



**ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO
FACULTAD DE INFORMÁTICA Y ELECTRÓNICA
ESCUELA DE INGENIERÍA EN SISTEMAS**

**“ESTUDIO COMPARATIVO DE LOS FRAMEWORKS TAPESTRY Y
WICKET PARA EL DESARROLLO DE APLICACIONES WEB. CASO
PRÁCTICO: INSTITUTO PARTICULAR SAN GABRIEL”.**

TESIS DE GRADO

Previa la obtención del título de:

INGENIERO EN SISTEMAS INFORMÁTICOS

Presentado por:

Edison Mauricio Jara Izurieta

RIOBAMBA – ECUADOR

2014

AGRADECIMIENTO

Agradezco a Dios por darme la sabiduría para tomar buenas decisiones y así alcanzar mis objetivos, a mi madre por su apoyo, amor, y por guiarme en cada momento de mi vida, a mi hermana por su apoyo incondicional.

A mi Director de tesis, Ingeniero Julio Santillán quien con sus conocimientos y experiencia me ha guiado en el desarrollo y en la culminación de mi tesis.

DEDICATORIA

Dedico este trabajo principalmente a mi Madre por darme su apoyo incondicional y consejos para continuar con mis estudios superiores, así como a Dios por darme la vida y paciencia para terminar mis estudios y la oportunidad de cumplir con un objetivo más en mi vida.

A mi hermana y amiga Mónica por brindarme su apoyo en todo momento durante toda mi vida estudiantil ya que ha sido muy importante en mi formación académica.

A toda mi familia y amigos por sus consejos y confianza para poder conseguir este objetivo, por su apoyo en todos los momentos más importantes de mi vida.

FIRMAS RESPONSABLES Y NOTA

NOMBRES	FIRMA	FECHA
Ing. Iván Menes Camejo DECANO FACULTAD DE INFORMÁTICA Y ELECTRÓNICA	_____	_____
Ing. Jorge Huilca DIRECTOR DE LA ESCUELA DE INGENIERIA EN SISTEMAS	_____	_____
Ing. Julio Santillán DIRECTOR DE TESIS	_____	_____
Ing. Paúl Paguay MIEMBRO DEL TRIBUNAL	_____	_____
DIRECTOR DEL CENTRO DE DOCUMENTACIÓN	_____	_____
NOTA DE LA TESIS: _____		

RESPONSABILIDAD DEL AUTOR

Yo, “Edison Mauricio Jara Izurieta, soy el responsable de las ideas, doctrinas y resultados expuestos en esta Tesis, y el patrimonio intelectual de la Tesis de Grado pertenecen a la ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO”.

Edison Mauricio Jara Izurieta

ÍNDICE DE ABREVIATURAS

API: Application Programming Interface

CRUD: Create, Read, Update and Delete

CSS: Cascading Style Sheet

IDE: Integrated Development Environment

IoC: Inversion of Control

J2EE: Java 2 Enterprise Edition

HTTP: Hyper Text Transfer Protocol

MSF: Microsoft Solution Framework

MVC: Modelo Vista Controlador

POJO: Plain Old Java Objects

OS: Sistema Operativo

SFSG: Sistema Financiero San Gabriel

TML: Tapestry Markup Language

XHTML: Extensible Hypertext Markup Language

W3C: World Wide Web Consortium

ÍNDICE GENERAL

PORTADA

AGRADECIMIENTO

DEDICATORIA

INTRODUCCIÓN

CAPITULO I

1.	Marco Referencial	16
1.1	Antecedente.....	16
1.2	Justificación del Proyecto de Tesis	17
1.2.1	Justificación Teórica.....	17
1.2.2	Justificación práctica	19
1.3	Objetivos	20
1.3.1	Objetivo General.....	20
1.3.2	Objetivos Específicos	20
1.4	Hipótesis.....	20

CAPITULO II

2.	Marco Teórico	21
2.1	Introducción	21
2.2	¿Qué es una Aplicación Web?	21
2.3	Java EE Framework capa de presentación	22
2.4	Arquitectura de Aplicaciones Web	23
2.5	Framework Apache Wicket	24
2.5.1	Introducción.....	24
2.5.2	Objetivos.....	25
2.5.3	Características.....	26
2.5.4	Wicket dice "¡Hola Mundo!"	27
2.5.5	Distribución y módulos Wicket.....	28
2.6.	Framework Apache Tapestry	29
2.6.1	Introducción.....	29
2.6.2	Historia	30
2.6.3	Versiones	33

2.6.4	Principios	33
2.6.5	Características.....	35
2.6.6	Desventaja de Tapestry.....	39
2.6.7	Páginas en Tapestry	40
2.7.	Diferencias Entre Wicket Y Tapestry.....	41
2.8.	Dependencias, Herramientas y Plugins	41
2.9.	Creación de la Estructura de una Aplicación Web	45
2.10.	Desarrollo de aplicaciones web con Tapestry.....	52
2.11.	CSS.....	56
2.12.	Ventajas de CSS	57

CAPITULO III

3	Estudio comparativo de los frameworks Tapestry y Wicket para el desarrollo de aplicaciones web.....	58
3.1.	Introducción	58
3.2.	Elaboración de prototipos con los frameworks	58
3.3.	Determinación de los parámetros de comparación.....	59
3.4.	Determinación de los indicadores de productividad	60
3.5.	Criterios de evaluación.....	60
3.6.	Análisis, interpretación y representación de resultados de los parámetros de comparación para los frameworks Wicket y Tapestry.	61
3.6.1	Líneas de código	61
3.6.2	Curva de Aprendizaje	64
3.6.3	Reutilización de Componentes	66
3.6.4	Documentación.....	67
3.6.5	Diseño.....	77
3.6.6	Tiempo de Desarrollo	79
3.6.7	Resumen de la comparación de los parámetros	81
3.7.	Comprobación de la Hipótesis	82
3.7.1	Hipótesis	82
3.7.2	Tipo de Hipótesis.....	82
3.7.3	Determinación de las variables	83
3.7.4	Operacionalización Conceptual.....	83

3.7.5	Operacionalización Metodológica.....	83
3.7.6	Justificación y resultado de la hipótesis.....	84

CAPITULO IV

4	Implementación del Sistema Financiero en el Instituto Particular San Gabriel.	91
4.1	Introducción	91
4.2	Desarrollo del Sistema Financiero San Gabriel (SFSG)	92
4.2.1	Fase de Visión.....	92
4.2.2	Fase de Planificación	94
4.2.3	Requerimientos No Funcionales.....	100
4.2.4	Actores.....	101
4.2.5	Fase de Desarrollo	108
4.2.6	Fase de Estabilización.....	110
4.2.7	Fase de Implementación	110

CONCLUSIONES

RECOMENDACIONES

RESUMEN

SUMMARY

GLOSARIO DE TÉRMINOS

ANEXOS

BIBLIOGRAFÍA

ÍNDICE DE TABLAS

Tabla II.I Módulos de Wicket.....	29
Tabla II.II Versiones de Tapestry	33
Tabla III.III Parámetros de comparación	59
Tabla III.IV Indicadores de productividad.....	60
Tabla III.V Criterios de Evaluación General.....	61
Tabla III.VI Criterios de evaluación para el parámetro Líneas de código.....	63
Tabla III.VII Parámetro Líneas de Código.....	63
Tabla III.VIII Criterios de evaluación para el parámetro Curva de Aprendizaje	65
Tabla III.IX Parámetro Curva de Aprendizaje.....	65
Tabla III.X Criterios de evaluación para el parámetro Reutilización de Componentes ..	66
Tabla III.XI Parámetro Reutilización de Componentes	67
Tabla III.XII Criterios de evaluación para el indicador Cantidad de Libros	68
Tabla III.XIII Indicador Cantidad de Libros.....	68
Tabla III.XIV Criterios de evaluación para el indicador Idioma de la Documentación ..	70
Tabla III.XV Indicador Idioma de la Documentación.....	70
Tabla III.XVI Criterios de evaluación para el indicador Videos Demostrativos.....	71
Tabla III.XVII Indicador Videos Demostrativos	71
Tabla III.XVIII Criterios de evaluación para el indicador Foros	73
Tabla III.XIX Indicador Foros	73
Tabla III.XX Criterios de evaluación para el indicador Páginas Oficiales.....	74
Tabla III.XXI Indicador Páginas Oficiales	74
Tabla III.XXII Resumen del parámetro Documentación	76
Tabla III.XXIII Criterios de evaluación para el parámetro Diseño.....	78
Tabla III.XXIV Parámetro Diseño.....	79
Tabla III.XXV Criterios de evaluación para el parámetro Tiempo de desarrollo	80
Tabla III.XXVI Parámetro Tiempo de desarrollo	80
Tabla III.XXVII Resumen de comparación entre los framework Wicket y Tapestry ...	81
Tabla III.XXVIII Operacionalización Conceptual.....	83
Tabla III.XXIX Operacionalización Metodológica.....	83
Tabla IV.XXX Hardware Existente	95

Tabla IV.XXXI Hardware Requerido.....	95
Tabla IV.XXXII Software Existente.....	95
Tabla IV.XXXIII Software Requerido.....	96
Tabla IV.XXXIV Recurso Humano Requerido.....	96
Tabla IV.XXXV Recurso Humano participativo del sistema.....	96
Tabla IV.XXXVI Nomenclatura y Estándares.....	108
Tabla IV.XXXVII Diccionario de Datos.....	108

ÍNDICE DE FIGURAS

Figura II.1 Estructura de clases MVC	24
Figura II.2 Estructura Java.....	28
Figura II.3 Historia de Tapestry.....	32
Figura II.4 Tapestry Orientado a Componentes.....	37
Figura II.5 Plantillas de Tapestry.....	40
Figura II.6 Crear variables de entorno	43
Figura II.7 Variables de Entorno	44
Figura II.8 Edición de variables.....	44
Figura II.9 Comando cmd.....	45
Figura II.10 Creación de una aplicación basada en Ant desde cero	46
Figura II.11 Creación de una aplicación basada en Maven desde cero	47
Figura II.12 Creación de una aplicación basada en el Arquetipo Maven	48
Figura II.13 Selección del Arquetipo del Framework	49
Figura II.14 Desarrollo de aplicaciones web con Wicket.....	50
Figura II.15 Esqueleto de una Aplicación Wicket.....	50
Figura II.16 Estructura de la aplicación Wicket en el Netbeans.....	51
Figura II.17 Pantallas de inicio de Wicket.....	51
Figura II.18 Selección del Arquetipo de Tapestry	52
Figura II.19 Esqueleto de una Aplicación en Tapestry.....	53
Figura II.20 Estructura de una Aplicación Tapestry en Netbeans	54
Figura II.21 Pantalla de Inicio de Tapestry.....	55
Figura II.22 Configuración de archivos TML en Tapestry.....	55
Figura III.23 Parámetro de Líneas de código.....	64
Figura III.24 Parámetro de Comparación Curva de Aprendizaje.....	65
Figura III.25 Parámetro de comparación Reutilización de Componentes.....	67
Figura III.26 Indicador Cantidad de Libros	69
Figura III.27 Indicador Idioma de la Documentación	70
Figura III.28 Indicador Videos Demostrativos	72
Figura III.29 Indicador Foros.....	73
Figura III.30 Indicador Páginas Oficiales.....	75

Figura III.31	Parámetro de comparación Documentación.....	76
Figura III.32	Parámetro de comparación Diseño	79
Figura III.33	Parámetro de comparación Tiempo de Desarrollo	80
Figura III.34	Resumen de comparación entre los framework Wicket y Tapestry	82
Figura IV.35	Fases de la Metodología MSF.....	92
Figura IV.36	Caso de Uso Administrador del Sistema	102
Figura IV.37	Diagrama de Secuencia Autenticación de Usuarios.....	102
Figura IV.38	Diagrama de Clases.....	103
Figura IV.39	Pantalla de Inicio de sesión en el sistema SFSG.....	104
Figura IV.40	Diagrama de Actividades para manipular el sistema SFSG.....	105
Figura IV.41	Diagrama de componentes.....	105
Figura IV.42	Diagrama de implementación.....	106
Figura IV.43	Modelo Físico.....	107
Figura IV.44	Sistema de Autenticación en MYSQL	109

INTRODUCCIÓN

Actualmente el uso y el desarrollo de las aplicaciones web han tenido gran aceptación por parte de clientes y proveedores, acorde al creciente del uso de los recursos de internet para realizar distintas tareas creciendo así la demanda de nuevos productos que faciliten la labor y reduzcan el tiempo de elaboración de cada una de sus tareas cotidianas. Por esta razón, la mayoría de empresas dedicadas al desarrollo de aplicaciones web han aumentado nuevos productos y frameworks que permitan compensar esta demanda.

El desarrollo de aplicaciones web pretende usar una diversidad de tecnologías para cumplir con los requerimientos del programador, para lo cual se han elaborado herramientas conocidas como frameworks, las cuales ayudan a los desarrolladores centrarse en los requerimientos del sistema, brindando funcionalidades habituales. El crecimiento de la cantidad de frameworks, cohibe la elección de éstos a los programadores comprometidos en iniciarse en éste ambiente de trabajo, cada uno de estos frameworks trabajan en entornos distintos, proporcionando ventajas y desventajas en el desarrollo.

Por esta razón se pretende hallar el framework más apropiado para el desarrollo de aplicaciones web, asumiendo la productividad con las que se elaboran las aplicaciones. En esta investigación se topa con la dudosa elección de dos frameworks para el desarrollo de aplicaciones web, para lo cual se pretende desarrollar prototipos para cada framework y escoger el más adecuado para el desarrollo de aplicaciones web en Java, con el objetivo de elaborar una comparación para indicar los problemas localizados, y así contribuir con aportaciones para el desarrollo de aplicaciones web establecidas en los frameworks escogidos.

En el Capítulo I, se describen los antecedentes en el Marco Referencial que fueron utilizados para esta investigación, se detalla los objetivos a conseguir al final del proyecto de tesis y la justificación de la hipótesis.

En el Capítulo II, se recalcan los conceptos más relevantes de esta investigación en el Marco Teórico, además la indagación sobre los frameworks de estudio, las clases, los

servicios y la programación de cada uno de ellos.

En el Capítulo III, permite establecer el mejor framework para el desarrollo de aplicaciones web en el Análisis comparativo, basándose en los parámetros e indicadores planteados que se relacionan con la productividad. Se puede justificar la hipótesis basándose en comparaciones de los prototipos realizados en los diferentes frameworks.

En el Capítulo IV, se detalla la metodología utilizada para el análisis y desarrollo de aplicaciones web, se elaboran diagrama de componentes, diagramas de casos de uso, diagrama de clases y el modelo físico de la base de datos, se detalla la implementación del sistema SFSG (Sistema Financiero San Gabriel) para el Instituto Particular San Gabriel de la ciudad de Riobamba de la provincia de Chimborazo.

CAPÍTULO I

1. Marco Referencial

1.1 Antecedente

En la actualidad existen varios frameworks y lenguajes de programación para desarrollar aplicaciones de calidad. El problema es que muchos programadores no tienen el hábito de utilizar patrones de diseño o estándares de desarrollo, debido a esto existen los problemas de compilación de proceso.

Es por eso que se ha adoptado la utilización de frameworks, que es la mejor forma de realizar un análisis, diseño, codificación de la aplicación a desarrollar.

Para esto se ha encontrado que en la actualidad en el INSTITUTO PARTICULAR SAN GABRIEL, no cuenta con un sistema financiero, lo cual es necesario para poder facilitar a los mismos un informe detallado de las cuentas por cobrar de los estudiantes.

Para este proyecto de tesis se ha seleccionado los siguientes frameworks MVC de Java: Tapestry y Wicket, porque son frameworks que permite la creación rápida de aplicaciones en un entorno web.

En este caso se ha buscado un problema con su cierto nivel de complejidad, para desarrollar un sistema que cumpla con las necesidades y requerimientos planteados en

dicho problema, y así poder reflejar las excelentes características que ofrece los frameworks MVC Tapestry y Wicket.

El principal propósito de estos frameworks es que implementan el patrón MVC (Modelo Vista, Controlador), es decir, divide el desarrollo en tres capas.

La capa de Modelo comprende los objetos de negocio (típico representan objetos almacenados en nuestras fuentes de datos).

La capa de Vista, es decir, la capa de presentación. Típico las aplicaciones web tienen tres tipos de presentación: las pantallas de información, las pantallas en las que un usuario introduce datos, y las pantallas donde se presentan los resultados solicitados.

La capa de Controlador, esta capa decide qué acciones se ejecutarán y cuál será la página de destino de estos resultados.

1.2 Justificación del Proyecto de Tesis

1.2.1 Justificación Teórica

La mayoría de instituciones no cuentan con normas, patrones o estándares de desarrollo de software. Las entidades que las poseen no saben cómo utilizarlas de forma óptima. La mejor forma es a través de la utilización de frameworks, la mayoría de empresas e instituciones probablemente deban manejar más de un marco de trabajo, es por eso que se pretende utilizar los frameworks MVC Tapestry y Wicket, para aplicaciones web.

Para realizar el estudio comparativo de los frameworks Tapestry y Wicket se tomará en consideración los parámetros necesarios, que permitirán evaluar las cualidades o falencias de cada una de las tecnologías escogidas con el fin de medir las mejores prestaciones de productividad en un determinado período de tiempo para el desarrollo del sistema financiero.

Podemos obtener diferentes medidas de productividad, evaluar diferentes prototipos de prueba, etc. Pero lo más importante sería ir definiendo los posibles parámetros de productividad como: curva de aprendizaje, líneas de código, tiempo de desarrollo y diseño.

En relación a los posibles parámetros de productividad, permitirán conocer y determinar las mejores prestaciones de productividad para el desarrollo del sistema financiero.

FRAMEWORK TAPESTRY

Este framework es un marco de código abierto para el desarrollo de aplicaciones web dinámicas en Java. El framework Tapestry compone y desarrolla la API estándar de Java Servlet y funciona en cualquier contenedor de servlets o servidor de aplicaciones web.

Tapestry separa una aplicación web en un conjunto de páginas, cada una elaborada a partir de componentes. Esto facilita una estructura consistente, admitiendo que el framework Tapestry tome el trabajo para las preocupaciones, así como la construcción de URL, almacenamiento de estado en el cliente o servidor, las validaciones de entrada en los campos de usuario, la localización / internacionalización y los mensajes de error o reportes de las excepciones.

Este framework está elaborado para desarrollar a partir de pequeñas aplicaciones inclusive aplicaciones masivas constituidas por miles de páginas particulares, elaboradas por grupos de trabajo dedicadas al desarrollo de aplicaciones web. Tapestry se puede integrar con cualquier prototipo de servidor, incluido JEE, HiveMind, Spring e Hibernate.

FRAMEWORK WICKET

El framework Wicket facilita una orientación orientado a objetos hacia la elaboración de aplicaciones dinámicas basadas en la Interfaz Gráfica Web. Gracias a que Wicket es Java puro y código HTML, se consigue aprovechar los conocimientos acerca de Java para desarrollar aplicaciones basadas en Wicket, reduciendo el tiempo de desarrollo.

Wicket es un framework de Java que tiene la simplicidad de separar las inquietudes y la disposición de desarrollo a un nivel totalmente nuevo. Las páginas Wicket se pueden ver solo como vista previa y a continuación se pueden revisar utilizando herramientas de diseño estándar como WYSIWYG HTML. El proceso de contenido dinámico y administración de formularios es manipulado en el código Java utilizando un modelo de

componentes de primera clase POJO, respaldo de beans de datos que fácilmente se puede conservar utilizando la tecnología preferida.

1.2.2 Justificación práctica

Es de vital importancia realizar una aplicación web en el INSTITUTO PARTICULAR SAN GABRIEL ya que permitirá automatizar la parte Financiera del Instituto, y al mismo tiempo nos permitirá comprobar las fortalezas de los frameworks en estudio, y de esta forma escoger y adoptar un framework para el desarrollo de las futuras aplicaciones dentro del INSTITUTO PARTICULAR SAN GABRIEL.

Para esto se ha encontrado que en la actualidad en el INSTITUTO PARTICULAR SAN GABRIEL, se requiere realizar un sistema con los siguientes módulos:

1. MODULO FINANCIERO

CUENTAS POR COBRAR DE CADA ESTUDIANTE

- Inscripción
- Matricula
- Derecho de Examine Principal
- Derecho de Examine de Suspensión
- Solicitudes

2. MODULO FACTURACIÓN

1.3 Objetivos

1.3.1 Objetivo General

Realizar un estudio comparativo de los frameworks Tapestry y Wicket para el desarrollo de aplicaciones web y aplicar en el desarrollo del sistema financiero en el Instituto Particular San Gabriel.

1.3.2 Objetivos Específicos

- Estudiar las tecnologías Tapestry y Wicket para el desarrollo de aplicaciones web.
- Definir Parámetros de Comparación y Medición de Productividad.
- Comparar los frameworks Tapestry y Wicket en base a la productividad para el desarrollador.
- Desarrollar el Sistema financiero utilizando el framework más apropiado para el desarrollo de la aplicación.

1.4 Hipótesis

El Framework Tapestry brinda las mejores prestaciones de productividad en el desarrollo del sistema financiero en el INSTITUTO PARTICULAR SAN GABRIEL.

CAPITULO II

2. Marco Teórico

2.1 Introducción

El uso de aplicaciones web en la actualidad es muy importante ya que las empresas o negocios requieren realizar actividades eficientes, manipulando menos recursos, ya que con las aplicaciones web se pueden realizar diferentes actividades desde distintos lugares no necesariamente en el lugar de trabajo, existen algunas herramientas y lenguajes de programación para desarrollar aplicaciones de calidad, es por ello que se ha adoptado la utilización de frameworks, que es una nueva modalidad de realizar un análisis, diseño, y codificación de la aplicación a desarrollar, lo difícil es al momento de elegir el framework más eficaz para crear aplicaciones de calidad.

JAVA es una tecnología que se usa para el desarrollo de aplicaciones web ya que cuenta con diversos frameworks que aceptan programaciones distintas, para dicho estudio comparativo se ha tomado estos dos frameworks: WICKET y TAPESTRY, por esta razón en este capítulo se tratará sobre definiciones, historia, características, ventajas, desventajas, herramientas de desarrollo, poniendo énfasis en la comparación de las características y componentes utilizados por los frameworks de estudio.

2.2 ¿Qué es una Aplicación Web?

“En la ingeniería de software se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una

intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web (HTML, JavaScript, Java, asp.net, php, etc.) en la que se confía la ejecución al navegador”. [23]

“Una aplicación web es un conjunto de páginas que interactúan unas con otras y con diversos recursos en un servidor web, incluidas bases de datos. Esta interacción permite implementar características en su sitio como catálogos de productos virtuales y administradores de noticias y contenidos. Adicionalmente podrá realizar consultas a bases de datos, registrar e ingresar información, solicitudes, pedidos y múltiples tipos de información en línea en tiempo real”. [23]

2.3 Java EE Framework capa de presentación

“Es adecuado contar con frameworks para aplicaciones web porque facilita la elaboración de las mismas, el principal problema que enfrenta un programador que no está familiarizado con estas tecnologías es la curva de aprendizaje, es decir, llegar a comprender su funcionamiento, por lo que pretende desarrollar un nuevo framework dedicado a aplicaciones web para facilitar el desarrollo en la capa de presentación, con la ayuda de una interfaz gráfica que guía al programador a generar código HTML y clases Java utilizando patrones de diseño y también hace mención a puntos de comparación entre frameworks Java EE de mayor uso que dan soporte a la arquitectura MVC”. [4]

“Java EE tiene varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, etc. y define cómo coordinarlos. Java EE también configura algunas especificaciones únicas para Java EE para componentes. Estas incluyen Enterprise JavaBeans, servlets, Java Server Pages y varias tecnologías de servicios web. Ello permite al desarrollador crear una Aplicación de Empresa portable entre plataformas, a la vez escalable e integrable con varias tecnologías”. [17]

Frameworks Java EE destacados:

- JSF (Java Server Faces)
- Struts (1.x y 2.x)
- Spring

- JBOSS SEAM
- GWT (Google Web Toolkit)
- Stripes
- Tapestry
- Wicket
- Maverick, etc.

2.4 Arquitectura de Aplicaciones Web

El Modelo Vista Controlador (MVC) es un modelo de arquitectura de software que divide los datos y la lógica de negocio de la interfaz de usuario de una aplicación web. Por esta razón MVC plantea la elaboración de tres componentes diferentes que son el modelo, la vista y el controlador [20].

Esta arquitectura se divide en tres capas:

Capa de Modelo: Gestiona el comportamiento y los datos del dominio de aplicación, responde a las solicitudes de información sobre su estado (por lo general de la vista), y responde a las instrucciones para cambiar de estado (por lo general desde el controlador).

Guarda de forma permanente los datos manejados por la aplicación hasta que se requiera su uso de nuevo, habitualmente en una base de datos relacional, es utilizada por la capa de lógica de negocio. Esta capa suele tener un servidor de bases de datos. [21]

Capa de Presentación: Se encarga de generar la interfaz de usuario y permitir la interacción. En las aplicaciones web la interfaz de usuario consiste en JSPs, HTML, etc., las imágenes, las hojas de estilo, javascript que se mostrarán al usuario a través del navegador con el que acceda el usuario a la aplicación. Se encarga de ser la interfaz entre el usuario y la lógica de negocio. En esta capa la aplicación se ejecuta en el navegador del usuario pero habitualmente se genera en el servidor [21].

Tiene tres tipos de presentación:

- Pantallas de información.

- Pantallas en las que un usuario introduce datos.
- Pantallas donde se presentan los resultados solicitados.

Capa de Controlador: Esta capa decide qué acciones se ejecutarán y cuál será la página de destino de estos resultados. El controlador está compuesto por diferentes entidades denominadas servicios que a su vez se encargan de una parte individual de la lógica, también suele incluir las entidades de dominio persistentes en una base de datos. Es utilizada por la capa de presentación y utiliza la capa de datos, esta capa se ejecuta en el servidor de aplicaciones junto con el framework web que genera el código para la capa de presentación. [21]

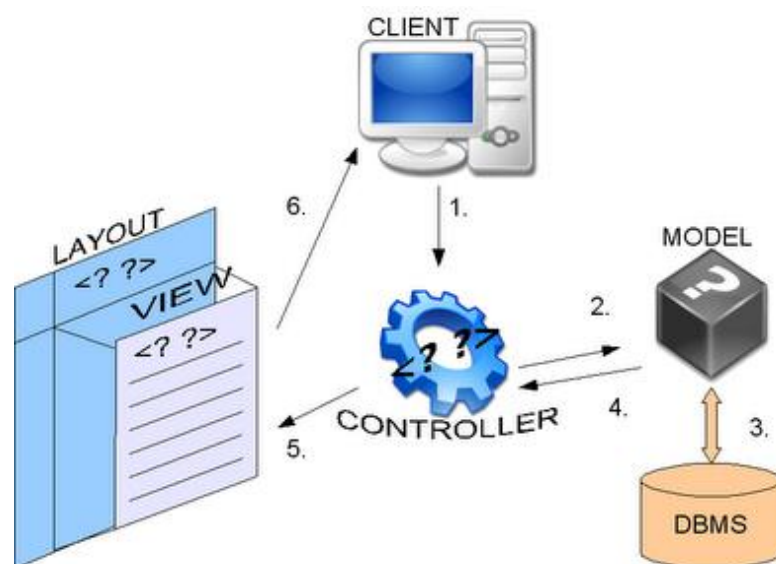


Figura II.1 Estructura de clases MVC

Fuente: <http://blog.zenphp.es/2008/03/19/patron-mvc-modificado/>

2.5 Framework Apache Wicket

2.5.1 Introducción

La programación de aplicaciones web en Java, tiene un gran número de opciones en estos días, de hecho hay muchos frameworks de aplicaciones web, Wicket es muy diferente a los frameworks que java proporciona.

Primos más cercanos de Wicket son probablemente Tapestry y Echo, pero incluso allí la

semejanza es muy superficial. Al igual que Tapestry, Wicket utiliza un atributo HTML¹ especial para denotar los componentes, lo que permite una fácil edición con editores de HTML normales. Como Echo, Wicket tiene un modelo de componentes de primera clase. Pero las aplicaciones Wicket no son como las aplicaciones escritas en Tapestry y Echo. [16]

Wicket es liberado bajo la Licencia de Apache Software 2.0.

2.5.2 Objetivos

Los objetivos para Wicket surgieron según lo observado en la página oficial de este framework [18]:

- Fácil (simple / consistente / obvio)
- POJO centrada.
- Minimizar superficie conceptual.
- Evita el uso excesivo de los archivos de configuración XML.
- Resolver un problema totalmente nuevo.
- Fácil de crear páginas bookmarkable.
- Diagnostico máximo de problemas en tiempo de ejecución.
- Dependencia mínima de herramientas especiales.
- Componentes, contenedores y convenciones deben ser coherentes.

- **Reusable**
 - Los componentes escritos en Wicket deben ser plenamente reutilizable
 - Componentes reutilizables deben ser fácilmente distribuidos en los archivos JAR ordinarias.

- **Eficiente / Escalable**
 - Eficiente y ligero, pero no a expensas de otros objetivos.
 - Agrupar a través de sesiones pegajosas preferidos.

¹ **HTML**: HyperText Markup Language

- **Completa**

- El equipo Wicket se compromete a entregar una función completa, lista para usar el framework para el desarrollo de aplicaciones web en Java.

2.5.3 Características

Wicket presenta las siguientes características las cuales se obtuvieron de la página oficial de este framework: [8]

Modelo de Componentes POJO

Páginas y componentes en Wicket son objetos reales de Java que soporte la encapsulación, la herencia y los acontecimientos.

Facilidad de Desarrollo

Debido a que Wicket es Java y HTML, puede aprovechar lo que sabes acerca de Java o su editor HTML favorito para crear aplicaciones Wicket.

Separación de las Preocupaciones

Los mundos de HTML y Java son paralelos y se asocian únicamente por los identificadores de Wicket, que son atributos en HTML y propiedades de componentes en Java. Wicket HTML es simplemente HTML y Wicket Java es Java, los programadores y diseñadores pueden trabajar de forma independiente, en gran medida, y sin depender de ninguna herramienta especial.

Multi-tab y soporte multi-ventana

Wicket proporciona una manera fácil de escribir aplicaciones que soporta multi-ventana y varias fichas de uso desarrollador dejando reaccionar adecuadamente cuando los usuarios abren una nueva ventana o pestaña del navegador.

Componentes Reutilizables

Componentes reutilizables en Wicket son particularmente fáciles de crear. No sólo se puede ampliar con los componentes existentes de Java se extiende de palabras clave, pero también se pueden crear componentes del panel que se asocian a un grupo de componentes como una unidad reutilizable.

Apoyo a todas las características básicas de HTML

Wicket soporta etiquetas de imágenes, enlaces, formularios y todo lo que usted está acostumbrado a utilizar en el desarrollo de su aplicación web.

Imágenes dinámicas

Wicket hace uso de imágenes, el intercambio y la generación es muy fácil. Las imágenes dinámicas se pueden crear, simplemente implementando un método de pintura.

Localización

Páginas HTML, imágenes y cadenas todos los recursos se pueden localizar.

2.5.4 Wicket dice "¡Hola Mundo!"

Wicket nos permite diseñar nuestras páginas Web en términos de componentes y contenedores, como AWT² hace con el escritorio de Windows.

Ambos frameworks comparten la misma arquitectura basada en componentes: en AWT tenemos un equipo con Windows instancia que representa las ventanas físicas que contienen componentes GUI (como campos de texto, botones de opción, las áreas de dibujo, etc...), en Wicket tenemos una instancia de página web que representa la página web que contiene los componentes físicos HTML (imágenes, botones, formularios, etc.)

² **AWT:** Abstract Window Toolkit

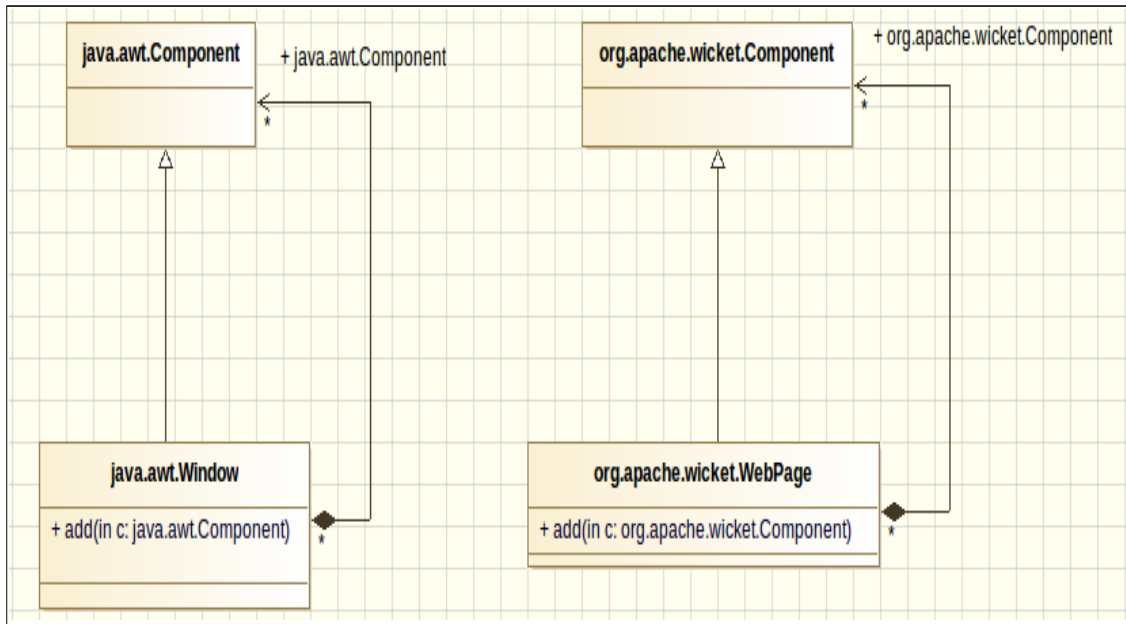


Figura II.2 Estructura Java

Fuente:

<https://ia801701.us.archive.org/14/items/WicketFreeGuide/Wicket%20free%20guide.pdf>

Páginas Wicket se pueden componer por muchos componentes, como las ventanas AWT están compuestos por componentes Swing / AWT.

Wicket ya viene con un completo conjunto de listas para utilizar los componentes, la construcción de componentes personalizados es una práctica común cuando se trabaja con este framework. [26]

2.5.5 Distribución y módulos Wicket

Wicket está disponible como un paquete binario, dentro de este archivo se encuentran los jars de distribución del framework. Cada jar corresponde a un sub-módulo de la secuencia. La siguiente tabla informa de estos módulos junto con una breve descripción de su propósito y con las dependencias relacionadas con:

Tabla II.I Módulos de Wicket

Nombre del módulo	Descripción	Dependencias
wicket-core	Contiene las principales clases de la estructura, como componente de clase y de la aplicación.	wicket-request
wicket-request	Este módulo contiene las clases que participan en el proceso de peticiones web.	- wicket-util
wicket-util	Contiene clases de utilidad de propósito general para las áreas funcionales, tales como I / O, lang, manipulación de cadenas, seguridad, etc.	Ninguna

Fuente:

<https://ia801701.us.archive.org/14/items/WicketFreeGuide/Wicket%20free%20guide.pdf>

Tenga en cuenta que el módulo de núcleo depende de los módulos de servicios públicos y la solicitud, por lo tanto, no se puede utilizar sin ellos. [26]

2.6. Framework Apache Tapestry

2.6.1 Introducción

Apache Tapestry es un framework de código abierto para la creación de robustas aplicaciones Web dinámicas, altamente escalables en Java. Tapestry complementa y desarrolla el API³ Servlet Java estándar, por lo que funciona en cualquier contenedor de servlets o servidor de aplicaciones.

Tapestry divide una aplicación Web en un conjunto de páginas, cada una construida a partir de componentes. Esto proporciona una estructura coherente, persistente almacenamiento de estado en el cliente o en el servidor, la validación de entrada de usuario, localización / internacionalización y el informe de excepciones. Las

³ **API:** Interfaz de Programación de Aplicaciones

aplicaciones Tapestry implican la creación de plantillas HTML con HTML plano, y la adición de una pequeña clase en java para cada uno, crea la aplicación en términos de objetos, así como los métodos y las propiedades de los objetos, trae el verdadero desarrollo orientado a objetos a las aplicaciones web de Java.

Tapestry está diseñado específicamente para que la creación de nuevos componentes, ya que este es un enfoque habitual en la construcción de aplicaciones. Posee una arquitectura escalar desde pequeñas aplicaciones, de una sola página hasta el final hasta aplicaciones masivas que consisten en cientos de páginas individuales, desarrollados por grandes y diversos equipos. Tapestry integra fácilmente con cualquier tipo de servidor, incluyendo JEE, Spring e Hibernate.

Es un entorno muy productivo, a los desarrolladores de Java les encantan porque pueden hacer cambios en el código Java y ver de inmediato, no redistribuir sin reiniciar. Y es increíblemente rápido para arrancar (incluso cuando los archivos se han cambiado), a los diseñadores les encanta porque las plantillas Tapestry están tan cerca de HTML normal, pero con extensiones tml⁴. [15]

Tapestry es liberado bajo la Licencia de Apache Software 2.0.

2.6.2 Historia

Un poco de historia sobre el framework Tapestry se pudo obtener del libro PlugIn Tapestry [21].

El framework fue ideado por Howard Lewis Ship (HLS) en el año 2000 cogiendo ideas similares al Web Objects de esa época. En el 2006 se gradúa como un proyecto de alto nivel de la fundación Apache. HLS en el año 2010 recibe el premio Java Champion.

En cada nueva versión la forma de hacer las cosas cambian aplicando nuevas ideas que facilitan el desarrollo, estos cambios hacían que el paso de una versión mayor a otra no

⁴ **TML**: Tapestry Markup Language

fuese simple. Ya en la versión 5 este problema se soluciona en gran medida y los cambios se limitan a no usar las cosas marcadas como obsoletas o que fueron quitadas.

Tapestry 3

En esta versión los componentes poseen 2 archivos como mínimo, uno para el código Java y otro para la especificación del componente page para la especificación de la página aunque normalmente suele incluir otro más para la plantilla que genera el html. Además, pueden necesitar otros recursos de estilos, imágenes o archivos de internacionalización. Para hacer un nuevo componente se ha de utilizar herencia extendiendo de las clases de Tapestry.

Tapestry 4

Esta versión hace varias aportaciones entre las principales incluir su propio contenedor de dependencias, Hive- mind, también desarrollado por HLS, que hace uso de xml para su configuración.

Tapestry 5

Esta versión sigue suponiendo una nueva ruptura con la versión anterior. Las mejoras que incluye son numerosas, se hace un uso extensivo de las nuevas características de Java como las anotaciones y genéricos y se desarrolla un módulo de contenedor de dependencias más integrado con el framework. Ahora en vez de usar un xml para la definición del contenedor IoC⁵ se usan clases Java.

