>
    <td> </td>
    <td><label><pre>Anexo:</pre></label></td>
    <td><form:input path="proAnexo" maxlength="50" class="text" placeholder="Anexo"/></td>
    <td><form:errors path="proAnexo" cssClass="campoConError"/></td>
```

```
@Component
public class ProyectoValidator implements Validator{

    @Autowired
    private ProyectoService proyectoService;

    public void setProyectoService(ProyectoService proyectoService) {
        this.proyectoService = proyectoService;
    }

    @SuppressWarnings("rawtypes")
    public boolean supports(Class clazz) {
        return Proyecto.class.isAssignableFrom(clazz);
    }

    public void validate(Object object, Errors errors) {
        Proyecto proyecto = (Proyecto) object;
        validateAnexo(proyecto, errors);
        validateEntidadesAuspiciantes(proyecto, errors);
        validateInvestigadoresAdicionales(proyecto, errors);
        validateNombre(proyecto, errors);
        validatePlanBuenVivir(proyecto, errors);
        validatePresupuesto(proyecto, errors);
//        validateArchivo(proyecto, errors);
    }

    private void validateNombre(Proyecto proyecto, Errors errors) {
        if (proyecto.getProNombre() == null
                || StringUtils.isEmpty(proyecto.getProNombre())) {
            errors.rejectValue("proNombre", "proyecto.proNombre.required");
        }
    }

    private void validateInvestigadoresAdicionales(Proyecto proyecto, Errors errors) {
        if (proyecto.getProInvadicionales() == null
                || StringUtils.isEmpty(proyecto.getProInvadicionales())) {
            errors.rejectValue("proInvadicionales", "proyecto.proInvadicionales.required");
        }
    }
```

Figura V.50. Validaciones prototipo 2.

Figura V.51. Interfaz de validación prototipo 2.

De acuerdo con este análisis se establecen los siguientes valores.

Tabla V.XXXIII. Evaluación del Índice Validaciones.

Tecnologías / Indicador	Prototipo 1	Prototipo 2
	JSP / JDBC	Spring MVC / Hibernate
Criterio de evaluación	3	4
Porcentaje	75%	100%

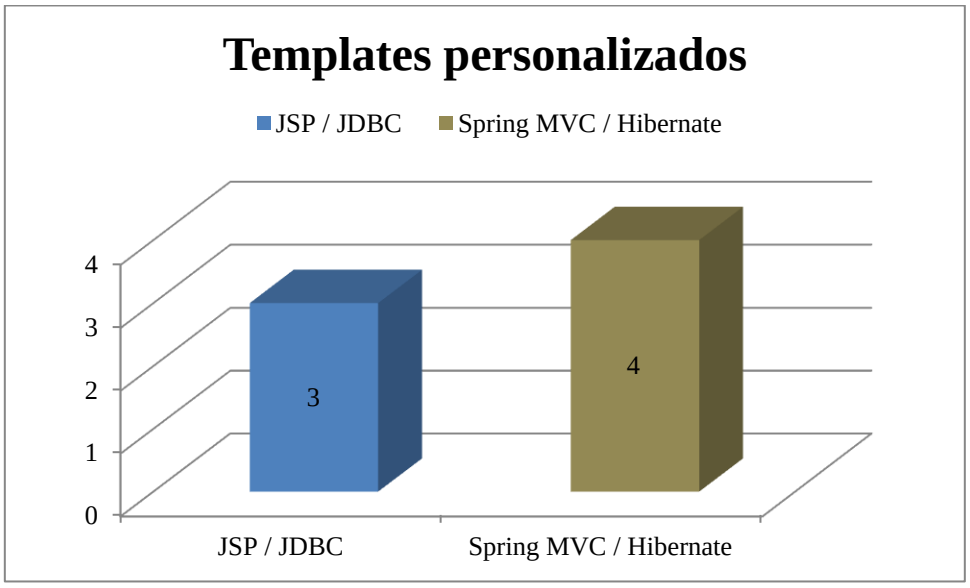


Figura V.52. Índice de comparación de Validaciones.

En la Figura V.52, se puede concluir diciendo que tanto la tecnología JSP / JDBC como la tecnología SPRING / HIBERNATE utilizan validaciones, pero la tecnología SPRING / HIBERNATE supera la tecnología JSP / JDBC, ya que trae incluidos mensajes personalizados de error con la utilización de las anotaciones.

5.1.1.4.5.4 Menor uso de java script

JavaScript, al igual que Flash, Visual Basic Script, es una de las múltiples maneras que han surgido para extender las capacidades del lenguaje HTML. Al ser la más sencilla, es por el momento la más extendida.

JavaScript no es un lenguaje de programación propiamente. Es un lenguaje script u orientado a documento, como pueden ser los lenguajes de macros que tienen muchos procesadores de texto y

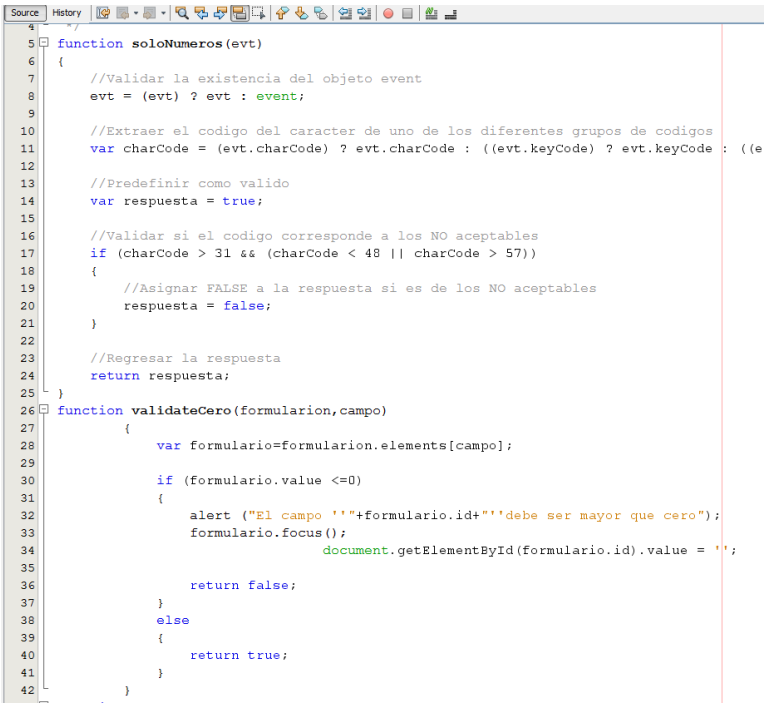
planillas de cálculo. No se puede desarrollar un programa con JavaScript que se ejecute fuera de un Navegador.

JavaScript es un lenguaje interpretado que se embebe en una página web HTML. Un lenguaje interpretado significa que a las instrucciones las analiza y procesa el navegador en el momento que deben ser ejecutadas.

Al utilizar JavaScript para realizar validaciones o para ejecutar menús desplegables, datepickers hace que el desarrollo de la aplicación sea mayor en tiempo; en cuanto tiene que ver en el mantenimiento y desarrollo de la aplicación web.

Es decir al utilizar menos java script en el desarrollo de una aplicación web, será más beneficioso en tiempo de desarrollo y construcción.

