

Aplicación del Proceso Unificado de Desarrollo de SOFTWARE mediante el uso de la tecnología Java

Índice

1.1.	Introducción	4
1.2.	Objetivos	5
1.2.1.	Objetivo General	5
1.2.2.	Objetivos Específicos.....	5
1.2.3.	Justificación De La Investigación	5
1.2.4.	Alcance	5
1.2.5.	Aspectos Metodológicos:.....	6
2.	Capítulo 2: Proceso Unificado	7
2.1.	Introducción	7
2.2.	Los Tres Principios Básicos del Proceso Unificado.....	8
2.2.1.	Dirigido por casos de uso.....	8
2.2.2.	Centrado en la arquitectura	8
2.2.3.	Iterativo e incremental	9
2.3.	Mejores prácticas o consideraciones.....	10
2.3.1.	El desarrollo iterativo.-	11
2.3.2.	Administración de requerimientos.-.....	11
2.3.3.	Uso de la arquitectura basada en.....	11
	componentes	11
2.4.	Conceptos claves del RUP	13
2.4.1.	Artefactos	14
2.4.2.	Roles o trabajadores	14
2.4.3.	Actividades	14
2.4.4.	Flujos de Trabajo de procesos.....	14
2.5.	Estructura del Proceso unificado de desarrollo.....	14
2.6.	Flujos de Trabajo de procesos.....	15
2.6.1.	Modelamiento del negocio.....	15
2.6.2.	Requerimientos	16
2.6.3.	Análisis y diseño	16
2.6.4.	Implementación.....	16
2.6.5.	Pruebas	17
2.6.6.	Despliegue.....	17
2.6.7.	Gestión del cambio y configuraciones	17
2.6.8.	Gestión de proyecto	17
2.6.9.	Entorno.....	17
2.6.10.	Las Disciplinas Producen Modelos.....	18
2.7.	Fases.....	18
2.7.1.	En la fase de inicio	19
2.7.2.	En la fase de elaboración	20
2.7.3.	En la fase de construcción.....	20
2.7.4.	En la fase de transición	20
2.8.	Lenguaje Unificado de Modelado, UML.....	22

2.8.1.	Introducción	22
2.8.2.	Diagrama de actividades	22
2.8.3.	Diagrama de Casos de uso	23
2.8.4.	Diagrama de clases	24
2.8.5.	Diagrama de componentes	25
2.8.6.	Diagrama de despliegue	25
2.8.7.	Diagrama de secuencia	26
2.8.8.	Diagrama de Colaboración	26
2.8.9.	Diagrama de estados	27
3.	Capítulo: Plataforma Java	28
3.1.	Introducción	28
3.2.	Introducción a J2EE	29
3.3.	Plataforma de J2EE	30
3.3.1.	Ventajas de J2EE	31
3.3.2.	J2EE Apis.....	32
3.3.3.	Los contenedores	32
3.4.	Tecnología de J2EE	34
3.4.1.	Applets	35
3.4.2.	Java Servlets.....	35
3.4.3.	Java Server Pages.....	36
3.4.4.	Enterprise JavaBeans	37
3.5.	Aplicaciones Distribuidas en Múltiples Capas	39
4.	Capítulo 4: Tecnología.....	45
4.1.	Hibernate	45
4.1.1.	Introducción	45
4.1.2.	Qué es hibernate.....	46
4.1.3.	La Arquitectura	47
4.1.4.	Mapas de objetos relacionales en.....	48
	ficheros XML.....	48
4.2.	Framework Struts	48
4.2.1.	Introducción	48
4.2.2.	Definición de las partes MVC.....	49
4.2.3.	Como funciona?	50
4.2.4.	Tiles en Struts.	52
4.3.	Herramientas Empleadas en el desarrollo.....	52
4.3.1.	Servidor Web Apache	52
4.3.2.	Servidor de Aplicaciones Jakarta Tomcat.....	52
4.3.3.	MySQL	53
4.3.4.	Jrun Studio	54
4.3.5.	La herramienta de construcción Ant	55
4.3.6.	La herramienta Xdoclet.....	55
4.3.7.	Eclipse.....	55
4.3.8.	Rational Rose	56
4.3.9.	Herramientas de Diseño Gráfico.....	56
5.	Capítulo 5: Desarrollo de la Aplicación.....	58
5.1.	Especificación De Requerimientos Del Software.....	58
5.1.1.	Objetivo del documento.....	58
5.1.2.	Alcance del sistema.....	58
5.1.3.	Propósito del sistema.	59

5.1.4.	Descripción del sistema.	59
5.1.5.	Definiciones y abreviaturas.....	60
5.1.6.	Visión general	60
5.1.7.	Descripción General.....	60
	Descripción de las Clases de los formularios	63
5.2.	Consideraciones Tecnológicas.....	64
5.2.1.	Arquitectura	64
5.2.2.	Fácil escalabilidad.....	66
5.2.3.	Plataforma	66
5.2.4.	Tecnologías	66
5.2.5.	Aplicaciones y Herramientas Complementarias	68
5.2.6.	Entorno de Ejecución y Especificaciones	69
5.2.7.	Personal.....	70
5.2.8.	Costo de Desarrollo.....	70
5.2.9.	Software	72
5.2.10.	Hardware.....	73
5.2.11.	Características de los usuario del sistema.....	74
5.3.	Análisis y Diseño	74
5.3.1.	Diagrama De Clases.....	75
5.3.2.	Modelo Entidad Relación	77
5.3.3.	Modelo Físico de Datos	79
5.3.4.	Diagrama De Casos de Uso	81
5.3.5.	Diagramas De Secuencia	83
5.3.6.	Diagrama De Actividades	90
5.3.7.	Diagrama De Estados.....	92
5.3.8.	Diagrama De Componentes	94
5.3.9.	Diagrama De Despliegue	96
5.4.	Implementación.....	98
5.5.	Pruebas	102
6.	Capítulo 6: Conclusiones y Recomendaciones	107
6.1.	Conclusiones	107
6.2.	Recomendaciones	108
	Anexos	110
	Anexo A Manual de Usuario	111
	Anexo B Manual de Instalación.....	120
	Anexo C Diccionario de Datos	133
	Anexo D Glosario Términos	144

Capítulo 1 : **Introducción**

1.1. Introducción

El objetivo principal en el desarrollo de software es construir sistemas de alta calidad a un menor costo con un tiempo de producción mínimo. Para poder alcanzar el fin propuesto es necesario seguir una metodología de desarrollo, la cual nos permita organizar mediante etapas y técnicas de análisis un proceso de desarrollo; el cual al ser aplicado correctamente nos permita obtener un software de alta calidad a un menor costo.

La aplicación de una metodología en el desarrollo de sistemas no es un concepto nuevo en lo absoluto, ya que existían con anterioridad distintas metodologías y procesos a seguir en el desarrollo sistemas.

Según mi criterio, actualmente el proceso unificado de desarrollo de software de la casa Rational es el más completo por el trabajo en conjunto que se empleó para realizar el mismo. Se reunieron grandes autores de metodologías como Ivar Jacobson, Grady Booch y James Rumbaugh entre otros.

Para la construcción de software de alta calidad es necesario además del uso correcto de una metodología de desarrollo y el empleo de una notación como el Lenguaje Unificado de Modelado y por último la correcta selección de la tecnología más adecuada, para brindar la mayor confiabilidad.

Por lo que la tecnología java a trascendido a través de los años en el mercado con éxito, desde que fue creada su primera versión; destacándose por su confiabilidad y robustez al momento de desarrollar aplicaciones modulares y escalables.

1.2. Objetivos

1.2.1. Objetivo General

Demostrar como la tecnología de última generación (UML, Proceso Unificado de Desarrollo de Software, plataforma J2EE) deben ser empleadas en el análisis, diseño, e implementación de un sistema de software.

1.2.2. Objetivos Específicos

- Investigar los conceptos y fundamentos de: UML, el Proceso Unificado de Software, plataforma J2EE
- Identificar los beneficios de aplicar esta tecnología en el desarrollo de sistemas de software.
- Desarrollar una aplicación de ejemplo de Shopping Cart.

1.2.3. Justificación De La Investigación

Detallar la forma como deben emplearse las herramientas de última generación (RUP,UML, Java 2 Enterprise Edition), para el análisis, diseño e implementación de un sistema de software, aplicándolas en el sistema ejemplo, el mismo que proporcionará una nueva alternativa para realizar ventas en línea de productos.

Lo complejo que es el desarrollo de software y como la tecnología estudiada aporta para minimizar esta complejidad.

1.2.4. Alcance

El proyecto está centrado en la investigación de cómo se debe aplicar tecnología de última generación cómo es el Proceso Unificado de Desarrollo, El Lenguaje Unificado de Modelado y la

Plataforma J2EE para el desarrollo de una aplicación práctica como es el caso de un ShoppingCart (Carrito de Compras).

El sistema de shoppingCart es un sistema de e-commerce que actualmente cuenta con una demanda creciente en el mercado, el sistema permitirá:

- Ingresar inventario
- Buscar productos
- Manipular el carrito de compras
- Crear órdenes de compras
- Cambiar el estado de las órdenes
- Administrar órdenes

El objetivo primordial del desarrollo del sistema, es demostrar el empleo de la tecnología; razón por la cual, no se implementará el proceso pago de la orden.

1.2.5. Aspectos Metodológicos:

Para el desarrollo del trabajo se realizará las siguientes actividades:

- Investigación del Proceso Unificado de Desarrollo de software.
- Investigación de UML.
- Estudiar la plataforma J2EE.
- Estudiar lenguaje JSP .
- Estudiar el Motor de Persistencia Hibernate.
- Estudiar el Framework Struts.
- Desarrollo de la programación del sistema de ejemplo Shopping Cart

2. Capítulo 2: Proceso Unificado

2.1. Introducción

El proceso unificado ¹de desarrollo de software, es un marco de trabajo genérico el cual puede aplicarse a una gran variedad de sistemas de software sin importar su tamaño o área de operación. El proceso unificado se basa en componentes interconectados a través de interfaces bien definidas y utiliza el Lenguaje Unificado de Modelado (Unified Modeling Language). Los aspectos que realmente lo definen son:

- Dirigido por casos de Uso.
- Centrado en la Arquitectura.
- Iterativo e incremental.

Además el proceso Unificado toma en cuenta ciertas consideraciones o sus mejores prácticas para el desarrollo las cuales son:

- Desarrollo iterativo
- Manejo de requerimientos
- Uso de la arquitectura basada en componentes
- Empleo del Modelado Visual
- Verificación continua de la calidad
- Gestión del cambio y configuraciones

Las mejores prácticas del uso del proceso unificado, nos llevan a concluir exitosamente el proceso de desarrollo de software.

¹ El proceso unificado de desarrollo de Software, Ivar Jacobson, Grady Booch, James Rumbaugh, Addison Wesley

2.2. Los Tres Principios Básicos del Proceso Unificado

2.2.1. Dirigido por casos de uso

Los Casos de Uso son la forma más efectiva y simple, para modelar y abstraer los requisitos de cualquier sistema desde la perspectiva del usuario. Además de especificar los requisitos del sistema, colaboran en las etapas de diseño, implementación y prueba del sistema, de tal manera que se puede considerar que los casos de uso guían y controlan el proceso de desarrollo en su totalidad.

El modelo de casos de uso está compuesto por el diagrama de casos de uso y la descripción de casos de uso. En el diagrama figuran actores y casos de uso: Los actores, representan usuarios, departamentos u otros sistemas que interactúan con el sistema, estos son representados con figuras de personas. Los casos de uso, representan el comportamiento del sistema; comportamiento que puede tomar el sistema según el estímulo de los actores.

2.2.2. Centrado en la arquitectura

La arquitectura se utiliza para conceptualizar, construir, gestionar y evolucionar el sistema en desarrollo. La arquitectura del software, es una vista del diseño completo con las características más importantes, ésta incluye los aspectos tanto dinámicos como estáticos del sistema.

Un arquitecto de software debe centrar sus actividades en:
Crear un esquema en borrador de la arquitectura (plataforma)
Trabajar con un subconjunto de los casos de uso especificados, los cuales representan las funciones claves del sistema. Se especifica y realiza en términos de subsistemas, clases y componentes.

2.2.3. Iterativo e incremental

La complejidad de los sistemas actuales dificultan la abstracción del problema y a medida que se va desarrollando se van encontrando más dificultades por lo cual producen pérdidas de recursos y demora en el tiempo de desarrollo. La ventaja de un proceso iterativo es que permite tener una comprensión gradual del problema mediante mejoras sucesivas y el crecimiento incremental de una solución global.

El desarrollo de software se divide en iteraciones o mini-proyectos que constituyen un incremento. Las iteraciones se refieren a los pasos en el flujo de trabajo y los incrementos al crecimiento del producto. Mediante el uso de un proceso iterativo controlado, se reduce el riesgo a los costos de un solo incremento, es decir, en cada iteración se va corrigiendo y mejorando mediante la implementación de una nueva versión del sistema; de esta manera se reducen los riesgos más significativos para el éxito del proyecto.

Cada etapa itera sobre 5 flujos de trabajo:

- Requerimientos: Averiguar lo que el sistema debe hacer.
- Análisis y Diseño: Conseguir una comprensión más precisa de los requisitos no funcionales y adaptación de los requisitos funcionales para su implementación.
- Implementación: Implementación de clases y pruebas de componentes individuales
- Pruebas: Diseñar y realizar las pruebas de

integración y de sistemas.

Planificación y Evaluación : Planificar y evaluar los resultados de los avances del proyecto.

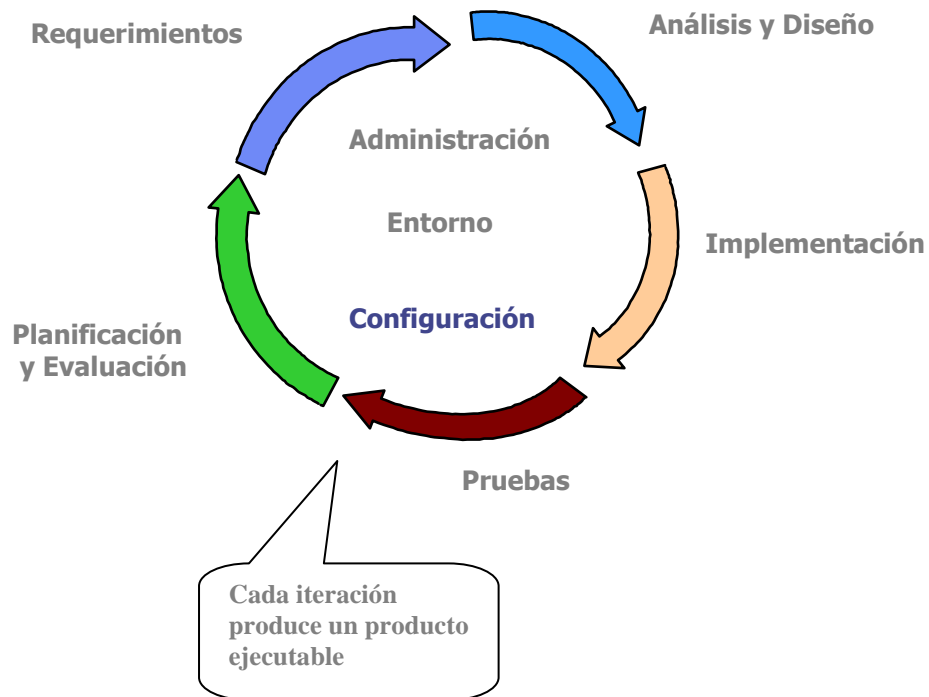


Figura 1.1. Proceso iterativo e incremental

Fuente: The Rational Unified Process an Introduction Second Edition, Philippe Kruchten , Capítulo 1

2.3. Mejores prácticas o consideraciones

Un proceso de Ingeniería de Software requiere herramientas que apoyen todas las actividades del ciclo de vida de los sistemas. Por lo cual las mejores prácticas del proceso Unificado son técnicas probadas por organizaciones exitosas en el campo de desarrollo. La solución de RATIONAL comprende los Seis Principios de Desarrollo de Software:

1. Administración de los requerimientos

2. Desarrollar en forma iterativa
3. Modelar visualmente
4. Verificación continua de la calidad
5. Uso de arquitecturas basadas en componentes
6. Administración de cambios

para los cuales se cuenta con la amplia gama de herramientas para el apoyo de los procesos.

2.3.1. El desarrollo iterativo.-

Esta es la primera mejor práctica², la cual también es un principio básico del proceso unificado el cual fue explicado anteriormente en el punto 1.2.3

2.3.2. Administración de requerimientos.-

Consiste en abstraer y documentar todos los requerimientos principales del sistema, cambiándolos y depurándolos durante el desarrollo del mismo midiendo el impacto que éstos tengan con el sistema.

Para llevar a cabo esta tarea se recomienda que, mediante el apoyo de los casos de uso se identifiquen los requerimientos del negocio y se lleve un registro y documentación de cambios y decisiones.

2.3.3. Uso de la arquitectura basada en componentes

² The Rational Unified Process an Introduction Second Edition, Philippe Kruchten , Capítulo 1

Se pueden encontrar bastantes ventajas al hacer uso de una arquitectura basada en componentes, como: la reutilización del software, facilidad de realizar cambios y reducción en la complejidad del problema.

Esta última, a mi criterio es una de las más importantes puesto que permite llegar a realizar aplicaciones más complejas, es decir, cuando trabajamos con componentes se reduce el grado de complejidad del problema, ya que se va desarrollando en partes el problema, de tal manera que el desarrollador se centra poco a poco en resolver partes del problema y no en el problema general.

2.3.4. Empleo del modelado visual

Es el empleo de varios modelos visuales, los cuales van a facilitar el entendimiento del sistema en desarrollo y la comunicación entre los miembros del grupo de desarrollo. El modelado visual, captura la estructura y el comportamiento de la arquitectura y los componentes del sistema; de esta forma muestra como se entrelazan todos los elementos del sistema. Además, ayuda a los desarrolladores a mantener consistencia entre el diseño y la implementación.

2.3.5. Verificación continua de la calidad

Consiste en la evaluación continua de la calidad del sistema, con respecto a los requisitos funcionales y no funcionales mediante la realización de pruebas en cada iteración. Se crean pruebas mediante los casos de uso, es decir, se crean pruebas para cada escenario (caso de uso); de esta forma se asegura que todos los requerimientos estén propiamente implantados.

Si, además se utiliza la práctica de desarrollar software iterativamente, se está verificando en cada iteración, logrando un proceso de evaluación continuo y cuantitativo.

La evaluación del estado del proyecto es objetiva, y no subjetiva. Se evalúan resultados de la prueba. Estas evaluaciones objetivas exponen inconsistencias en requerimientos, diseños e implementaciones. La evaluación y la verificación se enfocan en las áreas de riesgo más alto, incrementando la calidad y efectividad. Los defectos se identifican en forma temprana, reduciendo el costo de repararlos después de la puesta en producción.

2.3.6. Administración de cambios

La administración de cambios consiste en controlar y establecer una política de cambios, la cual pueda referirse a: Requerimientos, tecnología, recursos y plataformas. Esta administración brinda una manera eficiente para realizar cambios entre diferentes grupos de trabajo, versiones, plataformas entre otros.

La administración del cambio, establece espacios de trabajo seguros para cada desarrollador, es decir, aísla los cambios hechos en otros espacios de trabajo y controla todos los artefactos, modelos, códigos, documentos, entre otros.

2.4. Conceptos claves del RUP

Los conceptos claves del proceso unificado son los que nos ayudan a resolver preguntas como;

Quien hace qué? como? y cuando? Mediante los roles o trabajadores (Quien?), Artefactos(que?) , actividades (como?) y las fases, iteraciones y detalles de flujos de trabajo (cuando)

2.4.1. Artefactos

Son cualquier tipo de información creada, producida, cambiada o utilizada por los trabajadores o roles en el desarrollo del sistema como por ejemplo: Diagramas UML, bocetos de la interfaz de usuario y los prototipos, componentes, planes de prueba y procedimientos de prueba.

2.4.2. Roles o trabajadores

A partir del proceso unificado de 2001 se cambió el nombre de trabajadores a roles.

Estos roles son los papeles que cumplen o puede desempeñar un conjunto de personas que trabajan juntas o a su vez una persona que pueda desempeñar muchos roles ya sea por experiencia personal o formación.

2.4.3. Actividades

Son las responsabilidades que van a realizar los roles o trabajadores, en donde cada actividad está asignada a un rol. Estas se pueden aplicar varias veces sobre el mismo artefacto. Las actividades normalmente crean o actualizan artefactos como por ejemplo un modelo, una clase o un plan.

2.4.4. Flujos de Trabajo de procesos

El proceso unificado consta de nueve disciplinas , conformadas por un conjunto de actividades agrupadas en áreas de interés común como: Modelamiento del negocio, Requerimientos, Análisis y diseño, Implementación, Prueba, Despliegue, Configuración y administración del cambio, Administración de proyecto y Ambiente.

2.5. Estructura del Proceso unificado de desarrollo.

El proceso unificado tiene una estructura bi-dimensional que comprende del tiempo y el contenido. Donde el eje horizontal representa el tiempo mostrando el ciclo de vida del proceso descrita en fases e iteraciones y el vertical el contenido el cual muestra las disciplinas o los flujos de trabajo fundamentales.

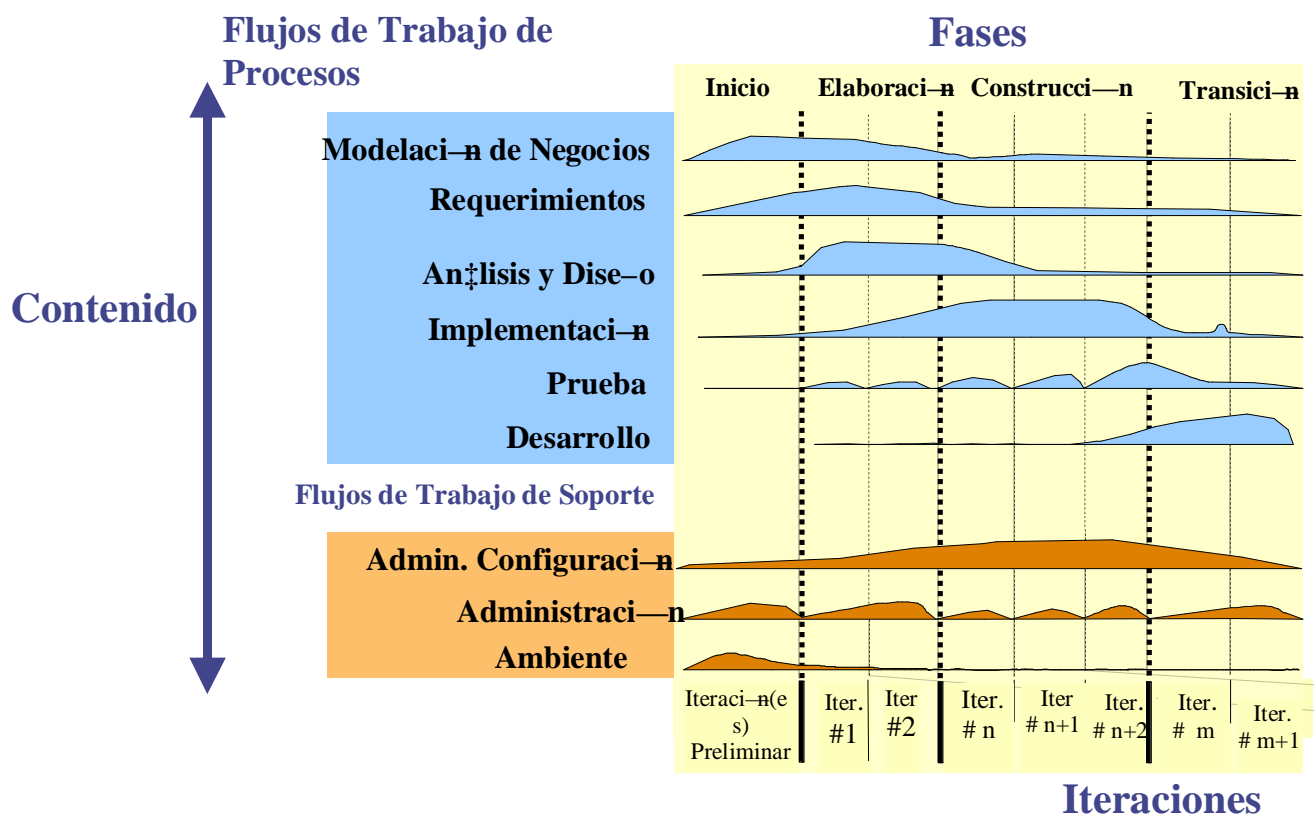


Figura 1.2. Estructura del Proceso

Fuente: The Rational Unified Process an Introduction Second Edition, Philippe Kruchten , Capítulo 1

Como se puede ver en la figura 1.2 muestra como se dedica más tiempo y se hace más énfasis en cada disciplina dependiendo de la fase en la que se encuentre: construcción o transición.