Se proporcionan numerosas anotaciones que describen las clases y que en tiempo de ejecución añaden la funcionalidad. Se hace políglota soportando cualquier lenguaje ejecutable en la JVM. Deja de usar las expresiones ognl en las plantillas y desarrolla un lenguaje de expresiones similar. Se liberan varias versiones menores 5.1, 5.2, 5.3, 5.7 en los que cambiar a la nueva versión es poco más que actualizar las dependencias, las actualizaciones son mucho más pacíficas gracias a las anotaciones y la no herencia.

⁵ **IoC**: Inversion of Control (Inversión de Control)

Ha sido un líder desde una perspectiva puramente tecnológica. Estas son algunas cosas que hizo primero y todavía su autor, Howard Lewis Ship, piensa que lo hace mejor que nadie:

- Componentes reusables (2001)
- Detallado y útil informe de excepciones (2001)
- Instrumentación invisible en plantillas (2002)
- Informe de excepción con líneas precisas (2004)
- Meta programación de bytecode integrada (2005)
- Recarga en caliente de cambios (2006)
- Informe completo para errores en peticiones Ajax (2012).

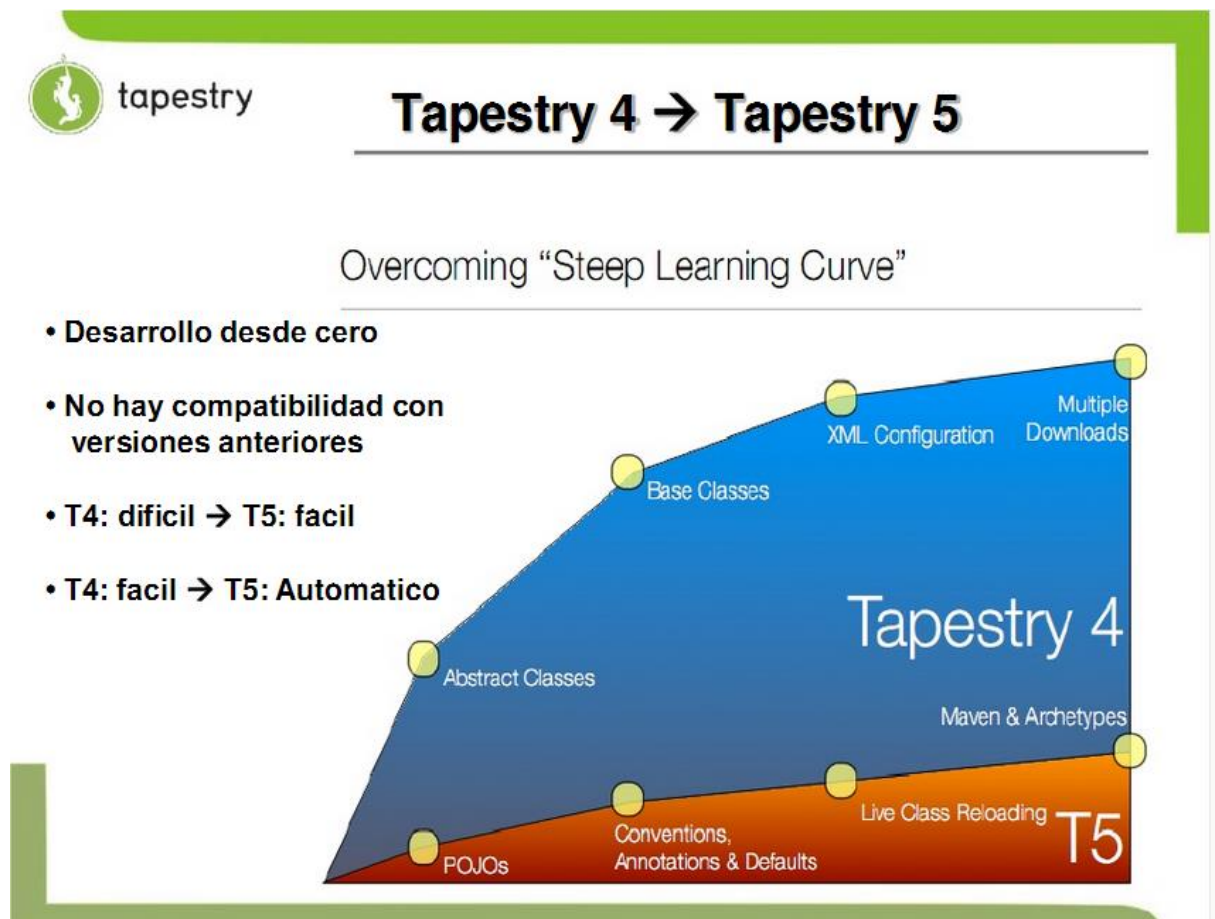


Figura II.3 Historia de Tapestry

Fuente: <http://es.scribd.com/doc/98793422/Tapestry-5>

2.6.3 Versiones

Estas notas de la versión describen los cambios en cada versión de Tapestry las cuales se pudieron obtener de la página oficial de Tapestry [24].

Tabla II.II Versiones de Tapestry

Liberación	Notas de la versión	Estado	liberado
Tapestry 5.4	Release Notes for 5.4	En desarrollo	
Tapestry 5.3.7	Release Notes for 5.3.7	Actual versión estable	24 Apr 2013
Tapestry 5.3.6	Release Notes for 5.3.6		12 Oct 2012
Tapestry 5.3.5	Release Notes for 5.3.5		29 Aug 2012
Tapestry 5.3.4	Release Notes for 5.3.4		16 Jul 2012
Tapestry 5.3.3	Release Notes for 5.3.3		24 Apr 2012
Tapestry 5.3.2	Release Notes for 5.3.2		07 Feb 2012
Tapestry 5.3.1	Release Notes for 5.3.1		21 Dec 2011
Tapestry 5.3	Release Notes for 5.3		21 Nov 2011
Tapestry 5.2.6	Release Notes for 5.2.x		22 Jun 2011
Tapestry 5.1	Release Notes for 5.1.x		12 Apr 2009
Tapestry 5.0	Release Notes for 5.0.x		Dec 2008

Fuente: <http://tapestry.apache.org/release-notes.html>

2.6.4 Principios

Los principios más relevantes sobre el framework Tapestry se pudo conseguir del libro PlugIn Tapestry [21].

Tapestry es un framework orientado a componentes para el desarrollo de aplicaciones web programadas en el lenguaje Java dinámicas, robustas y altamente escalables. Se sitúa en el mismo campo que Wicket y JSF en vez de junto con Spring MVC, Grails y Play, estos últimos orientados a acciones como Struts.

Los principios que guían el desarrollo de Tapestry son los siguientes.

Simplicidad

Esto implica que las plantillas y código sean legibles y concisos. También implica que no se extiende de las clases de Tapestry. Al ser las clases POJO (Plain Old Java Objects) se facilitan las pruebas unitarias y se evitan problemas al realizar actualizaciones a versiones superiores.

Consistencia

Significa que lo que funciona a un nivel, como en una página completa, funciona también para los componentes dentro de una página. De esta manera los componentes anidados tienen la misma libertad de persistir datos, anidar otros componentes y responder a eventos como en la página de nivel superior. De hecho los componentes y páginas se distinguen más bien en poco salvo por que los componentes están contenidos en otras páginas o componentes y las páginas están en el nivel superior de la jerarquía.

Feedback

Quizá sea la más importante. En los primeros días de los servlets y JSP la realidad era que en el mejor de los casos se obtenía una excepción en el navegador o el log del servidor en el momento de producirse una. Únicamente a partir de esa traza de la excepción se tenía que averiguar qué había sucedido. Tapestry proporciona más feedback que una excepción, genera un informe que intenta incluir toda la información necesaria para determinar la causa real. El informe incluye líneas precisas de código que pueden decir que tienes un error en la línea 50 de una plantilla y mostrar un extracto directamente en el informe de error además de los parámetros, algunos atributos de la request, las cabeceras y cookies que envió del navegador además de las variables de entorno, el classpath y atributos y valores de la sesión, por supuesto también incluye la traza de la excepción.

Eficiencia

A medida que ha evolucionado ha ido mejorando tanto en velocidad operando de forma más concurrente y en consumo de memoria. Se ha priorizado primero en escalar verticalmente que horizontalmente mediante clusters, que aún con afinidad de sesiones

complican la infraestructura considerablemente y supone retos a solucionar. Tapestry usa la sesión de forma diferente a otros frameworks tendiendo a usar valores inmutables y simples como números y cadenas que están más acordes con el diseño de la especificación de la API de los servlets.

2.6.5 Características

Las características más importantes del framework Tapestry se pudo obtener del libro PlugIn Tapestry [21].

A pesar de todo estos solo son principios y están en constante evolución, lo principal al añadir nuevas funcionalidades es cuán útiles son. En ocasiones añadir una funcionalidad implica mejorar un principio y bajar en otros. A continuación veamos unas cuantas de sus características más importantes.

Java

El lenguaje de programación empleado habitualmente para el código asociado a las páginas y componentes de programación es Java. Es un lenguaje orientado a objetos compilado a bytecode que a su vez es interpretado y ejecutado por la máquina virtual de Java (JVM, Java Virtual Machine). El bytecode es el mismo e independiente de la arquitectura de la máquina donde se ejecuta por la JVM. Esto permite que una vez escrito el código y compilado pueda ser ejecutado en cualquier máquina que tenga una JVM instalada. Esto nos permite desarrollar y producir el archivo war de una aplicación web en Windows o Mac para luego ser ejecutado en un servidor con GNU/Linux.

El compilador es una gran ayuda y una de sus razones de existencia además de producir bytecode es evitar que lleguen estos errores al momento de ejecución, para nada hay que menospreciar al compilador y sobrevalorar la interpretación, desde luego tampoco hay que confundir dinámico con ágil.

Políglota

Dicho en el apartado Java si prefieres programar los componentes y páginas en cualquier otro lenguaje soportado por la JVM es perfectamente posible. Tapestry acepta cualquiera

de ellos (Groovy, Scala, etc.).

No fullstack

El no ser un framework fullstack tiene ciertas ventajas y desventajas. Entre las desventajas es que al no darte un paquete tan completo y preparado deberás pasar un poco más de tiempo en seleccionar las herramientas que necesites para desarrollar la aplicación, como podría ser la herramienta de construcción del proyecto, la librería para hacer pruebas unitarias, de integración o para persistir los datos en la base de datos. Las ventajas son que tú eres el que elige las herramientas para trabajar y la forma de hacerlo son como tú decidas. Tapestry proporciona lo necesario para la capa de presentación (con el añadido del contenedor de dependencias) y tiene algunas librerías de integración con otras herramientas para persistencia con Hibernate, seguridad con Shiro, etc.

Basado en Componentes

Esta es la esencia de las aplicaciones de este framework, todos son componentes incluso las páginas lo son. Un componente es completamente autónomo, esto es, incluye en la página todo lo necesario para funcionar, como usuarios de uno no necesitaremos conocer más de él que los parámetros que necesita para usarlo. Las imágenes que vayan a mostrar, los textos localizados, las hojas de estilo y los archivos javascripts serán incluidos en la página únicamente en el caso de que se use, de manera que no se debe incluir previamente, ni globalmente y en todas las páginas todos los posibles recursos que se necesiten aunque en determinadas sepamos que algunos no son muy necesarios. Esto hace que las páginas sean más eficientes y carguen más rápido.

Los componentes son la forma de reutilizar código, con crear un componente podrás reutilizar ese código. También a través de ellos se consigue un alta productividad por si fuera poco los componentes pueden almacenarse en librerías y para tenerlos disponibles en una aplicación sólo será necesario incluir una dependencia en el proyecto o un archivo jar. Como son autónomos en el jar están todos los recursos necesarios, solo deberemos preocuparnos por sus parámetros y cómo usarlos. Tapestry se encargará de extraer del jar los recursos y servirlos. Si los componentes permiten reutilizar código en una misma aplicación, las librerías de componentes permiten reutilizar código en distintas

aplicaciones o por parte de terceros.

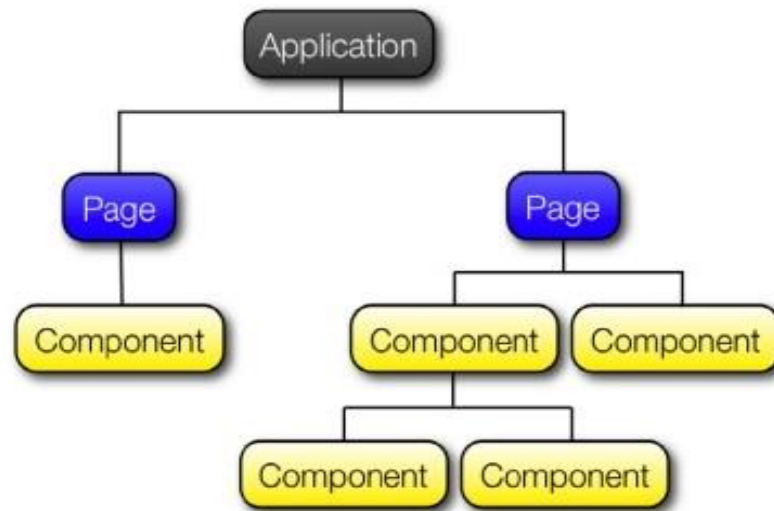


Figura II.4 Tapestry Orientado a Componentes

Fuente: <http://es.scribd.com/doc/98793422/Tapestry-5>

Modular, adaptable y extensible

Este es otro punto fuerte del framework. Usa su propio contenedor de dependencias (**IoC**, Inversión of Control) que viene incluido de serie en el framework encargándose de administrar los servicios, controlar su ciclo de vida, construirlos únicamente en el momento en que se necesitan y proporcionales las dependencias sobre otros servicios de los que hagan uso.

Usa su propio contenedor de dependencias porque no había ningún otro que permitiese una configuración distribuida, significa que para incluir nuevos servicios en el contenedor basta con dejar caer en la aplicación un jar y automáticamente los servicios y configuraciones que tenga ese jar serán administrados por el contenedor. Los servicios se definen en módulos, los módulos no son más que clases Java especiales usadas para conocer los servicios y contribuciones del módulo. También posee un potente sistema de configuración, los servicios se configuran mediante contribuciones que cualquier módulo puede hacer, un módulo puede hacer contribuciones a servicios de otros módulos.El

contenedor en el momento de construir el servicio le pasa el objeto con las contribuciones realizadas por cualquier módulo a ese servicio además de las dependencias que tenga sobre otros servicios.

Dado que mucha de la funcionalidad propia de Tapestry está proporcionada mediante servicios y que la implementación de un servicio puede reemplazarse por otra en el contenedor hace de él altamente adaptable y extensible tanto si necesitamos añadir nuevos servicios como si necesitamos que los existentes se comporten de otra forma, solo deberemos proporcionarle al contenedor la interfaz del servicio (sólo si es nuevo) y la implementación que deseamos. Con unas pocas líneas se puede personalizar casi todo aunque a veces puede llevar un tiempo saber cuáles son esas líneas.

Toda esta definición de servicios y configuraciones se hace a través de código Java con lo que tendremos la ayuda del compilador y cualquier error de escritura, al contrario de lo que ocurre en archivos xml, lo detectaremos rápidamente.

Convención sobre configuración

Las convenciones permiten evitar la configuración y los posibles errores que podemos cometer al realizarla. Pero más importante, hace que cualquier programador que conozca las convenciones sepa inmediatamente como están organizadas todas las cosas con lo que el tiempo de aprendizaje se reduce considerablemente. Por estos motivos Tapestry es un framework en el que se usan varias convenciones.

Documentado

Ya tiene una década y todo este tiempo ha servido para que tenga una documentación bastante extensa y de calidad que por sí sola sirve para aprender cada concepto de este framework de forma autodidacta. Además de la documentación de los conceptos está disponible el correspondiente Javadoc y la documentación de los componentes como consulta para el desarrollo.

Productivo

Tapestry es un framework con el que se es bastante productivo. Por la facilidad para

encapsular funcionalidad en componentes que son fáciles de crear y reutilizar. Por la recarga en caliente de los cambios que permiten ver los cambios inmediatamente evitando reinicios del servidor y esperas. Y por ser un framework que proporciona mucha información cuando se produce un error en forma de excepción que ayuda a resolver los problemas rápidamente.

2.6.6 Desventaja de Tapestry

Las desventajas más relevantes del framework Tapestry se obtuvo de la siguiente dirección [14].

Elevada Curva de Aprendizaje

Una de las desventajas siempre ha sido su dificultad de aprendizaje, los desarrolladores que recién inician con la utilización de este framework suelen verse frustrados mientras no se adaptan a las convenciones de nombrado al ver como algunas cosas parecen funcionar por arte de magia y otras no por un simple error en el nombre.

Incompatibilidad entre Versiones

La política de actualizaciones de Tapestry ha recibido en el pasado numerosas y merecidas críticas debido a la ruptura de compatibilidad entre las versiones 3 y 4, posteriormente repetida entre las versiones 4 y 5.

Actualmente, el equipo desarrollador ha anunciado que el rediseño recibido en la versión 5 (la actual) es el definitivo y se ha comprometido a que las actualizaciones futuras mantendrán la compatibilidad hacia atrás.

Estructura Estática

Tapestry usa una estructura de páginas semi-estática, donde el desarrollador puede incluir condicionales e iteraciones para conseguir un comportamiento dinámico. Pero la inexistencia de la posibilidad de definir estructuras y componentes en tiempo de ejecución imposibilita el uso de Tapestry para ciertas aplicaciones que necesiten una estructura de páginas totalmente dinámica.

2.6.7 Páginas en Tapestry

Cada página o componente es representada por una clase Java y una plantilla (XML⁶), ambas deben tener el mismo nombre, el objetivo de la clase java es recibir los eventos relativos a la página y el objetivo de la plantilla es generar la vista de la página.

Las plantillas de los componentes son:

- Documentos gramaticales correctos en XML
- No hay JSP
- La mayoría son archivos XHTML
- Tiene extensión “.tml”
- El XML incluirá referencias a componentes y también puede incluir expansiones. [19]

```
<html xmlns:t="...tapestry_5_0_0.xsd">
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>Welcome, ${userId}</p>

    <a t:type="pagelink"
      t:page="Start">
      Refresh
    </a>

  </body>
</html>
```

Figura II.5 Plantillas de Tapestry

Fuente: <http://es.scribd.com/doc/98793422/Tapestry-5>

⁶ **XML:** eXtensible Markup Language (Lenguaje de Marcado eXtensible)

2.7. Diferencias Entre Wicket Y Tapestry

Algunas diferencias son:

- Tapestry utiliza una estructura de página semi-estática, donde se puede trabajar con condicionales y bucles para lograr un comportamiento dinámico. Wicket es completamente dinámica, puede cargar dinámicamente componentes, sustituirlos en tiempo de ejecución, etc... Las consecuencias de esto son que tapestry es más fácil de optimizar.
- En Tapestry su código base es probablemente menor y es bueno para el mantenimiento lo que hace más fácil navegar con un IDE y comprobar con un compilador, mientras que en Wicket sus líneas de código son superior.

2.8. Dependencias, Herramientas y Plugins

Para configurar el entorno de desarrollo, en Java existen varias opciones disponibles como JDK, IDEs y otras herramientas, todas de código abierto y de libre acceso, que se tendrán que configurar.

JDK 1.5 o superior

Tanto Wicket como Tapestry hacen uso de las características del kit de desarrollo de Java (JDK). Esto incluye anotaciones Java, y un poco de los genéricos de Java.

Netbeans IDE 7.2.

NetBeans IDE es un entorno de desarrollo de código abierto, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. [22]

Para el desarrollo de los prototipos con los framework WICKET y TAPESTRY se utilizó Netbeans IDE 7.2.

Apache Tomcat

Apache Tomcat (o Jakarta Tomcat) es un contenedor de servlets de código abierto desarrollado bajo el proyecto Jakarta en Apache Software Foundation, es un software desarrollado con Java (con lo cual puede funcionar en cualquier sistema operativo, con su máquina virtual java correspondiente) que sirve como servidor web con soporte de servlets y JSPs.

Los dos prototipos se ejecutaron sobre Apache Tomcat.

Apache Maven 3

Apache Maven es un software de gestión de proyectos y una herramienta de comprensión, tiene un sistema de plugins muy sofisticado que le permite hacer prácticamente cualquier cosa, compilar código Java, la creación de archivos WAR y JAR, y la creación de informes y sitios web son su punto fuerte, la mayor ventaja de Maven es que se puede descargar dependencias del proyecto como, los archivos JAR necesarios para desarrollar las aplicaciones web.

Maven no es esencial para el uso de los framework Wicket y Tapestry, pero es especialmente útil cuando se realiza la configuración inicial de las aplicaciones web, y se encarga de descargar las dependencias necesarias y los framework de estudio. [11]

Configuración de Maven

Para configurar Maven se siguió los siguientes pasos de esta dirección [11], se debe ir a Configuración avanzada del sistema y seleccionar variables de entorno como se indica en la **Figura II. 6**.

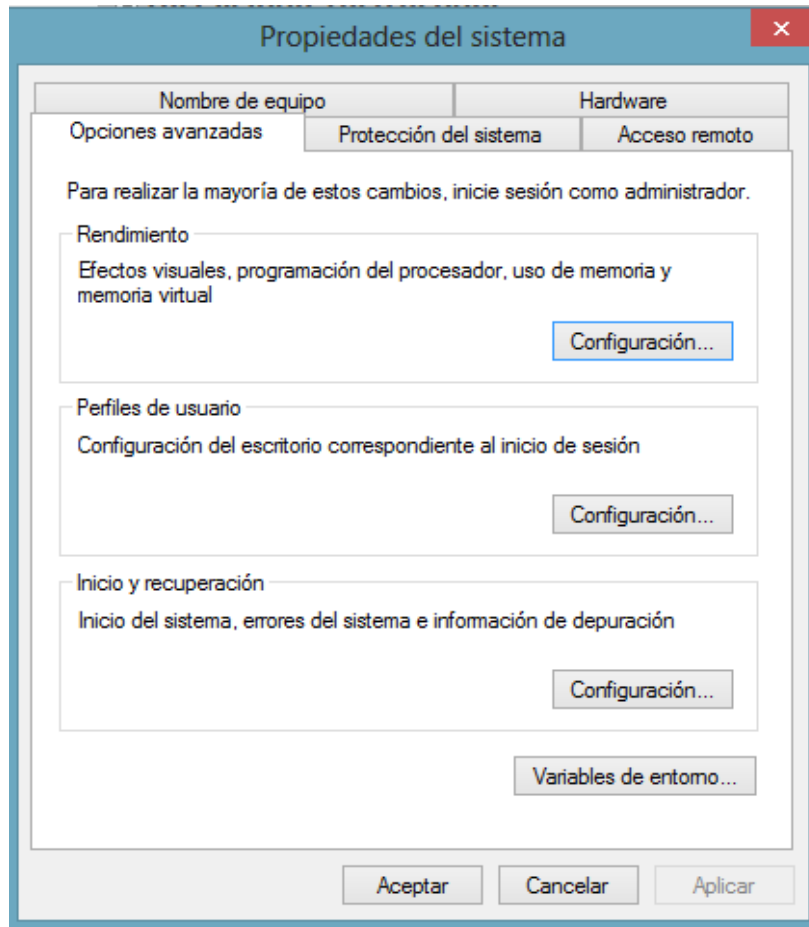


Figura II.6 Crear variables de entorno

Crear las variables de entorno JAVA_HOME, M2, M2_HOME y MAVEN_OPTS como se puede observar en la **Figura II. 7**.

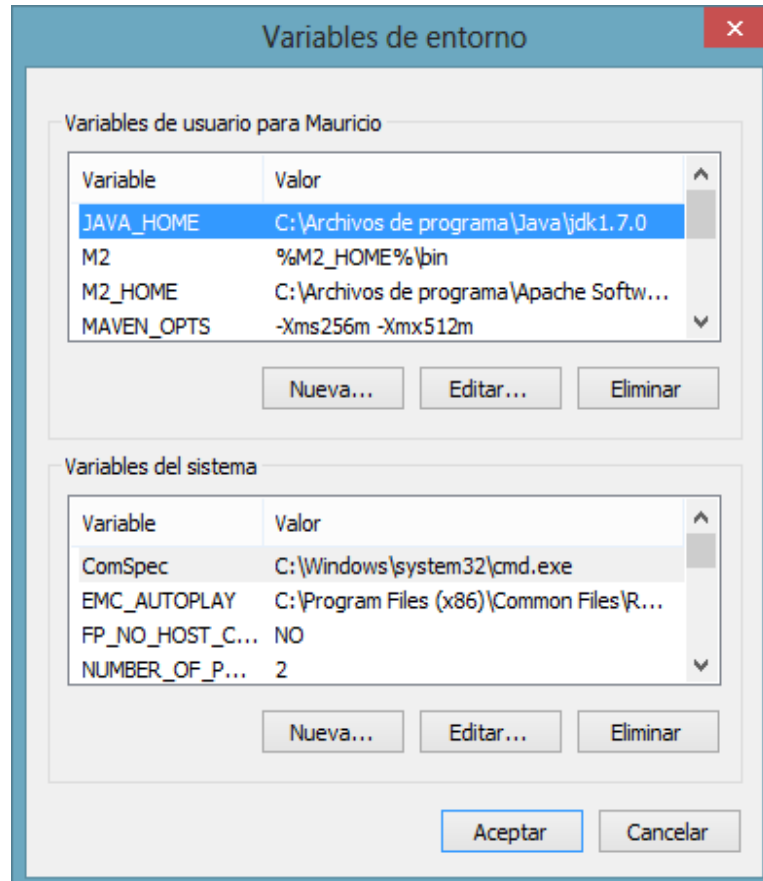


Figura II.7 Variables de Entorno

Crear la variable de usuario PATH y agregar los valores de la variable como se muestra en la **Figura II. 8**.

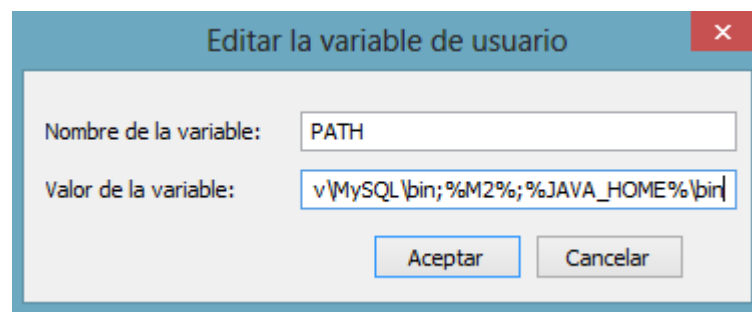
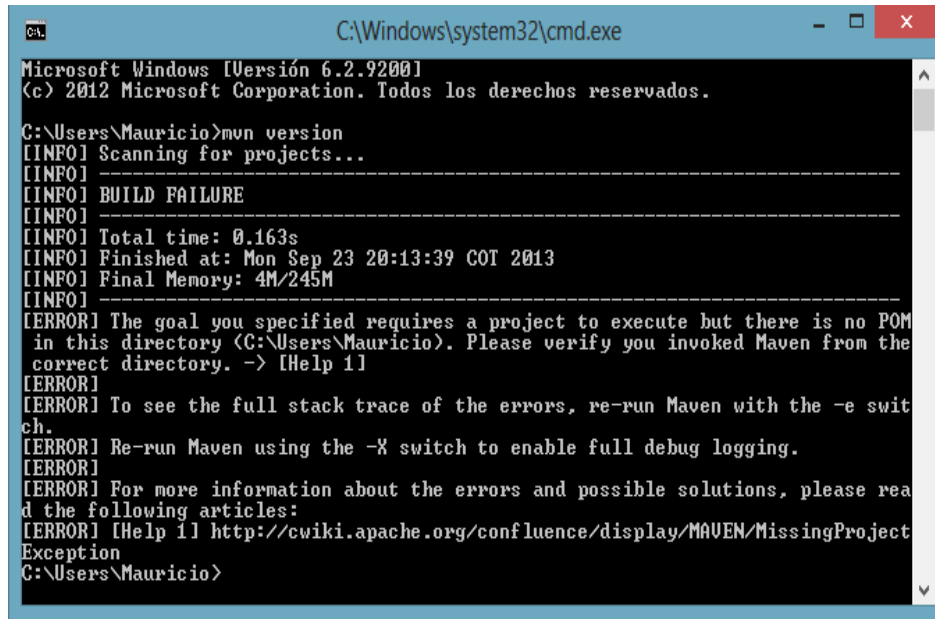