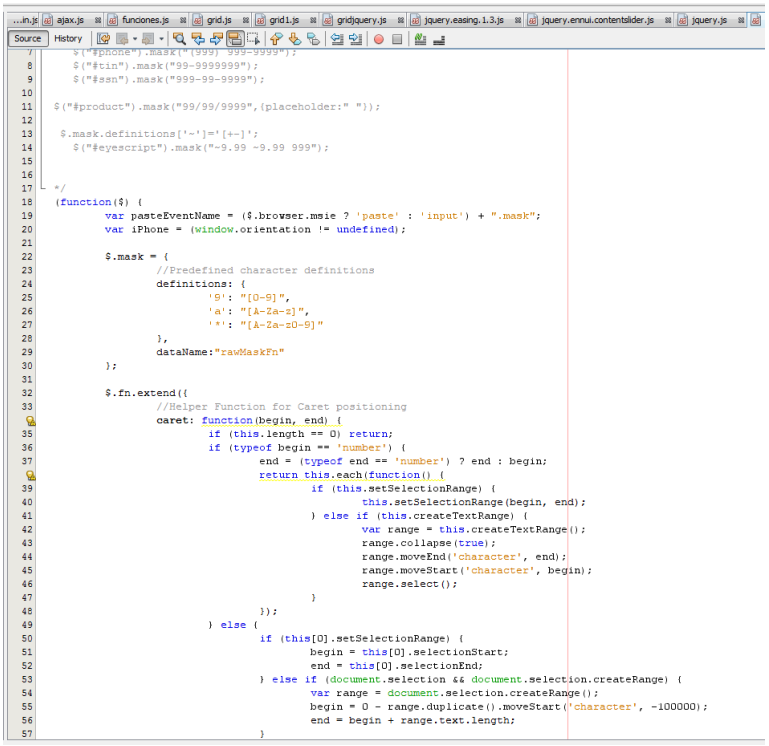
Java Script en Tecnología JSP



```
Source  History
 4   */
 5   function soloNumeros(evt)
 6   {
 7       //Validar la existencia del objeto event
 8       evt = (evt) ? evt : event;
 9
10       //Extraer el codigo del caracter de uno de los diferentes grupos de codigos
11       var charCode = (evt.charCode) ? evt.charCode : ((evt.keyCode) ? evt.keyCode : ((e
12
13       //Predefinir como valido
14       var respuesta = true;
15
16       //Validar si el codigo corresponde a los NO aceptables
17       if (charCode > 31 && (charCode < 48 || charCode > 57))
18       {
19           //Asignar FALSE a la respuesta si es de los NO aceptables
20           respuesta = false;
21       }
22
23       //Regresar la respuesta
24       return respuesta;
25   }
26   function validateCero(formularion,campo)
27           {
28               var formulario=formularion.elements[campo];
29
30               if (formulario.value <=0)
31               {
32                   alert ("El campo ''"+formulario.id+"''debe ser mayor que cero");
33                   formulario.focus();
34                                   document.getElementById(formulario.id).value = '';
35
36                   return false;
37               }
38               else
39               {
40                   return true;
41               }
42           }
```

Figura V.53. JavaScript prototipo 1.

Java Script en Tecnología SPRING.



```
$("#phone").mask("(999) 999-9999");
$("#tin").mask("99-9999999");
$("#ssn").mask("999-99-9999");

$("#product").mask("99/99/9999",{placeholder:" "});

$.mask.definitions['~']='[+-]';
$("#eyescript").mask("~9.99 ~9.99 999");


*/
(function($) {
        var pasteEventName = ($.browser.msie ? 'paste' : 'input') + ".mask";
        var iPhone = (window.orientation != undefined);

        $.mask = {
                //Predefined character definitions
                definitions: {
                        '9': "[0-9]",
                        'a': "[A-Za-z]",
                        '*': "[A-Za-z0-9]"
                },
                dataName:"rawMaskFn"
        };

        $.fn.extend({
                //Helper Function for Caret positioning
                caret: function(begin, end) {
                        if (this.length == 0) return;
                        if (typeof begin == 'number') {
                                end = (typeof end == 'number') ? end : begin;
                                return this.each(function() {
                                        if (this.setSelectionRange) {
                                                this.setSelectionRange(begin, end);
                                        } else if (this.createTextRange) {
                                                var range = this.createTextRange();
                                                range.collapse(true);
                                                range.moveEnd('character', end);
                                                range.moveStart('character', begin);
                                                range.select();
                                        }
                                });
                        } else {
                                if (this[0].setSelectionRange) {
                                        begin = this[0].selectionStart;
                                        end = this[0].selectionEnd;
                                } else if (document.selection && document.selection.createRange) {
                                        var range = document.selection.createRange();
                                        begin = 0 - range.duplicate().moveStart('character', -100000);
                                        end = begin + range.text.length;
                                }
```

Figura V.54. JavaScript prototipo 2.

De acuerdo con este análisis se establecen los siguientes valores

Tabla V.XXXIV. Evaluación del Índice Menor Uso de Java Script.

Indicador \ Tecnologías	Prototipo 1	Prototipo 2
	JSP / JDBC	Spring MVC / Hibernate
Criterio de evaluación	2	3
Porcentaje	50%	75%

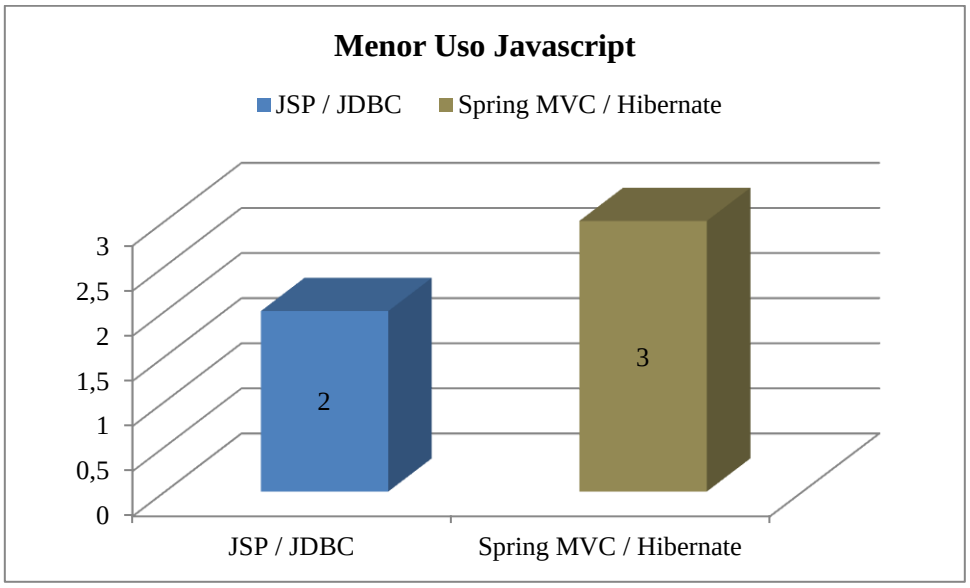


Figura V.55. Índice de comparación de Menor Uso Javascript

En la Figura V.55, se observa los resultados obtenidos para el índice de Menor Java Script, se puede notar que la tecnología SPRING / HIBERNATE es más óptima en cuanto tiene que ver con este índice ya que hace un uso casi nulo de Javascript y no así la tecnología JSP / JDBC que usa Javascript para realizar muchas de sus acciones.

Tabla V.XXXV. Resultados globales del índice de Presentación.

Tecnologías / Indicador	JSP / JDBC	Spring MVC / Hibernate	Peso Máximo
Templates personalizados	3	4	4
Componentes JQuery Javascript	1	3	4
Validaciones	3	4	4
Menor Uso de JavaScript	2	3	4
Total	9	14	16
Porcentaje de generación de código	56,25%	87,50%	100,00%

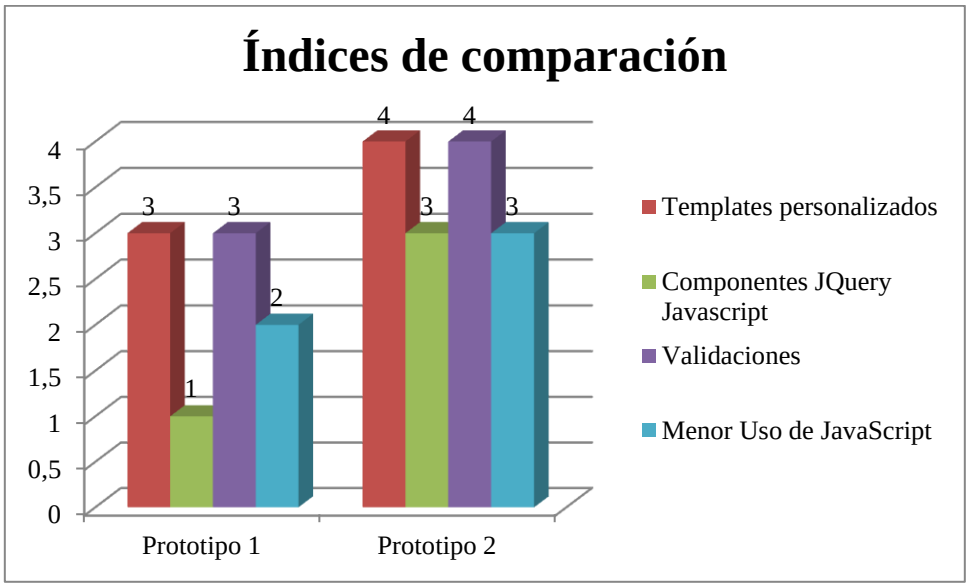


Figura V.56. Resultados del Índice de comparación de Presentación.