2.6. Flujos de Trabajo de procesos

2.6.1. Modelamiento del negocio

Esta disciplina se centra en el entendimiento de la organización, donde el sistema va a trabajar identificando los problemas actuales de la organización para poder plantear las posibles mejoras. De esta forma, aseguramos un entendimiento sobre la organización entre los usuarios finales, clientes y desarrolladores.

Además esta disciplina nos permite extraer los requerimientos necesarios para sistema de la organización objetivo.

Mediante un modelo de negocio podemos definir una nueva visión de la organización en donde podemos identificar procesos, roles y responsabilidades de la organización.

2.6.2. Requerimientos

Definen y establecen el alcance del sistema con los clientes y las personas relacionadas con el proyecto, de esta manera los desarrolladores van a tener un mejor entendimiento de los requerimientos del sistema.

La disciplina de requerimientos define una base para planificar el contenido técnico de las iteraciones y para estimar el costo y tiempo de desarrollo.

2.6.3. Análisis y diseño

Esta disciplina transforma los requerimientos en un diseño el cual nos guiará como implementar el sistema, además ayuda a establecer una arquitectura robusta y a ajustar el diseño con el entorno de implementación.

2.6.4. Implementación

Esta disciplina lleva a cabo varias actividades: se define la organización de la implementación, se implementa el diseño de los

elementos, se prueban los componentes como unidades y se integran los resultados modulares producidos para la creación de un ejecutable.

2.6.5. Pruebas

En las pruebas se buscan y documentan defectos con respecto a la calidad del software, se valida lo asumido en el diseño y en la especificación de requerimientos mediante una demostración.

Además, se comprueba y se valida que las funciones del sistema hayan sido implementadas como se diseñó y si cumplen satisfactoriamente con los requerimientos.

2.6.6. Despliegue

En el despliegue se asegura que el producto de software esté disponible para los usuarios finales.

2.6.7. Gestión del cambio y configuraciones

En esta disciplina se identifican los elementos de la configuración, se les hace un seguimiento y se restringen cambios de los elementos, para finalmente definir y administrar la configuración de estos elementos.

2.6.8. Gestión de proyecto

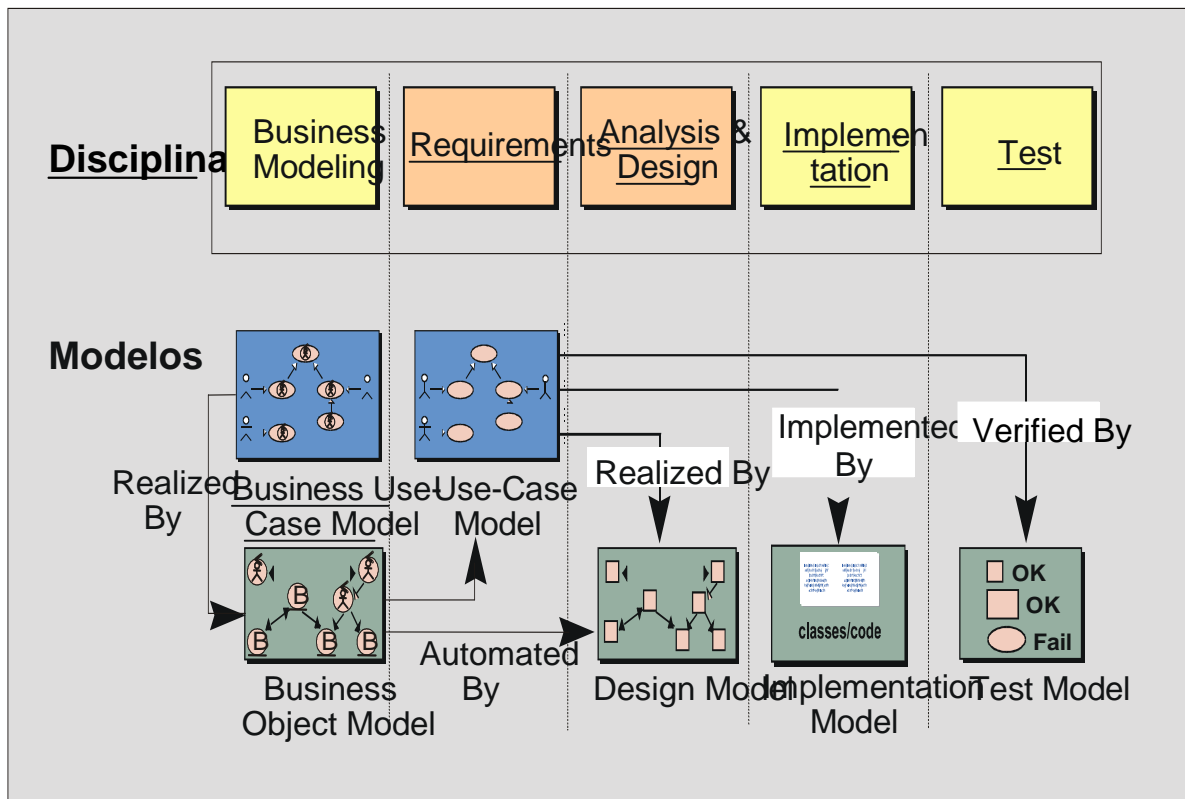
En la administración del proyecto se conduce, planifica, se asigna personal, se ejecuta y se monitorea, intensivamente el proyecto de software , y finalmente se analiza el riesgo.

2.6.9. Entorno

En esta disciplina se organiza el desarrollo de software con el entorno del mismo lo cual será un gran apoyo para el equipo de desarrollo. Además, se personalizará la configuración para un

proyecto específico, así como el desarrollo de guías de trabajo que brindarían soporte al proyecto.

2.6.10. Las Disciplinas Producen Modelos



FUENTE: <http://www.baufest.com>, Baufest software Engineering

Dado que el Proceso Unificado³ es un proceso dirigido por casos de uso explicado anteriormente. Cada una de las disciplinas o flujos de trabajo, vistas anteriormente nos llevan a desarrollar distintos modelos para el análisis del sistema.

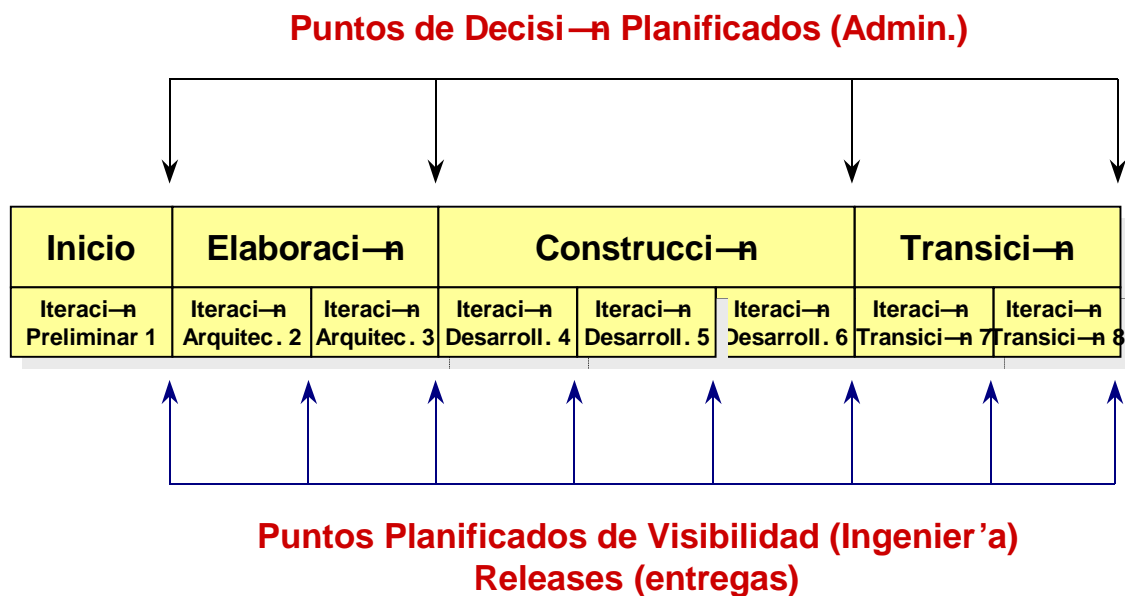
2.7. Fases

³ El proceso unificado de desarrollo de Software, Ivar Jacobson, Grady Booch, James Rumbaugh, Addison Wesley

Como se vio anteriormente cada iteración produce un producto ejecutable, que se le irá corrigiendo y añadiendo mejoras para llegar a convertirse en un sistema final.

Dentro de cada iteración existen fases, en donde se da más énfasis dependiendo de la iteración. Una fase es un intervalo de tiempo definido entre dos hitos.

Cada fase termina con un hito que es en donde los directores de proyectos deben tomar decisiones cruciales antes que el trabajo continúe.



FUENTE: <http://www.baufest.com>, Baufest software Engineering

2.7.1. En la fase de inicio

Se desarrolla una descripción del producto final y se responde a las siguientes preguntas:

- Cuales son las principales funciones del sistema para sus usuarios más importantes?
- Como podría ser la arquitectura del sistema?
- Cuál es el plan de proyecto y cuánto costará el desarrollo?

2.7.2. En la fase de elaboración

Se especifica en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema. Al final de esta fase el director está en disposición de planificar las actividades y estima los recursos necesarios para terminar el proyecto.

2.7.3. En la fase de construcción

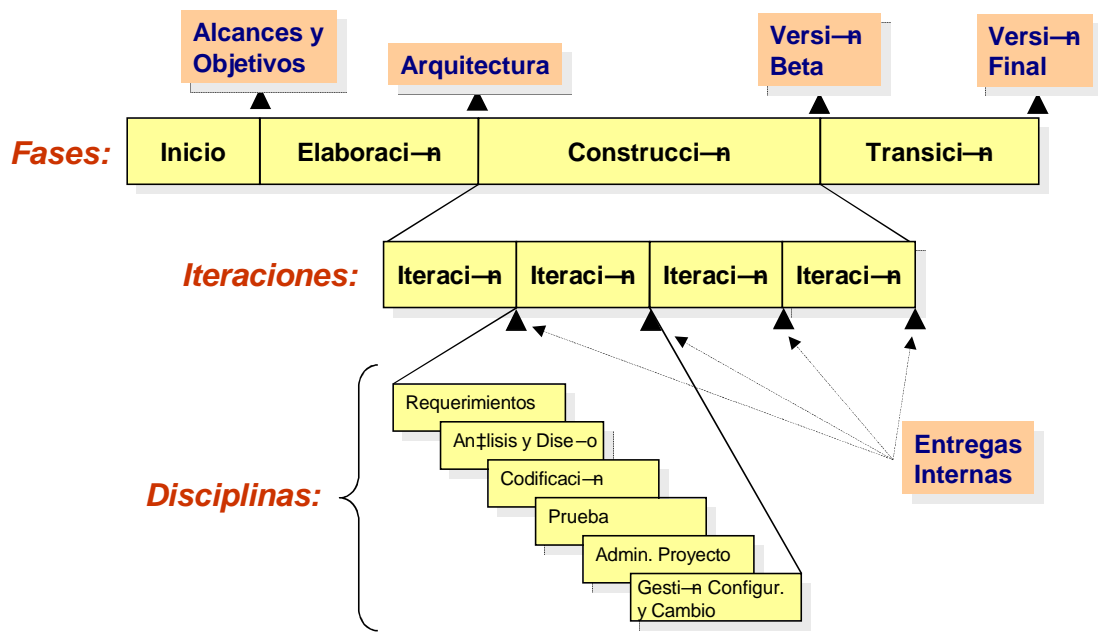
Se crea el producto, se añade el software terminado a la arquitectura. En esta fase la línea base de la arquitectura crece hasta convertirse en el sistema completo.

La pregunta que se emplea en esta fase es:

¿ Cubre el producto las necesidades de los usuarios de manera suficiente como para hacer una primera entrega?

2.7.4. En la fase de transición

Cubre el periodo durante el cual el producto se convierte en versión beta , y a medida que los desarrolladores con la ayuda de un número reducido de usuarios corrigen los problemas e incorporan mejoras dirigidas a todos los usuarios. El producto llegará a una versión final (o la primera versión en producción) después de las iteraciones que sean necesarias aplicar.



FUENTE: <http://www.baufest.com>, Baufest software Engineering

2.8. Lenguaje Unificado de Modelado, UML

2.8.1. Introducción

UML es un lenguaje gráfico para especificar, construir, visualizar y documentar los artefactos de un sistema de software orientado a objetos. UML ⁴ proporciona una forma estándar de escribir los planos de un sistema cubriendo tanto los conceptos y los procesos de negocio, como las cosas concretas y las clases escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes de software reutilizables.

UML se ha convertido en un lenguaje estándar con el que es posible modelar todos los componentes del proceso de desarrollo de aplicaciones. Sin embargo, hay que tener en cuenta un aspecto importante del proceso: no pretende definir un modelo estándar de desarrollo, sino únicamente un lenguaje de modelado

Uno de los objetivos del UML es llegar a convertirse en una manera de definir modelos, no sólo establecer una forma de modelo, de esta forma simplemente estamos diciendo que UML, además, define un lenguaje con el que podemos abstraer cualquier tipo de modelo.

2.8.2. Diagrama de actividades

Representa la naturaleza dinámica de un sistema modelando el flujo de control de una actividad a otra. Una actividad representa una operación de alguna clase en el sistema que resulta en un cambio de estado. los diagramas de actividades son usados durante el flujo de trabajo o procesos de negocios y operaciones internas.

⁴ El lenguaje unificado de modelado, Ivar Jacobson, Grady Booch, James Rumbaugh, Addison Wesley

Son similares a los diagramas de flujo de algunas metodologías OO. En realidad, se corresponden con un caso especial de los diagramas de estado donde los estados son estados de acción (estados con una acción interna y una o más transiciones que suceden al finalizar esta acción, o lo que es lo mismo, un paso en la ejecución de lo que será un procedimiento) y las transiciones vienen provocadas por la finalización de las acciones que tienen lugar en los estados de origen. Siempre van unidos a una clase o a la implementación de un caso de uso o de un método (que tiene el mismo significado que en cualquier metodología OO).

Los diagramas de actividad se utilizan para mostrar el flujo de operaciones que se desencadenan en un procedimiento interno del sistema.

2.8.3. Diagrama de Casos de uso

Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar como reacciona una respuesta a eventos que se producen en el mismo. En este tipo de diagrama intervienen algunos conceptos nuevos: un actor es una entidad externa al sistema que se modela y que puede interactuar con él; un ejemplo de actor podría ser un usuario o cualquier otro sistema. Las relaciones entre casos de uso y actores pueden ser las siguientes:

- Un actor se comunica con un caso de uso.
- Un caso de uso extiende otro caso de uso.
- Un caso de uso usa otro caso de uso

2.8.4. Diagrama de clases

Muestra las clases (descripciones de objetos que comparten características comunes) que componen el sistema y cómo se relacionan entre sí.

Los diagramas de clases representan un conjunto de elementos del modelo que son estáticos, como las clases y los tipos, sus contenidos y las relaciones que se establecen entre ellos.

Las clases se relacionan entre sí de distintas formas, que marcan los tipos de relaciones existentes:

Asociación:

Es una relación que describe un conjunto de vínculos entre clases. Pueden ser binarias o n-arias, según se implican a dos clases o más. Las relaciones de asociación vienen identificadas por los roles, que son los nombres que indican el comportamiento que tienen los tipos o las clases, en el caso del rol de asociación. Es posible indicar el número de instancias de una clase que participan en una relación mediante la llamada multiplicidad.

Las relaciones de asociación permiten especificar qué objetos van a estar asociados con otro objeto mediante un calificador. El calificador es un atributo o conjunto de atributos de una asociación que determina los valores que indican cuales son los valores que se asociarán.

Generalización:

Cuando se establece una relación de este tipo entre dos clases, una es una Superclase y la otra es una Subclase. La subclase comparte la estructura y el comportamiento de la superclase. Puede haber más de una clase que se comporte como subclase.

Dependencia:

Una relación de dependencia se establece entre clases (u objetos) cuando un cambio en el elemento independiente del modelo puede requerir un cambio en el elemento dependiente.

Agregación:

Forma de asociación que especifica una relación todo – parte entre un agregado (el todo) y las partes que lo componen. Una agregación se representa mediante un rombo en el extremo de la línea de la asociación que se conecta a la clase agregada.

2.8.5. Diagrama de componentes

Muestra la dependencia entre los distintos componentes de software, incluyendo componentes de código fuente, binario y ejecutable. Un componente es un fragmento de código software (un fuente, binario o ejecutable) que se utiliza para mostrar dependencias en tiempo de compilación.

El diagrama de componentes muestra los tipos de componentes del sistema; una configuración particular de la aplicación puede tener más de una copia de un componente.

2.8.6. Diagrama de despliegue

En el diagrama de despliegue se indica la situación física de los componentes lógicos desarrollados. Es decir, se sitúa el software en el hardware que debe contener.

La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos. Un nodo es un recurso de ejecución representado por un cubo, tal como una computadora, un dispositivo, o memoria .

Esta vista permite determinar las consecuencias de la distribución y de la asignación de recursos.

2.8.7. Diagrama de secuencia

Muestran las interacciones o mensajes entre un conjunto de objetos, dispuestos en una secuencia temporal, es decir el diagrama de secuencia de un Sistema muestra gráficamente los eventos que fluyen de los actores al sistema.

En los diagramas de este tipo intervienen objetos, que tienen un significado parecido al de los objetos representados en los diagramas de colaboración, es decir son instancias concretas de una clase que participa en la interacción.

El objeto puede existir sólo durante la ejecución de la interacción, se puede crear o puede ser destruido durante la ejecución de la interacción. Un diagrama de secuencia representa una forma de indicar el período durante el que un objeto está desarrollando una acción directamente o a través de un procedimiento.

Los diagramas de secuencia son utilizados para mostrar la secuencia del comportamiento de un caso de uso. Cuando está implementando el comportamiento, cada mensaje en un diagrama de secuencia corresponde a una operación en una clase.

2.8.8. Diagrama de Colaboración

Muestra la interacción entre varios objetos y los enlaces que existen entre ellos. Representa las interacciones entre objetos, organizadas alrededor de los objetos y sus vinculaciones. A diferencia de un diagrama de secuencias, un diagrama de colaboraciones muestra las relaciones entre los objetos, no la secuencia en el tiempo en que se producen los mensajes.

Los diagramas de secuencias y los diagramas de colaboraciones expresan información similar, pero en una forma diferente. Cada diagrama debe ser utilizado cuando su aspecto principal es el foco de atención.

Durante la ejecución de un diagrama de colaboración se crean y destruyen objetos y enlaces.

2.8.9. Diagrama de estados

Representa la secuencia de estados por los que un objeto o una interacción entre objetos pasa durante su tiempo de vida, en respuesta a estímulos (eventos) recibidos. Representa lo que podemos denominar en conjunto, una máquina de estados. Este diagrama solo debe ser realizado para las clases más importantes. Un estado en UML es cuando un objeto o una interacción satisface una condición, desarrolla alguna acción o se encuentra esperando un evento.

Cuando un objeto o una interacción pasa de un estado a otro por la ocurrencia de un evento se dice que ha sufrido una transición, existen varios tipos de transiciones entre objetos: simples (normales y reflexivas) y complejas. Además una transición puede ser interna si el estado del que parte el objeto o interacción es el mismo al que llega, no se provoca un cambio de estado y se representan dentro del estado, no de la transición.

Son especialmente útiles los modelos en los cuales es necesario representar objetos que reaccionan a eventos específicos.

3. Capítulo: Plataforma Java

3.1. Introducción

Cuando hablamos de la tecnología Java ⁵nos referimos a 3 aspectos

El lenguaje Java : un lenguaje de programación.

La Java Virtual Machine : Una máquina virtual con su propio set de instrucciones.

La Java Platform API : Una interfaz para la programación de aplicaciones.

La plataforma Java se divide en tres distintas áreas de tecnología de desarrollo, cada cual con distintas necesidades y especificaciones para crear distintos tipos de aplicaciones.

Las distintas áreas son: J2SE (Java 2 platform, Standard Edition), J2EE (Java 2 platform, Enterprise Edition) y J2ME (Java 2 platform, Micro Edition).

J2SE es la plataforma más común, se centra en el desarrollo de una amplia variedad de aplicaciones que corren en el lado del cliente.

La plataforma J2EE proporciona un ambiente integrado para crear aplicaciones de Java empresariales de n-niveles .

J2ME se encarga de la construcción de aplicaciones para micro dispositivos como teléfonos celulares, PDA's, etc.

⁵ Profesional Java Server Programming - J2EE 1.3 Edition, Editorial Wrox,

La tecnología java a diferencia de seguir un modelo tradicional de una arquitectura de dos capas (cliente-servidor) usa una arquitectura multi-capa.

El estudio de esta investigación se centrará en la plataforma J2EE.

3.2. Introducción a J2EE

La plataforma J2EE fue diseñada para desarrollar sistemas empresariales, provee de especificaciones y un entorno de desarrollo estándar, esta plataforma principalmente toma en cuenta 3 consideraciones: Consideraciones del negocio, Consideraciones de Integración y consideraciones de desarrollo.

Consideraciones del negocio.- estas son la que contienen "la realidad del negocio" por lo general las aplicaciones empresariales no hacen cálculos complejos más bien estas manejan una compleja manipulación de datos.

Consideraciones de Integración.- La integración de los sistemas empresariales, se refiere a cómo se acoplan estos con otros sistemas puesto que estos sistemas pueden llegar a tener una infraestructura distribuida.

Consideraciones de desarrollo.- Se debe estar conciente de la complejidad de desarrollo y mantenimiento de sistemas empresariales, puesto que estos pueden presentar cambios continuos de las condiciones del negocio.

La ventaja de la arquitectura multi-capa es que el desarrollo de responsabilidades se realiza a través de diferentes capas. En cada capa de desarrollo se tiene un enfoque a un proceso específico y se realizan cambios independientemente.