Figura II.8 Edición de variables

Luego abrir la consola cmd, en la cual se escribe mvn versión para instalar maven como se representa en la **Figura II.9**.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.2.9200]
(c) 2012 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Mauricio>mvn version
[INFO] Scanning for projects...
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 0.163s
[INFO] Finished at: Mon Sep 23 20:13:39 COT 2013
[INFO] Final Memory: 4M/245M
[INFO] -----
[ERROR] The goal you specified requires a project to execute but there is no POM
in this directory (C:\Users\Mauricio). Please verify you invoked Maven from the
correct directory. -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e swit
ch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please rea
d the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MissingProject
Exception
C:\Users\Mauricio>
```

Figura II.9 Comando cmd

2.9. Creación de la Estructura de una Aplicación Web

Para crear la estructura de una aplicación web tanto en Wicket como en Tapestry existen tres formas para crear, las cuales se puede observar de la siguiente dirección [13].

Creación de una aplicación basada en Ant desde cero

1. Cuando se crea una nueva aplicación web en el IDE, el último panel del Asistente para aplicaciones web va a ser muy útil en el contexto de nuestra Aplicación Web.
2. Seleccione Archivo> Nuevo proyecto. En Categorías, seleccione Web. En Proyectos, seleccione Aplicación Web. Haga clic en Siguiente.
3. En el nombre y el panel de direcciones, escribe MyFirstWicketApp en el nombre del proyecto. Cambie la ubicación del proyecto en cualquier directorio de su ordenador. Haga clic en Siguiente.
4. Deje todas las configuraciones sin cambiar. O, si lo desea, puede cambiarlos. Wicket es compatible con cualquier versión de Java EE. Una aplicación Wicket se puede implementar en cualquier servidor. Haga clic en Siguiente.

1. En el panel Frameworks, elija Wicket, como se muestra en la **Figura II.10**:

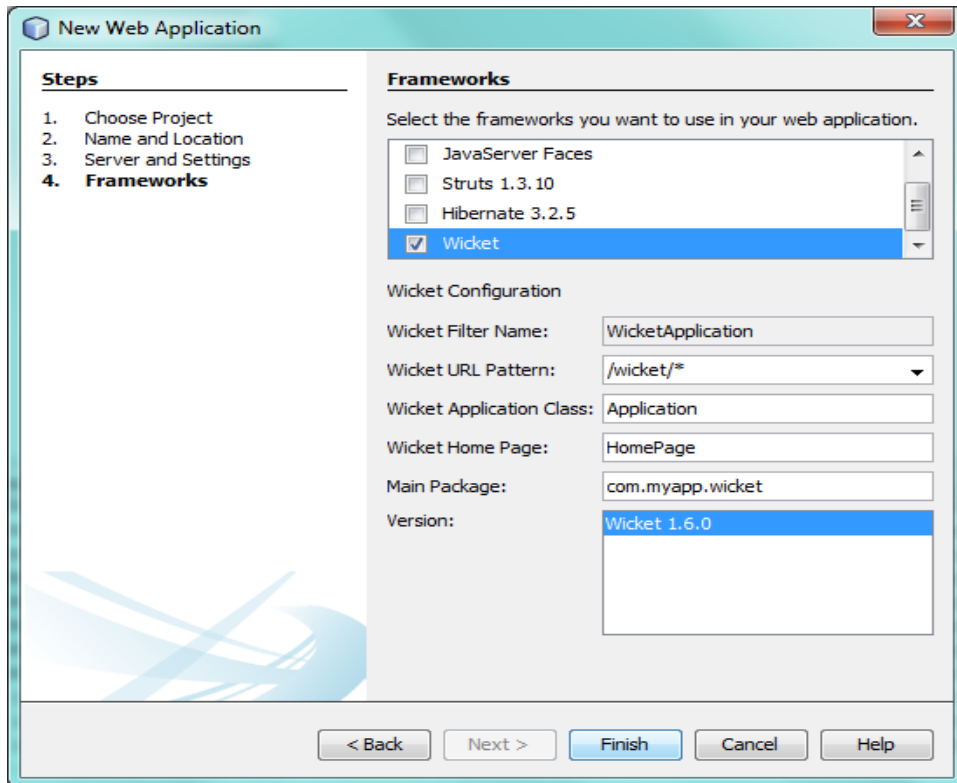


Figura II.10 Creación de una aplicación basada en Ant desde cero

En la Figura II.10, deje todos los valores por defecto sin cambios. Los campos en el panel proporcionan la siguiente información:

Wicket Filter Name: Muestra el nombre del filtro que se define en el web.xml archivo.

Wicket URLPattern: Agrega el patrón de URL relativa al web.xml archivo.

Wicket Application Class: Especifica el nombre de la clase en la que se establecen los valores de toda la aplicación, tales como la página de inicio.

Wicket HomePage: Especifica el nombre de la página de inicio, que consistirá en un archivo llamado java y html/tml.

Main Package: El paquete de Java en el que todos los artefactos generados se colocarán por el IDE.

Versión: La versión Wicket. Cualquier biblioteca en el Administrador de bibliotecas con un nombre que comienza con "Wicket" será incluido en la versión desplegable de arriba. De forma predeterminada, la única versión que aparece es 1.6.0, ya que es la versión proporcionada por el plugin.

Haga clic en Finalizar.

Creación de una aplicación basada en Maven desde cero

Si ya se tiene una aplicación basada en Ant, el IDE puede ayudar a añadir soporte Wicket o Tapestry.

1. Seleccione Archivo > Nuevo proyecto. En Categorías, seleccione Maven. En Proyectos, seleccione Aplicación Web.

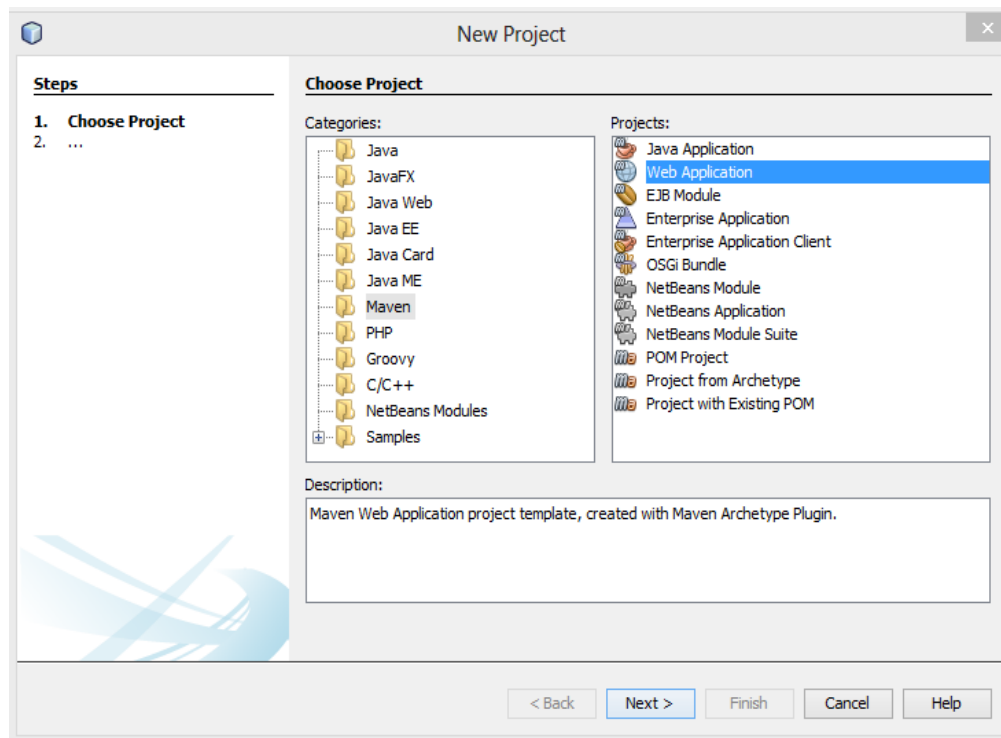


Figura II.11 Creación de una aplicación basada en Maven desde cero

1. En el nombre y el panel de direcciones, escribe MyFirstWicketApp en el nombre del proyecto. Cambie la ubicación del proyecto y los ajustes por defecto Maven como es requerido por sus necesidades.

2. Seleccione el servidor adecuado para sus necesidades, así como "Java EE 6 Web" para el ajuste Versión Java EE.
3. Agregar el archivo web.xml porque requiere un filtro la aplicación.
4. Agregar las dependencias necesarias en el archivo pom.xml.

Creación de una aplicación basada en el Arquetipo Maven

Existen arquetipos en los repositorios de Maven para la creación de aplicaciones web.

1. Seleccione Archivo> Nuevo proyecto. En Categorías, seleccione Maven. En Proyectos, seleccione Proyecto de Arquetipo.

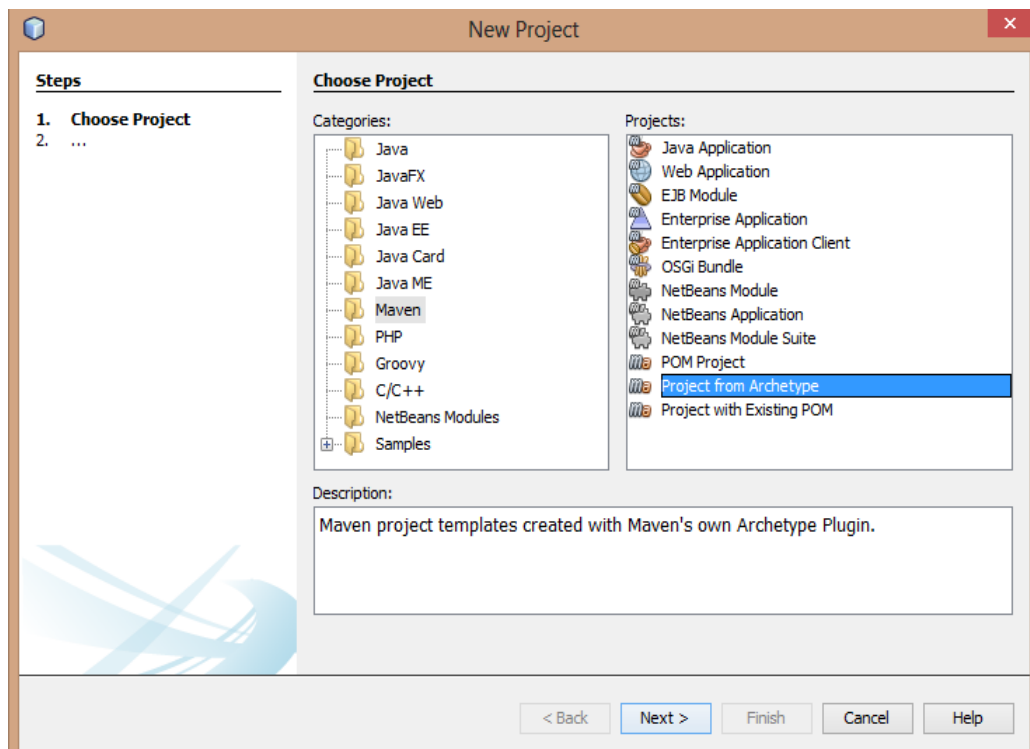


Figura II.12 Creación de una aplicación basada en el Arquetipo Maven

2. En el campo Buscar, escriba el nombre del framework y luego seleccione el arquetipo que le gustaría usar.

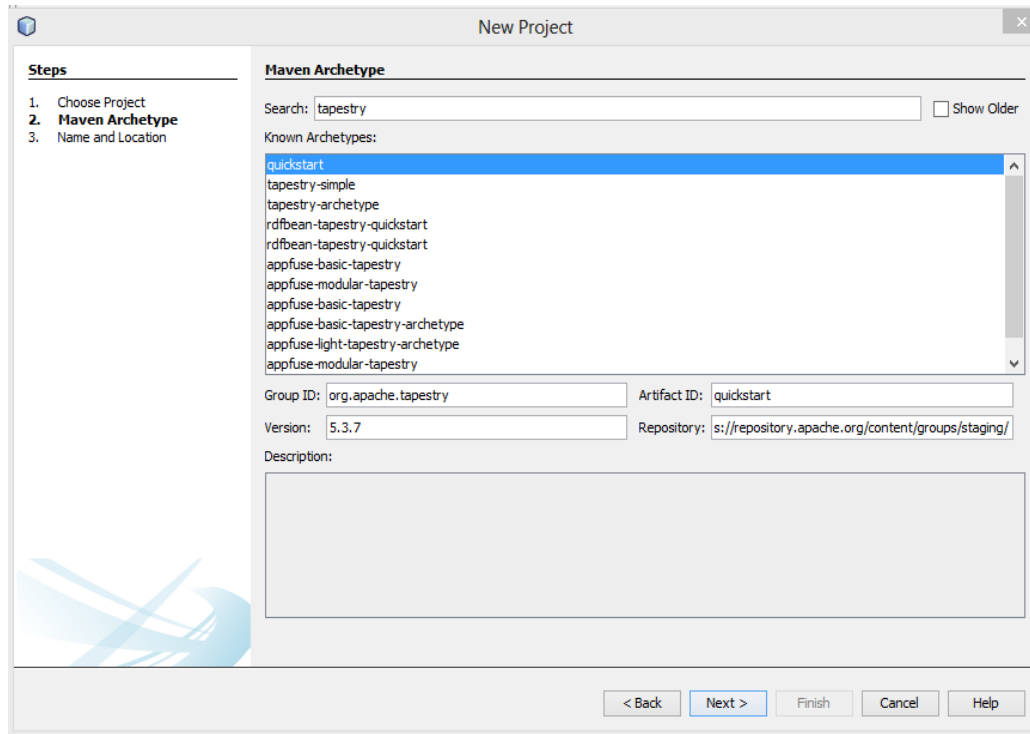


Figura II.13 Selección del Arquetipo del Framework

3. Complete el asistente con los valores apropiados a sus necesidades. Haga clic en Finalizar.

El IDE añade soporte a la aplicación creada a partir del arquetipo.

Desarrollo de aplicaciones web con Wicket

Para crear un proyecto con WICKET se debe tener instalado las herramientas necesarias:

- Tener instalado el JDK 1.7.
- Utilizar la herramienta Netbeans IDE 7.2.
- Tener instalado Maven 3.1.0.
- Configurar Apache Tomcat 7.0.27.
- Crear la aplicación basada en el arquetipo Maven mencionado anteriormente.
- Seleccionar el arquetipo a utilizar de Wicket 6.10.0.

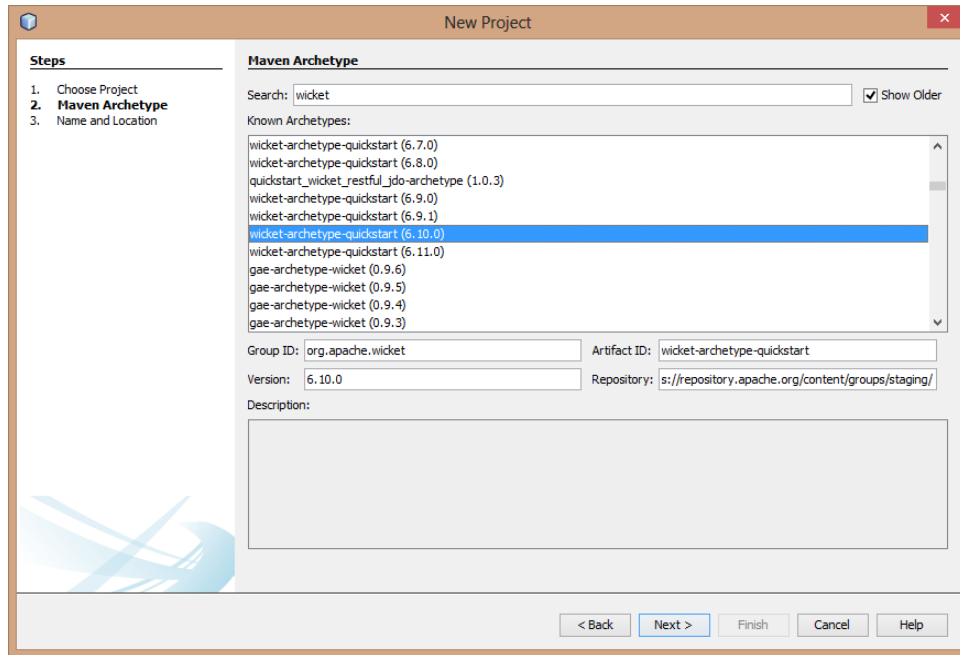


Figura II.14 Desarrollo de aplicaciones web con Wicket

En la **Figura II.14** se observa que al escribir el framework Wicket seleccionamos **wicket-archetype-quickstart** y automáticamente Maven crea el esqueleto de la aplicación web.



Figura II.15 Esqueleto de una Aplicación Wicket

Fuente: <http://wicket.apache.org/start/quickstart.html>

Maven al crear el inicio rápido Wicket 6.10.0 con Netbeans IDE 7.2, se crea con los siguientes archivos necesarios para empezar a trabajar con este framework, como se observa en la **Figura II.16**.

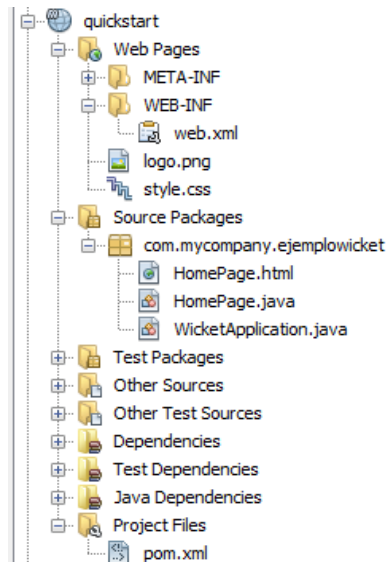


Figura II.16 Estructura de la aplicación Wicket en el Netbeans

Una vez creada nuestra estructura de la aplicación web lo ejecutamos y obtenemos el siguiente resultado como indica la **Figura II.17**.

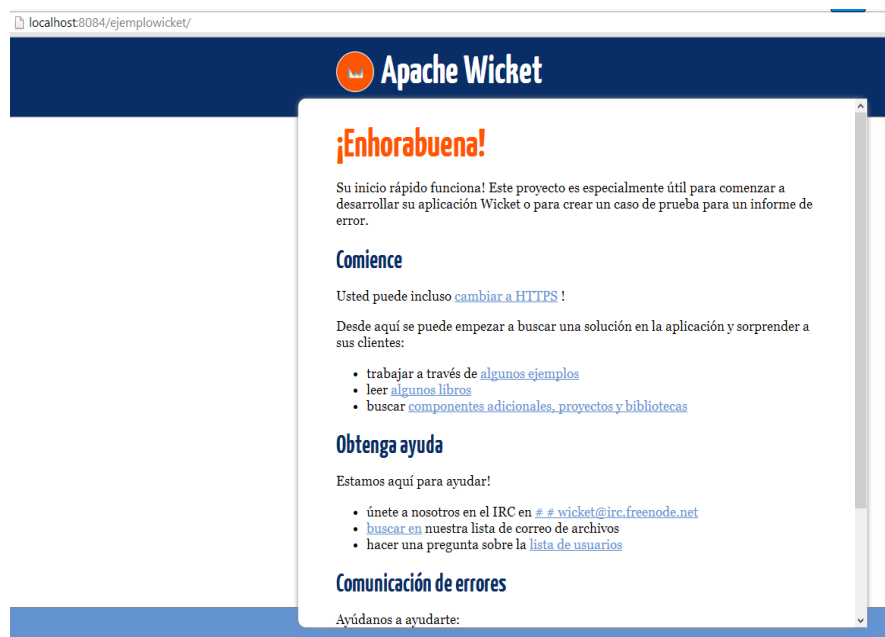


Figura II.17 Pantallas de inicio de Wicket

En la cual se observa que la aplicación con Wicket se ejecutó con éxito.

2.10. Desarrollo de aplicaciones web con Tapestry

Para crear un proyecto con Tapestry se debe tener instalado las herramientas necesarias:

- Tener instalado JDK 1.7.
- Utilizar la herramienta Netbeans IDE 7.2
- Instalar Maven 3.1.0.
- Configurar Apache Tomcat 7.0.27.
- Crear la aplicación basada en el arquetipo Maven mencionado anteriormente.
- Seleccionar el arquetipo a utilizar de Tapestry 5.3.7.

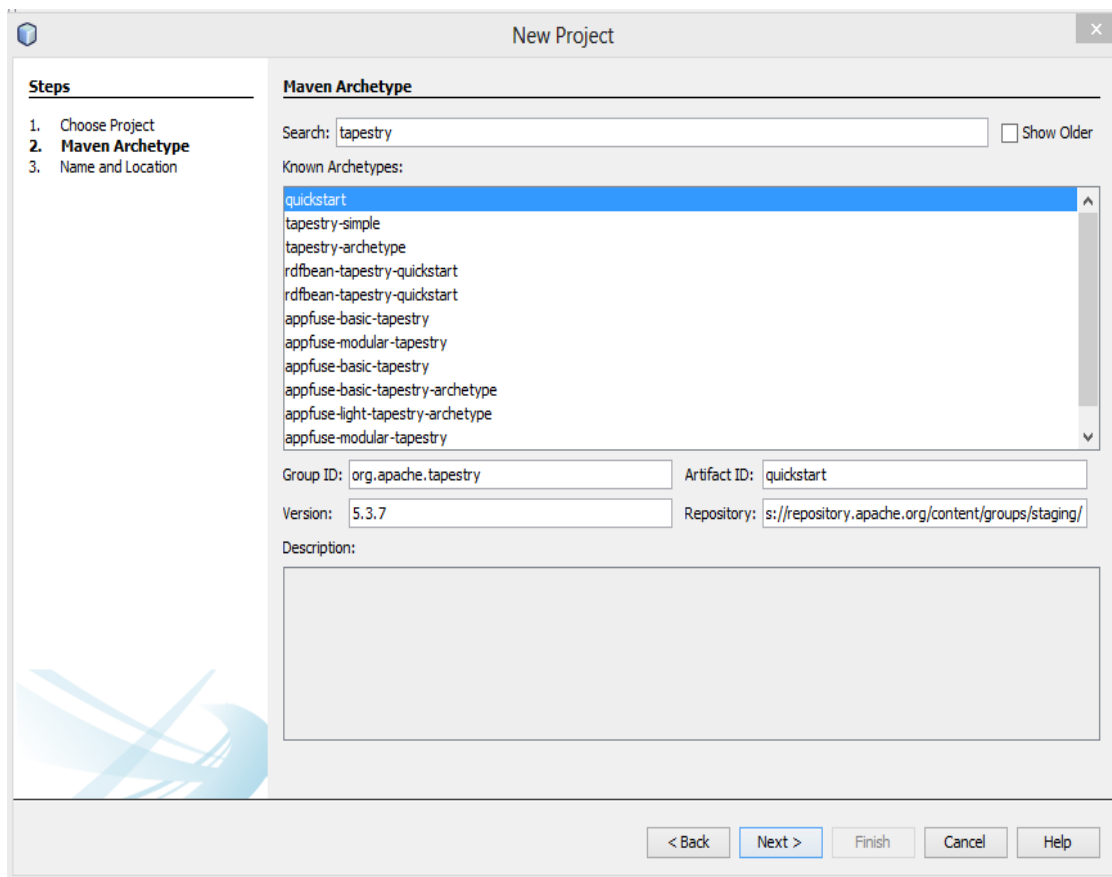


Figura II.18 Selección del Arquetipo de Tapestry

En la **Figura II.18** se observa que al escribir el framework Tapestry seleccionamos quickstart y automáticamente Maven crea el esqueleto de la aplicación web.

Maven utiliza el ID de grupo y de Identificación del artefacto para proporcionar una identidad única para la aplicación y tapestry tiene que tener un nombre de paquete base para que sepa dónde buscar páginas y componentes.

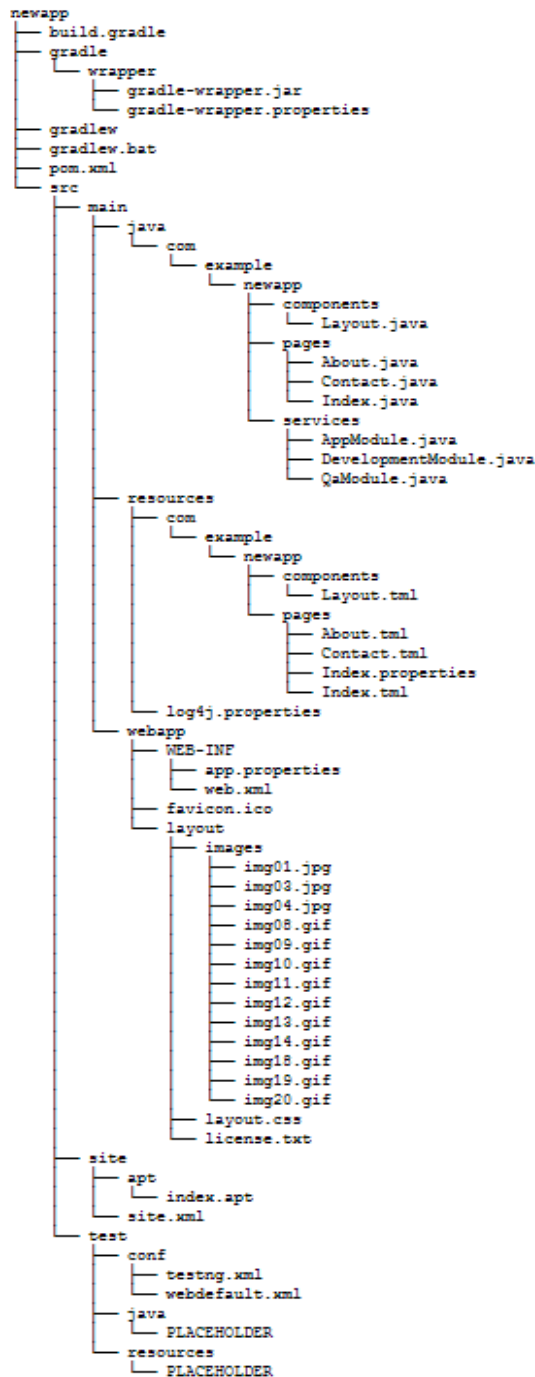


Figura II.19 Esqueleto de una Aplicación en Tapestry

Fuente: <http://tapestry.apache.org/getting-started.html>

Maven al crear el inicio rápido Tapestry 5.3.7 con Netbeans IDE 7.2, se crea con los siguientes archivos necesarios para empezar a trabajar con este framework, como se observa en la **Figura II.20**.

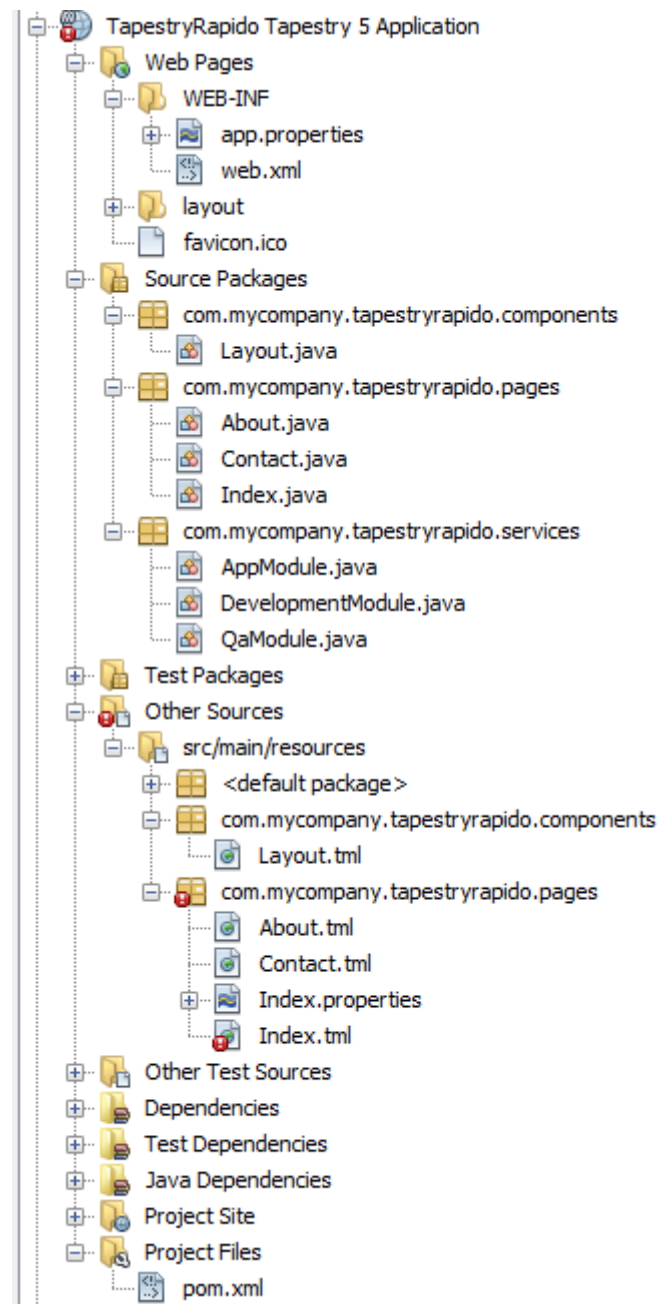


Figura II.20 Estructura de una Aplicación Tapestry en Netbeans

Una vez creada nuestra estructura de la aplicación web lo ejecutamos y obtenemos el siguiente resultado como indica la **Figura II. 21**.

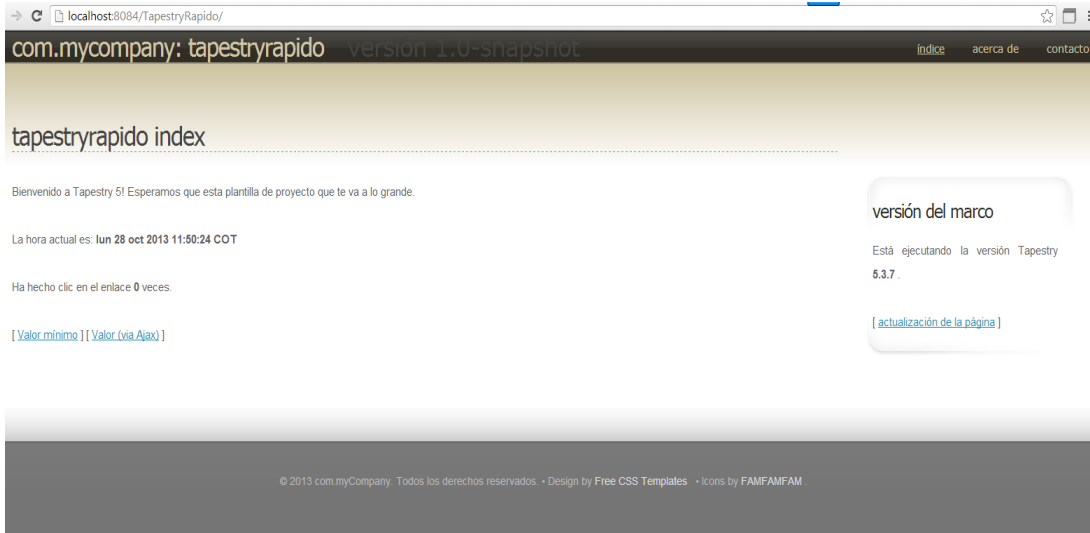


Figura II.21 Pantalla de Inicio de Tapestry

En la cual se observa que la aplicación con Tapestry se ejecutó con éxito.

Configuración de Netbeans IDE para que lea los archivos tml de tapestry

Asociar la extensión tml a "Editor XHTML": seleccione el menú "Herramientas \ Opciones", seleccione la pestaña "Files", inserte una nueva extensión de archivo "TML" y asociarlo a "Archivos de XHTML (text / xhtml)", pulse "OK". [12]

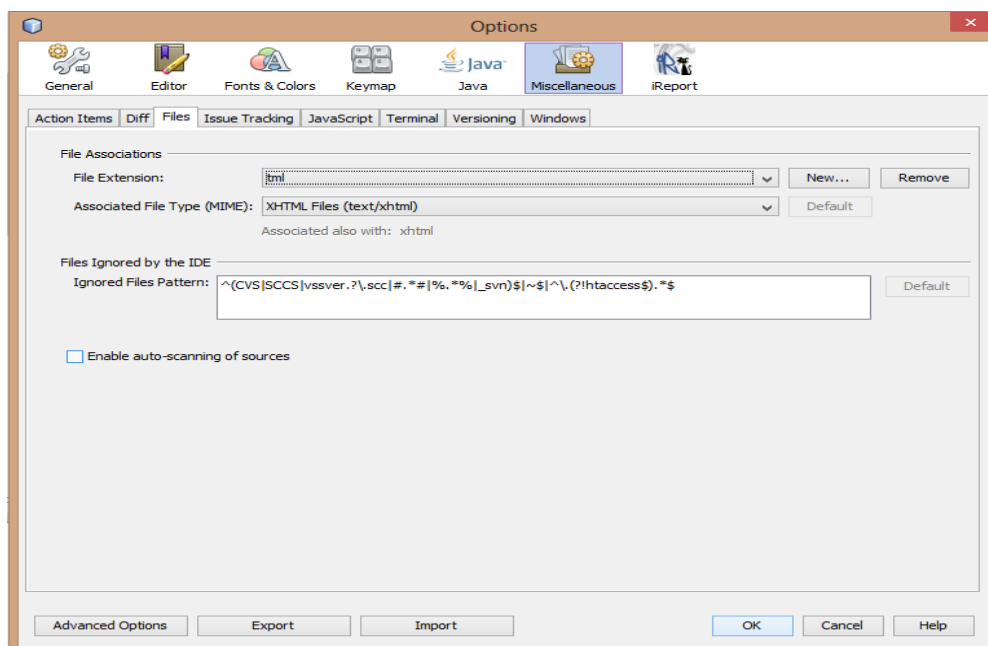


Figura II.22 Configuración de archivos TML en Tapestry

2.11. CSS

Para saber y entender un poco más sobre las hojas de estilo se pudo observar de la tesis que se encuentra en el Centro de Documentación [5].

Las hojas de estilo en cascada (Cascading Style Sheets, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirá de estándar para los agentes de usuario o navegadores.

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación. Los estilos CSS (Cascading Style Sheets) son hojas de estilo de actualización automática.

Se usan principalmente para definir estilos que luego se aplicarán a las páginas de nuestro sitio, incluso a veces permiten definir características que no permiten definir los estilos HTML, como el color de fondo para el texto por ejemplo.

Al estar la definición de los estilos en un archivo externo a las páginas y común a todas las páginas del sitio (es recomendable) el aspecto de nuestras páginas será más homogéneo y además se puede cambiar ese aspecto de manera segura e inmediata cambiando únicamente la hoja de estilos.

Se pueden definir estilos independientes o estilos asociados a determinadas etiquetas por ejemplo a la etiqueta <a> (que corresponde a los hiperenlaces). De este modo, todos los hiperenlaces de la página o del sitio adquirirían la apariencia definida en ese estilo y con un sólo cambio en la hoja de estilos se puede cambiar de golpe el estilo de todos los enlaces en todas las páginas vinculadas a este estilo.

El inconveniente que se tiene al trabajar con hojas de estilos es que algunos navegadores no las soportan y las ignoran, aunque estos navegadores suelen ser versiones antiguas, por lo que ocurrirá en pocos casos.

Las hojas de estilo pueden crearse con cualquier editor de texto, como puede ser el

Bloc de notas, y pueden guardarse con la extensión TXT.

2.12. Ventajas de CSS

Las ventajas de las hojas de estilo se observó de la tesis que se encuentra en el Centro de Documentación [5].

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.
- Los Navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web remoto, con lo que aumenta considerablemente la accesibilidad. Por ejemplo, personas con deficiencias visuales pueden configurar su propia hoja de estilo para aumentar el tamaño del texto o remarcar más los enlaces.
- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario.
- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño.
- Aligera la carga de nuestro sitio al juntar todas las imágenes en una.
- Permiten ahorrar tiempo y trabajo al poder seguir varias técnicas (bordes redondeados, sombra en el texto, sombra en las cajas, etc) sin necesidad de usar un editor gráfico.

CAPITULO III

3 Estudio comparativo de los frameworks Tapestry y Wicket para el desarrollo de aplicaciones web.

3.1. Introducción

La necesidad de establecer el mejor framework de desarrollo, además el framework que brinde una mejor productividad para el desarrollo de aplicaciones web, para esto se pretende realizar un análisis en base a los parámetros e indicadores que proporcione la comparación entre los dos frameworks.

En este capítulo se pretende realizar dos prototipos con los frameworks de estudio Wicket y Tapestry, y determinar los parámetros para su comparación. Además, establecer los indicadores para comprobar la productividad de los dos frameworks, y plasmar un estudio en base a los parámetros e indicadores establecidos y conseguir un resultado de la comparación.

3.2. Elaboración de prototipos con los frameworks

Para estudiar y medir los frameworks de estudio se implementó dos prototipos, el **Prototipo 1** fue realizado con el framework WICKET y el **Prototipo 2** con el framework TAPESTRY.

La función que realiza el prototipo es recoger las tareas más importantes de la aplicación, el mismo que cumple con dos tareas específicas, la **Tarea 1** gestionar los métodos

CRUD⁷ de las carreras y la **Tarea 2** permite al usuario inscribir al alumno y a su vez guardar la información en la base de datos del sistema web SFSG, en la cual contiene una cantidad considerable de datos que permiten obtener información relevante para el estudio comparativo de los frameworks, y así poder obtener los indicadores necesarios para analizar e interpretar la productividad de cada framework, como la curva de aprendizaje, líneas de código, diseño y tiempo de desarrollo los cuales se describen a continuación.

Al analizar los parámetros de comparación se escogió el **Prototipo 2**, porque brinda mejores prestaciones de productividad en el desarrollo de aplicaciones web, sin embargo es complejo su estructura lo cual dificulta el aprendizaje, pero su funcionalidad lo hace agradable para el usuario por la reutilización de los componentes que utiliza, y el diseño predefinido que algunos componentes poseen.

3.3. Determinación de los parámetros de comparación

Se determinan algunos parámetros que admitirán determinar la comparación entre los frameworks WICKET y TAPESTRY como indica en la **Tabla III.III**, los cuales han sido determinados según lo observado en tesis situadas en el Centro de Documentación de la ESPOCH. [5]

Tabla III.III Parámetros de comparación

Nº	PARÁMETRO	CONCEPTO
1	Líneas de código	Cantidad de líneas de código para la ejecución de una tarea específica.
2	Curva de aprendizaje	Capacidad para aprender por parte del desarrollador.
3	Reutilización de Componentes	Capacidad de reutilizar los componentes en cualquier página o aplicación web.
4	Documentación	Capacidad de encontrar una documentación clara que sirva para la investigación.
5	Diseño	Capacidad de generar un diseño para una tarea específica.
6	Tiempo de desarrollo	Tiempo requerido para realizar una tarea específica.

⁷ **CRUD**: Create, Read, Update and Delete (Crear, Leer, Actualizar y Eliminar)

3.4. Determinación de los indicadores de productividad

A continuación se ha escogido los siguientes indicadores que permiten medir la productividad de los frameworks WICKET y TAPESTRY como se observa en la **Tabla III.IV**, los cuales fueron determinados de los parámetros de comparación entre los frameworks de estudio.

Tabla III.IV Indicadores de productividad

N°	PARAMETROS	INDICADORES	CONCEPTO
1	Líneas de código	Líneas de código	Cantidad de líneas de código para la ejecución de una tarea específica.
2	Curva de aprendizaje	Curva de aprendizaje	Capacidad para aprender por parte del desarrollador.
3	Reutilización de Componentes	Reutilización de Componentes	Capacidad de reutilizar los componentes en cualquier página o aplicación web.
4	Documentación	Cantidad de Libros	Cantidad de libros para la investigación.
		Idioma de la Documentación	Los diferentes idiomas que esta la documentación.
		Videos Demostrativos	Videos donde indiquen la funcionalidad del framework.
		Foros	La comunidad de apoyo del framework.
		Páginas Oficiales	Las páginas oficiales del framework.
5	Diseño	Diseño	Capacidad de generar un diseño para una tarea específica.
6	Tiempo de desarrollo	Tiempo de desarrollo	Tiempo requerido para realizar una tarea específica.

3.5. Criterios de evaluación

En la **Tabla III.V** se expresan los valores cualitativos y cuantitativos que se proveerán a los parámetros a ser medidos en la comparación de los frameworks, para esto se ha determinado utilizar los valores comprendidos en el rango 0 – 4, los mismos que fueron

determinados según lo observado en tesis situadas en el Centro de Documentación de la ESPOCH. [5]

Tabla III.V Criterios de Evaluación General

Cuantitativa	0	1	2	3	4
Cualitativa	Bajo	Medio Bajo	Medio	Medio Alto	Alto
	Excesivo	Mucho	Lo necesario	Poco	Muy Poco
	No	-	-	-	Si
	Muy Difícil	Difícil	Medio	Fácil	Muy Fácil
	No disponible	Casi no disponible	Limitado	Casi si disponible	Si disponible

3.6. Análisis, interpretación y representación de resultados de los parámetros de comparación para los frameworks Wicket y Tapestry.

En este trabajo de investigación la comparación entre los frameworks de estudio será elaborado en base a los datos y conceptos teóricos de la indagación bibliográfica y de la investigación realizada por el autor de la tesis manipulando para esto los dos prototipos realizados.

3.6.1 Líneas de código

Los frameworks WICKET y TAPESTRY utilizan lenguajes básicos (HTML, CSS y XML) para el desarrollo de la aplicación, cabe mencionar que los archivos XML son utilizados para describir información de la aplicación y sus elementos, dependencias y componentes utilizados.

Para el desarrollo de las 2 tareas tanto en WICKET como en TAPESTRY se utilizó el repositorio **maven** para generar el esqueleto de la aplicación, donde se genera algunos archivos necesarios como **web.xml** y **pom.xml** y los paquetes que se generan automáticamente.

Para contabilizar la cantidad de líneas de código se utilizó el programa Universal Code Lines Counter, una herramienta fácil y rápida para el cálculo de código fuente, en blanco,

líneas de comentarios e incluso líneas mixtas (código fuente y comentarios en una sola línea) de los proyectos de software, y es muy fácil de configurar el programa para más lenguajes de programación / script, por defecto ya muchos idiomas se incluyen: C #, C / C + +, CSS, COBOL, Delphi / Pascal, Fortran, HTML, Java, JavaScript, MySQL, Perl, PHP, Python, Ruby, VBScript / ASP, Visual Basic, XML, etc., (ver Anexo 1).

En WICKET en el archivo **web.xml** va el filtro Wicket definido, que maneja todas las solicitudes, sólo se necesita el parámetro de filtro **applicationClassName**. También, se observa la **url-mapping a /***. El filtro Wicket sólo procesará las solicitudes que son peticiones Wicket, esto garantiza que recursos estáticos fuera del ámbito de la aplicación Wicket, tales como hojas de estilo, archivos JavaScript, imágenes y otros serán atendidos por el contenedor, en el archivo **pom.xml** van las dependencias que se utilizan, en el cual se implementó las dependencias necesarias para el desarrollo de las tareas.

El prototipo 1 desarrollado en el framework WICKET, requirió de algunos formularios como: **Insertar_Carrera.html**, **Lista_Carrera.html**, **Modificar_Carrera.html** de 48, 49 y 49 líneas respectivamente, así como sus respectivas clases las cuales tienen el mismo nombre **Insertar_Carrera.java**, **Lista_Carrera.java**, **Modificar Carrera. java** de 75, 95 y 58 líneas respectivamente, también necesita de una clase para poder implementar los procedimientos **Carrera.java** de 138 líneas, para cumplir con la tarea 1. Para cumplir con la tarea 2 requirió de un formulario **Insertar_Alumno_Pago.html** de 102 líneas con su clase **Insertar_Alumno_Pago.java** de 188 líneas, con la ayuda de una clase **Alumno_Pago.java** de 324 líneas.

En TAPESTRY aparece como un filtro de servlet, esto le da una gran flexibilidad a las URLs sin necesidad de un montón de configuración XML, el archivo **web.xml** lo utiliza para localizar las páginas y clases de los componentes, aunque la mayoría de la configuración se realiza con Java, un pequeño pero necesario trabajo de configuración se produce en el interior del descriptor de implementación servlet, WEB-INF/web.xml.

Por convención, el nombre del filtro (filter-name) es casi siempre "app", Tapestry utiliza esto para determinar qué clase de módulo debe buscar. También utiliza una clase de módulo de la aplicación en la cual define nuevos servicios, proporciona reemplazos de los servicios, o contribuye a las configuraciones de servicio.

En el archivo **pom.xml** van las dependencias que se utilizan, en el cual se implementó las dependencias necesarias para las dos tareas desarrolladas.

El prototipo 2 desarrollado en TAPESTRY, requirió de algunos formularios como: **InsertarCarrera.tml**, **ListaCarrera.tml**, **ActualizarCarrera.tml** de 35, 22 y 42 líneas respectivamente, así como sus respectivas clases las cuales tienen el mismo nombre **InsertarCarrera.java**, **ListaCarrera.java**, **ActualizarCarrera.java** de 35, 40 y 43 líneas respectivamente, pero necesita de algunos servicios como **CarreraDAO.java** y **CarreraDAOImp.java** de 20 y 104 líneas respectivamente, para cumplir con la tarea 1. Para cumplir con la tarea 2 requirió de un formulario **InsertarAlumno_Pago.tml** de 75 líneas con su clase **InsertarAlumno_Pago.java** de 190 líneas, pero necesita algunos servicios como **AlumnoPagoDAO.java** y **AlumnoPagoDAOImp.java** de 22 y 159 líneas respectivamente.

Tabla III.VI Criterios de evaluación para el parámetro Líneas de código

Cuantitativa	0	1	2	3	4
Cualitativa	≥ 800	600 - 799	400 - 599	200 - 399	1 - 199

Tabla III.VII Parámetro Líneas de Código

FRAMEWORKS	WICKET		TAPESTRY	
	Tarea 1	Tarea 2	Tarea 1	Tarea 2