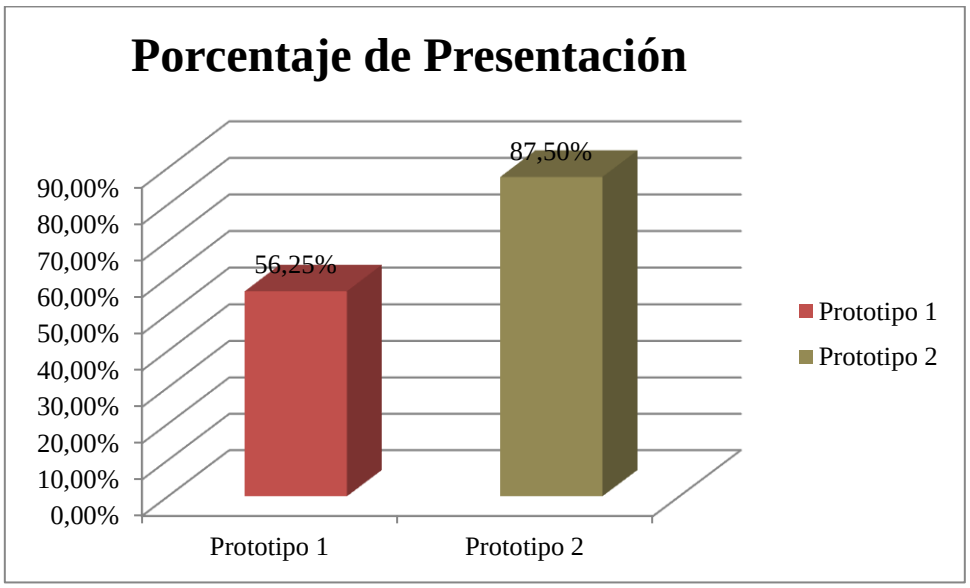


Figura V.57. Porcentajes totales de Presentación.

5.1.1.4.5.5 Interpretación

En la Figura V.57, consecuencia de la valoración de los diferentes índices para este parámetro se tiene como resultado que el Prototipo que utiliza la tecnología JSP / JDBC ha alcanzado un porcentaje de 56,25% y que el Prototipo que utiliza la tecnología SPRING / HIBERNATE ha alcanzado un porcentaje del 87,50% esto quiere decir que utilizando la tecnología SPRING / HIBERNATE estamos optimizando tiempo en cuanto tiene que ver con el parámetro de Presentación para el desarrollo y tiempo de manteniblidad de páginas web.

5.1.2 COMPROBACIÓN HIPÓTESIS

5.1.2.1 HIPÓTESIS

La hipótesis del presente estudio planteó que:

"La utilización del Framework Spring MVC que permitirá obtener mayor productividad en el desarrollo de aplicaciones web"

A partir de esta información se puede identificar las variables dependientes e independientes que intervienen.

La implementación es en sí el núcleo del desarrollo de un sistema, así como, la ingeniería de software realiza investigaciones para poder llegar a presentar la solución deseada, este trabajando es realizado con la finalidad de disminuir el trabajo que realiza el programador, disminuyendo de esta manera, el tiempo empleado en realizar una tarea, sin descuidar características como la calidad, la seguridad, la eficiencia, etc.

Si se puede contar con herramientas de entorno visual que permiten agilitar nuestro trabajo, estaremos en la capacidad de presentar una solución más productiva, de esta forma se demuestra que la utilización del Framework Spring MVC para el desarrollo Web permitirá disminuir el tiempo en la creación de aplicaciones web. Principalmente por permitir una mayor productividad de la gestión de registros en menor tiempo.

5.1.2.1.1 DETERMINACIÓN DE VARIABLES.

Variable Independiente.- Utilización del framework Spring Web MVC 3.0 e Hibernate.

Variable Dependiente.- Productividad en el desarrollo de aplicaciones web.

5.1.2.1.2 OPERACIONALIZACIÓN DE LAS VARIABLES.

5.1.2.1.2.1 Operacionalización Conceptual.

Tabla V.XXXVI. Operacionalización Conceptual.

VARIABLE	TIPO	CONCEPTO
Utilización del framework Spring Web MVC 3.0 e Hibernate.	Independiente	Conjunto de etapas formalmente estructuradas, de manera que brinden a los interesados parámetros de acción en el desarrollo de sus proyectos: plan general y detallado, tareas y acciones, tiempos, aseguramiento de la calidad, involucrados, etapas, revisiones de avance, responsables, recursos requeridos, entre otros.
Productividad en el desarrollo de aplicaciones web.	Dependiente	Es el periodo o lapso de tiempo utilizado desde que se inicia el desarrollo de la aplicación hasta su fin.

5.1.2.1.2.2 Operacionalización.

Tabla V.XXXVII. Operacionalización de Variables

VARIABLE	INDICADORES	TÉCNICAS	FUENTES DE VERIFICACIÓN
Utilización del framework Spring Web MVC 3.0 e Hibernate.	Tecnología para el desarrollo de software	Revisión de Documentos	•Internet. •Libros. •Manuales. •Tutoriales.
Productividad en el desarrollo de aplicaciones web.	Reusabilidad.	Observación	•Prototipo1 •Prototipo2
	Generación de Código.	Observación	•Prototipo1 •Prototipo2
	Modularidad.	Observación	•Prototipo1 •Prototipo2
	Mantenibilidad.	Observación	•Prototipo1 •Prototipo2
	Presentación.	Observación	•Prototipo1 •Prototipo2

5.1.3 COMPARACIÓN DE LOS RESULTADOS OBTENIDOS

Para la comprobación de hipótesis se utilizara el Método Estadístico Chi cuadrado, trabajaremos con los porcentajes, valores observados:

De acuerdo a la evaluación realizada por cada parámetro se establecieron valores que permiten determinar la minimización del tiempo en la construcción y mantenimiento de aplicaciones web. En la tabla siguiente se muestra los valores obtenidos.

5.1.3.1 Criterios de evaluación mejora de las tecnologías

A continuación se muestran los valores cualitativos y cuantitativos que se darán a los parámetros a ser analizados en la comparación de las tecnologías, para lo cual se utilizara los siguientes valores.

Tabla V.XXXVIII. Criterios de Evaluación General.

CRITERIOS DE EVALUACIÓN GENERAL				
Cuantitativa	0-35	36-70	71-85	86-100
Cualitativa	No Mejora		Mejora	
Porcentajes	$\leq 35\%$	36%-70%	71%-84%	$\geq 86\%$

Aquí se presenta la tabla general de valores encontrada para la clasificación de valores para aplicar el proceso de CHI-Cuadrado.

