Tabla de contenido

[1. GLOSARIO DE TERMINOS 4](#_Toc403771186)

[2. INTRODUCCIÓN 5](#_Toc403771187)

[2.1 Objeto del Documento. 5](#_Toc403771188)

[2.2 Equipo de Trabajo. 5](#_Toc403771189)

[3. DESCRIPCIÓN Y ENFOQUE DE LA SOLUCIÓN 6](#_Toc403771190)

[3.1 AUTOMATIZACIÓN DE PROCESOS 6](#_Toc403771191)

[4. FUNCIONALIDAD Y ALCANCE DEL SISTEMA 7](#_Toc403771192)

[4.1 Módulos resultantes 7](#_Toc403771193)

[4.1.1 Datos Generales 7](#_Toc403771194)

[4.1.2 Alumnos 7](#_Toc403771195)

[5. ARQUITECTURA 8](#_Toc403771196)

[5.1 Arquitectura Lógica 8](#_Toc403771197)

[5.2 Arquitectura J2EE 8](#_Toc403771198)

[5.2.1 Capa de Presentación 8](#_Toc403771199)

[5.2.2 Capa de interfaces de Servicios de Negocio 8](#_Toc403771200)

[5.2.3 Capa de Persistencia de Datos 9](#_Toc403771201)

[6. MEJORES PRÁCTICAS 10](#_Toc403771202)

[6.1 Compresor 10](#_Toc403771203)

[6.2 PMD (Programming Mistake Detector) 10](#_Toc403771204)

[6.3 Checkstyle 10](#_Toc403771205)

[6.4 Codetemplate 10](#_Toc403771206)

[6.5 Yslow 10](#_Toc403771207)

[6.6 JUnit 11](#_Toc403771208)

[6.7 Pruebas de Rendimiento y Cobertura 11](#_Toc403771209)

[6.8 Ant 11](#_Toc403771210)

[7. Calidad 12](#_Toc403771211)

[8. CONSIDERACIONES 13](#_Toc403771212)

# GLOSARIO DE TERMINOS

**Definiciones:**

Dentro del contexto del presente documento se definen los siguientes términos:

**TI**: Tecnología de la Información.

**JSLint**: Es un analizador online de código javaScript creado por Douglas Crockford que nos permitirá mostrar puntos en los que el código no cumpla unas determinadas reglas establecidas de “código limpio“.

**J2EE**: Comprende un conjunto de especificaciones y funcionalidades orientadas al desarrollo de aplicaciones empresariales.

**JBoos**: Es un servidor de aplicaciones Java EE de código abierto implementado en Java puro. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible la máquina virtual de Java.

**JSF**: JavaServer Faces (JSF) es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE.

**PrimeFaces**: Es un componente para JavaServer Faces (JSF) de código abierto que cuenta con un conjunto de componentes enriquecidos que facilitan la creación de las aplicaciones web. Primefaces está bajo la licencia de Apache License V2. Una de las ventajas de utilizar Primefaces, es que permite la integración con otros componentes como por ejemplo RichFaces.

**Ajax**: Es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página.

**EJB**: Los Enterprise JavaBeans (también conocidos por sus siglas EJB) son una de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE (ahora JEE 6.0).

**XML**: Es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

**JPA**: Java Persistence API, más conocida por sus siglas JPA, es la API de persistencia desarrollada para la plataforma Java EE. Es un framework del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la Plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE).

**Birt Report:** Es un código abierto un sistema de información basado en Eclipse que se integra con la aplicación Java / Java EE para producir informes convincentes.

# INTRODUCCIÓN

## Objeto del Documento.

Presenta la descripción de cada uno de los componentes de la arquitectura para el desarrollo del sistema de **Financiero** “Jano”.

Presenta la Arquitectura de Software, también denominada Arquitectura lógica que consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.

La arquitectura de la aplicación permite definir la organización interna de la misma estableciendo una funcionalidad en capas independientes y bien definidas (SoC Separation of Concern) que permiten tener mejor control de la ubicación de las reglas de negocio y por lo tanto facilita las tareas de mantenimiento de la mismas.

## Equipo de Trabajo.

Las personas que directa o indirectamente intervinieron fueron las siguientes:

|  |  |
| --- | --- |
| Nombre | Cargo |
|  |  |
|  |  |
|  |  |

**Por parte de JIPOVI participaron:**

* **Paul Jiménez**

# DESCRIPCIÓN Y ENFOQUE DE LA SOLUCIÓN

## AUTOMATIZACIÓN DE PROCESOS

XXXXX tiene la necesidad de optimizar y automatizar sus procesos de negocio, de apoyo y los docentes operativos para una mejor gestión y control de los mismos.