Nº DE LÍNEAS DE CÓDIGO	512	614	341	446
CRITERIO DE EVALUACIÓN	2	1	3	2
PROMEDIO	1.5		2.5	

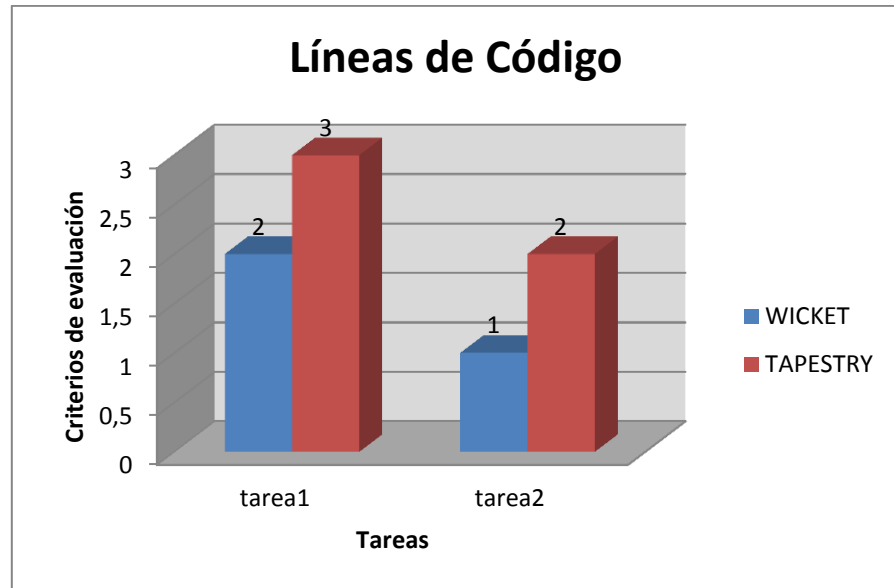


Figura III.23 Parámetro de Líneas de código

Interpretación:

En la **Figura III.23**, se observa los resultados obtenidos para el parámetro de Líneas de código, en el cual el framework TAPESTRY con los valores (3 y 2) toma ventaja sobre el framework WICKET con los valores (2 y 1) para las tareas 1 y 2 respectivamente, al utilizar menos cantidad de líneas de código en el cumplimiento de las dos tareas.

3.6.2 Curva de Aprendizaje

En cuanto a la administración de requerimientos, el framework TAPESTRY es prácticamente el trabajo de Howard Lewis Ship y es quien decide qué incluir y qué no dentro del framework. Por otro lado, el framework WICKET es producto de una comunidad de desarrolladores, quienes debaten y ajustan este tipo de funcionalidad en foros de discusión.

El framework WICKET ha sido pensado con la facilidad de implementar una aplicación web, aunque al principio es medio complicado comprender pero con la práctica se vuelve más fácil de entender su funcionamiento, ya que esto es desarrollado por la comunidad Wicket, sin embargo su comunidad no es muy amplia ya que los desarrolladores optan por la utilizan de otros frameworks más rápidos y fáciles de aprender.

El framework TAPESTRY es muy complejo y a la vez difícil de aprender, por esta razón la inquietud de los programadores es al momento de desarrollar aplicaciones web con este framework, los mismos que buscan facilidad en la programación y aumentar la productividad, esto puede ser un obstáculo para desarrolladores novatos, porque se ven frustrados cuando no se acomodan a las convenciones de nombrado, al observar que algunas cosas funcionan sin saber que se hizo y otras no porque está mal escrito o por un error en el nombre, así como la utilización de clases con su respectiva implementación para acceder y manipular la base de datos. Se pudo determinar este parámetro gracias a la ayuda de Ernesto Arteaga Zavala Ingeniero de Desarrollo (ver Anexo 12).

Tabla III.VIII Criterios de evaluación para el parámetro Curva de Aprendizaje

Cuantitativa	0	1	2	3	4
Cualitativa	Muy Difícil	Difícil	Medio	Fácil	Muy Fácil

Tabla III.IX Parámetro Curva de Aprendizaje

FRAMEWORKS	WICKET		TAPESTRY	
	Tarea 1	Tarea 2	Tarea 1	Tarea 2
CRITERIO DE EVALUACIÓN	2	2	1	1
PROMEDIO	2		1	

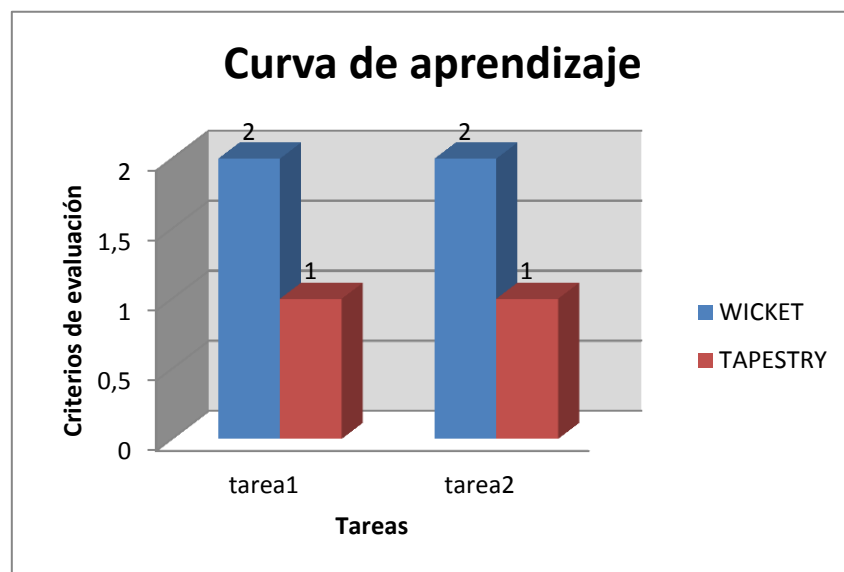


Figura III.24 Parámetro de Comparación Curva de Aprendizaje

Interpretación:

En la **Figura III.24** se observa los resultados obtenidos para el parámetro Curva de Aprendizaje en la que se puede determinar que la programación con el framework WICKET requiere de un conocimiento no muy complejo al obtener un valor de 2 en su criterio de evaluación mientras que con el framework TAPESTRY se debe tener un conocimiento más avanzado y conocer más sobre los servicios que se debe implementar ya que tiene una valoración de 1. Teniendo como resultado que en el parámetro Curva de Aprendizaje tiene una gran ventaja el framework WICKET sobre el framework TAPESTRY ya que se requiere de más conocimientos.

3.6.3 Reutilización de Componentes

Con el framework WICKET los componentes escritos deben ser reutilizables en el desarrollo de las aplicaciones web, una vez diseñados los componentes necesarios construir una aplicación no es más que unirlos. Wicket es un framework de presentación basado en componentes.

Con el framework TAPESTRY facilita la reusabilidad y el mantenimiento en las aplicaciones web, porque una vez diseñados los componentes necesarios se puede construir una aplicación con solo unirlos. Tapestry es un framework de presentación realmente basado en componentes ya que es de los frameworks que mejor representa el concepto de componente, llevándolo hasta el límite de que en él todo son componentes.

Tabla III.X Criterios de evaluación para el parámetro Reutilización de Componentes

Cuantitativa	0	1	2	3	4
Cualitativa	Bajo	Medio Bajo	Medio	Medio Alto	Alto

Tabla III.XI Parámetro Reutilización de Componentes

FRAMEWORKS	WICKET		TAPESTRY	
	Tarea 1	Tarea 2	Tarea 1	Tarea 2
CRITERIO DE EVALUACIÓN	4	4	4	4
PROMEDIO	4		4	

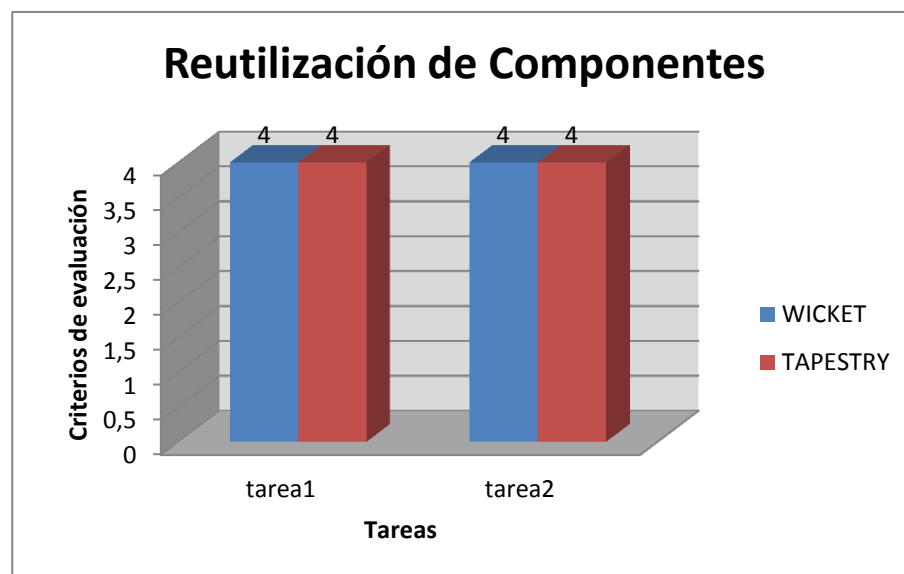


Figura III.25 Parámetro de comparación Reutilización de Componentes

Interpretación:

En la **Figura III.25** se puede determinar que en el parámetro de Reutilización de Componentes, los frameworks WICKET y TAPESTRY tienen el mismo valor de 4 ya que son frameworks de presentación basado en componentes.

3.6.4 Documentación

La documentación tanto de los frameworks Wicket y Tapestry dificulta el proceso de aprendizaje, ya que no hay una documentación clara y detallada, para un mejor análisis del parámetro Documentación se va a describir los indicadores antes mencionados.

3.6.4.1 Cantidad de Libros

En el framework WICKET lo que dificulta enormemente es la cantidad de libros ya que la mayoría son pagados, pero así también se pueden conseguir libros gratuitos, los cuales son muy escasos, como los libros Wicket in Action [1] y Wicket User Guide [26], estos dos únicos libros que han ayudado al análisis y programación del framework Wicket.

Mientras que en el framework TAPESTRY la cantidad de libros se los puede conseguir pagados y gratuitos, pero hay pocos libros muy buenos los cuales sirvieron mucho en la investigación, especialmente en la programación como los libros Tapestry 5: A Step-By-Step Guide to Java Web Development with the Developer-Friendly Apache Tapestry Framework (una guía paso a paso para el Desarrollo Web en Java con el framework Apache Tapestry amistoso con el desarrollador) [2], Tapestry 101 [3], el enfoque de este libro es que le guiará a través del diseño e implementación de su aplicación y el suministro de información que se puede mejorar en él con sus propios componentes personalizados y el libro PlugIn Tapestry [21], desarrollo de aplicaciones y páginas web con Apache Tapestry, este ayudo mucho en la documentación y programación sobre el framework Tpestry.

Tabla III.XII Criterios de evaluación para el indicador Cantidad de Libros

Cuantitativa	4	3	2	1	0
Cualitativa	≥9	7 - 8	5 - 6	3 - 4	1 - 2

Tabla III.XIII Indicador Cantidad de Libros

FRAMEWORKS		
INDICADOR	WICKET	TAPESTRY
CANTIDAD DE LIBROS	2	3
CRITERIO DE EVALUACIÓN	0	1

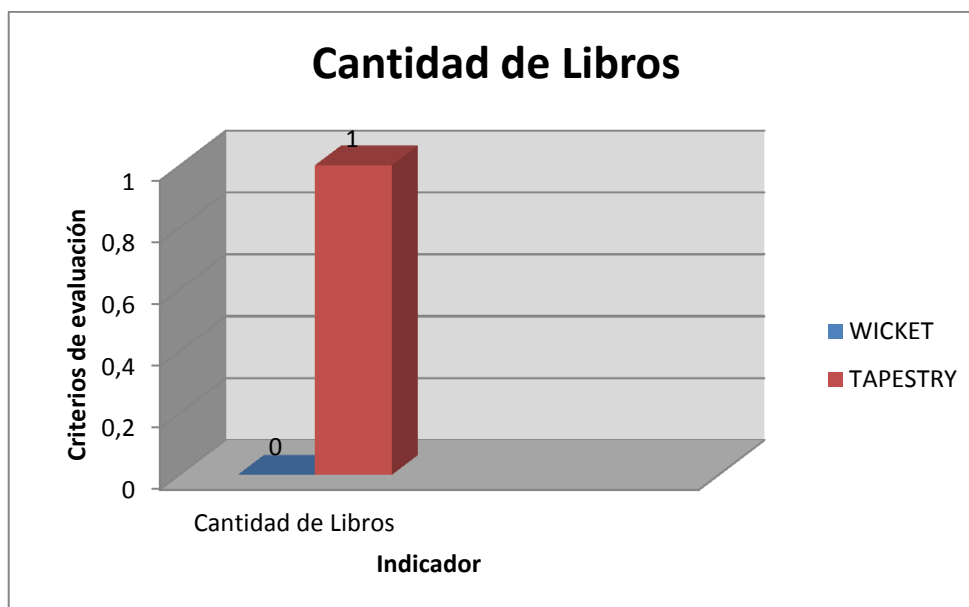


Figura III.26 Indicador Cantidad de Libros

Interpretación:

En la **Figura III.26** se observa los resultados obtenidos para el indicador Cantidad de Libros, en la que se puede determinar que existe mayor cantidad de libros sobre el framework Tapestry al obtener un valor de 1, mientras que Wicket obtuvo 0 en el criterio de evaluación. Teniendo como resultado que el indicador Cantidad de Libros tiene una gran ventaja el framework TAPESTRY sobre el framework WICKET por el número de libros consultados.

3.6.4.2 Idioma de la Documentación

En el framework WICKET la documentación está en inglés casi nada en español, así como los libros, páginas oficiales lo cual dificultó un poco el aprendizaje y el desarrollo de este framework, ya que tomó más tiempo en el análisis de este framework por el idioma en que se encuentra su documentación.

La documentación del framework TAPESTRY se puede encontrar en diferentes idiomas, así como buenos tutoriales en inglés y en español, videos demostrativos en chino. El idioma de la documentación en el sitio oficial se encuentra en inglés, encontrar en español tutoriales, guías o ayuda para algunos conceptos de Tapestry puede ser un

trabajo difícil, pero el poco contenido sigue siendo de gran utilidad y ha sido una gran fuente de información para la realización de esta investigación.

Tabla III.XIV Criterios de evaluación para el indicador Idioma de la Documentación

Cuantitativa	4	3	2	1	0
Cualitativa	≥9	7 - 8	5 - 6	3 - 4	1 - 2

Tabla III.XV Indicador Idioma de la Documentación

FRAMEWORKS		
INDICADOR	WICKET	TAPESTRY
IDIOMA DE LA DOCUMENTACIÓN	1	3
CRITERIO DE EVALUACIÓN	0	1

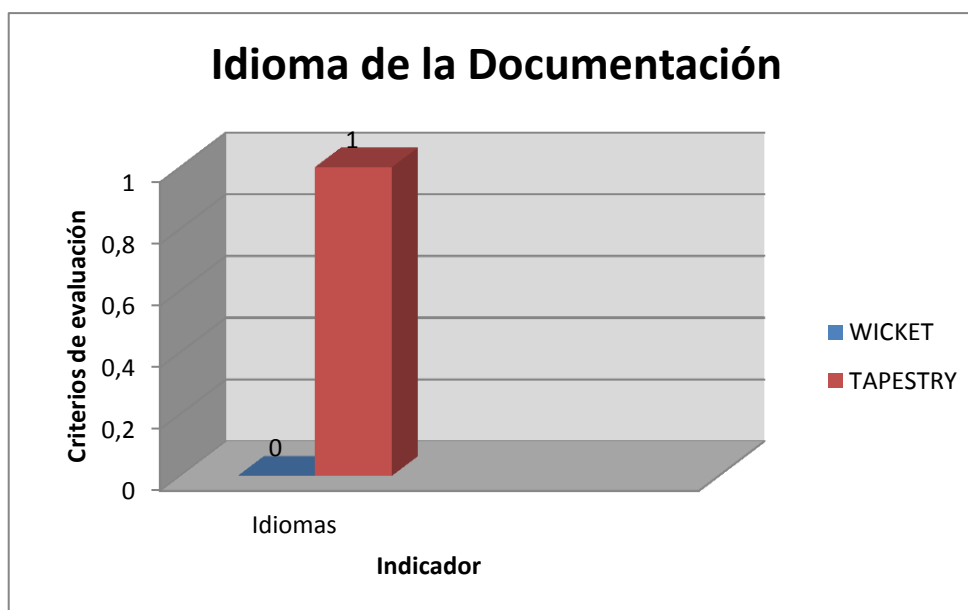


Figura III.27 Indicador Idioma de la Documentación

Interpretación:

En la **Figura III.27** se observa los resultados obtenidos para el indicador Idioma de la Documentación, en la cual se puede determinar que en Tapestry la documentación se

puede encontrar en diferentes idiomas al obtener un valor de 1, mientras que Wicket obtuvo 0 en el criterio de evaluación. Teniendo como resultado que el indicador Idioma de la Documentación tiene ventaja el framework TAPESTRY sobre el framework WICKET por el número de idiomas en que se puede encontrar la información.

3.6.4.3 Videos Demostrativos

En WICKET no hay videos demostrativos de la funcionalidad sobre los componentes o sobre el desarrollo del framework, solo existen videos sobre conferencias y conceptos de Wicket, lo cual no ayudo mucho ya que se encuentra en inglés.

Mientras que en el framework TAPESTRY si existen videos demostrativos [25] sobre el desarrollo y funcionamiento de los componentes de este framework, lo cual ayudó en el aprendizaje y entendimiento de algunos componentes, pero su desventaja que hay en el idioma chino.

Tabla III.XVI Criterios de evaluación para el indicador Videos Demostrativos

Cuantitativa	0	1	2	3	4
Cualitativa	No disponible	Casi no es posible	Limitado	Casi si es posible	Si disponible

Tabla III.XVII Indicador Videos Demostrativos

FRAMEWORKS		
INDICADOR	WICKET	TAPESTRY
CRITERIO DE EVALUACIÓN	0	4

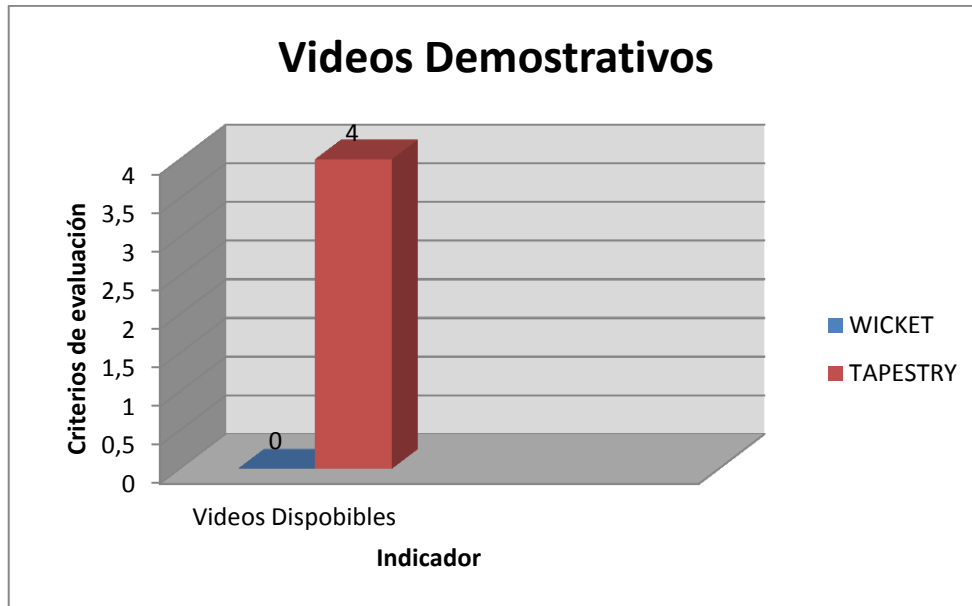


Figura III.28 Indicador Videos Demostrativos

Interpretación:

En la **Figura III.28** se observa los resultados obtenidos para el parámetro Videos Demostrativos, donde el framework TAPESTRY obtuvo 4 en el criterio de evaluación mientras que el framework WICKET obtuvo 0, ya que no existen videos demostrativos para el framework WICKET.

3.6.4.4 Foros

Para el framework WICKET los pocos foros que se encontró los hay en inglés, los cuales no ayudaron mucho, las comunidades de apoyo como javamexico para este framework es escasa, ya que esta comunidad de desarrolladores no están muy familiarizados con este framework.

En el framework TAPESTRY hay algunos foros, como javamexico el cual ayudó a la realización y análisis de este framework, donde se pudo aprender sobre la funcionalidad de algunos componentes, así como el blog pico.dev en el cual existe una gran cantidad de información sobre este framework, se contactó con una persona particular de México (ICE Ernesto Arteaga Zavala Ingeniero de Desarrollo) que ha desarrollado sistemas con este framework.

Tabla III.XVIII Criterios de evaluación para el indicador Foros

Cuantitativa	0	1	2	3	4
Cualitativa	No disponible	Casi no es posible	Limitado	Casi si es posible	Si disponible

Tabla III.XIX Indicador Foros

FRAMEWORKS		
INDICADOR	WICKET	TAPESTRY
CRITERIO DE EVALUACIÓN	2	4

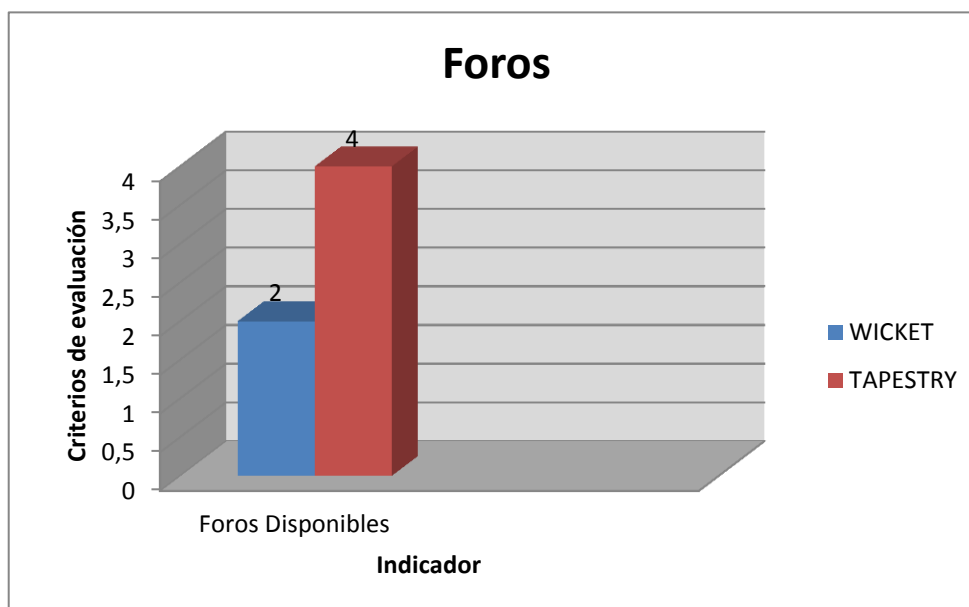


Figura III.29 Indicador Foros

Interpretación:

En la **Figura III.29**, se observa los resultados obtenidos para el Indicador Foros, en el cual el framework TAPESTRY toma ventaja sobre el framework WICKET con los valores (4 y 2) respectivamente, ya que la comunidad de desarrolladores se ha inclinado más por el desarrollo de Tapestry para realizar aplicaciones web.

3.6.4.5 Páginas Oficiales

El framework WICKET tiene su propia página oficial [7] la cual es contribuida por la comunidad que sigue este framework, la cual al realizar cambios en el framework lo hacen pensando en la comunidad Wicket, aunque esta comunidad sea escasa ya que los desarrolladores se han inclinados por optar por otros frameworks.

Así también el framework TAPESTRY tiene su propia página oficial [6] la cual es escrita y contribuida por su autor Howard M. Lewis, quien tiene el control para realizar cambios en este framework y el decide que cambiar o que quitar en Tapestry. La documentación que se encuentra disponible en el sitio oficial es muy útil pero insuficiente, aunque su contenido sigue siendo de gran utilidad y ha sido una gran fuente de información para la realización de este proyecto.

Tabla III.XX Criterios de evaluación para el indicador Páginas Oficiales

Cuantitativa	0	1	2	3	4
Cualitativa	No	-	-	-	Si

Tabla III.XXI Indicador Páginas Oficiales

FRAMEWORKS		
INDICADOR	WICKET	TAPESTRY
CRITERIO DE EVALUACIÓN	4	4

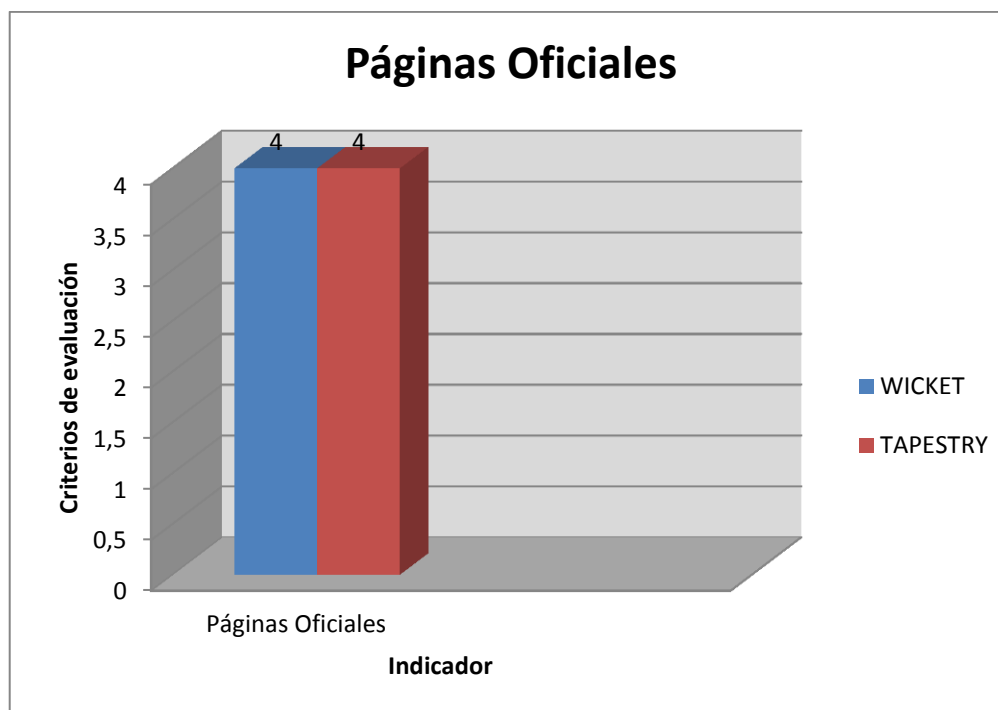


Figura III.30 Indicador Páginas Oficiales

Interpretación:

En la **Figura III.30** se observa los resultados obtenidos para el indicador Páginas Oficiales, en el cual se puede observar que tanto el framework TAPESTRY como el framework WICKET obtuvieron 4 en el criterio de evaluación, ya que los dos framework cuentan con su propia página oficial.

3.6.4.6 Resumen del parámetro Documentación

En el parámetro Documentación se determinó los indicadores antes mencionados los cuales sirvieron para su correcto análisis y comparación de los frameworks Tapestry y Wicket sobre este parámetro, los valores se los puede observar en la **Tabla III. XXII**.

Tabla III.XXII Resumen del parámetro Documentación

INDICADORES	WICKET	TAPESTRY	PESOS MÁXIMOS
Cantidad de Libros	0	1	4
Idioma de la Documentación	0	1	4
Videos Demostrativos	0	4	4
Foros	2	4	4
Páginas Oficiales	4	4	4
Total	6	14	20
Porcentuales	30%	70 %	100 %
Criterio de Evaluación	1.2	2.8	4

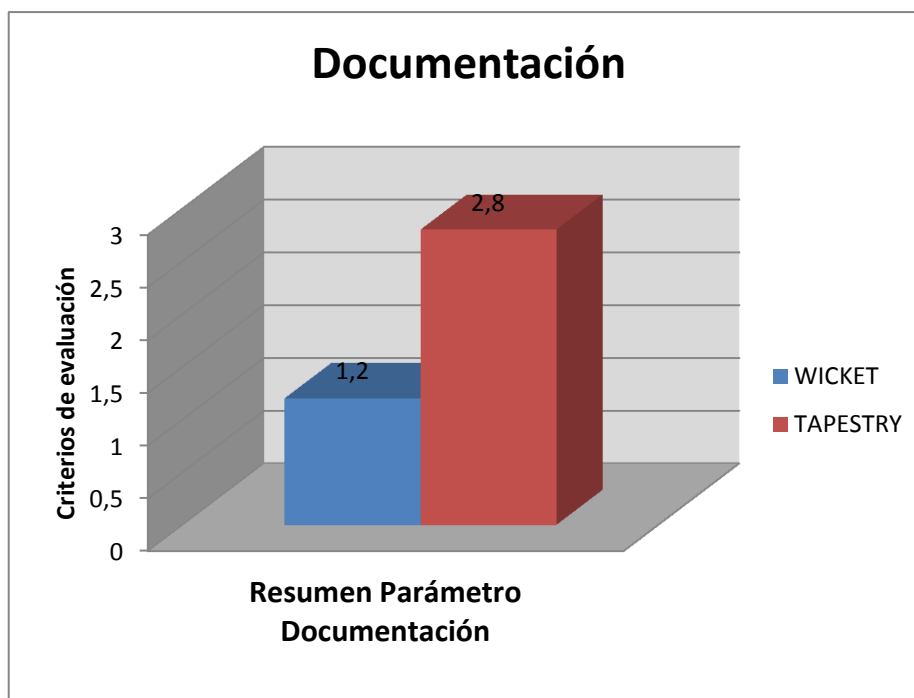


Figura III.31 Parámetro de comparación Documentación

Interpretación:

En la **Figura III.31** se puede observar en el resumen del parámetro Documentación que

TAPESTRY ha obtenido un porcentaje de 70% con un valor de 2.8 en el criterio de evaluación, mientras que WICKET obtuvo un 30% con un valor de 1.2, determinando que existe más información sobre el framework Tapestry ya que se puede encontrar en diferentes idiomas, y existe más cantidad de libros y videos demostrativos.

3.6.5 Diseño

En su mayor parte, las plantillas de componentes son estándar HTML / XHTML y combinando las plantillas con pequeños trozos de código Java usando un descriptor de archivos XML bien formado, esto significa que cada etiqueta de apertura debe tener una etiqueta de cierre coincidente, cada atributo debe ser citado, y así sucesivamente, las extensiones para el marcado ordinario se proporcionan en forma de un espacio de nombres para ambos framework.

Para las aplicaciones en WICKET el diseño es un poco más complicado ya que se debe saber dónde poner `<wicket:extend>` para que facilite el mismo diseño que se pretende obtener, las páginas comparten el mismo nombre que el archivo de clase correspondiente, pero con un **.html** final y se almacenan en el mismo paquete que la clase del componente correspondiente.

Bajo la estructura típica del directorio Maven, los archivos de clases Java y las páginas del prototipo 1 se encuentran en:

Clase Java: `src/main/java /com/mycompany/prototipowicketitsga/BasePage.java`

Plantilla: `src/main/ java /com/mycompany/prototipowicketitsga/BasePage.tml`

Ya que los archivos html tanto como las clases Java deben estar en el mismo paquete.

Para las aplicaciones en TAPESTRY el diseño lo hace más simple ya que existe algunos componentes como la **grid [9]** que da un aspecto más personalizado y agradable para el usuario, así como, el componente **datafield [10]** que muestra un calendario el cual es de gran ayuda para facilitar al usuario el ingreso de fechas, las páginas conlleva el mismo

nombre que el archivo de clase correspondiente, pero con un **.tml**⁸ final y se almacenan en el paquete con el mismo nombre que la clase del componente correspondiente.

Bajo la estructura del directorio Maven, los archivos de clases Java y html de los componentes del prototipo 2 se encuentran en:

Clase Java: `src/main/java /com/mycompany/prototipotapestryitsga/components/Plantilla.java`

Plantilla: `src/main/ resources /com/mycompany/prototipotapestryitsga/components/Plantilla.tml`

Del mismo modo, los archivos de clases Java y las páginas del prototipo 2 se encuentran en:

Clase Java: `src/main/java /com/mycompany/prototipotapestryitsga/pages/Index.java`

Plantilla: `src/main/ resources /com/mycompany/prototipotapestryitsga/pages/Index.tml`

Para las páginas (pero no otros componentes), un segundo lugar se buscará en el contexto de la aplicación web. La ubicación se basa en el nombre lógico de la página, en el Prototipo 2, la página **Index.tml** iría en la carpeta raíz de la aplicación web (Web Pages).

En Tapestry además es más fácil llamar a las plantillas para la reutilización en las páginas de la aplicación solo con **t:type="Plantilla"** dentro de la etiqueta `<html >`.

Tabla III.XXIII Criterios de evaluación para el parámetro Diseño

Cuantitativa	0	1	2	3	4
Cualitativa	Muy Difícil	Difícil	Medio	Fácil	Muy Fácil

⁸ **TML:** Tapestry Markup Lenguaje (Lenguaje de Mercado Tapestry)

Tabla III.XXIV Parámetro Diseño

FRAMEWORKS	WICKET		TAPESTRY	
	Tarea 1	Tarea 2	Tarea 1	Tarea 2
CRITERIO DE EVALUACIÓN	2	2	4	4
PROMEDIO	2		4	

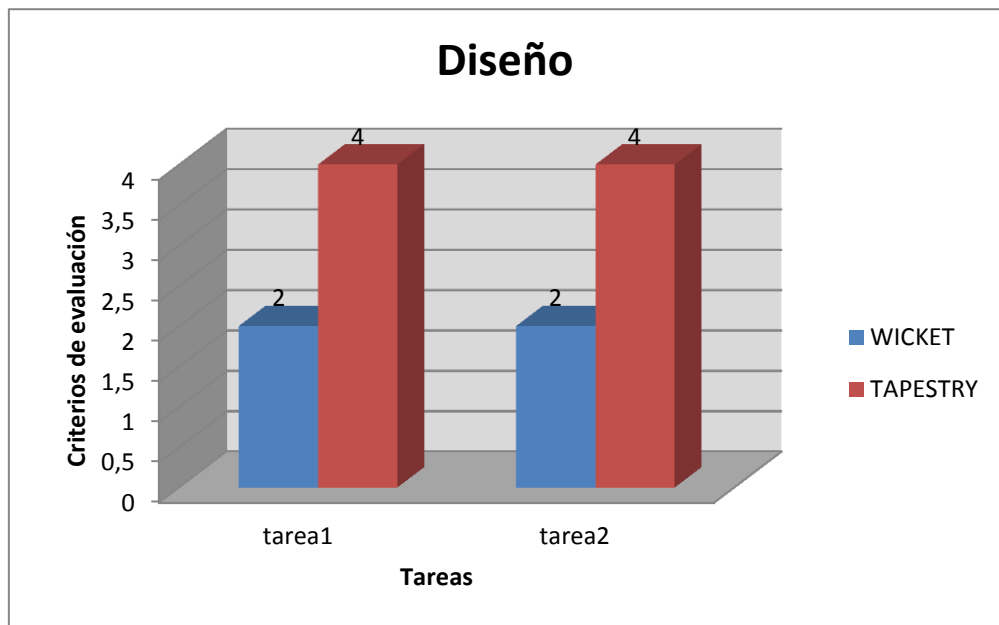


Figura III.32 Parámetro de comparación Diseño

Interpretación:

En la **Figura III.32** teniendo en cuenta el parámetro Diseño para el desarrollo de las aplicaciones web se puede observar que TAPESTRY toma ventaja sobre WICKET obteniendo valores de 4 y 2 respectivamente.

3.6.6 Tiempo de Desarrollo

Con WICKET, el tiempo empleado para el desarrollo de la **Tarea 1** fue de 10 días, y para la **Tarea 2** se utilizó 14 días laborables, los cuales fueron de mucho esfuerzo hasta aprender este framework, pero lo cual no tomó mucho tiempo en comparación de Tapestry.

Con TAPESTRY, se empleó 16 días para el cumplimiento de la **Tarea 1** y 20 días laborables para el desarrollo de la **Tarea 2**, lo cual fue un gran trabajo desarrollar con este framework, por lo tanto tomó mucho esfuerzo y dedicación aprender lo necesario sobre este framework ya que es un framework complejo.

Tabla III.XXV Criterios de evaluación para el parámetro Tiempo de desarrollo

Cuantitativa	0	1	2	3	4
Cualitativa	≥ 29 días	22-28 días	15-21 días	8-14 días	1-7 días

Tabla III.XXVI Parámetro Tiempo de desarrollo

FRAMEWORKS	WICKET		TAPESTRY	
	Tarea 1	Tarea 2	Tarea 1	Tarea 2
TIEMPO EMPLEADO	10	14	16	20
CRITERIO DE EVALUACIÓN	3	3	2	2
PROMEDIO	3		2	

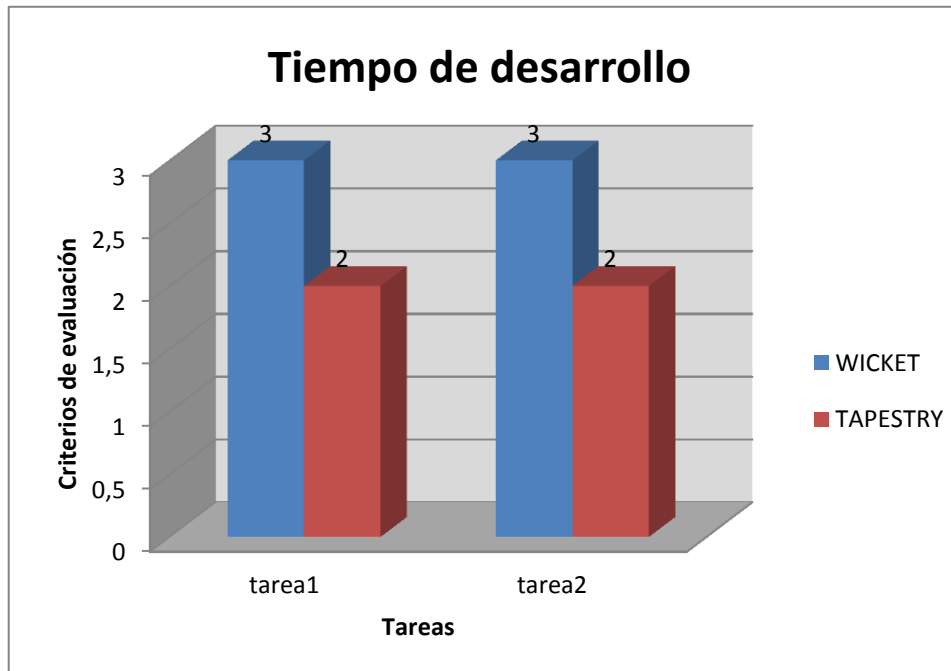


Figura III.33 Parámetro de comparación Tiempo de Desarrollo

Interpretación:

En la **Figura III.33** se observa la comparación de los framework de estudio con respecto al parámetro Tiempo de Desarrollo, se puede determinar que WICKET toma ventaja al obtener un promedio de 3 sobre el 2 que obtiene TAPESTRY.

3.6.7 Resumen de la comparación de los parámetros

Para un análisis claro y conciso en la comparación de los frameworks Tapestry y Wicket, se determinó los parámetros antes mencionados, los resultados se los puede observar en la **Tabla III.XXVII.**, que determina el framework que ofrece una excelente productividad en el desarrollo del sistema financiero en el INSTITUTO PARTICULAR SAN GABRIEL.

Tabla III.XXVII Resumen de comparación entre los framework Wicket y Tapestry

PARÁMETRO	WICKET	TAPESTRY	PESOS MÁXIMOS
Líneas de código	1.5	2.5	4
Curva de Aprendizaje	2	1	4
Reutilización de Componentes	4	4	4
Documentación	1.2	2.8	4
Diseño	2	4	4
Tiempo de desarrollo	3	2	4
Total	13.7	16.3	24
Porcentuales	57.09%	67.92 %	100 %

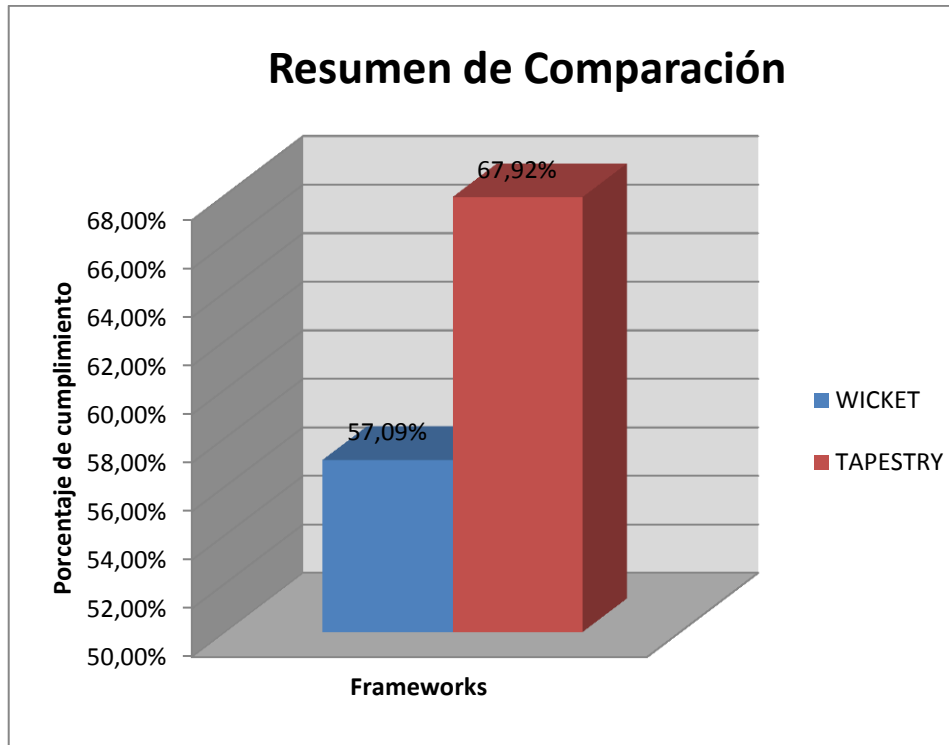


Figura III.34 Resumen de comparación entre los framework Wicket y Tapestry

Interpretación:

En la **Figura III.34** se puede establecer que el framework TAPESTRY destaca con el resultado de 67.92% sobre el framework WICKET que alcanzó el 57.09%, esto expresa que la aplicación web será desarrollada con el framework TAPESTRY ya que este framework brinda una mejor productividad en el desarrollo de aplicaciones web.

3.7. Comprobación de la Hipótesis

3.7.1 Hipótesis

El Framework Tapestry brinda las mejores prestaciones de productividad en el desarrollo del sistema financiero en el INSTITUTO PARTICULAR SAN GABRIEL.

3.7.2 Tipo de Hipótesis

La Hipótesis de esta indagación es de tipo Causa – Efecto.

3.7.3 Determinación de las variables

- **Variable Independiente:** El framework TAPESTRY.
- **Variable Dependiente:** La productividad en el desarrollo del sistema financiero en el INSTITUTO PARTICULAR SAN GABRIEL.

3.7.4 Operacionalización Conceptual

Tabla III.XXVIII Operacionalización Conceptual

VARIABLE	TIPO	CONCEPTO
El framework TAPESTRY	Variable Independiente	Framework desarrollado por Howard Lewis Ship que permite desarrollar aplicaciones web, y se basa en el patrón MVC.
La productividad en el desarrollo del sistema financiero en el INSTITUTO PARTICULAR SAN GABRIEL	Variable Dependiente	Es el resultado entre los valores obtenidos y los recursos empleados, o a su vez el mejor manejo de los recursos para llegar a la realización óptima de una meta, ya sea en términos de producto = ganancia o esfuerzo = recompensa, en el despliegue del desarrollo del sistema financiero en el INSTITUTO PARTICULAR SAN GABRIEL.