Tabla V.XXXIX. Resultados obtenidos globales de la comparación de las tecnologías

PARÁMETROS	**TECNOLOGÍAS**		**PESO MÁXIMO**
OPTIMIZACIÓN DE TIEMPO	**JSP / JDBC**	**Spring MVC / Hibernate**	
REUSABILIDAD	58,33	91,76	100
GENERACIÓN DE CÓDIGO	55	55	100
MODULARIDAD	56,25	100	100
MANTENIBILIDAD	75	75	100
PRESENTACIÓN	56,25	87,5	100
TOTAL	300,83	409,26	500
PORCENTAJES	60,17%	81,85%	100,00%

Como se puede observar en la Tabla V.XXXIX son los valores obtenidos, a los se procede a clasificar según Tabla V.XXXVIII de criterios de evaluación general. Obteniendo así una nueva tabla de valores como se indica.

Tabla V.XL. Resultados obtenidos clasificados de la comparación de las tecnologías

Productividad	**Parámetros**	**Tecnologías**	
	Productividad de tiempo	**JSP / JDBC**	**Spring MVC / Hibernate**
No mejora	Reusabilidad	58,33	0
	Generación de código	55	55
	Modularidad	56,25	0
	Mantenibilidad	0	0
	Presentación	56,25	0
Mejora	Reusabilidad	0	91,76
	Generación de código	0	0
	Modularidad	0	100
	Mantenibilidad	75	75
	Presentación	0	87,5

Como se aprecia, una vez obtenida esta clasificación se procede al análisis de los parámetros "mejor" y "no mejora" de manera individual.

Tabla V.XLI. Resumen de clasificación de No Mejora.

Productividad	**Parámetros**	**Tecnologías**		
	Productividad de tiempo	**JSP / JDBC**	**Spring MVC / Hibernate**	**Total**
No mejora	Reusabilidad	58,33	0	58,33
	Generación de código	55	55	110
	Modularidad	56,25	0	56,25
	Mantenibilidad	0	0	0
	Presentación	56,25	0	56,25
Total		225,83	55	280,83
Porcentajes		80,42%	19,58%	100,00%

Tabla V.XLII. Resumen de clasificación de Mejora.

Productividad	Parámetros	Tecnologías		
	Productividad de tiempo	**JSP / JDBC**	**Spring MVC / Hibernate**	**Total**
Mejora	Reusabilidad	0	91,76	91,76
	Generación de código	0	0	0
	Modularidad	0	100	100
	Mantenibilidad	75	75	150
	Presentación	0	87,5	87,5
Total		75	354,26	429,26
Porcentajes		17,47%	82,53%	100,00%

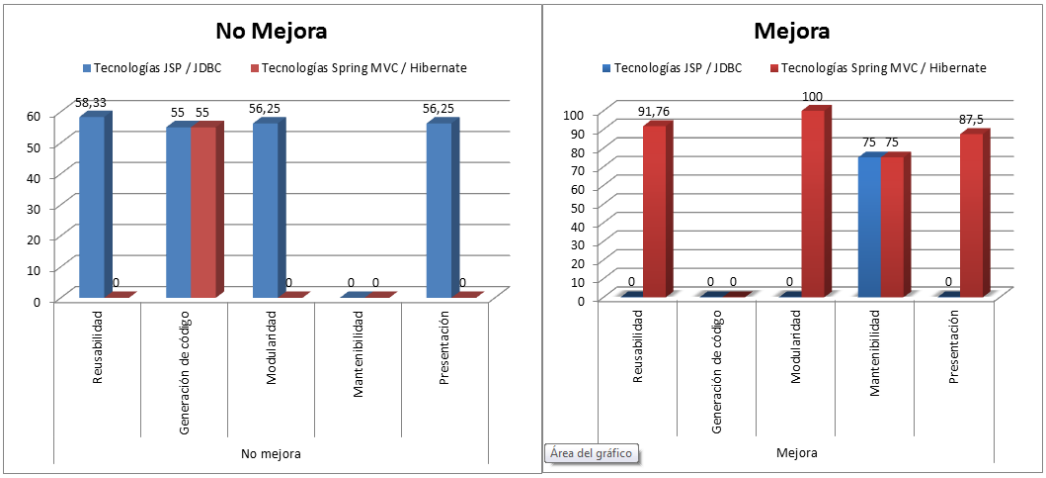


Figura V.58. Porcentajes de la clasificación de mejora y no mejora.

En la Figura V.58 se visualiza los porcentajes individuales de cada uno de los parámetros de no mejora y mejora para poder obtener una decisión breve sobre la eficiencia de cada una de las tecnologías, como se capta en los para metros con mayor porcentaje se presenta en la parte de mejora favoreciendo de esta manera a la tecnologíaSPRING / HIBERNATE.

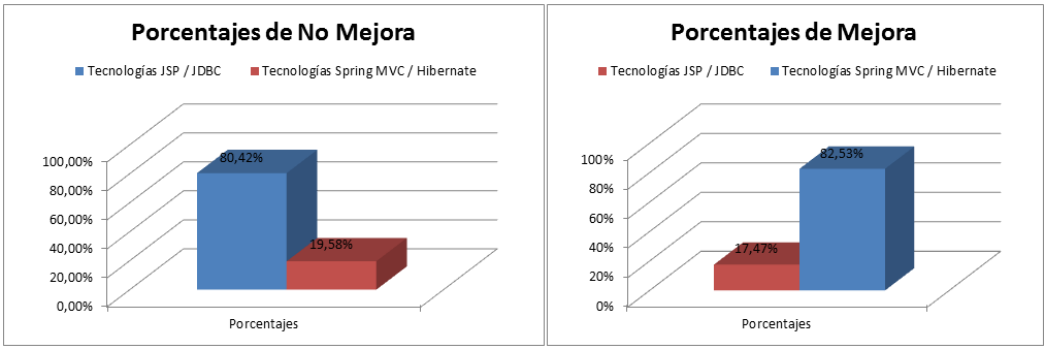


Figura V.59. Porcentaje global de no mejora.

En la Figura V.59 se aprecia los porcentajes globales de (mejora) y (no mejora) de las tecnologías. Usando la tecnología SPRING / HIBERNATE se obtiene mayor Reusabilidad, Modularidad y Presentación de interfaces con los valores obtenidos en la contraposición de las variables de hipótesis tenemos que no mejora en 19,58 % y mejora en 82,53%, permitiendo de esta manera Maximizar la productividad de construcción y mantenimiento de aplicaciones web. Entonces, con respecto a la tecnología JSP / JDBC, la tecnología SPRING / HIBERNATE permite minimizar el tiempo en la construcción y mantenimiento de aplicaciones web en un20% aproximadamente más, así como se observa en la figura, concepto al parámetro de no mejora se tiene que la tecnología SPRING / HIBERNATE contiene al valor más bajo lo que su ves beneficia a las misma.

5.1.4 APLICACIÓN DE CHI CUADRADO.

Planteamiento de la hipótesis:

Hi: "La utilización del Framework Spring MVC que permitirá obtener mayor productividad en el desarrollo de aplicaciones web"

Ho: "La utilización del Framework Spring MVC no permitirá obtener mayor productividad en el desarrollo de aplicaciones web"

Encontramos los totales por cada fila y por cada columna:

Tabla V.XLIII. Parámetros de comparación - Totales de filas y columnas.

Productividad	Parámetros	Tecnologías		
	Productividad de tiempo	**JSP / JDBC**	**Spring MVC / Hibernate**	**Total**
No mejora	**Reusabilidad**	58,33	0	58,33
	Generación de código	55	55	110
	Modularidad	56,25	0	56,25
	Mantenibilidad	0	0	0
	Presentación	56,25	0	56,25
Mejora	**Reusabilidad**	0	91,76	91,76
	Generación de código	0	0	0
	Modularidad	0	100	100
	Mantenibilidad	75	75	150
	Presentación	0	87,5	87,5
Total		300,83	409,26	710,09

5.1.4.1 Valores esperados

De la tabla que son los valores observados, se encuentra los valores esperados, con la siguiente formula.

$$\text{Valor Esperado} = \frac{\text{Total Columna} * \text{Total Fila}}{\text{Total Suma}}$$

Ejemplo:

$$\text{Valor Esperado}(1,1) = \frac{300,83 * 58,33}{710,09} = 63.02$$

Obteniendo como resultado los siguientes valores esperados:

Tabla V.XLIV Valores Esperados y Totales.

