
**ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO**

**FACULTAD DE INFORMÁTICA Y ELECTRONICA**

**ESCUELA DE INGENIERÍA EN SISTEMAS**

# ANÁLISIS DE FRAMEWORKS MVC DE JAVA PARA EL DESARROLLO DE APLICACIONES WEB EMPRESARIALES. CASO PRÁCTICO: SISTEMA DE BIENESTAR POLITÉCNICO

**TESIS DE GRADO**

Previa a la obtención del título de

**INGENIERO EN SISTEMAS INFORMÁTICOS**

**Presentado por:**

TANIA PAOLA AGUIRRE BUENAÑO

ANDREA ISABEL MONCAYO ÁLVAREZ

**RIOBAMBA-ECUADOR**

**2013**

**AGRADECIMIENTO**

En primer lugar a Dios, por cuidar todos mis pasos y guiar mi camino, a mis padres por apoyarme en toda etapa de mi vida, por ser un ejemplo de trabajo, superación y haberme brindado amor y cariño.

Agradezco a la ESPOCH en donde compartí momentos inolvidables.

A mis profesores quienes me brindaron todos sus conocimientos, en especial a la Ing. Natalia Layedra quien ha estado siempre pendiente y guiarme con sus conocimientos y experiencias.

Tania Aguirre

**AGRADECIMIENTO**

*Agradezco a Dios,* por protegerme, darme fuerzas

para superar obstáculos y dificultades a lo largo de toda mi vida.

*A mis padres,* por su esfuerzo, sacrificio

y el amor que me han brindado en todo el trayecto de mi vida

*A mi hermano,* con el cual hemos compartido momentos inolvidables.

*A Tania,* por haber logrado nuestro objetivo profesional.

*A Ing. Natalia Layedra,* directora de tesis,

por su valiosa guía y asesoramiento con sus valiosas aportaciones

hicieron posible la culminación de la tesis y por su gran calidad humana.

*A mis amigas y amigos*, gracias por su tiempo,

su apoyo y conocimientos que me supieron transmitir

Gracias a todas las personas que ayudaron directa e indirectamente.

Andrea Moncayo

## DEDICATORIA

La presente tesis va dedicada a mis queridos padres Julio Aguirre y Norma Buenaño, a mi amada hija Emily, a mi Esposo Oljer, a mis hermanos Junior, Alexander, Maritza, Evelyn, por ser protagonistas principales de todos los logros conseguidos en mi vida, brindándome su apoyo incondicional, paciencia estando conmigo en los malos y buenos momentos.

Con cariño para ustedes

Tania Aguirre

## DEDICATORIA

*A Dios*, por permitirme llegar a este momento tan especial en mi vida,
por los triunfos y los momentos difíciles que
me han enseñado a valorarlo cada día más.

*A mis padres, Raúl Moncayo y Gloria Álvarez*
por ser las personas que me ha acompañado durante toda mi vida,
con sus consejos ha sabido guiarme.

*A mi hermano Marcelo y sobrino Nicolay*,
que siempre ha estado junto a mí y brindándome su apoyo.

*A mis amigas y amigos,* por su apoyo incondicional y su amistad brindada.
"La amistad duplica las alegrías y divide las angustias por la mitad"

Les quiero mucho Andru

**FIRMAS DE RESPONSABILIDAD**

| **NOMBRES** | **FIRMAS** | **FECHA** |
|---|---|---|
| Ing. Iván Menes<br>**DECANO DE LA FACULTAD**<br>**DE INFORMÁTICA Y ELECTRÓNICA** | ---------- | ---------- |
| Ing. Raúl Rosero<br>**DIRECTOR DE LA ESCUELA**<br>**DE INGENIERÍA EN SISTEMAS** | ---------- | ---------- |
| Ing. Natalia Layedra<br>**DIRECTOR DE TESIS** | ---------- | ---------- |
| Dr. Julio Santillán<br>**MIEMBRO DE TESIS** | ---------- | ---------- |
| Tlgo. Carlos Rodríguez<br>**DIR. DPTO. CENTRO DOCUMENTACIÓN** | ---------- | ---------- |
| **NOTA DE LA TESIS** | ---------- | |

"Nosotras, Andrea Isabel Moncayo Álvarez y Tania Paola Aguirre Buenaño, somos responsables de las ideas, doctrinas y resultados expuestos en esta tesis; y, el patrimonio intelectual de la Tesis de Grado pertenece a la Escuela Superior Politécnica de Chimborazo"

| | |
|---|---|
| _____ | _____ |
| Andrea Isabel Moncayo Álvarez | Tania Paola Aguirre Buenaño |

## ÍNDICE DE ABREVATURAS

**ABREVIATURAS**

**AOP:** Aspect-Oriented Programming (Programación Orientada al Aspecto)

**ACID:** Atomicity (atomicidad), Consistency (coherencia), Isolation (aislamiento), Durability (permanencia).

**API:** Application Programming Interface (Interfaz de Programación de Aplicaciones).

**BD:** Base de Datos.

**BMP:** BeanManaged Persistence (Persistencia Gestionada por el Bean)

**CGI:**Common Gateway Interface (Interfaz de Pasarela Común)

**CMP:** Container ManagedPersistence (Persistencia Gestionada por el Contenedor)

**CORBA:** Common Object Request Broker Architecture.

**DAO:** Data Access Object (Objeto de Acceso a Datos)

**DDL:** Data Definition Language (Lenguaje de definición de Datos).

**DML:** Data Manipulation Language (Lenguaje de Manipulación de Datos).

**EJB:** Enterprise Java Bean

**GNU:** GNU no es Unix.

**HQL:** Hibernate Query Languaje (Lenguaje de Consultas de Hibernate).

**HTML:** HyperText Markup Language (Lenguaje de Marcado Hipertextual)

**HTTP:** HyperText Transfer Protocol (Protocolo de Transferencia De Hipertexto)

**IDE:** Integrated Development Environment (Entorno Integrado de Desarrollo)

**IoC:** Inversion of Control (Inversión de Control)

**J2EE:** Java Enterprise Edition

**JCA:** Arquitectura de Conexion Java

**JDBC:** Java Database Connectivity (Conector de Base de Datos Java).

**JMX:** Java Management Extensions (Administración de Extensiones Java)

**JNDI:** Java Naming and Directory Interface (Interfaz de Nombres y Directorios Java)

**JSF:** JavaServer Faces

**JSTL:** JavaServer Pages Standard Tag Library

**JTA:** Api para Transacciones Java.

**MSF:** Microsoft Solution Framework.

**MVC:** Modelo Vista Controlador

**OOP:** Object Oriented Programming (Programación Orientada a Objetos)

**ORM:** Object Relational Mapping.

**PAO:** Programación Orientada a Aspectos

**POJO:** Plain Old Java Objtect.

**POO:** Programación Orientada a Objetos

**RDBMS:** Relational Database Management System (Sistema de Administración de Base de Datos Relacional).

**RUP:** Rational Unified Process (Proceso Unificado de Racional)

**SQL:** Structured Query Language (Lenguaje de Consulta Estructurados)

**TCP/IP:** Transmission Control Protocol/Internet Protocol *(*Protocolo de Control de Transmisión/Protocolo de Internet*)*

**URL:** Uniform Resource Locator (Localizador de Recursos Uniforme)

**WWW:**World Wide Web, Sistema de Documentos de Hipertexto

**XML:** Extensible Markup Language (Lenguaje de Marcas Extensibles)

# ÍNDICE GENERAL

**ÍNDICE DE TABLAS**

## ÍNDICE DE FIGURAS

# CAPÍTULO I

# 1. MARCO REFERENCIAL

## 1.1 Antecedentes

### Planteamiento del Problema

En la actualidad existen aplicaciones desarrolladas con framework que utilizan JEE, pero esto provoca algunas complicaciones como las siguientes:

- Desarrollo JEE es duro: Integración de sistemas dispares, Problemas de rendimiento, Difícil de probar, Complicado mantenimiento (el software nunca es terminado).
- JEE no es la solución: La buena tecnología no garantiza el éxito, así como la "mala" tecnología no garantiza el fracaso
- JEE tiene cosas buenas: servidores de aplicaciones y herramientas maduras
- Razones de fallas comunes: Pobre entendimiento de los requerimientos.
- Malas dinámicas de equipo

.

- Construir aplicaciones JEE puras: Sin lógica de negocio (y de otro tipo) en la Base de Datos donde Java es el centro del mundo. JEE es distribuido, entonces todo es remoto. La tecnología es más importe que el problema.
- Inversión de Control: Todos los frameworks tienen como característica común IoC, donde el control es invertido en el fw, para los contenedores IoC (como Spring), IoC es la forma en la cual buscan implementaciones, IoC es ambiguo y poco claro.
- DependecyInjection (DI):
    - Premisa: Los objetos no pueden vivir aislados, siempre dependen de otros.
    - Es necesario de "algo" que los una.
    - Un objeto ensamblador establece las dependencias de otros objetos.
    - El ensamblador debe saber cómo "inyectar" las dependencias.
- Creación de Objetos
    - Es un reto saber cuándo y cómo crear un objeto.
    - El patrón de diseño Factory ayuda
    - Al crear objetos usar SIEMPRE interfaces como referencia a ellos.
    - Un Factory no debería NUNCA regresar clases concretas.

Actualmente en el Departamento de Bienestar Politécnico se realizan los procesos de seguimiento, control y almacenamiento de información de los datos obtenidos de los estudiantes de forma manual, lo cual no presta ninguna clase de seguridad puesto que no cuentan con una base de datos o una aplicación informática que facilite estos procesos y ayude en la toma de decisiones.

## 1.2 Objetivos

### 1.2.1 Objetivo General

Realizar un análisis comparativo de frameworks MVC de java para el desarrollo de aplicaciones web empresariales. Caso práctico: Sistema de Bienestar Politécnico de la ESPOCH.

### 1.2.2 Objetivos Específicos

- Estudiar frameworks MVC de JAVA para el desarrollo de aplicaciones web empresariales.
- Realizar el análisis comparativo y definir el framework de desarrollo óptimo para la construcción de aplicaciones entre los distintos frameworks MVC de JAVA para el desarrollo de aplicaciones web empresariales.
- Desarrollar e implementar el Sistema de Bienestar Politécnico de la ESPOCH.

## 1.3 Justificación

### 1.3.1 Justificación Teórica

A medida que los frameworks de Aplicaciones Web, evolucionan, se requiere obtener mejores ventajas y beneficios en el entorno de programación y de esa manera poder disminuir el tiempo de desarrollo y facilidad para la construcción de Aplicaciones Web.

Los frameworks java MVC, son extremadamente eficaces y sorprendentemente versátiles, y se han convertido actualmente en un recurso inestimable proporcionando buenas perspectivas de futuro para el desarrollo de Aplicaciones Web.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo, una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original.

El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.).
El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de él, por ejemplo, el tipo de motor de bases de datos utilizado por la aplicación.

### 1.3.2 Justificación Aplicativa

El desarrollo de este proyecto nace del resultado de una evaluación de las necesidades vistas durante los procesos de control de las diferentes actividades que se realizan en el departamento de bienestar politécnico puesto que todo lo realizan manualmente sin tener ningún respaldo de la información generada en una base de datos, lo que es esencial para la toma de decisiones.

Por ende es necesario desarrollar una aplicación informática para facilitar estos procesos utilizando las herramientas informáticas actuales.

La aplicación informática a desarrollarse constara de los siguientes módulos:

- Módulo de auditoría

Este módulo tiene por finalidad controlar las acciones realizadas por los usuarios sobre el sistema (inserción, actualización, eliminación de datos).

- Módulos de autenticación de usuarios, administradores definidos por el Departamento de Bienestar Politécnico, basado en roles.
- Módulo de Control y Seguimiento de la Ficha del Estudiante.

  Tiene por objetivo controlar los procesos que se realiza para el ingreso de la información personal, académica, financiamiento de estudios, educación media, grupo familiar, vivienda familiar y patrimonio familiar de los estudiantes. También proporcionara reportes donde se visualice la información agrupada por parámetros establecidos por el Departamento de Bienestar Politécnico.

## 1.4 Hipótesis

La utilización de un framework MVC de Java facilitará el desarrollo de aplicaciones Web empresariales con mejor rapidez y seguridad.

# CAPÍTULO II

# 2. MARCO TEÓRICO

## 2.1 Aplicaciones Web Empresariales

### 2.1.1 Introducción

En la actualidad se está asistiendo a una auténtica revolución, los avances en las comunicaciones y las nuevas tecnologías están acercando la información al usuario final, así como facilitando su procesamiento. Uno de los cambios más importantes, tiene que ver con el soporte y canal de transmisión de la información, elinternet y las tecnologías web, han conseguido que el usuario esté familiarizado con información hipermedia, incluyendo texto, imágenes, audio y vídeo.

Cualquier ordenador conectado a la red constituye una fuente fácil de entrada de información y de servicios. Este hecho hace que cada vez cobre más fuerza la idea de que nos encontramos inmersos en una "sociedad de la información".

World Wide Web,  o simplemente Web, constituye uno de los intentos más recientes y a la vez más poderosos de sistematizar y simplificar el acceso a la información en internet.

Este nuevo sistema ha revolucionado la forma en que los usuarios se comunican y utilizan los servicios de la llamada "red de redes", y constituye la causa fundamental del espectacular aumento en el número de personas que usan internet y de la popularidad e importancia que ha adquirido en la actualidad.

La tecnología web permite el desarrollo de aplicaciones distribuidas basadas en el modelo cliente/servidor. Las aplicaciones web suponen un importante cambio de enfoque con respecto al desarrollo de aplicaciones tradicionales.

### 2.1.2 Definición

Las aplicaciones web son aplicaciones basadas en el modelo cliente/servidor que gestionan servidores web, y que utilizan como interfaz páginas web.

La colección de páginas son en una buena parte dinámicas (ASP, PHP, etc.), y están agrupadas lógicamente para dar un servicio al usuario.El acceso a las páginas está agrupado también en el tiempo (sesión).

Son aplicaciones basadas en el modelo cliente/servidor que gestionan datos almacenados en un servidor web, y que utilizan como interface páginas en formato HTML, conteniendo datos hipermedia. El usuario se comunica con la aplicación desde cualquier cliente conectado a la red.

### 2.1.3 Características

Su principal característica consiste en que la comunicación con el usuario se establece utilizando páginas web, que se pueden visualizar desde un navegador que se esté ejecutando en cualquier ordenador conectado a la red.

El código de la aplicación se puede ejecutar en el cliente, en el servidor o distribuirse entre ambos; además debido al gran volumen de información que se maneja, las aplicaciones web suelen utilizar una Base de Datos, para organizar y facilitar el acceso a la información.Lógica de negocio, es la parte más importante de la aplicación, define los procesos que involucran a la aplicación.

- Conjunto de operaciones requeridas para proveer el servicio.
- Administración de los datos.
- Manipulación de BD y archivos.
- Interfaz, los usuarios acceden a través de navegadores, móviles, PDAs, etc.
- Funcionalidad accesible a través del navegador.
- Limitada y dirigida por la aplicación.
- Acceso a bases de datos, normalmente con BD relacionales.

  Transaccionales: Propiedades ACID: Atomicity (atomicidad), Consistency (coherencia), Isolation (aislamiento), Durability (permanencia).

- Escalable: Deberían poder soportar más carga de trabajo sin necesidad de modificar el software (sólo añadir más máquinas).
- Disponibilidad: Idealmente no deben dejar de prestar servicio.
- Seguras: No todos los usuarios pueden acceder a la misma funcionalidad.

- Integración: Es preciso integrar aplicaciones construidas con distintas tecnologías.
- Separación clara entre la interfaz gráfica y el modelo.
- Modelo: encapsula la lógica de negocio.
- El modelo debería ser reusable con distintas interfaces gráficas.

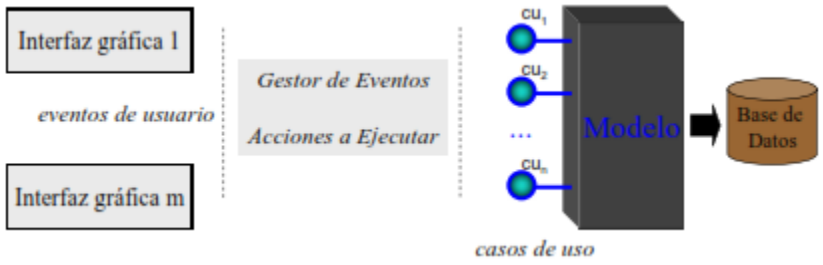



**Figura II. 1Modelo Reusable**

**Fuente:** http://www.tic.udc.es/~fbellas/teaching/is-2002-2003/Tema1.pdf

- Facilidad de manejo: ya que la interacción con el usuario se establece en base a elementos a los que está cada vez más acostumbrado, páginas web, que le permiten conocer la funcionalidad del sistema con poco esfuerzo.
- Accesibilidad: las aplicaciones web son accesibles desde cualquier punto de la red, lo cual significa que un usuario autorizado (se pueden establecer controles de acceso) puede utilizarla si dispone de cualquier conexión a Internet (salvo el caso de que se trate de una aplicación que funcione en una red TCP/IP propia, en cuyo caso será necesario disponer de acceso a la misma).
- Portabilidad: los navegadores web se han desarrollado para todo tipo de máquinas, por lo que cualquier usuario de internet, dispone de la herramienta básica para lanzar la aplicación

- Facilidad de desarrollo: en este sentido, hay dos aspectos a destacar que determinan el desarrollo de este tipo de sistemas. En primer lugar, la descomposición intrínseca en componentes, así como al hecho de que en cierta medida, algunos aspectos que tienen que ver con el carácter distribuido de la aplicación están resueltos de antemano (por ejemplo el protocolo HTTP controla el acceso a datos en el servidor).
- La seguridad en Internet, uno de sus puntos más débiles, constituye uno de los aspectos que hoy más interesan, y a cuya investigación se dedica más esfuerzo y recursos, sobre todo si además se une el hecho de que es necesario realizar un control de utilización de la aplicación. Por ejemplo, cuando se envía información confidencial sobre nuestra persona, a través de formularios CGI, nos interesa que el servidor conozca los datos, pero no el resto de la Red, especialmente si se está realizando una transacción comercial electrónica y se envía el número de tarjeta de crédito, o simplemente la dirección. Se pone de manifiesto la necesidad de asegurar mediante algún mecanismo la intimidad y la integridad en las sesiones con el servidor Web.

### 2.1.4 Arquitectura

Una aplicación web es proporcionada por un servidor web y utilizada por usuarios que se conectan desde cualquier punto vía clientes web (navegadores).Las aplicaciones web se modelan mediante lo que se conoce como modelo de capas.Una capa representa un elemento que procesa o trata información.

**Tipos:**

- Modelo de dos capas: La información atraviesa dos capas entre la interfaz y la administración de los datos.

- Modelo de n-capas: La información atraviesa varias capas.El más habitual es el modelo de tres capas

**Arquitectura en dos capas**

Gran parte de la aplicación corre en el lado del cliente (fatclient).

Capas:

- Cliente (fatclient):

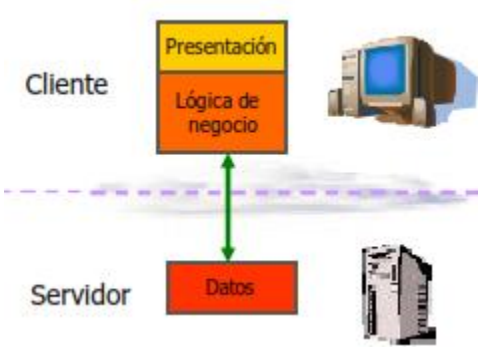
  La lógica de negocio está inmersa dentro de la aplicación que realiza el interfaz de usuario, en el lado del cliente.

- Servidor:

  Administra los datos.

Limitaciones.

- Es difícilmente escalable
- Número de conexiones reducida
- Alta carga de la red
- La flexibilidad es restringida
- La funcionalidad es limitada



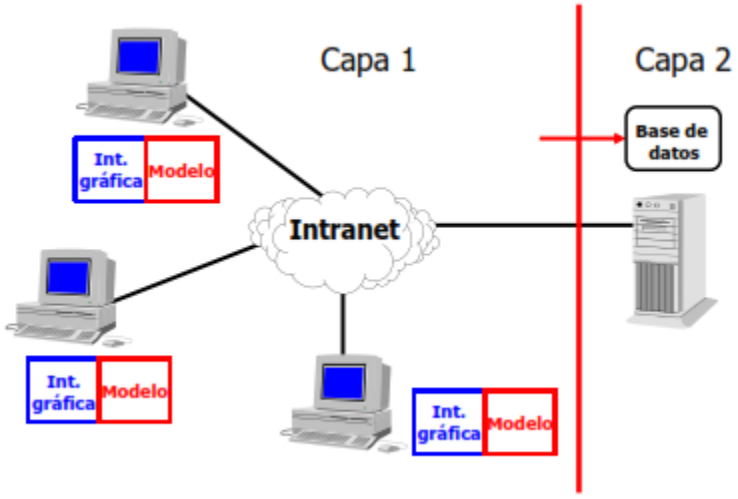



**Figura II. 2Arquitectura en 2 capas**

**Fuente:** http://jlcc170.wordpress.com/

- Problema

  Cambios en la implementación de la capa modelo => recopilación de toda la aplicación y reinstalación en clientes.

  - Cambios de drivers de acceso a la BD.
  - Cambios en la lógica del modelo.
  - Cambio de tipo de BD.

- Solución:

  Modelo en servidor intermedio.Un cambio en la implementación del modelo sólo afecta al servidor.

  Clientes standalone.Sólo disponen de la interfaz gráfica.Acceden al servidor que implementa el modelo.

**Arquitectura en tres capas**

Diseñada para superar las limitaciones de las arquitecturas ajustadas al modelo de dos capas

Introduce una capa intermedia (la capa de proceso) entre presentación y los datos

- Los procesos pueden ser manejados de forma separada al interfaz de usuario y a los datos
- La capa intermedia centraliza la lógica de negocio, haciendo la administración más sencilla.

Pueden integrar datos de múltiples fuentes. Las aplicaciones web actuales se ajustan a este modelo.

Capas:

- Capa de presentación (parte en el cliente y parte en el servidor)

  Recoge la información del usuario y la envía al servidor (cliente)

  Manda información a la capa de proceso para ser procesado

  Recibe los resultados de la capa de proceso

  Generan la presentación

  Visualizan la presentación al usuario (cliente)

- Capa de proceso (servidor web)

  Recibe la entrada de datos de la capa de presentación

  Interactúa con la capa de datos para realizar operaciones

Manda los resultados procesados a la capa de presentación

- Capa de datos (servidor de datos)

  Almacena los datos

  Recupera datos

  Mantiene los datos

  Asegura la integridad de los datos

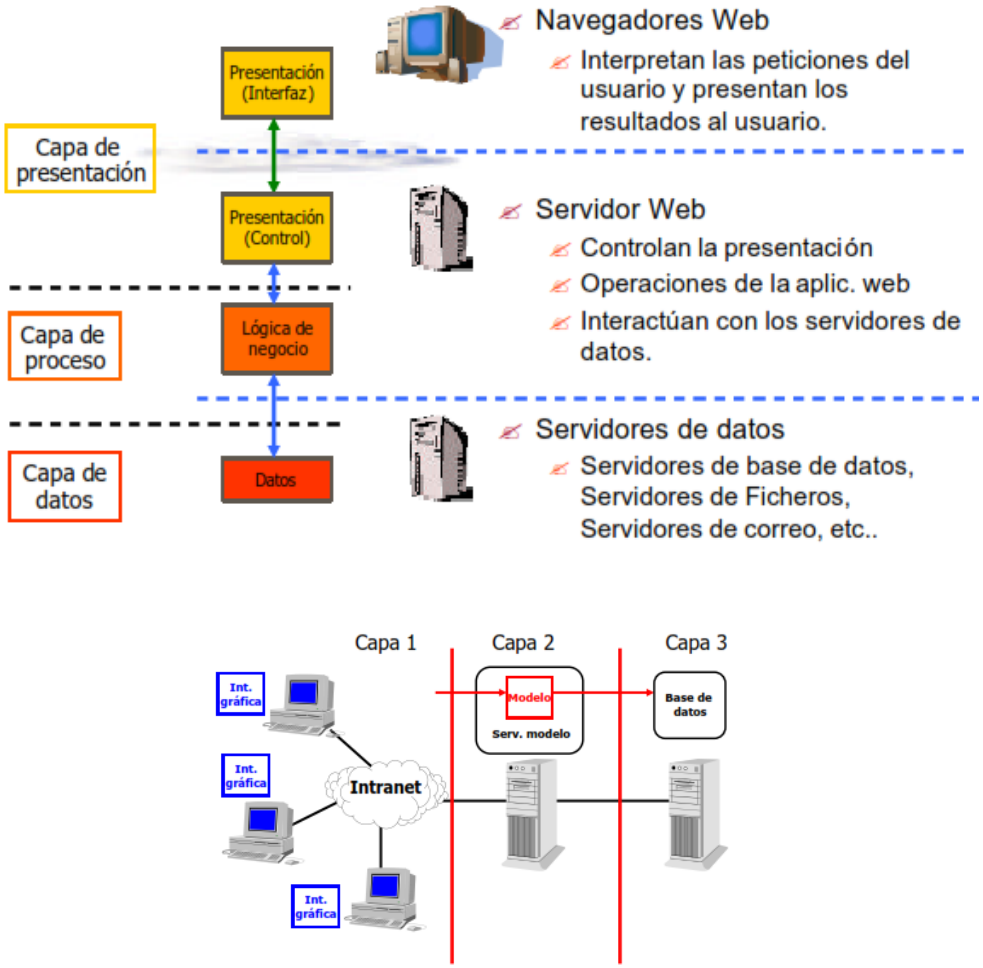



**Figura II. 3Arquitectura en 3 capas**

**Fuente:** http://www.tic.udc.es/~fbellas/teaching/is-2002-2003/Tema1.pdf

**Arquitectura en cuatro capas**

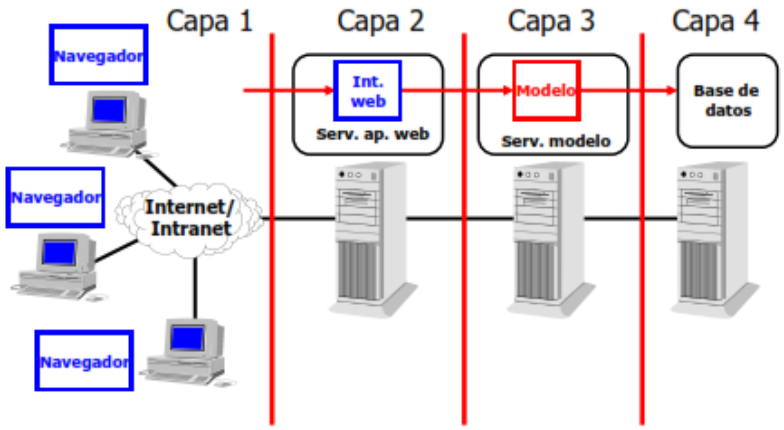



**Figura II. 4Arquitectura en cuatro capas**

**Fuente:** http://www.tic.udc.es/~fbellas/teaching/is-2002-2003/Tema1.pdf

Para una aplicación web, la arquitectura en tres capas es más eficiente. En la arquitectura en tres capas, la comunicación entre la interfaz gráfica y el modelo es local.

En la arquitectura en cuatro capas, la comunicación entre la interfaz gráfica y el modelo es remota.

### 2.1.5 Java EE

Java EE es un conjunto de especificaciones de APIs Java para la construcción de aplicaciones empresariales. La mayor parte de las abstracciones de las APIs corresponden a interfaces y clases abstractas. Existen múltiples implementaciones de distintos fabricantes, incluso algunas OpenSource. Una aplicación construida con Java EE no depende de una implementación particular.

Java 2 PlataformEnterprise Edition (J2EE) es un conjunto de especificaciones y prácticas coordinadas que juntas permiten soluciones para el desarrollo, implantación y administración de aplicaciones de múltiples capas con un servidor centralizado. Añade las capacidades necesarias para proveer una plataforma Java completa, estable, segura y rápida a un nivel empresarial. Provee valor al reducir significativamente el costo y complejidad del desarrollo e implantación de soluciones de múltiples capas, lo que resulta en servicios que pueden ser rápidamente implementados y fácilmente aumentados.

**Componentes de J2EE**

La especificación de J2EE define las siguientes capas de una aplicación:

- Capa de cliente. Corre en la máquina cliente
- Capa de web. Corre en el servidor J2EE
- Capa de negocio. Corre en el servidor J2EE
- Capa de Sistema de Información Empresarial (EIS). Corre en el servidor EIS

**Capa cliente**

Clientes web. Consisten de dos partes: páginas web dinámicas y un navegador web. Se les conoce como "clientes livianos" porque no hacen querys a bases de datos, ni ejecutan reglas complejas de negocio, ni se conectan a aplicaciones heredadas. Dichas operaciones son manejadas por el servidor J2EE

Applets es una pequeña aplicación cliente escrita en Java que es ejecuta por la máquina virtual de Java instalada en el navegador web

Cliente de aplicación. Son aplicaciones que corren en la máquina cliente y permiten a los usuarios manejar tareas que requieren una interfaz más rica que la que es otorgada por el html o xml. Generalmente tienen una interfaz gráfica (GUI) creada usando Swing o Abstract Windows Toolkit (AWT). También es posible usar un intérprete de comandos

JavaBeans. Los clientes pueden tener componentes basados en JavaBeans para el manejo de flujo de datos entre un cliente de aplicación o applet y los componentes que corren en el servidor J2EE o entre componentes del servidor y una base de datos. Dichos JavaBeans no se consideran componentes del J2EE

Comunicaciones con servidor J2EE. El cliente se comunica con el componente de negocio ya sea directamente o a través de páginas JSP o Servlets que corren en el componente web

**Capa web**

Servlets: son clases del lenguaje Java que procesan solicitudes y construyen respuestas de forma dinámica

Páginas JSP: Son documentos de texto que son ejecutados como Servlets pero permiten un acercamiento más natural a la creación de contenido estático

Applets y Páginas html estáticas: Son usadas por componentes web pero no se consideran componentes de J2EE. Lo mismo ocurre con las clases utilitarias y JavaBeans del lado del servidor

**Capa del negocio**

Es toda la parte lógica que resuelve o satisface las necesidades de un dominio de negocio particular, dicha lógica es manejada por Enterprise JavaBeans. Hay tres tipos de enterprisebeans: beans de sesión, beans de entidad y beans manejados por mensajes

Maneja el software del sistema de información empresarial. Maneja sistemas de infraestructura empresariales como planificación de recursos empresariales (ERP), procesamiento de transacciones del mainframe, sistemas de base de datos y otros sistemas de información heredados

**Servicios de la plataforma J2EE**

- Seguridad

Portabilidad, transparencia, aislamiento, extensibilidad, flexibilidad, independencia, testing de compatibilidad, interoperabilidad segura

- Transaccionalidad

Conjunto de operaciones asociadas bajo un orden y que cumplen las propiedades ACID. Los atributos de transacción determinan el alcance de una transacción especificados en el descriptor de despliegue.

- Servicio de nombres

Servicio de directorio JNDI (Java NamingDirectory Interface) API a través del cual los componentes J2EE encuentran los objetos, extensión de la plataforma Java que provee una interfaz unificada a los múltiples servicios de nombre y directorio.

Servicio de directorio JNDI (Java Naming Directory Interface) Un nombre JNDI está atado a su objeto a través del servicio de directorios y naming del servidor J2EE. Una referencia a recurso es un elemento de un descriptor de despliegue que identifica el nombre en código del recurso. Este a su vez referencia a una fábrica de conexiones para el recurso

- Conectividad remota

### 2.1.6 Ventajas y Desventajas

**Ventajas de las aplicaciones web**

- Acceso desde cualquier computadora: al estar en internet lo único que se necesita es una conexión en un computador para empezar a trabajar, son útiles cuando la gente no puede llevarse su computador a otro lugar.
- No se necesita un sistema operativo específico: es quizá de lo mejor que ofrece estas aplicaciones, no necesita un SO definido porque al estar en la web quizá el problema de compatibilidad de navegador sea un pequeño problema pero en general las aplicaciones están hechas para que sirvan en cualquier navegador.
- Extrapolación y sindicación absoluta. El hecho de que todas las aplicaciones se realicen sobre Web, va a permitir que entre ellas se pueda compartir toda la información (*xml*).
- Propagación inmediata de contenido e información (RSS) que va a permitir un mejor desarrollo de la estructura en red**.**
- Uso de otras fuentes para desarrollar nuevas aplicaciones. Esta cuestión va a permitir que el desarrollo de nuevas aplicaciones se centren en la aportación de valor añadido**,** centrando los recursos en lo nuevo, y sacando partido de lo hecho por otros.
- Aplicaciones (software) como servicio y no como producto.

Esto elimina el costo de acceso de las pymes a la tecnología más moderna, anulando las barreras de entrada a competir en los mercados por esa vía.El pago se hace por servicio, lo que en pequeñas empresas es una muy pequeña cantidad en relación al costo de la plataforma que obtiene y fuera de su alcance a través de inversiones que las sacarían del mercado.

- Ubicuidad. Las aplicaciones basadas en web pueden desarrollarse en cualquier terminal (y no necesariamente en los PC): ordenadores, móviles, PDAs, TV digital. Esto va a permitir tener la información en todo momento y desde cualquier terminal con conexión a internet.
- Cooperación. Las necesidades de la sociedad y empresa red radican principalmente en la cooperación entre los diferentes actores**,** permitiendo anular prácticamente los costos**.**
- Seguridad. Si bien es un aspecto en debate, a nivel de pymes la capacidad de seguridad y de protección de datos de servidores de empresas profesionales será siempre mucho mayor que la mantenida en servidores compartidos o en los mismos ordenadores de gestión diaria. Pérdidas de datos por fallos del sistema, virus, ataques son constantes en los ordenadores personales sin que se mantengan copias de seguridad adecuadas y siendo el coste de restauración muy elevado para estas empresas.

**Desventajas de las aplicaciones web**

- Tus datos no los tienes tu: un problema algo serio, ya que si se cae el servicio o hackean la aplicación, los datos y documentos quedarían expuestos fácilmente. Las aplicaciones

web deben estar en constante resguardo porque si no son seguras la posibilidad de que no encuentres tus documentos es mucha.

- Compatibilidad con los idiomas: en general estas aplicaciones vienen en inglés y el soporte a más lenguajes es muy difícil, sin mencionar que las versiones en otros idiomas tienen menos funciones que la que está en inglés, por lo que mucha gente no siente a gusto con una aplicación a medias.
- Espacio de almacenamiento: muchos servicios dan un espacio limitado para los archivos, dependiendo del tipo de aplicación.
- La seguridad de datos confidenciales, como la contabilidad, facturación, al estar almacenados en servidores ajenos. Centrándonos en las necesidades de las pymes es probable que los datos estén en mejor recaudo de servidores de empresas dedicadas a ello que en ordenadores que normalmente son mucho más vulnerables a ataques de virus, troyanos, espías.
- La conexión a Internet, la dependencia del sistema a la conexión de Internet sigue siendo una barrera a su adopción. Si bien las empresas de telecomunicaciones cada vez son más fiables y mantienen mejores conexiones, siempre existe la posibilidad de quedarse sin conexión en la mitad de una jornada laboral, lo que impediría el uso del sistema.

## 2.2 Spring un framework de aplicación

### 2.2.1. SPRING

**Introducción**

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 ExpertOne-on-one J2EE Design And Development (Programmertoprogrammer), 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 códigos abiertos o comerciales que proveyeran las características requeridas para aquellos desarrollos.

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 tradicional; 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-orientedprogramming, AOP) consiguió hacer más popular su paradigma de programación en la comunidad Java.

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.

**Definición**

Spring es un framework de aplicaciones Java/J2EE desarrollado usando licencia de OpenSource.

Se basa en una configuración a base de javabeans bastante simple. Es potente en cuanto a la gestión del ciclo de vida de los componentes y fácilmente ampliable. Es interesante el uso

de programación orientada a aspectos (IoC). Tiene plantillas que permiten un más fácil uso de Hibernate, iBatis, JDBC, se integra "de fábrica" con Quartz, Velocity, Freemarker, Struts, Webwork2 y tienen un plugin para eclipse.

Ofrece un ligero contenedor de Bean para los objetos de la capa de negocio, DAOs y repositorio de Datasources JDBC y sesiones Hibernate. Mediante un xml definimos el contexto de la aplicación siendo una potente herramienta para manejar objetos Singleton o "factorías" que necesitan su propia configuración.

**Objetivos**

- El objetivo de Spring es no ser intrusivo, aquellas aplicaciones configuradas para usar beans mediante Spring no necesitan depender de interfaces o clases de Spring, pero obtienen su configuración a través de las propiedades de sus beans.
- La meta a conseguir es separar los accesos a datos y los aspectos relacionados con las transacciones, para permitir objetos de la capa de negocio reutilizables que no dependan de ninguna estrategia de acceso a datos o transacciones.
- Proveer el soporte para desarrollar aplicaciones empresariales con java.
- Enfocarnos en resolver nuestro problema de dominio.

**El rol en la arquitectura de aplicaciones empresariales.**

- El core de Spring nos da la posibilidad de configurar nuestra aplicación de maneras sencillas, así también el de poder realizar una integración con aplicaciones empresariales, realizar testing y poder manejar el acceso a la data.
- Permite integrar y configurar componentes de manera sencilla; parte de los componentes son plain java objects.

- Integrar pool de conexiones de base de datos, transacciones, seguridad, messaging, acceso remoto, entre otras.
- Realizar testing, desacoplando los objetos desde nuestros ambientes (testing, producción), realizar test unitarios o de integración.
- El acceder a la data de una manera mucho más fácil, Spring nos entrega soporte para la gran parte de las tecnologías de acceso de datos, como lo son JDBC,JPA,JDO,HIBERNATE,IBATIS, ya ahora llamado MyIbatis, otra característica es que maneja los recursos por nosotros, un ejemplo de esto es que puede adquirir la conexión, participar en la transacción, manejar las exceptions, procesar resultados, etc.; nos provee de helpers como lo es JdbcTemplate.
- Soporte para el desarrollo de aplicaciones Web, permite la integración con JSF, Struts, Velocity, FreeMaker y algunos otros, nos permite trabajar con el patrón de diseño MVC, rendear las vistas,validación de formularios, manejo de estados con web flow, acciones de usuarios, etc., así también, aplicar una capa de seguridad con Spring Security.
- Soporte en el desarrollo de aplicaciones empresariales, desarrollando webservices, asegurando servicios con accesos de control, planificando jobs y procesos, integrando servicios de mensajería.