3.7.5 Operacionalización Metodológica

Tabla III.XXIX Operacionalización Metodológica

VARIABLE	CATEGORÍA	INDICADOR	TÉCNICAS	FUENTES DE VERIFICACIÓN
El framework TAPESTRY	Investigación	Framework de desarrollo	Revisión de documentos	<ul style="list-style-type: none"> ➤ Internet ➤ Videos ➤ Libros

La productividad en el desarrollo del sistema financiero en el INSTITUTO PARTICULAR SAN GABRIEL	Líneas de código	Líneas de código	Observación	<ul style="list-style-type: none"> ➤ Prototipo 1 ➤ Prototipo 2
	Curva de aprendizaje	Curva de aprendizaje	Observación	<ul style="list-style-type: none"> ➤ Prototipo 1 ➤ Prototipo 2
	Reutilización de Componentes	Reutilización de Componentes	Observación	<ul style="list-style-type: none"> ➤ Prototipo 1 ➤ Prototipo 2
	Documentación	<ul style="list-style-type: none"> ➤ Cantidad de Libros ➤ Idioma de la Documentación ➤ Videos Demostrativos ➤ Foros ➤ Páginas Oficiales 	<ul style="list-style-type: none"> Revisión de documentos Revisión de videos. 	<ul style="list-style-type: none"> ➤ Internet ➤ Libros ➤ Foros ➤ Videos ➤ Páginas Oficiales
	Diseño	Diseño	Observación	<ul style="list-style-type: none"> ➤ Prototipo 1 ➤ Prototipo 2
	Tiempo de desarrollo	Tiempo de desarrollo	Observación	<ul style="list-style-type: none"> ➤ Prototipo 1 ➤ Prototipo 2

Hi: El Framework Tapestry brinda las mejores prestaciones de productividad en el desarrollo del sistema financiero en el INSTITUTO PARTICULAR SAN GABRIEL.

3.7.6 Justificación y resultado de la hipótesis

Una vez realizado el análisis y la comparación de los parámetros e indicadores de productividad en cuanto al número de líneas de código, curva de aprendizaje, reutilización de componentes, documentación, diseño y tiempo de desarrollo, se logró demostrar que el framework TAPESTRY permite mejorar la productividad en el desarrollo del sistema financiero en el INSTITUTO PARTICULAR SAN GABRIEL, aplicando estadística descriptiva que se encarga de recolectar, ordenar, analizar y representar gráficamente, y la estadística inferencial aplicando la Distribución T - Student para comprobar la hipótesis planteando una hipótesis nula.

DISTRIBUCIÓN T – STUDENT

Para comprobar la hipótesis se utilizara la distribución T – Student, planteando una hipótesis nula (Ho).

Planteamiento de la Hipótesis Nula (Ho)

Ho = El Framework Tapestry no brinda las mejores prestaciones de productividad en el desarrollo del sistema financiero en el INSTITUTO PARTICULAR SAN GABRIEL.

$$\text{Fórmula: } t = \frac{\tilde{X} - U}{\frac{S}{\sqrt{n}}}$$

\tilde{X} = Promedio n observaciones

U = Media

S = Desviación Estándar

n = Muestra

Promedio n Observaciones

Para obtener las medias de n observaciones, se tiene la siguiente fórmula:

$$\text{Fórmula: } \tilde{X} = \sum_{i=1}^n Xi = \frac{X1+X2+X3+\dots +Xn}{n}$$

n = total de los datos

Media

Para obtener la media poblacional, se tiene la siguiente fórmula:

$$\text{Fórmula: } U = \sum_{i=1}^n Ui = \frac{U1+U2+U3+\dots +Un}{n}$$

Desviación Estándar

Para poder obtener la Desviación Estándar primero debemos obtener la Varianza, se tiene la siguiente fórmula:

Varianza

$$\text{Fórmula: } S^2 = \sum_{i=1}^n \frac{(X_i - \bar{X})^2}{n-1}$$

Desviación Estándar

Para obtener la desviación estándar se saca la raíz cuadrada de la varianza, se tiene la siguiente fórmula:

$$\text{Fórmula: } S = \sqrt{S^2}$$

Procedimiento:

A continuación se procederá a calcular la Distribución T – Student obteniendo el procedimiento de n observaciones, la media, y la desviación estándar.

$$\text{Promedio n observaciones: } \tilde{X} = \sum_{i=1}^n X_i = \frac{X_1 + X_2 + X_3 + \dots + X_n}{n}$$

$$\tilde{X} = \sum_{i=1}^n X_i = \frac{X_1 + X_2 + X_3 + X_4 + X_5 + X_6}{n}$$

WICKET

$$\tilde{X}_w = \frac{X_1 + X_2 + X_3 + X_4 + X_5 + X_6}{n}$$

$$\tilde{X}_w = \frac{1.5 + 2 + 4 + 1.2 + 2 + 3}{6}$$

$$\tilde{X}_w = \frac{13.7}{6}$$

$$\tilde{X}_w = 2.28333$$

TAPESTRY

$$\tilde{X}t = \frac{X1+X2+X3+X4+X5+X6}{n}$$

$$\tilde{X}t = \frac{2.5+1+4+2.8+4+2}{6}$$

$$\tilde{X}t = \frac{16.3}{6}$$

$$\tilde{X}t = 2.71667$$

$$\text{Media: } U = \sum_{i=1}^n Ui = \frac{U1+U2+U3+\dots+Un}{n}$$

$$U = \sum_{i=1}^n Ui = \frac{U1+U2+U3+U4+U5+U6}{n}$$

WICKET

$$Uw = \frac{U1+U2+U3+U4+U5+U6}{6}$$

$$Uw = \frac{4+4+4+4+4+4}{6}$$

$$Uw = \frac{24}{6}$$

$$Uw = 4$$

TAPESTRY

$$Ut = \frac{U1+U2+U3+U4+U5+U6}{6}$$

$$Ut = \frac{4+4+4+4+4+4}{6}$$

$$Ut = \frac{24}{6}$$

$$Ut = 4$$

Varianza: $S^2 = \sum_{i=1}^n \frac{(X_i - \bar{X})^2}{n-1}$

$$S^2 = \sum_{i=1}^n \frac{(X_i - \bar{X})^2}{n-1}$$

WICKET

$$Sw^2 = \frac{(X_1 - \bar{X})^2 + (X_2 - \bar{X})^2 + (X_3 - \bar{X})^2 + (X_4 - \bar{X})^2 + (X_5 - \bar{X})^2 + (X_6 - \bar{X})^2}{6-1}$$

$$Sw^2 =$$

$$\frac{(1.5 - 2.28333)^2 + (2 - 2.28333)^2 + (4 - 2.28333)^2 + (1.2 - 2.28333)^2 + (2 - 2.28333)^2 + (3 - 2.28333)^2}{5}$$

$$Sw^2 = \frac{0.61361 + 0.08028 + 2.94696 + 1.17360 + 0.08028 + 0.51362}{5}$$

$$Sw^2 = \frac{5.40835}{5}$$

$$Sw^2 = 1.08167$$

TAPESTRY

$$St^2 = \frac{(X_1 - \bar{X})^2 + (X_2 - \bar{X})^2 + (X_3 - \bar{X})^2 + (X_4 - \bar{X})^2 + (X_5 - \bar{X})^2 + (X_6 - \bar{X})^2}{6-1}$$

$$St^2 =$$

$$\frac{(2.5 - 2.71667)^2 + (1 - 2.71667)^2 + (4 - 2.71667)^2 + (2.8 - 2.71667)^2 + (4 - 2.71667)^2 + (2 - 2.71667)^2}{5}$$

$$St^2 = \frac{0.04695 + 2.94696 + 1.64694 + 0.00694 + 1.64694 + 0.51362}{5}$$

$$St^2 = \frac{6.80835}{5}$$

$$St^2 = 1.36167$$

Desviación Estándar: $S = \sqrt{S^2}$

$$S = \sqrt{S^2}$$

WICKET

$$S_w = \sqrt{S_w^2}$$

$$S_w = \sqrt{1.08167}$$

$$S_w = 1.04003$$

TAPESTRY

$$S_t = \sqrt{S_t^2}$$

$$S_t = \sqrt{1.36167}$$

$$S_t = 1.16691$$

Distribución T - Student: $t = \frac{\tilde{X} - U}{\frac{S}{\sqrt{n}}}$

$$t = \frac{\tilde{X} - U}{\frac{S}{\sqrt{n}}}$$

WICKET

$$tw = \frac{\tilde{X}_w - U_w}{\frac{S_w}{\sqrt{n}}}$$

$$tw = \frac{2.28333 - 4}{\frac{1.04003}{\sqrt{6}}}$$

$$tw = \frac{-1.71667}{\frac{1.04003}{2.44949}}$$

$$tw = \frac{-1.71667}{0.42459}$$

$$tw = - 4.04312$$

TAPESTRY

$$Tt = \frac{\widetilde{Xt} - Ut}{\frac{St}{\sqrt{n}}}$$

$$Tt = \frac{2.71667 - 4}{\frac{1.16691}{\sqrt{6}}}$$

$$Tt = \frac{-1.28333}{\frac{1.16691}{2.44949}}$$

$$Tt = \frac{-1.28333}{0.47639}$$

$$Tt = - 2.69386$$

Interpretación:

Al analizar mediante la Distribución T - Student se pudo determinar que la hipótesis nula **Ho** se rechaza y se acepta la hipótesis planteada inicialmente **Hi**, en donde se concluye que el Framework Tapestry brinda mejores prestaciones de productividad que el Framework Wicket.

CAPÍTULO IV

4 Implementación del Sistema Financiero en el Instituto Particular San Gabriel.

4.1 Introducción

Para la implementación del sistema financiero en el Instituto Particular San Gabriel se utilizara la metodología MSF (Microsoft Solution Framework), la cual se siguió el modelo de algunos informes de prácticas, así también como de tesis ubicado en el Centro de Documentación [5], y se obtuvo el siguiente modelo.

Es una metodología ágil que aporta las mejores prácticas para crear, plantear y desarrollar soluciones a problemas empresariales, con una serie de conceptos, modelos y prácticas de uso. La cual permitirá identificar los beneficios y objetivos, los requerimientos, sus alcances, limitaciones y los riesgos esenciales del sistema financiero.

La metodología MSF está compuesta por las siguientes fases:



Figura IV.35 Fases de la Metodología MSF

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

4.2 Desarrollo del Sistema Financiero San Gabriel (SFSG)

4.2.1 Fase de Visión

Para el desarrollo eficaz del proyecto es importante obtener una visión del proyecto compartida, comunicada, desarrollada y ordenada con los objetivos del sistema. Además identificar los beneficios, requerimientos funcionales, sus alcances y limitaciones; y los riesgos esenciales al proceso.

4.2.1.1 Definición del Problema

Ante la necesidad de facilitar al Instituto Particular San Gabriel requiere la implementación de un sistema financiero para automatizar los procesos financieros que replazce el trabajo manual de los administradores quienes constantemente cumplen con esta labor.

4.2.1.2 Situación Actual

En la actualidad el Instituto Particular San Gabriel de la ciudad de Riobamba no cuenta con un sistema Financiero. El problema se presenta al momento de registrar los pagos de las inscripciones, matriculas, solicitudes, derechos de exámenes, etc. que son realizadas por los administrativos del instituto de forma manual.

4.2.1.3 Visión del Sistema

El Sistema a ser implementado se encargara de automatizar el proceso de los pagos que realizan los estudiantes dentro del Instituto. El Sistema también permitirá gestionar los datos de los alumnos, carreras, semestres, los pagos que los estudiantes realizan y en lo cual se implementa los métodos CRUD necesarios para poder implementar el sistema.

4.2.1.4 Determinación y Análisis de Requerimientos

4.2.1.4.1 Requerimientos Funcionales

Los requerimientos funcionales del sistema, los cuales fueron obtenidos mediante el planteamiento del problema y de las necesidades del Instituto (ver Anexo 2).

4.2.1.4.2 Requerimientos No Funcionales

- Rendimiento
- Seguridad
- Mantenibilidad
- Escalabilidad
- Interfaces
- Usabilidad

4.2.1.5 Objetivos para el Sistema

➤ Objetivos del Sistema:

- Permitir el registro de los pagos automáticamente realizados por la secretaria.
- Almacenar la información de los alumnos, carreras, semestres, pagos, estado civil y requerimientos en la Base de Datos.
- Presentar un recibo para los alumnos como prueba del pago realizado.

➤ Objetivos del Diseño:

- Implementar interfaces amigables para el usuario.

4.2.1.6 Riesgos

4.2.1.6.1 Identificación del Riesgo

En este proyecto se tomara en cuenta tres tipos de riesgo:

- Riesgo del proyecto.
- Riesgo técnico.
- Riesgo del sistema.

La identificación del riesgo se los puede ver detalladamente en el (Anexo 3).

4.2.1.6.2 Análisis de Riesgos

En el análisis de riesgos se va a describir la valoración, la probabilidad, el impacto, la exposición, y el resumen de los riesgos en los cuales se va a determinar las prioridades de los riesgos que se pueden presentar en sistema SFSG, los cuales se puede ver detalladamente en el (Anexo 4).

4.2.1.6.3 Planeación y programación del riesgo

Para la planeación y programación del riesgo se va a describir las gestiones de riesgo, en las cuales se describirá las causas, consecuencias, el estado actual de los riesgos, los cuales se pueden ver en el (Anexo 5).

4.2.2 Fase de Planificación

Se obtiene un cronograma de trabajo que se ajuste a lo especificado en la fase de visión, se desarrollara los requerimientos funcionales y no funcionales, se describirá el escenario, los cuales van a ser desarrollados.

4.2.2.1 Planificación Inicial

4.2.2.1.1 Factibilidad

Factibilidad Técnica

Permite determinar si la propuesta puede ser implementada con el hardware, software y

recurso humano disponible.

Para el desarrollo de la aplicación web SFSG se cuenta con casi todos los recursos hardware y software necesarios. A continuación se detalla el hardware, software existente, requerido así como también el personal técnico requerido para el desarrollo del mismo.

Hardware Existente

Hardware con el que se cuenta para el desarrollo de la aplicación es el siguiente:

Tabla IV.XXX Hardware Existente

CANTIDAD	DESCRIPCIÓN	OBSERVACIONES
1	Laptop	Desarrollo de la aplicación y documentación
1	Modem CNT	Acceder al internet para consultar las dudas en el desarrollo de la aplicación y realizar las pruebas

Hardware Requerido

Hardware requerido para el desarrollo de la aplicación es el siguiente:

Tabla IV.XXXI Hardware Requerido

CANTIDAD	DESCRIPCIÓN	OBSERVACIONES
1	Impresora	Imprimir comprobantes de pagos

Software Existente

El Software que se necesita para el desarrollo del sistema es el siguiente:

Tabla IV.XXXII Software Existente

NOMBRE	DESCRIPCIÓN
Windows 8	Sistema Operativo
ArgoUML	Herramienta de diseño UML
NetBeans 7.2	IDE de desarrollo
Tapestry 5.3.7	Framework de ayuda
JAVA	Lenguaje de programación

Software Requerido

El Software requerido para el desarrollo del sistema es el siguiente:

Tabla IV.XXXIII Software Requerido

NOMBRE	DESCRIPCIÓN
Apache	Servidor web de distribución libre.

Recurso Humano Requerido

El Recurso Humano requerido para el desarrollo del sistema es el siguiente:

Tabla IV.XXXIV Recurso Humano Requerido

FUNCIÓN	FORMACIÓN
Desarrollador	Estudiante de Ingeniería en Sistemas
Diseñador	Estudiante de Ingeniería en Sistemas

Factibilidad Operativa

Recurso Humano

El recurso humano que participará en la operación del sistema es el siguiente:

Tabla IV.XXXV Recurso Humano participativo del sistema

NOMBRE	FUNCIÓN
U. Técnico	Encargado del Área Informática del Instituto Particular San Gabriel.

Factibilidad Económica

El tiempo de duración del proyecto será de 8 meses.

Costos

Costos de desarrollo Costos

Personal Mensual Total

Jefe de Proyecto y Desarrollador

\$400 \$3200.00

Costo Personal Total \$3200.00

Costo de hardware y software Costo

Hardware

Impresora \$100,00

Costo Software

Internet \$160,00

Costos varios

Suministros \$400,00

Alimentación \$400.00

Papel A4 \$10.00

Costo Hardware y Software Total \$1070.00

Costos Total de desarrollo \$4270,00

Análisis costo-beneficio

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

Permitirá realizar los pagos de manera rápida y eficaz, almacenar información importante que permita obtener reportes exactos de la situación actual de los pagos

realizados y así se podrá tomar decisiones que mejoren la calidad en la atención por parte de la secretaria.

Se podrá tener un registro de todos los alumnos, carreras, semestres, pagos y requisitos del instituto san Gabriel. Se optimizará el tiempo en la elaboración de pagos por parte de los alumnos.

4.2.2.2 Especificación de Requerimientos

1. ESPECIFICACIÓN FUNCIONAL

1.1. Diseño Conceptual

1.1.1. Requerimientos Funcionales

1.1.1.1 Requerimiento Funcional 1

Especificaciones

Introducción

El sistema podrá gestionar los datos de las carreras.

Entrada

Fuentes de Entrada

Id carrera

Carrera

Frecuencia

Bajo demanda

Requisitos de control

- Controla que los campos del formulario no estén vacíos.

- En la actualización de los datos evita la modificación del campo id carrera.

Entradas válidas

Todos los campos sean válidos

Procesos

1. La Secretaria selecciona el proceso a realizar
2. Si La Secretaria selecciona “Nueva Carrera”
 - 2.1. La Secretaria llena los datos del formulario “Nueva Carrera”
 - 2.2. El sistema validará los datos.
 - 2.3. Si los datos son correctos
 - 2.3.1. El sistema guarda los datos Caso contrario
 - 2.3.2. Mensaje de error
3. Si la Secretaria selecciona “Actualizar Carrera”
 - 3.1. La Secretaria selecciona la carrera a modificar
 - 3.2. Llena los campos del formulario “Actualizar Carrera”
 - 3.3. El sistema validará los datos.
 - 3.4. Si los datos son correctos
 - 3.4.1. El sistema guarda los datos Caso contrario
 - 3.4.2. Mensaje de error
4. Si La Secretaria selecciona “Eliminar Carrera”
 - 4.1. La Secretaria selecciona la carrera a eliminar.
 - 4.1.1 El sistema elimina los datos Caso contrario
 - 4.1.2 Mensaje de error

Salidas

Destino de las salidas

El formulario se cierra al guardar los datos y se muestra la lista de carreras.

Interfaces de Hardware

- El monitor será el medio de visualización utilizado para mostrar cada uno de los procesos que se efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará es JAVA, Netbeans 7.2, Apache Tomcat y Tapestry 5.3.7 para cumplir con el requerimiento.
- La Base de Datos está implementada en MYSQL.

El resto de requerimientos funcionales se los puede ver detalladamente en el (Anexo 6).

4.2.3 Requerimientos No Funcionales

A continuación se muestran los requerimientos no funcionales:

4.2.3.1 Rendimiento

El tiempo de respuesta para acceder a cualquier página sea de 10 a 20 segundos.

4.2.3.2 Seguridad

Para ingresar al sistema SFSG solo tendrá acceso la secretaria la cual estará bajo responsabilidad de la misma.

4.2.3.3 Mantenibilidad

Se emplea el Modelo Microsoft Solution Framework y se documentará el diseño y la

codificación de la solución.

4.2.3.4 Escalabilidad

Diseño de la arquitectura empleando módulos para el desarrollo de la aplicación web.

4.2.3.5 Interfaces

Interfaces realizadas en Netbeans 7.2, con la ayuda de Adobe DreamWeaver CS5.

4.2.3.6 Usabilidad

Facilidad de uso para el usuario.

4.2.4 Actores

4.2.4.1 Administrador

Será el encargado de administrar en su totalidad el sistema desarrollado.

- Ingresar datos.
- Modificar datos.
- Eliminar datos.
- Generar comprobantes de pago.
- Imprimir los comprobantes de pago.
- Visualizar comprobantes de pago.

4.2.4.2 Diseño Lógico

➤ Casos de Uso

Diagrama de Casos de Uso Administrador del Sistema

A continuación se presenta el diseño lógico de los procesos que se realiza en la SFSG, para hacer una representación más amigable se realizó el diseño de casos de uso como se muestra en la **Figura IV.36**.

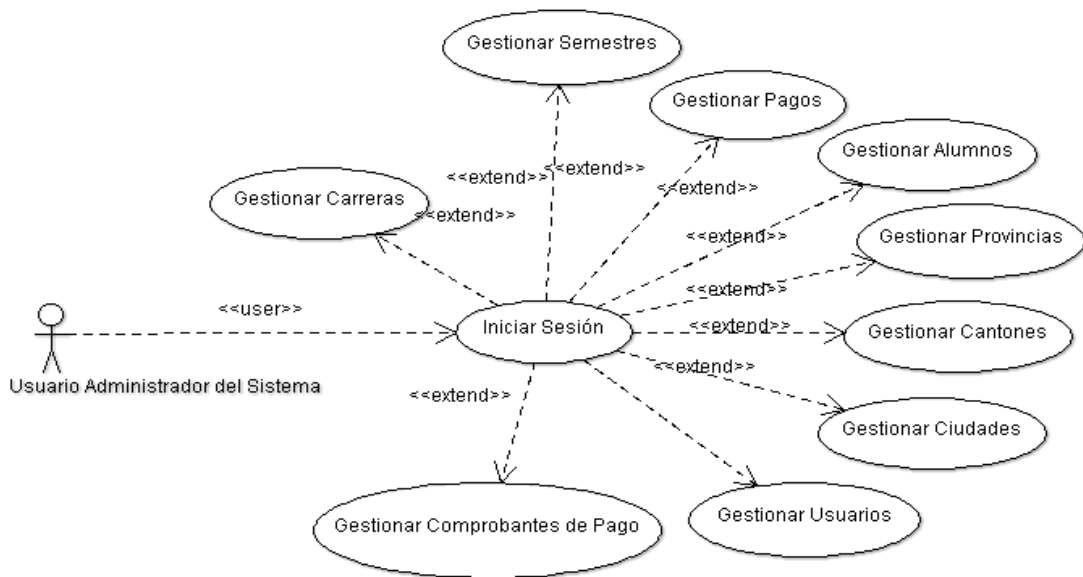


Figura IV.36 Caso de Uso Administrador del Sistema

➤ **Diagrama de Secuencia**

En el diagrama de secuencia se puede describir el funcionamiento de los procesos que realiza el sistema.

Diagrama de secuencia para autenticar el usuario.

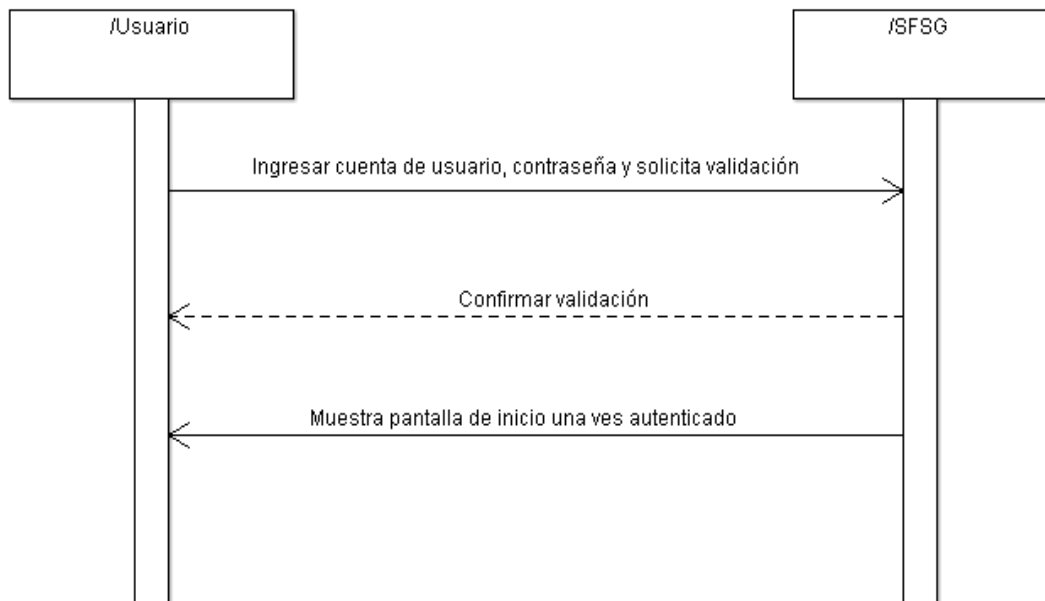


Figura IV.37 Diagrama de Secuencia Autenticación de Usuarios.

Para poder ver todos los diagramas de secuencia se puede ver en el (Anexo 7).

➤ **Diagrama de clases**

En el diagrama de clases se va a describir las clases, métodos y atributos como se observa en la **Figura IV. 38**.

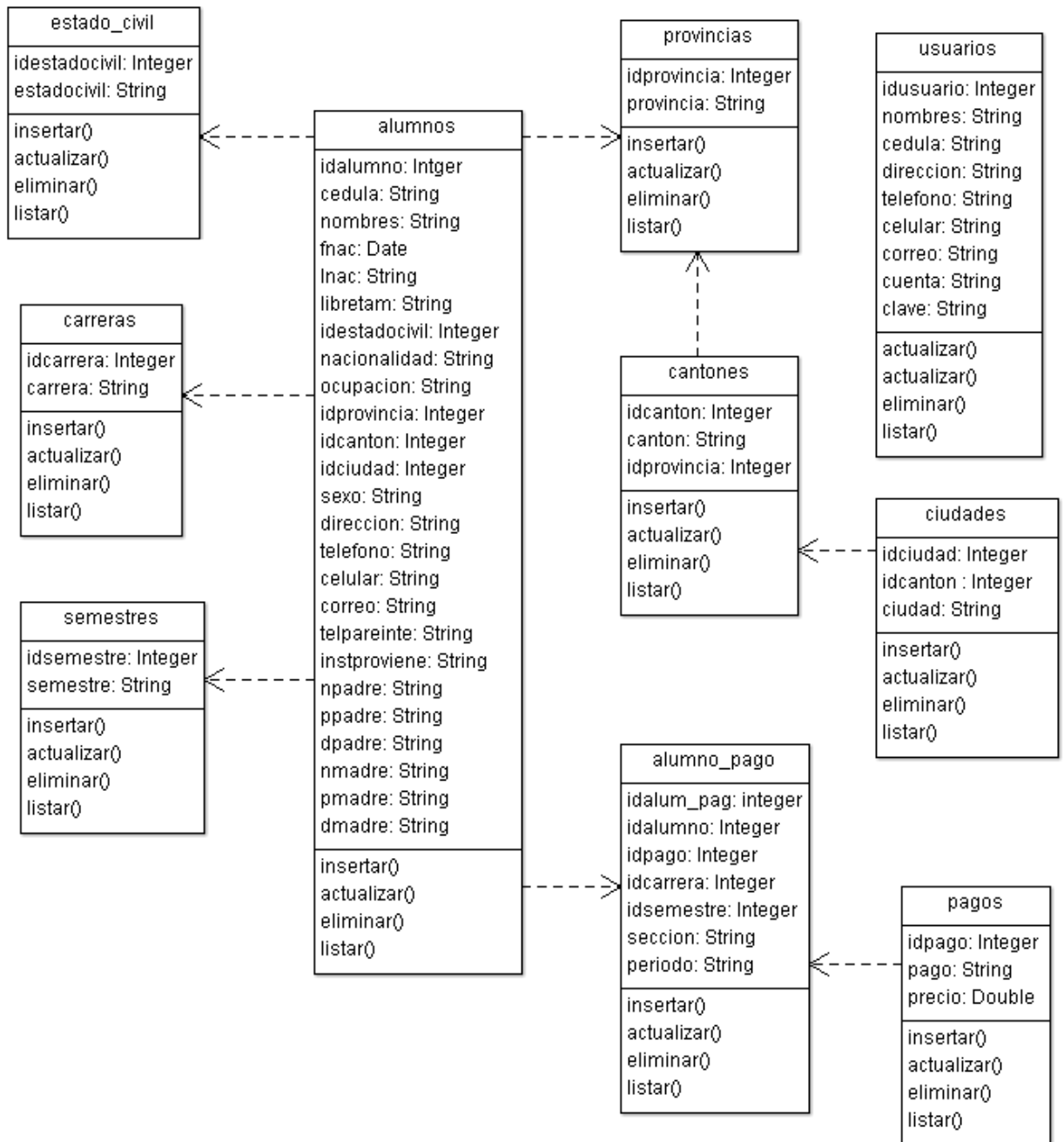


Figura IV.38 Diagrama de Clases

➤ Interfaces de Usuario

Para el desarrollo de las Interfaces de Usuario se determinó un esquema gráfico que posee las siguientes características: color, funciones, campos del formulario HTML, fecha y otros componentes importantes para el desarrollo del sistema.

Pantalla de Autenticación en el sistema SFSG:



Figura IV.39 Pantalla de Inicio de sesión en el sistema SFSG.

Para poder observar todas las interfaces más importantes del sistema SFSG se los puede ver en el (Anexo 8).

4.2.4.3 Diseño Físico

➤ Diagrama de Actividades

A continuación se presenta el diagrama de actividades para poder manipular el sistema, como se muestra en la **Figura IV. 40**.

Diagrama de Actividades para manipular el sistema SFSG.

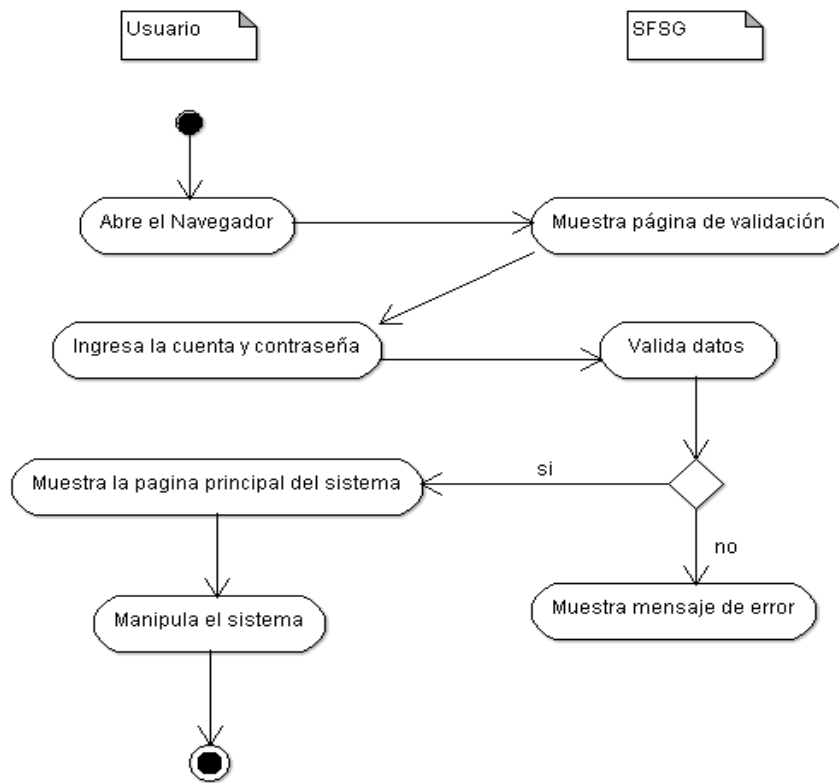


Figura IV.40 Diagrama de Actividades para manipular el sistema SFSG.

➤ **Diagrama de Componentes**

En el diagrama de componentes se puede observar la estructura el sistema, como se muestra en la **Figura IV. 41**.

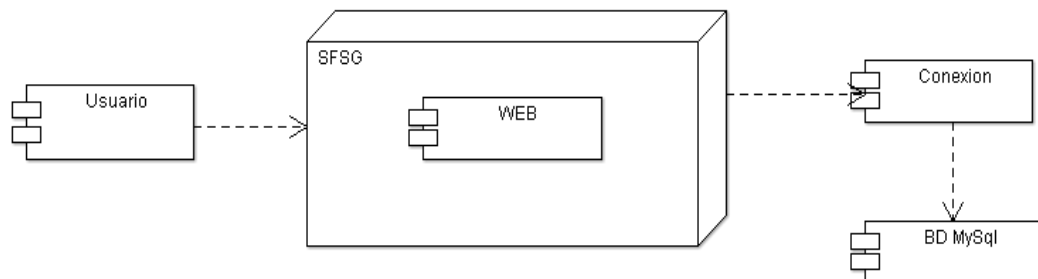


Figura IV.41 Diagrama de componentes

➤ **Diagrama de Implementación**

En el diagrama de implementación se puede ver cómo va a funcionar el sistema en el Instituto, como se observa en la **Figura IV. 42**.

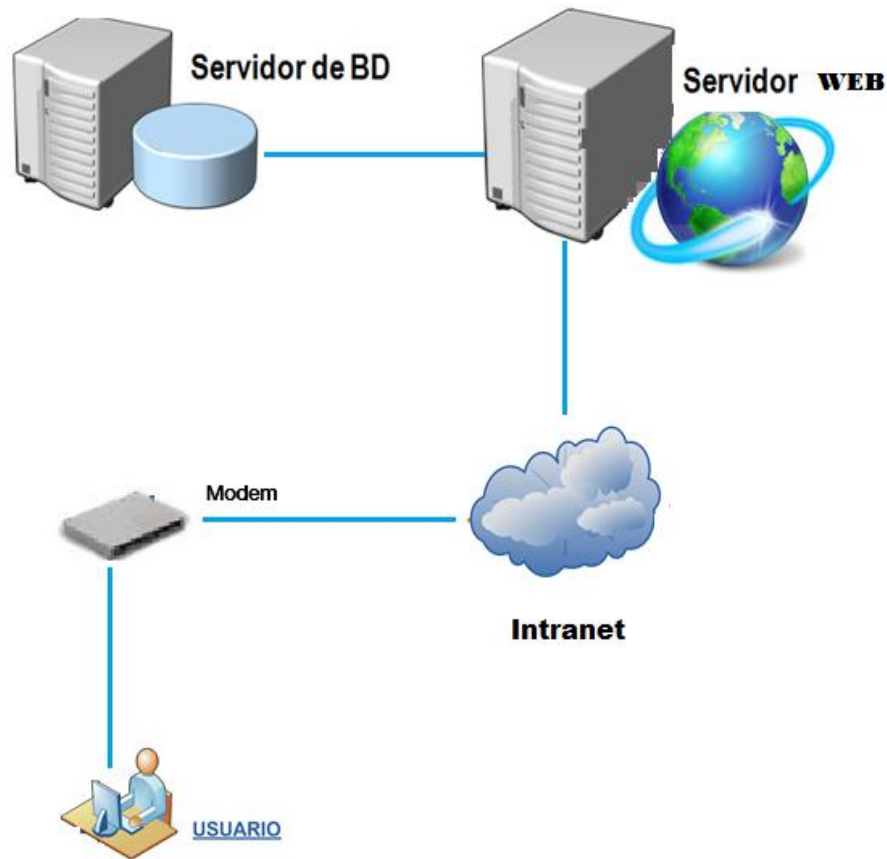


Figura IV.42 Diagrama de implementación

➤ **Modelo físico de la base de datos**

En el modelo físico se puede observar la estructura de las tablas de la base de datos del sistema, como se muestra en la **Figura IV. 43**.

4.2.5 Fase de Desarrollo

4.2.5.1 Nomenclatura y Estándares

La siguiente tabla muestra los estándares manejados en el desarrollo del sistema.

Tabla IV.XXXVI Nomenclatura y Estándares

ARCHIVO	EXTENSIÓN	NOMENCLATURA
Programas en Java	.java	NombreDeArchivo.java
Imágenes	.png	NombreDeImagen.png
	.jpg	NombreDeImagen.jpg
Páginas Web	.tml	NombreDeArchivo.tml

4.2.5.2 Capa de Datos

4.2.5.2.1 Diccionario de Datos

En esta sección se va a describir cada una de las tablas de la base de datos del Instituto, el cual nos interesa para conocer en detalle cada una de las columnas que componen las tablas.

Tabla IV.XXXVII Diccionario de Datos

TABLA: carreras				
Nombre Campo	Tipo dato	longitud	Clave Primaria	Calculado
Idcarrera	Number	11	x	No
Carrera	Varchar	60		No
Observación	Text			No

Para ver el Diccionario de Datos de las demás tablas con sus respectivos campos, puede ver en el (Anexo 9).

4.2.5.2.2 Implementación de la base de datos

Para implementar la base de datos se utilizó MySQL (AppServer 2.5.10) para implementar la base de datos del Instituto San Gabriel.

Sistema de Autenticación

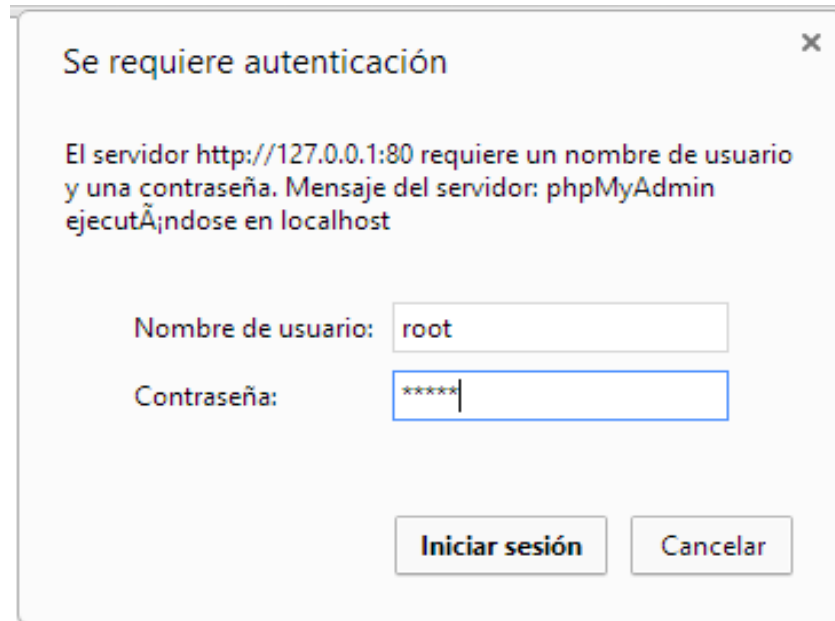


Figura IV.44 Sistema de Autenticación en MYSQL

Para poder observar más sobre la implementación de la base de datos, lo puede ver en el (Anexo 10).

4.2.5.2.3 Script de los procedimientos almacenados de la base de datos

Procedimiento para insertar los pagos de los alumnos