Productividad	Parámetros	Tecnologías		Total
	Productividad de tiempo	**JSP / JDBC**	**Spring MVC / Hibernate**	
No mejora	**Reusabilidad**	24,71	33,62	58,33
	Generación de código	46,6	63,4	110
	Modularidad	23,83	32,42	56,25
	Mantenibilidad	0	0	0
	Presentación	23,83	32,42	56,25
Mejora	**Reusabilidad**	38,87	52,89	91,76
	Generación de código	0	0	0
	Modularidad	42,37	57,63	100
	Mantenibilidad	63,55	86,45	150
	Presentación	37,07	50,43	87,5
Total		300,83	409,26	710,09

Se obtiene el valor de Chi cuadrado mediante la fórmula:

fo = frecuencia de valor observado.

fe = frecuencia de valor esperado.

$$x^2 calc = \sum \frac{(fo - fe)^2}{fe}$$

Para la clasificación final de valores finales para el cálculo de Chi- Cuadrado se tiene que la siguiente tabla para obtener un resultado óptimo de Chi- Cuadrado:

Tabla V.XLV. Clasificación de resultados obtenidos y esperados

CRITERIOS DE EVALUACIÓN DE RESULTADOS				
Cuantitativa	0-35	36-70	71-85	86-100
Cualitativa	1	2	3	4
Porcentajes	≤ 35%	36%-70%	71%-84%	≥ 86%

Tabla V.XLVI. Resumen Valores Observados y valores Esperados.

Productividad	Parámetros	Tecnologías Valores Observados		Tecnologías Valores Esperados	
	Productividad de tiempo	JSP / JDBC	Spring MVC / Hibernate	JSP / JDBC	Spring MVC / Hibernate
No mejora	Reusabilidad	2	0	1	1
	Generación de código	2	2	2	2
	Modularidad	2	0	1	1
	Mantenibilidad	0	0	0	0
	Presentación	2	0	1	1
Mejora	Reusabilidad	0	4	2	2
	Generación de código	0	0	0	0
	Modularidad	0	4	2	2
	Mantenibilidad	3	3	2	4
	Presentación	0	4	2	2

$$X^2_{calc} = \frac{(2-1)^2}{1} + \frac{(0-1)^2}{1} + \frac{(2-2)^2}{2} + \frac{(2-2)^2}{2} + \frac{(2-1)^2}{1} + \frac{(0-1)^2}{1} + \frac{(2-1)^2}{1} + \frac{(0-2)^2}{2} + \frac{(4-2)^2}{2} + \frac{(0-2)^2}{2} + \frac{(4-2)^2}{2} + \frac{(3-2)^2}{2} + \frac{(3-4)^2}{4} + \frac{(0-2)^2}{2} + \frac{(4-2)^2}{2}$$

$$X^2_{calc} = 1 + 1 + 0 + 0 + 1 + 1 + 0 + 0 + 1 + 1 + 2 + 2 + 0 + 0 + 2 + 2 + 0{,}5 + 0{,}25 + 2 + 2$$

$$\mathbf{X^2_{calc} = 18{,}75}$$

Grados De Libertad

Para calcular los grados de libertad se realiza con la fórmula:

V = (Cantidad de filas - 1) (Cantidad de columnas - 1)

Grados de libertad = (10-1) (2-1) = 9

Nivel de confianza.

Por lo general se trabaja con un porcentaje de 95% (0,95), es decir el nivel de error es del 5%(0,05).

ν	0,005	0,01	0,025	0,05	0,95	0,975	0,99	0,995
1	0,00003935	0,000157	0,000982	0,00393	3,841	5,024	6,635	7,879
2	0,010	0,020	0,051	0,103	5,991	7,378	9,210	10,597
3	0,072	0,115	0,216	0,352	7,815	9,348	11,345	12,838
4	0,207	0,297	0,484	0,711	9,488	11,143	13,277	14,860
5	0,412	0,554	0,831	1,145	11,070	12,832	15,086	16,750
6	0,676	0,872	1,237	1,635	12,592	14,449	16,812	18,548
7	0,989	1,239	1,690	2,167	14,067	16,013	18,475	20,278
8	1,344	1,647	2,180	2,733	15,507	17,535	20,090	21,955
9	1,735	2,088	2,700	3,325	16,919	19,023	21,666	23,589
10	2,156	2,558	3,247	3,940	**18,307**	20,483	23,209	25,188
11	2,603	3,053	3,816	4,575	19,675	21,920	24,725	26,757
12	3,074	3,571	4,404	5,226	21,026	23,337	26,217	28,300
13	3,565	4,107	5,009	5,892	22,362	24,736	27,688	29,819
14	4,075	4,660	5,629	6,571	23,685	26,119	29,141	31,319
15	4,601	5,229	6,262	7,261	24,996	27,488	30,578	32,801
16	5,142	5,812	6,908	7,962	26,296	28,845	32,000	34,267
17	5,697	6,408	7,564	8,672	27,587	30,191	33,409	35,718
18	6,265	7,015	8,231	9,390	28,869	31,526	34,805	37,156
19	6,844	7,633	8,907	10,117	30,144	32,852	36,191	38,582
20	7,434	8,260	9,591	10,851	31,410	34,170	37,566	39,997

Figura V.60. Tabla de CHI CUADRADO.

Obteniendo el valor crítico con 9 grados de libertad y valor de error de 0,05 = 16,92 Conclusión:

18,75 > 16,92

Como el **CHI CUADRADO** experimental es mayor al valor crítico, cae en la zona de rechazo de la Hipótesis Nula (Ho), aceptando la Hipótesis Alternativa.

5.1.4 Evaluación la aplicación para determinar la mejora en la gestión proyectos de la COMPROTEC.

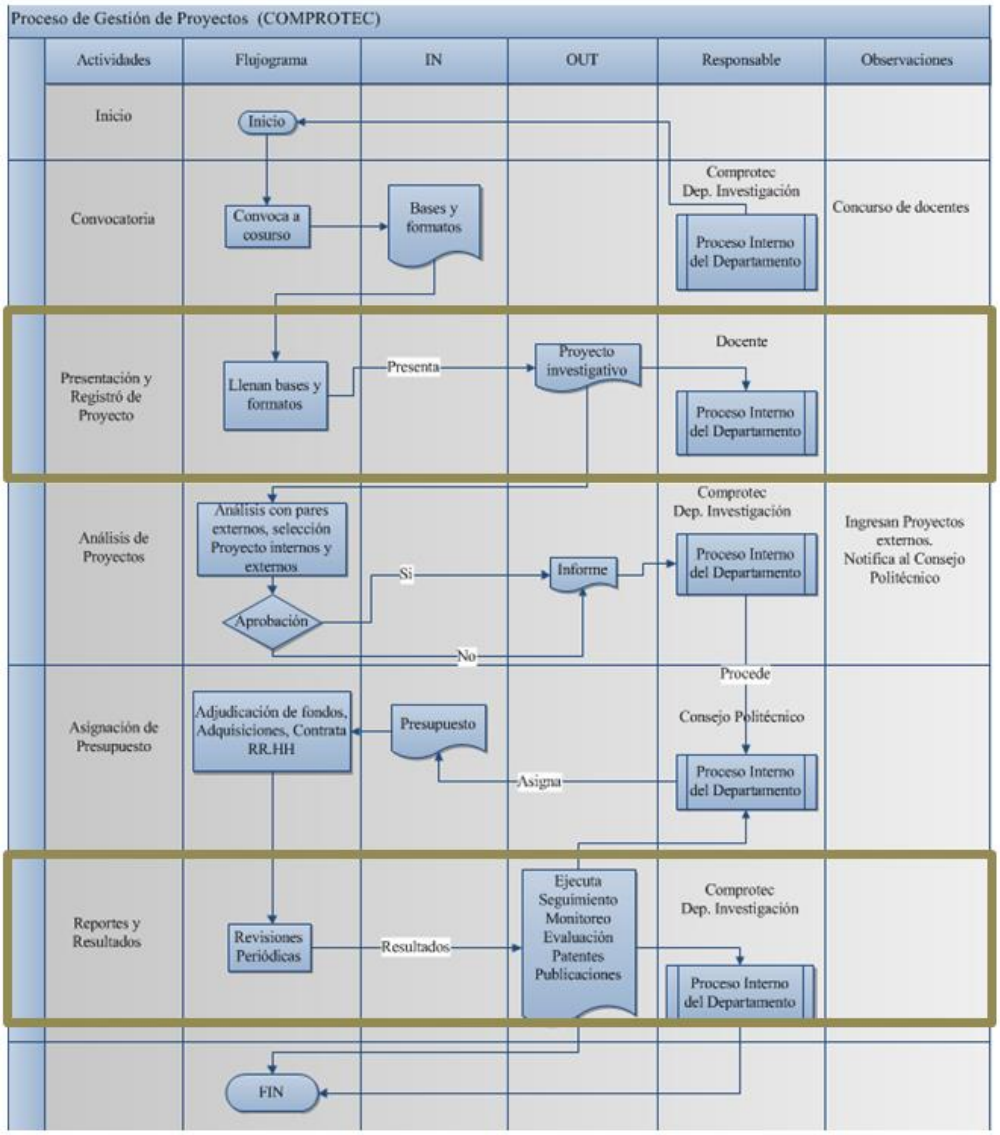


Figura V.61. Proceso de gestión de proyectos.