Una de las características principales de la plataforma J2EE es que ésta se basa en contenedores, lo cual permite que los componentes desarrollados se ejecuten en un entorno el cual les va a proveer de todos los servicios que estos requieran.

Otra característica que marca a la plataforma J2EE es el Internet, dado a la creciente demanda de programas empresariales en línea han puesto que las organizaciones actualmente brinden nuevos servicios en línea, los cuales deberán ser de alta calidad y eficientes puesto a que la competencia de hoy en día se encuentra a un clic de distancia .

Es por esto que la plataforma J2EE nos brinda la tecnología necesaria para implantar estos sistemas.

Y por último la característica principal que brinda Java , la filosofía de que *Una vez escrito corre en cualquier plataforma.*

Todas estas características se toman en cuenta para el desarrollo e implementación de Sistemas empresariales.

3.3. Plataforma de J2EE

La plataforma J2EE brinda soporte para construir aplicaciones distribuidas empresariales del lado del servidor que ofrecen servicios a través de internet o redes privadas. Esta plataforma básicamente provee:

- Un set de Java Apis para construir aplicaciones, estos Apis definen un modelo de programación de J2EE.
- Una infraestructura en tiempo de ejecución que brinda hospedaje y soporte de administración para aplicaciones.

3.3.1. Ventajas de J2EE

- *Independencia de plataforma.* Las aplicaciones desarrolladas en J2EE ⁶son portables esto quiere decir, que si cambiamos el sistema operativo del servidor o el programa servidor de aplicaciones se puede seguir utilizando las aplicaciones J2EE sin necesidad de hacerles modificaciones.
- *Objetos gestionados por los contenedores.* Los contenedores gestionan los objetos de tal manera que se permite al desarrollador enfocarse solo en la solución del problema.
- *Reusabilidad.* Puesto que las aplicaciones bajo la especificación de J2EE son construidas en base a componentes, lo cual permite su reutilización de funciones . De esta forma se consigue desarrollar más rápido y a un menor costo, y se proporciona mayor robustez a las aplicaciones.
- *Modularidad.* Esta va a facilitar el mantenimiento de las aplicaciones, ya que si se desea realizar modificaciones en algún módulo, éstas no afectan al resto.

⁶ Profesional Java Server Programming - J2EE 1.3 Edition, Editorial Wrox,

En base a estas ventajas J2EE provee un estándar simple y unificado para la construcción de aplicaciones distribuidas a través de un modelo de aplicaciones basado en componentes.

3.3.2. J2EE Apis

Es un conjunto de servicios que brinda la plataforma para las aplicaciones distribuidas que van a desarrollar bajo esta plataforma, servicios como el proceso de transacciones, acceso a base de datos, manejo de mensajes , etc.

Todas las aplicaciones desarrolladas en J2EE acceden a estos apis o servicios a través del contenedor.

Las principales Apis de Java que soportan toda plataforma J2EE son:

- JDBC
- Enterprise Java Beans (EJB)
- Java Servlets
- Java Server Pages (JSP)
- Java Message Service (JMS)
- Java Transaction Api
- Java Mail
- Java Beans Activation Framework
- Java Api for Xml Parsing
- The Java Conector Architecture
- Java Authentication and Authorization Service

3.3.3. Los contenedores

Los contenedores son entornos los cuales proveen servicios de soporte para aplicaciones de componentes que se encuentran en

ejecución. La forma en que se ejecutan los componentes dentro del contenedor es transparente para el cliente .

Existen cuatro tipos de contenedores:

Contenedor de Applets, el cual provee los servicios para desplegar los Java Applets.

Contenedor de aplicaciones para cliente, provee los servicios necesarios para las aplicaciones que corren en el entorno del cliente

Contenedor Web, este contenedor provee los servicios necesarios para los Api´s de Jsp y Servlets.

Contenedor EJB (Enterprise Java Beans), este contenedor provee los servicios necesarios para Los Enterprise Java Beans.

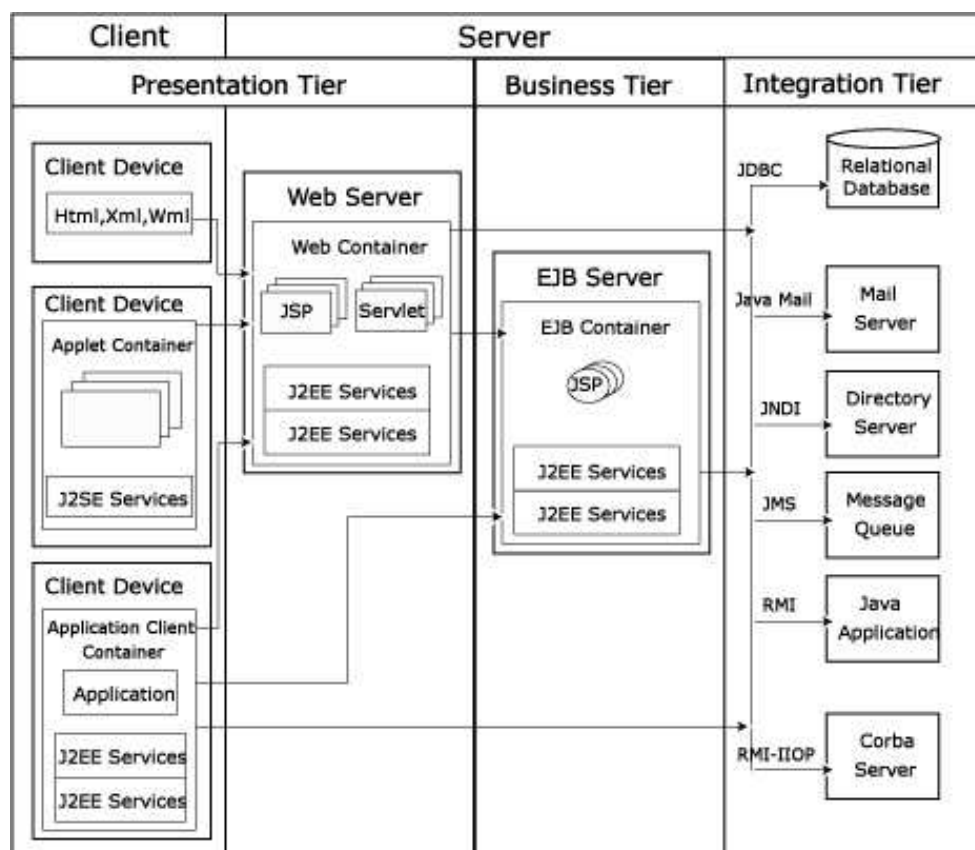


Figura 2.4. Tecnologías y Plataforma de J2EE

FUENTE: Building Applications with the Rational Unified Process, Peter Eeles-Kelli Houston-Wojtek Kozacynsky , Capítulo 2

En este gráfico se puede ver el despliegue de la configuración multi-capa.

Capa de Presentación

La capa de presentación está compuesta por elementos que residen tanto del lado del servidor, como del lado del cliente.

Los elementos del lado del cliente, son los que se encargan de mostrar la información y van a permitir la interacción con el cliente. Los elementos del lado del servidor, son los que se van a encargar de procesar las peticiones del cliente, devolviendo el resultado requerido.

Capa de Negocio

La capa de negocio, es la responsable de la lógica del negocio. Esta capa la componen los Enterprise Java Beans y otros objetos de negocios que residen en el servidor.

Capa de Integración

Este patrón se va a encargar de controlar la utilización de un Objeto de Acceso a Datos para abstraer y encapsular todos los accesos a fuentes de datos. De tal forma que se maneja y controla las conexiones con las fuentes de datos para obtener y guardar la información solicitada.

3.4. Tecnología de J2EE ⁷

Los componentes de la tecnología J2EE son los applets y las aplicaciones cliente, Java Servlets, Java Server Pages y Enterprise Java Beans.

⁷ Building Applications with the Rational Unified Process, Peter Eeles-Kelli Houston-Wojtek Kozacynsky , Capítulo 2

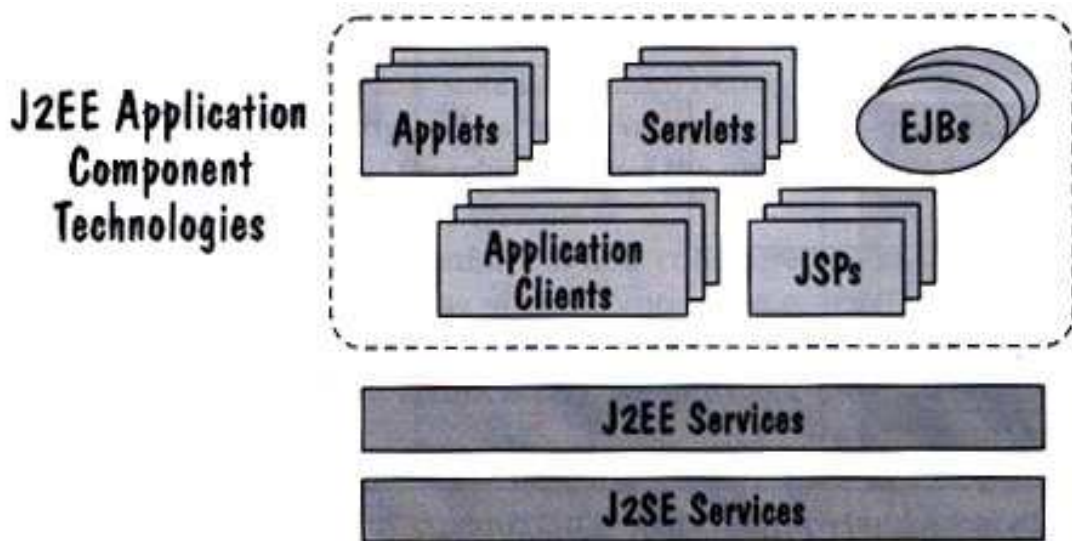


Figura 2.5. Resumen de la tecnología J2EE

FUENTE: Building Applications with the Rational Unified Process, Peter Eeles-Kelli Houston-Wojtek Kozacynsky , Capítulo 2

3.4.1. Applets

Los applets básicamente son pequeños programas hechos en Java, que se ejecutan en un contenedor de applet y se transfieren a las páginas web y que el navegador ejecuta en el espacio de la página.

3.4.2. Java Servlets

Es un programa independiente de la plataforma que aporta la misma funcionalidad a la programación en el lado del servidor.

El API Servlet consiste básicamente en dos paquetes:

javax.servlet En este paquete se definen 6 interfaces y 3 clases para la implementación de servlets genéricos, sin especificación de protocolo. Hoy en día no tienen utilidad práctica más que para servir de base en la jerarquía de clases de los servlets. Conforme pase el tiempo se supone que constituirán la base para la implementación de otros protocolos distintos de http.

javax.servlet.http Ofrece la implementación específica de servlets para el protocolo http.

3.4.3. Java Server Pages

JSP es una tecnología basada en Java que simplifica el proceso de desarrollo de páginas Web y aplicaciones que generan contenido dinámico. Dicho contenido incluye HTML, DHTML, XHTML, WML y XML. Las páginas JSP tienen la apariencia de una página HTML o WML tradicionales con código Java embebido.

En esencia, una página JSP describe cómo procesar una petición del cliente para generar una respuesta. Las páginas JSP son, en realidad, servlets: un JSP se compila a un programa en Java la primera vez que se invoca, y del programa en Java se crea una clase que empieza a ejecutarse en el servidor como un servlet.

La principal diferencia entre los servlets y los JSPs es el enfoque de la programación: un JSP es una página Web con tags especiales y código Java incrustado, mientras que un servlet es un programa que recibe peticiones y genera a partir de ellas una página web.

Como parte de la plataforma Java, la tecnología JSP permite desarrollar aplicaciones basadas en el Web que pueden ejecutarse en cualquier servidor que soporte Java.

Esta tecnología separa la presentación del contenido, es decir, la interface de usuario de la programación. Esto permite al diseñador modificar la página sin alterar el contenido dinámico.

JSP se basa en la reutilización de componentes (Java Beans). Esto es, en las páginas JSP se codifica dentro de los tags, y para resolver tareas complejas se accede a componentes beans reutilizables, de

esta manera, se optimiza considerablemente la utilización de recursos en el servidor.

Además, la tecnología JSP permite a los desarrolladores extender los tags JSP disponibles. Es decir, es posible crear librerías de tags personalizados, que ofrezcan mayor funcionalidad, mayor capacidad de reutilización de código y reduzcan la complejidad de la lógica de creación de la página.

3.4.4. Enterprise JavaBeans

Los Enterprise JavaBeans son una colección de clases y de archivos xml, atados a una sola unidad. Son componentes de software que permiten crear aplicaciones distribuidas y transaccionales mediante java. Los Enterprise JavaBeans, son las que contienen las funciones de la lógica del negocio y simplifican el desarrollo de las aplicaciones de bases de datos.

Las características principales son:

Permiten crear aplicaciones distribuidas, gracias a la combinación de componentes desarrollados por el usuario o por distintos proveedores de software.

Con ellos resulta fácil escribir aplicaciones. Los desarrolladores de aplicaciones no tienen que preocuparse por los detalles de bajo nivel relacionados con la gestión de transacciones o con la gestión del estado, la seguridad, la persistencia, los procesos multihebra, la agrupación de recursos y otras complejas interfaces de programación de aplicaciones (API) de bajo nivel.

Los programadores pueden tener acceso directo a las API de bajo nivel, pero es el servidor EJB el que gestiona la mayor parte de estos detalles.

Pueden desarrollarse y, a continuación, desplegarse en múltiples plataformas sin necesidad de recompilarlos ni de modificar el código fuente. Además que permiten la interoperatividad con otras aplicaciones Java y con aplicaciones que no sean Java.

Beans de Sesión

Un bean de sesión encapsula datos que no son permanentes a un cliente de EJB específico.

Existen dos tipos de Beans de Sesión

- Sin estado :
 - Beans ligeros
 - Ideales para procesos que no dependan del cliente

- Con estado :
 - Algo más pesados
 - Guardan el estado conversacional con el cliente
 - Ideales para procesos "con sesión" Ej: tienda

Beans de entidad

- Representan datos
- Son objetos pesados*

* Objetos que utilizan una considerable cantidad de recursos (memoria, conexiones de red, recursos de base de datos, entre otros).

- Obtienen servicios del contenedor(transacciones, concurrencia, ciclo de vida)
- Tienen persistencia propia, BMP o CMP

3.5.Aplicaciones Distribuidas en Múltiples Capas

El uso del Internet continuamente está creciendo en importancia y cada vez existen más empresas que emplean este medio para llegar directamente a sus clientes.

Si se considera que la competencia se encuentra a solo un clic de distancia, las empresas deben brindar continuamente servicios de alta calidad, para alcanzar una ventaja competitiva y ganar la lealtad de sus clientes . Es por esto que, es preciso que los sistemas de información ofrezcan la flexibilidad necesaria para una rápida adaptación a cambios en las reglas del negocio o un aumento del número de usuarios, de tal manera que estas modificaciones se efectúen de forma transparente para los usuarios.

A lo largo de la historia, el desarrollo de sistemas de información se ha realizado mediante el empleo de diferentes metodologías de desarrollo y arquitecturas. Así, en el ya tradicional modelo dos-capas (cliente-servidor), la carga del procesamiento reside en el cliente mientras que el servidor simplemente actúa como un controlador de tráfico entre la aplicación y los datos.

Cuando la aplicación completa es procesada en el cliente, ésta es forzada a efectuar múltiples peticiones de información al servidor antes de presentar una respuesta al usuario. Un número elevado de peticiones puede congestionar la red.

Otro problema común de las aplicaciones cliente-servidor, es el mantenimiento. Cualquier cambio efectuado en la aplicación implica una actualización en cada uno de los clientes, lo cual puede repercutir en diferentes estaciones de trabajo (clientes) utilizando diferentes versiones de la aplicación.

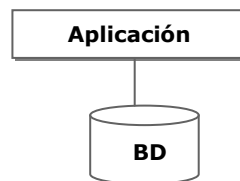


Figura 2.1. Modelo dos capas, cliente-servidor
FUENTE: Profesional Java Server Programming - J2EE 1.3 Edition, Editorial Wrox, Capítulo 1

Con el fin de superar estas limitaciones aparece la noción de la arquitectura de tres-capas. Una aplicación basada en este modelo está compuesta por tres capas lógicas independientes, cada una con un juego bien definido de interfaces.

La primera capa se conoce con el nombre de **capa de presentación**, y típicamente consiste en una interfaz gráfica de usuario. La capa intermedia, o **capa del negocio**, corresponde a la aplicación o lógica del negocio. Por último, la tercera capa -**capa de datos**- contiene la información necesaria para la aplicación.

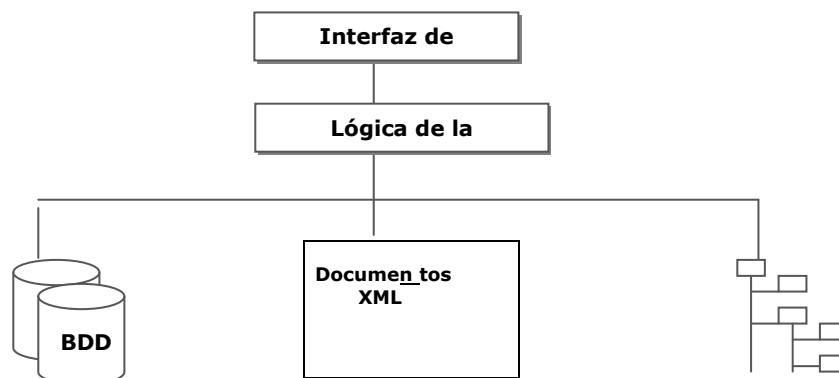


Figura 2.2. Modelo tres capas
FUENTE: Profesional Java Server Programming - J2EE 1.3 Edition, Editorial Wrox, Capítulo 1

La capa intermedia (lógica de la aplicación) es básicamente el código que el usuario llama (a través de la capa de presentación) para obtener la información deseada.

La capa de presentación entonces recibe los datos para desplegarlos al usuario. Los datos contenidos en la tercera capa pueden provenir de diferentes fuentes de información como una base de datos, archivos XML, o servicio de directorios.

La separación de la lógica de la aplicación de la interfaz de usuario, proporciona gran flexibilidad al diseño de la aplicación. Es posible añadir múltiples interfaces de usuario sin tener que efectuar cambios en la lógica de la aplicación.

Existe otro tipo de arquitectura, el modelo n-capas. N-capas (n-tier) es un término que ha ganado popularidad rápidamente. N-capas se refiere a un número cualquiera de capas o niveles, sin límite.

Los sistemas basados en este modelo están en la capacidad de soportar un número de diferentes configuraciones .

La lógica de la aplicación está dividida por funciones de forma lógica antes que física.

La arquitectura n-capas se compone por:

- Una **interfaz de usuario**, que maneja la interacción del usuario con la aplicación. Esta puede ser un navegador (browser) corriendo a través de un firewall, una aplicación de escritorio o incluso un dispositivo inalámbrico.

- La **lógica de presentación**, que define lo que despliega la interfaz de usuario y cómo se manejan las peticiones que éste realiza. Dependiendo del tipo de interfaces soportadas, es necesario utilizar diferentes versiones de *lógica de presentación* para manejar el cliente de forma apropiada.
- La **lógica del negocio**, que modela las reglas del negocio de la aplicación, a menudo a través de la interacción con los datos de la aplicación.
- Los **servicios de infraestructura**, que proveen funcionalidad adicional requerida por componentes de la aplicación como mensajería o soporte transaccional (Java Beans, Servlets).
- La **capa de datos** donde residen los datos del negocio.

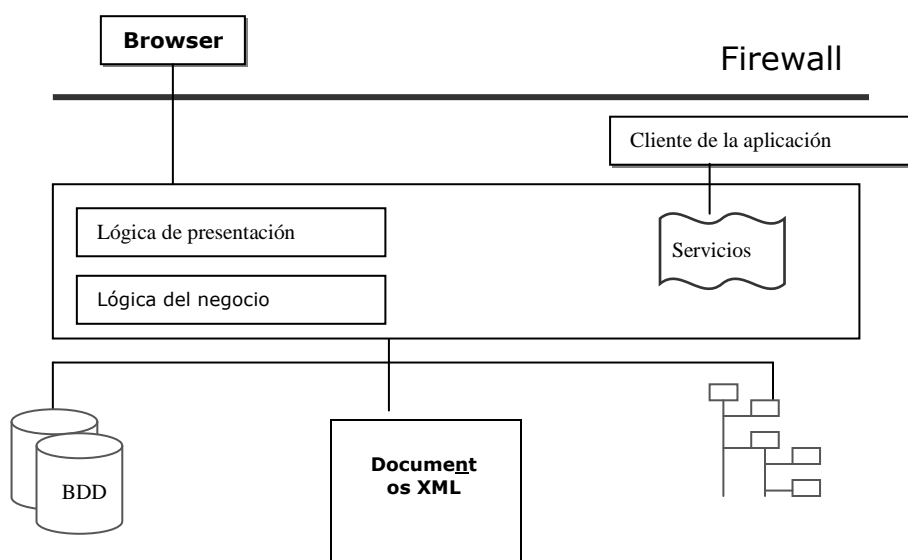


Figura 2.3. Modelo multicapa

FUENTE: Profesional Java Server Programming - J2EE 1.3 Edition, Editorial Wrox, Capítulo 1

Una aplicación basada en la arquitectura n-capas está caracterizada por una descomposición funcional de aplicaciones y componentes, lo cual provee una mayor escalabilidad, una mejor administración y reutilización de recursos.

Una capa o nivel es un componente de software y hardware separado funcionalmente que ejecuta una operación específica. Generalmente, las capas intermedias de esta arquitectura son administradas por uno o más servidores de aplicaciones.

La función principal de un servidor de aplicaciones es brindar servicios a uno o más clientes. Así como crece la necesidad de soportar un número de clientes cada vez mayor, también crece la importancia de responder a los clientes de la manera más rápida y eficiente posible.

Una de las técnicas utilizadas para mejorar la eficiencia del servicio consiste en reutilizar, en tiempo de ejecución, instancias de objetos pesados (objetos que consumen gran cantidad de recursos). Por lo general, es aconsejable reducir el número de objetos instanciados en una aplicación; más aún en aplicaciones de servidor (aplicaciones web), donde no es fácil prever el número de clientes. Si se puede reutilizar instancias de estos objetos en lugar de crearlos cada vez que se requieran, la eficiencia de los servidores mejora sustancialmente.

En el modelo n-capas, los problemas de tráfico de red debido a un elevado número de peticiones a la base de datos (común en aplicaciones cliente servidor) es tratado mediante lo que se conoce como *Object Pooling*. El *Object Pooling* permite administrar y

reutilizar instancias de objetos en tiempo de ejecución, evitando así que se creen estos objetos cada que se requieran o pidan.

Con respecto al mantenimiento, el modelo n-capas exhibe una gran ventaja en relación a sus predecesores. Cada vez que se realiza una actualización en la aplicación no se requiere reinstalar la aplicación en cada terminal cliente, sino únicamente en la capa intermedia correspondiente (servidor de aplicaciones). Por tanto, no existe la posibilidad de que ocurra un conflicto de versiones.

4. Capítulo 4: Tecnología

4.1. Hibernate

4.1.1. Introducción

Para entender la tecnología de Hibernate ⁸ primero hay que entender qué es un motor de persistencia y que ventaja nos ofrece.