```
CREATE PROCEDURE sp_insertar_alumnopago (IN nidalumno int, IN nidpago int, IN nidcarrera int, IN nidsemestre int, IN nseccion varchar(60), IN nperiodo varchar(60))
BEGIN
    INSERT INTO alumno_pago
    (idalumno, idpago, idcarrera, idsemestre, seccion, periodo) VALUES
    (nidalumno, nidpago, nidcarrera, nidsemestre, nseccion, nperiodo);
END$$
```

Procedimiento para actualizar los pagos de los alumnos

```
CREATE PROCEDURE sp_actualizar_alumnopago (IN nidalum_pag int, IN nidalumno
int,IN nidpago int,IN nidcarrera int,IN nidsemestre int,IN nseccion varchar(60),IN nperiodo
varchar(60))
BEGIN
    UPDATE alumno_pago SET idalumno=nidalumno,
                                idpago=nidpago,
                                idcarrera=nidcarrera,
                                idsemestre=nidsemestre,
                                seccion=ncseccion,
                                periodo=nperiodo
    WHERE idalum_pag=nidalum_pag;
END$$
```

Procedimiento para actualizar los pagos de los alumnos

```
CREATE PROCEDURE sp_eliminar_alumnopago (IN nidalum_pag int)
BEGIN
    DELETE FROM alumno_pago WHERE idalum_pag=nidalum_pag;
END$$
```

Los demás procedimientos se los puede ver en el (Anexo 11).

4.2.6 Fase de Estabilización

Se validó la solución de la implementación de la aplicación WEB, mediante la realización de pruebas, para poder comprobar su correcto funcionamiento, como se puede observar en (Anexos 12).

4.2.6.1 Revisión general del sistema

4.2.6.1.1 Documentación

Creación de manuales para facilitar la utilización de la aplicación del sistema:

- Manual Técnico: Se explica en el Capítulo IV del presente documento.
- Manual de Usuario: Se encuentra en el módulo ayuda del sistema WEB.

4.2.7 Fase de Implementación

4.2.7.1 Requisitos Hardware

Para el correcto funcionamiento del sistema financiero del Instituto Particular San

Gabriel debe cumplir con los siguientes requerimientos:

El servidor deberá reunir los siguientes requisitos mínimos:

- Disponer de 5Gb de espacio libre en el Disco Duro.
- Disponer mínimo de 2Gb de memoria RAM.
- Procesador doble núcleo Core 2 Duo.

4.2.7.2 Requisitos Software

El servidor deberá reunir los siguientes requisitos de software:

- Apache Tomcat.
- Tapestry 5.3.7.
- AppServer 2.5.10 (MySQL 5.0.5).
- Navegador Web Mozilla firefox v 2.0 o superior.

4.2.7.3 Plan de instalación

Para la instalación inicial del sistema en el equipo servidor, se debe cumplir los siguientes pasos:

- Instalar el servidor de aplicaciones Apache Tomcat.
- Instalar el motor de base de datos MySQL 5.0.5. (AppServer 2.5.10).
- Instalar el framework Tapestry 5.3.7.
- Restaurar la base de datos.
- Copiar el archivo.war del SFSG.
- Iniciar el servidor de aplicaciones.

4.2.7.4 Tareas a realizar

La aplicación ha sido culminada y entregada al Instituto Particular San Gabriel.

Además se entregará la documentación técnica, con el propósito de facilitar la utilización del sistema.

CONCLUSIONES

- La investigación de los frameworks WICKET y TAPESTRY permitió determinar que cada framework maneja archivos de configuración similares, de igual manera que el desarrollar una aplicación con cualquiera de los frameworks no resulta costoso debido a que la licencia es de código abierto, sin dejar de ser fiel a los objetivos originales del Grupo Apache y de apoyo al desarrollo de colaboración.
- El contar con una lista de parámetros medibles o comprobables facilitó la comparación entre los framework de estudio, permitiendo determinar que TAPESTRY es el adecuado en cuanto a la productividad que brinda para el desarrollo de aplicaciones web al superar en un 10.83% a WICKET, por lo que la aplicación web utilizada para acceder a los datos provenientes del Sistema Financiero del Instituto Particular San Gabriel fue desarrollada con el framework TAPESTRY.
- WICKET obtuvo una valoración de 75% en el parámetro tiempo de desarrollo mientras que TAPESTRY obtuvo un 50%, lo que significa que si el proyecto debe ser desarrollado en el menor tiempo posible la mejor opción sería utilizar el framework WICKET.
- El diseño resulta más fácil con el framework TAPESTRY esto pudo ser comprobado gracias a los prototipos implementados durante la investigación, los mismos que permitieron dar una valoración de 100% mientras que WICKET obtuvo 50%, es decir que si se desea un diseño amigable y personalizado para el usuario el framework recomendado es TAPESTRY.
- TAPESTRY es la mejor opción si lo que interesa en el desarrollo de una aplicación web es tener el menor número de líneas de código posible ya que en el parámetro Líneas de código TAPESTRY obtuvo una valoración de 62.5% mientras que WICKET obtuvo una valoración de 37.5%.

- La utilización de Netbeans IDE como entorno de desarrollo para la programación facilita el proceso y despliegue de la aplicación SFSG, ya que brinda un entorno amigable facilitando el orden de los componentes, clases y librerías de cada framework.
- La curva de aprendizaje es compleja para los dos frameworks de estudio ya que no cuentan con mucha información detallada, pero con lo investigado y analizado se pudo determinar que TAPESTRY es más complejo aprender que WICKET según su valoración de 25% y 50% respectivamente.
- La utilización de los archivos de configuración como web.xml y pom.xml son muy importantes en los frameworks de estudio ya que el archivo web.xml es utilizado para describir información de la aplicación y sus elementos, mientras que el archivo pom.xml se utiliza para agregar las dependencias necesarias de la aplicación web.

RECOMENDACIONES

- Antes de la construcción de los prototipos, se debe tener conocimientos sobre las ventajas, características, componentes, elementos y el tipo de programación que cada framework de estudio utiliza.
- El estudio de los frameworks con sus respectivos componentes deben ser guiadas por fuentes de investigación comprobadas, es decir obtener información confiable del sitio web oficial de TAPESTRY y de WICKET.
- Para implementar cualquier aplicación web ya sea WICKET o TAPESTRY para facilitar el trabajo es necesario instalar Maven para crear el esqueleto de la aplicación.
- Las aplicaciones desarrolladas en TAPESTRY deben estar en uso como una opción para satisfacer el trabajo profesional, lograr satisfacer necesidades y agilizar el trabajo ofertando servicios nuevos y eficientes.
- La configuración de los archivos XML son muy importantes por lo que se sugiere que se tomen un poco de tiempo en estos archivos para entender el funcionamiento de cada uno.

RESUMEN

Se determinó el análisis comparativo entre los framework WICKET y TAPESTRY para desarrollar aplicaciones web. Caso práctico: Instituto Particular San Gabriel de la ciudad de Riobamba de la provincia de Chimborazo.

Se desarrolló dos prototipos en cada uno de estos frameworks de estudio y se estableció una comparación entre estos, las funciones que realizó: recogió las tareas más importantes del sistema, el mismo que cumplió con dos tareas específicas, la tarea 1 gestionó los métodos Crear, Leer, Actualizar y Eliminar (CRUD) de carreras y la tarea 2 permitió al usuario inscribir al alumno, en la cual obtuvo una cantidad considerable de datos que permitieron obtener información relevante para el estudio comparativo de WICKET y TAPESTRY, se investigó y se seleccionó los parámetros de comparación e indicadores de productividad; como la curva de aprendizaje, líneas de código, diseño y tiempo de desarrollo que permitieron seleccionar el framework más eficaz para el desarrollo de aplicaciones web.

Como resultado en la comparación de parámetros el framework WICKET obtuvo 57.09% y TAPESTRY 67.92%.

La aplicación web con el framework TAPESTRY fue desarrollada en Java, utilizando la metodología Microsoft Solution Framework (MSF); permitió: gestión de alumnos, carreras; pagos: inscripciones, matrículas, solicitudes; generar comprobante de pago.

Mediante el análisis comparativo se concluye que el framework TAPESTRY brinda una mejor productividad en relación a WICKET.

Se recomienda el uso del framework TAPESTRY para el desarrollo de aplicaciones web, debido a la productividad que brinda en la implementación del Sistema Financiero.

SUMMARY

A comparative analysis was determined between WICKET and TAPESTRY frameworks to develop web applications. Case Study: San Gabriel Particular Institute in the city of Riobamba, Chimborazo Province.

Two prototypes were developed for each of the study frameworks and a comparison between both of them was established. The performed functions were the following: It collected the most important system jobs which had two specific tasks, task 1 managed the methods Create, Read, Update and Delete (CRUD) the different subjects, and task 2 allowed the user registering the student. This generated a considerable amount of data that allowed obtaining relevant information for the comparative study of WICKET and TAPESTRY. The productivity indicators and the comparison parameters were researched and selected. The showed the learning curve, code lines, and design and development time that permitted selecting the most efficient framework for the development of web applications.

As a result, the parameter comparison showed that WICKET got 57.09% and TAPESTRY got 67.92%.

The web application with TAPESTRY framework was developed in Java by using the Microsoft Solution Framework (MSF). It allowed the following: management of students, subjects, payments, enrolments, registrations, requests, payment receipt generation.

The comparative analysis concluded that TAPESTRY framework offers better productivity than WICKET.

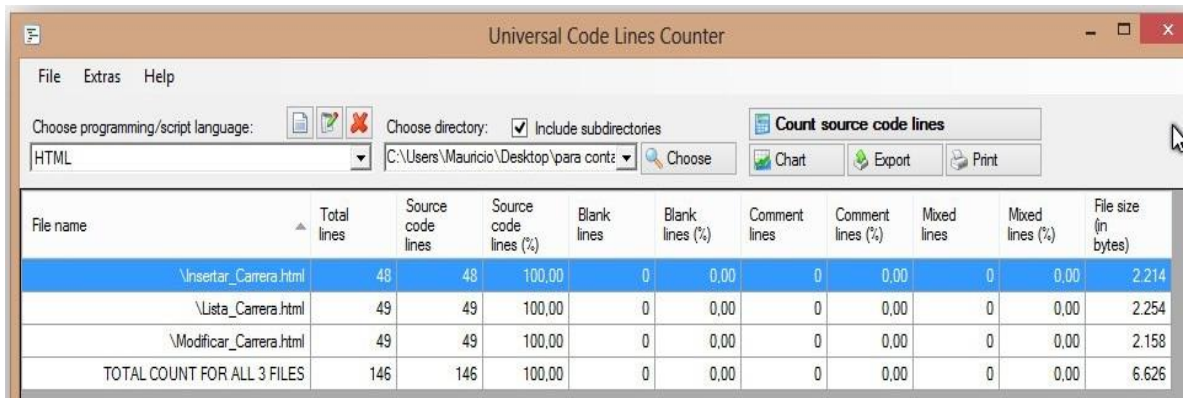
It is recommended to use TAPESTRY framework for the development of web applications since it yields good productivity when implementing the Financial System.

ANEXOS

ANEXO 1 Universal Code Lines Counter

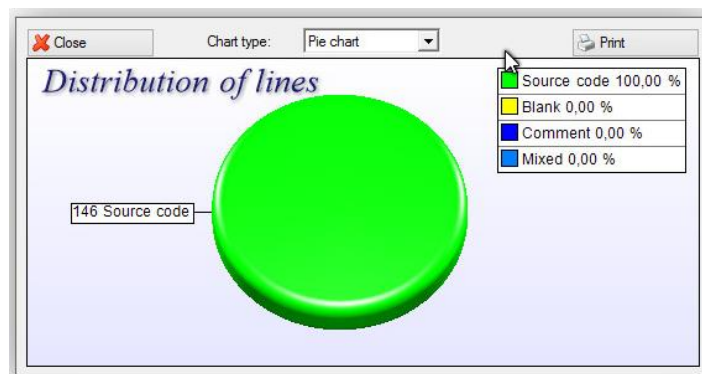
FRAMEWORK WICKET

Cantidad de Lineas de Código de las páginas Insertar_Carrera.html, Lista_Carrera.html y Modificar_Carrera.html.

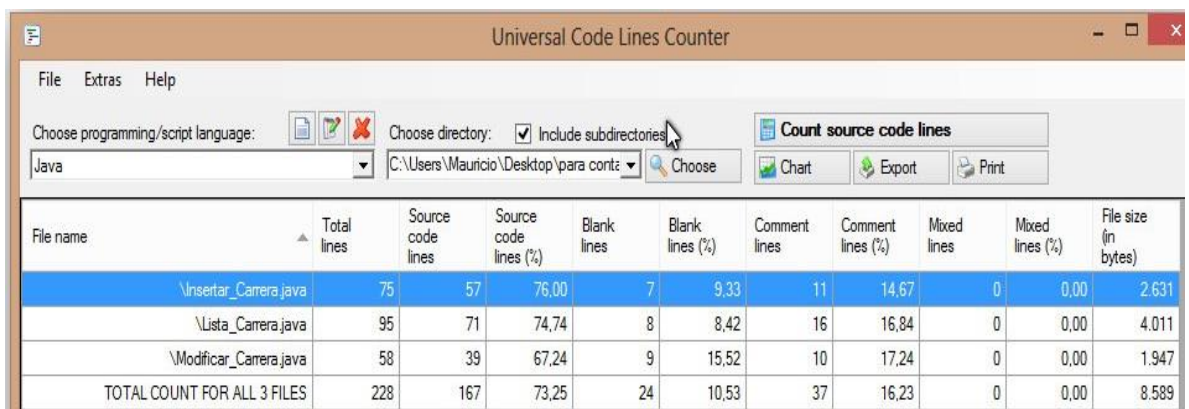


The screenshot shows the Universal Code Lines Counter application with the programming language set to HTML. The table below displays the analysis results for three HTML files.

File name	Total lines	Source code lines	Source code lines (%)	Blank lines	Blank lines (%)	Comment lines	Comment lines (%)	Mixed lines	Mixed lines (%)	File size (in bytes)
\\Insertar_Carrera.html	48	48	100,00	0	0,00	0	0,00	0	0,00	2.214
\\Lista_Carrera.html	49	49	100,00	0	0,00	0	0,00	0	0,00	2.254
\\Modificar_Carrera.html	49	49	100,00	0	0,00	0	0,00	0	0,00	2.158
TOTAL COUNT FOR ALL 3 FILES	146	146	100,00	0	0,00	0	0,00	0	0,00	6.626

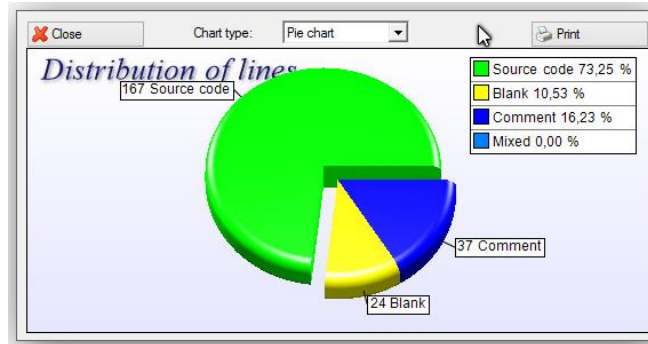


Cantidad de Lineas de Código de las clases Insertar_Carrera.java, Lista_Carrera.java y Modificar Carrera.java.



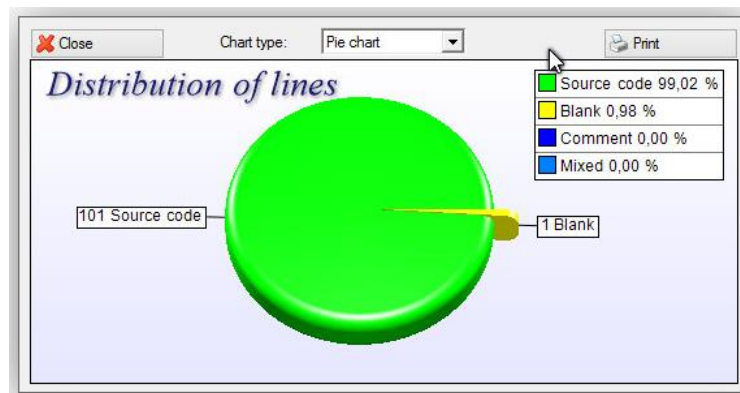
The screenshot shows the Universal Code Lines Counter application with the programming language set to Java. The table below displays the analysis results for three Java files.

File name	Total lines	Source code lines	Source code lines (%)	Blank lines	Blank lines (%)	Comment lines	Comment lines (%)	Mixed lines	Mixed lines (%)	File size (in bytes)
\\Insertar_Carrera.java	75	57	76,00	7	9,33	11	14,67	0	0,00	2.631
\\Lista_Carrera.java	95	71	74,74	8	8,42	16	16,84	0	0,00	4.011
\\Modificar_Carrera.java	58	39	67,24	9	15,52	10	17,24	0	0,00	1.947
TOTAL COUNT FOR ALL 3 FILES	228	167	73,25	24	10,53	37	16,23	0	0,00	8.589



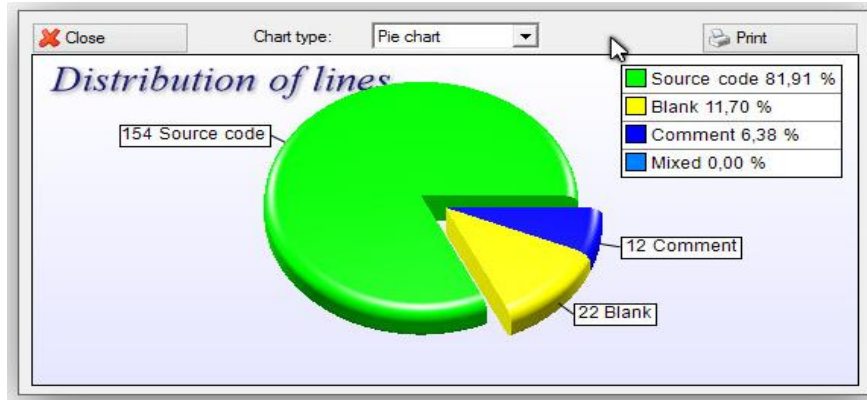
Cantidad de Lineas de Código de la página Insertar_Alumno_Pago.html.

File name	Total lines	Source code lines	Source code lines (%)	Blank lines	Blank lines (%)	Comment lines	Comment lines (%)	Mixed lines	Mixed lines (%)	File size (in bytes)
Insertar_Alumno_Pago.html	102	101	99.02	1	0.98	0	0.00	0	0.00	4.696
TOTAL COUNT FOR ALL 1 FILES	102	101	99.02	1	0.98	0	0.00	0	0.00	4.696



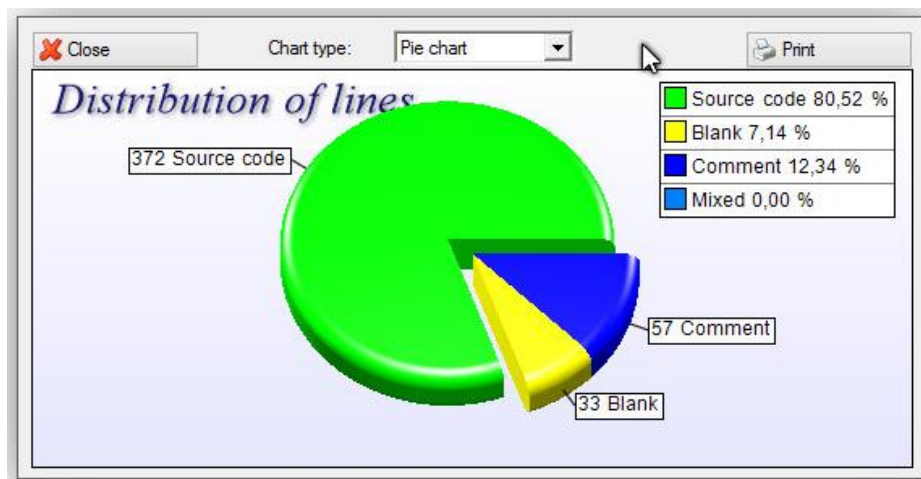
Cantidad de Lineas de Código de la clase Insertar_Alumno_Pago.java.

File name	Total lines	Source code lines	Source code lines (%)	Blank lines	Blank lines (%)	Comment lines	Comment lines (%)	Mixed lines	Mixed lines (%)	File size (in bytes)
Insertar_Alumno_Pago.java	188	154	81.91	22	11.70	12	6.38	0	0.00	8.020
TOTAL COUNT FOR ALL 1 FILES	188	154	81.91	22	11.70	12	6.38	0	0.00	8.020



Cantidad de Lineas de Código de las clases `Alumno_Pago.java` y `Carrera.java`, donde se realiza los métodos CRUD (Cread, Read, Update and Delete) con la base de datos.

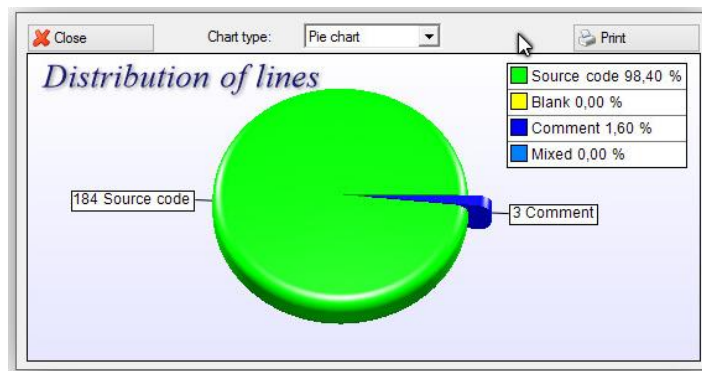
File name	Total lines	Source code lines	Source code lines (%)	Blank lines	Blank lines (%)	Comment lines	Comment lines (%)	Mixed lines	Mixed lines (%)	File size (in bytes)
Alumno_Pago.java	324	263	81,17	22	6,79	39	12,04	0	0,00	11.042
Carrera.java	138	109	78,99	11	7,97	18	13,04	0	0,00	4.628
TOTAL COUNT FOR ALL 2 FILES	462	372	80,52	33	7,14	57	12,34	0	0,00	15.670



FRAMEWORK TAPESTRY

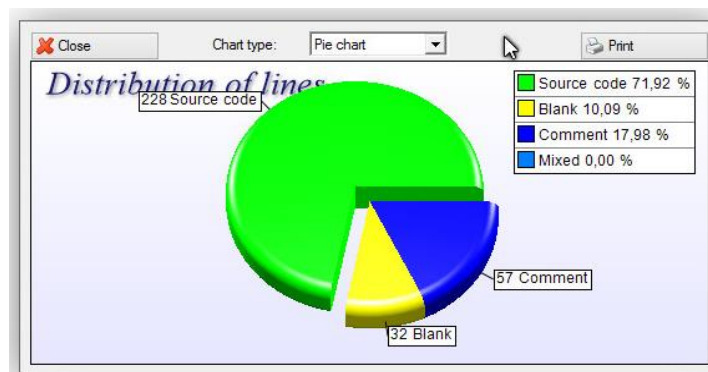
Cantidad de Lineas de Código de las páginas `InsertarCarrera.tml`, `ListaCarrera.tml`, `ActualizarCarrera.tml` e `InsertarAlumno_Pago.tml`.

File name	Total lines	Source code lines	Source code lines (%)	Blank lines	Blank lines (%)	Comment lines	Comment lines (%)	Mixed lines	Mixed lines (%)	File size (in bytes)
\ActualizarCarrera.tml	42	42	100,00	0	0,00	0	0,00	0	0,00	2.035
\Index.tml	13	12	92,31	0	0,00	1	7,69	0	0,00	536
\InsertarAlumno_Pago.tml	75	74	98,67	0	0,00	1	1,33	0	0,00	3.831
\InsertarCarrera.tml	35	34	97,14	0	0,00	1	2,86	0	0,00	1.613
\ListaCarrera.tml	22	22	100,00	0	0,00	0	0,00	0	0,00	1.133
TOTAL COUNT FOR ALL 5 FILES	187	184	98,40	0	0,00	3	1,60	0	0,00	9.148



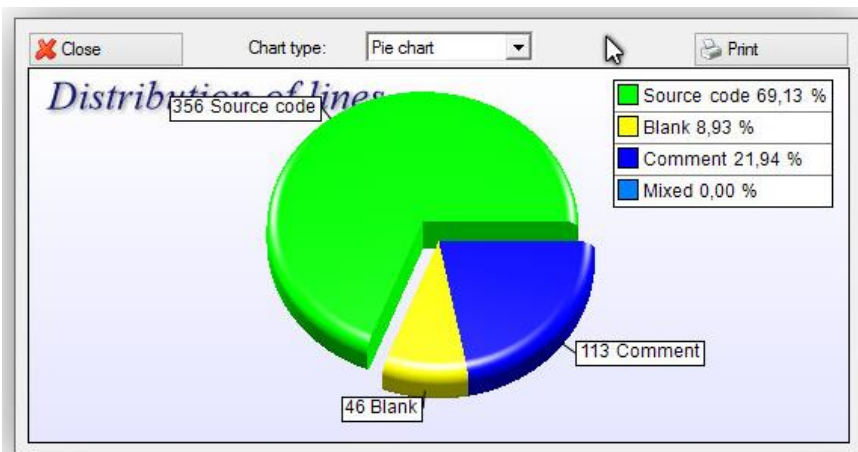
Cantidad de Lineas de Código de las clases InsertarCarrera.java, ListaCarrera.java, ActualizarCarrera.java e InsertarAlumno_Pago.java.

File name	Total lines	Source code lines	Source code lines (%)	Blank lines	Blank lines (%)	Comment lines	Comment lines (%)	Mixed lines	Mixed lines (%)	File size (in bytes)
\ActualizarCarrera.java	43	27	62,79	8	18,60	8	18,60	0	0,00	1.085
\Index.java	9	4	44,44	2	22,22	3	33,33	0	0,00	146
\InsertarAlumno_Pago.java	190	152	80,00	8	4,21	30	15,79	0	0,00	6.362
\InsertarCarrera.java	35	21	60,00	6	17,14	8	22,86	0	0,00	816
\ListaCarrera.java	40	24	60,00	8	20,00	8	20,00	0	0,00	939
TOTAL COUNT FOR ALL 5 FILES	317	228	71,92	32	10,09	57	17,98	0	0,00	9.348



Cantidad de Lineas de Código de las clases CarreraDAO.java, CarreraDAOImp.java, AlumnoPagoDAO.java y AlumnoPagoDAOImp.java, aquí se encuentran los servicios de Tapestry donde se manipulan los métodos CRUD con la base de datos.

File name	Total lines	Source code lines	Source code lines (%)	Blank lines	Blank lines (%)	Comment lines	Comment lines (%)	Mixed lines	Mixed lines (%)	File size (in bytes)
\AlumnoPagoDAO.java	22	12	54,55	2	9,09	8	36,36	0	0,00	833
\AlumnoPagoDAOImp.java	159	134	84,28	10	6,29	15	9,43	0	0,00	5.539
\AppModule.java	131	67	51,15	14	10,69	50	38,17	0	0,00	6.301
\CarreraDAO.java	20	10	50,00	2	10,00	8	40,00	0	0,00	559
\CarreraDAOImp.java	104	88	84,62	8	7,69	8	7,69	0	0,00	3.247
\DevelopmentModule.java	36	21	58,33	4	11,11	11	30,56	0	0,00	1.602
\QaModule.java	43	24	55,81	6	13,95	13	30,23	0	0,00	1.837
TOTAL COUNT FOR ALL 7 FILES	515	356	69,13	46	8,93	113	21,94	0	0,00	19.918



ANEXO 2 Requerimientos Funcionales

Req 1	Gestionar los datos de las carreras.
Req 2	Gestionar los datos de los alumnos.
Req 3	Gestionar los datos de los semestres.
Req 4	Gestionar los datos del estado civil.
Req 5	Gestionar los pagos realizados por los alumnos.
Req 6	Gestionar los datos de las provincias.
Req 7	Gestionar los datos de los cantones.
Req 8	Gestionar los datos de las ciudades.
Req 9	Gestionar los datos de los usuarios.
Req 10	Permitir la autenticación de un usuario.
Req 11	Generar el Comprobante de Pago.

ANEXO 3 Identificación del Riesgo

ID	DESCRIPCIÓN DEL RIESGO	CATEGORÍA	CONSECUENCIA
Ri1	Especificación de requerimientos Incompletos	RP	➤ Retardo en el proyecto. ➤ Pérdida de Tiempo.
Ri2	Interfaz de usuario mal definida	RT	➤ Interfaces complejas de usar para el usuario.
Ri3	Fecha de entrega muy apretada.	RS	➤ La implementación de la aplicación será apurada.
Ri4	Cambio en los requerimientos por parte del usuario	RS	➤ Pérdida de tiempo.

ANEXO 4 Análisis de Riesgos

Valoración del Riesgo

RANGO DE PROBABILIDAD	DESCRIPCIÓN	VALOR
1% - 33%	Baja	1
34% - 67%	Media	2
68% - 99%	Alta	3

Probabilidad

IDENTIFICACIÓN	PROBABILIDAD		
	%	VALOR	PROBABILIDAD
Ri1	20	1	BAJA
Ri2	40	2	MEDIA
Ri3	30	1	BAJA
Ri4	40	2	MEDIA

Impacto 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 meses	Proyecto no puede ser culminado	> 20%	4

Riesgo - Impacto

IDENTIFICACIÓN	IMPACTO	
	VALOR	IMPACTO
Ri1	2	MODERADO
Ri2	3	ALTO
Ri3	2	MODERADO
Ri4	3	ALTO

Exposición del Riesgo

EXPOSICIÓN AL RIESGO	VALOR	COLOR
Baja	1 o 2	Verde
Media	3 o 4	Amarillo
Alta	> 6	Rojo

Impacto - Probabilidad

Impacto / Probabilidad	Bajo =1	Moderado =2	Alto =3	Crítico =4
Alta = 3	3	6	9	12
Media = 2	2	4	6	8
Baja = 1	1	2	3	4

Resumen del Riesgo

Identificación	Probabilidad			Impacto		Exposición al riesgo	
	%	Valor	Probabilidad	Valor	Impacto	Valor	Exposición
R1	20	1	BAJA	2	MODERADO	2	BAJA
R2	40	2	MEDIA	3	ALTO	6	ALTA
R3	30	1	BAJA	2	MODERADO	2	BAJA
R4	40	2	MEDIA	3	ALTO	6	ALTA

Prioridades del Riesgo

IDENTIFICACIÓN	PRIORIDAD	EXPOSICIÓN
R2	1	6
R4	1	6
R1	2	2
R3	2	2

ANEXO 5 Planeación y Programación del Riesgo

Gestión del Riesgo 1

HOJA DE GESTIÓN DEL RIESGO			
ID DEL RIESGO: Ri1		FECHA:	
Probabilidad: Baja Valor: 1	Impacto: Moderado Valor: 2	Exposición: Baja Valor: 2	Prioridad: 2
DESCRIPCIÓN: Especificación de requerimientos incompleta.			

REFINAMIENTO:	
Causas:	
<ul style="list-style-type: none"> ➤ No existió una comunicación adecuada entre el usuario y el responsable del proyecto ➤ El usuario no explicó correctamente las necesidades que posee. 	
Consecuencias:	
<ul style="list-style-type: none"> ➤ Retraso del proyecto ➤ Pérdida de tiempo 	
REDUCCIÓN:	
<ul style="list-style-type: none"> ➤ Analizar las necesidades del usuario para establecer correctamente los requerimientos. ➤ Que exista una adecuada comunicación entre el cliente y el programador 	
SUPERVISIÓN:	
<ul style="list-style-type: none"> ➤ Ponerse de acuerdo al inicio: el cliente y el responsable acerca de sus necesidades. ➤ Que el ambiente de comunicación sea el más propicio entre el cliente y el responsable del proyecto. 	
GESTIÓN:	
<ul style="list-style-type: none"> ➤ Que una vez conocidas las necesidades del cliente se deberán poner de acuerdo el responsable del proyecto y cliente para establecer los requerimientos. 	
ESTADO ACTUAL:	
Fase de reducción iniciada:	<input checked="" type="checkbox"/>
Fase de supervisión iniciada:	<input type="checkbox"/>
Gestionando el riesgo:	<input type="checkbox"/>
RESPONSABLE:	
Edison Jara	

Gestión del Riesgo 2

HOJA DE GESTIÓN DEL RIESGO			
ID DEL RIESGO: Ri2		FECHA:	
Probabilidad: Medio Valor: 2	Impacto: Alto Valor: 3	Exposición: Alto Valor: 6	Prioridad: 1
DESCRIPCIÓN: Interfaz de usuario mal definida.			
REFINAMIENTO:			
Causas:			
<ul style="list-style-type: none"> ➤ El diseñador no elabore las interfaces con los datos necesarios para que el usuario se valide realice las operaciones básicas. 			
Consecuencias:			
<ul style="list-style-type: none"> ➤ Amenazan la calidad del software y la implementación puede llegar a ser difícil. 			
REDUCCIÓN:			
<ul style="list-style-type: none"> ➤ Diseñar correctamente las interfaces para validar el acceso al sistema de Gestión 			
SUPERVISIÓN:			
<ul style="list-style-type: none"> ➤ Revisar que el diseño de las interfaces sea el más óptimo para la Gestión. 			
GESTIÓN:			
<ul style="list-style-type: none"> ➤ Exigir al equipo de trabajo que las interfaces sean aprobadas unánimemente para garantizar la validación. 			
ESTADO ACTUAL:			
Fase de reducción iniciada:		<input checked="" type="checkbox"/>	
Fase de supervisión iniciada:		<input type="checkbox"/>	
Gestionando el riesgo:		<input type="checkbox"/>	
RESPONSABLE:			
Edison Jara			

Gestión del Riesgo 3

HOJA DE GESTIÓN DEL RIESGO			
ID DEL RIESGO: Ri3		FECHA:	
Probabilidad: Baja Valor: 1	Impacto: Moderado Valor: 2	Exposición: Baja Valor: 2	Prioridad: 2
DESCRIPCIÓN: Fecha de entrega muy apretada			
REFINAMIENTO: Causas: <ul style="list-style-type: none"> ➤ Falta de organización en el tiempo propuesto para el sistema. 			
Consecuencias: <ul style="list-style-type: none"> ➤ La implementación del proyecto será apurada. 			
REDUCCIÓN: <ul style="list-style-type: none"> ➤ Establecer un cronograma con tiempos accesibles para los desarrolladores. 			
SUPERVISIÓN: <ul style="list-style-type: none"> ➤ Revisar que las tareas se cumplan y sean entregadas según los tiempos establecidos. 			
GESTIÓN: <ul style="list-style-type: none"> ➤ Exigir al equipo de trabajo que las tareas sean cumplidas en los tiempos establecidos. 			
ESTADO ACTUAL: Fase de reducción iniciada: Fase de supervisión iniciada: Gestionando el riesgo:		<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
RESPONSABLE: Edison Jara			

Gestión del Riesgo 4

HOJA DE GESTIÓN DEL RIESGO			
ID DEL RIESGO: Ri4		FECHA:	
Probabilidad: Medio Valor: 2	Impacto: Alto Valor: 3	Exposición: Alto Valor: 6	Prioridad: 1
DESCRIPCIÓN: Cambio en los requerimientos por parte del usuario			
REFINAMIENTO: Causas: <ul style="list-style-type: none"> ➤ Los procesos del negocio pueden cambiar. 			
Consecuencias: <ul style="list-style-type: none"> ➤ Pérdida de tiempo. 			
REDUCCIÓN: <ul style="list-style-type: none"> ➤ Predecir posibles cambios a suceder en el negocio. 			
SUPERVISIÓN: <ul style="list-style-type: none"> ➤ Estar en constante comunicación con el usuario. 			
GESTIÓN: <ul style="list-style-type: none"> ➤ Procurar que el trabajo sea entregado con tiempo, anticipándonos a posibles cambios. 			
ESTADO ACTUAL: Fase de reducción iniciada: Fase de supervisión iniciada: Gestionando el riesgo:		<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	
RESPONSABLE: Edison Jara			

ANEXO 6 Especificación de Requerimientos

1.1.1.1. Requerimiento Funcional 2

Especificaciones

Introducción

El sistema podrá gestionar los datos de los alumnos.

Entrada

Fuentes de Entrada

Cedula

Nombres

Fecha de Nacimiento

Lugar de Nacimiento

Libreta Militar

Estado Civil

Nacionalidad

Ocupación

Ciudad

Cantón

Provincia

Sexo

Dirección

Teléfono

Celular

Correo

Teléfono del Pariente

Institución de donde Proviene

Nombre del Padre

Profesión del Padre

Dirección del Padre

Nombre de la Madre
Profesión de la Madre
Dirección de la Madre

Frecuencia

Bajo demanda

Requisitos de control

- Controla que los campos del formulario no estén vacíos.
- Controla que en el campo Cedula se ingresen solo diez números.
- Controla que en el campo Teléfono se ingresen solo diez números.
- Controla que en el campo Celular se ingresen solo diez números.
- En la actualización de los datos evita la modificación del campo cédula.

Entradas válidas

Todos los campos sean válidos

Procesos

Destino de las salidas

El formulario se cierra al guardar los datos y se muestra la lista de alumnos.

Interfaces de Hardware

- El monitor será el medio de visualización utilizado para mostrar cada uno de los procesos que se

efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará es JAVA, Netbeans 7.2, Apache Tomcat y Tapestry 5.3.7 para cumplir con el requerimiento.
- La Base de Datos está implementada en MYSQL.

1.1.3 Requerimiento Funcional 3

Especificaciones

Introducción

El sistema podrá gestionar los datos de los semestres.

Entrada

Frecuencia

Bajo demanda

Requisitos de control

- Controla que los campos del formulario no estén vacíos.
- En la actualización de los datos evita la modificación del campo id semestre.

Entradas válidas

Todos los campos sean válidos.

Procesos

1. La Secretaria selecciona el proceso a realizar
2. Si la Secretaria selecciona “Nuevo Semestre”

- 2.1. La Secretaria llena los datos del formulario
“Nuevo Semestre”
- 2.2. El sistema validará los datos.
- 2.3. Si los datos son correctos
 - 2.3.1. El sistema guarda los datos
 - Caso contrario
 - 2.3.2. Mensaje de error
3. Si la Secretaria selecciona “Actualizar Semestre”
 - 3.1. Secretaria selecciona el semestre a modificar.
 - 3.2. Llena los campos del formulario “Actualizar Semestre”
 - 3.3. El sistema validará los datos
 - 3.4. Si los datos son correctos
 - 3.4.1. El sistema guarda los datos
 - Caso Contrario
 - 3.4.2. Mensaje de Error
4. Si la Secretaria selecciona “Eliminar Semestre”
 - 4.1. Secretaria selecciona el semestre a eliminar.
 - 4.1.1. El sistema elimina los datos
 - Caso Contrario
 - 4.1.2. Mensaje de Error

Salida

Destino de las salidas

El formulario se cierra al guardar los datos y se muestra la lista de semestres.

Interfaces de Hardware

- El monitor será el medio de visualización utilizado para mostrar cada uno de los procesos que se

efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará es JAVA, Netbeans 7.2, Apache Tomcat y Tapestry 5.3.7 para cumplir con el requerimiento.
- La Base de Datos está implementada en MYSQL.

1.1.1.4 Requerimiento Funcional 4

Especificaciones

Introducción

Gestionar los datos de los estados civiles.

Entrada

Fuentes de Entrada

Id Estado Civil

Estado Civil

Frecuencia

Bajo demanda

Requisitos de control

- Controla que los campos del formulario no estén vacíos.

Entradas válidas

Todos los campos sean válidos

Procesos

1. La Secretaria selecciona el proceso a realizar
2. Si la Secretaria selecciona “Nuevo Estado Civil”
 - 2.1. La Secretaria llena los datos del formulario “Nuevo Estado Civil”
 - 2.2. El sistema validará los datos.
 - 2.3. Si los datos son correctos
 - 2.3.1. El sistema guarda los datos
Caso contrario
 - 2.3.2. Mensaje de error
3. Si la Secretaria selecciona “Actualizar Estado Civil”
 - 3.1. La Secretaria selecciona el estado civil a modificar
 - 3.2. Llena los campos del formulario “Actualizar Estado Civil”
 - 3.3. El sistema validará los datos.
 - 3.4. Si los datos son correctos
 - 3.4.1. El sistema guarda los datos
Caso contrario
 - 3.4.2. Mensaje de error
4. Si la Secretaria selecciona “Eliminar Estado Civil”
 - 4.1. La Secretaria selecciona el estado civil a eliminar
 - 4.1.1. El sistema elimina los datos

1.1.1.5 Requerimiento Funcional 5

Especificaciones

Introducción

Gestionar los pagos realizados por los alumnos.

Entrada

Fuentes de Entrada

Id de Alumno Pago

Alumno

Pago

Carrera

Semestre

Sección

Período

Frecuencia

Bajo demanda

Requisitos de control

- Controla que los campos del formulario no estén vacíos.

Entradas válidas

Todos los campos sean válidos

Procesos

1. La Secretaria selecciona el proceso a realizar

2. Si la Secretaria selecciona “Nuevo Pago de Alumnos”
 - 2.1.El usuario llena los datos del formulario “Nuevo Pago de Alumnos”
 - 2.2. El sistema validará los datos.
 - 2.3. Si los datos son correctos
 - 2.3.1. El sistema guarda los datos
Caso contrario
 - 2.3.2. Mensaje de error
3. Si la Secretaria selecciona “Actualizar Pago de Alumnos”
 - 3.1. La Secretaria selecciona el pago del alumno a modificar.
 - 3.2. Llena los campos del formulario “Actualizar Pago de Alumnos”
 - 3.3. El sistema validará los datos.
 - 3.4. Si los datos son correctos.
 - 3.4.1. El sistema guarda los datos
Caso contrario
 - 3.4.2. Mensaje de error
4. Si La Secretaria selecciona “Eliminar Pago de Alumnos”
 - 4.1. La Secretaria selecciona el Pago a ser eliminado
 - 4.1.1.El sistema elimina los datos
Caso Contrario
 - 4.1.2. Mensaje de Error

1.1.1.6 Requerimiento Funcional 6

Especificaciones

Introducción

Gestionar los datos de las provincias.

Entrada

Fuentes de Entrada

Id Provincia

Provincia

Frecuencia

Bajo demanda

Requisitos de control

- Controla que los campos del formulario no estén vacíos.

Entradas válidas

Todos los campos sean válidos

Procesos

1. Si la Secretaria selecciona “Nueva Provincia”
 - 1.1. La Secretaria llena los datos del formulario “Nueva Provincia”
 - 1.2. El sistema validará los datos.
 - 1.3. Si los datos son correctos
 - 1.3.1. El sistema guarda los datos
- Caso contrario

- 1.3.2. Mensaje de error
- 2. Si la Secretaria selecciona “Actualizar Provincia”
 - 2.1. La Secretaria selecciona la provincia a modificar
 - 2.2. Llena los campos del formulario “Actualizar Provincia”
 - 2.3. El sistema validará los datos.
 - 2.4. Si los datos son correctos
 - 2.4.1. El sistema guarda los datos
 - Caso contrario
 - 2.4.2. Mensaje de error
- 3. Si la Secretaria selecciona “Eliminar Provincia”
 - 3.1. La Secretaria selecciona la provincia a eliminar
 - 3.1.1. El sistema elimina los datos

1.1.1.7 Requerimiento Funcional 7

Especificaciones

Introducción

Gestionar los datos de los cantones.

Entrada

Fuentes de Entrada

Id Cantón

Id Provincia

Cantón

Frecuencia

Bajo demanda

Requisitos de control

- Controla que los campos del formulario no estén vacíos.

Entradas válidas

Todos los campos sean válidos

Procesos

1. La Secretaria selecciona el proceso a realizar
2. Si la Secretaria selecciona “Nuevo Cantón”
 - 2.1. La Secretaria llena los datos del formulario “Nuevo Cantón”
 - 2.2. El sistema validará los datos.
 - 2.3. Si los datos son correctos
 - 2.3.1. El sistema guarda los datos
Caso contrario
 - 2.3.2. Mensaje de error
3. Si la Secretaria selecciona “Actualizar Cantón”
 - 3.1. Secretaria selecciona el cantón a modificar
 - 3.2. Llena los campos del formulario “Actualizar Cantón”
 - 3.3. El sistema validará los datos.
 - 3.4. Si los datos son correctos
 - 3.4.1. El sistema guarda los datos
Caso contrario
 - 3.4.2. Mensaje de error
4. Si la Secretaria selecciona “Eliminar Cantón”
 - 4.1. La Secretaria selecciona el cantón a eliminar

Destino de las salidas

El formulario se cierra al guardar los datos y se

muestra la lista de los cantones.

Interfaces de Hardware

- El monitor será el medio de visualización utilizado para mostrar cada uno de los procesos que se efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará es JAVA, Netbeans 7.2, Apache Tomcat y Tapestry 5.3.7 para cumplir con el requerimiento.
- La Base de Datos está implementada en MYSQL.

1.1.1.8 Requerimiento Funcional 8

Especificaciones

Introducción

Gestionar los datos de las ciudades.

Entrada

Fuentes de Entrada

Id Ciudad

Id Canton

Ciudad

Frecuencia

Bajo demanda

Requisitos de control

- Controla que los campos del formulario no estén vacíos.

Entradas válidas

- Todos los campos sean válidos

Procesos

1. La Secretaria selecciona el proceso a realizar
2. Si la Secretaria selecciona “Nueva Ciudad”
 - 2.1. La Secretaria llena los datos del formulario “Nueva Ciudad”
 - 2.2. El sistema validará los datos.
 - 2.3. Si los datos son correctos
 - 2.3.1. El sistema guarda los datos
Caso contrario
 - 2.3.2. Mensaje de error
3. Si la Secretaria selecciona “Actualizar Ciudad”
 - 3.1. Secretaria selecciona la ciudad a modificar
 - 3.2. Llena los campos del formulario “Actualizar Ciudad”
 - 3.3. El sistema validará los datos.
 - 3.4. Si los datos son correctos
 - 3.4.1. El sistema guarda los datos
Caso contrario
 - 3.4.2. Mensaje de error
4. Si la Secretaria selecciona “Eliminar Ciudad”
 - 4.1. La Secretaria selecciona la ciudad a eliminar
 - 4.1.1. El sistema elimina los datos
Caso contrario
 - 4.1.2. Mensaje de error

Salidas

Destino de las salidas

El formulario se cierra al guardar los datos y se muestra la lista de las ciudades.

Interfaces de Hardware

- El monitor será el medio de visualización utilizado para mostrar cada uno de los procesos que se efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará es JAVA, Netbeans 7.2, Apache Tomcat y Tapestry 5.3.7 para cumplir con el requerimiento.
- La Base de Datos está implementada en MYSQL.

1.1.1.9 Requerimiento Funcional 9

Especificaciones

Introducción

Gestionar los datos de los usuarios.

Entrada

Fuentes de Entrada

Id Usuario

Cedula

Nombres

Dirección

Teléfono

Celular

Correo

Cuenta

Clave

Frecuencia

Bajo demanda

Requisitos de control

- Controla que los campos del formulario no estén vacíos.

Entradas válidas

- Todos los campos sean válidos

Procesos

1. La Secretaria selecciona el proceso a realizar
2. Si la Secretaria selecciona “Nuevo Usuario”
 - 2.1. La Secretaria llena los datos del formulario “Nuevo Usuario”
 - 2.2. El sistema validará los datos.
 - 2.3. Si los datos son correctos
 - 2.3.1. El sistema guarda los datos
 - Caso contrario
 - 2.3.2. Mensaje de error
3. Si la Secretaria selecciona “Actualizar Usuario”
 - 3.1. Secretaria selecciona el usuario a modificar
 - 3.2. Llena los campos del formulario “Actualizar Usuario”
 - 3.3. El sistema validará los datos.
 - 3.4. Si los datos son correctos
 - 3.4.1. El sistema guarda los datos

Caso contrario

3.4.2. Mensaje de error

4. Si la Secretaria selecciona “Eliminar Usuario”

4.1. La Secretaria selecciona el usuario a eliminar

4.1.1. El sistema elimina los datos

Caso contrario

4.1.2. Mensaje de error

Salidas

Destino de las salidas

El formulario se cierra al guardar los datos y se muestra la lista de las ciudades.

Interfaces de Hardware

- El monitor será el medio de visualización utilizado para mostrar cada uno de los procesos que se efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará es JAVA, Netbeans 7.2, Apache Tomcat y Tapestry 5.3.7 para cumplir con el requerimiento.
- La Base de Datos está implementada en MYSQL.

1.1.1.10 Requerimiento Funcional 10

Especificaciones

Introducción

Permitir la autenticación de usuario.

Entrada

Fuentes de Entrada

Cuenta de usuario
Contraseña

Frecuencia

Bajo demanda

Requisitos de control

- Controla que los campos del formulario no estén vacíos.

Entradas válidas

- Todos los campos sean válidos

Procesos

1. El Usuario ingresa la cuenta de usuario y contraseña.
2. El sistema verifica los datos ingresados.
3. Si los datos son correctos
 - 3.1. El usuario podrá manipular el sistema
 - Caso contrario
 - 3.2. Mensaje de error

Salidas

Destino de las salidas

El sistema sigue abierto mientras el usuario no presione el botón salir.

Interfaces de Hardware

- El monitor será el medio de visualización utilizado para mostrar cada uno de los procesos que se efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará es JAVA, Netbeans 7.2, Apache Tomcat y Tapestry 5.3.7 para cumplir con el requerimiento.
- La Base de Datos está implementada en MYSQL.

1.1.1.11 Requerimiento Funcional 11

Especificaciones

Introducción

Gestionar el Comprobante de Pago.

Entrada

Fuentes de Entrada

Pagos de Alumnos

Frecuencia

Bajo demanda

Requisitos de control

- Controla que los campos del formulario no estén vacíos.

Entradas válidas

- Todos los campos sean válidos

Procesos

1. La Secretaria al ingresar los pagos de los alumnos.
2. Selecciona el comprobante de pago.
3. Si los datos son correctos
 - 3.1. El sistema muestra el comprobante de pago.
Caso contrario
 - 3.2. Mensaje de error

Salidas

Destino de las salidas

El formulario sigue abierto mientras el usuario no presione el botón cancelar.

Interfaces de Hardware

- El monitor será el medio de visualización utilizado para mostrar cada uno de los procesos que se efectuarán.
- La impresora puede ser un medio para obtener un comprobante de pago tangible.

Interfaces de Software

- La herramienta de desarrollo que se utilizará es JAVA, Netbeans 7.2, Apache Tomcat y Tapestry 5.3.7 para cumplir con el requerimiento.
- La Base de Datos está implementada en MYSQL.

ANEXO 7 Diagramas de Secuencia

Diagrama de secuencia para ingresar carreras.

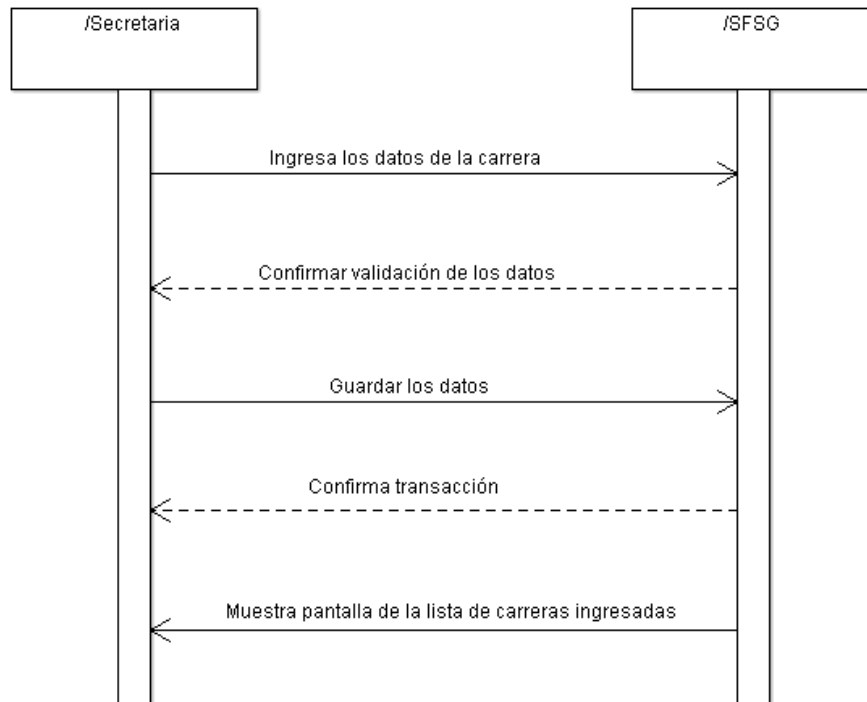


Diagrama de secuencia para listar las carreras.

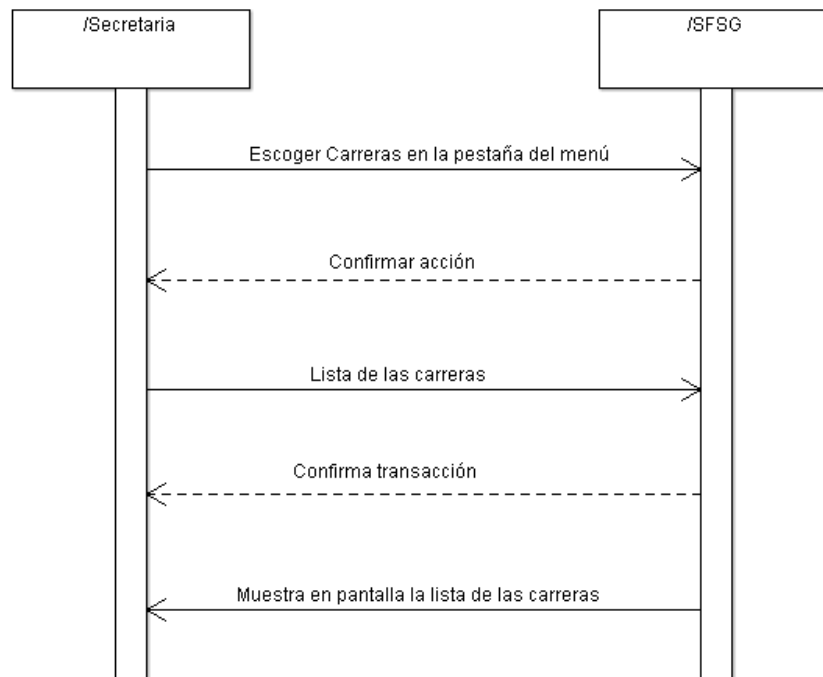


Diagrama de secuencia para modificar las carreras.

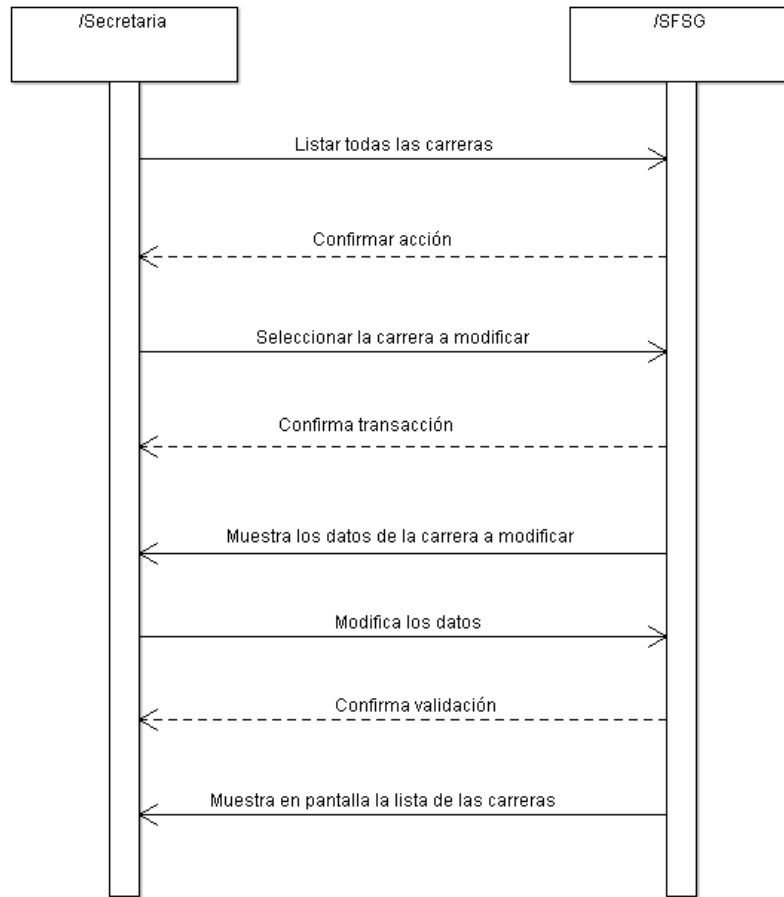


Diagrama de secuencia para eliminar las carreras.

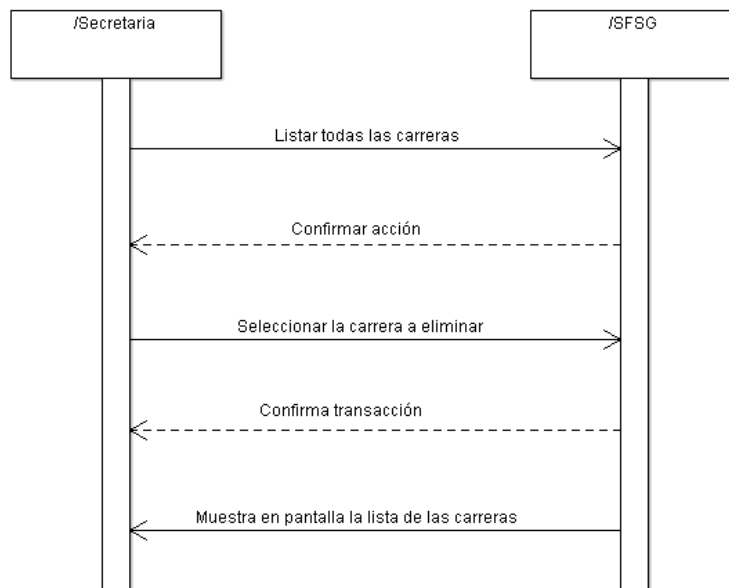
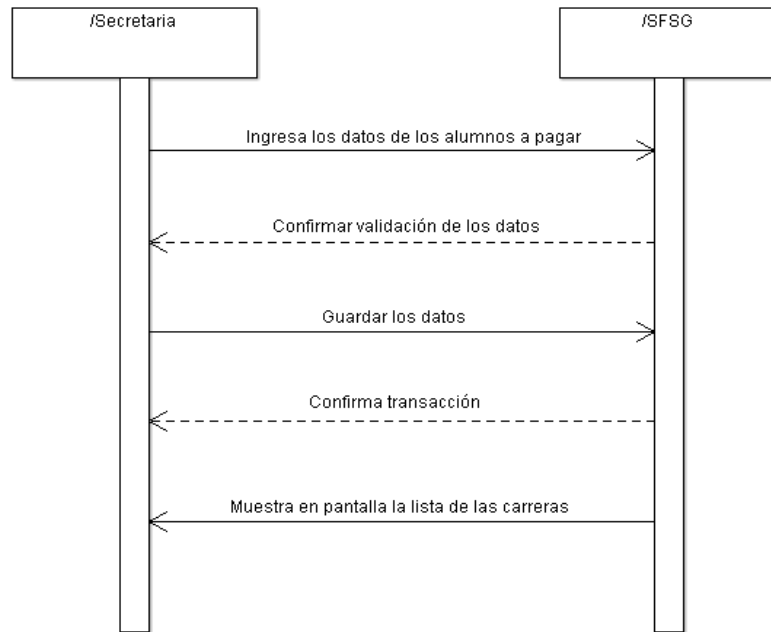


Diagrama de Secuencia para ingresar los pagos de los alumnos



ANEXO 8 Interfaces de Usuario

Pantalla de inicio una vez logueado en el sistema SFSG.



Pantalla para ingresar las carreras en el sistema SFSG.

Usuario: edison lunes 16 de junio de 2014 Hora: 22:31:48

INSTITUTO PARTICULAR SAN GABRIEL

Inicio Datos Alumnos Pagos de Alumnos Salir

Insertar Carreras

INFORMACION DE LA CARRERA

Carrera:


Derechos reservados © 2013 Instituto Tecnológico Particular "San Gabriel"
Dirección: Veloz 31-15 y Juan de Lavalle Telf: 03 (296 2347)
Riobamba-Ecuador

Pantalla para visualizar la lista de carreras en el sistema SFSG.

Usuario: edison lunes 16 de junio de 2014 Hora: 22:30:44

INSTITUTO PARTICULAR SAN GABRIEL

Inicio Datos Alumnos Pagos de Alumnos Salir

Lista de Carreras


Nuevo

1 2 3

N.-	Carrera	Actualizar	Eliminar
1	Tecnología en Sistemas	Actualizar	Eliminar
2	Contabilidad y Tributación	Actualizar	Eliminar


Derechos reservados © 2013 Instituto Tecnológico Particular "San Gabriel"
Dirección: Veloz 31-15 y Juan de Lavalle Telf: 03 (296 2347)
Riobamba-Ecuador

Pantalla para actualizar la información de las carreras.

lunes 16 de junio de 2014 Hora: 22:33:46

Usuario: edison

INSTITUTO PARTICULAR SAN GABRIEL

Inicio Datos Alumnos Pagos de Alumnos Salir

Actualizar Carreras

INFORMACION DE LA CARRERA	
N.-:	<input type="text" value="1"/>
Carrera:	<input type="text" value="Tecnología en Sistemas"/>
<input type="button" value="Actualizar"/> <input type="button" value="Cancelar"/>	



Derechos reservados © 2013 Instituto Tecnológico Particular "San Gabriel"
Dirección: Veloz 31-15 y Juan de Lavalle Telf: 03 (296 2347)
Riobamba-Ecuador

Pantalla para ingresar los pagos de los alumnos en el sistema SFSG.

lunes 16 de junio de 2014 Hora: 22:35:23

Usuario: edison

INSTITUTO PARTICULAR SAN GABRIEL

Inicio Datos Alumnos Pagos de Alumnos Salir

Insertar Pagos de los Alumnos

INFORMACION DEL PAGO	
Alumnos:	<input type="text" value="Jorge Marcelo Montesdeoca"/>
Pagos:	<input type="text" value="Inscripción"/>
Carreras:	<input type="text" value="Tecnología en Sistemas"/>
Semestres:	<input type="text" value="Primero"/>
Seccion:	<input type="text"/>
Periodo:	<input type="text"/>
<input type="button" value="Guardar"/> <input type="button" value="Cancelar"/>	

Pantalla de Mensajes de error desde el sistema SFSG

lunes 16 de junio de 2014 Hora: 22:35:23

Usuario: edison

INSTITUTO PARTICULAR SAN GABRIEL

Inicio Datos Alumnos Pagos de Alumnos Salir

Insertar Pagos de los Alumnos

INFORMACION DEL PAGO

Alumnos:	Jorge Marcelo Montesdeoca ▼
Pagos:	Inscripción ▼
Carreras:	Tecnología en Sistemas ▼
Semestres:	Tiene que ingresar un valor para Seccion. ✖
Seccion:	<input type="text"/> ✖
Periodo:	<input type="text"/> ✖

ANEXO 8 Diccionario de datos

TABLA: alumnos				
Nombre Campo	Tipo dato	longitud	Clave Primaria	Calculado
Idalumno	Number	11	x	No
Cedula	Varchar	10		No
Nombres	Varchar	100		No
Fnac	Date			No
Lnac	Varchar	60		No
Libretam	Varchar	12		No
Idestadocivil	Number	11		No
Nacionalidad	Varchar	60		No
Ocupacion	Varchar	60		No
Ciudad	Varchar	60		No
Cantón	Varchar	60		No
Provincia	Varchar	60		No

Sexo	Enum	M,F	No
Dirección	Varchar	100	No
Teléfono	Varchar	10	No
Cellular	Varchar	10	No
Correo	Varchar	60	No
Telpariente	Varchar	10	No
Instproviene	Varchar	60	No
Npadre	Varchar	60	No
Padre	Varchar	60	No
Dpadre	Varchar	60	No
Nmadre	Varchar	60	No
Pmadre	Varchar	60	No
Dmadre	Varchar	60	No

tabla: alumno_pago				
Nombre Campo	Tipo dato	longitud	Clave Primaria	Calculado
idalum_pag	Number	11	x	No
Idalumno	Number	11		No
Idpago	Number	11		No
Idcarrera	Number	11		No
Idsemestre	Number	11		No
Seccion	Varchar	60		No
Period	Varchar	60		No

TABLA: semestres				
Nombre Campo	Tipo dato	longitud	Clave Primaria	Calculado
Idsemestre	Number	11	x	No
Semester	Varchar	60		No

TABLA: provincias				
Nombre Campo	Tipo dato	longitud	Clave Primaria	Calculado
Idprovincia	Number	11	x	No
Provincia	Varchar	60		No

TABLA: cantones				
Nombre Campo	Tipo dato	longitud	Clave Primaria	Calculado
Idcanton	Number	11	x	No
Canton	Varchar	60		No

TABLA: ciudades				
Nombre Campo	Tipo dato	longitud	Clave Primaria	Calculado
Idciudad	Number	11	x	No
Ciudad	Varchar	60		No

TABLA: pagos				
Nombre Campo	Tipo dato	longitud	Clave Primaria	Calculado
Idpago	Number	11	x	No
Pago	Varchar	60		No
Precio	Double	5,2		No

TABLA: semestres				
Nombre Campo	Tipo dato	longitud	Clave Primaria	Calculado
Idsemestre	Number	11	x	No
Semester	Varchar	60		No

TABLA: requisitos				
Nombre Campo	Tipo dato	longitud	Clave Primaria	Calculado

Idrequisito	Number	11	x	No
Requisito	varchar	60		No

TABLA: estado_civil				
Nombre Campo	Tipo dato	longitud	Clave Primaria	Calculado
Idestadocivil	Number	11	x	No
Estadocivil	Varchar	60		No

TABLA: alumno_requerimiento				
Nombre Campo	Tipo dato	longitud	Clave Primaria	Calculado
idalum_req	Number	11	x	No
Idalumno	Number	11		No
Idrequerimiento	varchar	11		No
Observación	Text			No

TABLA: usuario				
Nombre Campo	Tipo dato	longitud	Clave Primaria	Calculado
Idusuario	Number	11	x	No
Cedula	varchar	10		No
Nombres	Varchar	100		No
Direccion	Varchar	100		No
Telefono	Varchar	10		No
Celular	Varchar	10		No
Correo	Varchar	60		No
Cuenta	Varchar	60		No
Clave	Varchar	60		No

ANEXO 9 Implementación de la Base de Datos en MySQL.

Esquemas o Base de Datos Existentes.

127.0.0.1/phpMyAdmin/

Servidor: localhost

Bases de datos

Base de datos	Cotejamiento
<input type="checkbox"/> control_personal	utf8_general_ci
<input type="checkbox"/> instituto_miguel	utf8_general_ci
<input type="checkbox"/> minimarket	utf8_general_ci
<input type="checkbox"/> mysql	utf8_general_ci
<input type="checkbox"/> phpmyadmin	utf8_bin
<input type="checkbox"/> san_gabriel	utf8_general_ci
<input type="checkbox"/> tapestry	utf8_general_ci
<input type="checkbox"/> test	utf8_general_ci
<input type="checkbox"/> trimat	utf8_general_ci
<input type="checkbox"/> trimat2	utf8_general_ci
Total: 10	utf8_general_ci

Marcar todos/as / Desmarcar todos Para los elementos que están marcados:

Activar las estadísticas

Nota: Activar aquí las estadísticas de la base de datos podría causar tráfico pesado entre el servidor web y el servidor MySQL.

Crear nueva base de datos

Cotejamiento Crear

Estructura de una Tabla

127.0.0.1/phpMyAdmin/

Servidor: localhost Base de datos: san_gabriel Tabla: carreras

Examinar Estructura SQL Buscar Insertar Exportar Importar Operaciones Vaciar Eliminar

Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
<input type="checkbox"/> idcarrera	int(11)			No		auto_increment	
<input type="checkbox"/> carrera	varchar(60)	utf8_general_ci		No			

Marcar todos/as / Desmarcar todos Para los elementos que están marcados:

Vista de impresión Planteamiento de la estructura de tabla

Añadir campo(s) Al final de la tabla Al comienzo de la tabla Después de idcarrera Continuar

Índices: 0					Espacio utilizado		Estadísticas de la fila	
Nombre de la clave	Tipo	Cardinalidad	Acción	Campo	Tipo	Uso	Enunciado	Valor
PRIMARY	PRIMARY	4		idcarrera	Datos	120 Bytes	Formato	dinámico/a
Crear un índice en <input type="text"/> columna(s) Continuar					Índice	2,048 Bytes	Cotejamiento	utf8_general_ci
					Total	2,168 Bytes	Filas	4
							Longitud de la fila 0	30
							Tamaño de la fila 0	542 Bytes
							Próxima Autoindex	5
							Creación	23-10-2013 a las 10:58:17
							Última actualización	23-10-2013 a las 11:01:47

Crear Tablas

The screenshot shows the phpMyAdmin interface for the 'san_gabriel' database. The left sidebar lists the database and its tables: alumnos, alumno_pago, carreras, estado_civil, pagos, requisitos, and semestres. The main area displays a table structure with columns: Tabla, Acción, Registros, Tipo, Cotejamiento, Tamaño, and Residuo a depurar. The tables listed are:

Tabla	Acción	Registros	Tipo	Cotejamiento	Tamaño	Residuo a depurar
alumnos	[Icons]	2	MyISAM	latin1_swedish_ci	3.4 KB	-
alumno_pago	[Icons]	0	MyISAM	utf8_general_ci	1.0 KB	-
carreras	[Icons]	4	MyISAM	utf8_general_ci	2.1 KB	-
estado_civil	[Icons]	0	MyISAM	utf8_general_ci	1.0 KB	-
pagos	[Icons]	2	MyISAM	utf8_general_ci	2.1 KB	-
requisitos	[Icons]	0	MyISAM	utf8_general_ci	1.0 KB	-
semestres	[Icons]	4	MyISAM	utf8_general_ci	2.1 KB	-

Summary: 7 tabla(s), 12 Registros, MyISAM Tipo, utf8_general_ci Cotejamiento, 12.6 KB Tamaño, 0 Bytes Residuo a depurar.

Crear Campos de una Tabla

The screenshot shows the 'usuario' table field creation screen in phpMyAdmin. The table structure is as follows:

Campo	Tipo	Longitud/Valores ¹	Cotejamiento	Atributos	Nulo	Predeterminado ²	Extra	Comentarios
	VARCHAR				not null			
	VARCHAR				not null			

Motor de almacenamiento: MyISAM, Cotejamiento: [Dropdown]

Grabar | O Añadir 1 campo(s) Continuar

¹ Si el tipo de campo es "enum" o "set", por favor ingrese los valores usando este formato: 'a','b','c'...
² Para valores predeterminados, por favor ingrese solamente un valor, sin caracteres de escape '\\' ni comillas, usando este formato: a

ANEXO 10 Scripts de los procedimientos utilizados en la Base de Datos

Procedimiento para insertar los pagos de los alumnos