**Versiones**

**Spring** es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java.

- La primera versión fue escrita por Rod Johnson, quien lo lanzó junto a la publicación de su libro ExpertOne-on-One J2EE Design and Development (WroxPress, octubre 2002). El framework fue lanzado inicialmente bajo la licencia Apache 2.0 en junio de 2003. El primer gran lanzamiento fue la versión 1.0, que apareció en marzo de 2004 y fue seguida por otros hitos en septiembre de 2004 y marzo de 2005. La versión 1.2.6 de Spring Framework obtuvo reconocimientos JoltAwards y JaxInnovationAwards en 2006.
- Spring Framework 2.0 fue lanzada en 2006, la versión 2.5 en noviembre de 2007.
- Spring 3.0 en diciembre de 2009, y Spring 3.1 dos años más tarde.
- La versión actual es 3.2.0. El inicio del desarrollo de la versión 4.0 fue anunciado en enero de 2013.

Si bien las características fundamentales de Spring Framework pueden ser usadas en cualquier aplicación desarrollada en Java, existen variadas extensiones para la construcción de aplicaciones web sobre la plataforma Java EE. A pesar que no impone ningún modelo de programación en particular, este framework se ha vuelto popular en la comunidad al ser considerado una alternativa, sustituto, e incluso un complemento al modelo EJB (Enterprise JavaBean).

**Arquitectura**

Spring es bastante grande, por ello cuenta con una arquitectura dividida en siete capas o módulos, lo cual permite tomar y ocupar únicamente las partes que interesen para el proyecto y juntarlas con gran libertad.

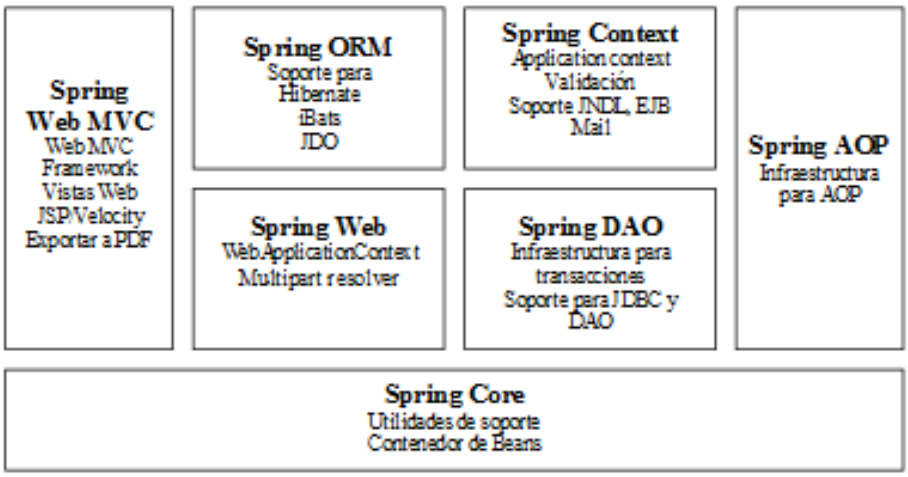



**Figura II. 5Módulos de Spring Framework**

**Fuente:** http://oness.sourceforge.net/proyecto/html/ch06s03.html

## Componentes

- **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 (DependecyInjection). El núcleo de Spring es el paquete org.spingframework.beans el cual está diseñado para trabajar con JavaBeans.

- **Bean Factory**

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

Todas las BeanFactories implementan la interfaseorg.springframework.beans.factory.BeanFactory, con instancias que pueden ser accesadas a través de esta interfaz. Además también soportan objetos de dos modos diferentes.

- Singleton: Existen únicamente una instancia compartida de un objeto con un nombre particular, que puede ser regresado o llamado cada vez que se necesite. Este es el método más común y el más usado. Este modo está basado en el patrón de diseño que lleva el mismo nombre.
- Prototype: También conocido como non-singleton, en este método cada vez que se realiza un regreso o una llamada, se crea un nuevo objeto independiente.

La implementación de BeanFactory más usada es org.springframework.beans.factory.xml.XmlBeanFactory que se encarga de cargar la definición de cada Bean que se encuentra guardad en un archivo XML y que consta de:
Id (que será el nombre que él se le conocerá en las clases), clase (tipo de Bean), Singleton o Prototype (modos del Bean antes mencionados), propiedades, con sus atributos name, value y ref, además argumentos del constructor, método de inicialización y método de destrucción. A continuación se muestra un ejemplo de un Bean.

```
<beans>
    <bean id=”exampleBean” class=”eg.ExampleBean” singleton=”true”/>
    <property name=”driverClassName” value=”com.postgreSql.jdbc.Driver”/>
</beans>
```

Como se muestra en el ejemplo la base de este documento SML es el elemento <beans>, y adentro puede contener uno o más elementos de tipo <vean>, para cada una de las diferentes funciones que se requieran.

Para cargar dicho XML se le manda un InputStream al constructor de XmlBeanFactory de la siguiente manera.

BeanFactory fact = new XmlBeanFactory (new FileInputStream (“bean.xml”));

Esto sin embargo no quiere decir que sea instanciado directamente, una vez que la definición es cargada, únicamente cuando se necesite el Bean se creara una instancia dependiendo de sus propiedades. Para tomar un Bean de un Factory simplemente se usa el método getBean(), mandándole el nombre del Bean que se desea;

```
Bean myBean = (MyBean) factory.getBean ("myBean");
```

- **Inversion of Control**

  “Inversion of Control se encuentra en el corazón de Spring”. BeanFactory utiliza el patrón de Inversión de Control o como se le conoce IoC, que es una de las funcionalidades más importantes de Spring. Esta parte se encarga de separar del código de la aplicación que se está desarrollando, los aspectos de configuración y las especificaciones de dependencia del framework. Todo esto configurando los objetos a través de Inyección de Dependencia o DependecyInjection.

  Una forma sencilla de explicar el concepto de IoC es el “Principio Hollywood”: “No me llames, yo te llamare a ti”. Traduciendo este principio a términos de este trabajo, en lugar de que el código de la aplicación llame a una clase de una librería, un framework que utiliza IoC llama al código. Es por esto que se le llama “Inversión”, ya que invierte la acción de llamada a laguna librería externa

- **DependencyInjection**

Es una forma de inversión de Control, que está basada en constructores de Java, en vez de usar interfaces específicas del framework. Con este principio en lugar de que el código de la aplicación utilice 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.

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 de 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.

**Setter Injection:** en este tipo la inyección de Dependencia es aplicado por medio de métodos JavaBeans setters, que a la vez tiene un getter respectivo.

**Constructor Injection:** esta Inyección es a través de los argumentos del constructor

A continuación se muestra un ejemplo 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 AccesObject). Con el método SetterInjection se puede configurar ServiceImpl utilizando las propiedades de un JavaBean para satisfacer estas 2 dependencias.

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

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

```
Public class ServiceImpl implements Service {
    privateint timeout;
    privateAccountDaoaccountDao;
publicServiceImpl (int timeout, AccountDaoaccountDao) {
    this.timeout = timeout;
this.accountDao = accountDao;
}
```

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

- **Spring Context**

El módulo BeanFactory del núcleo de Spring es lo que lo hace un contenedor, y el módulo de contexto es lo que hace un framework.

En si Spring Context es un archivo de configuración que provee de información contextual al framework general. Además provee servicios Enterprise como JNDI, EJB, e-mail, validación y funcionalidad de agenda.

- **ApplicationContext**

ApplicationContext es una subinterfaz de BeanFactory, ya que org.springframework.context.ApplicationContext es una subclase de BeanFactory.

En si todo lo que se pueda realizar una BeanFactory también lo puede realizar ApplicationCOntext. En si agrega información de la aplicación que puede ser utilizada por todos los componentes:

Además brinda las siguientes funcionalidades extra:

- Localización y reconocimiento automático de las definiciones de los Beans
- Cargar múltiples contextos
- Contextos de herencia
- Búsqueda de mensajes para encontrar su origen
- Acceso a recursos
- Propagación de eventos, para permitir que los objetos de las aplicaciones puedan publicar y opcionalmente registrarse para ser notificados de los eventos.
- Agrega soporte para internacionalización (il8n)

En algunos casos es mejor utilizar ApplicationContext ya que obtienes más funciones a un costo muy bajo, en cuanto a recursos se refiere.

Ejemplo de un ApplicationContext:

```
ApplicationContextct= new FileSystemXmlApplicationCOntext (“c:\bean.xml”);
ExampleBeaneb = (ExampleBean) ct.getBean(“exampleBean”);
```

- **Spring AOP**

Aspect-orientedprogramming, o AOP, es una técnica que permite a los programadores modularizar ya sea las preocupaciones crosscutting, o el comportamiento que corta a través de las divisiones de responsabilidad, como logging, y manejo de transacciones. El núcleo de construcción es el aspecto, que encapsula comportamiento que afectan a diferentes clases, en módulos que pueden ser utilizados.

AOP se puede utilizar para:

- Persistencia
- Manejo de transacciones
- Seguridad
- Logging
- Debugging

AOP es un enfoque diferente y un poco más complicado de acostumbrarse en comparación con OOP (ObjectOrientedProgrammming). Rob Johnson prefiere referirse a AOP como un complemento en lugar de como un rival o un conflicto.

Spring AOP es portable entre servidores de aplicación, funciona tanto en servidores Web como en contenedores EJB.

Spring AOP soporta las siguientes funcionalidades:

- **Intercepción:** se puede insertar comportamiento personalizado antes o después de invocar a un método en cualquier clase o interfaz

- **Introducción:** Especificando que un advice (acción tomada en un punto particular durante la ejecución de un programa) debe causar que un objeto implemente interfaces adicionales.
- **Poincuts dinámicos y estáticos:** para especificar los puntos en la ejecución del programa donde debe de haber intercepción.

Spring implementa AOP utilizando proxies dinámicos. Además se integra transparentemente con los BeanFactory que existen. En el ejemplo siguiente e muestra cómo definir un proxy AOP.

```
<bean id=”myTest”
        Class=”org.springframework.aop.Framework.ProxyFactorBean”>
        <property  name=”proxyInterfaces”>
                <value>org.sprigframwework.beans.ITestBean</value>
        <property>
        <property  name=”interceptorNames”>
                <list>
                        <value>txInterceptor</value>
                        <value>target</value>
                </list>
        <property>
</bean>
```

- **Spring ORM**

En lugar de que Spring proponga su propio módulo ORM (Object-RelationalMapping), para los usuarios que no se sientan confiados en utilizar simplemente JDBC, propone un módulo que soporta los frameworks ORM más populares del mercado, entre ellos:

- 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.
- iBATIS SQL Maps (1.3 y 2.0): una solución sencilla pero poderosa para hacer externas las declaraciones de SQL en archivos XML.

- Apache OJB1 (1.0): plataforma de mapeo O/R con múltiples APIs para clientes.
- Entre otros como JDO (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:

- **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 SeesionFactories 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 el 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 pedazo que se pueden aislar y probar por separado, ya sean sesiones o una fuente de datos (Datasources).

**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.

- **DAO y JDBC**

Existen dos opciones para llevar a cabo el acceso, conexión y manejo de base 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.

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 de datos. Este es un ejemplo de cómo se hace la inserción de datos en una base de datos utilizando JDBC tradicional:

```
Public void insertPerson (Person person) throws SQLException {
//Se declaran recursos
Connection con = null;
PreparedStamentstmt = null;
try {
//Se abre una conexión
Con = dataSource.getConnection ();
//Se crea la declaración
Stnt = con.prepareStatement (“insert into person (“ + “id, firstName, lastName)
values (?,?,?)”);
//Se introducen los parámetros
Stnt.setINt(0, person.getId().intValue());
Stnt.setString(1, person.getFirstName());
Stnt.setString(2, person.getLastName());
//Se ejecuta la instrucción
Stnt.executeUpdate();
//Se atrapan las excepciones
Catch(SQLException e){
        LOGGER.error(e);
}
finally {
                //Se limpian los recursos
        try { if (stmt ¡=null) stmt.close( );}
catcah (SQLException e) {LOGGER. Warn(e);}
try { if (con ¡= null) con.close( );}
catch(SQLException e) {LOGGER.warn (e );}
        }
}
```

De todo este código únicamente el 20% es para un propósito específico, el otro 80% es de propósito general en cada una de las diferentes operaciones de acceso a una base de datos. Esto no quiere decir que no sea importante si no que al contrario es lo que le da la estabilidad y robustez al guardado y recuperación de información.

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.

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

En el ejemplo que se muestra a continuación la clase base es JdbcTemplate, que es la clase central que maneja la comunicación con la base de datos y el manejo de excepciones. Esta clase provee varios métodos que realizan la carga pesada y resultan bastante convenientes, como el método execute( ) que aparece en el ejemplo, el cual recibe un comando de SQL como su único parámetro. Para este ejemplo se supone que se tiene una tabla ya creada dentro de una base de datos utilizando el siguiente comando SQL:

```
create table mytable (id integer, name varchar (100))
```

Ahora se quiere agregar algunos datos de prueba:

```
importorg.springframework.jdbc.core.JdbcTemplate;
importorg.springframework.jdbc.datasource.DriverManagerDataource;
public class MinimalTestextendiTestCase{
        privateDriverManagerDataSourcedataSource;

        public void setup ( ){
        dataSource = new DriverManagerDataSource ( );
        dataSource.setDriverClassName (“org.hsqldb.jdbcDriver”);
        dataSource.setUrl(“jdbc:hsqldb:hsql://localhost:”);
        dataSource.setUsername (“sa”);
        dataSource.setPassword(“ ”);
        JdbcTemplateJt = new JdbcTemplate (dataSource);
        Jt.execute (“delete from mytable”);
        Jt.execute (“insert into mytable (id,name) values (1, ‘John’)”);
        Jt. Execute(“insert into mytable (id,name) values (2, ‘Jane’)”);
}
publicvoidtestSomething ( ){
// Aquí va el codigo de prueba que se puede aplicar
}
```

Como se pudo apreciar en el ejemplo anterior no existe manejo de excepciones, ya que el JdbcTemplate se encarga de atrapar todas las SQLExceptions y las convierte a una subclase de DataAccessException, que es la clase que se encuentra en el nivel superior de la jerarquía de las excepciones de acceso a datos. También el JdbcTemplate se encarga de controlar la conexión con la base de datos. Y se puede evitar configurar el DataSource en DAO, únicamente se tiene que establecer a través de la inyección de dependencia.

- **Spring Web**

El modulo 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.

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 (domainobjects o businessobjects).

- **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.

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

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 e bastante flexible
- Provee interceptores (interceptors) al igual que controladores.
- Spring no obliga a utilizar JSP como una 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 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 businessobjects.

La arquitectura básica de Spring MVC esta ilustrada en la Figura II.6.

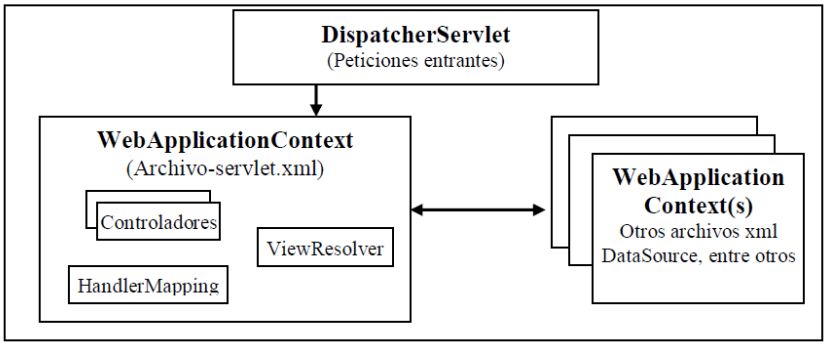



**Figura II. 6Arquitectura básica del Web MVC de Spring**

**Fuente:** http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/sanchez_r_ma/capitulo3.pdf

Para intentar comprender cada parte de la arquitectura del Web MVC de Spring es necesario conocer el ciclo de vida de una petición o request.

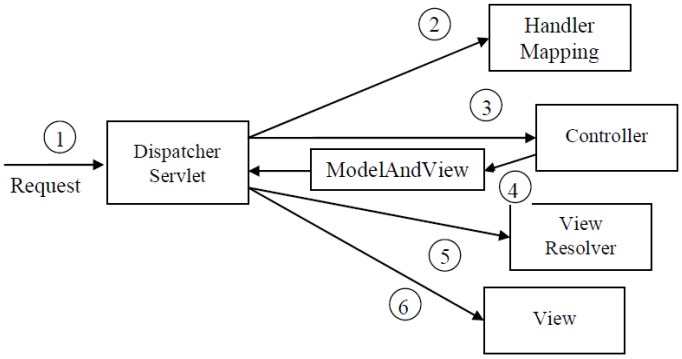



**Figura II. 7Ciclo de vida de un request**

**Fuente:** http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/sanchez_r_ma/capitulo3.pdf

1. El navegador manda un request y lo recibe un DispatcherServlet
2. Se debe escoger que Controller manejara el request, para esto el HandlerMapping mapea los diferentes patrones de URL hacia los controladores, y se le regresa al DispatcherServlet el Controller elegido.
3. El Contoller elegido toma el request y ejecuta la tarea
4. El Controller regresa un ModelAndView al DispatcherServlet
5. Si el ModelAndView contiene un nombre lógico de un View se tiene que utilizar un ViweResolver para buscar ese objeto View que representara el requestmodificado.
6. 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 el envío 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.

**DispatcherServlet**

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).

```
<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 buscara como está indicado por el tag<servlet-name> el contexto de aplicación (appplicationContext) correspondiente con el nombre que haya puesto dentro de ese tag acompañado de la terminación –servlet.xml, en este caso buscara 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 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.

**HandlerMappings**

Existen diversas maneras en que el DispatcherServlet pueda determinar y saber que controlador es el encargado de procesar un request, y a que vean del ApplicationContext se lo puede asignar. Esta tarea la lleva a cabo el HandlerMapping o el manejador de mapeo, existen 2 tipos principales que son los que más se usan:

**BeanNameUrlHandlerMapping:** mapea el URL hacia el controlador en base al nombre del vean del controlador. Ejemplo de cómo se declara:

Esta es la declaración principal del HandlerMapping:

<bean id=”beanNameUrlMapping”

class=”org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping”/>

A continuación se escribe cada uno de los diferentes URLs que se vayan a utilizar en la aplicación, poniéndolo en el atributo name del vean y en el atributo class la clase del Controlador que vaya a procesar ese dicho request, y finalmente se ponen las diferentes propiedades que utilizar dicho controlador.

```
<bean name=”ejemplo.htm” class=”web.ExampleController”>
<property name=”Servicio”>
<ref bean=”Servicio”/>
</property>
</bean>
```

Así cuando el usuario entre a una página con el URL /ejemplo.htm el request que haga será dirigido hacia el controlador ExampleController.

SimpleUrlHandlerMapping: mapea el URL hacia el controlador basando en una colección de propiedades declarada en el applicationContext. Ejemplo de cómo declarar un HandlerMapping de este tipo:

```
<bean id=”simpleUrlMapping” class=
“org.springframework.web.servlet.handler.SimpleUrlHandler
Mapping”>
<property name=”mappings”>
<props>
<prop key=”/ejemplo.htm”>ExampleController</prop>
<prop key=”/login.htm”>LoginController</prop>
</props>
</property>
</bean>
```

En este HndlerMapping se declara una lista de propiedades en las cuales se pondrá cada uno de los URLs como una propiedad, con el URL como atributo key y como valor el nombre del Bean del controlador que sea responsable de procesar la petición o request. Por ejemplo el URL/ login.htm será responsabilidad del controlador con el nombre del beanLoginController.

Se pueden trabajar con múltiples HandlerMappings por si son necesarios en diferentes situaciones, únicamente se le tiene que agregar la propiedad ‘order’ para poder asignarle en qué orden el DispatcherServlet va a considerarlas para poder invocarlas.

**View Resolvers**

En el Spring MVC una vista o Views es un Bean que transforma los resultados para que sean visibles para el usuario y los pueda interpretar de una mejor forma. En si un View Resolver es cualquier Bean que implemente la interfaz org.springframework.web.servlet.ViewResolver. Esto quiere decir que un ViewResolver 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.

Spring MVC cuenta con cuatro ViewResolvers diferentes:

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

El View Resolver m utilizado es el InternalResouceViewResolver, y se especifica en el web applicationContext de nuestra aplicación de la siguiente manera:

```
<bean id=”viewResolver”
class=”org.springframework.web.servlet.view.InternatlResourcesViewResolver”
>
        <property name=”viewClass”
        <value>org.springframework.web.servlet.view.JstlView</value>
        </property>
        <property     name=”prefix”><value>/WEB-INF/jsp/</value></property>
        <property name=”suffix”><value>.jsp</value></property>
</bean>
```

En este Bean de id viewResolver, se especifica de que clase se quiere que se implemente, para este caso será el tipo de View Resolver que se quiere de los 4 mencionados anteriormente. Después vienen tres propiedades:

**viewClass:** sirve para especificar qué tipo de plantilla se quiere usar para desplegar la vista, en este caso se utilizó JSTL (Java Standard Tag Library), que es una de las plantillas más comunes para este tipo de tarea, también se puede seleccionar Velocity o Tyles para desplegar la información.

**preffix:** el prefijo que antecederá al nombre lógico de la vista, generalmente es la ruta donde se encontraran los JSP, Ejemplo: /WEB-INF/jsp.

**Suffix:** el sufijo que tendrán nuestras vistas, ya que la mayoría de los nombres físicos que se utilizan son JSPs se les asigna la extensión .jsp.

Todo esto es con el afán de tener una programación que no sea tan dependiente, ya que si se quisiera cambiar de carpeta todas las vistas, lo único que se tendría que modificar seria el preffix del viewResolver.

### Controladores

Si el DispatcherServlet es el corazón del Spring MVC los controladores son los cerebros. 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 ayudara a detectar más fácilmente las fallas y errores que puedan surgir.

Existe una variedad de controladores, como se muestra en la Figura II.8,los cuáles poseen una jerarquía. Spring brinda libertad al desarrollador de escoger que tipo de controlador desea implementar, no lo limita como en algunos otros frameworks.

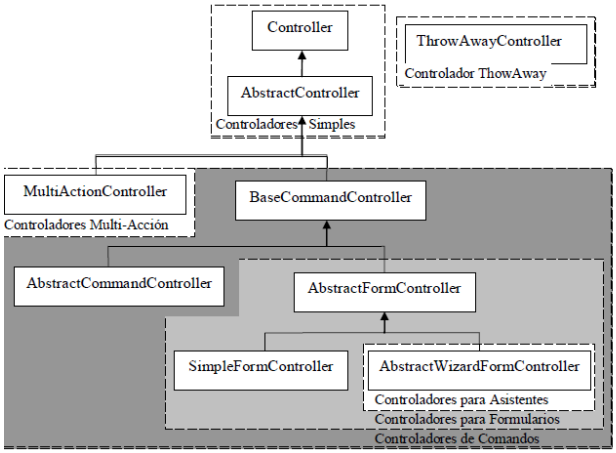



**Figura II. 8Controladores que provee Spring**

**Fuente:** http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/sanchez_r_ma/capitulo3.pdf

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. El principal es el nombre de la vista a la que se va a egresar el modelo para que sea desplegado, y los demás atributos pueden ser parámetros que se le agregan por medio del método .addObject ("nombre_parametro", valor_parametro). Además el Modelo puede estar formado por un solo objeto o por un Map de Objetos, los cuales se identificaran 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.

El siguiente ejemplo muestra el código básico de un controlador.

```
public class MoviesController extends AbstractController {
//Metodo que controla el Request
PublicModelAndViewhandleRequestInternal(
HttpServletRequest request, HttpServletResponse response)
Throws Exception{
        //Se recupera una lista de todas  las películas
        Listmovies = MovieService.getAllMovies ( );
        // Se manda a la vista movieList el objeto movies, con el nombre
lógico //movies
        Return new ModelAndView ("movieList", "movies", movies);
        }
}
```

El objeto ModelAndView contiene el nombre lógico de la vista, el cual el framework se encarga a través del ViewResolver de convertir ese nombre lógico al nombre físico que es el que buscara el servidor web.

**Procesamiento de formularios**

Una de las increíbles facilidades que brinda Spring es la de llenar, recibir y procesar formularios, a través de los 3 diferentes controladores que brinda Spring para el manejo de

formas o formularios. Ya que con estos controladores, se facilita el procesamiento, llenado de la forma, almacenamiento (si es necesario) de los datos, así como desplegar errores y confirmaciones. Para el usuario todo este proceso es totalmente transparente ya que el formulario que este llenando tendrá el mismo formato y los mismos elementos que cualquier otro formulario normal.

Existen varios pasos a seguir para poder utilizar un controlador de formularios, uno de ellos rellenar el archivo .jsp con tags que provee la librería spring, .tdl. Esta librería se debe de combinar de la siguiente manera: En el Deployment Descriptor de la aplicación se pone el siguiente tag:

```
<taglib>
<taglib-uri>/spring</taglib-uri>
<taglib-location>/WEB-INF/spring.tld</taglib-location>
</taglib>
```

Una vez realizada esta acción se configura el applicationContext de la aplicación con el Bean que describirá toda la información que el controlador necesitara.

Para el control de las formas, se recomienda crear un Objeto que tenga atributos, cada uno de ellos con sus respectivos métodos set y get, para que a través del patrón de Inyección de Dependencia se pueda poner la información dentro de ese objeto y sea más fácil de manipular y almacenar. Aeste objeto se le conoce como "objeto comando" o commandObject.

Estas son algunas de las propiedades que el controlador necesita, no todas son obligatorias.

- **formView:** nombre lógico de la vista que contiene la forma
- **succesView:** nombre lógico de la vista que se desplegara una vez que la forma sea llenada con éxito.

- **commandClass:** la clase a la que pertenecerá el Objeto comando u objeto base sobre el cual se va a trabajar.
- **commandName:** nombre lógico que se le quiere dar al objeto comando, con el cual se le reconocerá en la vista.
- **Validator:** este será el Bean de referencia de la clase que extiende a la interfaz org, springframework.validation.Validator que se encarga después de que se hizo la vinculación (binding) de validar que todos los campos cumplan con los requisitos que el desarrollador desee y regresara los errores y las violaciones, y en qué campo fue donde se cometieron, así como el mensaje de error que se debe desplegar.

Este es un ejemplo de un Bean de configuración de SimpleFormController:

```
<bean id=”registerControleer” class=”web.RegisterController”>
<property name=”formView”>
<value>newStudenForm</value>
</property>
<property name=”succesView”>
<value>studentWelcome</value>
</property>
<property name=”commandClass”>
<value>business.User</value>
</property>
<property name=”commandName”>
<value>user</value>
</property>
<property name=”validator”>
<ref bean=”registerValidator”/>
</property>
</bean>
<bean id=”registerValidator” class=”validators.RegisterValidator”/>
```

Una vez realizada esta configuración se debe crear una clase para el controlador, la cual debe extender a la clase SimpleFormController, en la cual se pueden sobrescribir varios métodos como por ejemplo:

- **MapreferenceDat (HttpServletRequestreq, Objectcommand, BindExceptionerrors):** en este método se realiza el llenado de los campos que el desarrollador desee que aparezcan llenos una vez que se cargue el formulario. Estos campos vendrán llenos con los datos que se encuentren en el Map que será regresado. Este se mezclara con el modelo para que la visa los procese y se desplieguen en los campos del formulario necesarios.
- **ModelAndViewonSubmit (HttpServletRequestreq, HttpServletResponseres, Objectcommand, BindExceptionerrors):** Este método puede tener variaciones en cuanto al número de parámetros que recibe, ya que existen formas más sencillas de este método. Estos métodos son llamados después de que ocurrió la validación y en el objeto BindException no hubo ningún error, se ejecuta este método, en el cual se realiza la acción que se desee una vez que el usuario haya enviado el formulario, ya sea almacenar la información en una base de datos, etc.

VoiddoSubmitAction(HttpServletRequestreq): este método es la forma más simple de los métodos para enviar los formularios ya que no se necesita crear un objeto ModelAndView, el método crea uno nuevo con la instancia succesView que se haya dado en la configuración del controlador en el applicationContext. Y se pueden llevar a cabo las mismas operaciones que en cualquier método de envió, por ejemplo almacenar en una base de datos la información del formulario.

Ahora solamente queda utilizar los tags dentro del JSP para poder vincular cada uno d los campos a un atributo del Objeto comando que se haya escogido. Para esto se utiliza el tag<spring: bind> que únicamente tiene el atributo path, que es donde se asigna a que atributo del objeto comando se va a referir ese campo del formulario. Este tag trae dentro también los errores de validación que se llegaran a dar al momento de enviar el formulario.

Esto se da gracias a un objeto status de tipo Bind-Status que a través de 3 parámetros ayudara a vincular:

- **expression:** La expresión usada para recuperar el nombre del campo, si por ejemplo la propiedad es Movie.name, el valor de la expresión y el nombre del campo será name.
- **value:** El valor del campo asociado, una vez que fue enviada la forma se utiliza para volver a poner el valor de este campo si es que hubo errores y que no se pierda toda la información que el usuario proporciono.
- **errorMessages:** Un arreglo de Strings que contiene los errores relacionados con este campo.

Ejemplo del código fuente de un JSP utilizando los tags de vinculación.