Las bases de datos utilizan un modelo "relacional", que se ha convertido en un estándar para el almacenamiento permanente de información, en cambio, los programas usan un modelo "orientado a objetos", que difiere en mucho del modelo relacional y que se ha extendido cada vez más. Es por ello que aparece un conflicto al momento de vincular estos dos componentes en una aplicación, ya que cada uno responde a diferente modelo y forma de operar. Cada componente maneja los datos con un formato diferente. Se podría decir que el programa y la base de datos hablan idiomas diferentes, razón por la cual la comunicación entre ellos resulta muy difícil.

Un motor de persistencia sirve de "traductor" entre los dos formatos de datos: de registros a objetos y de objetos a registros o de objetos a otro formato de almacenamiento de información.

El término **persistencia**, se refiere a la capacidad de poder almacenar el estado de un elemento de modo que sea posible recuperarlo posteriormente. De esta forma un motor de persistencia analógicamente se puede comparar con una librería que nos permite guardar y recuperar elementos almacenados ya sea en una base de datos o en cualquier otro mecanismo de persistencia .

⁸ (<http://www.programacion.com/java/tutorial/hibernate/>)

Los motores de persistencia no sólo nos permiten hacer persistentes nuestros objetos, sino que también nos permiten realizar consultas complejas en el modelo, efectuando optimizaciones de manera automática, resolviendo asociaciones y dependencias.

La ventaja de utilizar un motor de persistencia es la sencillez del modelo de programación que promueven.

El desarrollador ya no necesita preocuparse en como implementar un esquema de persistencia para sus objetos y puede concentrarse en la lógica del negocio. Una ventaja importante de utilizar un motor de persistencia, es la indiscutible independencia que se alcanza entre la aplicación y el esquema de almacenamiento de datos, es decir, la base de datos.

4.1.2. Qué es hibernate

Hibernate es un potente motor de persistencia y recuperación de objetos para Java el cual nos permite operar con elementos (objetos) y se encarga de realizar los accesos a la base de datos, además que nos permite generar sentencias SQL para aplicaciones o modelos, elementos como objetos, tipos primitivos y hasta arreglos brindando una seguridad integral a la aplicación.

Hibernate, la capa intermedia es un motor de aplicaciones o application server que contiene y ejecuta la aplicación. La capa interna la conforma la Base de Datos y se la usa como repositorio de información.

Gracias a Hibernate es posible cambiar de motor de base de datos sin tener que modificar una sola línea de código en la aplicación.

Con tan solo editar un archivo de configuración, es posible cambiar el repositorio de datos con el cual trabaja la aplicación

4.1.3. La Arquitectura

Una de las mayores ventajas de la arquitectura Hibernate es que se integra en cualquier tipo de aplicación, siendo totalmente independiente de la base de datos. En la siguiente configuración de hibernate se puede notar que están separados los distintos componentes de la aplicación

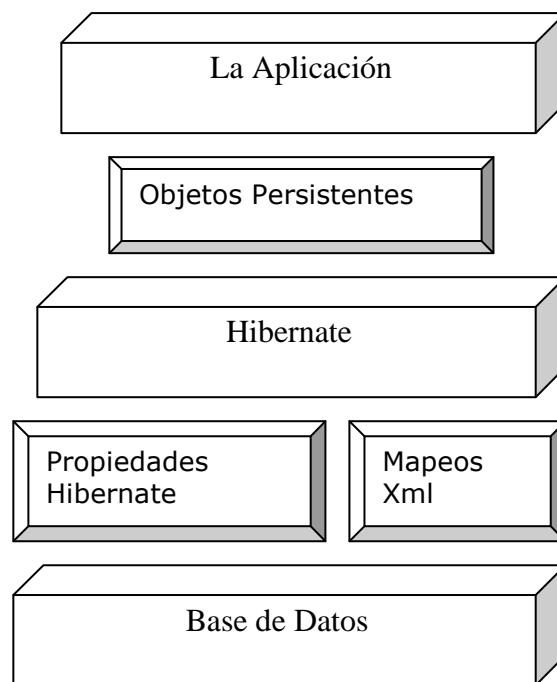


Figura 4.1. Configuración de ejemplo
Fuente: <http://www.programacion.com/java/tutorial/hibernate/>

Los Objetos persistentes son las clases del modelo de la aplicación y las propiedades hibernate es un archivo donde se mantiene la configuración de hibernate como la conexión a la base de datos, el driver, etc.

Y por último los archivos de mapeo xml los cuales son los archivos que se encargan de sincronizar los objetos con la base de datos.

4.1.4. Mapas de objetos relacionales en ficheros XML

En los ficheros de XML se configuran las propiedades de Hibernate para poder acceder a una base de datos determinada. Básicamente se "Mapea " los objetos que van a hacer persistencia contra la base de datos.

4.2. Framework Struts

4.2.1. Introducción

Struts⁹ es un framework o marco de trabajo para aplicaciones Web, el cual usa el patrón de arquitectura MVC (Modelo-Vista-Controlador).

Struts está conformado por parte del controlador, es un conjunto de clases y TAG-LIBS.

El Modelo o lógica de negocio es la parte que corresponde desarrollar al programar.

Y por último la integración con el Modelo (o lógica de negocio) y facilitan la construcción de vistas.

Struts define independientemente el modelo o lógica de negocio que es la parte que corresponde desarrollar al momento de programar.

⁹ (<http://jakarta.apache.org/struts>)

el controlador (flujo de la aplicación) la cual se encuentra conformado por es un conjunto de clases y TAG-LIBS.

Y la vista (interfaz con el usuario u otro sistema) facilita la construcción de vistas.

También es evidente que struts potencia la reutilización, soporte de múltiples interfaces de usuario (Html, sHtml, Wml, Desktop applications, etc.) y de múltiples idiomas, localismos, etc.

La arquitectura MVC (Model/View/Controller) fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos.

Se caracteriza porque el modelo, las Vistas y los controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el modelo se refleje automáticamente en cada una de las vistas.

Una de las principales ventajas principales de incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unirlos en tiempo de ejecución. Si uno de los componentes posteriormente se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas.

4.2.2. Definición de las partes MVC

El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema

el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.

El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

4.2.3. Como funciona?

En el caso del patrón MVC el procesamiento se lleva a cabo entre sus tres componentes. El controlador (un servlet especializado) recibe una orden y decide quien la lleva a cabo en el modelo, es decir llama al Action correspondiente pasándole los parámetros enviados.

El Action instanciará y/o utilizará los objetos de negocio para concretar la tarea. Una vez que el modelo (la lógica de negocio) termina sus operaciones devuelve el flujo al controlador y este envía el resultado a la capa de presentación.

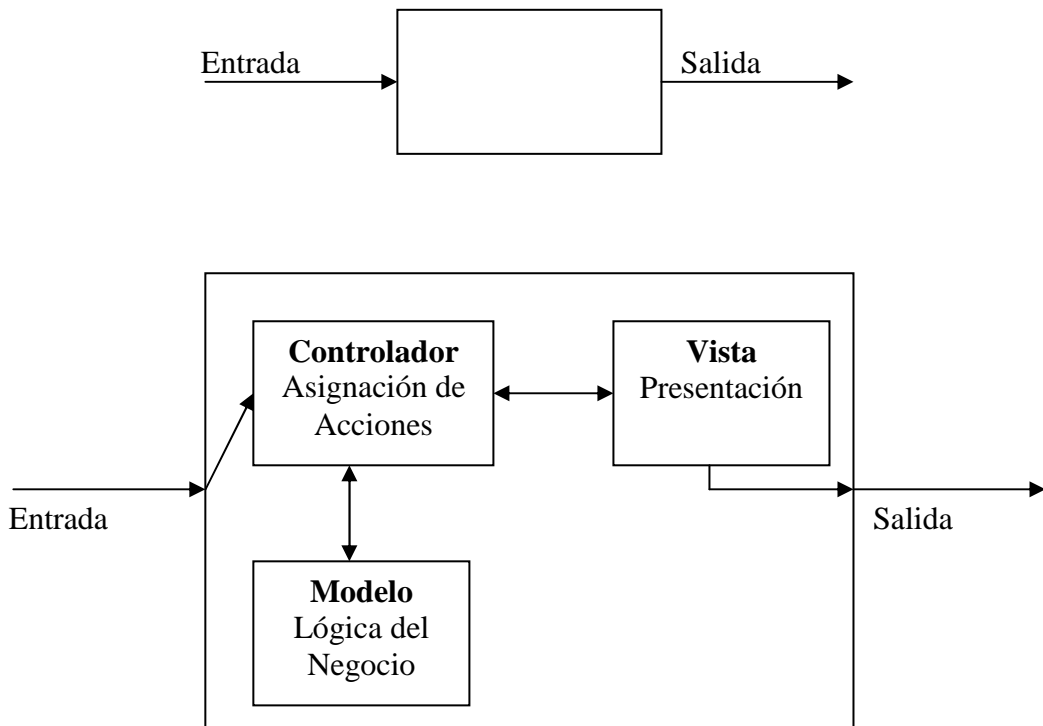


Figura 4.1. Configuración de ejemplo
Fuente: Professional Jakarta Struts, James Goodwill, Richard Hightower, Capítulo 1

La Ventaja principal que obtenemos a partir de este modelo es una separación total entre lógica de negocio y presentación. A esto se le pueden aplicar opciones como el multilinguaje, distintos diseños de presentación,.. etc sin alterar la lógica de negocio.

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

4.2.4. Tiles en Struts.

El "framework Tiles ¹⁰component view" nos presenta la solución a para la presentación visual de las aplicaciones basadas en web , puesto que los nos permite manejar plantillas y esquemas de diseño. Este framework usa un archivo de configuración para manipular estos tiles y además posee una amplia librería de etiquetas reutilizables las cuales complementan a Struts.

Tiles no solo permite reusar tiles, sino que también que permite organizar esquemas (layouts) de diseño .

4.3. Herramientas Empleadas en el desarrollo

4.3.1. Servidor Web Apache

Es el servidor web más utilizado con un gran porcentaje de cuota de mercado dentro de los equipos conectados a Internet. Éste es uno de los programas de más prestigio de "software libre" que se han desarrollado hasta ahora. Administra páginas de contenido estático.

4.3.2. Servidor de Aplicaciones Jakarta Tomcat

En el desarrollo del presente trabajo utilizamos la versión 4.0.1 del servidor de aplicaciones Jakarta Tomcat, del consorcio Apache.

Tomcat es el servidor de aplicaciones Open Source más popular debido a los beneficios que brinda, pues es relativamente rápido, y fácil de instalar. Tomcat puede servir páginas solo o bien como un añadido al servidor Apache.

¹⁰ (http://jakarta.apache.org/struts/userGuide/dev_tiles.html)

4.3.3. MySQL

MySQL es la base de datos SQL de código abierto más popular.

Entre las ventajas de utilizar esta base de datos están:

- Rendimiento. Es rápida tanto al conectar con el servidor como al ejecutar consultas.
- Es fácil de manejar, existen excelentes utilidades de administración (backup, recuperación de errores, entre otros).
- Es un motor de datos muy seguro, en el caso de que llegara a cerrarse, no suele perder información ni corromper los datos.
- No hay límites en el tamaño de los registros.
- Mejor control de acceso, en el sentido de qué usuarios tienen acceso a qué tablas y con qué permisos.
- Es una base de datos SQL-99 ANSI .
- MYSQL ha venido siendo utilizada en ambientes de producción extremadamente exigentes durante algunos años con gran éxito.
- Sin embargo, MySQL presenta algunos inconvenientes que van a ser superados en corto plazo:
- No soporta transacciones, "roll-backs" ni subselects (subqueries).
- No considera las claves foráneas. Es decir, ignora la integridad referencial, dejándola en manos del programador de la aplicación.

Las principales razones por las cuales se escogió trabajar con esta base de datos son las siguientes:

- Motor de base de datos de conocida eficiencia, rapidez de procedimiento y confiabilidad.¹¹
- Es uno de los motores de bases de datos más difundidos para la implementación de sistemas web, además se encuentra disponible en planes de hospedaje en Internet sin costo adicional.
- Su facilidad de uso.
- Su licencia de uso no tiene costo. Cualquiera puede bajar el código del Internet sin ningún costo. De igual manera, la herramientas de administración.

Para modelar la estructura lógica de la base de datos y crear el script de generación se utilizó una versión de evaluación de Power Designer 9 de Sybase.

Para administrar la base de datos se utilizó el frontal Mysqlfront Versión 2.4, el cual presta una interfaz intuitiva fácil de utilizar con varias herramientas para consultar y manipular la información.

4.3.4. Jrun Studio

JRun Studio proporciona un entorno de desarrollo y opciones de asistencia para escribir, compilar, depurar y desplegar páginas JSP y servlets. Además de herramientas personalizables para el desarrollo de contenido Web.

Se utilizó una versión de evaluación de Jun Studio para la programación de páginas JSP y páginas HTML del sitio Web del Sistema de Shopping Cart.

¹¹ Comparación de rendimiento con otros sistemas manejadores de bases de datos se puede encontrar en <http://www.mysql.com/doc/en/>

4.3.5. La herramienta de construcción Ant

Ant es una herramienta de construcción de proyectos basada en clases de Java, similar a make de Unix pero que no depende del Sistema Operativo.

En lugar de un entorno integrado, para 'construir' todos los ejercicios usaremos 'Ant', que es un producto de software libre del proyecto Jakarta

Ant usa un fichero XML para realizar tareas como compilar y desplegar todo tipo de aplicaciones.

El nombre por defecto de este fichero es 'build.xml'. Para conocer las tareas definidas en el fichero, podemos hacer, desde el directorio en el que está el fichero, 'ant -projecthelp'

4.3.6. La herramienta Xdoclet

XDoclet es un generador de código fuertemente integrado con Ant que, entre otras tareas, facilita la creación de EJB's. Al igual que Ant es un proyecto de código abierto. Xdoclet permite generar la interfaz home , externa y los descriptores a partir de la clase de implementación y comentarios javaDoc en ella. De esa manera toda la información está en un solo lugar. Al comienzo parece poco útil, pero cuando los proyectos crecen es indispensable.

4.3.7. Eclipse

Eclipse es una excelente herramienta IDE (Integrated Development Environment) de código abierto para Java y otros lenguajes que apoya el desarrollo de aplicaciones J2EE y que hacen más fácil tareas tediosas y repetitivas.

La escalabilidad de esta herramienta viene dado por plug-ins que expanden su funcionalidad. Permitiendo por ejemplo el Modelamiento de clases, Modelamiento de datos, generación de código y módulos especializados en frameworks (Hibernate, struts)

4.3.8. Rational Rose

Rational Rose es la herramienta CASE de la casa Rational desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables.

El navegador UML de Rational Rose nos permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable.

Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción.

4.3.9. Herramientas de Diseño Gráfico

En el diseño del sitio Web del Sistema de Shopping Cart, utilizamos las versiones de Evaluación de los siguientes programas:

- Macromedia Flash MX, es un software que permite crear aplicaciones y contenido dinámicos para el Internet, utilizando

potentes videos y multimedia que dan un mayor dinamismo a las interfaces de usuario. Esta herramienta fue utilizada para la animación de la presentación inicial de la interfaz Web y animaciones internas de artes gráficos del sistema de Shopping Cart.

- Macromedia Dreamweaver MX, con esta herramienta se armó la estructura gráfica de la página (tablas, iframes).
- Adobe Illustrator 10.0, herramienta que permite retocar, componer, mejorar y corregir imágenes fotográficas de alta calidad y resolución.
- Adobe Photoshop 7.0, es un programa de diseño con el cual se efectuó el corte de las imágenes elaboradas con Adobe Illustrator 9.0 para armar la estructura de las páginas. Este programa permite exportar las imágenes con formato .gif y .jpg, respectivamente.
- TopStyle Pro 3.0, facilita la creación y edición de archivos .css (Cascading Style Sheets), que contienen las definiciones de estilo de las páginas HTML. El archivo .css se importa o se enlaza con las páginas HTML.

5. Capítulo 5: Desarrollo de la Aplicación

5.1. Especificación De Requerimientos Del Software

5.1.1. Objetivo del documento.

Este documento tiene por objetivo definir exactamente como va a estar estructurado el sistema, así como el alcance que va a tener, Todo este contenido del documento se basa sobre todo en un profundo análisis de los requerimientos para estimar las soluciones que se plantea con el sistema propuesto.

Este es el documento en el cual el sistema se va a basar para su desarrollo, sin dejar de lado la posibilidad de cambios de acuerdo a requerimientos que puedan descubrirse sobre la marcha del diseño.

5.1.2. Alcance del sistema.

En el desarrollo del Shopping Cart cubre los siguientes puntos:

- Análisis de requerimientos.
- Diseño de las interfaces.
- Diseño de la navegación del sistema.
- Diseño de la base de datos.
- Diseño y desarrollo de la funcionalidad de la aplicación Web.
- Creación e integración de la interfaz entre la aplicación y la base de datos.
- Configuración de servicios.
- Pruebas funcionales.

5.1.3. Propósito del sistema.

El propósito principal del presente trabajo consiste en la investigación y aplicación del Proceso Unificado de Rational haciendo uso práctico de nuevas tecnologías orientadas a Internet en soluciones informáticas que ofrezcan servicios diferenciados y ventajas competitivas.

Para cumplir con este objetivo se seleccionó un caso de estudio real, en el cual la aplicación de estas tecnologías constituye un beneficio claro para la naturaleza del negocio o caso de estudio.

La aplicación seleccionada es el shopping Cart; una aplicación e-commerce que actualmente tiene una demanda muy alta en el mercado.

5.1.4. Descripción del sistema.

El Sistema Shopping Cart es un sistema en línea el cual permite a los clientes registrados crear sus pedidos añadiendo productos mostrados en el Catálogo Virtual . En cualquier momento el cliente puede visualizar o modificar los productos contenidos en su orden de compra.

La orden muestra información como: el producto, su valor unitario, el número de unidades solicitadas, el precio total por producto y el precio total de compra.

Una vez que el cliente tiene lista su orden de compra tiene la posibilidad de guardarla para recuperarla en una visita posterior, ó realizar inmediatamente la orden.

Antes de poder efectuar la orden el cliente deberá registrar sus datos personales así como también los datos de su tarjeta de su tarjeta de crédito y la dirección de entrega del pedido.

El sistema se deja la orden creada en estado pendiente para que según la organización realice el cobro a través del sistema de cobros de su preferencia.

5.1.5. Definiciones y abreviaturas.

Catálogo Virtual: En el sistema el catalogo virtual hace referencia al catalogo de productos que se encuentran almacenados en la base de datos.

Shopping Cart: El Carrito de Compras es un objeto el cual contiene productos y a este se le puede ir modificando a gusto del usuario.

5.1.6. Visión general

El principal objetivo que tiene este documento es dar a conocer el alcance que pretende el sistema propuesto, dando una detallada descripción de cómo va a funcionar y que resultados va a dar en base a los datos que requiere el mismo para sus procesos.

5.1.7. Descripción General

El sistema consta de dos partes principales:

- El catálogo virtual de productos donde los usuarios pueden buscar productos y registrarse en el sistema para crear órdenes.

- La interfase de administración la cuál permite al administrador actualizar y administrar el sistema.

Perspectiva de casos de uso.

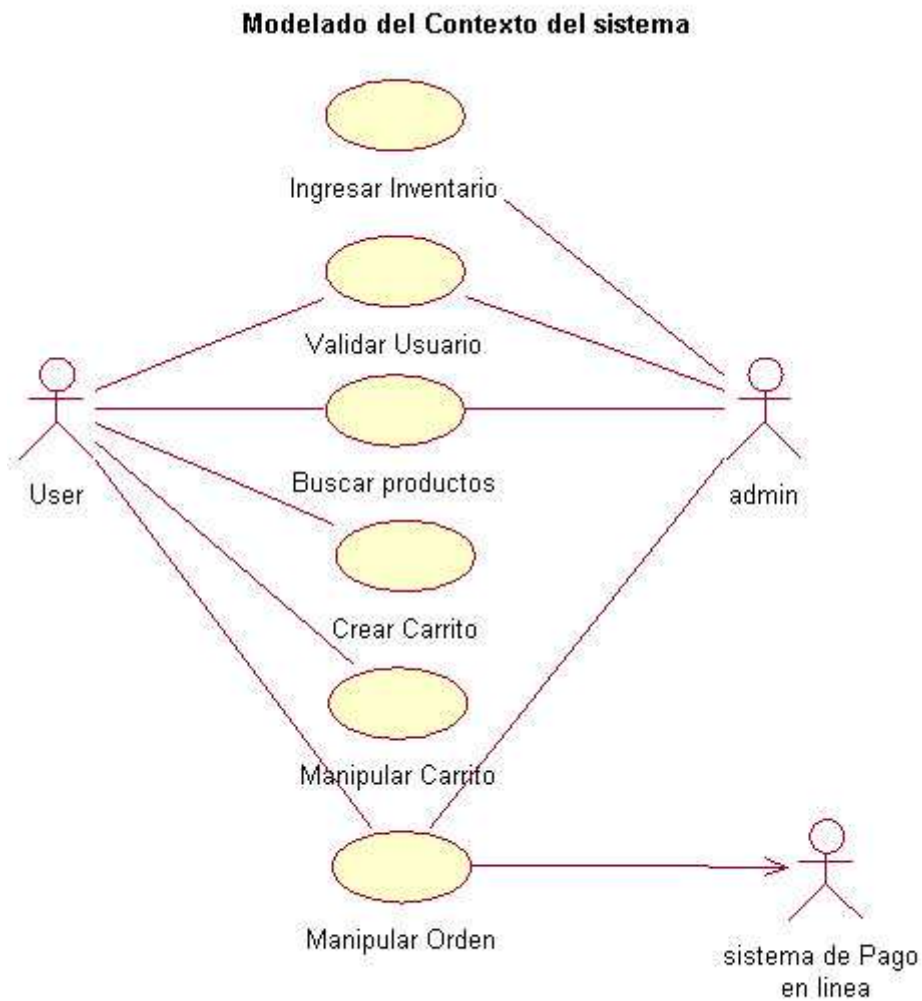


Figura 5.1. Diagrama de Casos de Uso
Fuente: Autor

Descripción de los principales casos de Uso

Caso de Uso	Función
Ingreso de Inventario.	El administrador ingresa toda la información sobre productos del sistema.
Validar usuario	El sistema valida al usuario por medio de un nombre de usuario y contraseña y si es un usuario de rol de administrador ingresa al sistema de administración, en caso que sea un cliente ingresa a su cuenta donde podrá acceder a realizar ordenes de productos en línea
Buscar Productos	El usuario ya este o no ingresado en el sistema puede buscar productos.
Crear Carrito	Un usuario puede crear un carrito de compras, el cual es un objeto en sesión que se le puede ir añadiendo o quitando productos.
Manipular Carrito	El usuario puede añadir , quitar productos y modificar las cantidades de los productos contenidos por el carrito de compras.
Registrar Ordenes	Una vez que el usuario registrado haya terminado de escoger los productos deseados y añadidos al carrito de compras , se puede registrar la orden .
Revisar ordenes	El usuario puede revisar sus ordenes ya sean las que están pendientes como las que se encuentra en una lista futuras compras. El administrador del sistema en cambio puede revisar todas las ordenes del sistema.
Administrar Ordenes	El administrados del sistema puede cambiar los datos de cualquier orden. En cambio el usuario solo puede cambiar el estado de sus ordenes.