Esta propuesta involucra los siguientes componentes:

Servicios:

* Talleres para identificación de mejoras y aprovechamiento de oportunidades de automatización en procesos diarios de los docentes.
* Capacitación y acompañamiento para levantar los requisitos necesarios para iniciar el desarrollo de los correspondientes módulos.
* Implementación de 5 módulos con sus respectivos reportes de acuerdo a las características que se especifican en el alcance.
* Una interfaz amigable para el docente, para que puede conocer la información detallada de sus obligaciones y beneficios en la Universidad Central del Ecuador.
* Capacitación en metodología y herramientas que se implementen. .
* Los reportes que genera el sistema son en herramienta de software libre ***Bird Report*** la cual permite obtener informes convincentes en formatos de Excel, Word y PDF.

# FUNCIONALIDAD Y ALCANCE DEL SISTEMA

## Módulos resultantes

A continuación se presentan los módulos en que se divide la solución y automatización de los procesos de negocio del **Portafolio Docente** “Jano”, también se define el alcance de cada uno de estos módulos.

### Datos Generales

El módulo de datos generales al cual solo van a tener acceso el personal del con el rol indicado, va a tener la siguiente funcionalidad:

Permite el registro de los datos del docente:

1. Ingreso de nombres y apellidos completos.
2. Ingreso de dirección de domicilio.
   1. Ingreso de contactos
   2. Teléfono convencional
   3. Teléfono móvil
   4. Correo electrónico
3. Ingreso de las instituciones educativas en las cuales el docente está laborando.
4. Ingreso de las asignaturas que está actualmente impartiendo el docente
5. Guardar en una base de datos la información registrada correspondiente a este modulo

### Alumnos

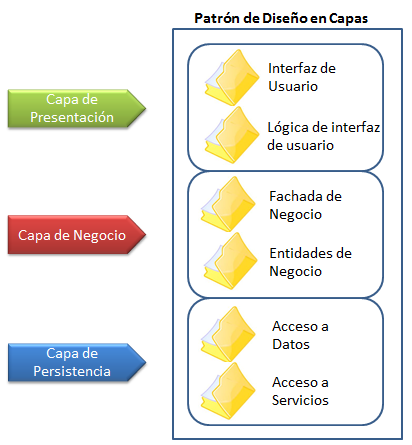
El módulo de siniestros debe tener la siguiente funcionalidad para el docente y los alumnos:

1. Clasificación de cada uno de los cursos a los cuales está dictando clases el docente.
2. Permitir ingresar el listado de la nómina de estudiantes de cada uno de los cursos.
3. Ingresar las notas correspondientes de cada uno de los alumnos.
4. Permitir llevar un sistema de calificación y evaluación de cada uno de los estudiantes.

# ARQUITECTURA

## Arquitectura Lógica

El patrón de arquitectura elegido para la construcción del sistema de **Portafolio Docente** “Jano” es el de niveles de abstracción o capas; este modelo de arquitectura se encuentra basado en la especificación J2EE y en patrones de diseño ampliamente usados, así como estándares abiertos.



## Arquitectura J2EE

### Capa de Presentación

A nivel de la capa de presentación Web se utilizará la especificación de PrimeFaces 4.0, permitiendo crear un ambiente Ajax de manera fácil, rápida y limpia.

Esta capa se encuentra compuesta por:

***Recursos***.- Se almacenan archivos de internacionalización de las páginas

***Control***.- Almacena los backing bean de las paginas JSF.

***Pages***.- Contiene las paginas JSF de cada módulo.

### Capa de interfaces de Servicios de Negocio

A nivel de la capa de interfaces de Servicios de Negocio se utilizará EJB 3.0 para la gestión (conexión) de los JavaBeans de negocio y persistencia, y el manejo transaccional de la aplicación.

Esta capa se encuentra compuesta por:

***Servicios***.- JavaBeans que contiene los métodos de negocio expuestos para su utilización desde la capa de presentación.

***Transacción***.- Maneja la transaccionabilidad de la aplicación.

***Interfaz***.- Funciona como intermediario entre las diferentes capas.

***Gestores***.- Contiene los JavaBeans que encierran la lógica del negocio.

***Recursos***.- Se almacenan las propiedades de configuración de la aplicación

### Capa de Persistencia de Datos

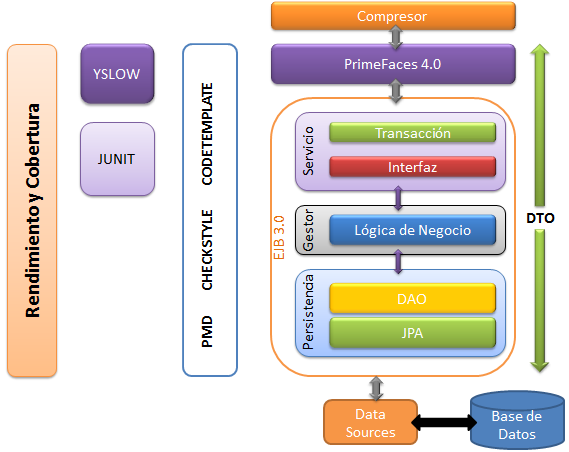
A nivel de la capa de Persistencia se utilizará JPA como herramienta de mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones.