```
<spring:biend>
<%@   taglib prefix=”c” uri=http://java.sun.com/jstl/core %>
<%@   taglib prefix=”spring” uri=”/spring”   %>
<form method=”POST” action=”/registerUser,htm”>
<spring:bind path=”user.firstName”>
Nombre:
<input  type=”text”
name=”<c:out value”${status.expression}”/>”
value=”<c:out value=”${status.value}”/>”>
<p>Errores:<c:out value=”${status.errorMessages}”/></p>
</spring:bind>
</form>
```

Una de las combinaciones más importante en cuanto a las vistas JSP es utilizar la tecnología JSTL para poder realizar ciclo, condiciones, entre otros de una manera más rápida, sencilla y eficaz para el desarrollador.

**Servicios**

- Inyección de Dependencias (o Inversión de Control): Un objeto no se encarga de instanciar a sus colaboradores si no que éstos son inyectados por el framework. De esta forma, el objeto no depende de la tecnología con la que estén implementados y además es más fácil sustituirlos por tests.
- Programación Orientada a Aspectos**:** Llamamos aspectos a funcionalidades que son transversales a las aplicaciones (por ejemplo la seguridad o la transaccionalidad). Al trabajar con **AOP** estamos centralizando el código que implementa estas funcionalidades y configurando Spring para que las ejecute donde queramos, en lugar de tener que hacerlo en cada método que haga uso de ellas. Spring se encarga de hacer transparente muchas tareas que hay que realizar al trabajar con servicios utilizados habitualmente en aplicaciones empresariales y que suelen resultar muy repetitivas. Por ejemplo al acceder a bases de datos mediante JDBC, él se encargará de gestionar las conexiones, unificar excepciones, etc.

**Características**

- Simplificación de la programación orientada a aspectos.
- Simplificación del acceso a datos.
- Simplificación e integración con JEE
- Soporte para planificación de trabajos.
- Soporte para envió de mail.
- Interacción con lenguajes dinámicos (como BeanShell, JRuby, y Groovy).
- Soporte para acceso a componentes remotos.

- Manejo de Transacciones.
- Su propio framework MVC.
- Su propio Web Flow.
- Manejo simplificado de excepciones.

La versión 3 de Spring es una versión revisada y mejorada de la versión estable anterior (2.5), que incluye nuevas características, entre las que se incluyen:

- **Soporte para Java 5:** Proporciona configuración basada en anotaciones y soporta características como varargs y generics, además la parte web es compatible con las versiones 1.4 y 5 de Java EE. Debido a esta nueva característica, ahora es necesario tener el JRE versión 5 o superior.
- **Lenguaje de Expresiones (SpEL):** En esta nueva versión se incluye un lenguaje de expresiones que puede ser usando cuando se definen beans, tanto en XML como con anotaciones y también da soporte a través de todos los módulos de Spring.
- **Soporte para Servicios Web REST**: Ahora Spring soporte servicios web de tipo REST.
- **Soporte para Java EE6:** Ofrece soporte de características como JPA 2.0, JSF 2.0 y JRS 303 (validaciones de Beans).
- **Soporte para bases de datos embebidas:** Un soporte conveniente para bases de datos embebidas como HSQL, H2 y Derby.
- **Soporte para formateo de datos mediante anotaciones:** Ahora los campos de fecha, divisas, etc., serán formateados automáticamente y convertidos usando anotaciones.

- **Nueva organización de los módulos:** Los módulos han sido revisados y separados en diferentes paquetes, mas organizados, de acuerdo a su funcionalidad.

**Ventajas**

**Mayor velocidad de Desarrollo**

- Desarrollo más rápido, de forma más organizada y más eficiente de aplicaciones empresariales J2EE.

**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.
- Es un framework maduro con varios años en el mercado.

**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.

**Mejor Integración y Flexibilidad**

- Facilidad de integración con los estándares JEE y herramientas comerciales existente.
- Puede emplearse en cualquier aplicación hecha en Java no solo Web.
- No obliga a usar un modelo de programación particular lo que conlleva a una mejor integración con otras herramientas.

**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.

**Desventajas**

- La configuración de Spring está inflada, es decir, para cada servicio que se tenga hemos de 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.
- No se puede evaluar si un objeto ha sido bien inyectado más que en tiempo de ejecución. Aunque hay herramientas como Spring IDE que sí que ayudan.
- 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.

# CAPÍTULO III

# 3. PATRÓN DE DISEÑO MVC

## 3.1 Introducción

## 3.2 Patrón de Diseño Modelo MVC

### 3.2.1. Definición

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC 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 modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista.

### 3.2.2. Elementos del Patrón MVC

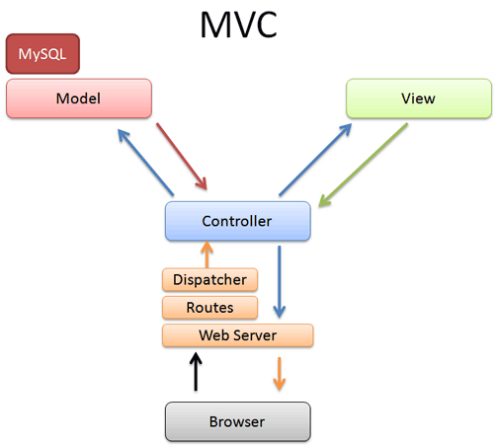



**Figura III. 9Elementos del Patrón MVC**

**Fuente:** http://drupaltroll.wordpress.com/2012/03/16/php-framworks-why-when-and-which/

**Modelo**: Es la representación de la información en el sistema. Trabaja junto a la vista para mostrar la información al usuario y es accedida por el controlador para añadir, eliminar, consultar o actualizar datos.

Realiza las siguientes funciones:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: “Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor”.
- Lleva un registro de las vistas y controladores del sistema.

- Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero bath que actualiza los datos, un temporizador que desencadena una inserción, etc.).

**Vista**: Es la representación del modelo en un formato adecuado para que el usuario pueda interactuar con él, casi siempre es la interfaz de usuario.

**Funciones:**

- Recibir datos del modelo y los muestran al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de "Actualización ()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

**Controlador**:Es el elemento más abstracto. Recibe, trata y responde los eventos enviados por el usuario o por la propia aplicación. Interactúa tanto con el modelo como con la vista.

### 3.2.3. Ciclo de vida de MVC

El ciclo de vida de MVCes normalmente representado por las 3 capas presentadas anteriormente y el cliente (también conocido como usuario). La siguiente figura representa el ciclo de vida de manera sencilla:

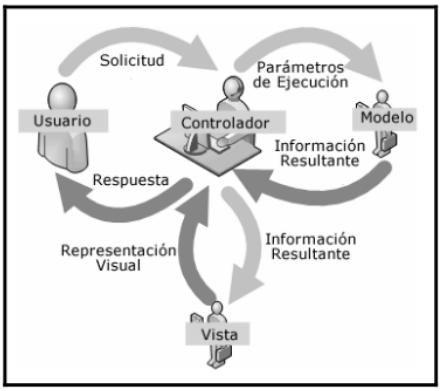



**Figura III. 10Ciclo de vida MVC**

**Fuente:**http://calidadysoftware.blogspot.com/2011/10/aspnet-mvc-ii-ventajas.html

El primer paso en el ciclo de vida empieza cuando el usuario hace una solicitud al controlador con información sobre lo que el usuario desea realizar. Entonces el Controlador decide a quién debe delegar la tarea y es aquí donde el Modelo empieza su trabajo. En esta etapa, el Modelo se encarga de realizar operaciones sobre la información que maneja para cumplir con lo que le solicita el Controlador. Una vez que termina su labor, le regresa al Controlador la información resultante de sus operaciones, el cual a su vez redirige a la P.13

Vista. La Vista se encarga de transformar los datos en información visualmente entendible para el usuario. Finalmente, la representación gráfica es transmitida de regreso al Controlador y éste se encarga de transmitírsela al usuario. El ciclo entero puede empezar nuevamente si el usuario así lo requiere.

### 3.2.4. Ventajas

- Una separación total entre lógica de negocio y presentación. A esto se le pueden aplicar opciones como el multilenguaje, distintos diseños de presentación, etc. sin alterar la lógica de negocio. La separación de capas como presentación, lógica de

negocio, acceso a datos es fundamental para el desarrollo de arquitecturas consistentes, reutilizables y más fácilmente mantenibles, lo que al final resulta en un ahorro de tiempo en desarrollo en posteriores proyectos.

- Crea independencia de funcionamiento.
- Es posible tener diferentes vistas para un mismo modelo (ej. representación de un conjunto de datos como una tabla o como un diagrama de barras.
- Modificar los objetos de negocios bien sea para mejorar el performance o para migrar a otra tecnología.
- Facilita el mantenimiento en caso de errores.
- Ofrece maneras sencillas para probar el correcto funcionamiento del sistema.
- Permite el escalamiento de la aplicación en caso de ser requerido.

### 3.2.5. Desventajas

- La separación de conceptos en capas agrega complejidad al sistema
- La cantidad de archivos a mantener y desarrollar se incrementa considerablemente
- La curva de aprendizaje del patrón de diseño es más alta que usando otros modelos más sencillos.

### 3.2.6. Frameworks MVC

#### Framework Web

Un Framework Web, se define como una estructura definida, reusable en el que sus componentes facilitan la creación de aplicaciones web.

Estos frameworks nos proveen un conjunto de componentes como por ejemplo: clases en java, descriptores y archivos de configuración en XML; que componen un diseño reutilizable que facilita y agiliza el desarrollo de Sistemas Web.

**Objetivos de un Framework Web Java EE**

- Los objetivos principales de un Framework son:
- Facilitar el desarrollo de aplicaciones empresariales.
- Acelerar el proceso de desarrollo.
- Reutilización de código.
- Promover buenas prácticas de desarrollo, como el uso de patrones.

**Tipos de Frameworks Web**

Según cuál sea el tipo de aplicación web en la que se especializa un Framework tenemos los siguientes tipos:

- **Frameworks para publicación web.-** centrados en el manejo y presentación de contenidos, como: JPublish y Cocoon.
- **Frameworks para la explotación de bases de datos.-** centrados en la gestión de
- datos almacenados en bases de datos relacionales.
- **Frameworks de presentación.**- centrados en el desarrollo de aplicaciones con interfaces ricas y altamente interactivas.
- Según la arquitectura del Framework se clasifican en:
- **Frameworks basados en transformaciones de documentos XML** la mayoría de este tipo de Frameworks están especializados en la publicación web. Su funcionamiento se basa en la transformación de documentos fuente, mezclados con

datos dinámicamente que se transforman para dar lugar a un documento final que puede estar en diversos formatos, principalmente HTML y PDF. El ejemplo más característico de este tipo de Frameworks es Cocoon.

- **Frameworks basados en el encadenamiento de acciones** de recepción de información de un formulario, selección de un objeto al que pasarle esa información para que la procese y selección de la salida tras realizar el procesamiento. Todo ello basado en la configuración dada en un fichero. Esta es la arquitectura de Framework más común. Ejemplos de él son: Struts, Maverick, entre otros.
- **Frameworks basados en componentes** estos Frameworks suelen basar su funcionamiento en la definición de componentes reutilizables. Esta arquitectura es muy usada en los Frameworks de presentación, ya que el uso de componentes independientes favorece la interactividad de la aplicación. Ejemplos de estos Frameworks son Tapestry, Java Server Faces.

**Características**

Existen una serie de características que son comunes a todos los Frameworks como las siguientes:

- **Abstracción de URLs y Sesiones.-** no es necesario manipular directamente las URLs ni las sesiones, ya que el Framework se encarga de hacerlo por nosotros.
- **Acceso a Datos.-** incluyen herramientas o interfaces necesarias para integrarse con herramientas de acceso a datos.
- **Controladores.-** la mayoría de Frameworks implementa controladores para gestionar eventos, como una introducción de datos mediante un formulario o el

acceso a una página. Estos controladores se adaptan fácilmente a las necesidades de un proyecto en concreto.

- **Autentificación y Control de Acceso.-** incluyen mecanismos para la identificación de usuarios mediante login y password; y además permiten restringir el acceso a determinadas páginas a determinados usuarios; por ejemplo: un usuario tipo Administrador tendrá acceso a páginas a las que otro tipo de usuario no podrá acceder.
- **Internacionalización.-** permite mostrar la aplicación según el idioma del usuario.
- **Separación entre presentación y comportamiento de la aplicación.**

**Ventajas de usar Frameworks**

El uso de frameworks en el desarrollo de aplicaciones web nos proporciona ventajas como:

- **Modularidad y reducción de la complejidad.-** la aplicación está formada por subsistemas especializados en distintos aspectos fundamentales de toda aplicación como: presentación, persistencia.
- **Fortaleza al cambio.-** los módulos pueden evolucionar o cambiar conservando la arquitectura global de la aplicación.
- **Documentación.-** la documentación del Framework promueve el uso correcto del mismo y disminuye el esfuerzo necesario para el mantenimiento.
- **Estructura.-** el desarrollo basado en Frameworks establece una estructura sobre la cual las aplicaciones pueden ser construidas, liberando al desarrollador de tomar el 100% de las decisiones de diseño.

- **Distribución de funciones.-** permite realizar en paralelo el trabajo de desarrollo, ya que la aplicación puede desarrollarse como un conjunto de piezas independientes que encajarán en el Framework usado.
- **Eficiencia.-** el desarrollador puede concentrarse en los requerimientos funcionales de la aplicación.
- **Aplicaciones ricas.-** posibilidad de dar más funcionalidad a los usuarios de la aplicación.
- **Mejoran notablemente la calidad del software resultante.-** Ya que el desarrollador puede dedicarse a resolver problemas específicos de la lógica del negocio, y no preocuparse por los detalles de programación de bajo nivel.

**Frameworks web que implementan MVC.**

Aunque ya se ha comentado que en la web no se puede podría hablar estrictamente de MVC, existen en el mercado varias implementaciones con un uso muy amplio y extendido como son los siguientes.

- Struts.
- Struts2.
- Spring.
- JSF

Los MVC cumplen perfectamente el fin particular de cualquier frameworks, (una estructura bien definida que da soporte a un proyecto web también nos ayuda a que nuestro proyecto sea organizado y bien desarrollado).

**Diagrama de Flujo de un framework MVC**

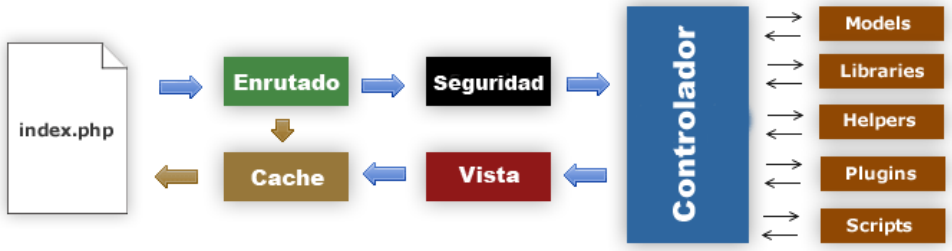



**Figura III. 11Diagrama de flujo de un framework MVC**

**Fuente:**http://www.neleste.com/modelo-vista-controlador/

- El index.php nos inicializa el núcleo de nuestra aplicación. El enrutador nos examina nuestra petición HTTP y nos ayuda a determinar que se debe de hacer.
- Si existe, la cache nos devuelve nuestro archivo HTML sin necesidad de pasar por el sistema, ahorrando la carga que esto nos conlleva.
- La Seguridad, ya que antes de que se cargue el controlador se filtran los datos enviados para que estos puedan resultar fiables.
- El controlador nos carga el modelo, librerías, helpers, plugins y todos los demás recursos necesarios para satisfacer nuestra petición.
- Finalmente, cuando la Vista está renderizada, esta es enviada al navegador, entonces si la cache se encuentra habilitada, se almacena el resultado para la próxima ocasión que la URL sea servida.

# CAPÍTULO IV

# 4. ANÁLISIS COMPARATIVO

## 4.1.Introducción

Desde siempre, desarrollar aplicaciones empresariales no ha sido una tarea fácil considerando que los avances de la tecnología han ido simplificando el manejo de varios aspectos de este problema. Sin embargo, los desarrolladores de este tipo de aplicaciones continúan enfrentándose a desafíos tales como datos complejos; la mayor parte del tiempo los requerimientos no son explícitos, usuarios simultáneos múltiples, los requerimientos del negocio cambian con frecuencia, plataformas heterogéneas, e interdependencias complejas entre aplicaciones distribuidas.

Además del hecho que no exista ningún estándar de la industria para el desarrollo de este tipo de aplicaciones ha llevado a muchas organizaciones a desarrollar sus propios modelos generando así un alto costo de desarrollo y mantenimiento de las aplicaciones.

El presente capítulo tiene como objetivo el estudio de los frameworks java MVC más destacados en la actualidad y tener una base que permita la selección de la tecnología más idónea, para el desarrollo de aplicaciones web.

## 4.2 Determinación de las Herramientas a Comparar

En la actualidad podemos encontrar una diversidad de frameworks para el desarrollo de Aplicaciones Web empresariales en Java, cada uno de los cuales nos brindan ciertas ventajas. Entre los principales frameworks MVC J2EE tenemos a Apache Struts, Spring, JSF, para esta investigación se va a comparar los frameworks Struts y JSF debido a queson los más utilizados y en el mercado existen varias implementaciones con un uso muy amplio y extendido de estos frameworks.

A continuación presentamos una descripción rápida de los mismos y además utilizaremos Hibernate que es una herramienta ORM completa que nos permite realizar el mapeo de nuestra base de datos a clases java que serán utilizadas en una aplicación, ahorrando tiempo de desarrollo.

## 4.3. Análisis de las Tecnologías

### 4.3.1. STRUTS

#### Introducción

Originalmente fue creado por Craig R. McClanahan y donado al proyecto Jakarta de Apache Software Foundation en el año 2000.

En junio de 2001 se libera la primera versión, y desde entonces muchos desarrolladores han participado en la mejora continua de este framework

Simplifica notablemente a implementación de una implementación siguiendo la arquitectura MVC. El controlador ya se encuentra implementado en Struts, sin embargo y si es necesario es posible extenderlo o modificarlo. El flujo de la aplicación se controla desde un archivo XML

Proporciona la integración con el modelo, la lógica de negocio se implementa basándose en clases predefinida por el framework. Facilita y soporta la construcción de la interfaz de la aplicación utilizando un conjunto de tags predefinidos. Se busca es evitar el uso de scriptlets, que son trozos de código Java entre "<%"y "%>"; para ganar mantenibilidad y performance. Permite el desarrollo de sus componentes en paralelo o por personal especializado

Los Servlets de Java han demostrado su superioridad frente a los CGIs, debido a su portabilidad y extensibilidad. Un Servlets no es más que una clase Java usada para extender las capacidades de los servidores que albergan aplicaciones accedidas mediante un modelo de programación basado en peticiones/respuestas. El problema de los Servlets es que escribir las respuestas mediante "println ()" y en formato HTML resulta tedioso. No obstante, la solución a este problema aparece con los JSPs, lo que permite trasladar la respuesta al usuario y embeberla dentro de una página HTML. Hoy en día los desarrolladores pueden mezclar fácilmente HTML y código Java, además de poder utilizar los Servlets a discreción.

**Definición**

Es un framework de código abierto para desarrollar aplicaciones Web J2EE usando el patrón de diseño Modelo-Vista-Controlador. Usa y extiende el Java Servlet API para motivar a los desarrolladores a adoptar la arquitectura MVC

Struts proporciona un marco unificado para la implementación servlet JSP y aplicaciones que utilizan la arquitectura MVC. Ofrece clases de utilidades para manejar muchos de las tareas más comunes en el desarrollo de aplicaciones Web.

Struts proporciona bibliotecas de etiquetas personalizadas para la salida de beans propiedades, la generación de formularios HTML, iterar sobre varios tipos diferentes de estructuras de datos y condicionalmente tipos de estructuras de datos, y condicionalmente la salida de HTML.

Struts es un framework de código abierto que se extiende de la API Java Servlet y utiliza un Modelo, Vista, Controlador (MVC). Le permite crear aplicaciones web flexibles basadas en tecnologías estándar, como las páginas JSP, JavaBeans, paquetes de recursos, y XML

Apache Struts es un país libre de código abierto marco para la creación de aplicaciones Web en Java.Las aplicaciones Web difieren de las páginas web convencionales en que las aplicaciones web pueden crear una respuesta dinámica. Muchos sitios web ofrecen sólo las páginas estáticas. Una aplicación web puede interactuar con bases de datos y motores de lógica de negocios para personalizar una respuesta.

**Objetivos**

Struts es un proyecto de la "Apache Software Foundation" que tiene como objetivo proporcionar un entorno específico para la construcción de grandes aplicaciones web. Es una variación del paradigma de diseño clásico conocido como "Model View Controller" (MVC), donde se separan claramente los roles de los diseñadores web, los programadores java y los administradores de bases de datos.

**Versiones**

Struts ha tenido varias versiones a través de su evolución como

- Struts 1.0.2
- Struts 1.1.1
- Struts 1.2.x
- Struts 1.3.x
- Así como la nueva versión Struts 2

Struts no es obsoleta hay una comunidad fuerte y vibrante de desarrolladores que utilizan Struts 1 en producción y se espera que miles de equipos se sigan basando en los nuevos proyectos en Struts 1, y continuar apoyando los proyectos existentes, para muchos, muchos años por venir.

Extensiones nuevas y mejoradas para Struts 1 siguen apareciendo regularmente. En 2006 hubo versiones de Hoople, Strecks, etiquetas JSP de control, Sprout, Spring Web Flow, DWR, Calyxo, FormDef, y Java Web Parts.

Desde la fusión, Struts 1 ha llegado a lanzar una nueva versión menor, Struts 1.3, y las nuevas versiones 1.x se están planeando. Struts 1 sigue siendo el marco de aplicaciones web más populares y mejor apoyo para Java.

Struts 1 es reconocido como el framework más popular para aplicaciones Web en Java. Este framework, en su versión 1.x, es maduro, está bien documentado, y cuenta con un amplio soporte. Es la mejor alternativa para afrontar problemas comunes de desarrollo.

Struts 2 fue originalmente conocido como WebWork 2; después de trabajar independientemente por varios años, las comunidades de Struts y WebWork unieron fuerzas para crear Struts 2. El framework 2.x es la mejor opción para afrontar elegantemente los problemas difíciles del desarrollo.Struts 2 es un elegante y extensible framework para la creación de la empresa de aplicaciones web en Java. El framework está diseñado para apoyar el ciclo completo de desarrollo, desde la construcción, el despliegue y a lo largo del periodo de mantenimiento de las aplicaciones.

**Struts 1 vs. Struts 2**

Struts2 es el framework más potente en comparación con Struts1. La tabla a continuación se describe algunas diferencias entre Struts1 y struts2.

**Tabla IV. IDiferencias entre Struts 1 y Struts 2**

| Característica | Struts 1 | Struts 2 |
|---|---|---|
| Clases de acción | Extiende la clase base abstracta por su acción de clase. El | Implementa una interfaz de acción, junto con otras interfaces de uso de los |

| | | |
|---|---|---|
| | problema es que utiliza las clases abstractas en vez de interfaces. | servicios opcionales y los servicios comunes |
| ThreadingModel o Modelo Enhebrado | Las acciones son únicas por lo tanto deben ser thread-safe, ya que sólo una instancia de una clase se encarga de todas las solicitudes de esa acción. | Struts 2 no tiene problemas de seguridad para hilos como objetos de acción ya que crea una instancia para cada solicitud.<br>. |
| La dependencia Servlet | Las acciones dependen de la API de servletHttpServletRequest y HttpServletResponse, porque se pasa al método de ejecución, cuando se invoca una acción. | Los Servlet se suelen representar como mapas simples que permiten que las acciones que se analizarán en forma aislada. Se puede acceder a la solicitud original y a la respuesta. |
| Capacidad de prueba | Tiene un gran problema durante las pruebas de la aplicación por culpa del método de ejecución de la API de Servlets. StrutsTestCase proporciona un conjunto de objetos simulados. | Es necesario ejecutar la acción, establecer las propiedades, y la invocación de métodos. Las dependencias también hace son más fáciles de probar. |
| La expresión del lenguaje | Une a los objetos en el contexto de la página utilizando el | Utiliza una tecnología ValueStack para hacer los valores accesibles a los |

| | | |
|---|---|---|
| | mecanismo estándar de JSP. | taglibs, permite reutilizar vistas a través de una amplia gama de tipos. |
| Conversión de tipos | Commons-Beanutils son utilizados por Struts 1 para la conversión de tipos. | Utiliza OGNL para la conversión de tipo y convertidores para convertir los tipos de objetos básicos y comunes y de los primitivos también. |
| Validación | Utiliza una validación manual a través de un método para validar en el ActionForm. | Permite la validación manual que se lleva a cabo mediante el método de validación y el marco de trabajo XWork. |
| Control de la ejecución de Acción | Cada módulo tiene una solicitud por separado, mientras que todas las acciones en el módulo deben compartir el mismo ciclo de vida. | En Struts 2 ciclos de vida diferentes se crean en una base de acción. Pilas personalizadas se crean y se utiliza con diferentes acciones. |

**Fuente**: http://og-java.blogspot.com/2009/05/struts-1-vs-struts-2x.html

## Arquitectura

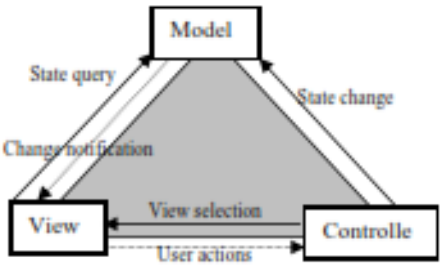



**Figura IV. 12MVC**

**Fuente:**http://xperienced.com.pl/blog/how-to-implement-mvc-pattern-in-cocos2d-game

En el modelo MVC, el flujo de la aplicación se controla mediante un agente central. El controlador reenvía las peticiones (HTTP-requests) hacia un manejador apropiado. El "Modelo" representa la lógica de negocio de la aplicación web. El "Control" pasa del manejador a la vista apropiada a través del controlador y finalmente se presenta el resultado de la transacción al usuario. El reenvío del controlador a la vista apropiada puede determinarse estáticamente mediante una serie de asociaciones a partir de un archivo de configuración. Este paso intermedio permite desacoplar la "vista" del "modelo", muy en línea con los conceptos de facilidad de manejo y extensibilidad de la aplicación que se persigue con el modelado bajo el entorno de Struts.

El "modelo" dentro del sistema MVC se subdivide en dos subsistemas, la parte que representa el estado interno del sistema y las acciones que pueden realizarse para modificar dicho estado.

La mayoría de las aplicaciones representan el estado interno del sistema como un conjunto de JavaBeans.

Las propiedades del Bean representan el estado del sistema y los métodos representan las operaciones relacionadas con la lógica de negocio. Por ejemplo, una cesta de la compra que almacena los artículos seleccionados por el usuario hasta la fecha, puede representarse mediante un JavaBean.

La "vista" del modelo MVC se construye utilizando JSPs. Los JSPs pueden contener texto estático (HTML) y además pueden presentar contenido dinámico basado en la interpretación en tiempo de solicitud de la página de determinadas etiquetas especiales. El entorno de JSPs también proporciona un conjunto de acciones estándar que suelen ser muy comunes en la programación con JSPs y que pueden reutilizarse en distintos proyectos.

El entorno Struts proporciona un conjunto de etiquetas personalizables ("customtags") para facilitar la creación de interfaces de usuario. Además, dichas etiquetas interaccionan con los "ActionForm" Beans, que son las herramientas que dentro del modelo Struts capturan y validan cualquier entrada proporcionada por el usuario.

El "controlador" del entorno de desarrollo Struts se centra en la recepción de solicitudes por parte del cliente. Este cliente suele ser un usuario ejecutando un navegador web. El controlador decide qué lógica de la aplicación debe ser ejecutada en función de la petición del cliente, y posteriormente delega la responsabilidad de producir la siguiente fase del interfaz de usuario a una determinada "vista". Una "ActionMapping" no es más que una ruta que se asocia a una determinada petición HTML y que típicamente especifica el nombre completo de una clase de tipo "Action". Todas las acciones son subclases de la clase "Action". Las acciones encapsulan llamadas a las clases que representan la lógica de

negocio, interpretan los resultados y eventualmente redireccionan el control al componente de la "vista" apropiado para crear una respuesta para el usuario.

En los siguientes apartados se describe paso a paso los elementos que Struts utiliza para implementar el modelo MVC/Model-2

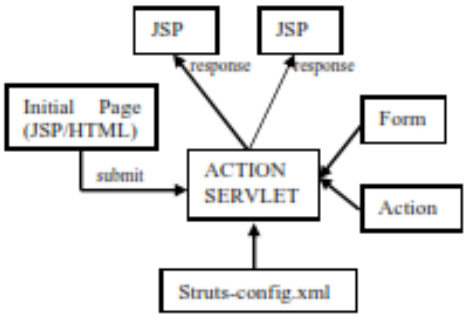



**Figura IV. 13Struts a vista de pájaro**

**Fuente:** http://www.ibm.com/developerworks/ibm/library/j-struts/

## Arquitectura funcional básica

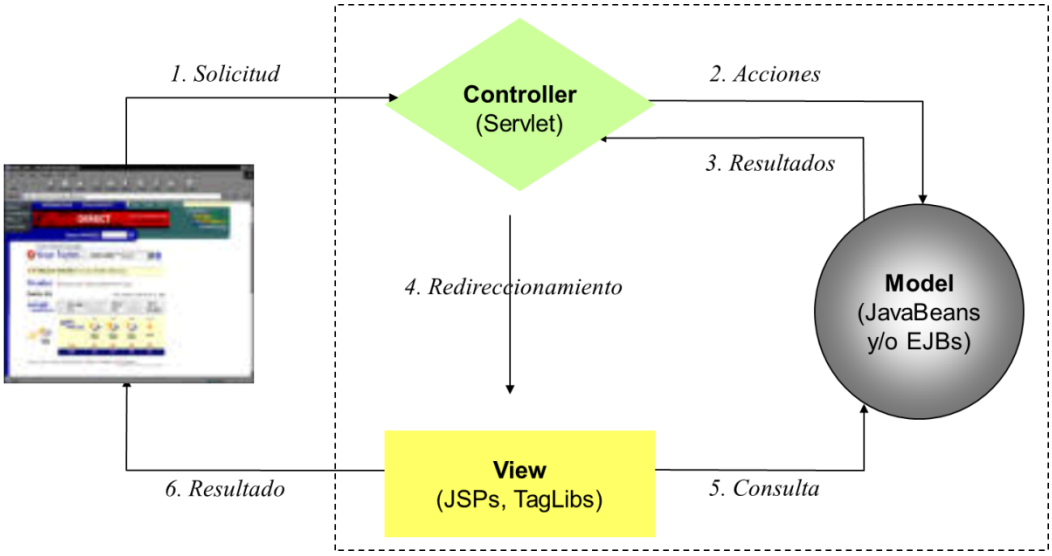



**Figura IV. 14Arquitectura funcional básica**

**Fuente:** http://curso_sin2.blogia.com/2005/julio.php

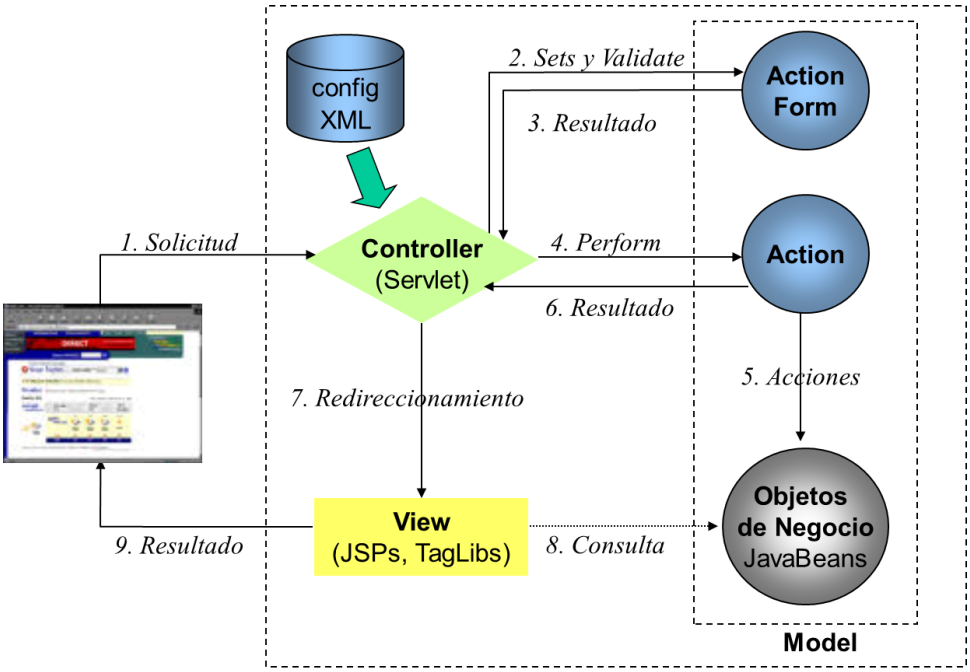



**Figura IV. 15Arquitectura funcional básica 1**

**Fuente:** http://curso_sin2.blogia.com/2005/julio.php

## Componentes

El marco proporciona tres componentes clave:

Una "solicitud" manejador proporcionado por el desarrollador de la aplicación que se asigna a una URI estándar.

Una "respuesta" handler que transfiere el control a otro recurso que completa la respuesta.

Una biblioteca de etiquetas que ayuda a los desarrolladores a crear interactivas basadas en formularios con las páginas del servidor.

La arquitectura del marco de trabajo y las etiquetas son compatibles palabra de moda. Struts trabaja bien con aplicaciones REST convencionales y con las tecnologías como SOAP y AJAX.

**Servicios**

Apache Struts es un Framework Open-Source que sirve para la creación de aplicaciones Web.

**Características**

Su carácter de "*software libre*" y su compatibilidad con todas las plataformas en que Java Entreprise esté disponible, lo convierte en una herramienta altamente disponible.

Una arquitectura general basada en los principios de diseño de Modelo-Vista-Controlador (MVC) en el que todas las peticiones son procesadas por el controlador que realiza todo el control y despacha las peticiones a los componentes de la aplicación apropiados, basándose en los identificadores lógicos que reducen el acoplamiento entre capas.

Capacidades de manejo de formularios, como el JavaBean ActionForm que representa el estado del lado del servidor de los campos de entrada de un formulario, y un marco de trabajo de validación que externaliza la configuración de un conjunto de chequeos de exactitud que se aplican a los valores de los campos de entrada, además implementa estos chequeos tanto en el lado del cliente como en el lado del servidor

No se puede deshabilitar los mensajes, ya que están en código duro.

Un conjunto de etiquetas JSP personalizadas que simplifican el proceso de crear las etiquetas HTML de la aplicación para la capa “Vista”, y que trabaja conjuntamente con las capacidades de manejo de formularios y la arquitectura general del controlador.

Es un framework de la capa de presentación que implementa el patrón MVC en Java:

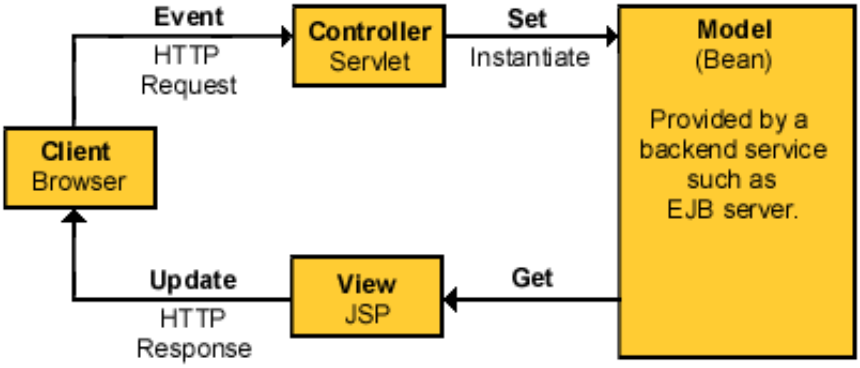



**Figura IV. 16Patrón MVC de STRUTS**

**Fuente:** http://rijkswatch.blogspot.com/2010/09/jsp-model-2-web-mvc.html

**Ventajas y Desventajas**

- Página de navegación gestión
- Validación de entrada de usuario
- Diseño coherente
- Extensibilidad
- Al utilizarStrutsse debe atener a las reglas no escritas:

  No hay código Java en las páginas JSP, toda la lógica de negocios debe residir en clases Java llamadas clases de acción.

Utilizar el lenguaje de expresión (si está usando JSP 2.0) o JSTL en JSP para acceder modelo de negocio. Poca o ninguna escritura de etiquetas personalizadas (debido a que son difíciles de código).

**Ventajas de Struts MVC (vs. Utilizando UtilizandoRequestDispatcher) RequestDispatcher)**

- Centralizada basada en archivos de configuración

Este enfoque también permite a Java Web y la edición de un solo archivo, también permite a Java y Web los desarrolladores se concentren en sus tareas específicas (aplicación de negocio lógica, presentando ciertos valores a los clientes, etc.) sin necesidad sabe acerca de la disposición general del sistema.

- Formbeans

En JSP, puede utilizar la propiedad = "*" con jsp: setProperty para rellenar automáticamente un componente JavaBean basado en entrante solicitar parámetros de la petición. Apache Struts extiende esta capacidad a Java código y agrega en varias utilidades de interés, todos los cuales sirven para mucho simplificar el procesamiento de parámetros de la petición.

- Beantags

Apache Struts proporciona un conjunto de etiquetas JSP (bean: write, en particular) que le permiten fácilmente la salida de las propiedades de JavaBeans componentes. Básicamente, se trata de variaciones breves y poderosos del estándar jsp: useBean y jsp: getProperty etiquetas.

**Ventajas de Struts (vs. Estándar MVC)**

- Etiquetas HTML

Apache Struts proporciona un conjunto de etiquetas JSP personalizadas para crear HTML formas que están asociados a componentes JavaBeans. Esta bean / form asociación tiene dos propósitos útiles:

Le permite obtener el formulario inicial le permite obtener iniciales de campo forma-valores de los campos de objetos Java valores de objetos Java.

Le permite volver a mostrar formularios con algunas o todas introducidas previamente valores intactos.

- Formulario de validación de campo de validación de campo

La capacidad de comprobar que los valores del formulario están en el formato requerido. Si faltan valores o están en un inadecuado formato, automáticamente vuelve a mostrar con error mensajes y con los valores introducidos anteriormente mantenidas.

- El enfoque consistente

Struts fomenta el uso consistente de MVC a través de su aplicación

**Desventajas de Struts (vs. MVC con RequestDispatcher) con RequestDispatcher)**

- Aprendizaje Bigger curva

Para utilizar MVC con el RequestDispatcher estándar, necesita para sentirse cómodo con el estándar JSP y servletAPIs. Para utilizar MVC con Struts, tiene que ser cómodo con el estándar JSP y servletAPIs y con el estándar de JSP y servletAPIs y un marco grande yelaborado que es casi igual en tamaño al sistema central. Este inconveniente es

especialmente con proyectos más pequeños, los plazos a corto plazo, y desarrolladores con menos experiencia, porque puede gastar tanto tiempo de aprendizaje Struts como la construcción de su sistema actual.

- Peor documentación

En comparación con el estándar API de servlet y JSP, Struts tiene menos recursos en línea y la documentación en línea de Apache esconfusa y está mal organizada. También hay un menor número de libros sobre Apache Struts que en Servlets y JSP estándar.

**Desventajas de Struts (vs. Estándar MVC),**

- Menos transparente

Con las aplicaciones Struts, hay mucho más en juego detrás de las escenas que con normalidad basada en Java Web aplicaciones.

- Las aplicaciones Struts son más difícil de entender
- Más difícil de comparar y optimizar