Descripción de las Clases

Clase	Descripción
Product	Es la clase que contiene a los productos o servicios que se van encontrar en el catálogo Virtual
User	Clase que contiene a los usuarios del sistema y sus propiedades
Rol	Clase la cual permite validar a los usuarios del sistema
Category	Clase que nos permite categorizar los productos del sistema
Order	Clase que va contener la orden o pedido realizadas por los usuarios
Status	Clase que nos permite validar la orden, identificando el estado.
Detail	Clase que contiene el detalle de la orden, como la cantidad.
CreditCard	Clase que contiene las distintas tarjetas de Crédito
UserCard	Clase que contiene las tarjetas de crédito de los Usuarios
Province	Clase que contiene las provincias, esta clase nos sirve cuando ingresamos datos del usuario
Country	Clase que contiene los Países, esta clase nos sirve cuando ingresamos datos del usuario
City	Clase que contiene las Ciudades, esta clase nos sirve cuando ingresamos datos del usuario
Promotion	Clase la cual contiene distintos productos que pueden estar en promoción

Descripción de las Clases de los formularios

Las clases de los formularios son clases que se crean al momento de crear formularios para ingreso de datos en las clases del modelo. Por ejemplo al momento de crear el formulario para registrar un nuevo usuario se crea una clase UserForm la cual tiene la misma cantidad de propiedades que clase del modelo, pero esta clase representa los datos del formulario. Y esta tiene como objetivo

principalmente para cargar los datos ingresados por el usuario al objeto del modelo.

5.2. Consideraciones Tecnológicas

5.2.1. Arquitectura

El Sistema de Shopping Cart utiliza una arquitectura de múltiples capas utilizando el patrón de MVC explicado anteriormente en el punto 4.2.2

Una arquitectura de múltiples capas esencialmente se compone de las siguientes partes:

- **Interfaz de Usuario**, que se encarga de manejar la interacción del usuario con la aplicación. Esta interfaz es simplemente un navegador de Internet.
- **Lógica de Presentación**, que define qué y cómo se muestra la información al usuario.
- **Lógica de Negocios**, que define el marco y las reglas del negocio a través de la interacción con los datos de la aplicación.
- **Servicios de Infraestructura**, los cuales proveen la funcionalidad adicional requerida por la aplicación y sus componentes.
- **Capa de Datos**, donde residen la información.

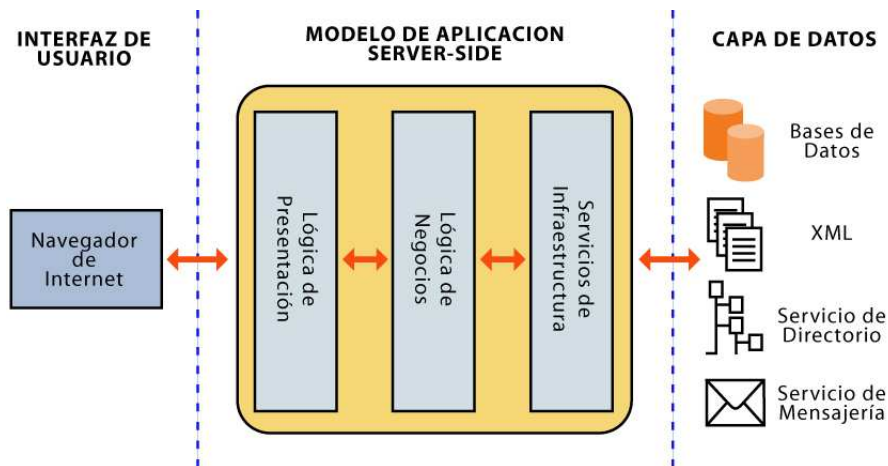


Figura 1. Arquitectura de múltiples capas

Este tipo de arquitectura posibilita enfocarse en cada parte del sistema de manera independiente de las demás, al mismo tiempo que permite integrarlas luego de manera modular y simple. Una arquitectura de múltiples capas nos permite además:

- **Independencia de la presentación con la lógica del negocio**

Se puede modificar de manera fácil y rápida la forma como se muestra la aplicación al usuario final (por ejemplo, hacer un rediseño del sitio Web) sin que esto afecte la lógica de negocios y viceversa.

- **Independencia de los datos**

La persistencia de datos se puede manejar de forma transparente para las distintas partes del sistema, lo cual permite mucha flexibilidad a la hora de escoger una plataforma de base de datos (RDBMS). La lógica de negocios no se ve afectada por cambios en el repositorio de datos ni viceversa.

5.2.2. Fácil escalabilidad

Si se necesita mayor nivel de servicio debido a un crecimiento en la demanda se puede escalar de manera rápida y transparente, es decir, no se requieren cambios de programación.

Adicionalmente cada parte del sistema esta dividida en módulos, lo cual permite enfocarse en la lógica de negocios de cada una de las partes del sistema de manera directa y objetiva.

5.2.3. Plataforma

Para el desarrollo e implementación del Sistema Shopping Cart se ha seleccionado la plataforma J2EE™, la cual provee un estándar simple y unificado para aplicaciones distribuidas de múltiples capas en Internet, a través de un modelo basado en componentes.

La Plataforma J2EE™ define un estándar para las aplicaciones empresariales de múltiples capas. J2EE™ simplifica las aplicaciones empresariales basándolas en componentes modulares y estandarizados, proveyendo un conjunto completo de servicios a estos componentes, y manejando muchas de las funciones de la aplicación de forma automática, sin necesidad de una programación compleja.

5.2.4. Tecnologías

El Sistema Shopping Cart usa un conjunto de tecnologías que proveen los mecanismos necesarios para construir aplicaciones distribuidas de múltiples capas que funcionen en Internet. Estas tecnologías son las siguientes:

- Hypertext Transfer Protocol (HTTP)
- Extensible HyperText Markup Language (XHTML)
- Cascading Style Sheets (CSS)
- JavaScript - DHTML
- Extensible Markup Language (XML)
- Extensible Stylesheet Language (XSL)
- Java Servlet
- JavaServer Pages (JSP)
- JavaBeans
- JSP Tag Extensions
- Java Data Base Connectivity (JDBC)
- JavaMail
- Java 2 Enterprise Edition APIs

Además de las tecnologías antes mencionadas el sistema esta construido en base a los siguientes esquemas de trabajo (frameworks):

- Jakarta Struts, un esquema de control Modelo-Vista-Controlador (MVC).
- Hibernate, un esquema que provee servicios de persistencia de datos transparente.

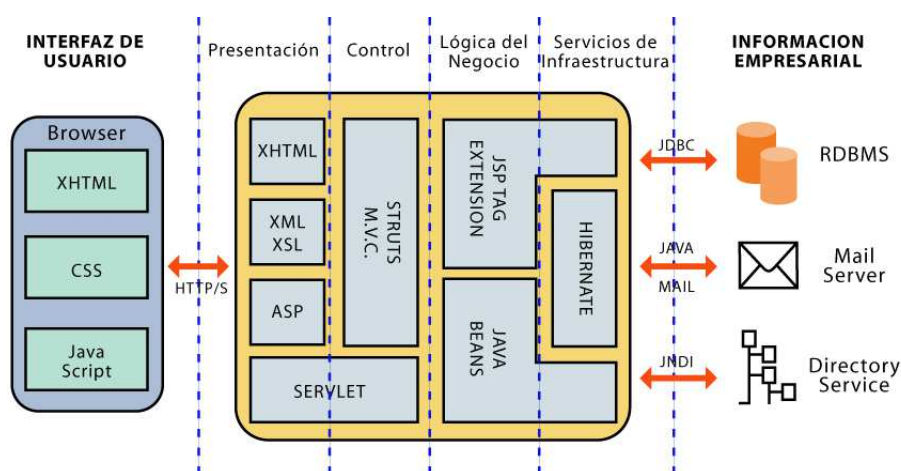


Figura 2. Estructura de colaboración de las tecnologías de desarrollo

5.2.5. Aplicaciones y Herramientas Complementarias

El sistema se complementa con poderosas aplicaciones y herramientas de ejecución, procesamiento y almacenamiento de información. Estas aplicaciones se comunican transmitiendo y procesando información de una hacia las otras para brindar un entorno estable y seguro que garantiza el correcto funcionamiento del Sistema de Shopping Cart.

Las aplicaciones y herramientas complementarias que utiliza el sistema son las siguientes:

- Navegador de Internet, el cual es utilizado para interactuar y mostrar la información a los usuarios del sistema.
- Servidor Web, encargado de recibir las peticiones generadas por los usuarios y retornar la respuesta adecuada a cada uno de ellos.
- Entorno de ejecución Java, que provee las librerías y otros componentes necesarios para ejecutar aplicaciones escritas en el lenguaje de programación Java.
- Contenedor Web, es un entorno de ejecución para manejar componentes de aplicaciones y a su vez proveer acceso a las librerías del entorno Java.
- Servidor de base de datos, en donde se almacena la información recopilada y generada por el sistema.

5.2.6. Entorno de Ejecución y Especificaciones

El Sistema Shopping Cart está construido utilizando las siguientes especificaciones:

Aplicaciones

Componente	Especificación
Sistema Operativo	Microsoft Windows 2000 Server
Navegador de Internet	Microsoft Internet Explorer 5.5
Servidor Web	Microsoft Information Server 5.5
Entorno Java	Sun Java RunTime Enviroment 1.4.2
Contenedor Web	Apache Tomcat 4.1.X
Servidor de Base de Datos	MYSQL

Tecnologías

Componente	Especificación
HTTP	Versión 1.0/1.1
XHTML	HTML 4.01 / XHTML 1.0
CSS	CSS Level 2
JavaScript	Versión 1.2
JavaServlet	Especificación 2.3
JSP	Especificación 1.2
JDBC	Especificación 2.X
J2EE	Especificación 1.3.1

Frameworks y Herramientas de Desarrollo

Componente	Especificación
Struts	Versión 1.1

Hibernate	Versión 2.0
Apache Ant	Versión 1.5.3
XDoclet	Versión 1.2

5.2.7. Personal

Para el desarrollo de la aplicación de shopping cart se necesita emplear mínimo 3 personas para realizar las siguientes funciones :

Gerente de Proyecto

Aplicando el proceso unificado de Rational , se analizará y se describirá la funcionalidad del sistema.

Programador

Se programa en Java para crear los objetos y las acciones es decir las funciones que contienen la lógica del negocio .

Diseñador Gráfico

Como es un sistema Web se debe diseñar y diagramar una interface navegacional intuitiva y sencilla y a su vez liviana para web, es decir que las imágenes que se utilizarán serán optimizadas para que el sitio Web se cargue de una manera eficaz.

5.2.8. Costo de Desarrollo

A continuación se analiza el costo de desarrollo del sistema de Shopping Cart y se establece un precio de comercialización del mismo. En general, los costos de desarrollo del sistema se basan en el costo del recurso humano y hardware empleados, ya que el software empleado es de libre distribución. Por tanto, el tiempo de desarrollo es la variable con mayor sensibilidad en el análisis de costos.

Los recursos empleados en el presente proyecto son:

Computadora de escritorio P4 valorada en USD 1000 con vida útil estimada de 2 años.

un desarrollador que conforman el recurso humano del proyecto cuyos honorarios estimados son de USD 40 dólares diarios.

El tiempo global de desarrollo del sistema fue de 120 días. El costo de uso de las computadoras empleadas se basa en su depreciación diaria.

	Valor (días)	Vida útil	Depreciación diaria
Computadora de Escritorio	1000	730	1.369863014

Tabla 3.1. Depreciación diaria de equipos
FUENTE: Autores

Recurso humano	Hora
Tiempo de análisis	60
Tiempo de investigación	400
Tiempo de desarrollo	520

Tabla 3.2. Detalle recurso humano
FUENTE: Autores

La siguiente tabla ilustra los costos de manera detallada y valor comercial estimado.

Recurso	Costo diario
Recurso humano	40

Computadora de Escritorio	1.37
COSTO TOTAL DIARIO	41.37
COSTO a 120 días	5430,4
Costos Operativos	466
COSTO TOTAL	5430,4
UTILIDAD 15%	814,56
VALOR COMERCIAL	6244,96

Tabla 3.3. Costos y precio de Control CCC
FUENTE: Autores

El shopping cart es un sistema demostrativo de la investigación razón por la cual no se puede establecer un esquema de recuperación de la inversión tomando en cuenta un volumen potencial de ventas del producto.

5.2.9. Software

Cliente:

Cualquier computador con al menos 64 de RAM y 50MB de espacio libre en disco, con cualquier sistema operativo con cualquier navegador de internet que soporte Javascript estándar

Servidor:

- No Importaría la plataforma ya que se desarrollaría en Java
- Web server con soporte a Servlets y JSP
- Base de Datos MySQL
- Servidor de Aplicaciones, el cual se encarga de procesar los requerimientos del cliente, ya sean estos consultas a bases de

datos, reportes, u otro servicio de comercio electrónico. El servidor de aplicaciones debe soportar las especificaciones Servlets 2.2 y JSP 1.1 de Sun Microsystems. Este requerimiento es opcional para sitios web estáticos o informativos.

- Servidor de Base de Datos, que es el repositorio utilizado para almacenar información.
- Driver JDBC tipo 4 para la Base de datos seleccionada. Este driver permite la interacción entre la base de datos y la aplicación. El driver es necesario si el website interactuará con una base de datos.
- Certificado digital de seguridad, que permite certificar la veracidad de la información transmitida desde y hacia la aplicación. Este componente es necesario si la empresa desea implementar servicios de comercio electrónico como pagos en línea, en cualquier otro caso es opcional.

5.2.10. Hardware

Cliente:

PC con al menos 256MB de RAM y 50MB de espacio libre en disco.

Servidor:

- Servidor con al menos 256MB de RAM y 50MB de espacio libre en disco.
- Infraestructura de red de área amplia WAN.

5.2.11. Características de los usuario del sistema.

Manejo de cualquier sistema operativo, manejo básico de cualquier navegador de Internet.

5.3. Análisis y Diseño

Para el Análisis y diseño se realizaron los siguientes diagramas.

Modelo de Datos:

Diagrama de Clases

Modelo de Apoyo:

Modelo Entidad Relación
Modelo físico

Modelo de Análisis:

Diagrama de Casos de Uso
Diagrama de Secuencia
Diagrama de Actividades
Diagrama de Componentes
Diagrama de Estados
Diagrama de Despliegue