```
CREATE PROCEDURE sp_insertar_alumnopago (IN nidalumno int, IN nidpago int, IN nidcarrera int, IN nidsemestre int, IN nseccion varchar(60), IN nperiodo varchar(60))  
BEGIN  
    INSERT INTO alumno_pago
```

```
(idalumno,idpago,idcarrera,idsemestre,seccion,periodo) VALUES
(nidalumno,nidpago,nidcarrera,nidsemestre,nseccion,nperiodo);
END$$
```

Procedimiento para actualizar los pagos de los alumnos

```
CREATE PROCEDURE sp_actualizar_alumnopago (IN nidalum_pag int, IN nidalumno
int,IN nidpago int,IN nidcarrera int,IN nidsemestre int,IN nseccion varchar(60),IN nperiodo
varchar(60))
BEGIN
    UPDATE alumno_pago SET idalumno=nidalumno,
                                idpago=nidpago,
                                idcarrera=nidcarrera,
                                idsemestre=nidsemestre,
                                seccion=ncseccion,
                                periodo=nperiodo
    WHERE idalum_pag=nidalum_pag;
END$$
```

ANEXO 11 Fase de Estabilización

FRAMEWORK WICKET

Parte 1: Código del prototipo 1 implementado con el framework WICKET.

WEB.XML

```
<?xml versión="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
version="2.5">
    <display-name>PrototipoWicketITSGA</display-name>
    <filter>
        <filter-name>wicket.PrototipoWicketITSGA</filter-name>
        <filter-class>org.apache.wicket.protocol.http.WicketFilter</filter-class>
        <init-param>
            <param-name>applicationClassName</param-name>
            <param-
value>com.mycompany.prototipowicketitsga.WicketApplication</param-value>
            </init-param>
        </filter>
        <filter-mapping>
            <filter-name>wicket.PrototipoWicketITSGA</filter-name>
            <url-pattern>/*</url-pattern>
        </filter-mapping>
</web-app>
```

POM.XML

```
<!--Dependencias utilizadas -->
<dependency>
```

```

    <groupId>org.apache.wicket</groupId>
    <artifactId>wicket</artifactId>
    <version>1.4.17</version>
    <type>jar</type>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.22</version>
</dependency>
<!--FIN Dependencias utilizadas -->

```

INSERTAR_ALUMNO_PAGO.HTML

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns:wicket="http://wicket.apache.org">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <title>Insertar_Alumno_Pago</title>
    <link href="style.css" rel="stylesheet" type="text/css" />
    <link href="Recursos/Css/estilos.css" rel="stylesheet" type="text/css"/>
  </head>
  <wicket:extend>
    <body bgcolor="#009999">
      <p>&nbsp;</p>
      <p>&nbsp;</p>
      <h2 align="center" class="style3"><strong>Registro de Pagos de los Alumnos</strong></h2>
    <div>
      <form wicket:id="alumno_pagoForm">
        <table border="1" width="884" cellspacing="1" cellpadding="1" align="center">
          <thead align="center">
            <tr align="center">
              <th colspan="4" align="center" class="style5"><h2>INFORMACIÓN DEL
PAGO</h2></th>
            </tr>
          </thead>
          <tbody>
            <tr>
              <td >
                <h3 align="center" wicket:id="lalumno" class="style5">texto</h3>
              </td>
              <td align="left">
                <select wicket:id="alumnosopciones">
                </select>
              </td>
            </tr>
            <tr>
              <td >
                <h3 align="center" wicket:id="lpago" class="style5">texto</h3>
              </td>
              <td align="left">
                <select wicket:id="pagosopciones">
                </select>
              </td>
            </tr>
          </tbody>
        </table>
      </form>
    </div>
  </wicket:extend>

```

```

        <td >
            <h3 align="center" wicket:id="lcarrera" class="style5">texto</h3>
        </td>
        <td align="left">
            <select wicket:id="carrerasopciones">
            </select>
        </td>
    </tr>
    <tr>
        <td >
            <h3 align="center" wicket:id="lsemestre" class="style5">texto</h3>
        </td>
        <td align="left">
            <select wicket:id="semestresopciones">
            </select>
        </td>
    </tr>
    <tr>
        <td >
            <h3 align="center" wicket:id="lseccion" class="style5">texto</h3>
        </td>
        <td align="left">
            <input type="text" size="12" wicket:id="tfseccion" maxlength="10"/>
        </td>
    </tr>
    <tr>
        <td>
        </td>
        <td>
            <td><h3 align="center" wicket:id="msjseccion" class="style5">texto </h3>
            </td>
        </td>
    </tr>
    <tr>
        <td >
            <h3 align="center" wicket:id="lperiodo" class="style5">texto</h3>
        </td>
        <td align="left" >
            <input type="text" size="12" wicket:id="tfperiodo" maxlength="10"/>
        </td>
    </tr>
    <tr>
        <td>
        </td>
        <td>
            <td><h3 align="center" wicket:id="msjperiodo" class="marcadolinks2">texto </h3>
            </td>
        </td>
    </tr>
    <tr>
        <td height="61">
        </td>
        <td align="center">
            <input type="submit" value="Registrar" />
        </td>
    </tr>
</tbody>
</table>
</form>
</div>
</body>
</wicket:extend>