**Funciones**

Struts implementa el patrón del Modelo Vista Controlador (MVC) el cual se utiliza ampliamente y es considerado de gran solidez. De acuerdo con este modelo, el procesamiento se separa en tres secciones diferenciadas, llamadas el modelo, las vistas y el controlador

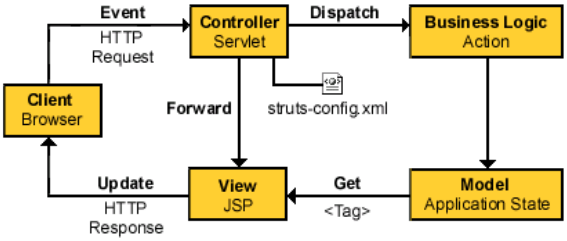



**Figura IV. 17Procesamiento en tres capas**

**Fuente:** http://www.programacion.com/articulo/manual_basico_de_struts_156/4

Cuando se programan aplicaciones Web con el patrón MVC, siempre surge la duda de usar un solo controlador o usar varios controladores, pues si se considera mejor usar un solo controlador para tener toda la lógica en un mismo lugar, surge un grave problema: dicho controlador se convierte en lo que se conoce como "fatcontroller", es decir un controlador saturado de peticiones.

Struts surge como la solución a este problema, pues implementa un único controlador (ActionServlet) que evalúa las peticiones del usuario mediante un archivo configurable (struts-config.xml).

Como trabaja

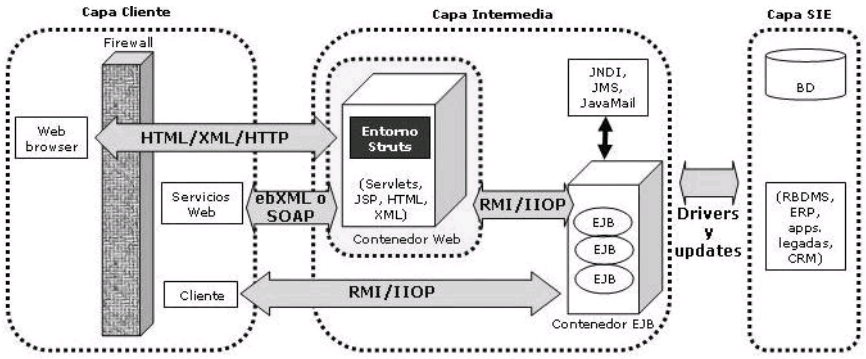



**Figura IV. 18Trabajo de struts-config.xml**

**Fuente:**http://techylib.com/en/view/nutmegact/java2_08_introduccion_a_struts

El navegador genera una solicitud que es atendida por el controlador (un Servlet especializado)[1].

El controlador también se encarga de analizar la solicitud, seguir la configuración que se le ha programado en su XML y llamar al Action correspondiente pasándole los parámetros enviados

El Action instanciará y/o utilizará los objetos de negocio para concretar la tarea. Según el resultado que retorne el Action, el Controlador derivará la generación de interfaz a una o más JSPs, las cuales podrán consultar los objetos del Modelo pata mostrar información de los mismos

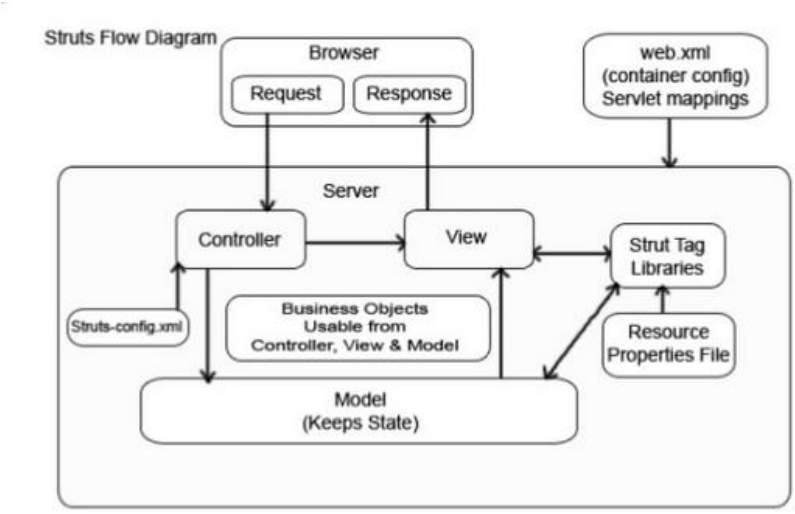



**Figura IV. 19Funcionamiento de Action**

**Fuente:** http://techylib.com/en/view/nutmegact/java2_08_introduccion_a_struts

### 4.3.2. JSF

[1] http://www.slideshare.net/siis/struts-en-java

## Introducción

Para el desarrollo de aplicaciones de negocio se utiliza frecuentemente el patrón de diseño MVC Modelo Vista Controlador (Model View Controller) que además es sencillo de implementar en las aplicaciones web. En este patrón el modelo es modificable por las funciones de negocio. Estas funciones son solicitadas por el usuario mediante el uso de un conjunto de vistas de la aplicación que solicitan dichas funciones de negocio a través de un controlador, que es el módulo que recibe las peticiones de las vistas y las procesa. Se suele clasificar en dos tipos a las aplicaciones basadas en MVC:

Tipo 1. Las vistas conocen la acción que se va a invocar en su petición. Normalmente la función esta cableada dentro de la vista

Tipo 2. El controlador introduce un conjunto de reglas que mapean a las peticiones con las funciones, controlando además el flujo de navegación por la aplicación.

Un ejemplo de aplicaciones de tipo 1 son las que se construyen utilizando JSF o ASP.NET y como ejemplo de tipo 2 serían las creadas con Struts.

La creación de aplicaciones basadas en el patrón MVC se ve facilitada por el uso de marcos de trabajo (frameworks). Un marco de trabajo es un conjunto de APIs y módulos normalmente acompañados de la documentación y guía de uso que definen la manera de implementar alguna de las capas de nuestra aplicación.

**Definición**

La tecnología Java Server Faces (JSF) es un marco de trabajo de interfaces de usuario del lado de servidor para aplicaciones Web basadas en tecnología Java[2].

JSF pretende normalizar y estandarizar el desarrollo de aplicaciones web

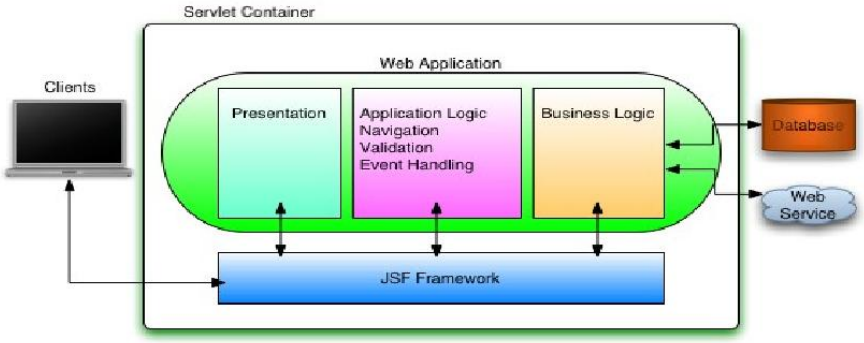



**Figura IV. 20Componentes de JSF**

**Fuente:** http://appfuse.org/display/APF/Using+JSF

Los dos componentes principales son:

- Una API para representar componentes y manejar su estado, manejar eventos, validadores, conversores, internacionalización, etc.
- Dos taglibs JSP para expresar los componentes y enlazarlos con objetos del lado servidor.

  <%@ taglibprefix="f" uri="http://java.sun.com/jsf/core" %>

  <%@ taglib prefix="h" uri="http://java.sun.com/jsf/html" %>

[2] http://www.lintips.com/files/Taller_JSF_1aSesion_Paulo_Clavijo-2010.pdf

Permite a los desarrolladores pensar en términos de componentes, eventos, backingbeans y otras interacciones, en vez de hablar de peticiones, respuestas y marcas.

JSF promete reutilización, separación de roles, facilidad de uso de las herramientas.

JSF tiene una meta específica: hacer el desarrollo web más rápido y fácil.

El framework JSF implementa el patrón de diseño MVC (Modelo Vista Controlador). Permitiendo una separación clara entre el código de interfaz y el de lógica de negocio.

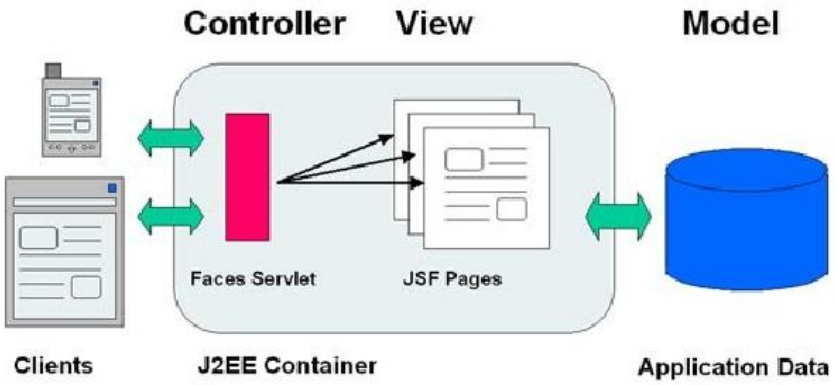



**Figura IV.21MVC de JSF**

**Fuente:** http://www.phpchina.com/download/handbook/openbiz_manual/manual.html

Un modelo de trabajo basado en componentes UI (user interface), definidos por medio de etiquetas y XML.Asocia (de forma modular) cada componente gráfico con los datos (beans de respaldo).

JSF es muy flexible, permite crear nuestros propios componentes.Aplicaciones Web basadas en JavaServerPages a veces se mezclan código base de datos, código de diseño de página y el código de control de flujo.

**Objetivos**

Definir un conjunto simple de clases base de Java para componentes de la interfaz de usuario, estado de los componentes y eventos de entrada. Estas clases tratarán los aspectos del ciclo de vida de la interfaz de usuario, controlando el estado de un componente durante el ciclo de vida de su página.

Proporcionar un conjunto de componentes para la interfaz de usuario, incluyendo los elementos estándares de HTML para representar un formulario. Estos componentes se obtendrán de un conjunto básico de clases base que se pueden utilizar para definir componentes nuevos.

Proporcionar un modelo de JavaBeans para enviar eventos desde los controles de la interfaz de usuario del lado del cliente a la aplicación del servidor.

Especificar un modelo para la internacionalización y localización de la interfaz de usuario.

Automatizar la generación de salidas apropiadas para el objetivo del cliente, teniendo en cuenta todos los datos de configuración disponibles del cliente, como versión del navegador.

**Versiones**

- JSF 1.0 (11-03-2004) - lanzamiento inicial de las especificaciones de JSF.
- JSF 1.1 (27-05-2004) - lanzamiento que solucionaba errores. Sin cambios en las especificaciones ni en el renderkit de HTML.
- JSF 1.2 (11-05-2006) - lanzamiento con mejoras y corrección de errores.
- JSF 2.0 (12-08-2009) - lanzamiento con mejoras de funcionalidad y performance y facilidad de uso.

- JSF 2.1 (22-10-2010) - versión actual. Lanzamiento de mantenimiento, con mínimos cambios.

## Arquitectura

El mecanismo de presentación basado en Ajax en ICE faces, su implementación es totalmente transparente para el desarrollador, es necesario comprender como sucede la magia bajo escena en las aplicaciones en ICEFaces. Hay tres elementos básicos de la arquitecturase ilustran en el diagrama a continuación:

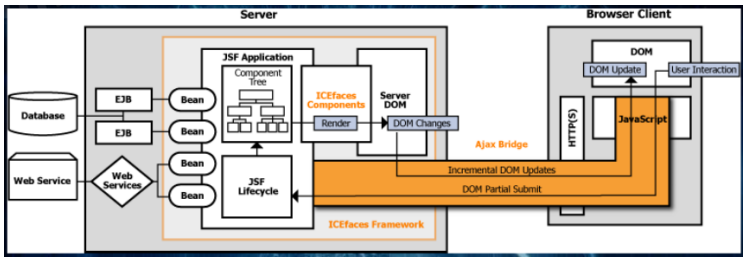



**Figura IV. 22Elementos básicos de la arquitectura**

**Fuente:** http://www.slideshare.net/maxmouse/java-server-faces-5561958

Arquitectura Sincroniza el Server-side DOM con el Client-Side DOM para leve actualizaciones en la página ICEfaces Framework. Provee todos las interfaces gráficas para lograr aplicaciones sofisticadas. Puente que comunica el servidor y el cliente para lograr actualizaciones de la página levemente ICEfacesComponentSuiteAjax Bridge


**Figura IV. 23Icefaces Component Suite Ajax Bridge**

**Fuente:** http://www.slideshare.net/maxmouse/java-server-faces-5561958

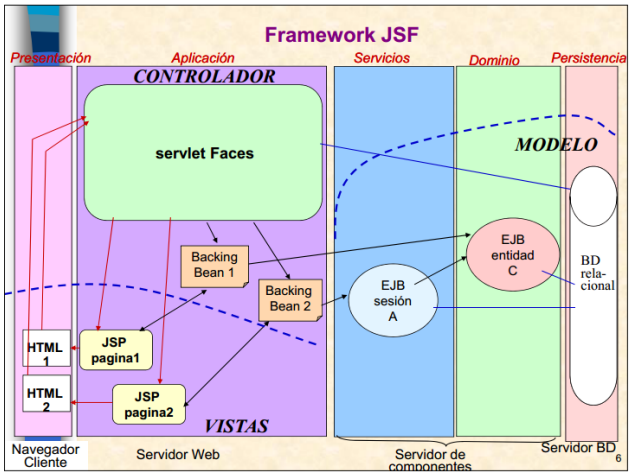



**Figura IV.24Elementos de JSF**

**Fuente:** http://www.slideshare.net/maxmouse/java-server-faces-5561958

**Componentes**

Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.

Un conjunto por defecto de componentes para la interfaz de usuario.

Dos bibliotecas de etiquetas personalizadas para JavaServerPages que permiten expresar una interfaz JavaServer Faces dentro de una página JSP.

Un modelo de eventos en el lado del servidor.

Administración de estados.

Beans administrados.

JSF introduce 2 nuevos términos al mundo del desarrollo de aplicaciones para JAVA: Managed Bean y Backing Bean

Un Managed Bean es un objeto identificado para el ambiente de la aplicación, para el cual se describe:

- Una identificación
- Un alcance (scope) que puede ser: request, session, application, etc
- Propiedades

Un BackingBean es usualmente un Bean común de java que sirve de soporte para un objeto manejando de la aplicación.

- La página JSP esta especificada como un ManagedBean para la aplicación, con un identificador que la describe para toda la aplicación en general.
- Esta página tiene asociado un BackingBean que es un Bean de Java. En este Bean se codifica los comportamientos específicos asociados a cada control del ManagedBean representado por la página JSP, la ventaja de los BackingBeans es que pueden ser compartidos por un mismo ManagedBean, de manera que para diferentes páginas se pueden agrupar comportamientos comunes en un mismo Bean que comparte con ambos

Las aplicaciones JSF están formadas por los siguientes elementos principales:

- Páginas JSP que incluyen los formularios JSF. Estas páginas generarán las vistas de la aplicación
- Beans java que se conectan con los formularios JSF
- Clases java para la lógica de negocio y utilidades.
- Ficheros de configuración, componentes a medida y otros elementos del framework.
- Resto de recursos de la aplicación web: recursos estáticos, JavaScript y otros elementos.

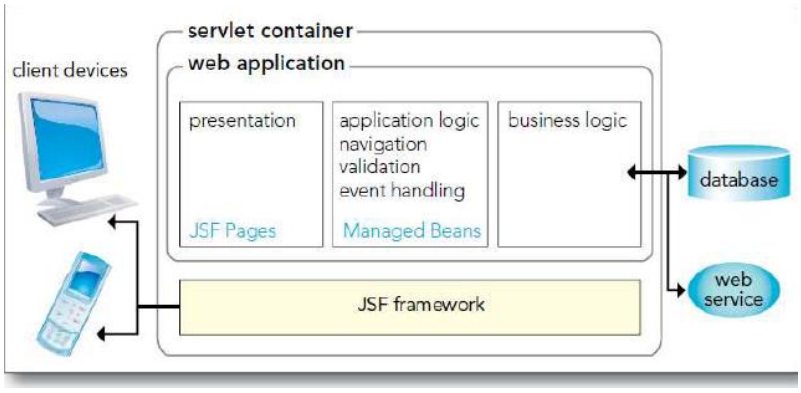



**Figura IV.25Elementos de Pagina JSF**

**Fuente:** http://www.slideshare.net/maxmouse/java-server-faces-5561958

**Servicios**

Los servicios más importantes son:

**Conversión de datos:** JSF soporta la conversión de etiquetas (<f:converter>), principalmente los usuarios ingresan datos en cadena desde los objetos de negocio como son los tipos de datos Integer o Short.

**Validación:** JSF soporta la validación de etiquetas (<f:validator>) y la validación personalizada (implementando la interfaz de validación) en la aplicación web.

**Control de Errores:** JSF es capaz de adjuntar reglas de error para el campo como "campo debe ser un número" o "formato de datos no válidos en el campo", este mensaje de error es llamado cuando el usuario introduzca datos no válidos, es necesario mostrar mensajes de error apropiados.

**Modelo-vista-controlador de Arquitectura:** La idea básica detrás del modelo MCV es separar los datos de aplicación y la lógica de negocio, la presentación de los datos, y la interacción con los datos en distintas entidades calificados como el modelo, la vista, y el Contralor, respectivamente.

**Componentes personalizados:** JSF soporta el desarrollo de componentes sofisticados, es decir, uno los diseñadores de páginas puede simplemente arrastrar y soltar los componentes creados en sus páginas.

**Internacionalización**: Al igual que otro marco, JSF también maneja los temas de internacionalización, como la codificación de caracteres y la selección de paquetes de recursos.

**Representadores alternativas:** Por defecto, JSF genera el marcado de las páginas HTML, ya que la página HTML es interpretado. Todas las etiquetas JSF o de texto (que no es una etiqueta JSF) que simplemente se pase por RENDERIZACIÓN. Ayuda a los clientes con el soporte de la tecnología móvil para desarrollar una aplicación web basada en el teléfono celular

**Características**

Utiliza páginas JSP para generar las vistas, añadiendo una biblioteca de etiquetas propia para crear los elementos de los formularios HTML.

Asocia a cada vista con formularios un conjunto de objetos java manejados por el controlador(managedbeans) que facilitan la recogida, manipulación y visualización de los valores mostrados en los diferentes elementos de los formularios.

Introduce una serie de etapas en el procesamiento de la petición, como por ejemplo la de validación, reconstrucción de la vista, recuperación de los valores de los elementos, etc.

Utiliza un sencillo fichero de configuración para el controlador en formato xml.

Es extensible, pudiendo crearse nuevos elementos de la interfaz o modificar los ya existentes.

Forma parte del estándar J2EE. En efecto, hay muchas alternativas para crear la capa de presentación y control de una aplicación web java, como Struts y otros frameworks, pero solo JSP forma parte del estándar.

Utiliza páginas JSP para generar las vistas, añadiendo una biblioteca de etiquetas propia para crear los elementos de los formularios HTML.

Asocia a cada vista con formularios un conjunto de objetos java manejados por el controlador (managedbeans) que facilitan la recogida, manipulación y visualización de los valores mostrados en los diferentes elementos de los formularios.

Introduce una serie de etapas en el procesamiento de la petición, como por ejemplo la de validación, reconstrucción de la vista, recuperación de los valores de los elementos, etc.

Utiliza un sencillo fichero de configuración para el controlador en formato xml

Es extensible, pudiendo crearse nuevos elementos de la interfaz o modificar los ya existentes.

Y lo que es más importante: forma parte del estándar J2EE. En efecto, hay muchas alternativas para crear la capa de presentación y control de una aplicación web java, como Struts y otros frameworks, pero solo JSP forma parte del estándar.

JSF nos permite desarrollar rápidamente aplicaciones de negocio dinámicas en las que toda la lógica de negocio se implementa en java, o es llamada desde java, creando páginas para las vistas muy sencillas (salvo que introduzcamos mucha maquetación HTML o JavaScript)

Especificación en las etiquetas de caminos (paths) relativos y absolutos para el servidor de acceso a recursos (imágenes, etc.). Traducción automática al cliente.

Asociación directa en las etiquetas a datos de la aplicación (Java Beans, etc.). Datos necesarios para la visualización de la página.Datos proporcionados por el usuario

(formularios).Otros datos que permiten parametrizar el código del servidor que genera la interfaz.

El modelo está formado por los Beans que guardan los datos generados por la aplicación.Cada solicitud incorpora datos nuevos, que pueden ser parámetros de control o información a incorporar al modelo del servidor. Utiliza exclusivamente peticiones POST, que van siempre al Servlets principal. No permiten motores de búsqueda ni guardar URLsfavoritas.

JSF proporciona una API basada encomponentes que se pueden usar paraensamblar aplicaciones web.

JSF proporciona un mecanismo de fácil empleomediante el cual los componentes UI en el ladoweb están débilmente acoplados (mediante unLenguaje de expresiones similar a JSTL) a losPOJO's del servidor (conocidos como"Managedbeans")Los "managedBeans" se declaran en el archivofaces-config,xmlo se usan anotaciones.El control de la conversación se realiza en el"JSF requestprocess Lifecycle".

El "JSF requestprocesslifecycle" tambiénpermite manejar las validaciones y conversionesdependiendo de los eventos que ocurren en laaplicación.JSF permite construir validacionespersonalizadas.

JSF proporciona el manejo de "resourcebundles" así como de localización (L10N). Loscomponentes UI automáticamente reconocenestas características una vez que el "bundle" hasido configurado.

JSF proporciona API's bastante flexiblesbasadas en tecnologías de "rendering" quepueden ser "enchufadas" bajo demanda. Porejemplo, si el cliente es un iPhone el "render" dela página será HTML específico para dichoequipo.

**Ventajas y Desventajas**

JSF nos ofrece una serie de ventajas:

- El código JSF con el que creamos las vistas (etiquetas jsp) es muy parecido al HTML estándar. Lo pueden utilizar fácilmente desarrolladores y diseñadores web.
- JSF se integra dentro de la página JSP y se encarga de la recogida y generación de los valores de los elementos de la página
- JSF resuelve validaciones, conversiones, mensajes de error e internacionalización
- JSF permite introducir JavaScript en la página, para acelerar la respuesta de la interfaz en el cliente (navegador del usuario).
- JSF es extensible, por lo que se pueden desarrollar nuevos componentes a medida, También se puede modificar el comportamiento del framework mediante APIs que controlan su funcionamiento.

Desde el punto de vista técnico podemos destacar los siguientes:

- JSF forma parte del estándar J2EE, mientras que otras tecnologías para creación de vistas de las aplicaciones no lo forman, como por ejemplo Struts.
- JSF dispone de varias implementaciones diferentes, incluyendo un conjunto de etiquetas y APIs estándar que forman el núcleo del framework.
- El desarrollo de JSF está realmente empezando. Las nuevas versiones del framework recogen la funcionalidad de versiones anteriores siendo su

compatibilidad muy alta, de manera que el mantenimiento de aplicaciones no se ve penalizado por el cambio de versiones.

El código JSF con el que se crean las vistas (etiquetas jsp) es muy parecido al HTML estándar. Lo pueden utilizar fácilmente desarrolladores y diseñadores web.

JSF se integra dentro de la página JSP y se encarga de la recogida y generación de los valores de los elementos de la página

JSF resuelve validaciones, conversiones, mensajes de error e internacionalización

JSF permite introducir JavaScript en la página, para acelerar la respuesta de la interfaz en el cliente (navegador del usuario).

JSF es extensible, por lo que se pueden desarrollar nuevos componentes a medida, También se puede modificar el comportamiento del framework mediante APIs que controlan su funcionamiento.

**Ventajas de 1.x JSF (Struts vs.Struts 1 x)**

- Los componentes personalizados

JSF hace que sea relativamente fácil de combinar complejas interfaces gráficas de usuarioen un componente único manejable; Struts no

- Mejor soporte para Mejor soporte para Ajax para Ajax

Varias bibliotecas de terceros componentes tienen una amplia Ajax apoyo (Apache Tomahawk, JBoss Ajax4jsf/RichFaces, Oracle ADF, ICEfaces). Struts no tiene verdaderas bibliotecas de componentes.

- Soporte

El soporte para otras tecnologías de visualización para otras tecnologías de visualización. JSF no se limita a HTML y HTTP; Struts si

- El acceso a los Beans por nombre

JSF permite asignar nombres a Beans, se refieren a por su nombre en los formularios. Struts tiene un proceso complejo con varios niveles de indirección.

**Ventajas de JSF (vs Struts)**

- Expresión idioma

El lenguaje de expresión JSF es más conciso y poderoso que Struts.

- Esto es menos ventajoso si se utiliza JSP 2.0 de todos modos.
- Controlador simple y definiciones de Beans
- JSF no requiere el controlador y clases de Beans a ampliar cualquier clase primaria particular o cualquier método particular. Struts lo hace.
- Simplificación de la configuración de archivos y la estructura general faces-config archivo xml es mucho más fácil de usar que es la struts-config.xml. En general, JSF es más simple.
- El apoyo potencial herramienta más poderosa
- La orientación en torno a los controles de GUI y sus manejadores abre la posibilidad de fácil de usar, arrastrar y soltar IDEs

**Desventajas de JSF (frente a Struts)**

- Base establecida y la industria impulso
- Struts llegó primero y se han implementado aplicaciones

- 1/2010 búsqueda en dice.com y monster.com

  "Puntales": 1.276 puestos de trabajo (dice.com), 596 puestos de trabajo (monster.com)

  "Jsf": 480 (dados), 615 (monstruo), pero algunos listados de usar "JavaServer Faces"

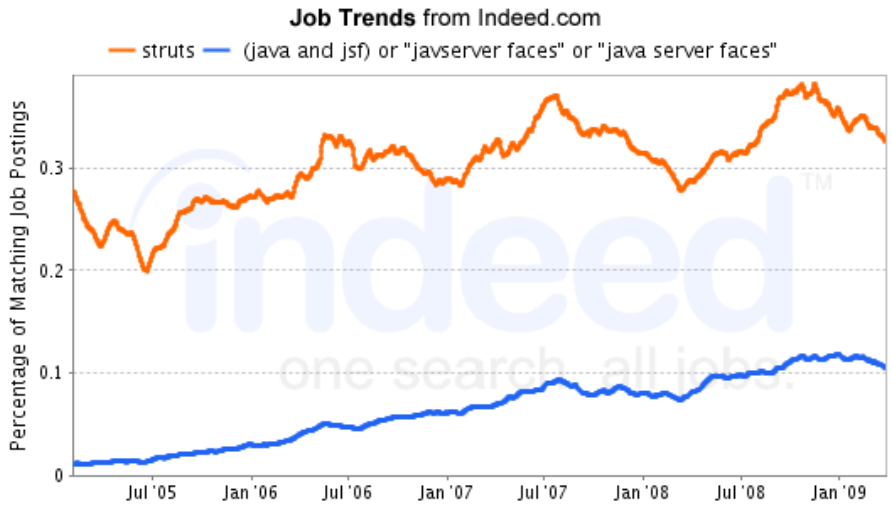



**Figura IV.26JSF vs STRUTS**

**Fuente:** http://blogs.jsfcentral.com/editorsdesk/category/Industry

- Apoyo a otras tecnologías de visualización JSF no se limita a HTML y HTTP; Struts sí.
- Confusión vs nombres de archivo, las páginas reales utilizados en el extremo de JSF JSP (1.x) o. xhtml (JSF 2.0). Pero el URL utilizadas terminan en. Caras o JSF. Esto causa muchos problemas, en particular, en JSF:
- No se puede navegar por los directorios y haga clic en los enlaces
- Dificultad para proteger primas páginas JSP de acceso
- Dificultad para referirse a los no rostros páginas en faces-config.xml

- Struts incluye instalación poderoso diseño de página; 1.x JSF no
- Facelets no es oficialmente parte de 1.x JSF (pero está en JSF 2.0)
- Validación mucho más débil automático
- Struts incluye validadores para la dirección de correo electrónico, tarjeta de crédito números, expresiones regulares, y más. JSF sólo viene con validadores para los valores perdidos, tiempo de entrada y números en un rango determinado.
- Pero MyFaces y otras bibliotecas de terceros componentes tienen varios validadores poderosos
- Falta de cliente de la validación del lado del cliente la validación
- Struts soporta JavaScript forma basada en la validación de campos; JSF no

**Funciones**

Normalmente las aplicaciones web se construyen como un conjunto de pantallas con las que va interactuando el usuario. Estas pantallas contienen textos, botones, imágenes, tablas y elementos de selección que el usuario modifica.

Todos estos elementos estarán agrupados en formularios HTML, que es la manera en que las páginas web envían la información introducida por el usuario al servidor.

La principal función del controlador JSF es asociar a las pantallas, clases java que recogen la información introducida y que disponen de métodos que responden a las acciones del usuario. JSF nos resuelve de manera muy sencilla y automática muchas tareas:

- Mostrar datos al usuario en cajas de texto y tablas.
- Recoger los datos introducidos por el usuario en los campos del formulario.

- Controlar el estado de los controles del formulario según el estado de la aplicación, activando, ocultando o añadiendo y eliminando controles y demás elementos
- Realizando validaciones y conversiones de los datos introducidos por el usuario
- Rellenando campos, listas, combos y otros elementos a medida que el usuario va interactuando con la pantalla
- Controlando los eventos que ocurren en los controles (pulsaciones de teclas, botones y movimientos del ratón).

## 4.4. Determinación de los parámetros de comparación

Con el fin de determinar los parámetros que nos servirán para comparar los dos frameworks Java MVC más importantes para el desarrollo de aplicaciones web empresariales Struts y JSF, estableciendo pautas que faciliten la elección a la hora de decidir la solución más adecuada para la implementación del sistema.

### 4.4.1. Parámetro 1: Producto

En este parámetro se va analizar las características más importantes propias de los dos frameworks considerados y que son independientes del rendimiento, importante al momento de la elección de la herramienta para evitar posibles problemas durante el desarrollo de la aplicación.

### 4.4.2. Parámetro 2: Desarrollo

Este parámetro tiene la finalidad de conocer las facilidades que brinda cada framework para el desarrollo de la aplicación web, así como los componentes que estos poseen para mejorar la interfaz del usuario en cuanto a accesibilidad.

### 4.4.3. Parámetro 3: Rendimiento

Con este parámetro se va a evaluar el porcentaje dela utilización de los recursos necesarios para ejecutar cada una de las aplicaciones desarrolladas en los dos frameworks.

### 4.4.4. Parámetro 4: Patrón de Diseño MVC

El patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos, es el mismo que caracteriza a los frameworks STRUTS y JSF.

### 4.4.6. Parámetro 5: Seguridad

Es importante mantener seguridades en las aplicaciones web, para de esta manera incrementar la confiabilidad del usuario en la navegación por las páginas puesto que ciertos datos necesitan ser privados, y que no puedan ser usados por personas no autorizadas.

En la siguiente tabla tenemos los parámetros que se usaran en el análisis.

**Tabla IV.II. Parámetros para el Análisis Comparativo**

| PARÁMETRO | INDICADORES |
|---|---|
| | Madurez |

| | |
|---|---|
| Producto | Documentación |
| | Modo de Licenciamiento |
| | Integración con otros frameworks |
| | Curva de Aprendizaje |
| Desarrollo | Configuración del Framework |
| | Líneas de código escritas |
| | Interfaz de usuario Accesibilidad |
| | Uso de Componentes |
| Rendimiento | Uso de CPU |
| | Uso de Memoria |
| Patrón de Diseño MVC | Modelo |
| | Vista |
| | Controlador |
| Seguridad<br>A nivel de Aplicación | Uso de sesiones |
| | Encriptación de Datos |
| | Validación de datos |

## 4.5. Análisis Comparativo

En esta sección se va a realizar la comparación de los Frameworks MVC JSF y STRUTS, para el desarrollo de aplicaciones web empresariales J2EE, la comprobación se realizará por medio de cuadros comparativos en los que se va a determinar las variables de cada uno

de los parámetros tomando en cuenta para el respectivo análisis, seguidos de una valoración, interpretación y calificación evaluado por parte de los autores.

Para obtener resultados cuantitativos y cualitativos que permitan una selección adecuada de uno de los frameworks analizados para desarrollar aplicaciones web empresariales, se realizara la calificación de cada uno de los parámetros de comparación que se basa en la siguiente tabla.

**Tabla IV.IIIEscala de calificación para parámetros de comparación**

| REGULAR | BUENO | MUY BUENO | EXCELENTE |
|---|---|---|---|
| < 70% | >=70% y <80% | >=80% y <90% | >=90% |

**Fuente**: Autoras

La siguiente tabla permitirá saber la valoración cuantitativa para las variables de los parámetros seleccionados:

**Tabla IV. IVEscala de valoración Cuantitativa**

| 1 | 2 | 3 |
|---|---|---|
| **No se cumple** | Se cumple parcialmente | Se cumple |
| **Malo** | Bueno | Excelente |
| **Deficiente** | Poco eficiente | Muy eficiente |
| **Nada** | Poco | Mucho |
| **No** | Más o menos | Si |
| **Empezando** | Estable | Maduro |

**Fuente:** Autoras

Donde cada una de las variables van a ser evaluadas sobre el máximo que es **3** y cada uno de los ítems de la interpretación incluye la siguiente nomenclatura (x, y)/w en donde cada letra significa lo siguiente:

**Tabla IV. VSignificado de nomenclatura**

| | |
|---|---|
| X | Representa el puntaje que obtiene el Framework JSF |
| **Y** | Representa el puntaje que obtiene el Framework STRUTS |
| **W** | Representa la base del puntaje sobre la cual se está calificando el parámetro |

**Fuente:** Autoras

La calificación definitiva de la herramienta en base a cada parámetro de comparación se obtiene sumando los puntajes obtenidos del análisis, utilizando las siguientes formulas:

$$P_{jsf} = \sum(x),\ P_{struts} = \sum(y),\ P_c = \sum(w)$$

Calificación de JSF: $(C_c - JSF) = (P_{jsf} / P_c) * 100\%$

Calificación de STRUTS: $(C_c - STRUTS) = (P_{struts} / P_c) * 100\%$

En donde:

**Tabla IV.VISignificado de variables**

| | |
|---|---|
| $P_{jsf}$ | Puntaje acumulado por JSF en el parámetro analizado |
| $P_{struts}$ | Puntaje acumulado por STRUTS en el parámetro analizado |
| $P_c$ | Puntaje sobre el que se califica el parámetro analizado |

| | |
|---|---|
| $C_c - jsf$ | Porcentaje de la calificación total que obtuvo JSF en el parámetro |
| $C_c - struts$ | Porcentaje de la calificación total que obtuvo STRUTS en el parámetro |

**Fuente:** Autoras

### 4.5.1. Parámetro 1: Producto

#### 4.5.1.1. Determinación de las Variables

a. Madurez
b. Documentación
c. Modo de Licenciamiento
d. Integración con otros frameworks
e. Curva de Aprendizaje

- **Valorizaciones**

A continuación se describe las consideraciones a evaluar en cada uno de las variables ya determinadas en el parámetro producto.

**a. Variable Madurez**

Este criterio se refiere al crecimiento del framework a través del tiempo, desde su creación hasta la actualidad, adaptándose a las nuevas tecnologías, procesos y modelos, con el objetivo de cubrir de mejor manera los requerimientos del usuario y adaptarse a nuevos entornos, sin perder rendimiento y compatibilidad, logrando con esto mantener o mejorar su presencia en el mercado.

**b. Variable Documentación**

La cantidad de documentación como manuales, tutoriales, blogs y foros de discusión es un aspecto importante, ya que estos permiten conocer conceptos básicos y resolver problemas

**c. Variable Modo de Licenciamiento**

Esta variable permite conocer la licencia bajo la cual se distribuyen los frameworks analizados.

**d. Variable Integración con otros frameworks**

Esta variable permitirá definir si los frameworks analizados tienen compatibilidad o integración con otros frameworks que ampliarían sus capacidades aún más.

**e. Curva de Aprendizaje**

Indica el esfuerzo requerido para la comprensión y aprendizaje de la forma de uso del producto. Esta medida es importante en lo que refiere a la capacitación de los desarrolladores en el caso de no tener experiencia con el producto.

**Tabla IV.VIIResumen variables parámetro Producto**

| VARIABLE | STRUTS | JSF |
|---|---|---|
| **Madurez** | Maduro | Maduro |
| **Documentación** | Buena | Excelente |
| **Modo de licenciamiento** | Bueno | Excelente |
| **Integración con otros frameworks** | Bueno | Excelente |
| **Curva de Aprendizaje** | Alta | Media |

**Fuente:** Autoras

#### 4.5.1.2. Interpretación de la valorización

Las escalas definidas en la Tabla IV.IV y en la Tabla IV.V, se han aplicado a las variables valorizadas como se indican en la tabla IV.VII, interpretándolas de la siguiente manera.

- En cuanto a documentación, STRUTS y JSF posee gran cantidad de información de alta calidad, artículos, foros, manuales de configuración, manuales donde se explica con claridad cada ejemplo. Pero como un punto a favor de JSF debido a que es utilizado por muchos desarrolladores para sus aplicaciones web existe una gran cantidad de ejemplos actuales realizados y tiene algunas comunidades como son GUJ, Grupo de JSF, Java Noroeste, PrimeFaces, RichFaces, Oracle JavaServer Faces caso contrario a lo que pasa con STRUTS aunque un framework más antiguo no se está utilizando actualmente por ello no existe documentación de aplicaciones actuales. (3,2)/3
- Struts es el framework web J2EE estándar de-facto desde principios del 2000, y ha alcanzado un alto grado de madurez. Tiene una comunidad de desarrolladores muy activa, existe mucha documentación disponible, libros, *mailing-list*, se han escrito centenares de artículos, etc. Las IDEs JAVA más importantes del mercado, lo soportan. JSF es un framework web J2EE, que ha emergido en el año 2004, cuenta con la ventaja de ser el primer framework J2EE con una especificación que está incluida en la versión actual de J2EE, lo que lo transforma en el primer estándar y de esta manera obliga a todas las implementaciones de servidores J2EE a soportarlo. Naturalmente, Struts es el más maduro de los dos frameworks, tuvo una gran aceptación desde su aparición, y en la actualidad sigue evolucionando en dos

sentidos, Struts 1 y Struts 2.Sin embrago a pesar que STRUTS es más antiguo JSF corrige algunos errores de STRUTS por ello es más utilizado actualmente. (3,3)/3

- Ambos frameworks afortunadamente son software libre. Pero JSF también es de código abierto lo que significa ventajas en el modelo de desarrollo. (3,2)/3
- Integración con otros frameworks, JSF puede integrarse a una cantidad considerable de frameworks ya sean de aplicaciones web o persistencia entre estos tenemos: dojo, jquery, spring, hibernate, gwt y ibatis. Caso contrario a STRUTS que tiene un número pequeño de frameworks a los que se puede integrar como: Spring, Hibernate, Ajax y Flex. (3,2)/3
- Curva de Aprendizaje, en el caso de JSF es media , es más fácil que aprender STRUTS puesto que es más ordenado, lo opuesto a STRUTS por el hecho de que provee una API con sus métodos y sus clases, las cuales tienes que extender para implementar la lógica de negocio. Esto hace que tengas que tener un conocimiento aceptable de dichas clases, susresponsabilidades y sus interacciones, lo que retrasa la hora de empezar a desarrollar. (2,1)/3

- **Calificación**

$$P_{jsf} = \sum(x) = 3+3+3+3+2 = 14$$

$$C_c - jsf : (P_{jsf} / P_c) * 100\% = (14/15) * 100\% = 93.33\%$$

$$P_{struts} = \sum(y) = 2+3+2+2+1 = 10$$

$$C_c - struts : (P_{struts} / P_c) * 100\% = (10/15) * 100\% = 66.67\%$$

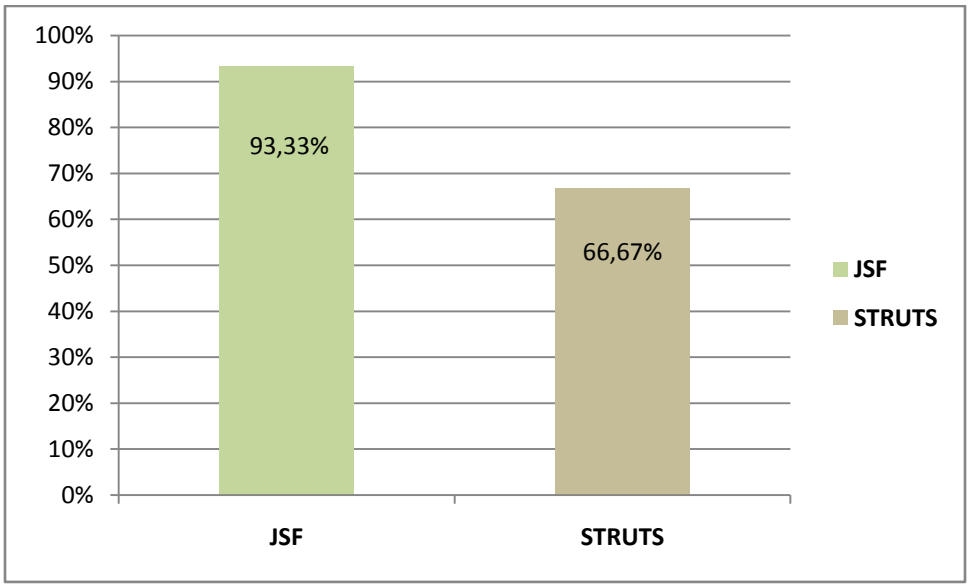



**Figura IV.27Comparación estadística del Parámetro Producto**

**Fuente:** Autoras

### 4.5.2. Parámetro 2: Desarrollo

Las siguientes variables han sido escogidas en función de las herramientas que provee cada uno de los frameworks para el desarrollo de una aplicación web.

#### 4.5.2.1. Determinación de las variables

**a.** Configuración del Framework

**b.** Líneas de código escritas

**c.** Interfaz de Usuario

**d.** Uso de Componentes

- **Valorizaciones**

A continuación se describe las consideraciones a evaluar en cada uno de las variables ya determinadas en el parámetro desarrollo.

**a. Configuración del framework**

Esta variable es muy importante debido a que la configuración previa de un framework es el paso inicial para su funcionamiento, se evaluara si los frameworks ofrecen una configuración transparente y el grado de dificultad que presenta.

**b. Líneas de código escritas**

Este indicador permitirá evaluar la cantidad de líneas de código que se llevan a cabo para realizar un determinado proceso con STRUTS y JSF, con la finalidad de establecer el framework que menos líneas de código genera.

**c. Interfaz de Usuario**

Por medio de esta variable se evaluara la accesibilidad, se observara si con la utilización de alguno de los dos frameworks se crean interfaces perceptibles, operables y comprensibles para personas con distintos tipos de capacidades.

**d. Uso de componentes**

Este indicador permitirá evaluar los componentes disponibles de cada framework para generar interfaces que sean intuitivas para el usuario final.

Para poder valorizar e interpretar cada variable de este parámetro se realizó una prueba mediante el desarrollo de dos aplicaciones una con JSF y la otra con STRUTS, con el fin de comparar las características de desarrollo, en estas pruebas se van a determinar la complejidad de configuración de cada framework, la cantidad de líneas de código necesarias, los componentes utilizados para que despliegue un listado y una forma como la que se muestra en la figura siguiente.

**Tabla IV.VIIIResumen Variables Parámetro Desarrollo**

| VARIABLE | STRUTS | JSF |
|---|---|---|
| **Configuración del Framework** | Mucho | Poco |
| **Líneas de Código escritas** | Mucho | Poco |
| **Interfaz de Usuario Accesibilidad** | Compleja | Fácil |
| **Uso de Componentes** | Excelente | Buena |

**Fuente:** Autoras

### 4.5.2.2. Interpretación de la valorización

Las escalas definidas en la Tabla IV.IV y en la Tabla IV.V, se han aplicado a las variables valorizadas como se indican en la tabla IV.VIII, interpretándolas de la siguiente manera.