5.3.1. Diagrama De Clases

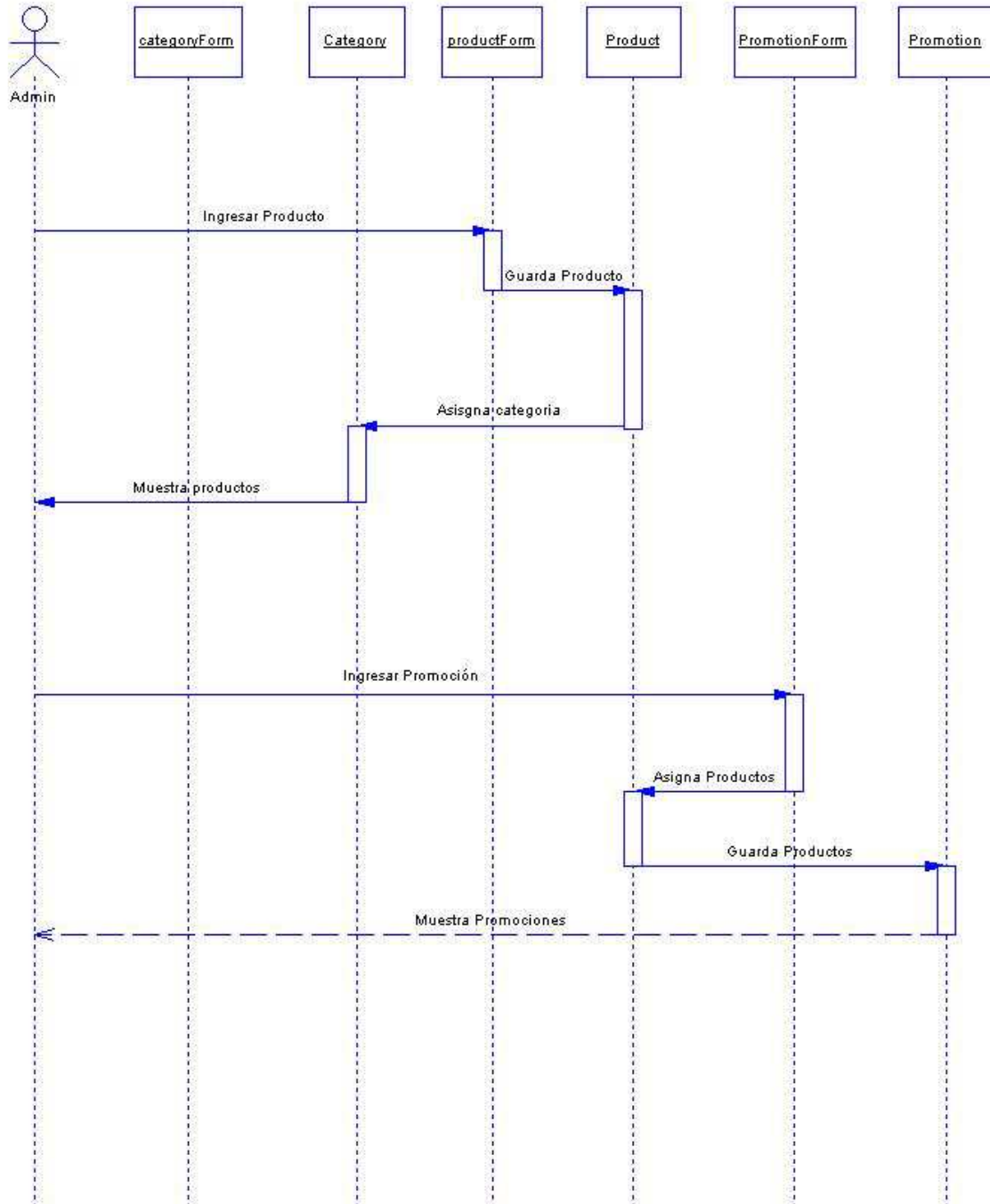
5.3.2. Modelo Entidad Relación

5.3.3. Modelo Físico de Datos

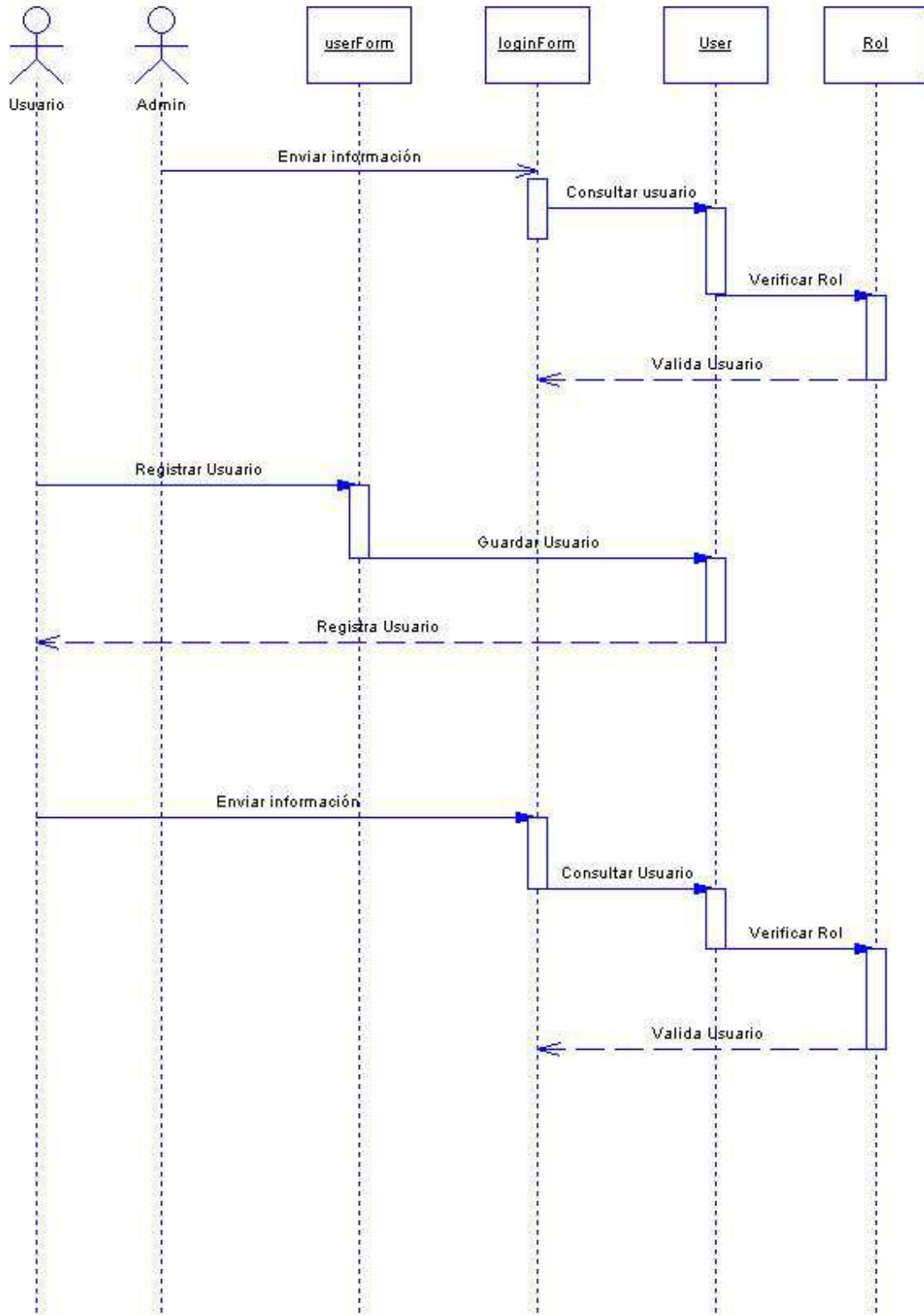
5.3.4. Diagrama De Casos de Uso

5.3.5. Diagramas De Secuencia

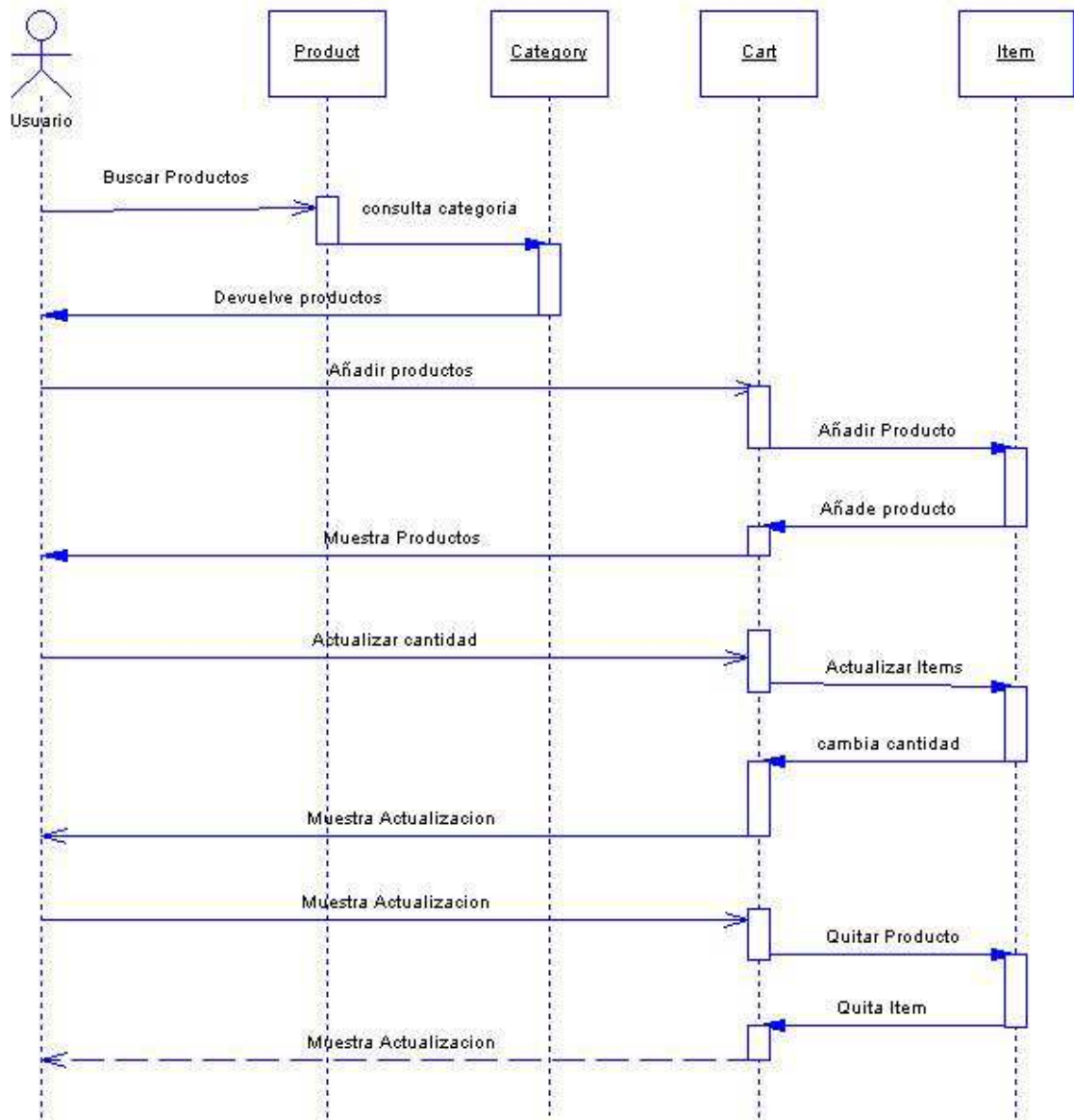
Ingreso de Inventario

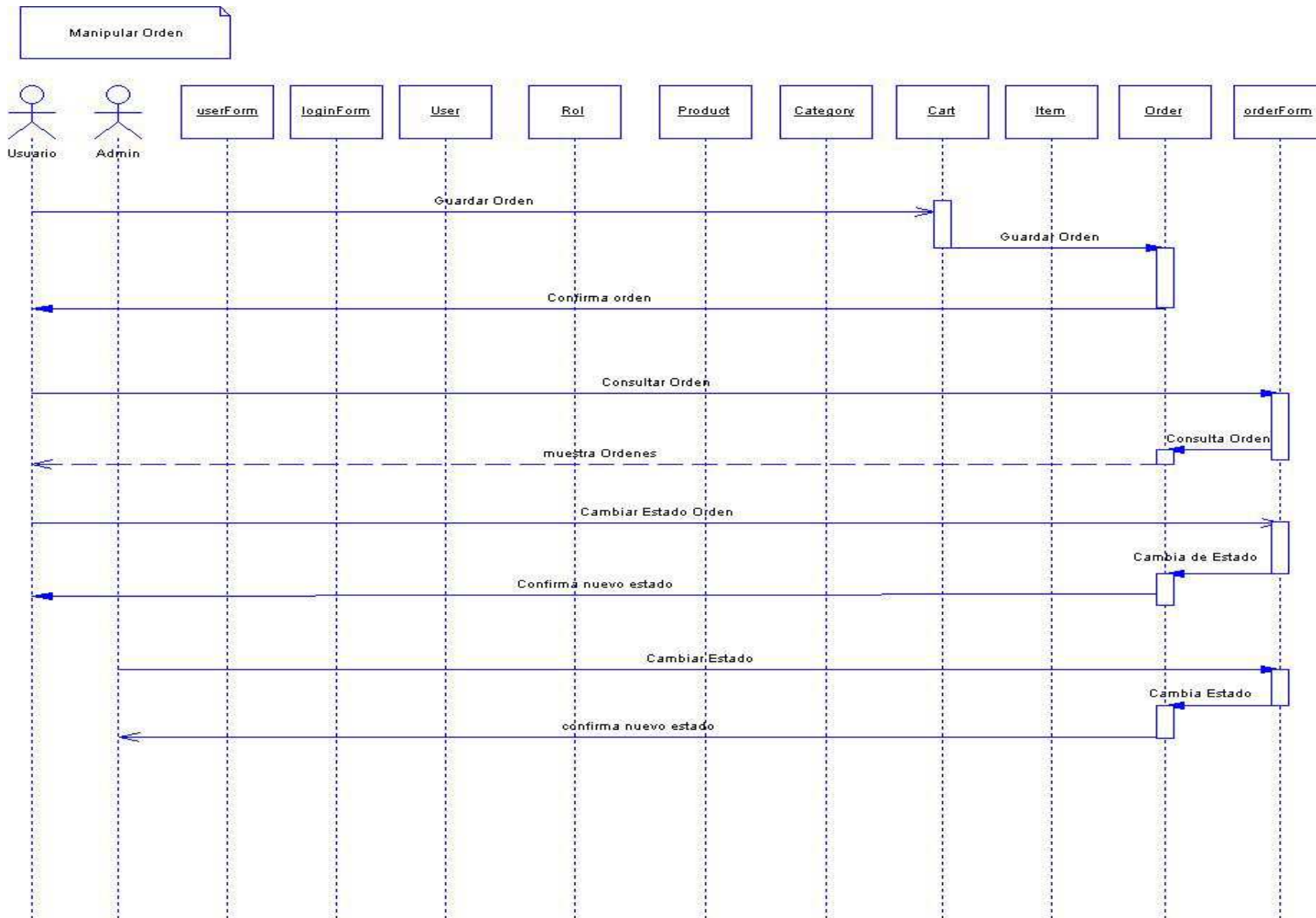


Escenario Validar Usuario



Escenario de Manipular Carrito





5.3.6. Diagrama De Actividades

5.3.7. Diagrama De Estados

5.3.8. Diagrama De Componentes

5.3.9. Diagrama De Despliegue

5.4. Implementación

Para la implementación del sistema se siguieron los siguientes pasos:

- Instalación y configuración de la aplicación en el servidor
- Creación de las clases del modelo
- Desarrollo con Hibernate
- Implementación de la lógica del negocio

Instalación y configuración de la aplicación en el servidor

Para la instalación y configuración de la aplicación en el servidor se deben realizar los siguientes pasos:

- Configuración de la aplicación para ejecutar en el servidor web
- Configuración de Hibernate
- Configuración de Struts

Lo cual es explicado a detalle en el Manual de instalación en el Anexo B

Creación de las clases del modelo

Una vez configurada la aplicación se procede a la creación de las clases del modelo. Las clases se crean a partir del diagrama de clases realizado previamente en el punto 5.3.1.

Desarrollo con Hibernate

Para el desarrollo del shopping cart se desarrollo de la siguiente forma:

Una vez implementadas las clases del modelo de la aplicación, se procede al "mapeo" de dichas clases. El mapeo consiste en relacionar las clases del modelo con las tablas de la base de datos de la aplicación (ver diagrama entidad-relación).

El mapeo de las clases con hibernate depende de las relaciones entre los objetos del sistema.

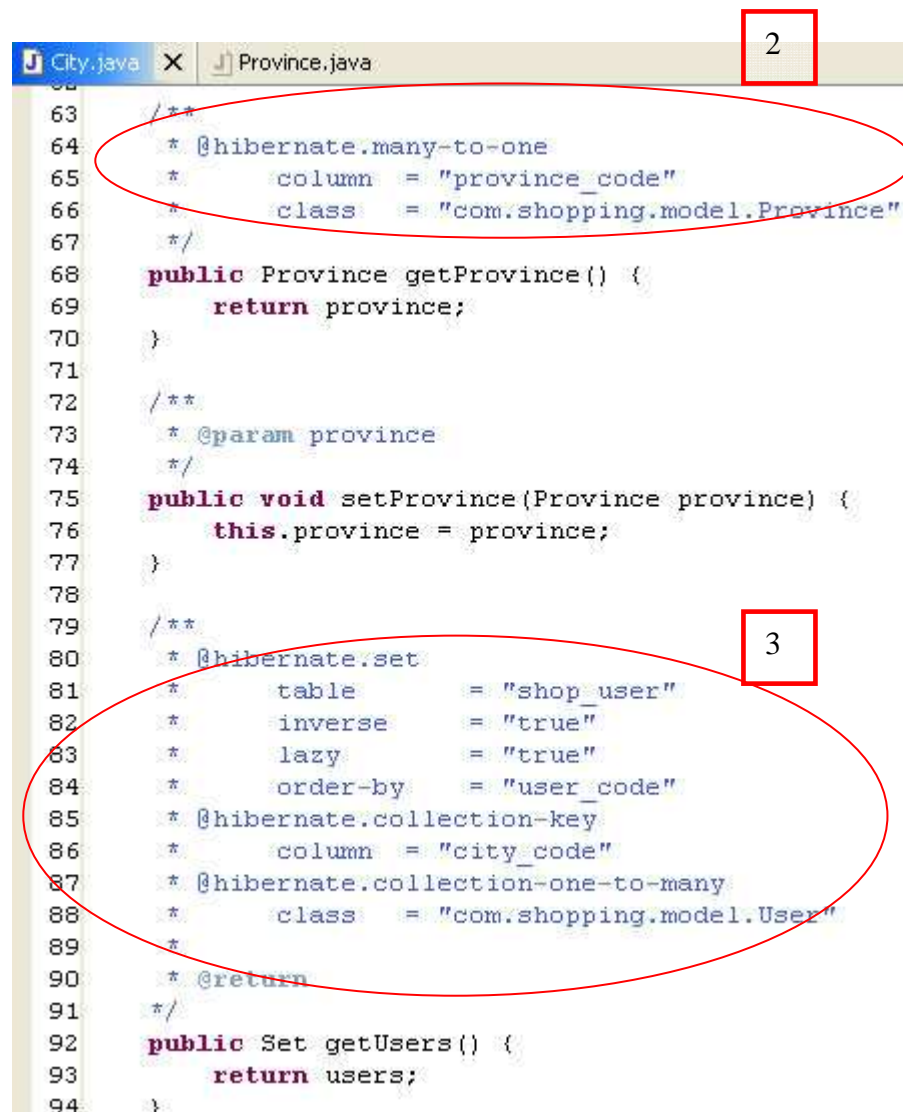
Por ejemplo:

Con la clase ciudad la cual se relaciona con la clase provincia, lo primero que se hace es definir en el objeto con que tabla de la base de datos se va hacer persistencia.

```
City.java X Province.java
7 package com.shopping.model;
8 import java.util.Set;
9
10 /**
11  * @author jjleon
12  * To change the template for this generated type comment go to
13  * Window>Preferences>Java>Code Generation>Code and Co
14  * @hibernate.class table = "shop_city"
15  */
16 public class City {
17
18     private Long code;
19     private String name;
20
21     private Province province;
22     private Set users;
23
24     /**
25     * @hibernate.id
26     *     column="city_code"
27     *     type="long"
28     *     generator-class="com.shopping.hibernate.SeqTableGenerator"
29     * @hibernate.generator-param name="table" value="shop_sequence"
30     * @hibernate.generator-param name="sequence" value="city"
31     * @hibernate.generator-param name="column" value="next_val"
32     */
33     public Long getCode() {
34         return code;
35     }
}
```

Luego se mapea las funciones miembro con cada uno de los campos de la base de datos. El mapeo se hace dependiendo de las relaciones entre los objetos

En el caso a continuación podemos ver que la clase ciudad tiene una relación de muchos a uno con la clase provincia es decir muchas ciudades pueden pertenecer a una sola provincia y este dato es almacenado en el campo province_code de la base de datos.



```
City.java X Province.java 2
63 /**
64  * @hibernate.many-to-one
65  *     column = "province_code"
66  *     class  = "com.shopping.model.Province"
67  */
68 public Province getProvince() {
69     return province;
70 }
71
72 /**
73  * @param province
74  */
75 public void setProvince(Province province) {
76     this.province = province;
77 }
78
79 /**
80  * @hibernate.set
81  *     table      = "shop_user"
82  *     inverse    = "true"
83  *     lazy       = "true"
84  *     order-by  = "user_code"
85  * @hibernate.collection-key
86  *     column    = "city_code"
87  * @hibernate.collection-one-to-many
88  *     class     = "com.shopping.model.User"
89  *
90  * @return
91  */
92 public Set getUsers() {
93     return users;
94 }
```

Por último en el 3 ejemplo podemos ver una relación entre el objeto ciudad y usuario donde es una relación uno a muchos para este tipo de relaciones usamos una propiedad set.

Una vez mapeadas todas las clases del modelo se procede a generar archivos xml de mapeo lo cual consta de 2 pasos :

- Primero se compila con la herramienta Ant y se verifica que no haya errores en el mapeo .
- Una vez compiladas las clases la herramienta ant llama a la librería de Xdoclet la cual genera los archivos de mapeo.

Implementación de la lógica del negocio

Consiste en la programación de las distintas funciones y clases usando las librerías que sean necesarias implementar para el correcto funcionamiento del sistema.

Todas estas funciones o acciones del sistema se rigen por el patrón MVC (visto previamente en el punto 4.2.2) , implementado a través de Struts y Tiles.

Struts tiene un archivo de configuración **struts-config.xml**, en el cuál se describen tanto todos los formularios de la aplicación como las acciones o funciones implementadas en el sistema. De esta forma se controla el flujo de los procesos de la aplicación visto previamente en el punto tiles-defs.xml.

De igual forma Tiles tiene un archivo de configuración **tiles-defs.xml**, en el se cual se configuran las paginas Jsp para presentación de la aplicación. Visto previamente en el punto 4.2.4

Por ejemplo:



```
<!-- ===== -->
<!-- Administración de Usuarios -->
<!-- ===== -->

<action path="/admin/listUsers"
        type="com.shopping.mvc.admin.ListUsersAction">
    <forward name="success" path="admin.listUsers" redirect="false" />
</action>

<action path="/admin/editUser"
        type="com.shopping.mvc.admin.UserAction"
        name="userForm"
        input="admin.listProvinces"
        scope="request"
        validate="false">
    <forward name="form" path="admin.userForm" redirect="false" />
    <forward name="list" path="/admin/listUsers.do" redirect="false" />
</action>

<action path="/admin/saveUser"
        type="com.shopping.mvc.admin.UserAction"
        name="userForm"
        input="admin.userForm"
        scope="request"
        validate="true">
    <forward name="success" path="/admin/listUsers.do" redirect="false" />
</action>
```

Tenemos la acción que se va encargar de guardar un nuevo usuario del sistema. Esta acción va ser configurada en el archivo xml de configuración del Struts y en el archivo xml de configuración del Tiles.

5.5. Pruebas

El objetivo de la prueba es descubrir errores en la aplicación. La prueba demuestra hasta qué punto la aplicación funciona de acuerdo con las especificaciones y requerimientos del negocio. Además, los datos obtenidos a medida que se lleva a cabo la prueba proporcionan una buena indicación de la fiabilidad del software.

Las pruebas efectuadas en el Sistema de Shopping Cart se dividieron en 3 módulos, pruebas globales al sistema, el del catálogo de productos y la administración del sistema.

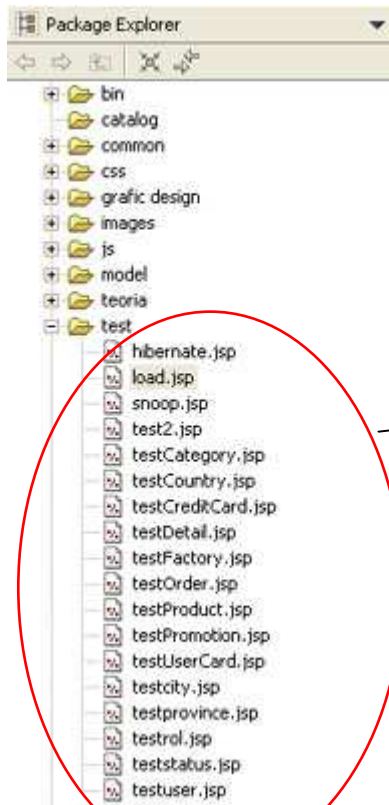
Módulo de catálogo de productos	
Proceso	Características a probarse
Verificación de enlaces	Se verifica los vínculos de todas las secciones del catálogo
Verificación de gráficos	Se verifica los vínculos de los gráficos del catálogo
Verificación del funcionamiento del carrito de compras.	<ul style="list-style-type: none"> • Se verifica que los mensajes de error que puedan producirse usando el carrito de compras • Confirmación de los productos que se quieren registrar en la orden
	<ul style="list-style-type: none"> • Validación de datos referentes a la orden. • Verificación de que los datos de la orden se hayan guardado en la base de datos.
Verificación de presentación de información.	Se verifica que los textos se presenten estandarizados.
Verificación de registro de usuarios	Se verifica tanto la validación de usuarios del sistema como usuarios que se registran en el sistema

Módulo de Administración	
Proceso	Características a probarse
Verificación de enlaces	Se verifica los vínculos de todas las secciones de la interfase de administración .
Verificación del proceso de la información	<ul style="list-style-type: none"> • Se verifica el despliegue de todos los datos de las tablas de la administración • Se verifica que los datos se puedan actualizar, grabar y borrar.
Verificación de presentación información.	Se verifica que los textos se presenten estandarizados.

La denominada *pruebas globales del sistema*, consiste en una serie de pruebas diferentes, tiene el propósito de verificar que se hayan integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas.

Pruebas Globales del sistema	
Proceso	Características a probarse
Prueba de Unidad	La prueba de unidad centra el proceso de verificación en cada módulo. En el caso del presente proyecto se efectuaron pruebas en cada componente (JavaBean) y objeto desarrollado, mediante una página jsp, en la cual se comprobó el correcto funcionamiento de cada método en cada componente y objeto implementado.

Prueba de Integración	Una vez probados todos los módulos individualmente, se prueba la aplicación en conjunto con la interacción de todos los módulos. En este punto se procedió con las pruebas de funcionamiento de los componentes interactuando entre sí en el proceso de registro de órdenes. Además se verificó que los cambios efectuados a través del módulo de administración sean reflejados en las interfaces de usuario correspondientes.
Prueba de recuperación	Fuerza al fallo del software de muchas formas y verifica que la recuperación se lleve a cabo apropiadamente.
Prueba de seguridad	Para verificar que los mecanismos de protección incorporados en el sistema lo protejan, de hecho, de la penetración de individuos no autorizados.
Prueba de resistencia	Diseñada para enfrentar al programa a situaciones anormales. Se ejecuta al sistema de forma que demande recursos en cantidad, frecuencia o volúmenes poco comunes. Se intenta bajar el sistema.



Páginas Jsp de prueba

PRUEBA DE UNIDAD

6. Capítulo 6: Conclusiones y Recomendaciones

6.1. Conclusiones

A mi juicio, el Proceso Unificado constituye el marco de trabajo de integración de técnicas y procesos de ingeniería de software que implementan las mejores prácticas de la industria actual.

De esta forma nos provee de una guía completa para optimizar las actividades en las distintas etapas de desarrollo de software.

El Proceso Unificado es un marco de trabajo genérico que puede ser aplicado a distintos tipos de sistemas sin importar su tamaño o área de aplicación.

Puesto que el Proceso Unificado es iterativo e incremental nos permite detectar puntos críticos y defectos en las distintas fases del proyecto, permitiendo corregirlos en etapas tempranas dado que desde el primer ciclo de desarrollo se crea un prototipo del proyecto el cual se va retroalimentado con el aporte del cliente.

En el desarrollo de aplicaciones web a pesar de la aplicación de un proceso y análisis preestablecido al momento de desarrollar se pierde orden y esto ocasiona demoras e incremento en el costo de producción, generalmente el uso del patrón (MVC) a través de Struts brinda una nueva herramienta más ágil y eficiente al momento de desarrollar además de apoyar al trabajo en equipo, ya que el sistema de Shopping Cart está basado en la arquitectura n-capas. Este modelo permite superar muchos de los inconvenientes de otros sistemas con diferente arquitectura.

Hibernate permite a los programadores centrarse sobre el esquema de las clases del sistema a desarrollarse y no sobre el esquema de la base de datos, de esta forma separa de manera clara y definida, la lógica de negocio de la aplicación con el diseño de la base de datos .

La plataforma en que se basó el sistema fue J2EE la cual es reconocida por su robustez y fiabilidad para soportar aplicaciones empresariales.

El diseño del Shopping Cart fue basado en la arquitectura n-capas la cual permite que el sistema supere la vulnerabilidad mencionada en las conclusiones. Sin embargo, para que dicho diseño funcione, es necesario que el sistema se instale en un ambiente n-capas. Ambiente que debe contar con la infraestructura adecuada que balance la carga del sistema entre datos, aplicación y cliente; además debe proveer servicios de recuperación según las normas pertinentes. Tal infraestructura, generalmente ofrece el proveedor de servicios de hospedaje para aplicaciones en línea, evitando así que el dueño de la aplicación tenga que efectuar esta inversión.

6.2. Recomendaciones

Se recomienda el uso del Proceso Unificado de Rational el cual es un proceso moderno que facilita todos los métodos para desarrollar sistemas y fomenta el trabajo en equipo a través de la asignación de tareas y responsabilidades.

Por lo general, al hablar de la aplicación de nuevas tecnologías, siempre se asociaba la idea de importantes inversiones económicas. Esta situación ha cambiado considerablemente en los últimos años, con la aparición del software de libre distribución. Es por esto, que

en el mercado existen numerosas herramientas y especificaciones sin costo de licenciamiento y, además, con un nivel de calidad en muchos casos superior a las herramientas comerciales. Razón por la cual creo que es conveniente el uso de estas herramientas y su explotación tanto a nivel comercial como académico.

Académicamente debería fomentarse el uso de este tipo de software, no solo a nivel de profesionales informáticos, sino también a nivel general. Por ejemplo, en centros educativos y empresas, en lugar de hacer fuertes inversiones en licencias de uso de sistemas operativos como Microsoft Windows, deberían utilizar sistemas operativos de libre distribución. De esta manera, se crearía una cultura de uso que incluso ampliaría la oferta de software libre y, consecuentemente, provocaría una caída en los precios del software comercial.

Anexos

Anexo A



Manual de Usuario

Sistema de Shopping Cart

El Sistema de Shopping Cart es una herramienta de apoyo tecnológico que permite mantener una amplia base de datos de productos, clasificados por distintas categorías. El sistema permite crear órdenes de compra para los usuarios registrados en el sistema.

El objetivo del presente manual es dar a conocer a los usuarios finales las características y funcionamiento del Sistema de Shopping Cart, Este documento supone que el usuario posee nociones básicas sobre navegación en Internet.

Convenio Iconos

-  Indicaciones Paso a Paso
-
-  Información de interés, notas.
-

1. Ingreso al Sistema



Figura 1.
Forma para validación de usuario

Para ingresar a la aplicación utilice cualquier navegador de Internet que tenga instalado. Ingrese a la dirección: <http://localhost/shopping> Aparecerá en pantalla la ventana de ingreso al sistema, si usted es usuario autorizado, entonces debe ingresar su "Usuario" y "Contraseña" y presionar en el botón **Go**.