Esta capa se encuentra compuesta por:

***Persistencia***.- Contiene los JavaBeans de los servicios persistentes.

***DTOs***.- Almacena los pojos que mapean las tablas de la base de datos y se los puede utilizar a nivel de todas las capas.

***DAOs***.- Un Data Access Object (DAO, Objeto de Acceso a Datos) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo.



# MEJORES PRÁCTICAS

## Compresor

Sobre la capa de presentación Web se utiliza un componente que permite comprimir las paginas HTML desde el servidor de aplicaciones y con esto se logre mejorar el rendimiento en la aplicación.

## PMD (Programming Mistake Detector)

PMD es una herramienta que comprueba que nuestra aplicación cumpla una serie de reglas que nos ayudan a obtener un código más elegante, sencillo y mantenible.

PMD busca en el código Java posibles problemas en potencia como pueden ser:

* Posibles bugs (sentencias try/catch/finally/switch vacías).
* Código muerto (variables locales, parámetros y métodos privados que no se usan).
* Código no óptimo (mal uso de String/StringBuffer/StringBuilder).
* Expresiones excesivamente complicadas (sentencias if innecesarias, implementaciones con bucles while).
* Código duplicado (código copiado y pegado sin valor agregado).

## Checkstyle

Checkstyle es una herramienta de desarrollo que ayuda a los programadores a escribir código Java que se adhiera a un estándar de codificación. Automatiza el proceso de chequeo de código Java, esto lo hace ideal para proyectos que desean aplicar un estándar de codificación.

Checkstyle es altamente configurable y puede soportar cualquier estándar de codificación. Un archivo de configuración de ejemplo es el proporcionado Oracle que posee las convenciones de codificación Java.

## Codetemplate

Codetemplate es una herramienta de desarrollo que ayuda a los programadores a escribir código Java que se adhiera a un estándar de comentarios.

Codetemplate es altamente configurable y puede soportar cualquier estándar de comentarios.

## Yslow

YSlow analiza las páginas web y sugiere formas de mejorar el rendimiento, en base a un conjunto de 34 reglas de optimización con las que se puede mejorar el rendimiento de la parte web y hacerla considerablemente más rápida. Las 34 normas se dividen en 6 categorías.

Características:

* Información sobre el rendimiento de la aplicación web.
* Resumen HTTP/HTML
* Listado de los componentes de una página.
* Herramientas como JSLint.

YSlow analiza cualquier página web y genera una calificación para cada regla definida. Si la página tiene aspectos mejorables, la extensión ofrece una lista de sugerencias con los cambios.

## JUnit

JUnit es un conjunto de bibliotecas que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java que se utilizara para los servicios significativos.

JUnit es un conjunto de clases (framework) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

JUnit es también un medio de controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación.

## Pruebas de Rendimiento y Cobertura

Permite mejorara el mantenimiento a futuro y estas pruebas se las hace una vez que la aplicación ya está en uso y devuelve resultados de rutinas de código que entorpecen el rendimiento de la aplicación utilizando herramientas como Jmeter o Performance Tester.

## Ant

Sirve para automatizar las tareas de despliegue (deployment) de una aplicación en diferentes ambientes. Suele usarse para tareas tales como compilar proyectos, empaquetar, versionar y publicar.

También se usa para tareas de automatización tales como correr pruebas de testing y tareas de procesamiento específicas de la aplicación.

# Calidad

Las consideraciones arquitectónicas y de diseño mostradas en este documento contribuyen a elevar la calidad del servicio de la siguiente manera:

**Escalabilidad y Confiabilidad**: Apoyada por el diseño modular y la separación en componentes, esta permite distribuir los componentes entre varios servidores en caso de ser requerido.

**Mantenimiento**: Los patrones arquitectónicos que se usan permiten que la solución sea lo suficientemente flexible separando claramente cada parte funcional de la solución para soportar cambios en una manera consistente y sencilla.

**Eficiencia**: La compresión y el uso del protocolo TCP/IP que permite codificación binaria, aseguran que las operaciones sean lo más eficientes en cuanto al uso de recursos tanto de red como de procesamiento.

# CONSIDERACIONES

La automatización, capacitación y acompañamiento en esta tarea es nuestro valor agregado al Proyecto.

El proceso de Mejoramiento, será un proceso posterior a la responsabilidad de JIPOVI en este proyecto, ya que el mismo dependerá de los hallazgos que se determinen luego de la puesta en producción de la automatización y la generación de datos de los reportes, lo cual permitirá volver a repetir el ciclo en función de la optimización.

Y en prueba de cuanto antecede en cuanto a la propuesta.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **AVILA & ASOCIADOS**  Gerente General  **Ing. José Avila Arroyo** |  | **JIPOVI SOLUTIONS**  Consultor  **Ing. Paúl Jiménez** |