- En lo relacionado en la configuración del framework encontramos que en struts-config.xml se debe crear un <form-bean> para cada forma y un <action> en action-mappings por cada evento y además un forward name para cada succes o failure lo que lo que consume más tiempo del programador, cado contrario a faces-config.xml donde se crea un managed bean por cada value object y una regla de navegación por cada evento haciendo menos tedioso el trabajo de su configuración. (3,2)/3
- Líneas de código escritas, en el caso de JSF se deben escribir menos líneas de código debido a que para el listado se puede utilizar un datatable, código javascript del componente sin escribirlo, dar de alta al managed vean en faces-config.xml, se configura la navegación en pocas líneas de código, y se realiza el código de manejo de eventos por componente. Contrario a STRUTS que necesita desarrollar el código para controlar el tipo de componente a desplegar según la comuna, código

para paginar la tabla, crear una clase action para cada botón donde cada una de ellas tienen sus respectivos action mappings, y se necesita por lo menos dos fordwards para cada action y los global forwards necesarios para todos. (3,2)/3

- En cuanto a la accesibilidad de la interfaz de usuario, se puede decir que JSF proporciona mayor cantidad de componentes para crear interfaces que sean claras ya que posibilita los editores visuales y es independiente de la tecnología de visualización caso contrario a STRUTS que no es tan flexible y extensible para proporcionar componentes que generen interfaces claras para el usuario. (3,2)/3
- Uso de componentes, la mayor ventaja que posee JSF ante STRUTS es que su modelo de componentes de interfaz de usuario es flexible y extensible que incluye una API de los componentes estándar para especificar el estado y comportamiento de una amplia gama de componentes, desde componentes simples y complejos, pero además los desarrolladores pueden crear sus propios componentes. Por ello es que se han creado IceFaces, RichFaces, PrimeFaces facilitando el desarrollo de aplicaciones web permitiendo combinar fácilmente complejas GUIS en un componente lo que no es posible en STRUTS. (3,2)/3
- **Calificación**

$$P_c = \sum(w) = 3+3+3+3 = 12$$

$$P_{jsf} = \sum(x) = 3+3+3+3 = 12$$

$$C_c - jsf : (P_{jsf} / P_c) * 100\% = (12/12) * 100\% = 100\%$$

$$P_{struts} = \sum(y) = 2+2+2+2 = 8$$

$$C_c - struts : (P_{struts} / P_c) * 100\% = (8/12) * 100\% = 66.67\%$$

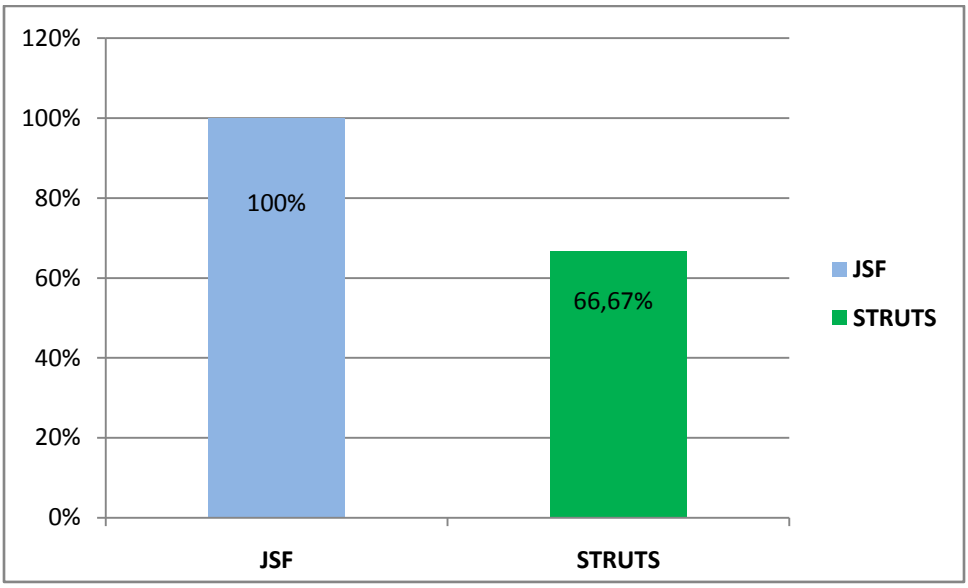



**Figura IV.28Comparación estadística del parámetro Desarrollo**

**Fuente:** Autoras

### 4.5.3. Rendimiento

En este grupo se consideran aquellos aspectos que puedan aportar a la diferenciación de los frameworks analizados.

#### 4.5.3.1. Determinación de Variables

a. Uso de CPU
b. Uso de Memoria

- **Valorizaciones**

A continuación se describe las consideraciones a evaluar en cada uno de las variables ya determinadas en el parámetro producto.

**a) Uso de CPU**

Se refiere a la cantidad de CPU ocupada por la aplicación web creada por medio del Framework al momento de crear un registro y listarlo junto con todos los ingresados anteriormente.

**b) Uso de Memoria**

Se refiere a la cantidad de memoria ocupada por la aplicación web creada por medio del Framework al momento de crear un registro y listarlo junto con todos los ingresados anteriormente.

Para poder valorizar e interpretar cada variable este parámetro se realizó una prueba mediante la ejecución de las dos aplicaciones anteriores, con el fin de comparar las características de rendimiento, en estas pruebas se va a determinar el uso de memoria y uso de procesador al momento de ejecutar la inserción y listado de registros. Con este propósito se crearon las aplicaciones en los siguientes escenarios.

**Tabla IV.IXEscenario 1. JSF**

| Ítem | Descripción |
|---|---|
| **HARDWARE** | |
| Procesador | Intel Core i5 2.5 Ghz |
| Memoria RAM | 4 GB |
| Disco Duro | 600 GB |
| Cache | 2.5 g Ghz |
| **SOFTWARE** | |
| Sistema Operativo | Windows 7 Home Premium |

| | |
|---|---|
| IDE | Netbeans |
| Base de Datos | PostgreSQL |
| Framework de Desarrollo | JSF |

**Fuente:** Autoras

**Tabla IV.XEscenario 2. STRUTS**

| **Ítem** | **Descripción** |
|---|---|
| **HARDWARE** | |
| Procesador | Intel Core i5 2.5 Ghz |
| Memoria RAM | 4 GB |
| Disco Duro | 600 GB |
| Cache | 2.5 g Ghz |
| **SOFTWARE** | |
| Sistema Operativo | Windows 7 Home Premium |
| IDE | Netbeans |
| Base de Datos | PostgreSQL |
| Framework de desarrollo | Struts |

**Fuente:** Autoras

Posterior al desarrollo de los aplicativos en las siguientes figuras se presentan los resultados obtenidos de las pruebas realizadas.

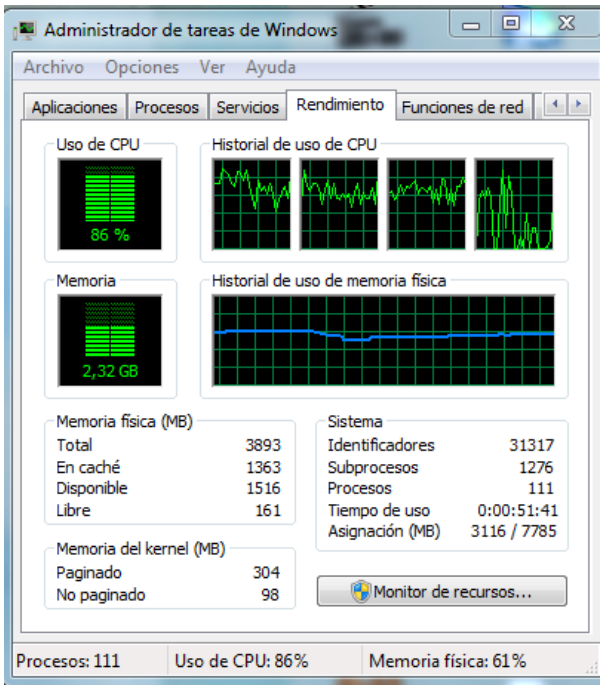



**Figura IV.29Test de Rendimiento de STRUTS**

**Fuente:** Autoras

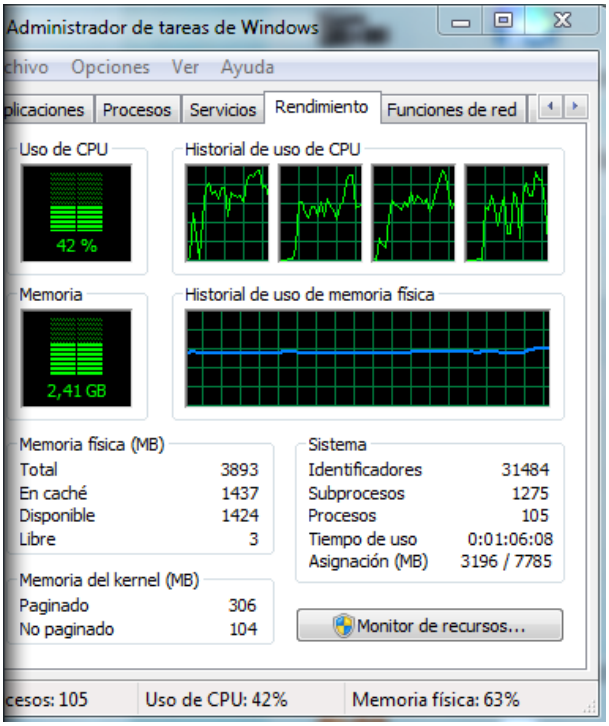



**Figura IV.30Test de Rendimiento de JSF**

**Fuente:** Autor

**Tabla IV.XIResumen variables parámetro Rendimiento**

| VARIABLE | STRUTS | JSF |
|---|---|---|
| **Uso de CPU** | Muy Eficiente | Poco Eficiente |
| **Uso de Memoria** | Poco eficiente | Muy eficiente |

**Fuente:** Autoras

### 4.5.3.2. Interpretación de la valorización

Las escalas definidas en la Tablas IV.IV y en la Tabla IV.V, se han aplicado a las variables valorizadas como se indican en la tabla IV.XI, interpretándolas de la siguiente manera.

- En cuanto al uso de la CPU se comprobó que STRUTS utiliza menor porcentaje caso contrario a JSF que utiliza en mayor cantidad este recurso. (2,3)/3
- Al analizar las gráficas de rendimiento de los dos frameworks se observa que la cantidad de memoria requerida por JSF es menor a la ocupada por STRUTS. (3,2)/3

- **Calificación**

$$P_c = \sum(w) = 3+3 = 6$$

$$P_{jsf} = \sum(x) = 2+3 = 5$$

$$C_c - jsf : (P_{jsf} / P_c) * 100\% = (5/6) * 100\% = 83{,}33\%$$

$$P_{struts} = \sum(y) = 3+2 = 5$$

$$C_c - struts : (P_{struts} / P_c) * 100\% = (5/6) * 100\% = 83{,}33\%$$

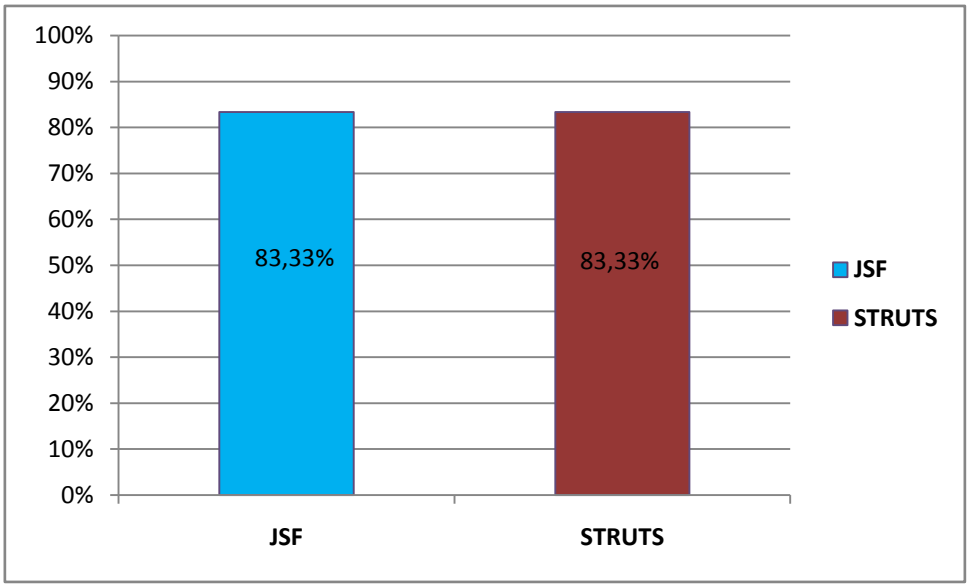



**Figura IV.31 Comparación estadística del Parámetro Rendimiento**

**Fuente:** Autoras

### 4.5.4. Patrón de diseño MVC

Las siguientes variables han sido escogidas en función de las herramientas que provee cada uno de los frameworks para el desarrollo de una aplicación web.

#### 4.5.2.1. Determinación de las variables

a. Modelo

b. Vista

c. Controlador

- **Valorizaciones**

A continuación se describe las consideraciones a evaluar en cada uno de las variables ya determinadas en el parámetro Patrón de diseño.

El patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos nos ayudaran a diferenciar los dos frameworks de estudio para ello se ha tomado como referencia las siguientes variables:

a. **Modelo:**Esta es la representación específica de la información con la cual el sistema opera.
b. **Vista:** Manejo de los componentes de Interfaz Gráfica para el diseño de páginas.
c. **Controlador:**Responde a eventos del usuario e invoca cambios en el modelo y probablemente en la vista.

**Tabla IV.XIIResumen variables parámetro patrón de diseño**

| VARIABLE | STRUTS | JSF |
|---|---|---|
| **Modelo** | Parcial | Excelente |
| **Vista** | Eficiente | Muy Eficiente |
| **Controlador** | Eficiente | Muy Eficiente |

**Fuente:** Autoras

### 4.5.3.2. Interpretación de la valorización

Las escalas definidas en la Tabla IV.IV y en la Tabla IV.V, se han aplicado a las variables valorizadas como se indican en la tabla IV.XII, interpretándolas de la siguiente manera.

- **Modelo**

Los componentes del modelo proporcionan un modelo de la lógica de negocio detrás de Struts. Proporcionan interfaces a la base de datos o sistemas “back-ends”. Los componentes

del modelo son generalmente clases java. No hay ningún formato específico para el modelo, lo cual permite reutilizar código ya escrito para otros proyectos. El modelos e suele elegir de acuerdo a los requerimientos del cliente. Por otro lado con JSF se introduce el concepto de bean administrado. El bean administrado es el pegamento de la lógica de la aplicación. Los beans administrados son definidos en el fichero faces-config.xml file y dan al desarrollador de la aplicación total acceso a los métodos mapeados de los beans. Este concepto de IoC (Inversion of control) es ya empleado de forma muy exitosa en Spring por ejemplo. La característica de beans administrados es responsable de crear beans de respaldo y otros beans tales como Data Access Objects (DAO). En JSF, un bean de respaldo es un POJO sin dependencias de una implementación específica en clases o interfaces. El controlador de JSF el FacesServlet no es consciente de que acción se ha tomado, solo es consciente del resultado una particular acción y utilizará este resultado para decidir sobre la navegación. En JSF este servlet es el componente que es consciente de que acción o método hay que llamar en cada evento de usuario. (3,2)/3

- **Vista**

  JSF cuenta con un conjunto de componentes UI, que pueden personalizarse y extenderse para crear nuevos componentes UI. La arquitectura flexible y extensible de JSF permite asociar renders diferentes para distintas tecnologías clientes como son teléfonos celulares, PDAs, además del típico cliente web. Caso contrario a Struts tiene un conjunto de custom tags que facilitan al desarrollador la creación de formularios HTML para ingreso de datos y que interactúa con Struts. Struts soporta un único cliente que es un navegador web. (3,2)/3

- **Controlador**

  JSF y Struts centralizan el manejo de peticiones de los clientes. Con JSF cada uno de los componentes que conforman una página JSF puede tener asociados comportamientos personalizados como son conversiones, validaciones y procesamiento de eventos. JSF da muchos beneficios al controlador, como la capacidad de manejar múltiples eventos sobre una página. El controlador de Struts está implementado siguiendo el concepto de caja gris, que permite definir puntos de extensión y proveer un comportamiento particular para procesar peticiones y manejar errores. Struts solo puede manejar un evento por página. (3,2)/3

- **Calificación**

$$P_c = \sum(w) = 3+3+3 = 9$$

$$P_{jsf} = \sum(x) = 3+3+3 = 9$$

$$C_c - jsf : (P_{jsf} / P_c) * 100\% = (9/9) * 100\% = 100\%$$

$$P_{struts} = \sum(y) = 2+2+2 = 6$$

$$C_c - struts : (P_{struts} / P_c) * 100\% = (6/9) * 100\% = 66{,}67\%$$

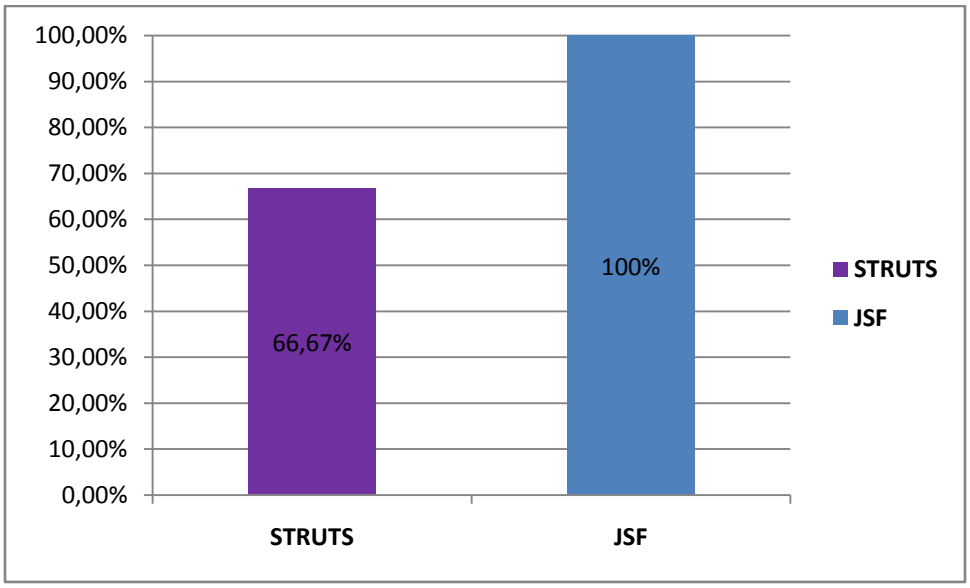



**Figura IV.32Comparación estadística del Parámetro patrón de Diseño MVC**

**Fuente:** Autoras

### 4.5.5. Seguridad

El manejo de seguridades podrá ser evaluado mediante las opciones que proporcionan los frameworks para generar los principales ítems de seguridades como sesiones y validación de datos, además de la facilidad que brinda cada framework para el manejo de variables de sesiones.

#### 4.5.2.1. Determinación de las variables

a. Uso de sesiones
b. Encriptación de Datos
c. Validación de datos

**A nivel de Aplicación**

- **Valorizaciones**

A continuación se describe las consideraciones a evaluar en cada uno de las variables ya determinadas en el parámetro Seguridad.

**A nivel de Aplicación:**Permite que la aplicación proporcione confianza al usuario, y de esta manera no existan ataques a la funcionalidad del sistema. Mediante el manejo de roles dinámicos en la aplicación (es decir solo usuario privilegiados tienen acceso a ciertas características de la aplicación, y estas pueden ser auditadas). Por ello se van a evaluar las siguientes variables, mediante la autenticación que es la que explota el método de validación de la identidad de un usuario, servicio o aplicación.

a. **Uso de sesiones**

   El manejo de sesiones en una página web es de gran importancia para mantener una administración del contenido de nuestra página por lo cual se ha toman en cuenta esta variable sonde se analizara la facilidad de uso y gestión de sesiones.

b. **Encriptación de datos**

   Un aspecto muy importante en cuanto a la seguridad de datos que atraviesa la gran red del internet es la capacidad de transportar datos cifrados.

c. **Validación de datos**

   La validación de datos es algo que siempre se debe tomar en cuenta para mantener más segura nuestra página dando así privilegios de uso de información.

**Tabla IV.XIIIResumen variables parámetro Seguridad a nivel de Aplicación**

| VARIABLE | STRUTS | JSF |
|---|---|---|
| **Uso de sesiones** | Poco Eficiente | Eficiente |
| **Encriptación de datos** | Poco Eficiente | Eficiente |
| **Validación de datos** | Bueno | Excelente |

**Fuente:** Autoras

### 4.5.3.2. Interpretación de la valorización

Las escalas definidas en la Tabla IV.IV y en la Tabla IV.V, se han aplicado a las variables valorizadas como se indican en la tabla IV.XIII, interpretándolas de la siguiente manera.

**A nivel de Aplicación**

- Uso de sesiones, JSF provee varios ámbitos para el manejo de sesiones como son el none, application, sesión y request, el ámbito de la sesión permanece desde que la sesión es establecida hasta que termina, no soporta GET lo cual puede ser una gran ventaja en el momento de mantener la seguridad de los datos. En STRUTS por otro ladotambién existen distintos ámbitos de sesiones pero no brinda muchas seguridades. (2,1)/3
- Encriptación de datos, ninguna de las dos tecnologías proporcionan elementos que faciliten esta tarea. Pero en el caso de JSF se puede desarrollar una librería de componentes JSF seguras que utilice las APIs criptográficas de Java (JCA y JCE) para ocultar los valores de las etiquetas ingresados por el desarrollador en los atributos de los componentes y STRUTS tiene control de acceso de los datos de login y control de permisos de los usuarios. (2,1)/3

- Validación de datos, JSF permite validad individualmente cada componente del formulario, se puede validar usando los validadores estándares, creando métodos validadores en los backing beans o creando clases validadoras especiales y realiza la conversión de datos con una granularidad más fina ya que es posible asociarle conversores específicos a los componentes, provee conversores de los tipos de datos más comunes como fechas y monedas, además soporta regionalización y es posible crear conversores especiales. En STRUTS por otro lado la validación se hace validando al objeto ActionForm completo, que representa todos los campos del formulario de entrada y la conversión de datos usa el estándar de JavaBeans. (3,2)/3

- **Calificación**

$$P_c = \sum(w) = 3+3+3 = 9$$

$$P_{jsf} = \sum(x) = 2+2+3 = 7$$

$$C_c - jsf : (P_{jsf} / P_c) * 100\% = (7/9) * 100\% = 77{,}78\%$$

$$P_{struts} = \sum(y) = 1+1+2 = 4$$

$$C_c - struts : (P_{struts} / P_c) * 100\% = (4/9) * 100\% = 44{,}44\%$$

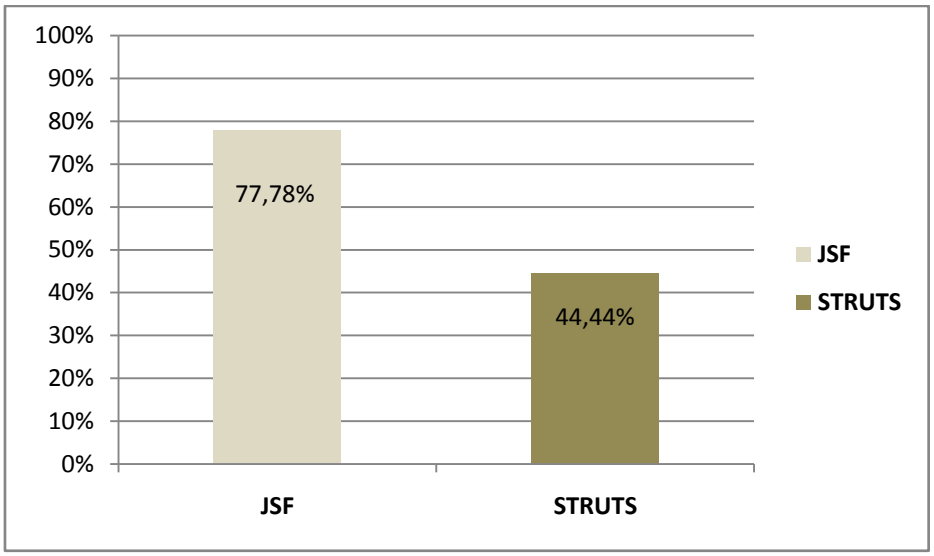



**Figura IV.33Comparaciónestadística del parámetro Seguridad a nivel de Aplicación**

**Fuente:** Autoras

## 4.6. Análisis de los Resultados

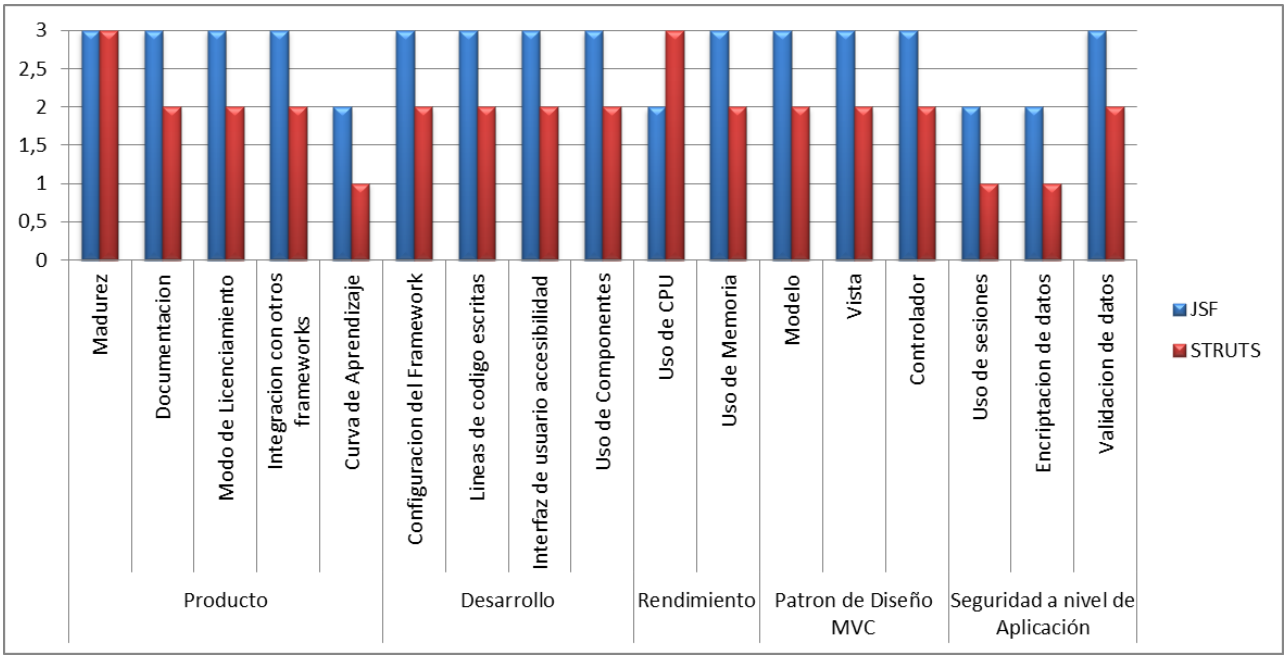



**Figura IV.34Resultado final por parámetro**

**Fuente:** Autoras

$P_T = 15+12+6+9+9 = 51$

$PT_{jsf} = 14+12+5+9+7 = 47$

$PT_{struts} = 10+8+5+6+4 = 33$

$(\%Jsf) = (47/51)*100 = 92{,}16\%$ **Equivalente a Excelente**

$(\%Struts) = (33/51)*100 = 64{,}71\%$ **Equivalente a Regular**

Resultados que podemos representar de la siguiente manera:

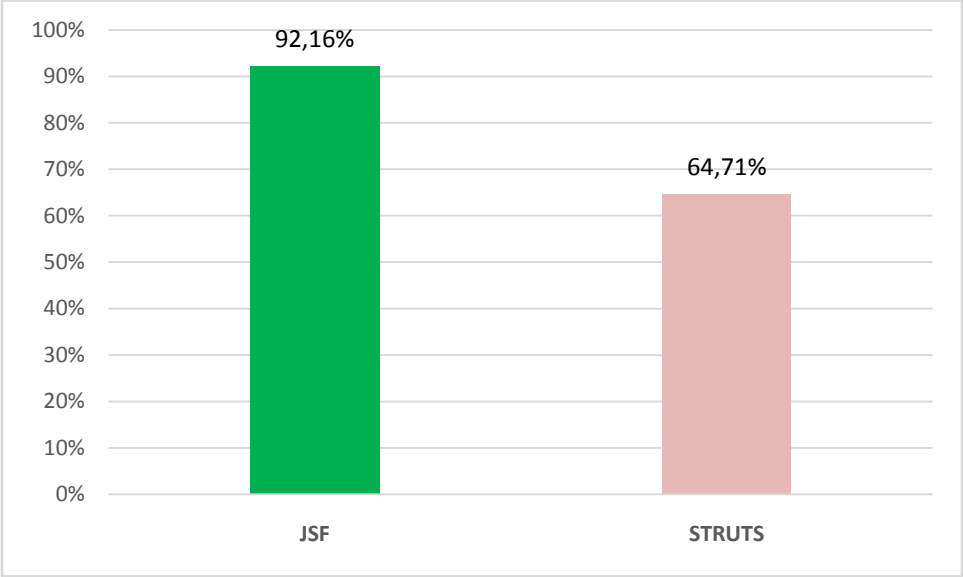



**Figura IV.35Diagrama general de resultados**

**Fuente:** Autoras

Después de haber realizado el análisis comparativo correspondiente de los respectivos parámetros con sus variables, como se puede observar en la Figura IV.35se ha determinado que para desarrollar una aplicación web empresarial sobre la arquitectura J2EE, el Framework JSF es el que tiene una máxima calificación de 92,16% correspondiente a

excelente por lo que se determina que es el framework más adecuado para desarrollar la aplicación web empresarial en ambiente J2EE; por otro lado STRUTS obtuvo una calificación de 64,71% que equivale a regular.

## 4.7. Demostración de la Hipótesis

Para poder realizar la demostración de la hipótesis de este trabajo investigativo se va a probar que la utilización de un framework MVC de Java facilitará el desarrollo de aplicaciones Web empresariales con mejor rapidez y seguridad.

Se tomará en consideración dos variables que inciden en el desarrollo de la aplicación web brindando rapidez y seguridad.

**Tabla IV.XIVResumen variables parámetro Desarrollo JSF vs STRUTS**

| VARIABLE | STRUTS | JSF |
|---|---|---|
| **Configuración del Framework** | Mucho | Poco |
| **Líneas de Código escritas** | Mucho | Poco |
| **Interfaz de Usuario Accesibilidad** | Compleja | Fácil |
| **Uso de Componentes** | Excelente | Buena |

**Fuente:** Autoras

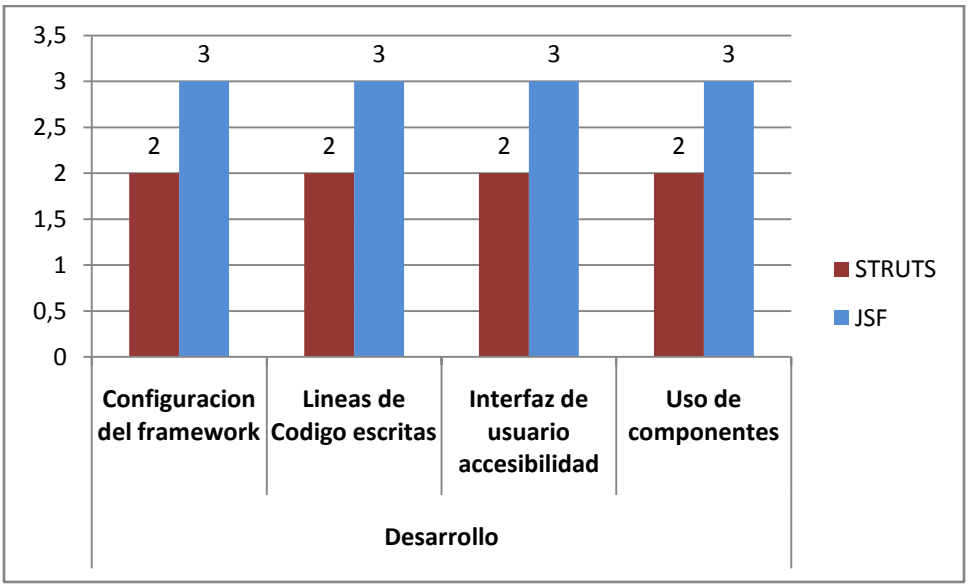



**Figura IV.36Resultado final del Parámetro Desarrollo**

**Fuente:** Autoras

Mediante la tabla y el cuadro estadístico anteriores se puede verificar que el desarrollo de una aplicación web en el framework JSF se reduce inevitablemente debido a que no se invierte demasiado tiempo en la configuración del framework, se escriben menos líneas de código, se puede generar interfaces accesibles y provee componentes para la generación de interfaces propios de jsf y componentes creados por otros usuarios como RichFaces, PrimeFaces y IceFaces.

**Tabla IV.XVResumen variables parámetro Seguridad JSF vs STRUTS**

| VARIABLE | STRUTS | JSF |
|---|---|---|
| **Uso de sesiones** | Poco Eficiente | Eficiente |
| **Encriptación de datos** | Poco Eficiente | Eficiente |
| **Validación de datos** | Bueno | Excelente |

**Fuente:** Autoras

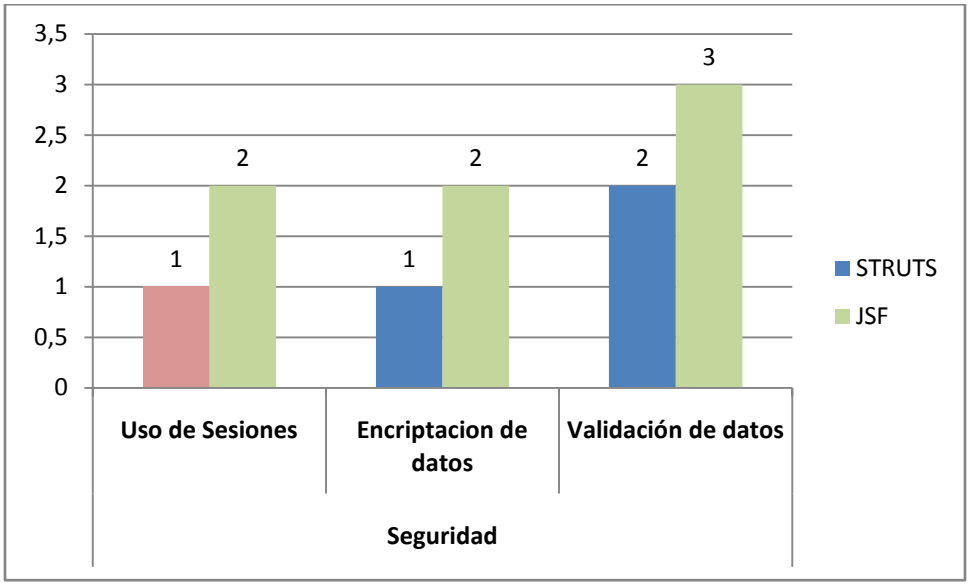



**Figura IV.37Resultado final del parámetro seguridad**

**Fuente:** Autoras

Mediante la tabla y el cuadro estadístico anteriores se puede verificar que el desarrollo de una aplicación web en el framework JSF es más segura que con otros frameworks MVC debido a que realiza un mejor manejo de sesiones, posee mejores mecanismos de seguridad en encriptación de datos y realiza una mejor validación de datos usando validaciones estándar, en el backing bean, en la capa de negocio, y con validators que pueden ser creados por el programador.

De los resultados obtenidos, se puede concluir que luego de haber realizado el análisis comparativo y realizado todos los cálculos correspondientes para la demostración de la hipótesis, y analizando variable por variable del estudio efectuado, se puede concluir que la utilización del framework MVC de Java JSF facilitará el desarrollo de aplicaciones Web

empresariales con mejor rapidez y seguridad. JSF alcanzó un 100% en reducir el desarrollo de la aplicación y en seguridad un 77,78%.

# 5. CAPÍTULO V

# DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE BIENESTAR POLITÉCNICO DE LA ESPOCH

## Introducción

En el presente capítulo se pone en ejecución la parte aplicativa del sistema, los módulos de la solución web para el registro de datos para la Ficha Estudiantil de la ESPOCH, usando los frameworks Spring y JSF.

## 5.1. Microsoft Solution Framework

### 5.1.1. Definición

[3]Microsoft Solution Framework es un marco de trabajo de referencia para construir e implantar sistemas empresariales distribuidos basados en herramientas y tecnologías de Microsoft. MSF comprende un conjunto de modelos, conceptos y guías que contribuyen a alinear los objetivos de negocio y tecnológicos, reducir los costos de la utilización denuevas

[3]http://wiki.monagas.udo.edu.ve/index.php/Microsoft_Solution_Framework_(MSF)

tecnologías, y asegurar el éxito en la implantación de las tecnologías Microsoft. MSF es el resultado de las experiencias de diferentes áreas de Microsoft con proyectos exitosos.

El MSF es un modelo diseñado específicamente para crear productos de buena calidad, donde para cumplir el objetivo prima la comunicación tanto entre el equipo de desarrollo como entre ellos y los clientes

### 5.1.2. Fases

El MSF está compuesto por las siguientes fases:

- Visión
- Planeación
- Desarrollo
- Estabilización
- Instalación
- Soporte


**Figura V.38Fases de MSF**

**Fuente**: http://audiemangt.blogspot.com/2010/05/metodologia-agil-msf-microsoft-solution.html

### 5.1.3. Ventajas

- MSF ayuda a implantar soluciones en base a la tecnología
- Crear una disciplina de análisis de riesgos que ayuda y evoluciona con el proyecto

- Vinculación con el cliente como también orientado al trabajo en equipo
- Tiene facilidad de soporte y mantenimiento
- Es adaptable, se puede utilizar para proyectos de cualquier magnitud
- El modelo tiene facilidad de manejo por ser una empresa conocida
- Aplica mucho e incentiva al trabajo en equipo y a la colaboración
- Cuenta con plantillas que ayudan para el desarrollo de documentos

### 5.1.4. Desventajas

- Al estar basado en tecnologías Microsoft, trata de obligar a usar herramientas de ellos mismo
- Solicita demasiada documentación en sus fases
- Si el análisis de riesgos se hace exhaustivo puede retardar el proyecto
- Los precios de licencias, capacitación y soporte son caros

### 5.1.5. Fase de Visión

En esta fase se debe realizar un estudio de lo que pretendemos en el futuro que haga nuestra aplicación para ello debemos realizar un documento de estrategia y alcance donde debe quedar pactada la necesidad de funcionalidad y servicio que se debe contar en la solución. Para asegurar el éxito del proyecto es importante tener en cuenta el análisis de riesgos y plan de contingencia.

### 5.1.6. Definición del Problema

La falta de automatización de ciertos procesos en la gestión de bienes identificados por el nivel estratégico de la institución, han ocasionado la falta de seguimiento y control de los bienes.

- La falta de información de los recursos disponibles de una obra dificulta el trabajo de asignación de materiales, genera desvíos de los mismos y conflictos al contabilizar
- La falta de control sobre el uso de combustible genera gastos excesivos a la institución
- La contraloría solicito a la institución implementar un mecanismo para tener el historial de cada equipo de cómputo de la Institución por lo que se implementó ese modulo en el sistema

### 5.1.7. Visión del Proyecto

Desarrollar una aplicación para el Departamento de Bienestar Politécnico que permita a laspersonasencargadas contar con la disponibilidad de una Ficha para el Estudiante Politécnico.

Proveer información detallada del estudiante.

Listar los datos del estudiante, de tal manera que sea una herramienta para tomar decisiones.

- **Módulo de Administración**

Este módulo consta del módulo Usuario, que permite crear información del usuario para que pueda interactuar con los datos del Estudiante.

Módulo de Administración de Usuarios

- **Módulo de Seguridad**

El módulo permitirá realizar la autentificación de nuestros a través del user y password, además del tipo de usuario.

- **Módulo de Tipos**

  Módulo Tipo de Colegio

  Módulo Tipo de Discapacidad

  Módulo Tipo de Patrimonio

  Módulo Tipo de Vivienda

- **Módulo de Ubicación**

  Módulo de País

  Módulo de Provincia

  Módulo de Cantón

- **Módulo de Educación**

  Módulo de Facultad

  Módulo de Escuela

  Módulo de Carrera

- **Módulo de Datos Socio-Demográficos**

  Módulo Aspecto

  Módulo Material

  Módulo Localización

- **Módulo de Vivienda**

  Módulo Parroquia

  Módulo Dirección

  Módulo Vivienda

- **Módulo del Estudiante**

  Módulo Estudiante

  Módulo Estudiante – Carrera

  Módulo Ficha

- **Módulo Información Complementaria**

  Módulo Colegio

  Módulo Discapacidad

  Módulo Grupo Familiar

  Módulo Patrimonio

  Módulo Vehículo

## 5.2. Perfil de Usuario

Al contar con la información requerida en el Departamento de Bienestar Politécnico se realizó el análisis para los usuarios que va a contener la aplicación para un mejor funcionamiento: Usuario Administrador y Usuario Estudiante

**Tabla V.XVIPerfil de Usuario**

| Tipos | Acceso | Características |
|---|---|---|
| Usuario Administrador | Web | Encargado de crear usuarios y de interactuar con todos la Ficha Estudiantil<br>Acceso a los reportes |
| Usuario Estudiante | Web | Encargado de llenar todos los datos requeridos para su Ficha Estudiantil |

**Fuente:** Autoras

## 5.3. Ámbito del Proyecto

Desarrollo de una aplicación web para el registro de datos de la Ficha Estudiantil para el Departamento de Bienestar Politécnico de la Escuela Superior Politécnica de Chimborazo

### 5.3.1. Requerimientos Funcionales

Requerimiento 1: Gestionar los datos del usuario

Requerimiento 2: Gestionar los datos del estudiante

Requerimiento 3: Registrar los datos de discapacidad

Requerimiento 4: Registrar los datos de bachillerato

Requerimiento 5: Registrar los datos del nivel superior

Requerimiento 6: Registrar los datos de la información complementaria

Requerimiento 7: Registrar los datos dela Ficha

Requerimiento 8: Validar los diferentes usuarios para el acceso al sistema

Reportes:

Emitir la ficha estudiantil

Emitir listado de estudiantes

Emitir un reporte de estudiantes según su sexo

Emitir un reporte de estudiantes por carrera

Emitir un reporte de estudiantes matriculados por sexo

Emitir un reporte de estudiantes que posean discapacidad por tipo, escuela y quienes son los estudiantes

### 5.3.2. Requerimientos No Funcionales

- Rendimiento
- Disponibilidad
- Seguridad
- Portabilidad
- Mantenibilidad
- Escabilidad
- Reusabilidad
- Interfaces
- Usabilidad

## 5.4. Objetivos del Proyecto

### 5.4.1. Objetivos del Negocio

- Registrar a los diferentes usuarios con su respectiva cuenta
- Llevar el control transparente de los procesos
- Limitar el acceso autorizado solo a personas autorizadas
- Generación de reportes

### 5.4.2. Objetivos del Diseño

- Garantizar un acceso rápido a la aplicación
- Proteger contra el acceso de intrusos mediante la autentificación de usuarios
- Proporcionar una interfaz web amigable y de fácil manejo

## 5.5. Riesgos

El riesgo implica modificaciones que pueden darse por cambios de opinión, de acciones, de lugares, es inevitable e implica incertidumbre, además pérdida de tiempo cuando el riesgo se ha convertido en problema.

Debemos analizar qué riesgos podrían hacer que nuestro proyecto fracasara, y se debe escoger adecuadamente que acciones pueden convertirse en riesgos para de esta manera lograr superarlos.

El propósito esencial de este proceso de análisis de riesgos va enfocado a prevenir muchos acontecimientos que pueden dificultar el desarrollo de la Ficha del Estudiante.

Por lo cual debemos darle una gran importancia a este análisis tomando en cuenta todos los tipos de riesgos que se puede encontrar en el desarrollo del sistema al mismo tiempo gestionar aquellos riesgos con gran probabilidad de impacto.

## 5.6. Identificación del Riesgo

La identificación del riesgo es un intento sistemático para especificar las amenazas al plan de proyecto (estimaciones, planificación temporal, carga de recursos, etc.).

Identificando los riesgos conocidos y predecibles, el gestor del proyecto da un paso adelante para evitarlos cuando sea posible y controlarlos cuando sea necesario.

Existen tres tipos de riesgo:

- Riesgo del proyecto
- Riesgo técnico

- Riesgo del negocio

**Tabla V. XVIIIdentificación de Riesgos**

| Id | Descripción del riesgo | Categoría | Consecuencia |
|---|---|---|---|
| Req1 | Incompleta especificación de los requerimientos | Proyecto | Retraso del proyecto<br>Pérdida de tiempo |
| Req2 | Cambios continuos de los requerimientos por parte del Bienestar Politécnico | Negocio | Diseños realizados sin necesidad<br>Pérdida de recursos económicos |
| Req3 | Falta de conocimiento de los desarrolladores del lenguaje de programación | Técnico | Amenaza la calidad<br>Capacitación a los programadores<br>Retraso del proyecto |
| Req4 | Costo final del proyecto exagerado | Proyecto | Retraso del proyecto<br>Costo del proyecto |
| Req5 | Interfaces inadecuadas para la validación | Técnico | Amenaza la calidad del software<br>Implementación difícil |
| Req6 | Falta de comunicación con el personal del Bienestar Politécnico | Proyecto | Retraso del proyecto |
| Req7 | Información modificada no se actualiza en la base de datos | Negocio | Provoca inconsistencia en los datos |

**Fuente:** Autoras

### 5.6.1. Análisis de Riesgos

**Tabla V.XVIIIValoración de Riesgos**

| Rango de Probabilidad | Descripción | Valor |
|---|---|---|
| 1% - 33% | Baja | 1 |
| 34% - 67% | Media | 2 |
| 68% - 99% | Alta | 3 |

**Fuente:** Autoras

**Tabla V.XIXProbabilidad**

| Identificación | Probabilidad | | |
|---|---|---|---|
| | % | Valor | Probabilidad |
| Req1 | 20 | 1 | Baja |
| Req2 | 20 | 1 | Baja |
| Req3 | 30 | 1 | Baja |
| Req4 | 40 | 2 | Baja |
| Req5 | 20 | 1 | Baja |
| Req6 | 50 | 1 | Media |
| Req7 | 60 | 1 | Media |

**Fuente:** Autoras

## Determinación del Impacto

**Tabla V.XXImpacto del Riesgo**

| Impacto | Retraso | Impacto Técnico | Costo | Valor |
|---|---|---|---|---|
| **Bajo** | 1 semana | Ligero efecto en el desarrollo del proyecto | <1% | 1 |
| **Moderado** | 2 semanas | Moderado efecto en el desarrollo del proyecto | <5% | 2 |
| **Alto** | 1 mes | Severo efecto en el desarrollo del proyecto | <10% | 3 |
| **Crítico** | >1 mes | Proyecto no puede ser culminado | >10% | 4 |

**Fuente:** Autoras

## Impacto - Riesgo

**Tabla V.XXIImpacto - Riesgo**

| Identificación | Impacto | |
|---|---|---|
| | **Valor** | **Impacto** |
| Req1 | 3 | Alto |
| Req2 | 3 | Alto |
| Req3 | 1 | Bajo |
| Req4 | 3 | Alto |
| Req5 | 1 | Bajo |
| Req6 | 2 | Media |
| Req7 | 3 | Media |

**Fuente:** Autoras

**Determinación de la exposición al Riesgo**

**Tabla V.XXIIIImpacto - Probabilidad**

| Exposición al riesgo | Valor | Color |
|---|---|---|
| Baja | 1 o 2 | |
| Media | 3 o 4 | |
| Alta | >6 | |

**Fuente:** Autoras

| Impacto / Probabilidad | Baja=1 | Mediado=2 | Alto=3 | Crítico=4 |
|---|---|---|---|---|
| **Alta=3** | 3 | 6 | 9 | 12 |
| **Media=2** | 2 | 4 | 6 | 8 |
| **Baja=1** | 1 | 2 | 3 | 4 |

**Fuente:** Autoras

A continuación en base a los valores de riesgo y a la línea de corte se determina los Riesgos más importantes a tomar en cuenta para poder tomar una solución oportuna.

**Tabla V.XXIIIRiesgo**

| Id | Probabilidad | | | Impacto | | Exposición al Riesgo | |
|---|---|---|---|---|---|---|---|
| | % | Valor | Probabilidad | Valor | Impacto | Valor | Exposición |
| Req1 | 20 | 1 | Baja | 3 | Alto | 3 | Media |
| Req2 | 20 | 1 | Baja | 3 | Alto | 3 | Media |
| Req3 | 30 | 1 | Baja | 1 | Bajo | 1 | Baja |
| Req4 | 40 | 2 | Media | 3 | Alto | 6 | Alta |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Req5 | 20 | 1 | Baja | 1 | Bajo | 1 | Baja |
| Req6 | 50 | 1 | Media | 2 | Moderado | 2 | Baja |
| Req7 | 60 | 2 | Media | 3 | Alto | 6 | Media |

**Fuente:** Autoras

**Tabla V.XXIVPrioridades del Riesgo**

| Id | Prioridad | Exposición |
|---|---|---|
| Req4 | 1 | 6 |
| Req1 | 2 | 3 |
| Req2 | 2 | 3 |
| Req7 | 2 | 3 |
| Req3 | 4 | 1 |
| Req5 | 4 | 1 |
| Req6 | 3 | 2 |

**Fuente:** Autoras

### 5.6.2. Planeación y Programación del Riesgo

Para mitigar el riesgo se utiliza la Hoja de Gestión de Riesgo

**TablaV. XXVHoja de Gestión de Riesgo 1**

<table>
<tr><th colspan="4">HOJA DE GESTIÓN DEL RIESGO</th></tr>
<tr><td colspan="2">ID. DEL RIESGO: Requerimiento 1</td><td colspan="2">FECHA: 27/03/2013</td></tr>
<tr><td>Probabilidad: Baja<br>Valor: 1</td><td>Impacto: Alto<br>Valor: 3</td><td>Exposición: Alta<br>Valor: 3</td><td>Prioridad: 1<br>Valor:</td></tr>
<tr><td colspan="4">DESCRIPCIÓN: Especificación incompleta de requerimientos</td></tr>
<tr><td colspan="4">REFINAMIENTO:<br>Causas:<br>Mala información de requerimientos por parte de los usuarios<br>Mala interpretación de los requerimientos por parte de los desarrolladores<br><br>Consecuencias:<br>Retardo del proyecto<br>Incremento del costo del proyecto</td></tr>
<tr><td colspan="4">REDUCCIÓN:<br>• Mejorar la comunicación entre cliente y desarrolladores<br>• Evitar los frecuentes cambios de requerimientos</td></tr>
<tr><td colspan="4">SUPERVISIÓN:<br><br>• Supervisar el ambiente de trabajo de los miembros del equipo de trabajo y cliente<br>• Acuerdo entre el cliente y responsables sobre sus necesidades</td></tr>
<tr><td colspan="4">GESTIÓN<br>• Realizar los trabajos manteniendo la integridad y ética profesional<br>• Mutuo acuerdo para establecer los requerimientos</td></tr>
<tr><td colspan="4">ESTADO ACTUAL:<br><br>Frase de reducción iniciada [x]<br>Fase de supervisión iniciada [ ]<br>Gestionando el riesgo [ ]</td></tr>
<tr><td colspan="4">RESPONSABLES:<br><br>Tania Aguirre - Andrea Moncayo</td></tr>
</table>

**Fuente:** Autoras

**Tabla V.XXVIHoja de Gestión de Riesgo 2**

<table>
<tr><th colspan="4">HOJA DE GESTIÓN DEL RIESGO</th></tr>
<tr><td colspan="2">ID. DEL RIESGO: Requerimiento 2</td><td colspan="2">FECHA: 27/03/2013</td></tr>
<tr><td>Probabilidad: Baja<br>Valor: 1</td><td>Impacto: Alto<br>Valor: 3</td><td>Exposición: Alta<br>Valor: 3</td><td>Prioridad: 1<br>Valor:</td></tr>
<tr><td colspan="4">DESCRIPCIÓN: Cambios continuos de los requerimientos por parte del Bienestar Politécnico</td></tr>
<tr><td colspan="4">REFINAMIENTO:<br>Causas:<br>Falta de análisis previo de los requerimientos por parte del cliente<br><br>Consecuencias:<br>Pérdida de recursos<br>No complacer las expectativas para el sistema</td></tr>
<tr><td colspan="4">REDUCCIÓN:<br>• Asignación de recursos para una correcta captación de los requerimientos</td></tr>
<tr><td colspan="4">SUPERVISIÓN:<br><br>• Seguimiento de las funcionalidades para los requerimientos planteados</td></tr>
<tr><td colspan="4">GESTIÓN<br>• Desplegar los beneficios de un requerimiento gestionado</td></tr>
<tr><td colspan="4">ESTADO ACTUAL:<br><br>Frase de reducción iniciada [X]<br>Fase de supervisión iniciada [ ]<br>Gestionando el riesgo [ ]</td></tr>
<tr><td colspan="4">RESPONSABLES:<br><br>Tania Aguirre - Andrea Moncayo</td></tr>
</table>

**Fuente:** Autoras

**Tabla V.XXVIIIHoja de Gestión de Riesgo 3**

<table>
<tr><th colspan="4">HOJA DE GESTIÓN DEL RIESGO</th></tr>
<tr><td colspan="2">ID. DEL RIESGO: Requerimiento 3</td><td colspan="2">FECHA: 27/03/2013</td></tr>
<tr><td>Probabilidad: Baja<br>Valor: 1</td><td>Impacto: Bajo<br>Valor: 1</td><td>Exposición: Baja<br>Valor: 1</td><td>Prioridad: 4<br>Valor:</td></tr>
<tr><td colspan="4">DESCRIPCIÓN: Falta de conocimiento de los desarrolladores del lenguaje de programación</td></tr>
<tr><td colspan="4">REFINAMIENTO:<br>Causas:<br>Especialización en diferentes lenguajes de programación y desarrollo de software<br>Falta de investigación acerca del lenguaje de programación<br>Consecuencias:<br>Capacitación a los programadores<br>Retraso del proyecto</td></tr>
<tr><td colspan="4">REDUCCIÓN:<br>• Capacitación a los programadores dependiendo del lenguaje seleccionado<br>• Promover investigación en nuevos lenguajes</td></tr>
<tr><td colspan="4">SUPERVISIÓN:<br><br>• Controlar la adecuada capacitación a los programadores</td></tr>
<tr><td colspan="4">GESTIÓN<br>• Fomentar la especialización en el lenguaje de programación</td></tr>
<tr><td colspan="4">ESTADO ACTUAL:<br><br>Frase de reducción iniciada [x]<br>Fase de supervisión iniciada [ ]<br>Gestionando el riesgo [ ]</td></tr>
<tr><td colspan="4">RESPONSABLES:<br><br>Tania Aguirre - Andrea Moncayo</td></tr>