2. Interfaz de Usuario

A través de la interfaz de usuario se puede ingresar al catálogo de productos y servicios donde se tiene acceso a información sobre productos, y promociones ofertadas en el catálogo virtual.

El usuario tiene la facultad de añadir, modificar o eliminar productos en su carrito de compras, para una vez seleccionados los productos deseados se cree una orden.

2.1. Comenzar a utilizar el sistema

Lo más importante en una tienda virtual es la facilidad de encontrar los productos, por este motivo el usuario siempre tiene un buscador de productos para que de esta forma los encuentre rápidamente.



Además el sistema consta con una opción de búsqueda avanzada, la cual recibe parámetros y permite realizar búsquedas específicas.



Los parámetros de consulta son opcionales y pueden contener datos parciales. Es decir, si en el campo 'Producto' se ingresa 'Cel', se buscarán todos los empleados cuyo nombre o apellido contenga la cadena 'Cel', como Celular, Celurares, etc.

3. Interfaz de Administrador

La interfaz de administración se presenta como un entorno Web al cual el administrador accede mediante un nombre de usuario y una contraseña. El administrador tiene la facultad de añadir, modificar o eliminar información y parámetros de la aplicación.

La información del sistema es almacenada en una base de datos centralizada, desde donde se la muestra a los usuarios. Gracias a este manejo de la información, cualquier cambio efectuado en el sitio, a través de la interfaz de administración, es automáticamente reflejado en la interfaz de usuario.

Es importante tomar en cuenta que un cambio en los parámetros de la aplicación puede afectar el correcto funcionamiento del mismo.

Como se mencionó anteriormente, las operaciones básicas que se efectúan a través de la interfaz de administrador son las de ingreso, modificación y eliminación de información. El proceso a seguir es el mismo para todos los casos, como se muestra a continuación.

3.1. Añadir Registro



1. Para añadir un registro, haga clic sobre el link


 [Añadir Nuevo](#) .

2. En la pantalla que aparece a continuación, ingrese la información correspondiente.

3. Haga clic sobre el botón *Actualizar*.

3.2. Modificar Registro



1. Para modificar un registro, selecciónelo de la lista haciendo clic sobre el nombre del registro que desea modificar o sobre el icono  correspondiente.

2. La pantalla que aparece a continuación, presenta la información almacenada en el sistema del registro seleccionado. Aquí puede editar los datos que correspondan.

3. Haga clic sobre el botón *Actualizar*.

3.3. Eliminar Registro




1. Para eliminar un registro de una lista, haga clic sobre el ícono  correspondiente al registro que desea eliminar.



Figura 4. Lista de registros

4. Información general

4.1. ¿Cómo saber si un campo es obligatorio?



Los campos con etiqueta ***formato azul** son obligatorios.
Los campos con etiqueta **formato gris** son opcionales.

4.2. ¿Cómo ingresar fechas?



Para ingresar una fecha en el sistema, haga clic sobre el botón que aparece junto al campo correspondiente.

***Fecha de Nacimiento**

2004-04-07 



año 2004 mes Abril

Dom	Lun	Mar	Mie	Jue	Vie	Sab
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

Figura5. Calendario

A continuación, se abre una ventana que le permite seleccionar de manera fácil el año, mes y día. Usted puede ingresar la fecha directamente en la casilla del campo Fecha, siguiendo el formato 'aaaa-mm-dd', esto es año, mes y día.

4.3. ¿Cómo subir fotos?



Para ingresar una fecha en el sistema, haga clic sobre el botón que aparece junto al campo correspondiente.

Foto

pet_2.jpg 

A continuación, se abre una ventana con las instrucciones:



1. Seleccionar imagen. Seleccione la imagen que desea dando clic sobre el botón **Examinar** (Browse si su navegador está en inglés).
2. Cargue la imagen. Haga clic sobre el botón **Cargar**. Se reemplazará la imagen anterior, si existe.
3. Después de unos segundos la imagen aparezca bajo el recuadro. Presione **Aceptar** para terminar.

4.4. Ejemplo Actualización de Usuario

 Añadir Nuevo Usuario

		Usuario	Nombre	Tipo de usuario	Tarjetas del usuario	Órdenes	
1.	 	admin	leon pepe	administrador	Tarjetas	Órdenes	juanjoseleo
2.	 	hleon	leon hernan	usuario	Tarjetas	Órdenes	hernanleon
3.	 	user1	system user	usuario	Tarjetas	Órdenes	user@shop
4.	 	user2	nuevo usuario	usuario	Tarjetas	Órdenes	user@whati
5.	 	catalina	Hernandez Cata	usuario	Tarjetas	Órdenes	catu@hotmail
6.	 	fernanda	encalada fernanda	usuario	Tarjetas	Órdenes	fernanda_ei
7.	 	usuario	practica usuario	usuario	Tarjetas	Órdenes	practica@st
8.	 	Ana lu	Vásconez AnaLucía	usuario	Tarjetas	Órdenes	analu83ec@

Figura 2. Menú de administración de usuarios

 **1.** Para actualizar datos de un usuario, selecciónelo de la lista haciendo clic sobre su nombre o sobre el ícono  correspondiente. Si se trata de un empleado nuevo, haga clic sobre el link Añadir Nuevo Usuario.

2. Aparece en pantalla la forma para el ingreso de los **datos personales** del empleado. Ingrese la información y haga clic sobre el botón *Siguiente* para guardar los cambios. Si se trata de un empleado nuevo, en este punto el registro ingresará al sistema y aparecerá disponible el menú de ingreso a las secciones restantes.

3. Para actualizar datos de las tarjetas de un usuario se selecciona en la palabra Tarjeta correspondiente a la fila donde está el nombre del usuario deseado.

4. Para actualizar datos de las órdenes de un usuario se selecciona en la palabra órdenes correspondiente a la fila donde se encuentra el nombre del usuario cuyas ordenes se deseen modificar.

Para los casos de tarjetas y órdenes, al seleccionar una opción se presenta en pantalla dos bloques. En el bloque superior, se muestra la lista de registros ingresados. El bloque inferior contiene la forma para el ingreso de datos.

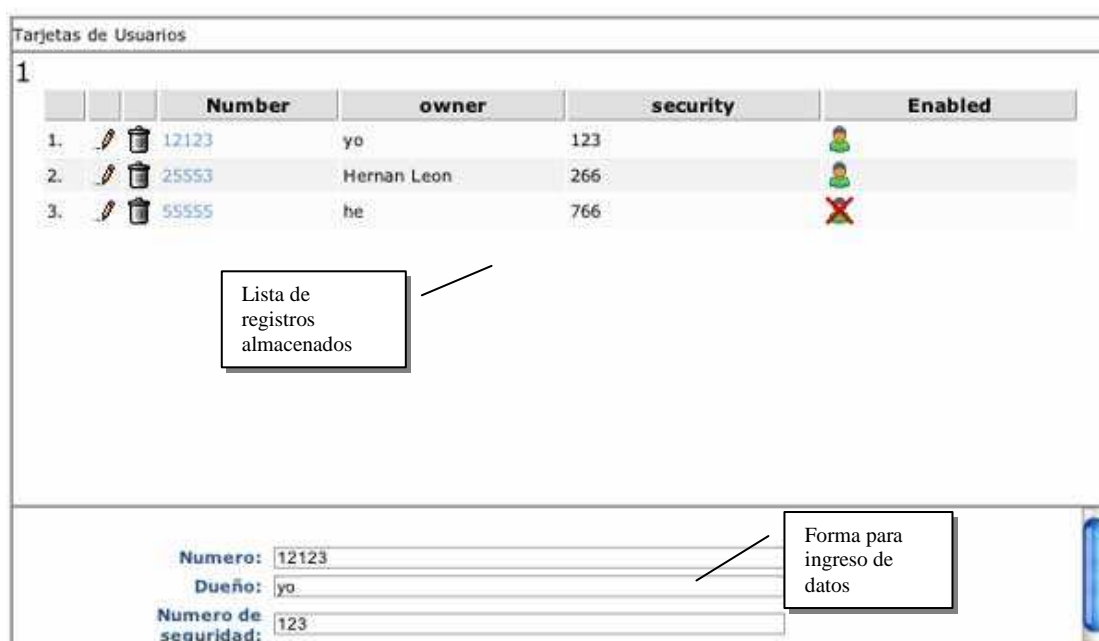




Figura 3. Bloques para el ingreso de datos que pueden contener más de un registro



7.1. Para insertar un nuevo un registro, ingrese los datos en la forma del bloque inferior y haga clic sobre el botón *Guardar*.

7.2. Para modificar un registro, selecciónelo de la lista haciendo clic sobre el nombre del registro que desea modificar o sobre el ícono  correspondiente. La forma del bloque inferior aparece

cargada con los datos del registro seleccionado, actualice la información y haga clic sobre el botón *Guardar*.



7.3. Para eliminar un registro de la lista, busque el registro que desea eliminar y haga clic sobre el icono  correspondiente.

Anexo B

Manual de Instalación

A continuación se describe el proceso de instalación del software necesario para que entre en funcionamiento el sistema desarrollado. El proceso de instalación descrito es aplicable a la plataforma windows. Los procesos de configuración de archivos no varían mucho con relación a otras plataformas.

Convenio ICONOS

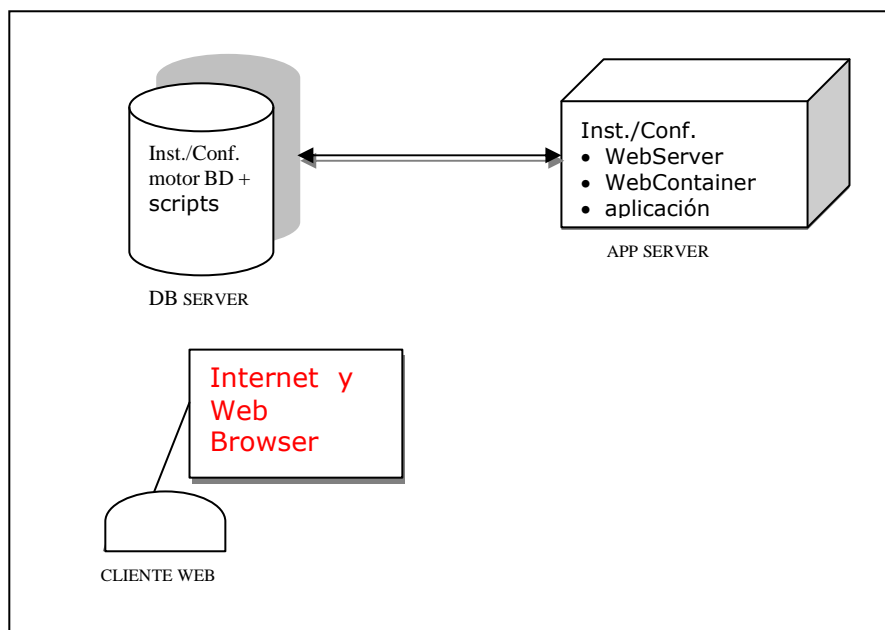
- | | |
|---|--------------------------------|
|  | Indicaciones Paso a Paso |
|  | Información de interés, notas. |

Shopping Cart es una aplicación n-capas basada en el patron MVC, el proceso de instalación se enfoca en las capas lógica y de datos. Los clientes del sistema pueden ser de distintas clases y por ende los procesos de instalación y configuración varían no solo entre distintos dispositivo cliente sino también dependiendo del servicio disponible. Sin embargo, se explica un procedimiento general para brindar una guía referencial en este aspecto.

Procedimiento General de Instalación

La instalación del sistema se consta de tres grandes procesos que pueden instalarse de manera independiente y aleatoria. En la figura 6.1. se describe de manera global estos procesos.

- Instalación y configuración de la base de datos.
- Instalación y configuración de la lógica del negocio.
- Instalación y configuración de clientes web.



Instalación clientes Web

El proceso de instalación de clientes Web consiste en

1. Contratar una conexión a Internet con cualquier ISP.
2. Instalar la conexión configurando módems, red o dispositivos adecuados para el tipo de conexión contratada.
3. Instalar un navegador http en la o las computadoras que tendrán acceso a Internet.
4. Acceder a la dirección Url del sistema ShoppingCart.

Instalación del Servidor Web (Apache)



1. El servidor Apache es un software de libre distribución. Usted puede obtener la última versión de este software en la dirección

<http://www.apache.org/dist/httpd/binaries/win32/>.

Es un archivo autoextraíble, por lo que debe hacer doble clic sobre el archivo bajado. En el caso del ShoppingCart se instaló el archivo apache_2.0.39-win32-x86-no_ssl.msi.

2. Una vez que comienza la instalación nos da la bienvenida y advierte que el programa está protegido por copyright. Haga clic sobre el botón Next.

3. Aparecerá la ventana con las características de la Licencia, que aceptamos pulsando el botón Yes.

4. Presenta una lista de lecturas sugeridas antes de ejecutar Apache en Windows. Haga clic en Next.

5. Aquí se ingresa información acerca de :

- Dominio de red (ej. controlccc.com)
- Nombre del servidor (ej. www.controlccc.com)
- Dirección email del administrador (ej. webmaster@controlccc.com)
- Además, debe escoger si desea proporcionar accesos director para todos los usuarios de la computadora o solo para el usuario actual.

6. Escoja el tipo de instalación (típica o personalizada). La opción más común es la recomendada para la mayoría de los usuarios. Pulsar Next.

7. Aparece la clásica pregunta de dónde instalar el servidor. Como siempre usted puede elegir un directorio diferente o el que aparece por defecto. Haga clic en el botón Next.

8. El proceso de instalación ha finalizado, haga clic en el botón de Finish para terminar.

9. Para probar el éxito de la instalación, abra la dirección <http://localhost>.

Instalación y configuración del Servidor de Aplicaciones (Tomcat)

Antes de proceder con la instalación del contenedor Web Jakarta-Tomcat es necesario la instalación del J2SDK de Java. Nosotros utilizamos Java 2 Standard Edition - J2SE la última al momento de escribir esta referencia es la versión 1.4 de J2SDK de Java.



- 1.** Obtener la última versión del SDK de Java de <http://java.sun.com/> (versión estándar <http://java.sun.com/j2se/1.4/download.html>).
- 2.** Ejecutar el archivo con el SDK de Java, en nuestro caso *j2sdk-1_4_0_01-windows-i586.exe*.
- 3.** Seguir las instrucciones en pantalla.
- 4.** Obtener la última versión estable de Jakarta Tomcat de <http://jakarta.apache.org/site/binindex.html>.
- 5.** Ejecutar el instalador (*jakarta-tomcat-4.0.4.exe*) y seguir instrucciones en pantalla.
- 6.** En la pantalla de opciones de instalación seleccionar "NT Service (NT/2k/XP only)" para instalar a Tomcat como servicio.
- 7.** Para probar el éxito de la instalación, abra la dirección <http://localhost:8080> y si se despliega una página de bienvenida que indica el éxito de la operación.

Configurar Tomcat como extensión de Apache¹²

La instalación de Tomcat en conjunción con otro servidor web resulta mucho más eficiente, de esta manera, Tomcat se encarga del contenido dinámico (páginas JSP y Servlets) mientras que el servidor Web maneja el contenido estático del sitio o aplicación.

¹² FUENTE: Guía de Configuración de Jeff Lundberg – jeff@jefflundberg.com Install & Configure Apache with PHP, JSP and MySQL on Windows XP Pro

Para configurar a Tomcat como extensión del servidor Apache, siga el siguiente procedimiento.



1. Obtener la librería `mod_jk.dll` de http://www.acg-gmbh.de/mod_jk/.

2. Añadir el siguiente código al final del archivo de configuración de Apache `httpd.conf` :

```
#####
## JSP Support
## Copied from Tomcat auto-configuration file
##   + added JkMount /*.jsp ajp13
##   + removed Virtual Host

<IfModule !mod_jk.c>
  LoadModule jk_module c:/WINDOWS/system32/mod_jk.dll
</IfModule>

JkWorkersFile "C:/Program Files/Apache Tomcat 4.0/conf/workers.properties"
JkLogFile "C:/Program Files/Apache Tomcat 4.0/logs/mod_jk.log"
JkLogLevel info
JkMount /manager ajp13
JkMount /manager/* ajp13
JkMount /examples ajp13
JkMount /examples/* ajp13
JkMount /control-app ajp13
JkMount /control-app/* ajp13
JkMount /webdav ajp13
JkMount /webdav/* ajp13
JkMount /*.jsp ajp13
```

Figura 6.2. Código de configuración para archivo `httpd.conf` del servidor Apache
FUENTE: Autor

3. Copiar el código de la figura 6.2 en un nuevo archivo llamado `workers.properties` y guardarlo en la carpeta `conf` de la raíz de Apache Tomcat. Por ejemplo, `C:\Program Files\Apache Tomcat 4.0\conf`.

```
workers.tomcat_home=C:/Program Files/Apache Tomcat 4.0
workers.java_home=C:/j2sdk1.4.0_01
ps=\

worker.list=ajp13, ajp14
worker.ajp13.port=8009
worker.ajp13.host=localhost
worker.ajp13.type=ajp13
worker.ajp13.lbfactor=1

worker.ajp14.port=8010
worker.ajp14.host=localhost
worker.ajp14.type=ajp14
worker.ajp14.secretkey=secret
worker.ajp14.credentials=myveryrandomentropy
worker.ajp14.lbfactor=1

worker.loadbalancer.type=lb
worker.loadbalancer.balanced_workers=ajp13
worker.inprocess.type=jni
worker.inprocess.class_path=$(workers.tomcat_home)$(ps)server$(ps)lib$(ps)catalina.jar
```

```
worker.inprocess.cmd_line=start
worker.inprocess.jvm_lib=$(workers.java_home)$(ps)jre$(ps)bin$(ps)server$(ps)jvm.dll
worker.inprocess.stdout=$(workers.tomcat_home)$(ps)logs$(ps)inprocess.stdout
worker.inprocess.stderr=$(workers.tomcat_home)$(ps)logs$(ps)inprocess.stderr
```

Figura 6.3. Archivo workers.properties para el Apache Tomcat
FUENTE: Autor

4. Incluir las líneas de la figura 6.3 en en archivo de configuración de Tomcat *server.xml* después de la declaración `<Server port="8005" ...>`

```
<Listener className="org.apache.ajp.tomcat4.config.apacheconfig"
  modJk="c:/WINDOWS/system32/mod_jk.dll" jkDebug="info"
  workersConfig="C:/Program Files/Apache Tomcat 4.0/conf/workers.properties"
  jkLog="C:/Program Files/Apache Tomcat 4.0/logs/mod_jk.log"/>
```

Figura 6.4. Configuración de listener para server.xml-Fuente: Autor

5. En el mismo documento *server.xml*, después de la declaración `<Service name="Tomcat-Standalone">` añadir lo que sigue.

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
  <Connector className="org.apache.ajp.tomcat4.Ajp13Connector"
    port="8009" minProcessors="5" maxProcessors="75"
    acceptCount="10" debug="0"/>
```

Figura 6.5. Definición de AJP 1.3 en el puerto 8009-Fuente: Autor

6. Finalmente, luego de cualquier declaración `<Host name="localhost" ...>`, incluir un listener como el que sigue.

```
<Listener className="org.apache.ajp.tomcat4.config.apacheconfig"
  append="true" />
```

Figura 6.6. Inclusión de listener -Fuente: Autor

7. Para comprobar el correcto funcionamiento de la configuración ingrese a `http://localhost/examples/` y si se despliega una lista de links, la instalación se ha realizado con éxito. Caso contrario revise el proceso y los directorios en los cuales se ha instalado el SDK de Java, el servidor Apache y el servidor Tomcat Apache.



Al iniciar los servidores Apache y Tomcat, siempre iniciar el servicio de Tomcat antes de iniciar a Apache. La razón es porque al iniciarse Apache busca el servicio de tomcat, si no lo encuentra no lo llama cuando recibe una petición que debería manejar a través de Tomcat aun cuando este ya se haya iniciado.

Instalación de la base de Datos MySQL y estructura de ShoppingCart

La base de datos constituye la fuente de información del sistema. ShoppingCart es un sistema independiente de la base de datos, esto quiere decir que puede ser instalado sobre cualquier motor de base de datos a partir de sus scripts de creación.

El sistema fue montado sobre la base de datos MySql, la más popular entre aplicaciones web. A continuación explicamos la instalación de MySql y los scripts de datos de la aplicación Control CCC

Instalación MySQL



- 1.** Obtener la última versión estable de MySql desde su sitio ne Internet www.mysql.com . La versión instalada es mysql-312.23.51-win.
- 2.** Descomprimir y ejecutar el programa de instalación setup.exe.
- 3.** Seguir instrucciones en pantalla.
- 4.** Instalar el administrador del servidor de base de datos. Funciona con muchos administradores del mercado. El administrador escogido fue MySQLFront, una herramienta excepcional, eficiente y fácil de usar desarrollada por Ansgar Becker (info@anse.de) <http://www.anse.de/> con licencia Freeware. Se la puede obtener de <http://www.anse.de/> o de www.mysqlfront.de.
- 5.** Ejecutar el instalador y seguir las instrucciones mostradas en pantalla.

Instalación Base de datos ShoppingCart



1. Iniciar una sesión de MySQL con usuario root¹³.
2. Crear un usuario para que sea el dueño de la base de control
3. Ingresar a la base de datos con el usuario creado
4. Crear una nueva Base de datos y asignarla un nombre, por ejemplo ccc.
5. Ejecutar el archivo data.sql que se encuentra en dentro del directorio /database/scripts en el CD del sistema.

Instalación de ShoppingCart

Una vez configurada la infraestructura básica, esto es el servidor de base de datos, el Web container para entorno de producción, Web Server y la conexión entre los dos últimos, el siguiente paso constituye la configuración y puesta en producción del Sistema de ShoppingCart.

El proceso se divide en dos pasos, el primero es la **creación de la Base de datos** y el segundo la **instalación de la aplicación JSP**.

Para la creación de la base de datos:

- Utilizando cualquier software de administración de MySQL crear una base de datos con el nombre deseado, por ejemplo **shopping**.
- Crear un usuario con todos los privilegios en la base de datos recién creada.
- Ejecutar el script "**database.sql**" (crea la estructura de la base de datos), una vez que este se ejecute adecuadamente, ejecutar el script "**data.sql**"; este último

inserta datos básicos para poder iniciar a trabajar con el sistema.

Una vez creada la base de datos, el siguiente paso es el de instalar la aplicación JSP, el proceso es simple y básicamente consiste en copiar los archivos del sistema en el directorio de aplicaciones del Web container. Adicionalmente, es necesario configurar La conexión del sistema con el servidor de base de datos.

A continuación se describe la instalación del Sistema de ShoppingCart en el Web Container **Apache-Tomcat**.



1. Copiar todo la carpeta */shopping/* en el directorio */webapps* ubicado en la raíz de instalación de **Tomcat**.

2. Modificar el archivo **hibernate.cfg.xml**, localizado en */shopping/WEB-INF/* del sistema de Auditoria laboral para que se conecte con la base de datos, el usuario y la contraseña creada previamente cuando instalo la base de datos del sistema.