En la Figura V.61 se observa las actividades que realiza la COMPROTEC para la gestión de proyectos dentro de las cuales, se han automatizados las siguientes actividades para minimizar los tiempos de registros de un proyecto presentado por un docente y la generación de reportes.

Presentación y registro de un proyecto

Esta actividad nos permite deducir el tiempo transcurrido con que el docente presenta un proyecto de forma manual, y mediante la aplicación web.

Tabla V.XLVII. Resumen de datos obtenidos Presentación y Registro de Proyectos.

Actividad	Gestión (min)	Generación (min)	Manual (min)	Gestión (min)	Generación (min)	Aplicación (min)	Diferencia (min)	Mejora (%)
Presentación y registro de un proyecto	1920	120	**2040**	1920	5	**1925**	115	**5,64%**
Porcentajes			**100%**			**94,36%**		

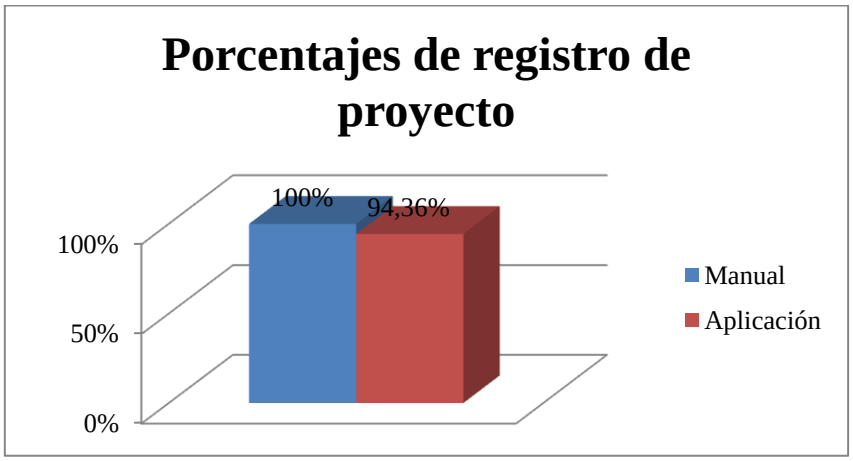


Figura V.62. Porcentaje de mejora registro de proyecto.

En la Tabla V.XLVII se puede deducir que de forma manual en el registro de un proyecto se demora 1920 minutos en gestión de trámites y 120 minutos en generación del ingreso, es así que nos da un total de 2040 minutos lo que equivale a 4 días con 2 hora, el registro de un proyecto mediante la aplicación. Se tiene 1920 minutos en gestión de trámites y 5 en generación del registro, tomando en cuenta que la información ya está disponible para generar el informe, dando así un total de 1925 minutos con una transformación de 4 días con 1 horas, con una diferencia de 115 minutos equivalente de mejora de 5.64%.

Reportes y resultados

Esta actividad permite calcular el tiempo que la institución puede emitir un resultado para la toma decisiones, de los estados de los proyectos, lo que es claro que de forma manual conlleva mucho tiempo para emitir dichos reportes, con la aplicación dichos resultados se minimiza al instante, tomando un tiempo laborable de 8 horas se procede hacer los cálculos trabajando en minutos.

Tabla V.XLVIII. Resumen de datos obtenidos Reportes.

Actividad	Reportes	Gestión (min)	Generación (min)	Manual (min)	Gestión (min)	Generación (min)	Aplicación (min)	Diferencia (min)	Mejora (%)
Reportes y resultados	Difusión científica de Investigación.	1440	1440	**2880**	1440	5	**1445**	1435	49,83%
	Difusión científica de transferencia.	1440	1440	**2880**	1440	5	**1445**	1435	49,83%
	Publicación de artículos	960	1440	**2400**	960	5	**965**	1435	59,79%
	Patentes y Marcas	1920	960	**2880**	1920	5	**1925**	955	33,16%
	Tesis de grado	2400	1440	**3840**	2400	5	**2405**	1435	37,37%
Total		**8160**	**6720**	**14880**	**8160**	**25**	**8185**	**6695**	**46,00%**
Porcentajes				**100%**			**55,00%**		

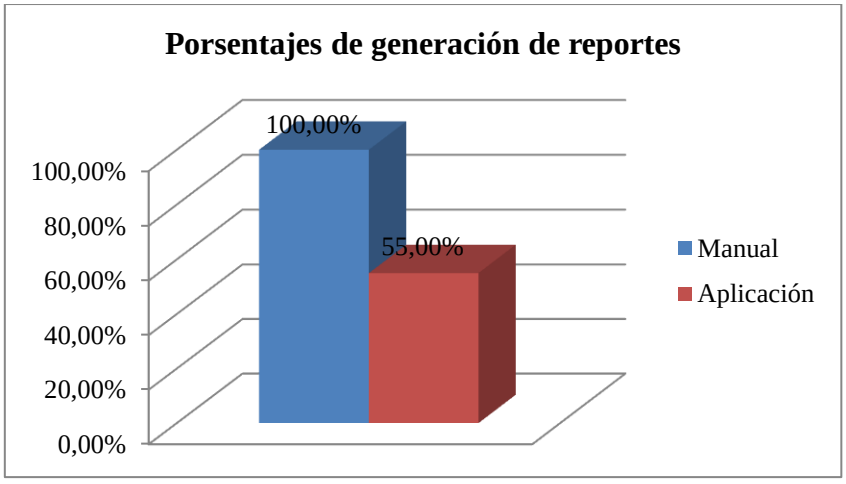


Figura V.63. Porcentaje de generación de reportes.

Observe la Tabla V.XLVIII tal es el caso de los reportes generados que prestan un mejor servicio al disminuir el tiempo de generación de los mismos con la aplicación, lo cual se observa que esta sobre el 30% de mejora en generación de reportes.

Se observa que se tiene un promedio de mejora en la actividad de reportes, que representa el 46.00%, en cuanto a tiempos, se tiene un 45% en disminución de tiempo en el trabajo cotidiano de la Comprotec, lo cual agilita sus actividades al servicio del usuario.

Como se puede observar los reportes con la aplicación son emitidas en pocos minutos aunque el tiempo de gestión de trámites para obtenerlos es el mismo en las dos actividades.

Tabla V.XLIX.Resumen global de mejora.