</table>

**Fuente:** Autoras

**Tabla V.XXVIIIHoja de Gestión de Riesgo 4**

<table>
<tr><th colspan="4">HOJA DE GESTIÓN DEL RIESGO</th></tr>
<tr><td colspan="2">ID. DEL RIESGO: Requerimiento 4</td><td colspan="2">FECHA: 27/03/2013</td></tr>
<tr><td>Probabilidad: Media<br>Valor: 2</td><td>Impacto: Alto<br>Valor: 3</td><td>Exposición: Alta<br>Valor: 6</td><td>Prioridad: 1<br>Valor:</td></tr>
<tr><td colspan="4">DESCRIPCIÓN: Costo final del proyecto exagerado</td></tr>
<tr><td colspan="4">REFINAMIENTO:<br>Causas:<br>Falta de una planificación adecuada para el presupuesto del desarrollo del proyecto<br>La culminación de cada etapa no fue de acuerdo a lo programación<br>Consecuencias:<br>Retraso en la planificación<br>Aumento al costo final del proyecto</td></tr>
<tr><td colspan="4">REDUCCIÓN:<br>• Realizar una correcta planificación del presupuesto<br>• Culminar las etapas asignadas según la programación</td></tr>
<tr><td colspan="4">SUPERVISIÓN:<br>• Controlar la planificación presupuestaria tomando en cuenta posibles inconvenientes</td></tr>
<tr><td colspan="4">GESTIÓN<br>• Al culminar obtener un proyecto eficiente</td></tr>
<tr><td colspan="4">ESTADO ACTUAL:<br>Frase de reducción iniciada [x]<br>Fase de supervisión iniciada [ ]<br>Gestionando el riesgo [ ]</td></tr>
<tr><td colspan="4">RESPONSABLES:<br>Tania Aguirre - Andrea Moncayo</td></tr>
</table>

**Fuente:** Autoras

**Tabla V.XXIXHoja de Gestión de Riesgo 5**

<table>
<tr><th colspan="4">HOJA DE GESTIÓN DEL RIESGO</th></tr>
<tr><td colspan="2">ID. DEL RIESGO: Requerimiento 5</td><td colspan="2">FECHA: 27/03/2013</td></tr>
<tr><td>Probabilidad: Baja<br>Valor: 1</td><td>Impacto: Bajo<br>Valor: 1</td><td>Exposición: Baja<br>Valor: 2</td><td>Prioridad: 4<br>Valor:</td></tr>
<tr><td colspan="4">DESCRIPCIÓN: Interfaces inadecuadas para la validación</td></tr>
<tr><td colspan="4">REFINAMIENTO:<br>Causas:<br>El diseño de las interfaces no contienen los datos necesarios para validar el proyecto<br>Consecuencias:<br>Amena de calidad del proyecto<br>Dificultad al manejar el proyecto</td></tr>
<tr><td colspan="4">REDUCCIÓN:<br>• Diseñar correctamente las interfaces para validar el acceso al proyecto</td></tr>
<tr><td colspan="4">SUPERVISIÓN:<br><br>• Revisar durante el diseño de las interfaces si son los datos necesarios</td></tr>
<tr><td colspan="4">GESTIÓN<br>• Requerir al cliente la aprobación de las interfaces</td></tr>
<tr><td colspan="4">ESTADO ACTUAL:<br><br>Frase de reducción iniciada [x]<br>Fase de supervisión iniciada [ ]<br>Gestionando el riesgo [ ]</td></tr>
<tr><td colspan="4">RESPONSABLES:<br><br>Tania Aguirre - Andrea Moncayo</td></tr>
</table>

**Fuente:** Autoras

**Tabla V.XXXHoja de Gestión de Riesgo 6**

| **HOJA DE GESTIÓN DEL RIESGO** | | | |
|---|---|---|---|
| **ID. DEL RIESGO:** Requerimiento 6 | | **FECHA:** 27/03/2013 | |
| **Probabilidad:** Baja<br>**Valor:** 1 | **Impacto:** Moderado<br>**Valor:** 1 | **Exposición:** Baja<br>**Valor:** 2 | **Prioridad:** 3<br>**Valor:** |

**REFINAMIENTO:**
**Causas:**
Ambiente de trabajo muy tenso
No exista la suficiente confianza
**Consecuencias:**
Retraso en el proyecto

**REDUCCIÓN:**
- Fomentar la unión para un correcto desarrollo del proyecto

**SUPERVISIÓN:**

- Actitud positiva de todo el personal en las actividades programadas

**GESTIÓN**
- El personal entienda la importancia del proyecto para obtener la cooperación
- Preparación a todo el equipo involucrado

**ESTADO ACTUAL:**

| | |
|---|---|
| Frase de reducción iniciada | X |
| Fase de supervisión iniciada | |
| Gestionando el riesgo | |

**RESPONSABLES:**

Tania Aguirre - Andrea Moncayo

**Fuente:** Autoras

**Tabla V.XXXIHoja de Gestión de Riesgo 7**

<table>
<tr><th colspan="4">HOJA DE GESTIÓN DEL RIESGO</th></tr>
<tr><td colspan="2">ID. DEL RIESGO: Requerimiento 7</td><td colspan="2">FECHA: 27/03/2013</td></tr>
<tr><td>Probabilidad: Baja<br>Valor: 1</td><td>Impacto: Moderado<br>Valor: 1</td><td>Exposición: Baja<br>Valor: 2</td><td>Prioridad: 3<br>Valor:</td></tr>
<tr><td colspan="4">DESCRIPCIÓN: Información modificada no se actualiza en la base de datos</td></tr>
<tr><td colspan="4">REFINAMIENTO:<br>Causas:<br>No se cuente con un método adecuado para facilitar las actualizaciones automáticas<br>Consecuencias:<br>Inconsistencia en los datos</td></tr>
<tr><td colspan="4">REDUCCIÓN:<br>• Validar continuamente de los datos y todos sus casos posibles</td></tr>
<tr><td colspan="4">SUPERVISIÓN:<br><br>• Controlar que los métodos estén acorde al modelo de la base de datos</td></tr>
<tr><td colspan="4">GESTIÓN<br>• Revisar que el método cumpla con la función por la que se implementó</td></tr>
<tr><td colspan="4">ESTADO ACTUAL:<br><br>Frase de reducción iniciada [X]<br>Fase de supervisión iniciada [ ]<br>Gestionando el riesgo [ ]</td></tr>
<tr><td colspan="4">RESPONSABLES:<br><br>Tania Aguirre - Andrea Moncayo</td></tr>
</table>

**Fuente:** Autoras

## 5.7. Planificación Inicial

### 5.7.1. Factibilidad

**Factibilidad Técnica**

La factibilidad técnica ayuda a determinar si la propuesta puede ser implementada con el hardware, software y recurso humano disponible.

Para el desarrollo de la aplicación web "Ficha Estudiantil" para el Departamento de Bienestar Politécnico de la ESPOCH, cuenta con los recursos necesarios tanto en la parte hardware y software. A continuación se detalla los recursos existentes

**Hardware Existente**

Hardware con el que cuenta para el desarrollo de la aplicación:

**Tabla V.XXXIIIHardware Existente**

| Cantidad | Descripción | Observación |
|---|---|---|
| 1 | Computadora | Desarrollo de la aplicación y la documentación |
| 1 | Infraestructura de red | Acceder a internet para consultas en el desarrollo de la aplicación y realizar las correspondientes pruebas |

**Fuente:** Autoras

## Hardware Requerido

**Tabla V.XXXIIIHardware Existente**

| Cantidad | Descripción | Observación |
|---|---|---|
| 1 | Servidor | Realizar pruebas de la configuración del servidor |
| 1 | Impresora | Impresión de reportes |

**Fuente:** Autoras

## Software Existente

**Tabla V.XXXIVSoftware Existente**

| Nombre | Descripción |
|---|---|
| Windows 7 Profesional | Sistema Operativo |
| Microsoft Visio 2013 | Herramienta para diseño UML |

**Fuente:** Autoras

## Software Requerido

**Tabla V.XXXVSoftware Existente**

| Nombre | Descripción |
|---|---|
| NetBeans IDE 7.2.1 | Entorno de desarrollo |
| Ireport 4.7.1 | Reportero |
| Apache Tomcat | Servidor de aplicaciones web |
| Netsparker Community | Analiza vulnerabilidades existentes en un sistema web |
| Practiline Source | Cuenta las líneas del código |

**Fuente:** Autoras

## Recurso Humano Requerido

**Tabla V.XXXVIRecurso Humano Requerido**

| Función | Formación |
|---|---|
| Jefe de Proyecto | Ingeniería en Sistemas |
| Equipo de desarrolladores | Estudiante de Ingeniería en Sistemas |
| Administrador de Base de Datos | Ingeniería en Sistemas |
| Diseñador | Estudiante de Ingeniería en Sistemas |

**Fuente:** Autoras

## Facilidad Operativa

### Recurso Humano

El recurso humano que participará del sistema son:

- Usuarios directos

  Los usuarios directos a capacitar para el manejo del sistema son:

#### Personal a Capacitar

**Tabla V.XXXVIIPersonal a Capacitar**

| Nombre | Función |
|---|---|
| U. Administrador | Directora del Departamento de Bienestar Politécnico de la ESPOCH |
| U. Estudiante | Todos los estudiantes de la ESPOCH |

**Fuente:** Autoras

## Factibilidad Económica

El tiempo de duración de este proyecto será de 12 meses

## Costos

**Tabla V.XXXVIIICosto de Desarrollo**

| | | |
|---|---|---|
| **Costos de desarrollo** | | |
| **Costo personal (mensual)** | | |
| Jefe de proyecto y desarrollador | $800 | $9.600 |
| Administrador de BD y diseñador | $160 | $3.200 |
| | **Total Costo Personal** | **$12.800** |
| **Costo de hardware y software** | | |
| **Costos Hardware** | | |
| Computadoras | $1200 | $2400 |
| | **Total Costo Hardware** | **$2400** |
| **Costo Software** | | |
| Internet | $400 | |
| | **Total Costo Software** | **$400** |
| **Costos varios** | | |
| Suministros | $300 | |
| Alimentación | $800 | |
| Papel A4 | $10 | |
| | **Total Costos Varios** | **$1.210** |
| **COSTO TOTAL DE DESARROLLO** | | **$16810** |

**Fuente:** Autoras

**Análisis Costo Beneficio**

Los beneficios que se podrá obtener con la utilización de este sistema son los siguientes:

Permitirá realizar el proceso de gestión de bienes (materiales) existentes para la obras de una forma eficiente y real, permitiendo organizar, reducir el flujo de trabajo de los departamentos, además usar la tecnología existen en la institución y reducir la contaminación con el uso de papel, tinta, cartuchos y control del uso de los materiales destinado a los lugares para los que fueron adquiridos.

Se podrá realizar un mayor control del uso de los combustibles y de los bienes que existen en la institución.

Mediante el sistema se puede generar reportes que ayudarán a la toma de decisiones de las autoridades.

Se optimizará el tiempo en la elaboración de informes.

La utilización del sistema será fácil de utilizar y además los usuarios pueden ingresar desde cualquier lugar.

## 5.8. Fase de Planificación

Obtener un cronograma Obtener un cronograma de trabajo que cumpla con lo especificado en la fase de visión, desarrollar los requerimientos funcionales y no funcionales, describir el escenario, diagramar casos de uso

**PLANEACIÓN**

En esta fase se realiza la preparación de la especificación funcional, diseño conceptual.

### 5.8.1. Diseño Conceptual

#### 5.8.1.1. Requerimientos Funcionales

A continuación se describe cada uno de los requerimientos funcionales

##### 5.8.1.1.1. Requerimiento Funcional 1

- **Introducción**

  El sistema deberá permitir gestionar los datos del usuario

- **Entrada**

  **Fuentes de entradas:** User y Password

  **Frecuencia:** Bajo demanda

  **Requisitos de control:** Controla que los campos del formulario estén llenos y que los datos sean los correctos

  **Entradas válidas:** Todos los campos sean válidos

- **Procesos:**

  1. El usuario deberá ingresar su user y password para la autenticación.

  2. El sistema validarála información

  3. Si el dato es verdadero

     a. Ingresará al sistema

4. Caso Contrario

a. Mensajes de error

b. Ingreso

c. Aceptar

5. Validación de datos

a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

**Destino de las salidas**

Muestra listado con los datos ingresados cuando son correctos

**Interfaces de hardware**

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

**Interfaces de Software**

- La herramienta de desarrollo que se utilizará será NetBeans IDE 7.2.1 para que la aplicación cumpla los requerimientos.
- El repositorio de datos será PostgreSQL 8.4.

**Interfaces de Software**

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

### 5.8.1.1.2. Requerimiento Funcional 2

- **Introducción**

  El sistema deberá permitir gestionar los datos del estudiante

- **Entrada**

  **Fuentes de entradas:** User (cédula) y password (cédula)

  **Frecuencia:** Bajo demanda

  **Requisitos de control:** Controla que los campos del formulario estén llenos y que los datos sean los correctos

  **Entradas válidas:** Todos los campos sean válidos

- **Procesos:**

  1. El estudiante deberá ingresar su cédula y password (cédula) para la autenticación.

  2. El sistema validarála información

  3. Si el dato es verdadero

     a. Ingresará al sistema

  4. Caso Contrario

a. Mensajes de error

b. Ingreso

c. Aceptar

5. Validación de datos

a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

**Destino de las salidas**

Muestra listado con los datos ingresados cuando son correctos

**Interfaces de hardware**

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

**Interfaces de Software**

- La herramienta de desarrollo que se utilizará será NetBeans IDE 7.2.1 para que la aplicación cumpla los requerimientos.
- El repositorio de datos será PostgreSQL 8.4

**Interfaces de Software**

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

#### 5.8.1.1.3. Requerimiento Funcional 3

- **Introducción**

  El sistema deberá permitir registrar los datos de discapacidad

- **Entrada**

  **Fuentes de entradas:** User y password

  **Frecuencia:** Bajo demanda

  **Requisitos de control:** Controla que los campos del formulario estén llenos y que los datos sean los correctos

  **Entradas válidas:** Todos los campos sean válidos

- **Procesos:**

  1. El estudiante deberá ingresar su cédula y password (cédula) para la autenticación.

  2. El sistema validarála información

  3. Si el dato es verdadero

     a. Ingresará al sistema

  4. Caso Contrario

a. Mensajes de error

b. Ingreso

c. Aceptar

5. Validación de datos

a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

**Destino de las salidas**

Muestra listado con los datos ingresados cuando son correctos

**Interfaces de hardware**

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

**Interfaces de Software**

- La herramienta de desarrollo que se utilizará será NetBeans IDE 7.2.1 para que la aplicación cumpla los requerimientos.
- El repositorio de datos será PostgreSQL 8.4

**Interfaces de Software**

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

#### 5.8.1.1.4. Requerimiento Funcional 4

- **Introducción**

  El sistema deberá permitir registrar los datos de bachillerato

- **Entrada**

  **Fuentes de entradas:** User y password

  **Frecuencia:** Bajo demanda

  **Requisitos de control:** Controla que los campos del formulario estén llenos y que los datos sean los correctos

  **Entradas válidas:** Todos los campos sean válidos

- **Procesos:**

  1. El estudiante deberá ingresar su cédula y password (cédula) para la autenticación.

  2. El sistema validarála información

  3. Si el dato es verdadero

     a. Ingresará al sistema

  4. Caso Contrario

a. Mensajes de error

b. Ingreso

c. Aceptar

5. Validación de datos

a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

**Destino de las salidas**

Muestra listado con los datos ingresados cuando son correctos

**Interfaces de hardware**

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

**Interfaces de Software**

- La herramienta de desarrollo que se utilizará será NetBeans IDE 7.2.1 para que la aplicación cumpla los requerimientos.
- El repositorio de datos será PostgreSQL 8.4.

**Interfaces de Software**

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

### 5.8.1.1.5. Requerimiento Funcional 5

- **Introducción**

  El sistema deberá permitir registrar los datos del nivel superior

- **Entrada**

  **Fuentes de entradas:** User y password

  **Frecuencia:** Bajo demanda

  **Requisitos de control:** Controla que los campos del formulario estén llenos y que los datos sean los correctos

  **Entradas válidas:** Todos los campos sean válidos

- **Procesos:**

  1. El estudiante deberá ingresar su cédula y password (cédula) para la autenticación.

  2. El sistema validarála información

  3. Si el dato es verdadero

     a. Ingresará al sistema

  4. Caso Contrario

a. Mensajes de error

b. Ingreso

c. Aceptar

5. Validación de datos

a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

**Destino de las salidas**

Muestra listado con los datos ingresados cuando son correctos

**Interfaces de hardware**

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

**Interfaces de Software**

- La herramienta de desarrollo que se utilizará será NetBeans IDE 7.2.1 para que la aplicación cumpla los requerimientos.
- El repositorio de datos será PostgreSQL 8.4.

**Interfaces de Software**

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

#### 5.8.1.1.6. Requerimiento Funcional 6

- **Introducción**

  El sistema deberá permitir registrar los datos de la información complementaria (grupo familiar, patrimonio, vehículo)

- **Entrada**

  **Fuentes de entradas:** User y password

  **Frecuencia:** Bajo demanda

  **Requisitos de control:** Controla que los campos del formulario estén llenos y que los datos sean los correctos

  **Entradas válidas:** Todos los campos sean válidos

- **Procesos:**

  1. El estudiante deberá ingresar su cédula y password (cédula) para la autenticación.

  2. El sistema validarála información

  3. Si el dato es verdadero

     a. Ingresará al sistema

  4. Caso Contrario

a. Mensajes de error

b. Ingreso

c. Aceptar

5. Validación de datos

a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

**Destino de las salidas**

Muestra listado con los datos ingresados cuando son correctos

**Interfaces de hardware**

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

**Interfaces de Software**

- La herramienta de desarrollo que se utilizará será NetBeans IDE 7.2.1 para que la aplicación cumpla los requerimientos.
- El repositorio de datos será PostgreSQL 8.4.

**Interfaces de Software**

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

### 5.8.1.1.7. Requerimiento Funcional 7

- **Introducción**

  El sistema deberá permitir registrar los datos de la Ficha

- **Entrada**

  **Fuentes de entradas:** User y password

  **Frecuencia:** Bajo demanda

  **Requisitos de control:** Controla que los campos del formulario estén llenos y que los datos sean los correctos

  **Entradas válidas:** Todos los campos sean válidos

- **Procesos:**

  1. El estudiante deberá ingresar su cédula y password (cédula) para la autenticación.

  2. El sistema validarála información

  3. Si el dato es verdadero

     a. Ingresará al sistema

  4. Caso Contrario

a. Mensajes de error

b. Ingreso

c. Aceptar

5. Validación de datos

a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

**Destino de las salidas**

Muestra listado con los datos ingresados cuando son correctos

**Interfaces de hardware**

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

**Interfaces de Software**

- La herramienta de desarrollo que se utilizará será NetBeans IDE 7.2.1 para que la aplicación cumpla los requerimientos.
- El repositorio de datos será PostgreSQL 8.4.

**Interfaces de Software**

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

### 5.8.1.2. Requerimientos no Funcionales

Requerimientos no funcionales con sus perspectivas

- **Rendimiento**

  Tiempos de respuesta para acceder a la página de autentificación del sistema máximo de 30 segundos

- **Disponibilidad**

  El sistema estará fuera de línea máximo 30 minutos. Empleo de sistemas de respaldo.

- **Seguridad**

  Para ingresar al sistema primero deberá el usuario autentificarse dependiendo su tipo de usuario asignado por el administrador para acceder al ambiente de trabajo

- **Portabilidad**

  Garantizar compatibilidad con los otros sistemas operativos: Windows XP, Windows 7, Linux

- **Mantenibilidad**

  Documentación del diseño y la codificación de la aplicación

- **Escabilidad**

  Diseño de la arquitectura empleando módulos independientes

- **Interfaces**

  Interfaces realizadas en NetBeans 7.2.1

- **Usabilidad**

  Factibilidad de uso

#### 5.8.1.3. Actores

**Usuario Administrador**

Es la persona que posee el control de todo el sistema, y es el encargado de gestionar los datos de usuarios y la ficha estudiantil, además podrá asignar a las personas responsables para acceder a la ficha estudiantil.

El usuario administrador tiene el control total es decir, permisos de lectura, escritura y la gestión de reportes (visualizar e imprimir)

**Usuario Estudiante**

Este es un tipo de usuario con limitaciones al momento de acceder ya que solo debe ingresar sus datos en la ficha correspondiente.

##### 5.8.1.3.1. Casos de Uso

**Caso de Uso - Administrador**

El usuario administrador tendrá control total de la aplicación

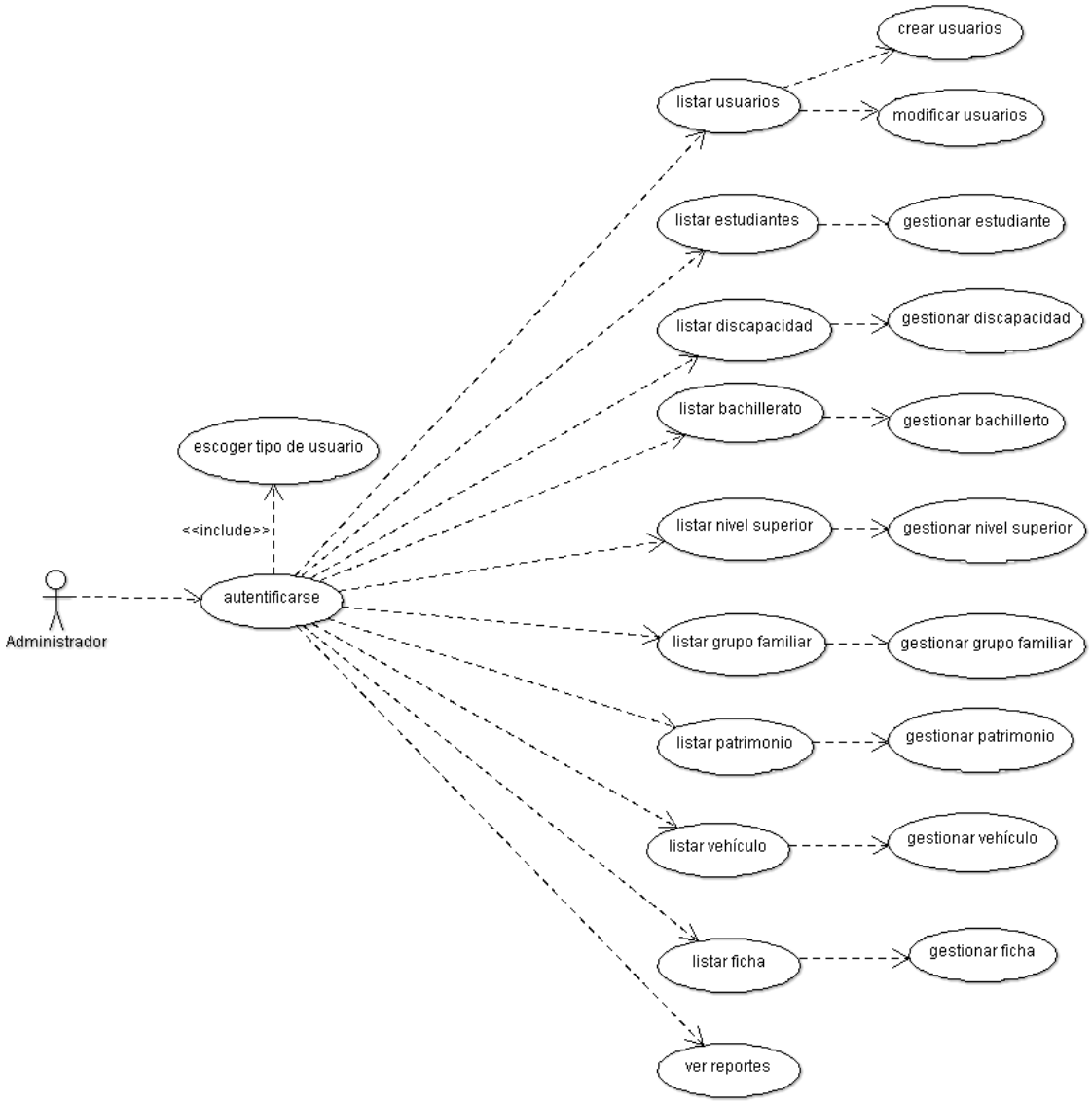



**Figura V.39Caso de Uso - Administrador**

**Fuente:** Autoras

## Caso de Uso – Estudiante

El usuario estudiante deberá ingresar sus datos a la aplicación

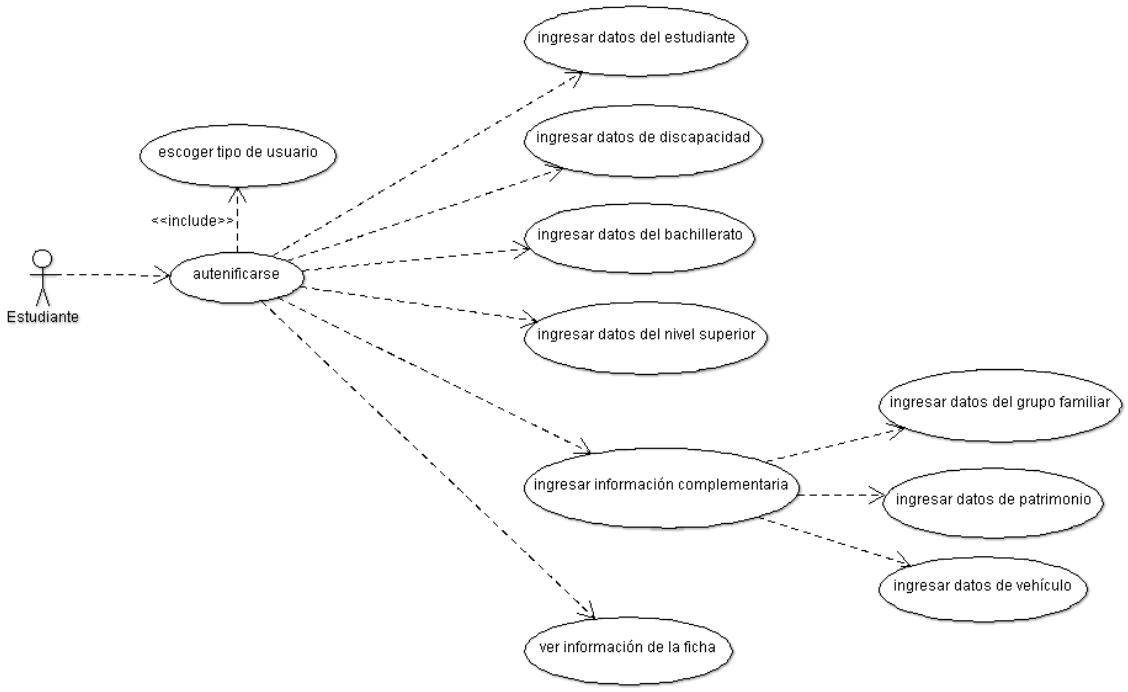



**Figura V.40Caso de Uso – Estudiante**

**Fuente:** Autoras

### 5.8.1.3.2. Escenarios

## Autentificación Usuario Administrador

- **Personas Alternativas**

**Tabla V.XXXIXPersonas Alternativas para el Administrador**

| Persona | Desviaciones (si es aplicable) |
|---|---|
| Persona delegada por el representante del Departamento de Bienestar Politécnico | Ninguno |

**Fuente:** Autoras

- **Descripción del Escenario**

  El administrador ingresará su usuario, password y tipo en la interfaz de autentificación. Podrá crear usuarios e interactuar con la ficha estudiantil

**Autentificación Usuario Estudiante**

- **Personas Alternativas**

**Tabla V.XLPersonas Alternativas para el Estudiante**

| Persona | Desviaciones (si es aplicable) |
|---|---|
| Estudiante | Ninguno |

**Fuente:** Autoras

- **Descripción del Escenario**

El estudiante ingresará su usuario, password y tipo en la interfaz de autentificación, si sus datos son correctos podrá ingresar los datos de su ficha estudiantil

### 5.8.1.3.3. Glosario de Términos

**Tabla V.XLIGlosario de Términos**

| Término | Descripción |
|---|---|
| MSF | Microsoft Solution Framework, metodología empleada para el desarrollo de la aplicación |
| UML | Lenguaje Unificado de Modelado, es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema |
| Módulos | Distintas dependencias del sistema que brinda información del estudiante |

| | |
|---|---|
| Hardware | Son las características físicas y tangibles del computador utilizado |
| Software | Es la parte intangible del computador |
| Password | Es una serie secreta de caracteres que permite al usuario tener acceso al computador, programa o sitio |
| Rendimiento | Capacidad de evolución y aprovechamiento del sistema |
| Servidor | Es un ordenador de gran potencia, que se encarga de "prestar un servicio" a otros ordenadores que se conectan a él |
| Sistema Operativo | Es un programa informático que actúa de interfaz entre los dispositivos de hardware y el usuario. |
| Base de Datos | Conjunto de datos organizados para su almacenamiento en la memoria del computador, diseñado para facilitar su mantenimiento y acceso de una estándar |
| Usuario | Persona que utiliza el sistema |
| Administrador | Es la persona encargada de administrar el sistema en su totalidad, además puede generar reportes |
| Reporte | Documento que se utilizará para visualizar información de acuerdo a lo necesitado |
| Metodología | Conjunto de pasos lógicos y ordenados para realizar una actividad |
| Internet | Conjunto descentralizado en redes de comunicación interconectadas que utilizan TCP/IP |

**Fuente:** Autoras

### 5.8.1.3.4. Refinar los Casos de Uso

**Tabla V.XLIIAutentificación del Usuario Administrador**

| | |
|---|---|
| **Identificar Caso de Uso** | CU- Administrador |
| **Nombre del Caso de Uso** | Realizar administración del sistema |
| **Actores** | Responsables del departamento |
| **Propósito** | Crear el correcto ambiente de trabajo |
| **Visión General** | El responsable para realizar los procesos deberá autentificarse con su user y password |
| **Tipo** | Primario, real y extendido |
| **Referencias** | Caso de Uso: Administrador |
| **Curso Típico de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El Administrador empieza creando el ambiente de trabajo | 2. El sistema cuenta con un control para la autentificación |
| 3. Autentificación, ingresar el user y password | 4. Valida los datos ingresados |
| | 5. Al autentificarse correctamente podrá ingresar a los procesos correspondientes |
| **Cursos Alternativos** | |
| 2.1 El sistema no presenta pantallas para el ingreso de datos | |
| Control de campos vacíos | |

| |
|---|
| 4.1 Ingreso del Usuario<br><br>Cuando no se ingresa algún campo obligatorio le informa con un error para llenar<br><br>4.2 Si el usuario no está en la base de datos, el sistema informará que no existe |

**Fuente:** Autoras

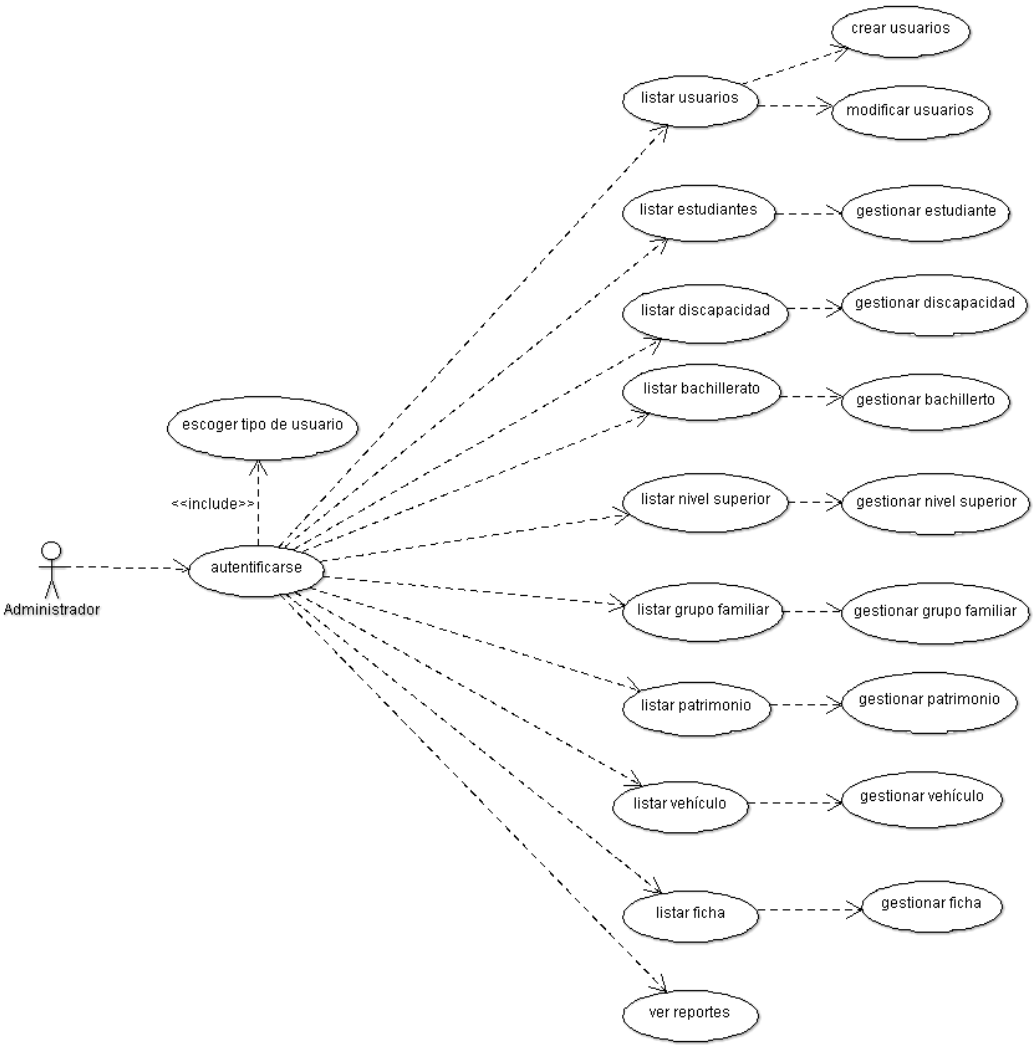



**Figura V.41Caso de Uso Refinado - Administrador**

**Fuente:** Autoras

**Tabla V.XLIIIAutentificación del Usuario Estudiante**

| | |
|---|---|
| **Identificar Caso de Uso** | CU- Estudiante |
| **Nombre del Caso de Uso** | Ingresar datos al sistema |

| | |
|---|---|
| **Actores** | Estudiantes |
| **Propósito** | Obtener los datos de la ficha estudiantil |
| **Visión General** | El estudiante deberá autentificarse con su user y password, luego ingresará sus respectivos datos |
| **Tipo** | Primario, real y extendido |
| **Referencias** | Caso de Uso: Estudiante |
| **Curso Típico de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El Administrador empieza creando el ambiente de trabajo | 2. El sistema cuenta con un control para la autentificación |
| 3. Autentificación, ingresar el user y password | 4. Valida los datos ingresados |
| | 5. Al autentificarse correctamente podrá ingresar a los procesos correspondientes |
| | 6. Presenta la interfaz en la que el estudiante ingresará los datos solicitados |
| **Cursos Alternativos** | |
| 2.1 El sistema no presenta pantallas para el ingreso de datos | |
| Control de campos vacíos<br>4.1 Ingreso del Estudiante<br>Cuando no se ingresa algún campo obligatorio le informa con un error para llenar | |

| |
|---|
| 4.2 Si el usuario no está en la base de datos, el sistema informará que no existe<br><br>6.1 Ingreso datos de la Ficha Estudiantil<br><br>Cuando no se ingresa algún campo obligatorio le informa con un error para llenar |

**Fuente:** Autoras

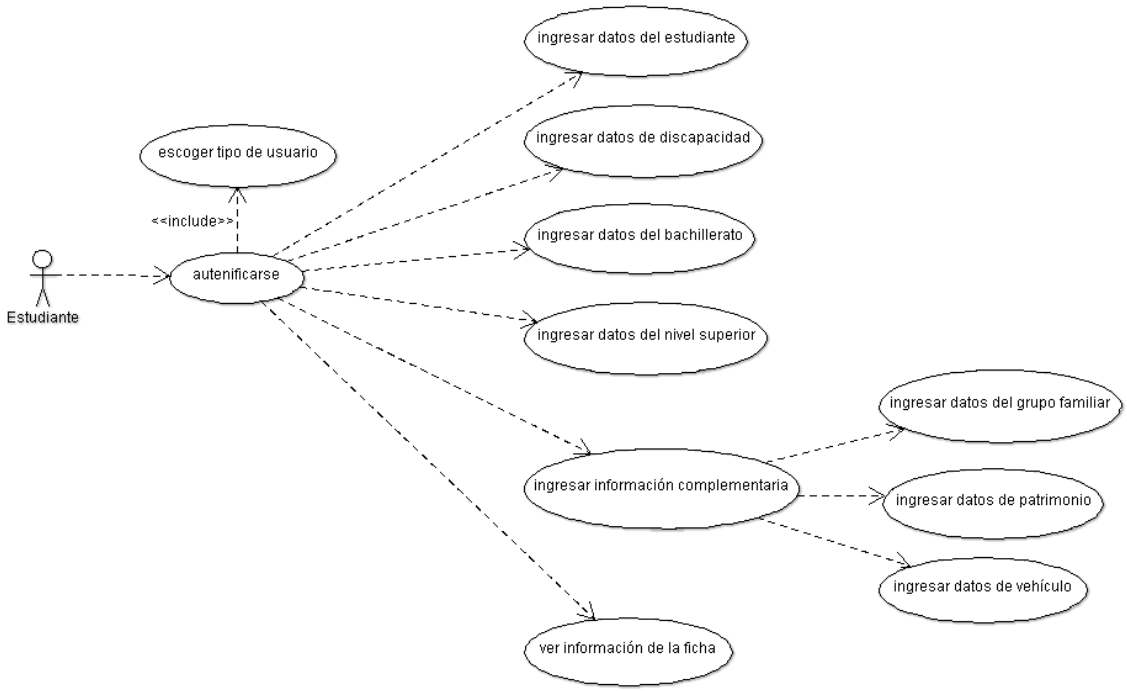



**Figura V.42Caso de Uso Refinado – Estudiante**

**Fuente:** Autoras

### 5.8.2. Diseño Lógico

#### 5.8.2.1.Tecnología a utilizar en el proyecto

- Entorno de desarrollo Java NetBeans IDE 7.2.1
- Base de Datos PostgreSQL 8.4
- Documentación MSF

#### 5.8.2.2.Diagrama de Secuencia

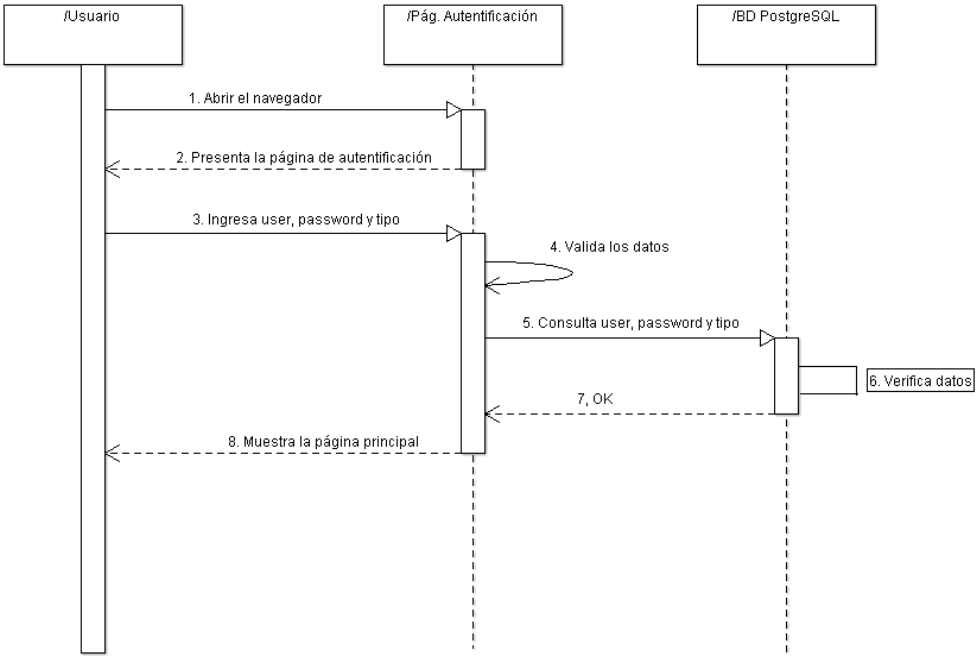



**Figura V.43Diagrama de Secuencia**

**Fuente:** Autoras

## 5.8.2.3.Diagrama de Clases

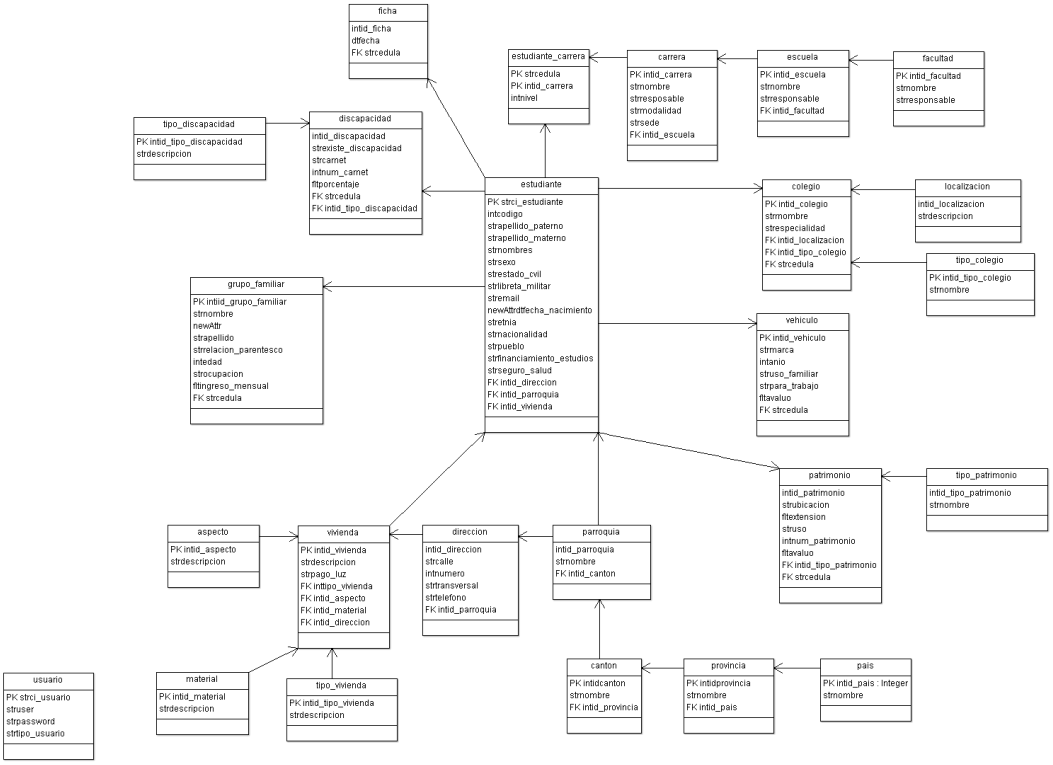



**Figura V.44Diagrama de Clases**

**Fuente:** Autoras

## 5.8.2.4.Diseño de Interfaces

- Autentificación de usuarios


- Interfaz principal del administrador


- Interfaz para el ingreso de la ficha estudiantil

### 5.8.3. Diseño Físico

#### 5.8.3.1. Diagrama de Actividades

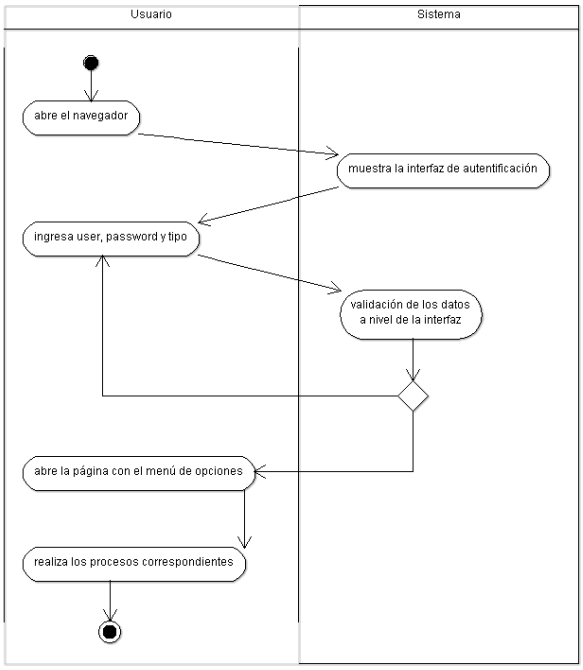



**Figura V.45Diagrama de Actividades**

**Fuente:** Autoras

### 5.8.3.2. Diagrama de Componentes

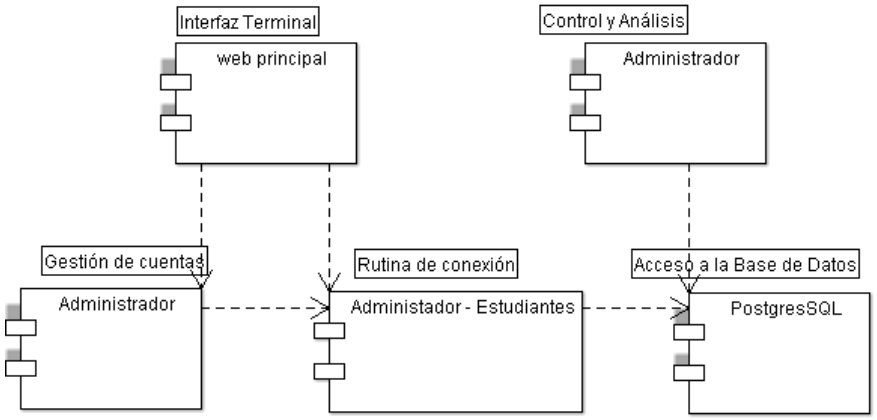



**Figura V.46Diagrama de Componentes**

**Fuente:** Autoras

### 5.8.3.3. Diagrama de Implementación

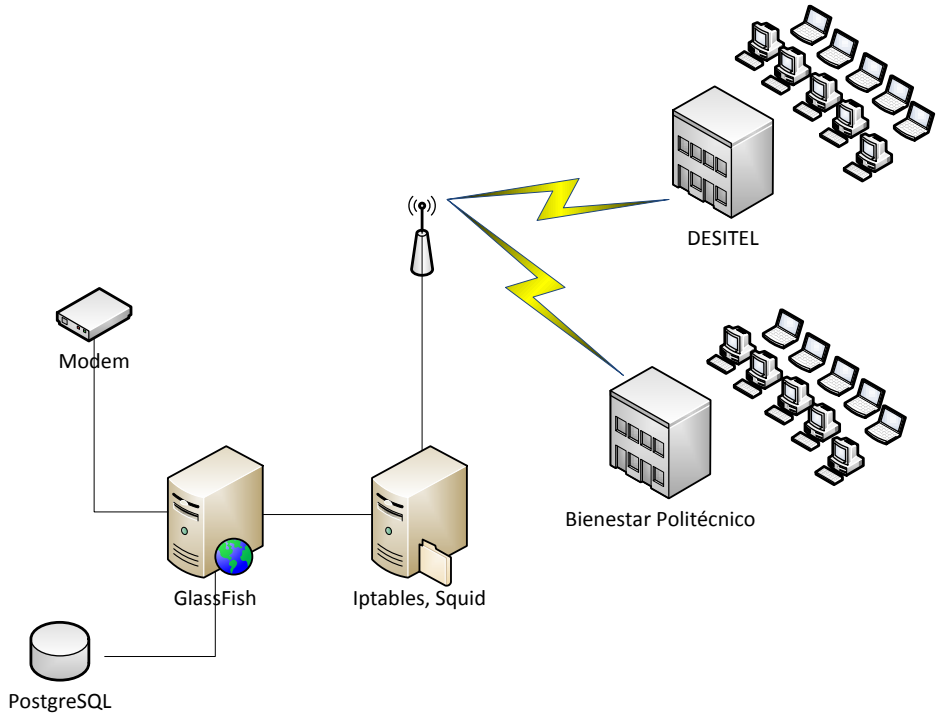



**Figura V.47Diagrama de Implementación**

**Fuente:** Autoras

### 5.8.3.4. Modelo Físico de la Base de Datos

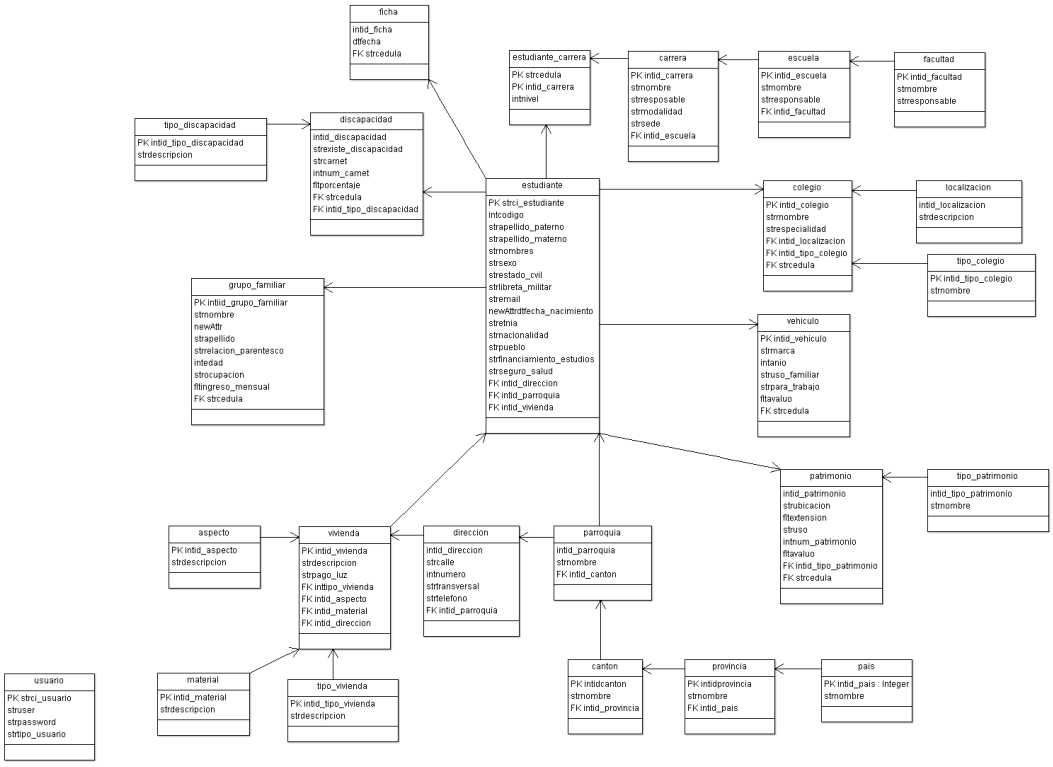



**Figura V.48Modelo Físico de la Base de Datos**

**Fuente:** Autoras

## 5.9. Fase de Desarrollo

Las versiones permiten a los clientes tener una visión clara de la aplicación desarrollada

### 5.9.1. Nomenclatura y Estándares

A continuación se detalla:

**Tabla V.XLIVNomenclatura y Estándares**

| Archivo | Extensión |
|---|---|
| Programa en java | fileName.jsp |
| Imágenes | Image.png |
| Clases | nameClase.java |

**Fuente:** Autoras

### 5.9.2. Capa de Datos

#### 5.9.2.1. Diccionario de Datos

**Tabla V.XLVDiccionario de Datos**

| Nombre del Campo | Tipo de Dato | Longitud | Clave Primaria | Calculado |
|---|---|---|---|---|
| **Tabla: Aspecto** | | | | |
| intid_aspecto | serial | - | X | No |
| strdescripcion | character varying | 60 | - | No |
| **Tabla: Cantón** | | | | |
| intid_canton | serial | - | X | No |
| strnombre | character varying | 40 | - | No |
| intid_provincia | integer | - | - | No |
| **Tabla: Carrera** | | | | |
| intid_carrera | serial | - | X | No |
| strnombre | character varying | 60 | - | No |

| | | | | |
|---|---|---|---|---|
| strresponsable | character varying | 40 | - | No |
| intid_escuela | integer | - | - | No |
| **Tabla: Colegio** | | | | |
| intid_colegio | serial | - | X | No |
| strnombre | character varying | 40 | - | No |
| strespecialidad | character varying | 40 | - | No |
| intid_localizacion | integer | - | - | No |
| intid_tipo_colegio | integer | - | - | No |
| intid_parroquia | integer | - | - | No |
| strcedula | character varying | 10 | - | No |
| **Tabla: Dirección** | | | | |
| intid_dirección | serial | - | X | No |
| strcalle | character varying | 40 | - | No |
| intnumero | integer | - | X | No |
| strtransversal | character varying | 40 | - | No |
| strtelefono | character varying | 40 | - | No |
| intid_parroquia | integer | - | - | No |
| **Tabla: Discapacidad** | | | | |
| intid_discapacidad | serial | - | X | No |
| strexiste_discapacidad | character varying | 40 | - | No |
| strcarnet | character varying | 40 | - | No |
| intnum_carnet | integer | - | - | No |

| | | | | |
|---|---|---|---|---|
| fltporcentaje | numeric | 9,2 | - | No |
| strcedula | character varying | 40 | - | No |
| intid_tipo_discapacidad | integer | - | - | No |
| **Tabla: Escuela** | | | | |
| intid_escuela | serial | - | X | No |
| strnombre | character varying | 60 | - | No |
| strresponsable | character varying | 40 | - | No |
| intid_facultad | integer | - | - | No |
| **Tabla: Estado Civil** | | | | |
| intid_estado_civil | serial | - | X | No |
| strdescripcion | character varying | 40 | - | No |
| **Tabla: Estudiante** | | | | |
| strcedula | character varying | 10 | X | No |
| intcodigo | integer | - | | No |
| strapellido_paterno | character varying | 40 | - | No |
| strapellido_materno | character varying | 40 | - | No |
| strnombres | character varying | 40 | - | No |
| strlibreta_militar | character varying | 40 | - | No |
| strtelefono | character varying | 40 | - | No |
| stremail | character varying | 40 | - | No |
| dtfecha_nacimiento | date | - | - | No |
| intid_sexo | integer | - | - | No |
| intid_estado_civil | integer | - | - | No |

| | | | | |
|---|---|---|---|---|
| intid_etnia | integer | - | - | No |
| intid_nacionalidad | integer | - | - | No |
| intid_pueblo | integer | - | - | No |
| intid_financiamiento | integer | - | - | No |
| intid_seguro | integer | - | - | No |
| intid_parroquia | integer | - | - | No |
| intid_direccion | integer | - | - | No |
| intid_parroquia | integer | - | - | No |
| intid_vivienda | integer | - | - | No |
| **Tabla: Estudiante-Carrera** | | | | |
| strcedula | character varying | 10 | X | No |
| intid_carrera | integer | - | X | No |
| intnivel | integer | - | - | No |
| intid_modalidad | integer | - | - | No |
| intid_sede | integer | - | - | No |
| **Tabla: Etnia** | | | | |
| intid_etnia | serial | - | X | No |
| strnombre | character varying | 40 | - | No |
| **Tabla: Facultad** | | | | |
| intid_facultad | serial | - | X | No |
| strnombre | character varying | 40 | - | No |
| strresponsable | character varying | 40 | - | No |
| **Tabla: Ficha** | | | | |

| | | | | |
|---|---|---|---|---|
| intid_ficha | Serial | - | X | No |
| dtfecha | Date | - | - | No |
| strcedula | character varying | 40 | - | No |
| **Tabla: Financiamiento** | | | | |
| intid_financiamiento | serial | - | X | No |
| strdescripcion | character varying | 40 | - | No |
| **Tabla: Grupo Familiar** | | | | |
| intid_grupo_familiar | serial | - | X | No |
| strcedula | character varying | 11 | - | No |
| strnombre | character varying | 40 | - | No |
| strapellido | character varying | 40 | - | No |
| strrelacion_parentesco | character varying | 40 | - | No |
| intedad | integer | - | - | No |
| strocupacion | character varying | 40 | - | No |
| fltingreso_mensual | numeric | 9,2 | - | No |
| **Tabla: Localización** | | | | |
| intid_localizacion | serial | - | X | No |
| strdescripcion | character varying | 40 | - | No |
| **Tabla: Material** | | | | |
| intid_material | serial | - | X | No |
| strdescripcion | character varying | 40 | - | No |
| **Tabla: Modalidad** | | | | |
| intid_modalidad | serial | - | X | No |

| | | | | |
|---|---|---|---|---|
| strdescripcion | character varying | 40 | - | No |
| **Tabla: Nacionalidad** | | | | |
| intid_nacionalidad | serial | - | X | No |
| strdescripcion | character varying | 40 | - | No |
| **Tabla: Pago Luz** | | | | |
| intid_pago_luz | serial | - | X | No |
| Strdescripcion | character varying | 40 | - | No |
| **Tabla: País** | | | | |
| intid_pais | serial | - | X | No |
| strnombre | character varying | 40 | - | No |
| **Tabla: Parroquia** | | | | |
| intid_parroquia | serial | - | X | No |
| strnombre | character varying | 40 | - | No |
| intid_canton | integer | - | - | No |
| **Tabla: Patrimonio** | | | | |
| intid_patrimonio | serial | - | X | No |
| strubicacion | character varying | 40 | - | No |
| fltextencion | numeric | 9,2 | - | No |
| struso | character varying | 40 | - | No |
| intnum_propiedades | integer | - | - | No |
| fltavaluo | numeric | 9,2 | - | No |
| intid_tipo_patrimonio | integer | - | - | No |
| strcedula | character varying | 10 | - | No |

| | | | | |
|---|---|---|---|---|
| **Tabla: Provincia** | | | | |
| intid_provincia | serial | - | X | No |
| strnombre | character varying | 40 | - | No |
| intid_pais | integer | - | . | No |
| **Tabla: Pueblo** | | | | |
| intid_pueblo | serial | - | X | No |
| strdescripcion | character varying | 40 | - | No |
| **Tabla: Sede** | | | | |
| intid_sede | serial | - | X | No |
| strdescripcion | character varying | 40 | - | No |
| **Tabla: Seguro** | | | | |
| intid_seguro | serial | - | X | No |
| strdescripcion | character varying | 40 | - | No |
| **Tabla: Tipo Colegio** | | | | |
| intid_tipo_colegio | serial | - | X | No |
| strnombre | character varying | 40 | - | No |
| **Tabla: Tipo Discapacidad** | | | | |
| intid_tipo_discapacidad | serial | - | X | No |
| strnombre | character varying | 40 | - | No |
| **Tabla: Tipo Patrimonio** | | | | |
| intid_tipo_patrimonio | serial | - | X | No |
| strnombre | character varying | 40 | - | No |
| **Tabla: Tipo Vivienda** | | | | |

| | | | | |
|---|---|---|---|---|
| intid_tipo_vivienda | serial | - | X | No |
| strnombre | character varying | 40 | - | No |
| **Tabla: Usuario** | | | | |
| strci_usuario | character varying | - | X | No |
| strpassword | character varying | 40 | - | No |
| strtipo_usuario | character varying | 40 | - | No |
| **Tabla: Vehículo** | | | | |
| intid_vehiculo | serial | - | X | No |
| strmarca | character varying | 40 | - | No |
| intanio | integer | - | - | No |
| struso_familiar | character varying | 40 | - | No |
| strpara_trabajo | character varying | 40 | - | No |
| fltavaluo | numeric | 9,2 | - | No |
| strcedula | character varying | 10 | - | No |
| **Tabla: Vivienda** | | | | |
| intid_vivienda | serial | - | X | No |
| strdescripcion | character varying | 40 | - | No |
| intid_pago_luz | integer | - | - | No |
| intid_tipo_vivienda | integer | - | - | No |
| intid_aspecto | integer | - | - | No |
| intid_material | integer | - | - | No |
| intid_direccion | integer | - | - | No |

**Fuente:** Autoras

### 5.9.2.2. Script de la Base de Datos