```

</html>

INSERATAR_ALUMNO_PAGO.JAVA

```
package alumno_pago;
import com.mycompany.prototipowicketitsga.BasePage;
import java.util.ArrayList;
import java.util.List;
import org.apache.wicket.PageParameters;
import org.apache.wicket.markup.html.basic.Label;
import org.apache.wicket.markup.html.form.DropDownChoice;
import org.apache.wicket.markup.html.form.Form;
import org.apache.wicket.markup.html.form.TextField;
import org.apache.wicket.model.Model;
import org.apache.wicket.model.PropertyModel;
import persistencia.Alumno_Pago;
public final class Insertar_Alumno_Pago extends BasePage {
    Alumno_Pago conexion;
    TextField tfseccion;
    TextField tfperiodo;
    DropDownChoice<String> cadenaalumnos;
    DropDownChoice<String> cadenapagos;
    DropDownChoice<String> cadenacarreras;
    DropDownChoice<String> cadenasemestres;
    String alumnoseleccionado;
    String pagoseleccionado;
    String semestreseleccionado;
    String carreraseleccionado;
    List<String> listaalumnos;
    List<String> listapagos;
    List<String> listacarrera;
    List<String> listasemestre;
    Label lalumno;
    Label lpago;
    Label lcarrera;
    Label lsemestre;
    String nmensajeseccion="";
    String nmensajeperiodo="";
    String mensajeseccion="";
    String mensajeperiodo="";
    public Insertar_Alumno_Pago(final PageParameters params) {
        conexion = new Alumno_Pago();
        Object alumnos[][] = conexion.getAlumnosLista();
        listaalumnos = new ArrayList<String>();
        for (int i = 0; i < alumnos.length; i++) {
            listaalumnos.add("" + alumnos[i][1]);
        }
        conexion = new Alumno_Pago();
        Object pagos[][] = conexion.getPagosLista();
        listapagos = new ArrayList<String>();
        for (int i = 0; i < pagos.length; i++) {
            listapagos.add("" + pagos[i][1]);
        }
        conexion = new Alumno_Pago();
        Object carreras[][] = conexion.getCarreraLista();
        listacarrera = new ArrayList<String>();
        for (int i = 0; i < carreras.length; i++){
            listacarrera.add("" + carreras[i][1]);
        }
    }
}
```



```

}
conexion = new Alumno_Pago();
Object semestres[][] = conexion.getSemestreLista();
listasemestre = new ArrayList<String>();
for (int i = 0; i < semestres.length; i++) {
    listasemestre.add("" + semestres[i][1]);
}
alumnoseleccionado = alumnos[0][1] + "Seleccione uno";
pagoseleccionado = pagos[0][1] + "Seleccione uno";
carreraseleccionado = carreras[0][1] + "Seleccione uno";
semestreseleccionado = semestres[0][1] + "Seleccione uno";
cadenaalumnos = new DropDownChoice<String>(
    "alumnosopciones", new PropertyModel<String>(this, "alumnoseleccionado"), listaalumnos);
cadenapagos = new DropDownChoice<String>(
    "pagosopciones", new PropertyModel<String>(this, "pagoseleccionado"), listapagos);
cadenacarreras = new DropDownChoice<String>(
    "carrerasopciones", new PropertyModel<String>(this, "carreraseleccionado"), listacarrera);
cadenasemestres = new DropDownChoice<String>(
    "semestresopciones", new PropertyModel<String>(this, "semestreseleccionado"),
listasemestre);
nmensajeseccion="" +params.getString("mensajeseccion");
mensajeseccion="" +params.getString("tfseccion");
nmensajeperiodo="" +params.getString("mensajeperiodo");
mensajeperiodo="" +params.getString("tfperiodo");
if(mensajeperiodo.equals("null")){ mensajeperiodo="";}
if(mensajeseccion.equals("null")){ mensajeseccion="";}
if(nmensajeperiodo.equals("null")){ nmensajeperiodo="";}
if(nmensajeseccion.equals("null")){ nmensajeseccion="";}
@Override
protected void onSubmit() {
    //get the entered password and pass to next page
    PageParameters pageParameters = new PageParameters();
    pageParameters.add("alumnosopciones", alumnoseleccionado);
    pageParameters.add("pagosopciones", pagoseleccionado);
    pageParameters.add("carrerasopciones", carreraseleccionado);
    pageParameters.add("semestresopciones", semestreseleccionado);
    pageParameters.add("lalumno", lalumno.toString());
    pageParameters.add("lpago", lpago.toString());
    pageParameters.add("lcarrera", lcarrera.toString());
    pageParameters.add("lsemestre", lsemestre.toString());
    int pasa=1;
    //VALIDAR periodo
    if((tfperiodo.getValue()+"").equals("")){
        nmensajeperiodo="Ingrese un Per\u00edodo";
        pageParameters.add("mensajeperiodo", ""+nmensajeperiodo);
        pasa=0;
    }
    else {
        pageParameters.add("mensajeperiodo", "");
    }
    if((tfseccion.getValue()+"").equals("")){
        nmensajeseccion="Ingrese una Secci\u00f3n";
        pageParameters.add("mensajeseccion", ""+nmensajeseccion);
        pasa=0;
    }
    else {
        pageParameters.add("mensajeseccion", "");
    }
}
if(pasa==0) {
    setResponsePage(Insertar_Alumno_Pago.class, pageParameters);
}
else {

```

```

        conexion.InsertarAlumno_Pago("'" + ((Integer.parseInt(cadenaalumnos.getValue())) + 1),
            "'" + ((Integer.parseInt(cadenapagos.getValue())) + 1), "'" +
((Integer.parseInt(cadenacarreras.getValue())) + 1),
            "'" + ((Integer.parseInt(cadenasemestres.getValue())) + 1), tfseccion.getValue(),
tfperiodo.getValue());
        setResponsePage(Insertar_Alumno_Pago.class);
    }    }    };
add(form);
form.add(tfperiodo);
form.add(tfseccion);
form.add(msjperiodo);
form.add(msjseccion);
form.add(lperiodo);
form.add(lseccion);
form.add(lalumno);
form.add(lcarrera);
form.add(lsemestre);
form.add(lpago);
form.add(cadenacarreras);
form.add(cadenapagos);
form.add(cadenasemestres);
form.add(cadenaalumnos);
}
}

```

Parte 2: Imágenes del prototipo 1 implementado con el framework WICKET.

localhost:8084/PrototipoWicketTSGA/?wicket:bookmarkablePage=:alumno_pago.Insertar_Alumno_Pago

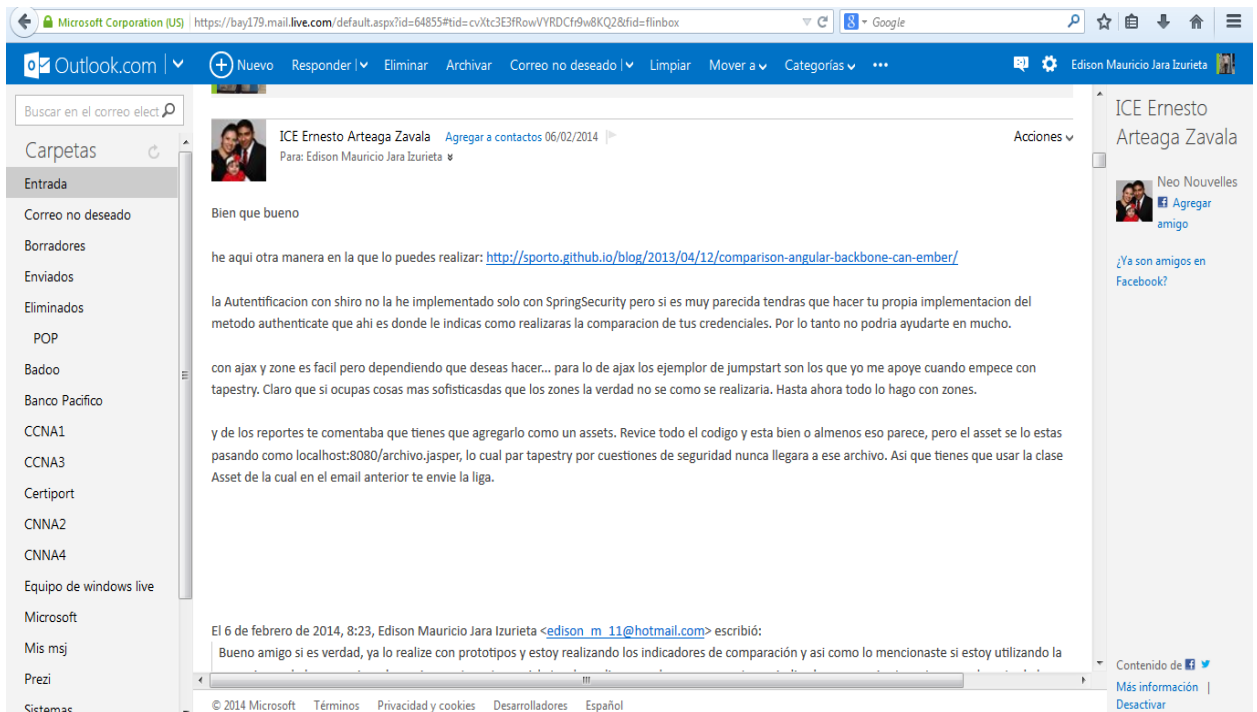
INSTITUTO PARTICULAR SAN GABRIEL

Inicio
Datos
Alumnos
Pagos de Alumnos
Acerca De
Contacto
Salir

Registro de Pagos de los Alumnos

INFORMACIÓN DEL PAGO	
Alumno:	<input type="text" value="Seleccione uno"/>
Pago:	<input type="text" value="Seleccione uno"/>
Carrera:	<input type="text" value="Seleccione uno"/>
Semestre:	<input type="text" value="Seleccione uno"/>
Sección:	<input type="text"/>
Periodo:	<input type="text"/>
<input type="button" value="Registrar"/>	

ANEXO 12 Información obtenida con la experiencia de Ernesto Arteaga Zavala Ingeniero de Desarrollo.



Microsoft Corporation (US) | <https://bay179.mail.live.com/default.aspx?id=64855#tid=cvXtc3E3fRowVVRDCf9w8KQ2&fid=flinbox> | Google

Outlook.com | Nuevo | Responder | Eliminar | Archivar | Correo no deseado | Limpiar | Mover a | Categorías | Edison Mauricio Jara Izurieta

Buscar en el correo electo

Carpetas

- Entrada
- Correo no deseado
- Borradores
- Enviados
- Eliminados
- POP
- Badoo
- Banco Pacífico
- CCNA1
- CCNA3
- Certiport
- CNNA2
- CNNA4
- Equipo de windows live
- Microsoft
- Mis msj
- Prezi
- Sistemas

ICE Ernesto Arteaga Zavala | Agregar a contactos 06/02/2014 | Acciones

Para: Edison Mauricio Jara Izurieta

Bien que bueno

he aqui otra manera en la que lo puedes realizar: <http://sporto.github.io/blog/2013/04/12/comparison-angular-backbone-can-ember/>

la Autenticacion con shiro no la he implementado solo con SpringSecurity pero si es muy parecida tendras que hacer tu propia implementacion del metodo authenticate que ahi es donde le indicas como realizaras la comparacion de tus credenciales. Por lo tanto no podria ayudarte en mucho.

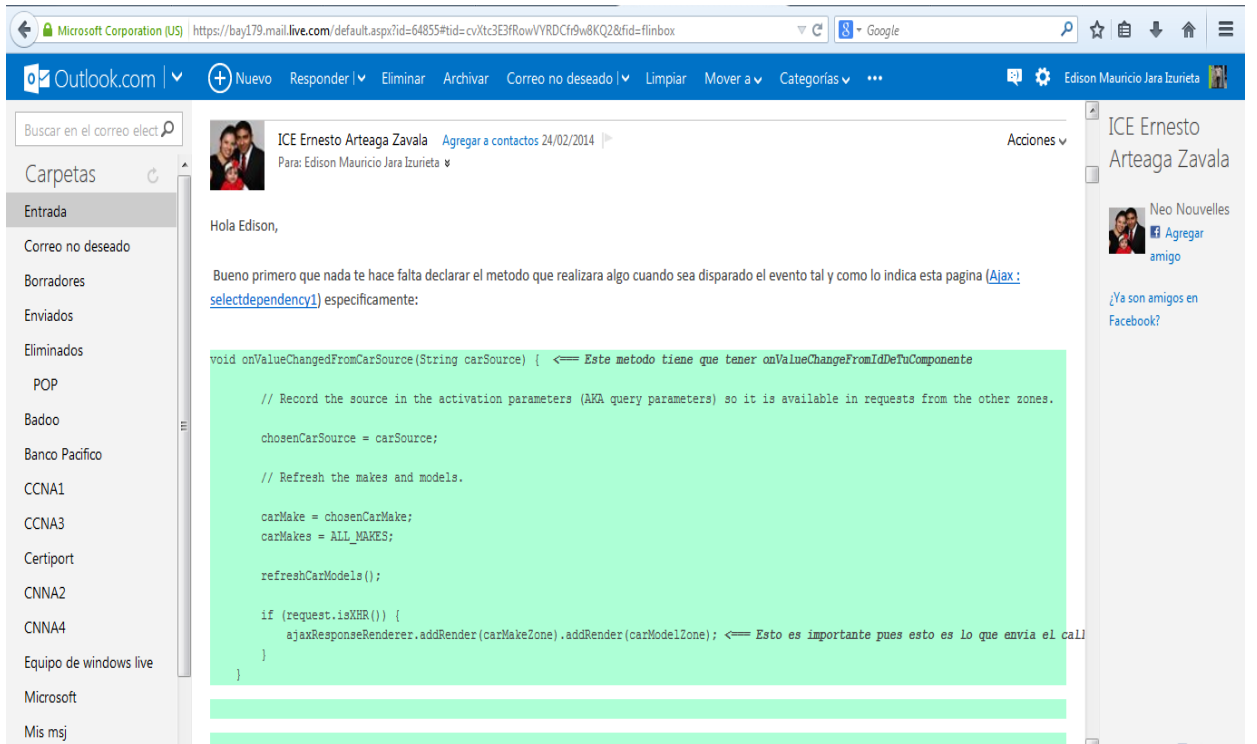
con ajax y zone es facil pero dependiendo que desees hacer... para lo de ajax los ejemplar de jumpstart son los que yo me apoye cuando empee con tapestry. Claro que si ocupas cosas mas sofisticadas que los zones la verdad no se como se realizaria. Hasta ahora todo lo hago con zones.

y de los reportes te comentaba que tienes que agregarlo como un assets. Revise todo el codigo y esta bien o almenos eso parece, pero el asset se lo estas pasando como localhost:8080/archivo.jasper, lo cual par tapestry por cuestiones de seguridad nunca llegara a ese archivo. Asi que tienes que usar la clase Asset de la cual en el email anterior te envia la liga.

El 6 de febrero de 2014, 8:23, Edison Mauricio Jara Izurieta <edison_m_11@hotmail.com> escribió:
Bueno amigo si es verdad, ya lo realice con prototipos y estoy realizando los indicadores de comparación y asi como lo mencionaste si estoy utilizando la

© 2014 Microsoft | Términos | Privacidad y cookies | Desarrolladores | Español

Contenido de Facebook | Más información | Desactivar



Microsoft Corporation (US) | <https://bay179.mail.live.com/default.aspx?id=64855#tid=cvXtc3E3fRowVVRDCf9w8KQ2&fid=flinbox> | Google

Outlook.com | Nuevo | Responder | Eliminar | Archivar | Correo no deseado | Limpiar | Mover a | Categorías | Edison Mauricio Jara Izurieta

Buscar en el correo electo

Carpetas

- Entrada
- Correo no deseado
- Borradores
- Enviados
- Eliminados
- POP
- Badoo
- Banco Pacífico
- CCNA1
- CCNA3
- Certiport
- CNNA2
- CNNA4
- Equipo de windows live
- Microsoft
- Mis msj

ICE Ernesto Arteaga Zavala | Agregar a contactos 24/02/2014 | Acciones

Para: Edison Mauricio Jara Izurieta

Hola Edison,

Bueno primero que nada te hace falta declarar el metodo que realizara algo cuando sea disparado el evento tal y como lo indica esta pagina ([Ajax: selectdependency1](#)) especificamente:

```
void onValueChangedFromCarSource(String carSource) { <== Este metodo tiene que tener onValueChangeFromIdDeTuComponente

    // Record the source in the activation parameters (AKA query parameters) so it is available in requests from the other zones.
    chosenCarSource = carSource;

    // Refresh the makes and models.
    carMake = chosenCarMake;
    carMakes = ALL_MAKES;

    refreshCarModels();

    if (request.isXHR()) {
        ajaxResponseRenderer.addRender(carMakeZone).addRender(carModelZone); <== Esto es importante pues esto es lo que envia el call
    }
}
```

Microsoft Corporation (US) | <https://bay179.mail.live.com/default.aspx?id=64855#tid=cvXtc3E3fRowVYRDcf9w8KQ2&fid=flinbox> | Google

Outlook.com | Nuevo | Responder | Eliminar | Archivar | Correo no deseado | Limpiar | Mover a | Categorías | Edison Mauricio Jara Izurieta

Buscar en el correo elect. | Carpetas | Entrada | Correo no deseado | Borradores | Enviados | Eliminados | POP | Badoo | Banco Pacifico | CCNA1 | CCNA3 | Certiport | CNNA2 | CNNA4 | Equipo de windows live | Microsoft | Mis msj

ICE Ernesto Arteaga Zavala | Agregar a contactos 02/03/2014 | Para: Edison Mauricio Jara Izurieta

Verifica que las propiedades Llista y providencia sean aparte de @Property @Persist

El 1 de marzo de 2014, 19:37, Edison Mauricio Jara Izurieta <edison_m_11@hotmail.com> escribió:
Amigo si tengo asi esta parte de las zonas pero no me sale y con esto me sale un error no se en que parte esta mal...espero que me ayudes por favor

```
@Property
private SelectModel provinciasModel;

@InjectComponent
private Zone provinciaZone;

@InjectComponent
private Zone provinciaModelZone;

void onValueChangedFromProvincia(Provincias provincia) {
    canton = null;
    if (provincia != null) {
```

Microsoft Corporation (US) | <https://bay179.mail.live.com/default.aspx?id=64855#tid=cvXtc3E3fRowVYRDcf9w8KQ2&fid=flinbox> | Google

Outlook.com | Nuevo | Responder | Eliminar | Archivar | Correo no deseado | Limpiar | Mover a | Categorías | Edison Mauricio Jara Izurieta

Buscar en el correo elect. | Carpetas | Entrada | Correo no deseado | Borradores | Enviados | Eliminados | POP | Badoo | Banco Pacifico | CCNA1 | CCNA3 | Certiport | CNNA2 | CNNA4 | Equipo de windows live | Microsoft | Mis msj | Prezi | Sistemas

ahí el código o la aplicación subirá a gut así como lo estás haciendo para que me ayudes....

ICE Ernesto Arteaga Zavala | Agregar a contactos 05/02/2014 | Para: Edison Mauricio Jara Izurieta

Que tanto inglés sabes existe esta página no se si ya la habrás revisado: [jumpstart](#)

Tienen demos de todo lo que puedas llevar a cabo en Tapestry y la verdad es que es muy fácil de aprender Tapestry pero tenemos en la cabeza que queremos hacer todo y para eso es el framework, creo que esa debería de ser una parte de tu conclusión, así como las trabas que tuviste al realizar tu ejercicio.

Por otro lado veo que en los assets (donde está el nombre del reporte) tienes un localhost:8080, y el nombre del reporte como alumnos.jasper trata de completar la url con la dirección correcta del asset los assets en tapestry se manejan de diferente manera se encuentra en la documentación ([Assets](#)).

Y para el envío de datos de una página a otra están:
[Pagelink](#)
[Passing Data Between Pages](#)

Y bueno si tienes la investigación ya realizada, la convención de nombres entendida y el uso de componentes analizado, no te debería de ser difícil. A comparación de JSP, tapestry es mucho más fácil codificarlo. Ese es el punto de un framework.

Ahora desde el punto de vista de investigación creo que estás en la ruta incorrecta pues para realizar un análisis correcto deberías de hacer un benchmark de desarrollo y no una aplicación para saber cuál es mejor, no puedes realizar una conclusión basándote en la documentación y solo desarrollando en uno solo.

© 2014 Microsoft | Términos | Privacidad y cookies | Desarrolladores | Español | Contenido de | Más información | Desactivar

Para obtener más información también puede acceder a los siguientes link:

Ejemplos: <https://github.com/Mauricio89/tapestry>

Foro: http://www.javamexico.org/blogs/arterzati/jasper_report_tapestry

Correo electrónico de Ernesto Arteaga Zavala: arterzati@gmail.com

FRAMEWORK TAPESTRY

Parte 1: Código del prototipo 1 implementado con el framework TAPESTRY.

WEB.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
         version="3.0">
  <display-name>PrototipoTapestryITSGA Tapestry 5 Application</display-name>
  <context-param>
    <!-- The only significant configuration for Tapestry 5, this informs Tapestry
of where to look for pages, components and mixins. -->
    <param-name>tapestry.app-package</param-name>
    <param-value>com.mycompany.prototipotapestryitsga</param-value>
  </context-param>
  <context-param>
    <param-name>tapestry.development-modules</param-name>
    <param-value>
      com.mycompany.prototipotapestryitsga.services.DevelopmentModule
    </param-value>
  </context-param>
  <context-param>
    <param-name>tapestry.qa-modules</param-name>
    <param-value>
      com.mycompany.prototipotapestryitsga.services.QaModule
    </param-value>
  </context-param>
  <filter>
    <filter-name>app</filter-name>
    <filter-class>org.apache.tapestry5.TapestryFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>app</filter-name>
    <url-pattern>*</url-pattern>
  </filter-mapping>
</web-app>
```

POM.XML

```
<!--Dependencias utilizadas -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.22</version>
</dependency>
<!--FIN Dependencias utilizadas -->
```

INSERTARALUMNO_PAGO.TML

```
<html t:type="Plantilla" titulo="ITS San Gabriel" t:sidebarTitle="Framework Version"
      xmlns:t="http://tapestry.apache.org/schema/tapestry_5_3.xsd"
      xmlns:p="tapestry:parameter">
```

```

<div align="center">
  <h20 align="center" class="style3"><strong>Insertar Pagos de los Alumnos</strong></h20>
  <br/>
  <br/>
  <t:form>
    <t:errors/>
    <table border="1" width="700" cellspacing="1" cellpadding="1" align="center">
      <thead align="center">
        <tr align="center">
          <th colspan="2" align="center" class="teeee"><h2>INFORMACION DE LA
CARRERA</h2></th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>
            <span>Alumnos:</span>
          </td>
          <td >
            <select t:type="select" t:model="alumnosModel" t:value="alumno"
t:encoder="alumnoEncoder"></select>
          </td>
        </tr>
        <tr>
          <td>
            <span>Pagos:</span>
          </td>
          <td >
            <select t:type="select" t:model="pagosModel" t:value="pago"
t:encoder="pagoEncoder"></select>
          </td>
        </tr>
        <tr>
          <td>
            <span>Carreras:</span>
          </td>
          <td >
            <select t:type="select" t:model="carrerasModel" t:value="carrera"
t:encoder="carreraEncoder"></select>
          </td>
        </tr>
        <tr>
          <td>
            <span>Semestres:</span>
          </td>
          <td >
            <select t:type="select" t:model="semestresModel" t:value="semestre"
t:encoder="semestreEncoder"></select>
          </td>
        </tr>
        <tr>
          <td>
            <span>Seccion:</span>
          </td>
          <td >
            <p><t:textfield t:id="seccion" value="alumno_pago.seccion" t:validate="required"
size="60" maxlength="60"/></p>
          </td>
        </tr>
      </tbody>
    </table>
  </t:form>

```

```

        </tr>
        <tr>
            <td>
                <span>Periodo:</span>
            </td>
            <td >
                <p><t:textfield t:id="periodo" value="alumno_pago.periodo" t:validate="required"
size="60" maxlength="60"/></p>
            </td>
        </tr>
        <tr>
            <td align="center" colspan="2">
                <!--input type="submit" value="Crear"/-->
                <p><t:submit value="Ingresar"/></p>
            </td>
        </tr>
    </tbody>
</table>
</t:form>
</div>
</html>

```

INSERTARALUMNO_PAGO.JAVA

```

package com.mycompany.prototipotapestryitsga.pages;
import com.mycompany.prototipo_tapestry.encoder.AlumnoEncoder;
import com.mycompany.prototipo_tapestry.encoder.CarreraEncoder;
import com.mycompany.prototipo_tapestry.encoder.PagoEncoder;
import com.mycompany.prototipo_tapestry.encoder.SemestreEncoder;
import com.mycompany.prototipotapestryitsga.services.AlumnoPagoDAO;
import com.mycompany.prototipotapestryitsga.services.AlumnosDAO;
import com.mycompany.prototipotapestryitsga.services.CarreraDAO;
import com.mycompany.prototipotapestryitsga.services.PagosDAO;
import com.mycompany.prototipotapestryitsga.services.SemestreDAO;
import entidades.Alumno_Pago;
import entidades.Alumnos;
import entidades.Carrera;
import entidades.Pagos;
import entidades.Semestre;
import java.util.List;
import org.apache.tapestry5.EventContext;
import org.apache.tapestry5.SelectModel;
import org.apache.tapestry5.annotations.Property;
import org.apache.tapestry5.ioc.annotations.Inject;
import org.apache.tapestry5.services.SelectModelFactory;

public class InsertarAlumno_Pago {
    private Integer idalumno;
    private Integer idcarrera;
    private Integer idpago;
    private Integer idsemestre;
    //Alumno_Pago
    @Inject
    private AlumnoPagoDAO dao;
    @Property
    private Alumno_Pago alumno_pago;
    //Alumnos
    @Inject

```

```

private AlumnosDAO daoalumno;
@property
private Alumnos alumno;
//Carreras
@Inject
private CarreraDAO daocarrera;
@property
private Carrera carrera;
//Pagos
@Inject
private PagosDAO daopago;
@property
private Pagos pago;
//Semestres
@Inject
private SemestreDAO daosemestre;
@property
private Semestre semestre;
//Modelos
@property
private SelectModel alumnosModel;
@property
private SelectModel carrerasModel;
@property
private SelectModel pagosModel;
@property
private SelectModel semestresModel;
@Inject
private SelectModelFactory selectModelFactory;
void setupRender() {
    alumno_pago = new Alumno_Pago();
}
void onPrepareForSubmit() {
    alumno_pago = new Alumno_Pago();
}
void onActivate(EventContext context) {
    //Alumnos
    if (context.getCount() > 0) {
        idalumno = context.get(Integer.class, 0);
    }
    //Carreras
    if (context.getCount() > 0) {
        idcarrera = context.get(Integer.class, 0);
    }
    //Pagos
    if (context.getCount() > 0) {
        idpago = context.get(Integer.class, 0);
    }
    //Semestres
    if (context.getCount() > 0) {
        idsemestre = context.get(Integer.class, 0);
    }
}
void onPrepareForRender() {
    //Obtiene toda la lista de Alumnos
    List<Alumnos> alumnos = daoalumno.listarTodos();
    if (idalumno != null) {
        alumno = encontrarAlumnoLista(idalumno, alumnos);    }
}

```



```

alumnosModel = selectModelFactory.create(alumnos, "nombres");
//Obtiene toda la lista de Carreras
List<Carrera> carreras = daocarrera.listarTodos();
if (idcarrera != null) {
    carrera = encontrarCarreraLista(idcarrera, carreras);
}
carrerasModel = selectModelFactory.create(carreras, "ncarrera");
//Obtiene toda la lista de Pagos
List<Pagos> pagos = daopago.listarTodos();
if (idpago != null) {
    pago = encontrarPagosLista(idpago, pagos);
}
pagosModel = selectModelFactory.create(pagos, "npago");
//Obtiene toda la lista de Semestres
List<Semestre> semestres = daosemestre.listarTodos();
if (idsemestre != null) {
    semestre = encontrarSemestresLista(idsemestre, semestres);
}
semestresModel = selectModelFactory.create(semestres, "nsemestre");
}
void onValidateFromForm() {
    idalumno = alumno == null ? null : alumno.getIdalumno();
    idcarrera = carrera == null ? null : carrera.getIdcarrera();
    idpago = pago == null ? null : pago.getIdpago();
    idsemestre = semestre == null ? null : semestre.getIdsemestre();
}
//Alumnos
private Alumnos encontrarAlumnoLista(Integer idalumno, List<Alumnos> alumnos) {
    for (Alumnos alumno : alumnos) {
        if (alumno.getIdalumno().equals(idalumno)) {
            return alumno;
        }
    }
    return null;
}
//Carreras
private Carrera encontrarCarreraLista(Integer idcarrera, List<Carrera> carreras) {
    for (Carrera carrera : carreras) {
        if (carrera.getIdcarrera().equals(idcarrera)) {
            return carrera;
        }
    }
    return null;
}
//Pagos
private Pagos encontrarPagosLista(Integer idpago, List<Pagos> pagos) {
    for (Pagos pago : pagos) {
        if (pago.getIdpago().equals(idpago)) {
            return pago;
        }
    }
    return null;
}
//Semestres
private Semestre encontrarSemestresLista(Integer idsemestre, List<Semestre> semestres) {
    for (Semestre semestre : semestres) {
        if (semestre.getIdsemestre().equals(idsemestre)) {
            return semestre;
        }
    }
}

```

```

    }
  }
  return null;
}
//Alumnos
public AlumnoEncoder getAlumnoEncoder() {
  return new AlumnoEncoder(daoalumno);
}
//Carrera
public CarreraEncoder getCarreraEncoder() {
  return new CarreraEncoder(daocarrera);
}
//Pagos
public PagoEncoder getPagoEncoder() {
  return new PagoEncoder(daopago);
}
//Semestres
public SemestreEncoder getSemestreEncoder() {
  return new SemestreEncoder(daosemestre);
}
void onSuccess() {
dao.insertarAlumno_Pago(alumno.getIdalumno(), pago.getIdpago(), carrera.getIdcarrera(), semestre.getIds
emestre(), alumno_pago.getSeccion(), alumno_pago.getPeriodo());
}
}
}

```

Parte 2: Imágenes del prototipo 2 implementado con el framework TAPESTRY.

localhost:8084/PrototipoTapestryITSGA/insertaralumno_pago

INSTITUTO PARTICULAR SAN GABRIEL

Inicio Datos Alumnos Pagos de Alumnos Acerca De Contacto Salir

Insertar Pagos de los Alumnos

INFORMACION DEL PAGO	
Alumnos:	<input type="text"/>
Pagos:	<input type="text"/>
Carreras:	<input type="text"/>
Semestres:	<input type="text"/>
Seccion:	<input type="text"/>
Periodo:	<input type="text"/>
<input type="button" value="Ingresar"/>	

GLOSARIO

SFSG: (Sistema Financiero San Gabriel) Nombre del sistema implementado en el Instituto Particular San Gabriel para el registro de pagos realizados por los alumnos.

APLICACIÓN WEB: Una aplicación web es cualquier aplicación que es accedida vía web por una red como internet o una intranet.

BASE DE DATOS: Es una colección de información organizada de forma que un programa de ordenador pueda seleccionar rápidamente los fragmentos de datos que necesite.

CONTRASEÑA: Es una serie secreta de caracteres que permite a un usuario tener acceso a un archivo, a un ordenador, o a un programa. En sistemas multiusuarios, cada usuario debe incorporar su contraseña antes de que el ordenador responda a los comandos.

JDK: JDK es el acrónimo de "Java Development Kit", es decir Kit de desarrollo de Java. Se puede definir como un conjunto de herramientas, utilidades, documentación y ejemplos para desarrollar aplicaciones Java.

TAPESTRY: Es un framework JAVA que facilita el desarrollo de las aplicaciones web. TAPESTRY se encarga de todos los aspectos comunes y aburridos de las aplicaciones web, dejando que el programador se dedique a aportar valor desarrollando las características únicas de cada proyecto.

NETBEANS: Netbeans es una plataforma de desarrollo de software multilenguaje que comprende de un IDE y un sistema de plugins para extenderla. Está escrita principalmente en Java y es utilizada para desarrollar aplicaciones en este lenguaje y, por medio de los varios plugins, soporta el desarrollo en otros lenguajes tales como C, C++, Cobol, Python, Perl, PHP y más.

HIPÓTESIS: Algo que se supone y que está por ser comprobado.

JAVA: Java es un lenguaje Orientado a Objetos, por tal razón soporta el concepto de

herencia no permite la herencia múltiple, es decir solamente podemos heredar de una sola clase más no de otras clases.

XML: (Extensible Markup Language) Facilita la transferencia de datos a través de diferentes plataformas, especialmente las conectadas a Internet. Todo esto se da ya que XML es un lenguaje que es ampliamente utilizado para definición de valores, estructuras y propiedades en general.

JAR: (Archivo Java) Es un formato de archivo que se utiliza para agrupar varios archivos dentro de un único archivo de almacenamiento. Normalmente un archivo JAR contiene los archivos de clase y recursos auxiliares asociados a applets y aplicaciones.

JAVA PLUGIN: Forma parte de Java Runtime Environment y establece la conexión entre los exploradores más conocidos y la plataforma Java. Esta conexión permite que los applets de sitios Web se ejecuten en el explorador en el escritorio.

JVM: (Java Virtual Machine) Es un conjunto de programas de software que permiten la ejecución de instrucciones y que normalmente están escritos en código byte de Java. Las máquinas virtuales de Java están disponibles para las plataformas de hardware y software de uso más frecuente.

SISTEMA: Grupo de componentes interrelacionados que trabajan en conjunto hacia una meta común mediante la aceptación de entradas y generando salidas en un proceso de transformación organizado en el que interactúan elementos como equipo computacional, software, recurso humano, datos e información.

INFORMACIÓN: Conjunto de datos procesados que sirven de insumo para el control y la toma de decisiones.

SISTEMA DE INFORMACIÓN: Conjunto de componentes interrelacionados que permiten capturar, almacenar, procesar y distribuir la información para apoyar la toma de decisiones, el control, la coordinación, el análisis y la visión en una organización.

RECURSOS HUMANOS: Las personas se requieren para la operación de todos los Sistemas de Información.

RECURSOS DEL HARDWARE: Incluyen todos los dispositivos físicos y materiales utilizados en el proceso de la información.

RECURSOS DE SOFTWARE: Este concepto incluye toda la serie de instrucciones de procesamiento e información; incluye programas y las series de instrucciones de procesamiento de información que necesitan las personas, llamadas procedimientos.

INDICADOR: Es una unidad sencilla de medición que permite a través del tiempo conocer el comportamiento de una variable dentro de un proceso. Es una unidad de información, que muestra las características de un proceso susceptible de adoptar distintos valores en el tiempo. Permite conocer y controlar de manera objetiva el desarrollo y los resultados de un proceso.

FRAMEWORK: En el desarrollo de software, un framework o marco de trabajo, un framework es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. Típico, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

POLÍGLOTA: soportando cualquier lenguaje ejecutable en la JVM.

BIBLIOGRAFÍA

1. **DASHORST M. Y HILLENUS E.,** Wicket in Action. United States of America, Manning, 2009. p. (139 – 158).
2. **KOLESNIKOV, A.,** Tapestry 5: A Step-By-Step Guide to Java Web Development with the Developer-Friendly Apache Tapestry Framework. Greenock - Scotland, Packt, 2008. 280 p.
3. **ONSTINE, W.,** Tapestry 101. Las Vegas, Nevada - United States of America, SourceBeat, 2011. p. (79 – 86)
4. **CRUZ, A.H.,** Análisis Comparativo de Frameworks para el desarrollo de aplicaciones web en Java. Facultad de Informática, Escuela de Licenciatura en Informática. Ixtlán de Juárez, Oaxaca - México. Universidad de la Sierra Juárez. **TESIS**, 2013, p.5
E-Book
www.unsij.edu.mx/tesis/digitales/tesis%20Angel%20Hernandez%20Cruz.pdf
5. **LARA, L.P Y LÓPEZ, V.A.,** Análisis de las herramientas JDE y WEBWORK SDK para el desarrollo de aplicaciones móviles

nativas en la plataforma blackberry. Aplicado para los centros Infantiles MIES Chimborazo. Facultad de Informática y Electrónica, Escuela de Ingeniería en Sistemas. Riobamba – Ecuador. Escuela Superior Politécnica de Chimborazo. **TESIS**, 2012, p.96.

E-Book

dspace.esPOCH.edu.ec/bitstream/123456789/2550/1/18T00526.pdf

6. APACHE TAPESTRY

<http://tapestry.apache.org/>

2013 - 05 – 22

7. APACHE WICKET

<http://wicket.apache.org/>

2013 - 05 – 15

8. CARACTERÍSTICAS DE WICKET

<http://wicket.apache.org/meet/features.html>

2013 - 05 – 20

9. COMPONENTE GRID DEL FRAMEWORK TAPESTRY

<http://tapestry.apache.org/current/apidocs/org/apache/tapestry/5/corelib/components/Grid.html>

2012 - 03 – 14

10. COMPONENTE DATEFIELD DE TAPESTRY

<http://tapestry.apache.org/current/apidocs/org/apache/tapestry/5/corelib/components/DateField.html>

2013 - 08 – 14

11. CONFIGURACIÓN DE MAVEN

<http://maven.apache.org/download.cgi>

2013 - 06 – 15

12. CONFIGURACIÓN DE NETBEANS PARA LAS APLICACIONES EN TAPESTRY

<http://diegosegalla.wordpress.com/tag/tapestry/>

2013 - 08 – 20

13. CREACIÓN DE LA ESTRUCTURA DEL ORIGEN DE UNA APLICACIÓN WICKET

<https://netbeans.org/kb/docs/web/quickstart-webapps-wicket.html#create-4>

2013 - 07 – 22

14. DESVENTAJAS DE TAPESTRY

http://wikis.uca.es/wikiIW/index.php/Apache_Tapestry

2013 - 06 – 22

15. INTRODUCCIÓN A TAPESTRY

<http://tapestry.apache.org/introduction.html>

2013 - 07 – 10

16. INTRODUCCIÓN A WICKET

<http://wicket.apache.org/meet/introduction.html>

2013 - 07 – 20

17. JAVA EE capa de presentación

http://es.wikipedia.org/wiki/Java_EE

2013 - 07 – 10

18. OBJETIVOS DE WICKET

<http://wicket.apache.org/meet/introduction.html>

2013 - 05 – 24

19. PÁGINAS EN TAPESTRY

<http://es.scribd.com/doc/98793422/Tapestry-5>

2013 - 08 – 24

20. PATRONES DE DISEÑO

<http://thelozu.blogspot.com/2013/07/modelo-vista-controlador.html>

2013 - 08 – 24

21. PLUGIN TAPESTRY: DESARROLLO DE APLICACIONES Y PÁGINAS WEB CON APACHE TAPESTRY

PICODOTDEV., PlugIn Tapestry: Desarrollo de Aplicaciones y Páginas Web con Apache Tapestry. México, 2013. 250p.

<http://picodotdev.github.io/blog-bitix/assets/custom/PlugInTapestry.pdf>

2013 - 11 – 10

22. ¿QUE ES NETBEANS?

https://netbeans.org/index_es.html

2013 - 07 – 15

23. ¿QUE ES UNA APLICACIÓN WEB?

http://www.ecured.cu/index.php/Aplicaci%C3%B3n_web

2013 - 06 – 15

<http://www.internetya.co/5-razones-para-tener-una-aplicacion-web-propia-dentro-de-su-compania/>

2013 - 06 - 17

24. VERSIONES DE APACHE TAPESTRY

<http://tapestry.apache.org/release-notes.html>

2013 - 09 - 05

**25. VIDEOS DEMOSTRATIVOS SOBRE TAPESTRY
START**

<http://flywind.org/tapestry>

2013 - 09 - 18

26. WICKET USER GUIDE

<https://ia801701.us.archive.org/14/items/WicketFreeGuide/Wicket%20free%20guide.pdf>

2013 - 05 - 22