```
        <!-- properties -->
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost/shopping</property>
          <property name="connection.username"> shopping </property>
          <property name="connection.password"> shopping </property>
        <property name="dialect">net.sf.hibernate.dialect.MySQLDialect</property>
          <property name="show_sql">>false</property>
```

Figura 1. Archivo connPooling.properties (pool a la BD)

3. En el archivo hibernate.cfg.xml ubicar la línea de conexión o "connection.url" (`jdbc:mysql://localhost/labor`) y reemplazar localhost por el nombre o dirección IP de la computadora donde instalo la base de datos (si solo si la base de datos esta instalada en una computadora distinta a la computadora donde instalo el Web container TOMCAT).

¹³ por defecto el password del usuario "root" no se encuentra establecido. Se aconseja escoger uno por razones de seguridad.

Reemplazar **"shopping"** por el nombre de la base de datos que creo en al configurar la base de datos del sistema.

4. Localizar las líneas donde se define el nombre de usuario "connection.username" y contraseña "connection.password" (`shopping` y `shopping`) y reemplazar por el nombre de usuario y contraseña con las que creo la base de datos del sistema. Guarde los cambios al archivo **hibernate.cfg.xml**.

5. Reiniciar el servidor **Tomcat** y la aplicación será reconocida automáticamente.

6. Para probar que el sistema este funcionando, en la computadora donde instalo el Web container abra Internet Explorer e ingrese a **http://localhost:8080/ shopping /** y la pantalla de bienvenida del sistema se presentara como sigue.



Figura 1. Pantalla de Bienvenida del Sistema

7. Finalmente para ingresar al sistema utilice el usuario **admin** y la contraseña **master**. Es muy importante que una vez que ingrese al sistema por primera vez se dirija a la sección de usuarios y cambie las contraseñas para los usuarios admin. Y user, que son creados al momento de instalar el sistema. Para mayor información de cómo cambiar la contraseña de un usuario, por favor revise el manual de usuario del sistema.

3. Configuración

A continuación se explica otros parámetros de conexión del sistema para su puesta en ejecución. Todos estos parámetros por defecto vienen inicializados con valores adecuados para la operación del sistema. Todos estos parámetros se encuentran ubicados en archivos **.xml**, localizado en / *shopping* /**WEB-INF**/ del sistema de ShoppingCart. En la siguiente tabla mostramos el nombre del parámetro, la propiedad en el archivo **xml** de configuración, el nombre del archivo y el valor por defecto.

Parámetro	Archivo	Propiedad en archivo	v. defecto
duración de la sesión	Web.xml	session-timeout	60
Max. numero conexiones a BD activas en Pool	hibernate.cfg.xml	dbcp.maxActive	10
Tiempo Max. de espera	hibernate.cfg.xml	dbcp.maxWait	10000
Tiempo Max. En Idle	hibernate.cfg.xml	dbcp.maxIdle	5

Anexo C

Diccionario de Datos

Dominios

Domains

Domain list

Name	Code	Data Type
SHORT_CODE	SHORT_CODE	N4
PHONE	PHONE	VA15
LONG_STRING	LONG_STRING	VA255
SHORT_STRING	SHORT_STRING	VA30
LONG_CODE	LONG_CODE	N8

Use list of domain LONG_STRING

Name	Code	Used By
ROL_DESCRIPTION	rol_description	rol
USER_ADDRESS	user_address	user
USER_EMAIL	user_email	user
ORDER_SHIPPING_ADDRESS	order_shipping_address	order
ORDER_CC_ADDRESS	order_cc_address	order
DETAIL_DESCRIPTION	detail_description	detail
PRODUCT_DESCRIPTION	product_description	product
PRODUCT_IMAGE1	product_image1	product
PRODUCT_IMAGE2	product_image2	product
USER_CARD_ADDRESS	user_card_address	user_card
promotion_description	promotion_description	promotion

Dependencies of domain LONG_STRING

Name	Code	Class Name
ROL_DESCRIPTION	rol_description	Data Item
USER_ADDRESS	user_address	Data Item
USER_EMAIL	user_email	Data Item
ORDER_SHIPPING_ADDRESSES	order_shipping_address	Data Item

ORDER_CC_ADDRESS	order_cc_address	Data Item
DETAIL_DESCRIPTION	detail_description	Data Item
PRODUCT_DESCRIPTION	product_description	Data Item
PRODUCT_IMAGE1	product_image1	Data Item
PRODUCT_IMAGE2	product_image2	Data Item
USER_CARD_ADDRESS	user_card_address	Data Item
promotion_description	promotion_description	Data Item

Use list of domain PHONE

Name	Code	Used By
USER_PHONE1	user_phone1	user
USER_PHONE2	user_phone2	user
USER_FAX	user_fax	user

Dependencies of domain PHONE

Name	Code	Class Name
USER_PHONE1	user_phone1	Data Item
USER_PHONE2	user_phone2	Data Item
USER_FAX	user_fax	Data Item

Use list of domain SHORT_CODE

Name	Code	Used By
ROL_CODE	rol_code	rol
USER_CODE	user_code	user
ORDER_CODE	order_code	order
STATUS_CODE	status_code	status
DETAIL_CODE	detail_code	detail
CATEGORY_CODE	category_code	category
PRODUCT_CODE	product_code	product
PRODUCT_PRICE	product_price	product
CREDIT_CARD_CODE	credit_card_code	credit_card
USER_CARD_CODE	user_card_code	user_card
COUNTRY_CODE	country_code	country
PROVINCE_CODE	province_code	province
city_code	city_code	city
promotion_code	promotion_code	promotion

Dependencies of domain SHORT_CODE

Name	Code	Class Name
ROL_CODE	rol_code	Data Item
USER_CODE	user_code	Data Item
ORDER_CODE	order_code	Data Item
STATUS_CODE	status_code	Data Item
DETAIL_CODE	detail_code	Data Item
CATEGORY_CODE	category_code	Data Item
PRODUCT_CODE	product_code	Data Item
PRODUCT_PRICE	product_price	Data Item
CREDIT_CARD_CODE	credit_card_code	Data Item
USER_CARD_CODE	user_card_code	Data Item
COUNTRY_CODE	country_code	Data Item
PROVINCE_CODE	province_code	Data Item
city_code	city_code	Data Item
promotion_code	promotion_code	Data Item

Use list of domain SHORT_STRING

Name	Code	Used By
ROL_NAME	rol_name	rol
USER_USERNAME	user_username	user
USER_FNAME	user_fname	user
USER_LNAME	user_lname	user
USER_PASSWORD	user_password	user
ORDER_SHIPPING_CITY	order_shipping_city	order
ORDER_CC_OWNER	order_cc_owner	order
ORDER_CC_BANK	order_cc_bank	order
STATUS_NAME	status_name	status
CATEGORY_NAME	category_name	category
PRODUCT_NAME	product_name	product
CREDIT_CARD_NAME	credit_card_name	credit_card
USER_CARD_OWNER	user_card_owner	user_card
USER_CARD_BANK	user_card_bank	user_card
COUNTRY_NAME	country_name	country
PROVINCE_NAME	province_name	province
city_name	city_name	city
promotion_name	promotion_name	promotion
name	name	sequence

Dependencies of domain SHORT_STRING

Name	Code	Class Name
ROL_NAME	rol_name	Data Item
USER_USERNAME	user_username	Data Item
USER_FNAME	user_fname	Data Item
USER_LNAME	user_lname	Data Item
USER_PASSWORD	user_password	Data Item
ORDER_SHIPPING_CITY	order_shipping_city	Data Item
ORDER_CC_OWNER	order_cc_owner	Data Item
ORDER_CC_BANK	order_cc_bank	Data Item
STATUS_NAME	status_name	Data Item
CATEGORY_NAME	category_name	Data Item
PRODUCT_NAME	product_name	Data Item
CREDIT_CARD_NAME	credit_card_name	Data Item
USER_CARD_OWNER	user_card_owner	Data Item
USER_CARD_BANK	user_card_bank	Data Item
COUNTRY_NAME	country_name	Data Item
PROVINCE_NAME	province_name	Data Item
city_name	city_name	Data Item
promotion_name	promotion_name	Data Item
name	name	Data Item

Entity list

Name	Code	Parent	Generate	Number
product	PRODUCT	Conceptual Data Model 'ConceptualDataModel_1'	TRUE	
user	USER	Conceptual Data Model 'ConceptualDataModel_1'	TRUE	
rol	ROL	Conceptual Data Model 'ConceptualDataModel_1'	TRUE	
category	CATEGORY	Conceptual Data Model 'ConceptualDataModel_1'	TRUE	

		el_1'		
order	ORDER	Conceptual Data Model 'ConceptualDataModel_1'	TRUE	
status	STATUS	Conceptual Data Model 'ConceptualDataModel_1'	TRUE	
detail	DETAIL	Conceptual Data Model 'ConceptualDataModel_1'	TRUE	
sequence	SEQUENCE	Conceptual Data Model 'ConceptualDataModel_1'	TRUE	
credit_card	CREDIT_CARD	Conceptual Data Model 'ConceptualDataModel_1'	TRUE	
user_card	USER_CARD	Conceptual Data Model 'ConceptualDataModel_1'	TRUE	
province	PROVINCE	Conceptual Data Model 'ConceptualDataModel_1'	TRUE	
country	COUNTRY	Conceptual Data Model 'ConceptualDataModel_1'	TRUE	
city	CITY	Conceptual Data Model 'ConceptualDataModel_1'	TRUE	
promotion	PROMOTION	Conceptual Data Model 'ConceptualDataModel_1'	TRUE	
promo_product	PROMO_PRODUCT	Conceptual Data Model 'ConceptualDataModel_1'	TRUE	

Identifier list of entity

Name	Code	Parent
Identifier_1	IDENTIFIER_1	Entity 'product'
Identifier_1	IDENTIFIER_1	Entity 'user'
username_ak	USERNAME_AK	Entity 'user'
Identifier_1	IDENTIFIER_1	Entity 'rol'
Identifier_1	IDENTIFIER_1	Entity 'category'
Identifier_1	IDENTIFIER_1	Entity 'order'
Identifier_1	IDENTIFIER_1	Entity 'status'
Identifier_1	IDENTIFIER_1	Entity 'detail'
Identifier_1	IDENTIFIER_1	Entity 'sequence'
Identifier_1	IDENTIFIER_1	Entity 'credit_card'
Identifier_1	IDENTIFIER_1	Entity 'user_card'
Identifier_1	IDENTIFIER_1	Entity 'province'
Identifier_1	IDENTIFIER_1	Entity 'country'
Identifier_1	IDENTIFIER_1	Entity 'city'
Identifier_1	IDENTIFIER_1	Entity 'promotion'

Data Items

Model level data items

Data item list

Name	Code	Domain	Data Type	Length	Precision
USER_EMAIL	user_email	LONG_STRING	VA255	255	
USER_PASSWORD	user_password	SHORT_STRING	VA30	30	
USER_USERNAME	user_username	SHORT_STRING	VA30	30	
USER_CODE	user_code	SHORT_CODE	N4	4	
ROL_DESCRIPTION	rol_description	LONG_STRING	VA255	255	
ROL_NAME	rol_name	SHORT_STRING	VA30	30	
ROL_CODE	rol_code	SHORT_CODE	N4	4	
USER_FAX	user_fax	PHONE	VA15	15	
USER_PHONE2	user_phone2	PHONE	VA15	15	
USER_PHONE1	user_phone1	PHONE	VA15	15	

USER_ADDRESS	user_address	LONG_STRING	VA255	255	
USER_LASTNAME	user_lastname	SHORT_STRING	VA30	30	
USER_FIRSTNAME	user_firstname	SHORT_STRING	VA30	30	
PRODUCT_CODE	product_code	SHORT_CODE	N4	4	
PRODUCT_NAME	product_name	SHORT_STRING	VA30	30	
PRODUCT_DESCRIPTION	product_description	LONG_STRING	VA255	255	
PRODUCT_PRICE	product_price	SHORT_CODE	N4	4	
CATEGORY_CODE	category_code	SHORT_CODE	N4	4	
CATEGORY_NAME	category_name	SHORT_STRING	VA30	30	
ORDER_CODE	order_code	SHORT_CODE	N4	4	
ORDER_DATE	order_date	<None>	DT		
DETAIL_CODE	detail_code	SHORT_CODE	N4	4	
DETAIL_QUANTITY	detail_quantity	LONG_CODE	N8	8	
DETAIL_DESCRIPTION	detail_description	LONG_STRING	VA255	255	
STATUS_CODE	status_code	SHORT_CODE	N4	4	
STATUS_NAME	status_name	SHORT_STRING	VA30	30	
name	name	SHORT_STRING	VA30	30	
next_val	next_val	LONG_CODE	N8	8	
USER_ENABLED	user_enabled	<None>	BL		
ORDER_AMOUNT	order_amount	<None>	N12,5	12	5
ORDER_T	order_tax	<None>	N12,5	12	5

AX					
CATEGORY_TAX	category_tax	<None>	N5,4	5	4
CREDIT_CARD_CODE	credit_card_code	SHORT_CODE	N4	4	
CREDIT_CARD_NAME	credit_card_name	SHORT_STRING	VA30	30	
USER_CARD_CODE	user_card_code	SHORT_CODE	N4	4	
USER_CARD_NUMBER	user_card_number	<None>	VA20	20	
USER_CARD_SECURITY	user_card_security	<None>	VA5	5	
USER_CARD_OWNER	user_card_owner	SHORT_STRING	VA30	30	
USER_CARD_EXPIRATION	user_card_expiration	<None>	D		
USER_CARD_ADDRESS	user_card_address	LONG_STRING	VA255	255	
USER_CARD_BANK	user_card_bank	SHORT_STRING	VA30	30	
ORDER_CC_NUMBER	order_cc_number	<None>	N14	14	
ORDER_CC_SECURITY	order_cc_security	<None>	N3	3	
ORDER_CC_OWNER	order_cc_owner	SHORT_STRING	VA30	30	
ORDER_CC_BANK	order_cc_bank	SHORT_STRING	VA30	30	
ORDER_CC_ADDRESS	order_cc_address	LONG_STRING	VA255	255	
ORDER_CC_EXPIRATION	order_cc_expiration	<None>	D		

ORDER_SHIPPING_ADDRESS	order_shipping_address	LONG_STRING	VA255	255	
CATEGORY_ENABLED	category_enabled	<None>	BL		
ORDER_SHIPPING_CITY	order_shipping_city	SHORT_STRING	VA30	30	
COUNTRY_CODE	country_code	SHORT_CODE	N4	4	
COUNTRY_NAME	country_name	SHORT_STRING	VA30	30	
PROVINCE_CODE	province_code	SHORT_CODE	N4	4	
PROVINCE_NAME	province_name	SHORT_STRING	VA30	30	
city_name	city_name	SHORT_STRING	VA30	30	
city_code	city_code	SHORT_CODE	N4	4	
promotion_code	promotion_code	SHORT_CODE	N4	4	
promotion_name	promotion_name	SHORT_STRING	VA30	30	
promotion_description	promotion_description	LONG_STRING	VA255	255	
promotion_from	promotion_from	<None>	DT		
promotion_to	promotion_to	<None>	DT		
promotion_enabled	promotion_enabled	<None>	BL		
promotion_rate	promotion_rate	<None>	N12,5	12	5
PRODUCT_STOCK	product_stock	<None>	I		
PRODUCT_ENABLED	product_enabled	<None>	BL		
PRODUCT_IMAGE1	product_image1	LONG_STRING	VA255	255	
PRODUCT_IMAGE2	product_image2	LONG_STRING	VA255	255	
USER_CARD	user_card	<None>	BL		

RD_ENAB LED	enabled				
----------------	---------	--	--	--	--

Relationship
Model level relationships
Relationship list

Name	Code	Parent	Entity 2	Entity 1
Relationship_1	RELATIONS HIP_1	Conceptual Data Model 'ConceptualD ataModel_1'	user	rol
Relationship_2	RELATIONS HIP_2	Conceptual Data Model 'ConceptualD ataModel_1'	order	user
Relationship_3	RELATIONS HIP_3	Conceptual Data Model 'ConceptualD ataModel_1'	order	status
Relationship_4	RELATIONS HIP_4	Conceptual Data Model 'ConceptualD ataModel_1'	detail	order
Relationship_6	RELATIONS HIP_6	Conceptual Data Model 'ConceptualD ataModel_1'	product	category
Relationship_6	RELATIONS HIP_7	Conceptual Data Model 'ConceptualD ataModel_1'	product	detail
Relationship_7	RELATIONS HIP_8	Conceptual Data Model 'ConceptualD ataModel_1'	category	category
Relationship_8	RELATIONS HIP_9	Conceptual Data Model 'ConceptualD ataModel_1'	user_card	credit_card
Relationship_9	RELATIONS HIP_10	Conceptual Data Model	user_card	user

		'ConceptualD ataModel_1'		
Relationship_ 10	RELATIONS HIP_11	Conceptual Data Model 'ConceptualD ataModel_1'	province	country
Relationship_ 12	RELATIONS HIP_13	Conceptual Data Model 'ConceptualD ataModel_1'	user	city
Relationship_ 12	RELATIONS HIP_12	Conceptual Data Model 'ConceptualD ataModel_1'	city	province
Relationship_ 13	RELATIONS HIP_14	Conceptual Data Model 'ConceptualD ataModel_1'	promo_produ ct	product
Relationship_ 14	RELATIONS HIP_15	Conceptual Data Model 'ConceptualD ataModel_1'	promo_produ ct	promotion

Anexo D

Glosario Términos

Artefactos: Es cualquier tipo de información creada , producida, cambiada o utilizada por los trabajadores en el desarrollo del sistema.

Clase: Referente a un conjunto de objetos que poseen similares características y funciones.

Cliente : Dispositivo (o aplicación) que inicia una petición de conexión con un servidor.

Connection Pooling : Object Pooling (concentración de objetos) es una técnica que cada vez toma mayor importancia a la hora mejorar el rendimiento de servidores de aplicaciones. Una de las operaciones más costosa que existe en términos de utilización de recursos es la creación de objetos de conexión a base de datos. Por tanto, el Object Pooling más utilizado es el conocido como Connection Pooling (concentración de conexiones).

Gateway WAP : Software capaz de conectarse a la red de telefonía móvil y a Internet. Esta doble conexión permite utilizar todos los servicios que se encuentran en la red.

Hipertexto : Es texto que está codificado usando un sistema estándar llamado: Hypertext Markup Language ([HTML](#)). Este código es usado para crear links, estos links pueden ser textos o gráficos, y al hacer clic sobre un link el usuario se conecta a otro documento HTML disponible, (archivo de textos, gráficos o sonidos). Cuando un usuario selecciona un link de hipertexto, el programa cliente en su computadora usa HTTP para contactarse con el servidor, identificar el recurso y preguntarle que acción tomar. Este último acepta el requerimiento, y también usando HTTP responde o ejecuta la acción correspondiente.

Hosting : Es el servicio de hospedaje de páginas Web en servidores que permiten el acceso a las mismas en cualquier momento.

HTML (Hypertext Markup Language) : Consiste en un conjunto de códigos standards o "tags" (marcas), que se utilizan para definir la estructura de la información de una Página Web. HTML define los distintos aspectos de una página, tipo de letra, quiebres de párrafos, hiperlinks, etc.

HTML es un lenguaje derivado de SGML (Standard Generalized Markup Language).

HTTP (HyperText Transfer Protocol) : Es un conjunto de reglas o [protocolos](#) usado para manejar la transferencia de páginas de hipertexto en el WWW, está basado en el principio [cliente/servidor](#).

Java Server Pages (JSP) : Es un conjunto de tecnologías que permiten la generación dinámica de paginas web combinando código Java (scriptlets) con un lenguaje de marcas como HTML ó XML, para generar el contenido de un página web dinámica.

JNDI (Java Naming and Directory Interfaces) : Interfaces Java diseñadas para facilitar acceso a base de datos de directorios.

Object Pooling (Concentración de Objetos) : Técnica para administrar y reutilizar conjuntos de objetos en tiempo de ejecución con el fin de optimizar los recursos de una aplicación (memoria, tiempo de procesamiento) y evitar sobrecargas debido a un elevado número de peticiones por parte de los usuarios. Consiste en contener en memoria un conjunto de objetos de determinado tipo y ofrecerlos cuando alguien lo requiera en lugar de tener que crearlos cada vez que se necesiten. Una vez utilizado el objeto regresa a la concentración a esperar por otra petición.

Objetos Pesados : Objetos que utilizan una considerable cantidad de recursos (memoria, conexiones de red, recursos de base de datos, entre otros).

Portador (bearer) : El portador o bearer es un protocolo que se encarga de transmitir los datos (mensajes, paquetes o frames) desde nuestro teléfono móvil a nuestra operadora de telefonía. WAP es independiente del portador de la información. De ello se encarga el WDP (Wireless Datagram Protocol), el cual adapta el transporte de información a cada una de las diferentes formas posibles.

Ejemplos de portadores son: Advanced Mobile Phone System (AMPS), Cellular Digital Packet Data (CDPD), Global System for Mobile Communications (GSM).

Protocolo : Acuerdo que dirige los procedimientos utilizados para el intercambio de información entre entidades que colaboran. Por regla general, este acuerdo comprende la cantidad de información que se ha de enviar, la frecuencia con la que se envía, la forma de recuperación de los errores de transmisión y quién va a recibir la información. En general, la definición de un protocolo debe incluir las definiciones de los formatos de mensajes, las reglas de las secuencias concernientes a éstos, y las reglas de interpretación referentes a los mensajes transferidos en la propia secuencia.

Proxy : El Servidor Proxy se utiliza para almacenar la información más frecuentemente consultada a páginas de Internet por un período de tiempo, con el fin de aumentar la velocidad de acceso. Al mismo tiempo libera la carga de los enlaces hacia Internet.

Servidor : Dispositivo (o aplicación) que aguarda por peticiones de conexión originadas por uno o más cliente. Un servidor puede aceptar o rechazar una petición hecha por un cliente.

Servidor http : Aplicación encargada de recibir peticiones enviadas de clientes, procesarlas y responderlas a dichos clientes utilizando el protocolo http.

SGML (Standard Generalized Markup Language). SGML es un sistema que define y da un formato standard a la estructura de los documentos.

Servlet : Son componentes del servidor. Estos componentes pueden ser ejecutados en cualquier plataforma o en cualquier servidor debido a la tecnología Java que se usa para implementarlos. Los Servlets incrementan la funcionalidad de una aplicación web. Se cargan de forma dinámica por el entorno de ejecución Java del servidor cuando se necesitan. Cuando se recibe una petición del cliente, el contenedor/servidor web inicia el Servlet requerido. El Servlet procesa la petición del cliente y envía la respuesta de vuelta al contenedor/servidor, que la envía al cliente.

Trabajador: Se denomina trabajador a los puestos a los cuales se puede asignar personal y cuando hablemos de roles se refiere a los papeles que cumple un trabajador.

URL (Uniform Resource Locator) : Es una cadena que indica la dirección Internet de un sitio Web o de un recurso en WWW, es la manera de acceder a una dirección en Internet y depende del tipo de recurso que se está buscando.

W3C (Consortio para la World Wide Web) : Fue fundado en octubre de 1994, para conducir a la **World Wide Web** a su máximo potencial desarrollando protocolos de uso común que promocionaran su evolución y aseguraran su interoperabilidad.

World Wide Web (WWW o Web) : Está constituida por documentos llamados Páginas Web que combinan textos, gráficos, sonido, animación, hipertexto y links llamados hiperlinks. Para navegar en la Web los usuarios surfean de una página a otra haciendo clic en los hiperlinks en textos o gráficos.

XML (eXtensible Markup Language - Lenguaje extensible de marcas) : Es un lenguaje abierto, derivado de SGML, optimizado para su uso en la WWW, y que permite describir el sentido o la

semántica de los datos. XML es un Meta-Lenguaje, que permite la definición de lenguajes concretos de representación de documentos