```
-- Database: ficha
-- DROP DATABASE ficha;
CREATE DATABASE ficha
  WITH OWNER = user_ficha
       ENCODING = 'UTF8'
       TABLESPACE = pg_default
       LC_COLLATE = 'Spanish_Ecuador.1252'
LC_CTYPE = 'Spanish_Ecuador.1252'
       CONNECTION LIMIT = -1;

-- Table: aspecto
-- DROP TABLE aspecto;
CREATE TABLE aspecto
(
  intid_aspecto serial NOT NULL,
  strdescripcion character varying(60) NOT NULL,
  CONSTRAINT pk_aspecto PRIMARY KEY (intid_aspecto)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE aspecto OWNER TO postgres;

-- Table: canton
-- DROP TABLE canton;
CREATE TABLE canton
(
  intid_canton serial NOT NULL,
  strnombre character varying(40) NOT NULL,
  intid_provincia integer NOT NULL,
  CONSTRAINT pk_canton PRIMARY KEY (intid_canton),
  CONSTRAINT provincia_canton FOREIGN KEY (intid_provincia)
      REFERENCES provincia (intid_provincia) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE canton OWNER TO postgres;
```

```
-- Table: carrera
-- DROP TABLE carrera;
CREATE TABLE carrera
(
intid_carrera serial NOT NULL,
  strnombre character varying(60) NOT NULL,
  strresponsable character varying(40) NOT NULL,
intid_escuela integer NOT NULL,
CONSTRAINT pk_carrera PRIMARY KEY (intid_carrera),
CONSTRAINT escuela_carrera FOREIGN KEY (intid_escuela)
      REFERENCES escuela (intid_escuela) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE carrera OWNER TO postgres;