Actividad	Gestión (min)	Generación (min)	Manual (min)	Gestión (min)	Generación (min)	Aplicación (min)	Diferencia (min)	Mejora (%)
Presentación y registro de un proyecto	1920	120	**2040**	1920	5	**1925**	115	**5,60%**
Reportes	8160	6720	**14880**	**8160**	**25**	**8185**	**6695**	**46,00%**
Totales			**16920**			**10110**		
Porcentaje			**100%**			**59,75%**		

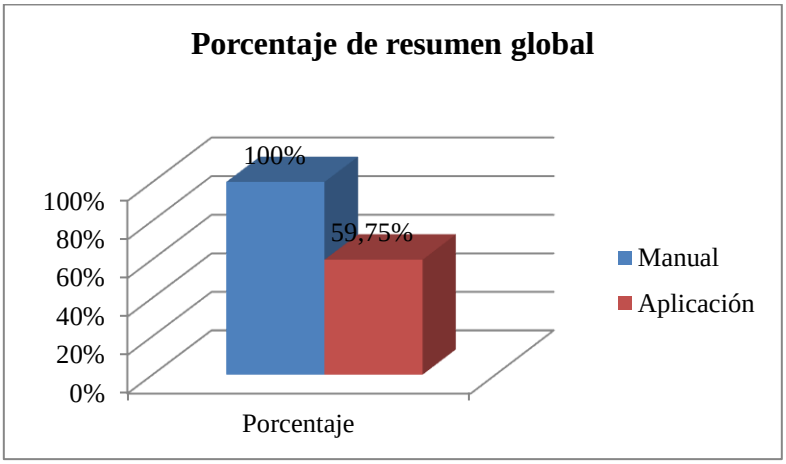


Figura V.64. Porcentaje resumen global.

En la Tabla V.XLIX se puede deducir que el porcentaje total de minimización de tiempo entre las dos actividades es de 40.25%.

CONCLUSIONES

1. La integración de Framework Spring Web MVC 3.0 e Hibernate permite realizar aplicaciones web de tipo empresarial, de forma estructurada, ordenada y escalable; además de ahorrar tiempo en la construcción en base a los resultados obtenidos de los parámetros analizados.
2. La utilización de las herramientas tecnológicas Framework Spring Web MVC 3.0 e Hibernate ha permitido desarrollar el módulo de gestión de proyectos de una forma rápida, que facilitará los procesos manuales que llevan a cabo actualmente y brindara mayores prestaciones a la misma.
3. Hibernate permite realizar un mapeo de la base de datos, y parte de la lógica de negocio sin esfuerzo alguno de una forma rápida minimizando el tiempo de desarrollo para el programador en 80%.
4. El nuevo sistema SGCOM ha permitido mejorar la gestión de proyecto en un porcentaje de 50% en tiempo y minimizando el gasto de útiles de oficina en 90%.
5. El desarrollo de módulos de prototipos de prueba permitió verificar en tiempo real las deficiencias y beneficios que brindan cada una de las herramientas comparadas, lo cual ayudó a determinar la mejor para construir la aplicación web SGCOM.
6. Se desarrolló una Metodología para la construcción de aplicaciones web mediante la integración de Framework Spring Web MVC 3.0 e Hibernate la cual fue aplicada para la construcción del software de gestión SGCOM, que ha permitió minimizar el tiempo de desarrollo de la aplicaciones Web en un 21.68% .
7. El rendimiento de la aplicación está dentro de las expectativas o niveles de aceptación que han sido aprobadas por el departamento DESITEL.

8. La mejora en cuanto a tiempo del registro de un proyecto y generación de reportes representa el 40,25%.

RECOMENDACIONES

1. Investigar más a fondo sobre las tecnologías utilizadas en la metodología DAWE Web, de modo que en un futuro se pueda mejorar esta metodología.
2. Utilizar librerías compatibles con JSF para integrar con Framework Spring Web MVC 3.0 para crear interfaces de usuarios de una forma más rápida y sencilla.
3. Investiga mucha información sobre el Framework Spring Web MVC 3.0, ya que en su mayor parte la información está en Inglés y no existe mucho soporte sobre los inconvenientes que se pueda tener durante el desarrollo.
4. Dedicar un tiempo considerable en el diseño y construcción de la base de datos de la aplicación, de modo que no haya inconvenientes al momento de que se genere la capa de acceso a datos y de lógica de negocios por parte del framework Hibernate.
5. Estudiar y analizar profundamente las tecnologías Framework Spring Web MVC 3.0 y los beneficios de su integración, para hacer uso de toda la potencialidad que estas tecnologías ofrecen para el desarrollo de Aplicaciones Web Empresariales.
6. Se recomienda investigar más a fondo de las tecnologías usadas en la metodología DAWE, de modo que se pueda mejorar esta metodología en un futuro.
7. La aplicación servirá de soporte para los nuevos cambios realizados en la Comprotec conocido ahora como Instituto de Investigación.

RESUMEN

Propuesta de una guía metodológica utilizando tecnología Spring, como framework de desarrollo para aplicaciones informáticas en la ESPOCH. Caso práctico COMPROTEC ESPOCH, implantada en la Facultad de Informática y Electrónica.

Se realizó dos prototipos de software; el primer prototipo de desarrolló utilizando la metodología tradicional, con herramientas como JSP y JDBC, el segundo prototipo con la metodología propuesta (DAWE), que usa tecnologías como Framework Spring Web MVC 3.0 e Hibernate.

Para la implementación del Portal SGCOM Web se utilizó, Framework Spring Web MVC 3.0 e Hibernate, PostgreSQL, Joomla 2.5 para la publicación de la información pública, bajo Sistema Operativo Centos 6.3 con Servidor Glassfish.

De acuerdo al análisis comparativo de las tecnologías jsp / jdbc y tecnologías spring / hibernate de cuerdo a los siguientes parámetros de comparación: Optimización de tiempo, Reusabilidad, Generación de código, Modularidad, Mantenibilidad, Presentación, se obtuvo como resultado un 81.85% de productividad para la tecnología SPRING / HIBERNATE y 60.17% para tecnología JSP / JDBC por lo que se ha concluido que las herramientas Spring MVC / Hibernate son la mejor opción según CHI- CUADRADO, es la que más prestaciones de desarrollo ofrece.

Concluyo de esta manera que utilizando la Metodología Propuesta DAWE minimiza el tiempo en el desarrollo y de aplicaciones web empresariales. El sistema mejora la gestión de proyectos en 40,25%.

Se recomienda el uso del Sistema de Gestión de la Comprotec (SGCOM), servirá de ayuda en la administración y control de todos los procesos que actualmente se los realiza manualmente en la Comisión de Proyectos y Transferencia Tecnológica (COMPROTEC).

SUMMARY

Proposal of a methodological guide, using the Spring methodology, as development framework for informatics applications at the ESPOCH. Practical case COMPROTEC ESPOCH, was established in the Faculty of Informatics and Electronics.

Two software prototypes were made. The first prototype was developed using traditional methodology, with tools such as JSP and JDBC. The second prototype was with the proposed methodology (DAWE), using technologies such as Spring Web MVC 3.0 Framework and Hibernate.

To implement the SGCOM Web Portal was used Spring Web MVC Framework 3.0 and Hibernate, PostgreSQL, Joomla 2.5 for publishing public information under Operative System Centos 6.3 with Glassfish Server.

According to the technologies comparative analysis JSP / JDBC and technologies SPRING / HIBERNATE according to the following parameters: Optimization of time, reusability, code generation, modularity, maintainability, and presentation. The obtained result was a productivity of 81.85% for SPRING / HIBERNATE technology and 60.17% for JSP / JDBC technology so it was found that Spring MVC / Hibernate tools are the best choice as CHI-SQUARE. It is the development that offers more features.

Thus, I conclude that using the DAWE proposed mythodology minimizes the time in the development and enterprise web applications. The system improves the project management in a 40.25%.

We recommend using System Management Comprotec (SGCOM), which will be useful in the management and control of all processes that are currently performed manually at the Commission of Projects and Technology Transfer (COMPROTEC).

GLOSARIO

API.-interfaz de programación de aplicaciones o API (del inglés Application Programming Interface) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usados generalmente en las bibliotecas.

Administrador Persona responsable del manejo del sistema.

DAWE. - Desarrollo de Aplicaciones Web Empresariales.