-- Table: colegio
-- DROP TABLE colegio;
CREATE TABLE colegio
(
  intid_colegio serial NOT NULL,
  strnombre character varying(40) NOT NULL,
  strespecialidad character varying(40) NOT NULL,
  intid_localizacion integer NOT NULL,
  intid_tipo_colegio integer NOT NULL,
  intid_parroquia integer NOT NULL,
  strcedula character varying(40) NOT NULL,
  CONSTRAINT pk_colegio PRIMARY KEY (intid_colegio),
  CONSTRAINT estudiante_colegio FOREIGN KEY (strcedula)
      REFERENCES estudiante (strcedula) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT localizacion_colegio FOREIGN KEY (intid_localizacion)
      REFERENCES localizacion (intid_localizacion) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT parroquia_colegio FOREIGN KEY (intid_parroquia)
      REFERENCES parroquia (intid_parroquia) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT tipo_colegio_colegio FOREIGN KEY (intid_tipo_colegio)
      REFERENCES tipo_colegio (intid_tipo_colegio) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
```

```
  OIDS=FALSE
);
ALTER TABLE colegio OWNER TO postgres;

-- Table: direccion
-- DROP TABLE direccion;
CREATE TABLE direccion
(
 intid_direccion serial NOT NULL,
  strcalle character varying(60) NOT NULL,
  intnumero integer,
  strtransversal character varying(60),
  strtelefono character varying(40) NOT NULL,
  intid_parroquia integer NOT NULL,
  CONSTRAINT pk_direccion PRIMARY KEY (intid_direccion),
  CONSTRAINT parroquia_direccion FOREIGN KEY (intid_parroquia)
      REFERENCES parroquia (intid_parroquia) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)WITH (
  OIDS=FALSE
);
ALTER TABLE direccion OWNER TO postgres;

-- Table: discapacidad
-- DROP TABLE discapacidad;
CREATE TABLE discapacidad
(
  intid_discapacidad serial NOT NULL,
  strexiste_discapacidad character varying(40) NOT NULL,
  strcarnet character varying(40),
  intnum_carnet integer,
  fltporcentaje numeric(9,2),
  strcedula character varying(40) NOT NULL,
intid_tipo_discapacidad integer NOT NULL,
CONSTRAINT pk_discapacidad PRIMARY KEY (intid_discapacidad),
  CONSTRAINT estudiante_discapacidad FOREIGN KEY (strcedula)
      REFERENCES estudiante (strcedula) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT tipo_discapacidad_discapacidad FOREIGN KEY (intid_tipo_discapacidad)
      REFERENCES tipo_discapacidad (intid_tipo_discapacidad) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
```

```
);
ALTER TABLE discapacidad OWNER TO postgres;

-- Table: escuela
-- DROP TABLE escuela;
CREATE TABLE escuela
(
  intid_escuela serial NOT NULL,
strnombre character varying(40) NOT NULL,
  strresponsable character varying(40) NOT NULL,
  intid_facultad integer NOT NULL,
  CONSTRAINT pk_escuela PRIMARY KEY (intid_escuela),
  CONSTRAINT facultad_escuela FOREIGN KEY (intid_facultad)
      REFERENCES facultad (intid_facultad) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE escuela OWNER TO postgres;

-- Table: estado_civil
-- DROP TABLE estado_civil;
CREATE TABLE estado_civil
(
intid_estado_civil serial NOT NULL,
strdescripcion character varying(40) NOT NULL,
  CONSTRAINT pk_estado_civil PRIMARY KEY (intid_estado_civil)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE estado_civil OWNER TO postgres;

-- Table: estudiante
-- DROP TABLE estudiante;
CREATE TABLE estudiante
(
strcedula character varying(40) NOT NULL,
intcodigo integer,
  strapellido_paterno character varying(40),
strapellido_materno character varying(40),
  strnombres character varying(40),
  strlibreta_militar character varying(40),
```

```
  strtelefono character varying(40),
  stremail character varying(40),
dtfecha_nacimiento date,
  intid_sexo integer,
  intid_estado_civil integer,
  intid_etnia integer,
  intid_nacionalidad integer,
  intid_pueblo integer,
  intid_financiamiento integer,
  intid_seguro integer,
  intid_direccion integer,
  intid_parroquia integer,
  intid_vivienda integer,
  CONSTRAINT pk_estudiante PRIMARY KEY (strcedula),
  CONSTRAINT direccion_estudiante FOREIGN KEY (intid_direccion)
      REFERENCES direccion (intid_direccion) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT estado_civil_estudiante FOREIGN KEY (intid_estado_civil)
      REFERENCES estado_civil (intid_estado_civil) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT etnia_estudiante FOREIGN KEY (intid_etnia)
      REFERENCES etnia (intid_etnia) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT financiamiento_estudiante FOREIGN KEY (intid_financiamiento)
      REFERENCES financiamiento (intid_financiamiento) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT nacionalidad_estudiante FOREIGN KEY (intid_nacionalidad)
      REFERENCES nacionalidad (intid_nacionalidad) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT parroquia_estudiante FOREIGN KEY (intid_parroquia)
      REFERENCES parroquia (intid_parroquia) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT pueblo FOREIGN KEY (intid_pueblo)
REFERENCES pueblo (intid_pueblo) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT seguro_estudiante FOREIGN KEY (intid_seguro)
      REFERENCES seguro (intid_seguro) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT sexo_estudiante FOREIGN KEY (intid_sexo)
      REFERENCES sexo (intid_sexo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT vivienda_estudiante FOREIGN KEY (intid_vivienda)
      REFERENCES vivienda (intid_vivienda) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
```

```
)
WITH (
  OIDS=FALSE
);
ALTER TABLE estudiante OWNER TO postgres;

-- Table: estudiante_carrera
-- DROP TABLE estudiante_carrera;
CREATE TABLE estudiante_carrera
(
strcedula character varying(40) NOT NULL,
  intid_carrera integer NOT NULL,
  intnivel integer NOT NULL,
  intid_modalidad integer NOT NULL,
  intid_sede integer NOT NULL,
  CONSTRAINT pk_estudiante_carrera PRIMARY KEY (strcedula, intid_carrera),
  CONSTRAINT carrera_estudiante_carrera FOREIGN KEY (intid_carrera)
      REFERENCES carrera (intid_carrera) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT estudiante_estudiante_carrera FOREIGN KEY (strcedula)
      REFERENCES estudiante (strcedula) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT modalidad_estudiante_carrera FOREIGN KEY (intid_modalidad)
      REFERENCES modalidad (intid_modalidad) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT sede_estudiante_carrera FOREIGN KEY (intid_sede)
      REFERENCES sede (intid_sede) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)WITH (
  OIDS=FALSE
);
ALTER TABLE estudiante_carrera OWNER TO postgres;

-- Table: etnia
-- DROP TABLE etnia;
CREATE TABLE etnia
(
  intid_etnia serial NOT NULL,
  strdescripcion character varying(40) NOT NULL,
  CONSTRAINT pk_etnia PRIMARY KEY (intid_etnia)
)
WITH (
  OIDS=FALSE
);
```

```
ALTER TABLE etnia OWNER TO postgres;

-- Table: facultad
-- DROP TABLE facultad;
CREATE TABLE facultad
(
  intid_facultad serial NOT NULL,
  strnombre character varying(60) NOT NULL,
  strresponsable character varying(40) NOT NULL,
  CONSTRAINT pk_facultad PRIMARY KEY (intid_facultad)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE facultad OWNER TO postgres;

-- Table: ficha
-- DROP TABLE ficha;
CREATE TABLE ficha
(
  intid_ficha serial NOT NULL,
  dtfecha date NOT NULL,
  strcedula character varying(40) NOT NULL,
  CONSTRAINT pk_ficha PRIMARY KEY (intid_ficha),
  CONSTRAINT estudiante_ficha FOREIGN KEY (strcedula)
      REFERENCES estudiante (strcedula) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE ficha OWNER TO postgres;

-- Table: financiamiento
-- DROP TABLE financiamiento;
CREATE TABLE financiamiento
(
  intid_financiamiento serial NOT NULL,
  strdescripcion character varying(40) NOT NULL,
  CONSTRAINT pk_financiamiento PRIMARY KEY (intid_financiamiento)
)
WITH (
  OIDS=FALSE
);
```

```
ALTER TABLE financiamiento OWNER TO postgres;

-- Table: grupo_familiar
-- DROP TABLE grupo_familiar;
CREATE TABLE grupo_familiar
(
intid_grupo_familiar serial NOT NULL,
  strcedula character varying(11) NOT NULL,
  strnombre character varying(40) NOT NULL,
  strapellido character varying(40) NOT NULL,
  strrelacion_parentesco character varying(40) NOT NULL,
  intedad integer NOT NULL,
  strocupacion character varying(40) NOT NULL,
  fltingreso_mensual numeric(9,2) NOT NULL,
  CONSTRAINT pk_grupo_familiar PRIMARY KEY (intid_grupo_familiar),
  CONSTRAINT estudiante_grupo_familiar FOREIGN KEY (strcedula)
      REFERENCES estudiante (strcedula) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTIONWITH (
  OIDS=FALSE
);
ALTER TABLE grupo_familiar OWNER TO postgres;

-- Table: localizacion
-- DROP TABLE localizacion;
CREATE TABLE localizacion
(
  intid_localizacion serial NOT NULL,
  strdescripcion character(40) NOT NULL,
  CONSTRAINT pk_localizacion PRIMARY KEY (intid_localizacion)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE localizacion OWNER TO postgres;

-- Table: material
-- DROP TABLE material;
CREATE TABLE material
(
  intid_material serial NOT NULL,
  strdescripcion character varying(40) NOT NULL,
  CONSTRAINT pk_material PRIMARY KEY (intid_material)
)
WITH (
```

```
  OIDS=FALSE
);
ALTER TABLE material OWNER TO postgres;

-- Table: modalidad
-- DROP TABLE modalidad;
CREATE TABLE modalidad
(
  intid_modalidad serial NOT NULL,
  strdescripcion character varying(40) NOT NULL,
  CONSTRAINT pk_modalidad PRIMARY KEY (intid_modalidad)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE modalidad OWNER TO postgres;

-- Table: nacionalidad
-- DROP TABLE nacionalidad;
CREATE TABLE nacionalidad
(
  intid_nacionalidad serial NOT NULL,
strdescripcion character varying(40) NOT NULL,
  CONSTRAINT pk_nacionalidad PRIMARY KEY (intid_nacionalidad)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE nacionalidad OWNER TO postgres;

-- Table: pago_luz
-- DROP TABLE pago_luz;
CREATE TABLE pago_luz
(
  intid_pago_luz serial NOT NULL,
strdescripcion character varying(40) NOT NULL,
  CONSTRAINT pk_pago_luz PRIMARY KEY (intid_pago_luz)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE pago_luz OWNER TO postgres;

-- Table: pais
```

```
-- DROP TABLE pais;
CREATE TABLE pais
(
  intid_pais serial NOT NULL,
  strnombre character varying(40) NOT NULL,
  CONSTRAINT pk_pais PRIMARY KEY (intid_pais)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE pais OWNER TO postgres;

-- Table: parroquia
-- DROP TABLE parroquia;
CREATE TABLE parroquia
(
  intid_parroquia serial NOT NULL,
  strnombre character varying(40) NOT NULL,
  intid_canton integer NOT NULL,
  CONSTRAINT pk_parroquia PRIMARY KEY (intid_parroquia),
  CONSTRAINT canton_parroquia FOREIGN KEY (intid_canton)
      REFERENCES canton (intid_canton) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE parroquia OWNER TO postgres;

-- Table: patrimonio
-- DROP TABLE patrimonio;
CREATE TABLE patrimonio
(
  intid_patrimonio serial NOT NULL,
  strubicacion character varying(40) NOT NULL,
  fltextension numeric(9,2) NOT NULL,
  struso character varying(40) NOT NULL,
  intnum_propiedades integer NOT NULL,
  fltavaluo numeric(9,2) NOT NULL,
intid_tipo_patrimonio integer NOT NULL,
strcedula character varying(40) NOT NULL,
  CONSTRAINT pk_patrimonio PRIMARY KEY (intid_patrimonio),
  CONSTRAINT estudiante_patrimonio FOREIGN KEY (strcedula)
      REFERENCES estudiante (strcedula) MATCH SIMPLE
```

```
    ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT tipo_patrimonio_patrimonio FOREIGN KEY (intid_tipo_patrimonio)
    REFERENCES tipo_patrimonio (intid_tipo_patrimonio) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE patrimonio OWNER TO postgres;

-- Table: provincia
-- DROP TABLE provincia;
CREATE TABLE provincia
(
  intid_provincia serial NOT NULL,
  strnombre character varying(40) NOT NULL,
  intid_pais integer NOT NULL,
  CONSTRAINT pk_provincia PRIMARY KEY (intid_provincia),
  CONSTRAINT pais_provincia FOREIGN KEY (intid_pais)
    REFERENCES pais (intid_pais) MATCH SIMPLE
    ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE provincia OWNER TO postgres;

-- Table: pueblo
-- DROP TABLE pueblo;
CREATE TABLE pueblo
(
  intid_pueblo serial NOT NULL,
strdescripcion character varying(40) NOT NULL,
  CONSTRAINT pk_pueblo PRIMARY KEY (intid_pueblo)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE pueblo OWNER TO postgres;

-- Table: sede
-- DROP TABLE sede;
CREATE TABLE sede
(
```

```
  intid_sede serial NOT NULL,
  strdescripcion character varying(40) NOT NULL,
  CONSTRAINT pk_sede PRIMARY KEY (intid_sede)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE sede OWNER TO postgres;

-- Table: seguro
-- DROP TABLE seguro;
CREATE TABLE seguro
(
  intid_seguro serial NOT NULL,
  strdescripcion character varying(40) NOT NULL,
  CONSTRAINT pk_seguro PRIMARY KEY (intid_seguro)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE seguro OWNER TO postgres;

-- Table: sexo
-- DROP TABLE sexo;
CREATE TABLE sexo
(
  intid_sexo serial NOT NULL,
  strdescripcion character varying(40) NOT NULL,
  CONSTRAINT pk_sexo PRIMARY KEY (intid_sexo)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE sexo OWNER TO postgres;

-- Table: tipo_colegio
-- DROP TABLE tipo_colegio;
CREATE TABLE tipo_colegio
(
  intid_tipo_colegio serial NOT NULL,
strnombre character varying(40) NOT NULL,
  CONSTRAINT pk_tipo_colegio PRIMARY KEY (intid_tipo_colegio)
)
WITH (
```

```
  OIDS=FALSE
);
ALTER TABLE tipo_colegio OWNER TO postgres;

-- Table: tipo_discapacidad
-- DROP TABLE tipo_discapacidad;
CREATE TABLE tipo_discapacidad
(
  intid_tipo_discapacidad serial NOT NULL,
strdescripcion character varying(40) NOT NULL,
CONSTRAINT pk_tipo_discapacidad PRIMARY KEY (intid_tipo_discapacidad)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE tipo_discapacidad OWNER TO postgres;

-- Table: tipo_patrimonio
-- DROP TABLE tipo_patrimonio;
CREATE TABLE tipo_patrimonio
(
  intid_tipo_patrimonio serial NOT NULL,
strdescripcion character varying(40) NOT NULL,
CONSTRAINT pk_tipo_patrimonio PRIMARY KEY (intid_tipo_patrimonio)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE tipo_patrimonio OWNER TO postgres;

-- Table: tipo_vivienda
-- DROP TABLE tipo_vivienda;
CREATE TABLE tipo_vivienda
(
  intid_tipo_vivienda serial NOT NULL,
strdescripcion character varying(40) NOT NULL,
  CONSTRAINT pk_tipo_vivienda PRIMARY KEY (intid_tipo_vivienda)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE tipo_vivienda OWNER TO postgres;

-- Table: usuario
```

```
-- DROP TABLE usuario;
CREATE TABLE usuario
(
  strci_usuario character varying(40) NOT NULL,
  strpassword character varying(40) NOT NULL,
  strtipo_usuario character varying(40) NOT NULL,
  CONSTRAINT pk_usuario PRIMARY KEY (strci_usuario)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE usuario OWNER TO postgres;

-- Table: vehiculo
-- DROP TABLE vehiculo;
CREATE TABLE vehiculo
(
  intid_vehiculo serial NOT NULL,
  strmarca character varying(40) NOT NULL,
  intanio integer NOT NULL,
  struso_familiar character varying(40) NOT NULL,
  strpara_trabajo character varying(40) NOT NULL,
  fltavaluo numeric(9,2) NOT NULL,
  strcedula character varying(40) NOT NULL,
  CONSTRAINT pk_vehiculo PRIMARY KEY (intid_vehiculo),
  CONSTRAINT estudiante_vehiculo FOREIGN KEY (strcedula)
      REFERENCES estudiante (strcedula) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE vehiculo OWNER TO postgres;

-- Table: vivienda
-- DROP TABLE vivienda;
CREATE TABLE vivienda
(
  intid_vivienda serial NOT NULL,
  strdescripcion character varying(40) NOT NULL,
intid_pago_luz integer NOT NULL,
intid_tipo_vivienda integer NOT NULL,
intid_aspecto integer NOT NULL,
  intid_material integer NOT NULL,
```

```
  intid_direccion integer NOT NULL,
  CONSTRAINT pk_vivienda PRIMARY KEY (intid_vivienda),
  CONSTRAINT aspecto_vivienda FOREIGN KEY (intid_aspecto)
      REFERENCES aspecto (intid_aspecto) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT direccion_vivienda FOREIGN KEY (intid_direccion)
      REFERENCES direccion (intid_direccion) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT material_vivienda FOREIGN KEY (intid_material)
      REFERENCES material (intid_material) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT pago_luz_vivienda FOREIGN KEY (intid_pago_luz)
      REFERENCES pago_luz (intid_pago_luz) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT tipo_vivienda_vivienda FOREIGN KEY (intid_tipo_vivienda)
      REFERENCES tipo_vivienda (intid_tipo_vivienda) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE vivienda OWNER TO postgres;
```

### 5.9.2.3. Implementación de la Base de Datos

Estructura objetos de la base de datos

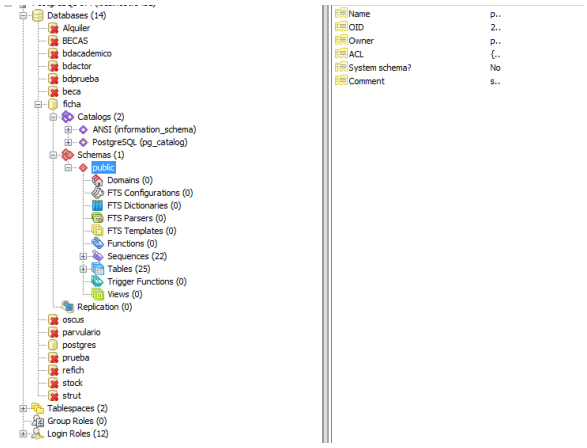



**Figura V. 49Objetos de la Base de Datos**

**Fuente:** Autoras

Estructura de las tablas de la base de datos

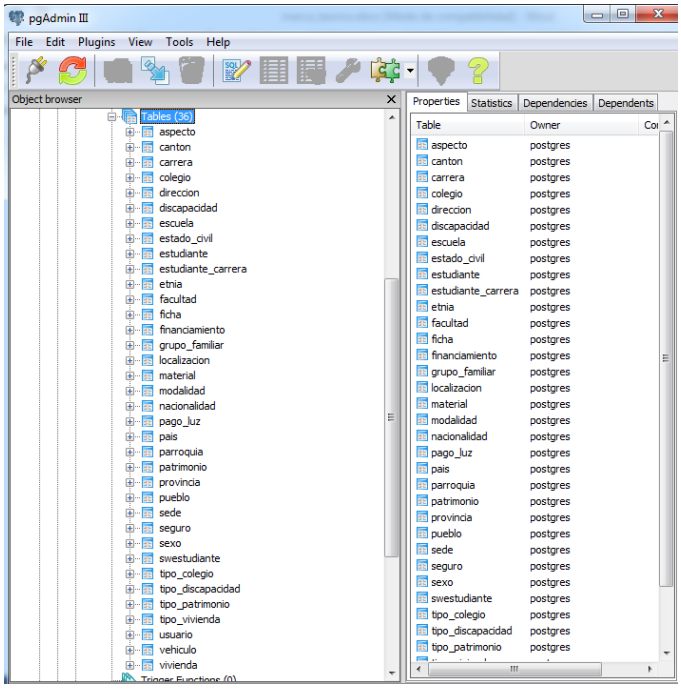



**Figura V.50Tablas de la Base de Datos**

**Fuente:** Autoras

Estructura de una Tabla

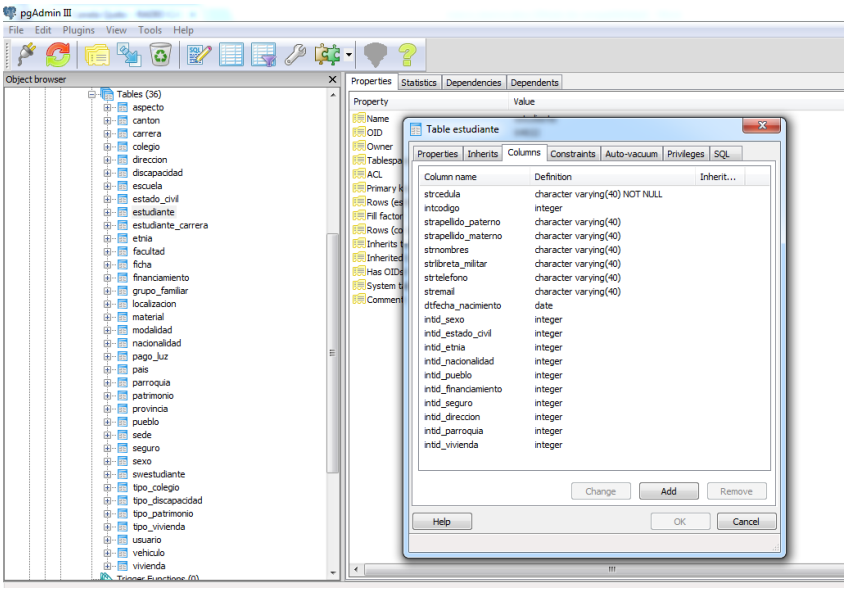



**Figura V.51Estructura de una Tabla**

**Fuente:** Autoras

## 5.10. Fase de Estabilización

Esta fase permite validar los requerimientos de la solución implementada. Es desarrollo de la solución del proyecto ha sido completado y realizadas las pruebas correspondientes

### 5.10.1. Código Fuente

El código fuente se presente en su respectivo anexo

**Documentación de la instalación**

Para facilitar la aplicación, su uso y manipulación se presenta los siguientes manuales:

- Manual Técnico
- Manual del Usuario

## 5.11. Fase de Instalación

### 5.11.1. Tareas a realizar

La aplicación se ha finalizado y se procede a la entrega de la misma a la Directora del Departamento de Bienestar Politécnico de la ESPOCH.

## CONCLUSIONES

- Actualmente existen varios frameworks MVC para el desarrollo de aplicaciones web en Java, estos han enfocado sus esfuerzos en simplificar la construcción de aplicaciones ricas y mucho más interactivas atacando los mismos problemas navegación, internacionalización, manejo de errores, validación de entradas, etc. Destacándose STRUTS, JSF y SPRING.
- JSF tiene capacidad para ser visto en dispositivos móviles mientras que Struts está limitado a HTML y HTTP.
- Una ventaja importante de JSF vs a otros frameworks MVC es que es el primer framework con una especificación incluida en la última versión de J2EE y es el principal estándar del mercado; razón por la cual todos los servidores J2EE deben soportar JSF y los principales IDEs permiten crear aplicaciones con JSF.
- Hoy en día existen frameworks MVC de java que con el paso del tiempo han madurado y solucionan de manera eficiente el problema de reutilización de código, tiempo de desarrollo, seguridades, generando aplicaciones web robustas. Con esta investigación se demostró que JSF alcanzo un 92,16% en eficiencia por lo que se determina que es el framework más adecuado para desarrollar la aplicación web empresarial en ambiente J2EE; por otro lado STRUTS obtuvo una calificación de 64,71% lo que indica que no es un framework que proporciones las facilidades en el momento de desarrollar una aplicación web.
- La aplicación web implementada en el Departamento de Bienestar Politécnico de la ESPOCH garantiza el correcto tratamiento de la información de los estudiantes, haciendo uso de los recursos existentes.

- Spring proporciona la posibilidad de integrar al framework con otras herramientas o incluso otros frameworks con el fin de obtener los beneficios que el desarrollador desea de cada uno de ellos generando una estructura más sólida y consistente reduciendo el mantenimiento de la aplicación.

## RECOMENDACIONES

- Se recomienda que el desarrollador comprenda claramente el funcionamiento general del modelo MVC así como el ciclo de vida de un framework puesto que esto ayudara a desarrollar y depurar de mejor manera la aplicación.
- Actualmente existen frameworks de desarrollo web que pueden integrar a otros, por ello recomendamos que se realicen trabajos a futuro de arquitecturas como STRUTS+JSF, JSF+AJAX, SPRING+STRUTS con el fin de explotar los beneficios que nos brindan estos frameworks en el desarrollo de aplicaciones web.
- Se recomienda que se realicen aplicaciones web usando el framework JSF debido a que muestra potencialidades mediante el manejo de eventos, reducción de código y posee múltiples componentes para las UI reduciendo el tiempo de desarrollo.

## RESUMEN

Se analizóframeworks MVC de java para el desarrollo de aplicaciones web empresariales. Caso práctico: Sistema de Bienestar Politécnico de la Escuela Superior Politécnica de Chimborazo.

Para el desarrollo de esta investigación se utilizó el método científico y el método descriptivo con el fin de recopilar información, analizar e interpretar los resultados para el análisis comparativo de los frameworks para ello se escogió los parámetros como: producto, rendimiento, desarrollo, patrón de diseño, seguridad cada uno con sus respectivas variables y gráficos estadísticos.

Para la demostración de la hipótesis se tomó en cuenta dos parámetros muy importantes en el desarrollo de aplicaciones web como son: seguridad y rapidez en donde se comprobó que la utilización de un framework Modelo Vista Controlador en este caso Java Server Facescumple con ambos requisitos alcanzando 100% de eficiencia en desarrollo y 77,78% en seguridad.

Analizando y comparando los frameworks Modelo Vista Controlador de java los más destacados para el desarrollo de aplicaciones web empresariales: Struts y Java Server Faces, se eligió JSF como el más adecuado, porque alcanzó el mayor puntaje de 92,16%, en comparación con 64,71% obtenido por Struts.

Se concluye que el framework Java Server Faces es de gran ayuda para los desarrolladores Java, debido a que reduce el tiempo de desarrollo de la aplicación brindando facilidades para el manejo de presentación y el modelo permitiendo reutilizar código y manejo de

componentes de una manera más fácil y posee una excelente documentación una comunidad activa grande.

Se recomienda el uso de este framework para el desarrollo de aplicaciones web empresariales puesto que ofrece grandes ventajas resolviendo muchos problemas en cuanto a validación, seguridad, vistas.

## SUMARY

Frameworks java MVC was analyzed for the development of applications managerial web. Practical case: System of Polytechnic Well-being of the Escuela Superior Politécnica de Chimborazo.

For the development of this investigation the scientific and descriptive methods were used with the purpose of gathering information, to analyze and interpret the results for the comparative analysis of the frameworks for it the parameters were chosen as: product, yield, development, design, pattern, security each one with their respective variables and statistical graphics.

For the demonstration of the hypothesis two very important parameters were selected in the development of applications web as they are: security and speed where it was checked that use of a framework Model View Controller in this case Java Server Faces fulfills both requirements reaching 100% of efficiency in development and 77,78% in security.

Analyzing and comparing the frameworks Model View Controller of java the most outstanding for the development of applications managerial web: Struts and Java Server Faces, was chosen JSF like the most appropriate, because it reached the biggest score of 92,16%, in comparison with 64,71% obtained by Struts.

It is concluded that the framework Java Server Faces is of great help for the developers Java, because it reduces the time of development of the application offering facilities for the presentation handling and the pattern allowing to reuse code and handling of components in an easier way it has excellent documentation a big active community.

The use of this framework is recommended for the development of applications managerial web since it offers big advantages solving many problems as for validation, security, views.

# GLOSARIO

**API:** Conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Representa una interfaz de comunicación entre componentes software.

**Aplicación web:** Aplicación informática que los usuarios utilizan accediendo a un servidor web a través de Internet o de una intranet.

**ApplicationContext:** Es una subinterfaz de BeanFactory

**Arquitectura:** Representación abstracta de los componentes de un sistema y su comportamiento

**Bean Factory:** Es uno de los componentes principales del núcleo de Spring es una implementación del patrón Factory

**Controlador**: Es el elemento más abstracto. Recibe, trata y responde los eventos enviados por el usuario o por la propia aplicación. Interactúa tanto con el modelo como con la vista.

**DAO:** 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.

**EntityManager:** Interfaz que define los métodos que son usados para interactuar con el contexto de persistencia.

**Framework Web:** Se define como una estructura definida, reusable en el que sus componentes facilitan la creación de aplicaciones web.

**Framework de Persistencia:** Componente de software encargado de traducir entre objetos y registros (de la base de datos relacional). Es decir es el encargado de que el programa y la base de datos se “entiendan”.

**Hibernate:** Herramienta de Mapeo objeto-relacional, que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones.

**HQL (Lenguaje de Consulta Hibernate):** Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Al mismo tiempo que una API para construir las consultas programáticamente

**Java:** Lenguaje de Programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90.

**JDBC:** Java DataBase Connectivity es el API de Java que define como una aplicación cliente accederá a una base de datos, independientemente del motor de base de datos al que accedamos.

**JSF:** La tecnología Java Server Faces (JSF) es un marco de trabajo de interfaces de usuario del lado de servidor para aplicaciones Web basadas en tecnología Java.

**Mapping:** Proceso de conectar objetos/atributos a tablas/columnas.

**Metadatos:** Descripciones estructuradas y opcionales que están disponibles de forma pública para ayudar a localizar objetos.

**Modelo**: Es la representación de la información en el sistema. Trabaja junto a la vista para mostrar la información al usuario y es accedida por el controlador para añadir, eliminar, consultar o actualizar datos.

**MVC:** Patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

**Objeto:** Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos). Se corresponde con los objetos reales del mundo que nos rodea, o a objetos internos del sistema (del programa). Es una instancia a una clase.

**ORM (ObjectRelational Mapping):** Técnica que realiza la transición de una representación de los datos de un modelo relacional a un modelo orientado a objetos y viceversa.

**Patrón de diseño:** Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas de diseño en distintas circunstancias).

**Persistencia:** Capacidad de almacenar y recuperar el estado de los objetos, de forma que sobrevivan a los procesos que los manipulan.

**POJO (Plain Old Java Object):** Simple clase Java que tiene métodos get y set para cada uno de los atributos.

**Serialización:** Secuencia de bytes escrita en un fichero en disco.

**Servlets:** Son clases del lenguaje Java que procesan solicitudes y construyen respuestas de forma dinámica

**Spring:** Spring es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java.2

**Spring Context:** Es un archivo de configuración que provee de información contextual al framework general

**Spring Core:** 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 (Dependecy Injection).

**Struts:** Es un framework de código abierto que se extiende de la API Java Servlet y utiliza un Modelo, Vista, Controlador.

**Vista**: Es la representación del modelo en un formato adecuado para que el usuario pueda interactuar con él, casi siempre es la interfaz de usuario.

## BIBLIOGRAFÍA


1. **BAUER C.Gavin K.**,Hibernate in Action Practical Object/Relational Mapping. Greenwich - EstadosUnidos, Manning, © 2004. Pp.29 - 50.

2. **MANN K.,**Java Server Faces in Action. Greenwich - EstadosUnidos, Manning, © 2005. Pp.3-87.

3. **RISBERG T., Evans R., y otros.,**Introduction to Spring MVC Developing a Spring Framework MVC application step-by-step, 2.ed, La Habana-Cuba. © 2004-2008. Pp. 5-62.

4. **COMPARING JAVA WEB FRAMEWORKS**
   http://es.slideshare.net/mraible/comparing-jsf-spring-mvc-stripes-struts-2-tapestry-and-wicket-presentation
   2013/05/15

5. **FRAMEWORKS**
   http://www.leepoint.net/notes-java/oodesign/frameworks.html
   2013/02/4

6. **HIBERNATE+JSF+SPRING**
   http://frameworksjava2008.blogspot.com/
   2013/03/11

7. **INSIGHTS STRUTS VS JSF PDF**
   http://ebookbrowse.com/090109184500-insights-struts-vs-jsf-pdf-d45007152
   2013/04/28

8. **INTEGRACIÓN DE JSF, SPRING E HIBERNATE PARA CREAR UNA APLICACIÓN WEB DEL MUNDO REAL**
   http://www.programacion.com/articulo/integracion_de_jsf_spring_e_hibernate_para_crear_una_aplicacion_web_del_mundo_real_307
   2013/02/1

9. **INTEGRACIÓN DE JSF-SPRING.HIBERNATE (JPA)**
   http://santiagotapiay.blogspot.com/2011/03/integracion-de-jsf-spring-hibernatejpa.html
   2013/02/9

10. **JAVASERVER FACES AND STRUTS: COMPETITION OR COEXISTENCE?**
    http://www.baychi.org/calendar/files/Struts-And-Faces/Struts-And-Faces.pdf
    2013/04/28
11. **JSF**
    http://proyectoremar.tripod.com/Documentos/Herramientas/JavaServerFaces.pdf
    2013/05/9

12. **JSF 2.0 + SPRING + HIBERNATE INTEGRATION EXAMPLE**
    http://www.mkyong.com/jsf2/jsf-2-0-spring-hibernate-integration-example/
    2013/03/19

13. **JSF - JAVA SERVER FACES (Y COMPARACIÓN CON STRUTS)**
    http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=jsf
    2013/04/17

14. **LA TECNOLOGÍA JAVASERVER FACES**
    http://www.programacion.com/articulo/introduccion_a_la_tecnologia_javaserver_faces_233
    2013/05/3

15. **MANUAL BÁSICO DE STRUTS**

http://www.programacion.com/articulo/manual_basico_de_struts_156/4
2013/05/8

16. **MVC EL PATRÓN MODELO-VISTA-CONTROLADOR (MVC)**
http://www.fdi.ucm.es/profesor/jpavon/poo/2.14.mvc.pdf
2013/03/5

17. **PATRÓN DE ARQUITECTURA MODELO VISTA CONTROLADOR (MVC)**
http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html
2013/02/11

18. **PRIMEFACES: FRAMEWORK SOBRE JSF 2.0. PRIMEROS PASOS.**
http://www.genbetadev.com/frameworks/primefaces-framework-sobre-jsf-2-0-primeros-pasos
2013/05/15

19. **SPRING**
http://static.springsource.org/spring/docs/2.5.x/spring-reference.pdf
2013/03/24

20. **SPRING 3 - PARTE 1: INTRODUCCIÓN**
http://www.javatutoriales.com/2010/09/spring-parte-1-introduccion.html
2013/03/28

21. **SPRING FRAMEWORK REFERENCE DOCUMENTATION**
http://static.springsource.org/spring/docs/3.1.0.M2/spring-framework-reference/html/
2013/03/9

22. **SPRING FRAMEWORK TUTORIAL**
http://static.springsource.org/downloads/tutorial.pdf
2013/03/3

23. **SPRING VS. STRUTS VS. JSF**
http://forum.springsource.org/showthread.php?19511-Spring-vs-Struts-vs-JSF
2013/04/13

24. **STRUTS AND JAVASERVER FACES**
http://www.pjug.org/JavaServerFaces.pdf
2013/05/20

25. **STRUTS IN ACTION: BUILDING WEB APPLICATIONS WITH THE LEADING JAVA FRAMEWORK**
http://www.amazon.com/dp/1930110502
2013/05/22

26. **STRUTS VS. JSF**
http://coyotevil.blogspot.com/2007/05/struts-vs-jsf.html
2013/05/6

27. **STRUTS VS JSF (A COMPARISON OF STRUTS AGAINST JSF)**
http://www.java-samples.com/showtutorial.php?tutorialid=748
2013/05/6

28. **TUTORIAL DE JAVASERVER FACES**
http://www.slideshare.net/ingeniods/manual-jsf
2013/05/8

29. **TUTORIAL DE JAVASERVER FACES**
http://www.sicuma.uma.es/sicuma/Formacion/documentacion/JSF.pdf
2013/05/8