Aplicación Tipo de programa informático diseñado como herramienta para permitir a un usuario realizar uno o diversos tipos de trabajo.

Arquitectura Nivel de diseño que hace foco en aspectos "más allá de los algoritmos y estructuras de datos de la computación; el diseño y especificación de la estructura global del sistema es un nuevo tipo de problema".[10]

Atributo Indica una o más características de un objeto.

Bean Componente software que tiene la particularidad de ser reutilizable y así evitar la tediosa tarea de programar los distintos componentes uno a uno.

Aplicaciones web.-aplicación web es aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador.

Applets.- es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web. El applet debe ejecutarse en un contenedor, que lo proporciona un programa anfitrión, mediante un plugin, o en aplicaciones como teléfonos móviles que soportan el modelo de programación por 'applets'.

Base de datos: Es una colección estructurada de datos y forma parte de un sistema de información.

Cliente web.- Un cliente web, es cualquier aplicación que sirve para utilizar la web. Por ejemplo, los navegadores de internet.

Cliente-servidor.- consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

CSS.- es un lenguaje de hojas de estilo usado para describir la semántica de la presentación (la apariencia y formato) de un documento escrito en un lenguaje de marcas .

DOM.- Document Object Model o DOM ('Modelo de Objetos del Documento' o 'Modelo en Objetos para la representación de Documentos') es esencialmente una interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML.

Framework Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado.

GDI.-Graphics Device Interface (cuyo acrónimo es GDI), es uno de los tres componentes o subsistemas de la interfaz de usuario de Microsoft Windows. Trabaja junto con el núcleo y la API de Windows.

Hardware Corresponde a todas las partes físicas y tangibles de una computadora: sus componentes eléctricos, electrónicos, electromecánicos y mecánicos.

HTML.- siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

HTML5.- es un lenguaje para la estructuración y presentación de contenidos para la World Wide Web , una tecnología de base de la Internet . Es la última revisión del HTML estándar (creado originalmente en 1990 y más recientemente como estándar HTML 4 en 1997.

IDE.- (Entorno de desarrollo integrado) es un programa informático compuesto por un conjunto de herramientas de programación.

Interfaz de usuario.- interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo, normalmente suelen ser fáciles de entender y fáciles de accionar.

Internet.- es un conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos TCP/IP, garantizando que las redes físicas heterogéneas que la componen funcionen como una red lógica única, de alcance mundial.

Java Es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90.

Javascript.- es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

JDK.- (Java Development Kit), es un software que provee herramientas de desarrollo para la creación de programas en java. Puede instalarse en una computadora local o en una unidad de red.

En la unidad de red se pueden tener las herramientas distribuidas en varias computadoras y trabajar como una sola aplicación.

JRE.- (Java Runtime Environment), es un conjunto de utilidades que permite la ejecución de programas Java.

Kernel.- es un software que constituye el núcleo del sistema operativo. Responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora

MainFrame.- es una computadora grande, potente y costosa usada principalmente por una gran compañía para el procesamiento de una gran cantidad de datos.

Middleware.- es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos

Navegador web.- es un programa que permite ver la información que contiene una página web, (ya se encuentre ésta alojada en un servidor dentro de la web o en un servidor local).

Open source: Código abierto. Permite que el software sea estudiado y distribuido para que la comunidad se beneficie.

Página Web.- es un documento o información electrónica adaptada para la World Wide Web que generalmente forma parte de un sitio web. Su principal característica son los hipervínculos de una página, siendo esto el fundamento de la WWW.

Plataformas.- es un sistema que sirve como base para hacer funcionar determinados módulos de hardware o de software con los que es compatible.

Profiling.- es la investigación del comportamiento de un programa de computadora usando información reunida del análisis dinámico del programa en oposición al análisis estático.

Red.- es un conjunto de equipos informáticos conectados entre sí por medio de dispositivos físicos que envían y reciben impulsos eléctricos, ondas electromagnéticas o cualquier otro medio para el transporte de datos para compartir información y recursos

RIA (Aplicaciones de Internet Enriquecidas).- nuevo tipo de aplicaciones que surge como una combinación de las ventajas que ofrecen las aplicaciones Web y las aplicaciones Ágil es.

Rich Clients.- es el programa "cliente" de una arquitectura cliente-servidor cuando la mayor carga de cómputo está desplazada hacia la computadora que ejecuta dicho programa

SDK (Kit de Desarrollo de Software).- es generalmente un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema concreto

Servidor.- un servidor es un tipo de software que realiza ciertas tareas en nombre de los usuarios. Se refiere también al dispositivo físico que posee recursos muy altos para poder ofrecer sus servicios a los clientes

SOA (Arquitectura Orientada a Servicios).- es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos del negocio.

Software.- equipamiento lógico o soporte lógico de una computadora digital; comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas

Threads Hilo de ejecución o subproceso es una característica que permite a una aplicación realizar varias tareas a la vez.

Tablets.- es una computadora portátil con la que se puede interactuar a través de una pantalla táctil o multitáctil.

UML (Lenguaje Unificado de Modelado).- Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema

XML(Lenguaje de Marcas Extensible).- es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específico.

Web Red informática, especialmente para referirse a internet.

ANEXOS

ANEXO 1

MANUAL DEL PORTAL JOOMLA

ANEXO 2

MANUAL DEL SGCOM

BIBLIOGRAFÍA GENERAL

[1]. **PETTERSSON, S.,** ENTERPRISE APPLICATION DEVELOPMENT USING DEPENDENCY INJECTION AND ASPECT-ORIENTED PROGRAMMING., Facultad de Ciencias de la Computación y Comunicación., Computación e Ingeniería., Instituto Real de Tecnología., Tesis., Suecia., 2008., Pp., 5-15., 29-34.

BIBLIOGRAFÍA DE INTERNET

ANÁLISIS HERRAMIENTAS DE ESTUDIO

[2]. http://materias.fi.uba.ar/7500/blanco-tesisingenieriainformatica.pdf

2012/11/04

FRAMEWORK SPRING MVC

[3]. http://viralpatel.net/blogs/spring-3-mvc-handling-forms/

2012/05/25

[4]. http://static.springsource.org/spring/docs/3.0.0.M3/reference/html/

2012/06/05

[5]. http://www.javamexico.org/foros/java_enterprise/%C2%BFpara_que_me_sirve_y_que_resuelvo_con_spring

2012/07/15

[6]. http://www.compujuy.com.ar/postx.php?id=93

2012/07/18

HIBERNATE

[7]. http://wad1.wikispaces.com/file/view/practicaHIBERNATE.Dulce.Carolina.Rueda.Torres.pdf

2012/05/22

JOOMLA

[8]. http://www.joomlaspanish.org/

2012/12/21

PERFORMANCE TESTING

[9]. http://msdn.microsoft.com/en-us/library/bb924375.aspx

2013/03/15

TEORÍA Y ARQUITECTURA DE DISEÑO

[10]. http://www.nada.kth.se/utbildning/grukth/exjobb/rapportlistor/2009/rapporter09/pettersson_stefan_09077.pdf

2012/09/17

TUTORIAL FRAMEWORK SPRING MVC

[11]. http://blog.theinit.com/2011/04/28/nociones-tutorial-spring-mvc-hibernate/

2012/06/15

[12]. http://www.jpalace.org/docs/tutorials/spring/mvc_21.html

2012/07/07

[13]. http://www.tutorialspoint.com/spring/spring_bean_definition.htm

2012/08/20

TUTORIAL DE PROGRAMACIÓN JAVA

[14]. http://www.javatutoriales.com/2010/12/contenedores-de-ioc-e-inyeccion-de.html

2012/07/27

TUTORIAL VALIDACIÓN EN SPRING

[15]. http://blog.teamextension.com/quick-spring-mvc-3-validation-tutorial-767

2012/10/11